PROPT - Matlab Optimal Control Software

- ONE OF A KIND, LIGHTNING FAST SOLUTIONS TO YOUR OPTIMAL CONTROL PROBLEMS!

Per E. Rutquist[1] and Marcus M. Edvall[2]

April 26, 2010

[1]Tomlab Optimization AB, Västerås Technology Park, Trefasgatan 4, SE-721 30 Västerås, Sweden.
[2]Tomlab Optimization Inc., 1260 SE Bishop Blvd Ste E, Pullman, WA, USA.

Contents

1 PROPT Guide Overview

The following sections describe the functionality of PROPT. It is recommended to get familiar with a few examples (included with software) to get accustomed with the notations used.

Please visit PROPT on the web for detailed problem categories, presentations and more: http://tomdyn.com/.

1.1 Installation

The PROPT software is included in the general TOMLAB installation executable. Please refer to the TOMLAB Installation manual for further information. The following Matlab commands can be used to verify a successful installation.

```
>> cd c:\tomlab\
>> startup
>> vanDerPol
```

After a few seconds two graphs should display the state and control variables for the example.

1.2 Foreword to the software

The software is provided without limitations to all registered customers in order to demonstrated what PROPT is capable of. The implementation of the extensive example library (more then 100 test cases are included) has eliminated most bugs, but there may still be a few that has escaped attention.

If problems are encountered please email a copy of the problem for debugging.[1] A status report can be expected within 24 hours.

1.3 Initial remarks

PROPT is a combined modeling, compilation and solver engine for generation of highly complex optimal control problems. The users will need to have licenses for the TOMLAB Base Module, TOMLAB /SNOPT (or other suitable nonlinear TOMLAB solver) (always included with a demo license).

It is highly recommended to take a look at the example collection included with the software before building a new problem (Tip: Copy one of the examples and use as a starting-point).

The brysonDenham problems in the examples folder contain very extensive documentation, explanations and general information about the usage and software design.

PROPT aims to encompass all areas of optimal control including:

- Aerodynamic trajectory control
- Bang bang control
- Chemical engineering
- Dynamic systems

[1] All examples and data are treated as strictly confidential unless other instructions are given.

- General optimal control
- Large-scale linear control
- Multi-phase system control
- Mechanical engineering design
- Non-differentiable control
- Parameters estimation for dynamic systems
- Singular control

At least one example from each area is included in the example suite.

2 Introduction to PROPT

PROPT is a software package intended to solve dynamic optimization problems. Such problems are usually described by

- A state-space model of a system. This can be either a set of ordinary differential equations (**ODE**) or differential algebraic equations (**DAE**).
- Initial and/or final conditions (sometimes also conditions at other points).
- A cost functional, i.e. a scalar value that depends on the state trajectories and the control function.
- Sometimes, additional equations and variables that, for example, relate the initial and final conditions to each other.

The goal of PROPT is to enable the formulation of such problem descriptions as seamlessly as possible, without having to worry about the mathematics of the actual solver. Once a problem has been properly defined, PROPT will take care of all the necessary steps in order to return a solution.[2]

PROPT uses pseudospectral collocation methods (and other options) for solving optimal control problems. This means that the solution takes the form of a polynomial, and this polynomial satisfies the DAE and the path constraints at the collocation points (Note that both the DAE and the path constraints can be violated between collocation points). The default choice is to use Gauss points as collocation points, although the user can specify any set of points to use.

It should be noted that the code is written in a general way, allowing for a DAE rather than just an ODE formulation with path constraints.

Parameter estimation for dynamic systems is intrinsically supported by the framework as scalar decision variables can be introduced in the formulation.

2.1 Overview of PROPT syntax

A PROPT problem is defined with tomSym objects and standard Matlab expressions (usually in cell arrays), which contain information about different aspects of the problem.

In general an optimal control consists of the following:

- **Phases:** A problem may contain several different phases, each with its own set of variables and equations.
- **Independent:** Variables that are independent (normally time).
- **States:** (Could also be called "dependent") States (short for *state variables*) are continuous functions of the independent variable. Control variables are (possibly) discontinuous functions of the same variable.
- **Controls:** Control variables (states that may be discontinuous).
- **Scalars:** A problem may also contain unknown scalar variables that are solved for at the same time as the controls and state trajectories.
- **Parameters:** Numeric constants that are used to define a problem.

[2]This is accomplished by translating the dynamic problem into a nonlinear programming problem, generating Matlab function files and a TOMLAB problem, that can be passed to a numeric solver.

- **Expressions:** Intermediate results in computations.
- **Equations and inequalities:** The core of the problem definition.
- **Cost:** A performance index or objective function to be minimized.

PROPT has many built-in features:

- Computation of the constant matrices used for the differentiation and integration of the polynomials used to approximate the solution to the trajectory optimization problem.
- Code manipulation to turn user-supplied expressions into Matlab code for the cost function f and constraint function c (with two levels of analytical derivative and sparsity patterns). The files are passed to a nonlinear programming solver (by default) in TOMLAB for final processing.
- Functionality for plotting and computing a variety of information for the solution to the problem.
- Integrated support for non-smooth (hybrid) optimal control problems.

Note: The tomSym and PROPT softwares are the first Matlab modules to enable complete source transformation for optimal control problem.

2.2 Vector representation

In PROPT, each state is represented by a scalar x_0 and a vector $\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_N \end{bmatrix}$, such that x_i corresponds to the value of the state at the i:th collocation point.

State equations are written on vector form, and hence apply at all collocation points. The initial point is treated separately, so the state equations do not apply at the initial point.

The final state is not a free variable. Instead, it is computed via the interpolating polynomial.[3]

2.3 Global optimality

PROPT does not use Pontryagin's maximum principle, rather it uses a pseudospectral method, however the results are mathematically equivalent. This means that if the solver prints "an optimal solution was found", the solution satisfies necessary, but not sufficient, conditions of optimality. It is guaranteed that the returned solution cannot be improved by an infinitesimal change in the trajectory, but there may exist completely different trajectories that are better.

A recommended procedure for finding other solutions is to set conservative bounds for all variables (states, controls and parameters) and change the solver to "multiMin". This will run a large number of optimizations from random starting points. If they all converge to the same minimum, this is an indication, but no guarantee, that the solution is indeed the global optimum.

In order guarantee that the global optimum is obtained, one must either solve the Hamilton-Jacobi-Bellman equation (which PROPT does not) or show that the problem is convex and therefore only has one optimum (which

[3]The final state could be computed by integrating the ODE, however, as we use a DAE, it is more convenient to go via the interpolating polynomial. Proofs that these methods are mathematically equivalent have been published.

may not be the case). If ezsolve claims that the problem type is "LP" or "QP", the problem is convex, and the solution is a global optimum.

It is also worth mentioning that a solution computed by PROPT only satisfies the ODE and constraints in the specified collocation points. There is no guarantee that the solution holds between those points. A common way of testing the integrity of a solution is to re-run the computation using twice as many collocation points. If nothing changes, then there was probably enough points in the first computation.

3 Modeling optimal control problems

In order to understand the basic modeling features in PROPT it is best to start with a simple example. Open the file called *brysonDenham.m* located in `/tomlab/propt/examples`.

There are also *brysonDenhamShort.m*, *brysonDenhamDetailed.m*, *brysonDenhamTwoPhase.m* that solve the same problem, utilizing different features of PROPT.

To solve the problem, simply enter the following in the Matlab command prompt:

```
>> brysonDenham
```

3.1 A simple example

The following example can be found in *vanDerPol.m* in `/tomlab/propt/examples`.

The objective is to solve the following problem:

minimize:

$$J = x_3(t_f)$$

subject to:

$$\begin{aligned}
\frac{dx_1}{dt} &= (1 - x_2^2) * x_1 - x_2 + u \\
\frac{dx_2}{dt} &= x_1 \\
\frac{dx_3}{dt} &= x_1^2 + x_2^2 + u^2
\end{aligned}$$

$$\begin{aligned}
-0.3 <= & \; u <= 1.0 \\
x(t_0) &= [0\ 1\ 0]' \\
t_f &= 5
\end{aligned}$$

To solve the problem with PROPT the following compact code can be used:

```
toms t
p = tomPhase('p', t, 0, 5, 60);
setPhase(p);

tomStates x1 x2 x3
tomControls u

% Initial guess
x0 = {icollocate({x1 == 0; x2 == 1; x3 == 0})
    collocate(u == -0.01)};

% Box constraints
```

```
cbox = {-10  <= icollocate(x1) <= 10
     -10  <= icollocate(x2) <= 10
     -10  <= icollocate(x3) <= 10
     -0.3 <= collocate(u)   <= 1};

% Boundary constraints
cbnd = initial({x1 == 0; x2 == 1; x3 == 0});

% ODEs and path constraints
ceq = collocate({dot(x1) == (1-x2.^2).*x1-x2+u
    dot(x2) == x1; dot(x3) == x1.^2+x2.^2+u.^2});

% Objective
objective = final(x3);

% Solve the problem
options = struct;
options.name = 'Van Der Pol';
solution = ezsolve(objective, {cbox, cbnd, ceq}, x0, options);
```

3.2 Code generation

It is possible to compile permanently, in order to keep the autogenerated code:

```
>> Prob = sym2prob('lpcon',objective, {cbox, cbnd, ceq}, x0, options);
>> Prob.FUNCS

ans =

      f: 'lp_f'
      g: 'lp_g'
      H: 'lp_H'
      c: 'c_AFBHVT'
     dc: 'dc_AFBHVT'
    d2c: 'd2c_AFBHVT'
      r: []
      J: []
    d2r: []
     fc: []
    gdc: []
     fg: []
    cdc: []
     rJ: []

>> edit c_AFBHVT
```

The code that was generated can be found in the $TEMP directory. The objective in this case is linear and can found in the Prob structure (*Prob.QP.c*).

Here is what the auto generated constraint code may look like:

```
function out = c_AFBHVT(tempX,Prob)
% c_AFBHVT - Autogenerated file.
tempD=Prob.tomSym.c;
x3_p = reshape(tempX(183:243),61,1);
u_p = reshape(tempX(1:60),60,1);
x2_p = reshape(tempX(122:182),61,1);
x1_p = reshape(tempX(61:121),61,1);
tempC7   = tempD{2}*x2_p;
tempC8   = tempC7.^2;
tempC10  = tempD{2}*x1_p;
out      = [(tempD{3}-tempC8).*tempC10-tempC7+u_p-0.2*(tempD{1}*x1_p);...
           tempC10.^2+tempC8+u_p.^2-0.2*(tempD{1}*x3_p)];
```

And the objective function to optimize (in this case a simple linear objective already available in TOMLAB (hence not auto generated, but defined by the *Prob* field used)):

```
function f = lp_f(x, Prob)
f = Prob.QP.c'*x(:);
```

3.3 Modeling

The PROPT system uses equations and expressions (collected in cells) to model optimal control problems.

Equations must be written either using (== <= >=) equality/inequality markers.

It is possible to include more than one equation on the same line.

For example:

```
toms a b c
cnbd = {a == b; b == c};
```

does the same job as:

```
toms a b c
cnbd = {a == b
 b == c};
```

The same is true for inequalities.

When working with optimal control problems, one typically work with expressions that are valid for all collocation points. The functions *collocate* and *icollocate* can be used to extend an arbitrary expressions to the necessary collocation points.

Consider for example the starting points:

```
tomStates x1
tomControls u1
x0 = {icollocate(x1==1); collocate(u1==1)};
```

Note: icollocate, as it is working with a state variable, also includes the starting point.

Scale the problem:

Proper scaling may speed up convergence, and conversely, improperly scaled problems may converge slowly or not at all. Both unknowns and equations can be scaled.

Don't use inf in equations:

It is strongly discouraged to use -inf/inf in equations. This is because the equation may be evaluated by subtracting its left-hand side from the right-hand side and comparing the result to zero, which could have unexpected consequences.

Equations like $x <= Inf$ are better left out completely (Although in many cases tomSym will know to ignore them anyway).

Avoid using non-smooth functions:

Because the optimization routines rely on derivatives, non-smooth functions should be avoided. A common example of a non-smooth function is the Heaviside function H, defined as $H(x) = 1$ for $x > 0$, and $H(x) = 0$ for $x \leq 0$, which in Matlab code can be written as $(x > 0)$. A smoother expression, which has the same behavior for $x >> a$ is:

$H_s = \frac{1}{2}\left(1 + \tanh(x/a)\right)$

When the problems are non-smooth in the independent variable (time, t), the problem should normally be separated into phases.

Use descriptive names for equations, states, controls, and variables:

The names used for states, controls, and variables make the code easier to read. Consider using names such as "speed", "position", "concentration", etc. instead of "x1", "x2", "x3".

The names used for equations do not matter in most cases. However, they will be useful when accessing Lagrange multipliers.

Re-solve on successively finer grids:

If a very fine grid is needed in order to obtain a good solution, it is usually a good idea to solve the problem on a less dense grid first, and then re solve, by using the obtained solution as a starting point. The following code will do the trick:

```
for n=[10 40]
   p = tomPhase('p', t, 0, 6, n);
   setPhase(p);
   tomStates x1 x2

   % Initial guess
```

```
    if n == 10
        x0 = {p1 == 0; p2 == 0};
    else
        x0 = {p1 == p1opt; p2 == p2opt
            icollocate({x1 == x1opt; x2 == x2opt})};
    end
    ...
    ...
    ...
    solution = ezsolve(objective, {cbox, cbnd, ceq}, x0, options);

    % Optimal x, p for starting point
    x1opt = subs(x1, solution);
    x2opt = subs(x2, solution);
    p1opt = subs(p1, solution);
    p2opt = subs(p2, solution);
end
```

See for example *denbighSystem.m* and *drugScheduling.m*.

3.3.1 Modeling notes

The examples included with the software in many cases just scratch the surface of the capabilities of PROPT. The software is designed for maximum user flexibility and with many "hidden" features. The following notes are worth keeping in mind when modeling optimal control problems.

Left-hand side treatment:

The left hand side of equations do not have to be a single time-derivative. Things like $collocate(m * dot(v) == F)$ work just fine, even if m is a state variable (A constraint that prevents m from becoming to small may improve convergence).

Second derivatives:

Second derivatives are allowed, as in $collocate(dot(dot(x)) == a)$, although convergence is almost always better when including extra states to avoid this.

Equations:

Equations do not have to include any time-derivative. For example $collocate(0.5 * m * v^2 + m * g * h) == initial(0.5 * m * v^2 + m * g * h)$ will work just fine in the PROPT framework.

Fixed end times:

Problems with fixed end-times enables the use of any form of expression involving the collocation points in time since the collocation points are pre-determined.

The following illustrates how t can (should) be used.

```
toms t
% Will cause an error if myfunc is undefined for tomSym
myfunc(t);
```

```
% This works since collocate(t) is a double vector and not a
% tomSym object.
myfunc(collocate(t))
```

It is also worth noting that feval bypasses tomSym, so it is possible to call any function, at the expense of slower derivatives. In the case of a fixed end time, there are no derivatives:

```
toms t
% Works since tomSym is bypassed.
collocate(feval('myfunc',t))
```

Segmented constraints:

With PROPT is it possible to define constraints that are valid for only a part of phase. In addition, it is possible to define constraints for other points than the collocation points which could assist in avoiding constraint violations in between the collocation points.

mcollocate automates this process for the entire phase and imposes twice the number of constraints, i.e. the solution will be constrained in between each collocation point as well.

The following example shows how to set segmented constraints (valid for a part of the phase) for time-free and time-fixed problems.

```
% For variable end time (the number of constraints need to be fixed)
% Relax all constraints below the cutoff, while 0 above
con = {collocate((8*(t-0.5).^2-0.5-x2) >= -1e6*(t<0.4))};

% When the end time is fixed there are many possible solutions.
% One can use custom points or a set of collocation points if needed.
tsteps = collocate(t);
n = length(tsteps);
y = atPoints(p,tsteps(round(n/3):round(n/2)),(8*(t-0.5).^2-0.5-x2));
con = {y >= 0};

% Alternatively one could write for custom points.
con = atPoints(p,linspace(0.4, 0.5, 10),(8*(t-0.5).^2-0.5-x2) >= 0);
```

Constant unknowns:

It is possible to mix constant unknowns (created with "toms") into the equations, and those will become part of the solution: $icollocate(0.5 * m * v^2 + m * g * h == Etot)$.

Higher-index DAEs:

Higher-index DAEs usually converge nicely with PROPT. One can use dot() on entire expressions to get symbolic time derivatives to use when creating lower index DAEs.

Lagrangian equations:

Deducing Lagrangian equations manually is usually not necessary, but if needed, one can use tomSym for derivatives. TomSym allows for computation of symbolic derivatives with respect to constant scalars and matrices using the "derivative" function, but this does not work for tomStates or tomControls. The current implementation is therefore to create expressions using normal tomSym variables (created with the "toms" command), use the derivative function and then use "subs" function to replace the constants with states and controls before calling collocate.

3.4 Independent variables, scalars and constants

Independent variables can be defined as follows (yes, it is that simple!):

```
toms t  % Time
toms tf % Final time
% Phase name, time, start, end, number of nodes
p = tomPhase('p', t, 0, tf, 25);
cbox = {20 <= tf <= 30}; % Bounds on end time
x0 = {tf == 22); % Starting point for end time (must be set before used
                 % in other expressions)
```

It is also possible use scalar variables within the PROPT framework:

```
toms p1
cbox = {1 <= p1 <= 2};
x0 = {p1 == 1.3);
```

where the variables define lower, upper bound and a suitable guess.

A constant variable ki0 can be defined with the following statement (i.e. no difference from normal Matlab code):

```
ki0 = [1e3; 1e7; 10; 1e-3];
```

3.5 State and control variables

The difference between state and control variables is that states are continuous between phases, while control variables can be discontinuous.

Examples:

Unconstrained state variable x_1 with a starting guess from 0 to 1:

```
tomStates x1
x0 = icollocate(x1 == t/tf);
```

State variable x_1 with bounds of 0.5 and 10. The starting guess should be set as well to avoid any potential singularities:

```
tomStates x1
cbox = {0.5 <= icollocate(x1) <= 10};
```

Control variable T with a "table" defining the starting point:

```
tomControls T
x0 = collocate(T==273*(t<100)+415*(t>=100));
```

3.6 Boundary, path, event and integral constraints

Boundary constraints are defined as expressions since the problem size is reduced. For example if state variable x_1 has to start in 1 and end in 1 and x_2 has to travel from 0 to 2 define:

```
cbnd1 = initial(x1 == 1);
cbnd2 = initial(x2 == 0);
cbnd3 = final(x1 == 1);
cbnd4 = final(x2 == 2);
```

A variety of path, event and integral constraints are shown below:

```
x3min      = icollocate(x3 >= 0.5);     % Path constraint
integral1 = {integrate(x3) - 1 == 0}; % Integral constraint
final3     = final(x3) >= 0.5;          % Final event constraint
init1      = initial(x1) <= 2.0;        % Initial event constraint
```

NOTE: When defining integral constraints that span over several phase, use either of the following notations:

```
integrate(p1,x3p1) + integrate(p2,x3p1) + integrate(p3,x3p1) <= 2.1e3
```

or with expressions:

```
p1qx3 == integrate(p1,x3p1);
p2qx3 == integrate(p2,x3p1);
p3qx3 == integrate(p3,x3p1);

qx2sum = {p1qx3 + p2qx3 + p3qx3 <= 2.1e3};
```

4 Multi-phase optimal control

brysonDenhamTwoPhase.m is a good example for multi-phase optimal control. The user simply have to link the states to complete the problem formulation.

```
link = {final(p1,x1p1) == initial(p2,x1p2)
        final(p1,x2p1) == initial(p2,x2p2)};
```

It is important to keep in mind that the costs have to be individually defined for the phases in the case that a integrating function is used.

5 Scaling of optimal control problems

Scaling of optimal control problems are best done after they are defined. The level of the cost function, variables, states and controls should be balanced around 1.

PROPT does however, have an automated scaling feature that could be tested if needed. The option can be activated by setting:

```
options.scale = 'auto';
```

shuttleEntry.m and *minimumClimbEng.m* exemplify how to use and enable the scaling of optimal control problems.

6 Setting solver and options

The solver options can be set in *options.**. See *help ezsolve* for more information.

The following code exemplifies some basic settings:

```
options.solver = 'knitro'; % Change the solver
options.type   = 'lpcon';  % Set the problem type to avoid identification
                           % Nonlinear problem with linear objective
```

It is also possible to gain complete control over the solver by bypassing the *ezsolve* command and converting the problem to a standard TOMLAB Prob structure.

The following code will, for example, run vanDerPol (it is possible to edit the example directly of course), then convert the problem and run it with ALG 3 (SLQP) in KNITRO, and finally extract the standard solution.

```
vanDerPol;
Prob = sym2prob(objective, {cbox, cbnd, ceq}, x0, options);
Prob.KNITRO.options.ALG = 3;
Result = tomRun('knitro', Prob, 1);
solution = getSolution(Result);
```

7 Solving optimal control problems

The preceding sections show how to setup and define an optimal control problem using PROPT. Many additional features have been implemented, such as advanced input error identification, to facilitate fast modeling and debugging.

7.1 Standard functions and operators

In the preceding sections several different Matlab functions and operators were used. This section lists the ones that are useful for the end user.

7.1.1 collocate — Expand a propt tomSym to all collocation points on a phase.

`y = collocate(phase, x)` for a m-by-n tomSym object x, on a phase with p collocation points, returns an p-by-m*n tomSym with values of x for each collocation point.

If x is a cell array of tomSym objects, then collocate is applied recursively to each element in the array.

See also: atPoints

7.1.2 tomSym/dot

Shortcut to overdot (alternatively dot product).

`dot(p,x)` gives the time derivative of x in the phase p.

`dot(x)` can be used to the same effect, if `setPhase(p)` has been called previously.

7.1.3 final — Evaluate a propt tomSym at the final point of a phase.

`y = final(phase, x)` for tomSym object x, returns an object of the same size as x, where the independent variable (usually t) has been replaced by its final value on the phase.

See also: initial, subs, collocate, atPoints

7.1.4 icollocate — Expand a propt tomSym to all interpolation points on a phase

`y = icollocate(phase, x)` is the same as `y = collocate(phase, x)`, except that the interpolation points are used instead of the collocation points. This is typically useful when constructing an initial guess.

See also: collocate, atPoints

7.1.5 initial — Evaluate a propt tomSym at the initial point of a phase.

`y = initial(phase, x)` for tomSym object x, returns an object of the same size as x, where the independent variable (usually t) has been replaced by its initial value on the phase (often 0).

If x is a cell array of tomSym objects, then initial is applied recursively to each element in the array.

See also: final, subs, collocate, atPoints

7.1.6 integrate — Evaluate the integral of an expression in a phase.

`y = integrate(phase, x)` for tomSym object x, returns an object which has the same size as x and is the integral of x in the given phase.

See also: atPoints

7.1.7 mcollocate — Expand to all collocation points, endpoints and midpoints on a phase.

`y = mcollocate(phase, x)` for a m-by-n tomSym object x, on a phase with p collocation points, returns an (2p+1)-by-m*n tomSym with values of x for each collocation point, the endpoints and all points that lie halfway inbetween these points.

The mcollocate function is useful in setting up inequalities that involve state variables. Because twice as many points are used, compared to collocate, the resulting problem is slightly slower to solve, but the obtained solution is often more correct, because overshoots in between collocation points are smaller.

Because it uses many more points than there are degrees of freedom, mcollocate should only be used for inequalities. Applying mcollocate to equalities will generally result in an optimization problem that has no solution. Care should also be taken to ensure that the mcollocated condition is not in conflict with any initial or final condition.

If x is a cell array of tomSym objects, then mcollocate is applied recursively to each element in the array.

If a finite element method is used, then mcollocate uses all points that are used in computing the numeric quadrature over elements.

See also: collocate, icollocate, atPoints

7.1.8 setPhase — Set the active phase when modeling PROPT problem.

`setPhase(p)` sets the active phase to p.

It is not strictly necessary to use this command, but when doing so, it is possible to omit the phase argument to the commands tomState, tomControl, initial, final, integrate, etc.

7.1.9 tomControl — Generate a PROPT symbolic state.

```
x = tomControl creates a scalar PROPT control with an automatic name.
x = tomControl(phase,label) creates a scalar control with the provided name.
x = tomControl(phase,label,m,n) creates a m-by-n matrix of controls.
x = tomControl(phase,[],m,n) creates a matrix control with an automatic name.
x = tomControl(phase,label,m,n,'int') creates an integer matrix symbol.
x = tomControl(phase,label,m,n,'symmetric') creates a symmetric matrix symbol.
```

The tomControl symbols are different from tomState symbols in that the states are assumed to be continuous, but not the controls. This means that derivatives of tomControls should typically not be used in the differential equations, and no initial or final condition should be imposed on a tomControl.

If setPhase has been used previously, then the phase is stored in a global variable, and the phase argument can be omitted.

Constructs like "`x = tomControl('x')`" are very common. There is the shorthand notation "tomControls x".

See also: tomControls, tomState, tom

7.1.10 tomControls — Create tomControl objects

Examples:

```
tomControls x y z
```

is equivalent to

```
x = tomControl('x');
y = tomControl('y');
z = tomControl('z');

tomControls 2x3 Q 3x3 -integer R -symmetric S
```

is equivalent to

```
Q = tomControl('Q', 2, 3);
R = tomControl('R', 3, 3, 'integer');
S = tomControl('S', 3, 3, 'integer', 'symmetric');
```

Note: While the "tomControls" shorthand is very convenient to use prototyping code, it is recommended to only use the longhand "tomControl" notation for production code. (See toms for the reason why.)

It is necessary to call setPhase before calling tomStates.

See also tomControl, setPhase, tomState tomStates, tom, toms

7.1.11 tomPhase — Create a phase struct

`phase = tomPhase(label, t, tstart, tdelta, ipoints, cpoints)` The ipoints (interpolation points) and cpoints (collocation points) input arguments must be vectors of unique sorted points on the interval 0 to 1.

`phase = tomPhase(label, t, tstart, tdelta, n)` automatically creates cpoints and ipoints using n Gauss points. (If n > 128 then Chebyshev points are used instead.)

`phase = tomPhase(label, t, tstart, tdelta, n, [], 'cheb')` uses Chebyshev points instead of Gauss points. This yields better convergence for some problems, and worse for others, as compared to Gauss points.

See also: collocate

7.1.12 tomState — Generate a PROPT symbolic state

```
x = tomState creates a scalar PROPT state with an automatic name.
x = tomState(phase,label) creates a scalar state with the provided name.
x = tomState(phase,label,m,n) creates a m-by-n matrix state.
x = tomState(phase,[],m,n) creates a matrix state with an automatic name.
x = tomState(phase,label,m,n,'int') creates an integer matrix symbol.
x = tomState(phase,label,m,n,'symmetric') creates a symmetric matrix symbol.
```

The tomState symbols are different from tomControl symbols in that the states are assumed to be continuous. This means that they have time derivatives, accessible via the `dot()` function, and that tomStates cannot be integers.

If setPhase has been used previously, then the phase is stored in a global variable, and the phase argument can be omitted.

Constructs like "`x = tomState('x')`" are very common. There is the shorthand notation "tomStates x".

See also: tomStates, tom, tomControl, setPhase

7.1.13 tomStates — Create tomState objects as toms create tomSym objects

Examples:

```
tomStates x y z
```

is equivalent to

```
x = tomState('x');
y = tomState('y');
z = tomState('z');

tomStates 2x3 Q 3x3 R -symmetric S
is equivalent to
Q = tomState('Q', 2, 3);
R = tomState('R', 3, 3);
S = tomState('S', 3, 3, 'symmetric')

tomStates 3x1! v
is equivalent to
v1 = tomState('v1');
v2 = tomState('v2');
v3 = tomState('v3');
v = [v1;v2;v3];
```

See the help for "toms" for more info on how to use the different flags.

Note: While the "tomStates" shorthand is very convenient to use prototyping code, it is recommended to only use the longhand "tomState" notation for production code. (See toms for the reason why.)

It is necessary to call setPhase before calling tomStates.

See also tomState, tomControl, setPhase, tomControls, toms, tom

7.2 Advanced functions and operators

Some user may wish to use more advanced function of PROPT as well. The following function can be useful in many situations.

7.2.1 atPoints — Expand a propt tomSym to a set of points on a phase.

`y = atPoints(phase, t, x)` for a m-by-n tomSym object x, and a vector t of length p, returns an p-by-m*n tomSym with values of x for each value of t (this is done via substitution).

If x is a cell array of tomSym objects, then atPoints is applied recursively to each element in the array.

If x is an equality or inequality, and the left-hand-side is divided by a symbolic phase.tdelta, then both sides are multiplied by phase.tdelta

See also: tomPhase

```
Overloaded methods:
   tomSym/atPoints
   tomSym/atPoints
```

7.2.2 interp1p — Polynomial interpolation.

`yi = interp1p(x,y,xi)` fits a polynomial to the points (x,y) and evaluates that polynomial at xi to compute yi.

The behavior of interp1p is different from that of interp1 in that the same high-degree polynomial is fitted to all points. This is usually not a good idea, and for general interpolation interp1 is to be preferred. However, it works well in some cases, such as when the points x are the roots of a Legendre polynomial.

```
Overloaded methods:
   tomSym/interp1p
   tomSym/interp1p
```

7.2.3 proptGausspoints — Generate a list of gauss points.

`[r, w] = gausspoints(nquad)` = `proptGausspoints(n)`

Input: n - The number of points requested

Output r - The Gauss points (roots of a Legendre polynomial of order n.) w - The weights for Gaussian quadrature.

7.2.4 proptDiffMatrix — Differentiation matrix for interpPoly.

`M = proptDiffMatrix(X,XI,N)` creates a matrix M of the values of the N:th derivatives of the interpolating polynomials defined by the roots X, at the points XI. Each column of M corresponds to a specific root, and each row corresponds to a component of XI.

7.3 Screen output

The screen output depends on the functionality of the nonlinear programming solver. In most cases screen output can be generated with the following command:

```
options.PriLevOpt = 1; % or higher for more output
```

A print level of 3 is recommended when using the global NLP solver *multiMin*.

7.4 Solution structure

The solution structure from PROPT contains the relevant results.

It is possible to use the solution information to evaluate any custom expressions. For example:

```
subs(x1.*x2,solution);
```

7.5 Plotting

The best way to plot the data is to directly use the data fields as needed. There are also some automated plotting routines included, see for example *help tomSym/ezplot*.

8 || OPTIMAL CONTROL EXAMPLES ||

The following sections contain all the optimal control examples included with PROPT.

9 Acrobot

Russ Tedrake. Underactuated Robotics: Learning, Planning, and Control for Efficient and Agile Machines. Working Draft of Course Notes for MIT 6.832 (Chapter 3).

9.1 Problem Formulation

The animation can be found here:

http://tomdyn.com/examples/acrobot.avi

```
Problem type appears to be: qpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Acrobot                        f_k      15.819743676706992000
                                      sum(|constr|)      0.000005838568196298
                              f(x_k) + sum(|constr|)     15.819749515275188000
                                             f(x_0)      0.000000000000000000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv  303 ConJacEv  303 Iter  159 MinorIter 1149
CPU time: 14.656250 sec. Elapsed time: 15.360000 sec.
```

10 A Linear Problem with Bang Bang Control

Paper: Solving Tough Optimal Control Problems by Network Enabled Optimization Server (NEOS)

Jinsong Liang, YangQuan Chen, Max Q.-H. Meng, Rees Fullmer Utah State University and Chinese University of Hong Kong (Meng)

EXAMPLE-1: A TEXTBOOK BANG-BANG OPTIMAL CONTROL PROBLEM

10.1 Problem description

Find u over t in [0; t_F] to minimize

$$J = t_F$$

subject to:

$$\frac{dx_1}{dt} = x_2$$
$$\frac{dx_2}{dt} = u$$

$$x_1(0) = 0$$
$$x_1(t_F) = 300$$
$$x_2(0) = 0$$
$$x_2(t_F) = 0$$

$$-2 <= u <= 1$$

Reference: [23]

10.2 Problem setup

```
toms t
toms t_f

p = tomPhase('p', t, 0, t_f, 30);
setPhase(p);

tomStates x1 x2
tomControls u

% Initial guess
% Note: The guess for t_f must appear in the list before expression involving t.
x0 = {t_f == 20
    icollocate({x1 == 300*t/t_f; x2 == 0})
    collocate(u==1-2*t/t_f)};

% Box constraints
cbox = {10 <= t_f <= 40
    -2 <= collocate(u) <= 1};

% Boundary constraints
cbnd = {initial({x1 == 0; x2 == 0})
    final({x1 == 300; x2 == 0})};

% ODEs and path constraints
ceq = collocate({dot(x1) == x2; dot(x2) == u});

% Objective
objective = t_f;
```

10.3 Solve the problem

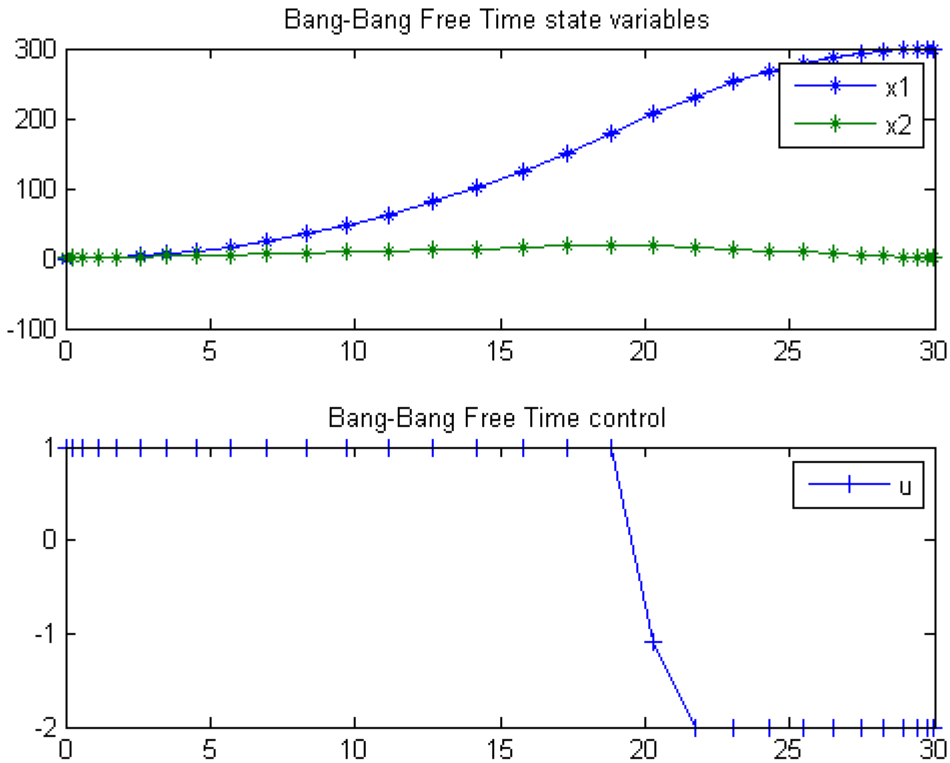
```
options = struct;
options.name = 'Bang-Bang Free Time';
options.prilev = 1;
solution = ezsolve(objective, {cbox, cbnd, ceq}, x0, options);
t  = subs(collocate(t),solution);
x1 = subs(collocate(x1),solution);
x2 = subs(collocate(x2),solution);
u  = subs(collocate(u),solution);
```

```
Problem type appears to be: lpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
```

```
===============================================================================================
Problem: ---  1: Bang-Bang Free Time             f_k      30.019823270451944000
                                       sum(|constr|)      0.000028878808460443
                              f(x_k) + sum(|constr|)     30.019852149260405000
                                              f(x_0)     20.000000000000000000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv  526 ConJacEv  526 Iter  136 MinorIter  185
CPU time: 0.750000 sec. Elapsed time: 0.766000 sec.
```

10.4 Plot result

```
subplot(2,1,1)
plot(t,x1,'*-',t,x2,'*-');
legend('x1','x2');
title('Bang-Bang Free Time state variables');

subplot(2,1,2)
plot(t,u,'+-');
legend('u');
title('Bang-Bang Free Time control');
```


Bang-Bang Free Time state variables
x1
x2
300
200
100
0
-100
0
5
10
15
20
25
30
Bang-Bang Free Time control
u
1
0
-1
-2
0
5
10
15
20
25
30

11 Batch Fermentor

Dynamic optimization of bioprocesses: efficient and robust numerical strategies 2003, Julio R. Banga, Eva Balsa-Cantro, Carmen G. Moles and Antonio A. Alonso

Case Study I: Optimal Control of a Fed-Batch Fermentor for Penicillin Production

11.1 Problem description

This problem considers a fed-batch reactor for the production of penicillin, as studied by Cuthrell and Biegler (1989). This problem has also been studied by many other authors (Dadebo & McAuley 1995, Banga & Seider 1996, Banga et al. 1997). We consider here the free terminal time version where the objective is to maximize the amount of penicillin using the feed rate as the control variable. It should be noted that the resulting NLP problem (after using CVP) does not seem to be multimodal, but it has been reported that local gradient methods do experience convergence problems if initialized with far-from-optimum profiles, or when a very refined solution is sought. Thus, this example will be excellent in order to illustrate the better robustness and efficiency of the alternative stochastic and hybrid approaches. The mathematical statement of the free terminal time problem is:

Find u(t) and t_f over t in [t0; t_f] to maximize

$$J = x_2(t_f) * x_4(t_f)$$

subject to:

$$\frac{dx_1}{dt} = h_1 * x_1 - \frac{u * x_1}{500 * x_4}$$
$$\frac{dx_2}{dt} = h_2 * x_1 - 0.01 * x_2 - \frac{u * x_2}{500 * x_4}$$
$$\frac{dx_3}{dt} = \frac{h_1 * x_1}{0.47} - \frac{h_2 * x_1}{1.2} - \frac{x_1 * 0.029 * x_3}{0.0001 + x_3} + u * x_4 * (1 - \frac{x_3}{500})$$
$$\frac{dx_4}{dt} = \frac{u}{500}$$

$$h_1 = \frac{0.11 * x_3}{0.006 * x_1 + x_3}$$
$$h_2 = 0.0055 * x_3 * (0.0001 + x_3 * (1 + 10 * x_3))$$

where x1, x2, and x3 are the biomass, penicillin and substrate concentrations (g=L), and x4 is the volume (L). The initial conditions are:

$$x(t_0) = [1.5\ 0\ 0\ 7]'$$

There are several path constraints (upper and lower bounds) for state variables (case III of Cuthrell and Biegler, 1989):

$$0 <= x1 <= 40$$

$$0 <= x3 <= 25$$

$$0 <= x4 <= 10$$

The upper and lower bounds on the only control variable (feed rate of substrate) are:

$$0 <= u <= 50$$

Reference: [3]

11.2 Solving the problem on multiple grids.

The problem is solved in two stages. First, a solution is computed for a small number of collocation points, then the number of collocation points is increased, and the problem is resolved. This saves time, compared to using the fine grid immediately.

```
toms t
toms t_f

nvec = [35 70 80 90 100];

for i=1:length(nvec)

    n = nvec(i);
    p = tomPhase('p', t, 0, t_f, n);
    setPhase(p);

    tomStates x1 x2 x3 x4
```

```
tomControls u

% Initial guess
% Note: The guess for t_f must appear in the list before
% expression involving t.
if i==1
    x0 = {t_f == 126
        icollocate(x1 == 1.5)
        icollocate(x2 == 0)
        icollocate(x3 == 0)
        icollocate(x4 == 7)
        collocate(u==11.25)};
else
    % Copy the solution into the starting guess
    x0 = {t_f == tf_init
        icollocate(x1 == x1_init)
        icollocate(x2 == x2_init)
        icollocate(x3 == x3_init)
        icollocate(x4 == x4_init)
        collocate(u == u_init)};
end

% Box constraints
% Setting the lower limit for t, x1 and x4 to slightly more than zero
% ensures that division by zero is avoided during the optimization
% process.
cbox = {1 <= t_f  <= 256
    1e-8 <= mcollocate(x1) <= 40
    0    <= mcollocate(x2) <= 50
    0    <= mcollocate(x3) <= 25
    1    <= mcollocate(x4) <= 10
    0    <= collocate(u)   <= 50};

% Various constants and expressions
h1 = 0.11*(x3./(0.006*x1+x3));
h2 = 0.0055*(x3./(0.0001+x3.*(1+10*x3)));

% Boundary constraints
cinit = initial({x1 == 1.5; x2 == 0
    x3 == 0; x4 == 7});

% This final condition is not necesary, but helps convergence speed.
cfinal = final(h2.*x1-0.01*x2) == 0;

% ODEs and path constraints
ceq = collocate({
    dot(x1) == h1.*x1-u.*(x1./500./x4)
```

```
        dot(x2) == h2.*x1-0.01*x2-u.*(x2./500./x4)
        dot(x3) == -h1.*x1/0.47-h2.*x1/1.2-x1.*...
        (0.029*x3./(0.0001+x3))+u./x4.*(1-x3/500)
        dot(x4) == u/500});

    % Objective
    objective = -final(x2)*final(x4);

    options = struct;
    options.name = 'Batch Fermentor';
    %options.scale = 'auto';
    %if i==1
    %    options.solver = 'multiMin';
    %    options.xInit = 20;
    %end
    solution = ezsolve(objective, {cbox, cinit, cfinal, ceq}, x0, options);
```

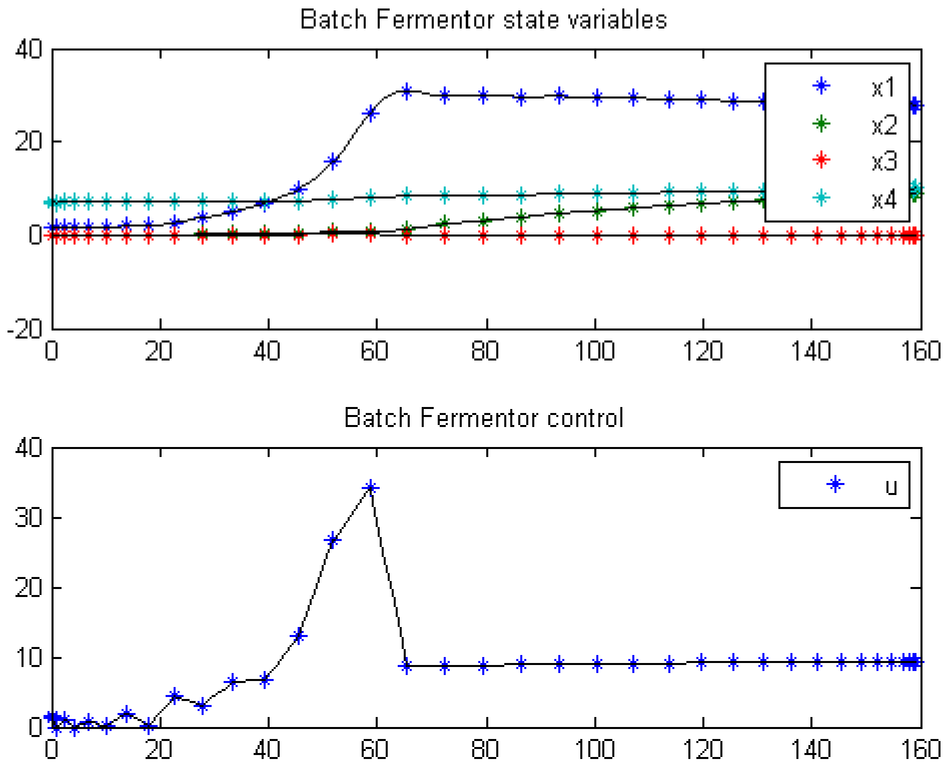
```
Problem type appears to be: qpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Batch Fermentor                f_k     -87.746072952335396000
                                       sum(|constr|)      0.000000000591461608
                              f(x_k) + sum(|constr|)    -87.746072951743940000
                                              f(x_0)      0.000000000000000000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv  924 ConJacEv  924 Iter  275 MinorIter 4942
CPU time: 8.250000 sec. Elapsed time: 8.485000 sec.


Problem type appears to be: qpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Batch Fermentor                f_k     -87.965550967205928000
                                       sum(|constr|)      0.000000291694933232
                              f(x_k) + sum(|constr|)    -87.965550675510997000
                                              f(x_0)    -87.746072952334629000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
```

```
Optimality conditions satisfied

FuncEv    1 ConstrEv   98 ConJacEv   98 Iter   58 MinorIter 1466
CPU time: 6.328125 sec. Elapsed time: 6.453000 sec.


Problem type appears to be: qpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Batch Fermentor                f_k     -87.989983108591034000
                                      sum(|constr|)      0.000000063125343100
                              f(x_k) + sum(|constr|)    -87.989983045465692000
                                             f(x_0)    -87.966078735435829000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv  116 ConJacEv  116 Iter   92 MinorIter 2087
CPU time: 14.265625 sec. Elapsed time: 14.438000 sec.


Problem type appears to be: qpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Batch Fermentor                f_k     -88.030366209342986000
                                      sum(|constr|)      0.000000440801301912
                              f(x_k) + sum(|constr|)    -88.030365768541685000
                                             f(x_0)    -87.990147691715819000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv   99 ConJacEv   99 Iter   88 MinorIter 1713
CPU time: 15.859375 sec. Elapsed time: 16.078000 sec.


Problem type appears to be: qpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Batch Fermentor                f_k     -88.044843218600391000
```

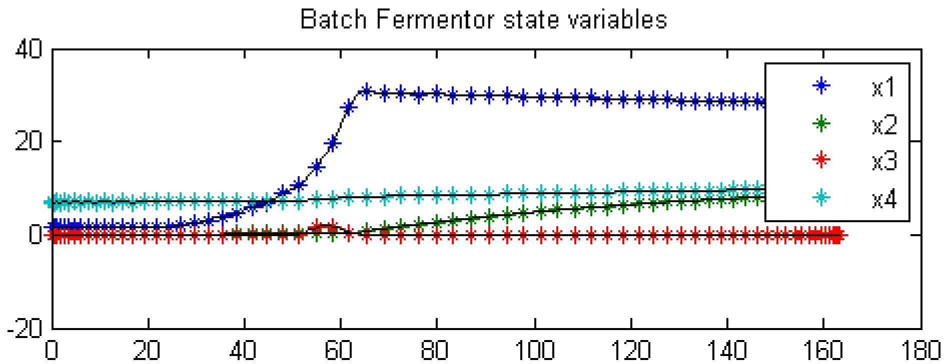
```
                    sum(|constr|)      0.000000326396054727
           f(x_k) + sum(|constr|)    -88.044842892204343000
                           f(x_0)    -88.030366874603772000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv  142 ConJacEv  142 Iter  122 MinorIter 2743
CPU time: 32.906250 sec. Elapsed time: 33.547000 sec.
```

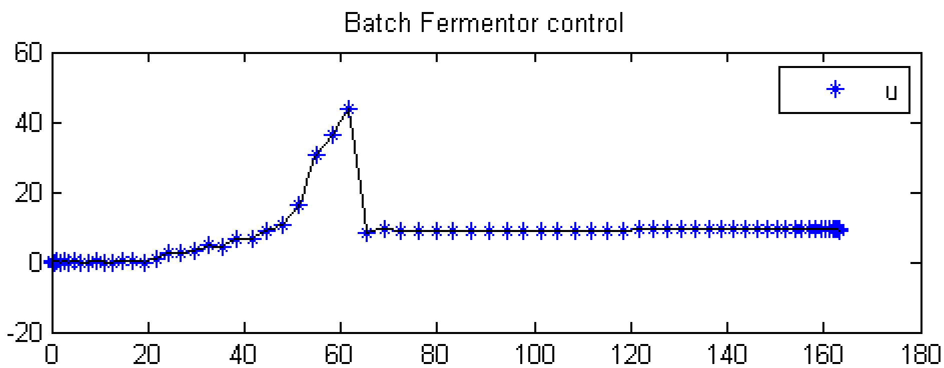
11.3 Plot result

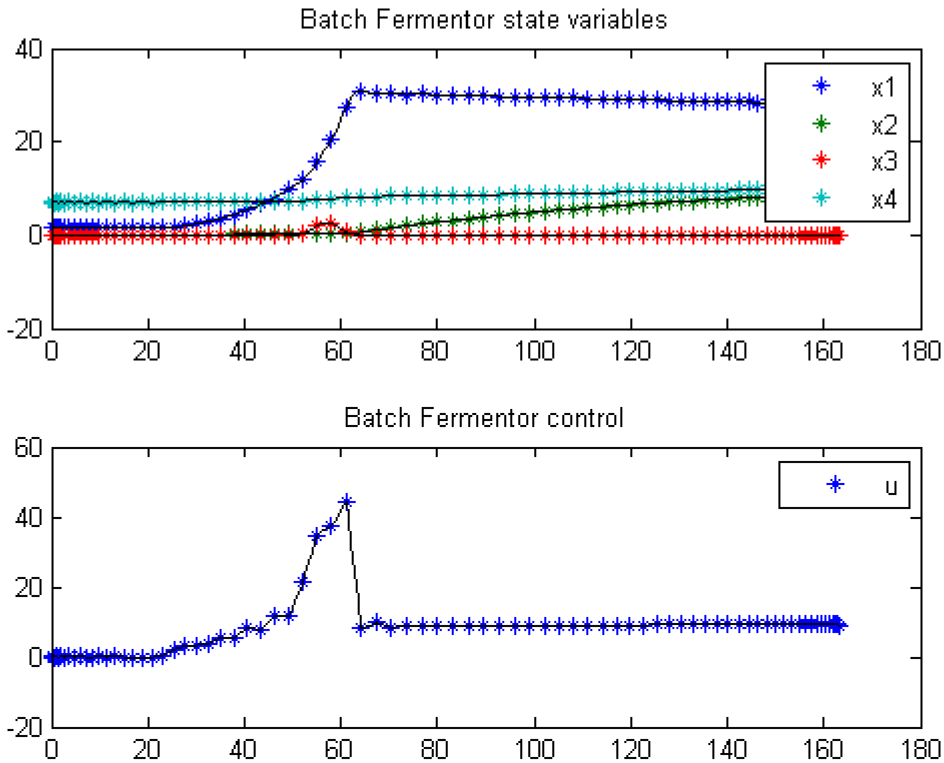
```
subplot(2,1,1);
ezplot([x1; x2; x3; x4]);
legend('x1','x2','x3','x4');
title('Batch Fermentor state variables');

subplot(2,1,2);
ezplot(u);
legend('u');
title('Batch Fermentor control');
drawnow

% Copy solution for initializing next round
x1_init  = subs(x1,solution);
x2_init  = subs(x2,solution);
x3_init  = subs(x3,solution);
x4_init  = subs(x4,solution);
u_init   = subs(u,solution);
tf_init  = subs(t_f,solution);
```

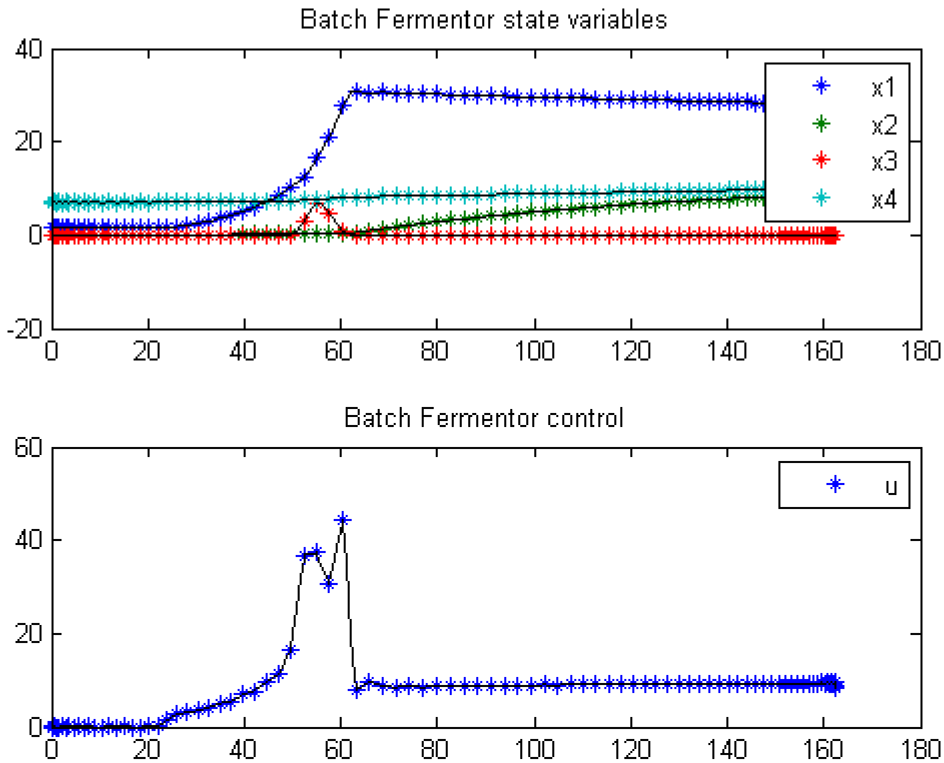


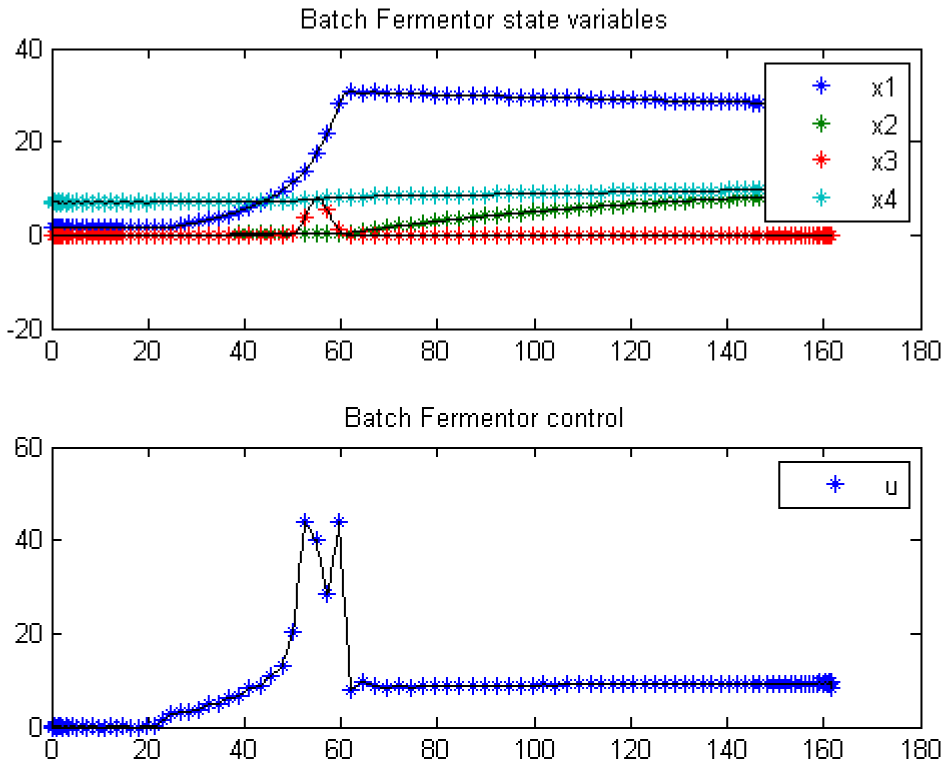
Batch Fermentor state variables
x1
x2
x3
x4
Batch Fermentor control
u


Batch Fermentor state variables
x1
x2
x3
x4


Batch Fermentor control
u


Batch Fermentor state variables
x1
x2
x3
x4
40
20
0
-20
0
20
40
60
80
100
120
140
160
180
Batch Fermentor control
u
60
40
20
0
-20
0
20
40
60
80
100
120
140
160
180


Batch Fermentor state variables
x1
x2
x3
x4
Batch Fermentor control
u


Batch Fermentor state variables
x1
x2
x3
x4
Batch Fermentor control
u

```
end
```

12 Batch Production

Dynamic optimization of bioprocesses: efficient and robust numerical strategies 2003, Julio R. Banga, Eva Balsa-Cantro, Carmen G. Moles and Antonio A. Alonso

Case Study II: Optimal Control of a Fed-Batch Reactor for Ethanol Production

12.1 Problem description

This case study considers a fed-batch reactor for the production of ethanol, as studied by Chen and Hwang (1990a) and others (Bojkov & Luus 1996, Banga et al. 1997). The (free terminal time) optimal control problem is to maximize the yield of ethanol using the feed rate as the control variable. As in the previous case, this problem has been solved using CVP and gradient based methods, but convergence problems have been frequently reported, something which has been confirmed by our own experience. The mathematical statement of the free terminal time problem is:

Find the feed flow rate u(t) and the final time t_f over t in [t0; t_f] to maximize

$$J = x_3(t_f) * x_4(t_f)$$

subject to:

$$\frac{dx_1}{dt} = g_1 * x_1 - u * \frac{x_1}{x_4}$$
$$\frac{dx_2}{dt} = -10 * g_1 * x_1 + u * \frac{150 - x_2}{x_4}$$
$$\frac{dx_3}{dt} = g_2 * x_1 - u * \frac{x_3}{x_4}$$
$$\frac{dx_4}{dt} = u$$

$$g_1 = \frac{0.408}{1 + x_3/16} * \frac{x_2}{0.22 + x_2}$$
$$g_2 = \frac{1}{1 + x_3/71.5} * \frac{x_2}{0.44 + x_2}$$

where x1, x2 and x3 are the cell mass, substrate and product concentrations (g/L), and x4 is the volume (L). The initial conditions are:

$$x(t_0) = [1\ 150\ 0\ 10]'$$

The constraints (upper and lower bounds) on the control variable (feed rate, L/h) are:

$$0 <= u <= 12$$

and there is an end-point constraint on the volume:

$$0 <= x_4(t_f) <= 200$$

Reference: [3]

12.2 Problem setup

```
toms t
toms tfs

t_f = 100*tfs;

% initial guesses
tfg = 60;
x1g = 10;
x2g = 150-150*t/t_f;
x3g = 70;
x4g = 200*t/t_f;
ug  = 3;

n = [  20     60   60   60];
e = [0.01  0.002 1e-4    0];

for i = 1:3

    p = tomPhase('p', t, 0, t_f, n(i));
    setPhase(p);

    tomStates x1s x2s x3s x4s
    if e(i)
        tomStates u
    else
```

```
    tomControls u
end
%tomControls g1 g2

% Create scaled states, to make the numeric solver work better.
x1 = 10*x1s;
x2 = 1*x2s;
x3 = 100*x3s;
x4 = 100*x4s;

% Initial guess
% Note: The guess for t_f must appear in the list before expression involving t.
x0 = {t_f == tfg
    icollocate({
    x1 == x1g
    x2 == x2g
    x3 == x3g
    x4 == x4g
    })
    collocate({u==ug})};

% Box constraints
cbox = {
    0.1 <= t_f <= 100
    mcollocate({
    0    <= x1
    0    <= x2
    0    <= x3
    1e-8 <= x4  % Avoid division by zero.
    })
    0 <= collocate(u) <= 12};

% Boundary constraints
cbnd = {initial({
    x1 == 1;
    x2 == 150
    x3 == 0;
    x4 == 10})
    final(0 <= x4 <= 200)};

% Various constants and expressions
g1 = (0.408/(1+x3/16))*(x2/(x2+0.22));
g2 = (1/(1+x3/71.5))*(x2/(0.44+x2));

% ODEs and path constraints
ceq = collocate({
    dot(x1) == g1*x1 - u*x1/x4
```

```
        dot(x2) == -10*g1*x1 + u*(150-x2)/x4
        dot(x3) == g2*x1 - u*x3/x4
        dot(x4) == u});

    % Objective
    J = final(x3*x4);
    if e(i)
        % Add cost on oscillating u.
        objective = -J/4900 + e(i)*integrate(dot(u)^2);
    else
        objective = -J/4900;
    end
```

12.3 Solve the problem

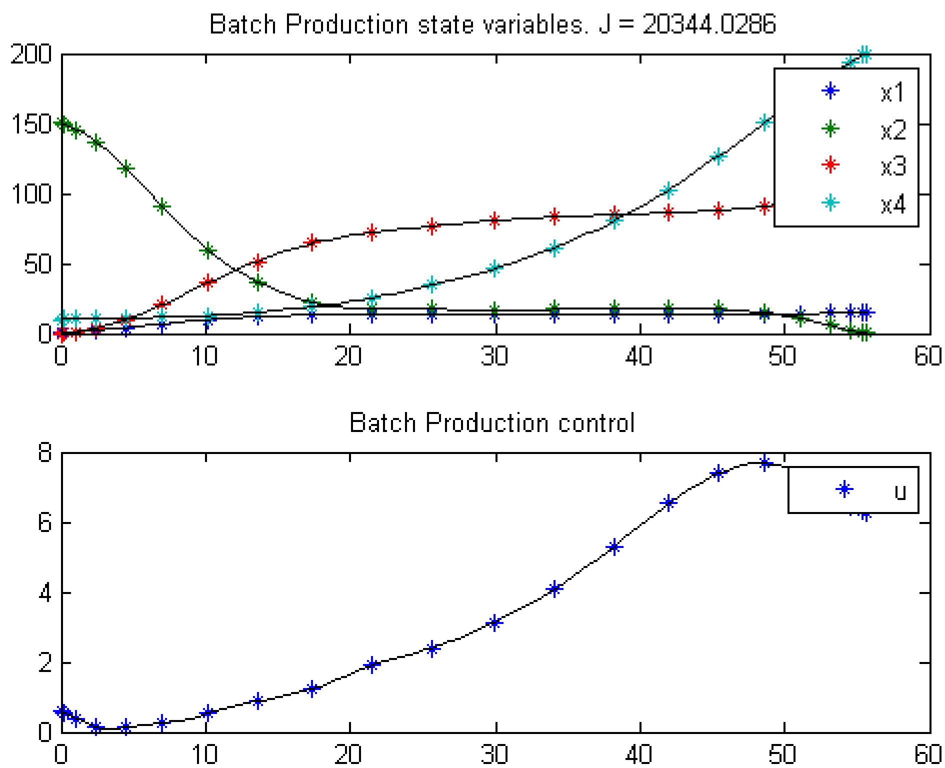
```
    options = struct;
    options.name = 'Batch Production';
    solution = ezsolve(objective, {cbox, cbnd, ceq}, x0, options);
```

```
Problem type appears to be: con
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Batch Production               f_k      -4.131438264402751400
                                       sum(|constr|)      0.316463714336870480
                              f(x_k) + sum(|constr|)     -3.814974550065881200
                                              f(x_0)      1.259999999999963600

Solver: snopt.  EXIT=1.  INFORM=32.
SNOPT 7.2-5 NLP code
Major iteration limit reached

FuncEv 4791 GradEv 4789 ConstrEv 4789 ConJacEv 4789 Iter 1000 MinorIter 5646
CPU time: 22.078125 sec. Elapsed time: 22.469000 sec.
Warning: Solver returned ExitFlag = 1
The returned solution may be incorrect.


Problem type appears to be: con
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Batch Production               f_k      -4.222925108597277000
                                       sum(|constr|)      0.000004304588346354
```

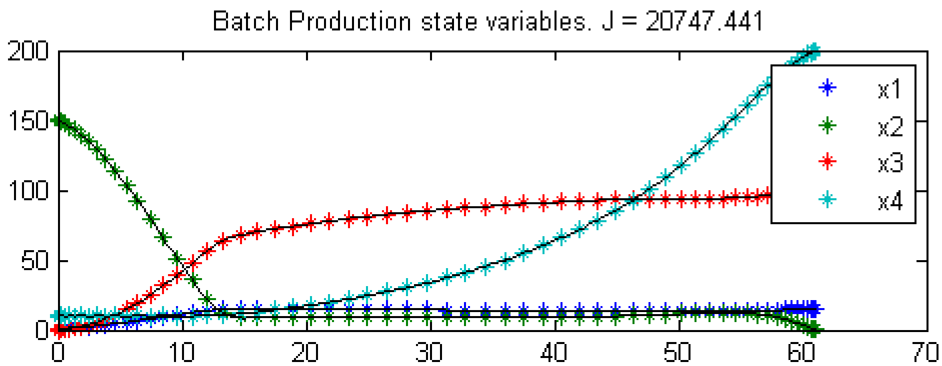
```
                          f(x_k) + sum(|constr|)     -4.222920804008930800
                                          f(x_0)     -1.781431131250453600

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv  517 GradEv  515 ConstrEv  515 ConJacEv  515 Iter  455 MinorIter  988
CPU time: 17.078125 sec. Elapsed time: 17.484000 sec.


Problem type appears to be: con
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Batch Production               f_k      -4.248461985888144300
                                       sum(|constr|)      0.000062226344424072
                          f(x_k) + sum(|constr|)     -4.248399759543720400
                                          f(x_0)     -4.126187662864473400

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv  277 GradEv  275 ConstrEv  275 ConJacEv  275 Iter  191 MinorIter  624
CPU time: 7.843750 sec. Elapsed time: 7.985000 sec.
```

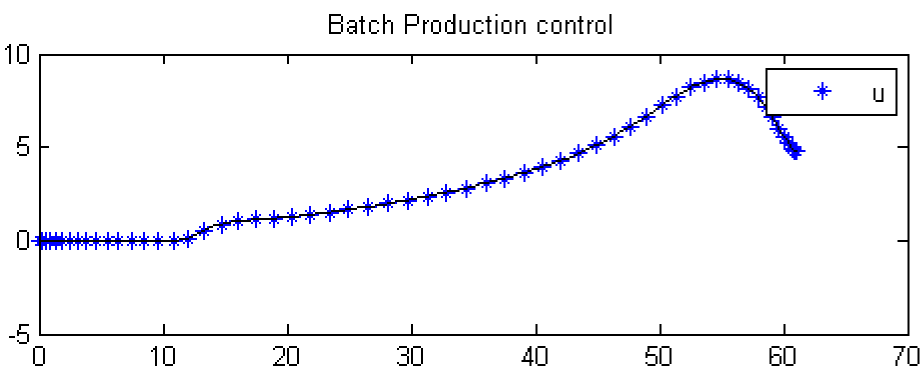
12.4 Plot result

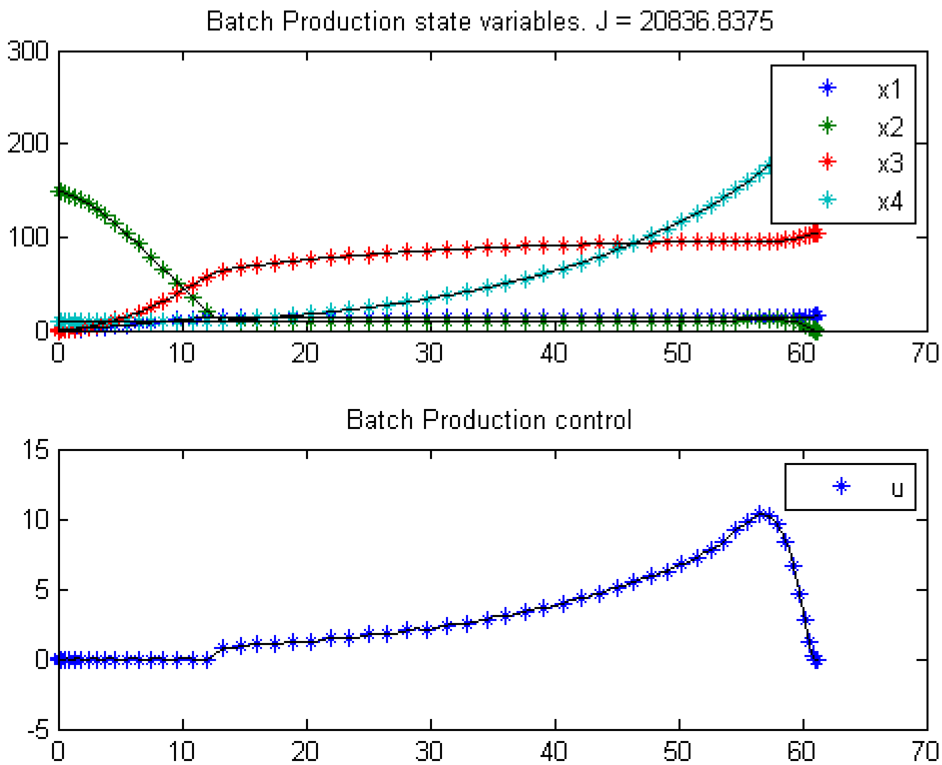
```
subplot(2,1,1)
ezplot([x1 x2 x3 x4]);
legend('x1','x2','x3','x4');
title(['Batch Production state variables. J = ' num2str(subs(J,solution))]);

subplot(2,1,2)
ezplot(u);
legend('u');
title('Batch Production control');
drawnow

% Copy inital guess for next iteration
tfg = subs(t_f,solution);
x1g = subs(x1,solution);
x2g = subs(x2,solution);
x3g = subs(x3,solution);
x4g = subs(x4,solution);
```



Batch Production state variables. J = 20344.0286
x1
x2
x3
x4
Batch Production control
u


Batch Production state variables. J = 20747.441
x1
x2
x3
x4


Batch Production control
u


Batch Production state variables. J = 20836.8375
x1
x2
x3
x4
300
200
100
0
0
10
20
30
40
50
60
70
Batch Production control
u
15
10
5
0
-5
0
10
20
30
40
50
60
70

```
end
```

13 Batch Reactor Problem

Example 6: DYNOPT User's Guide version 4.1.0

Batch reactor with reactions: A -> B -> C.

M. Cizniar, M. Fikar, M. A. Latifi, MATLAB Dynamic Optimisation Code DYNOPT. User's Guide, Technical Report, KIRP FCHPT STU Bratislava, Slovak Republic, 2006.

13.1 Problem description

Find T over t in [0; 1] to maximize

$$J = x_2(t_f)$$

subject to:

$$\begin{aligned}
\frac{dx_1}{dt} &= -k_1 * x_1^2 \\
\frac{dx_2}{dt} &= k_1 * x_1^2 - k_2 * x_2 \\
k_1 &= 4000 * exp^{-\frac{2500}{T}} \\
k_2 &= 620000 * exp^{-\frac{5000}{T}}
\end{aligned}$$

where

$$\begin{aligned}
x(0) &= [1\ 0] \\
298 <&= T <= 398
\end{aligned}$$

Reference: [13]

13.2 Problem setup

```
toms t
p = tomPhase('p', t, 0, 1, 30);
```

```
setPhase(p);

tomStates x1 x2
tomControls T

% Initial guess
% Note: The guess for t_f must appear in the list before expression involving t.
x0 = {icollocate({x1 == 1; x2 == 0})
    collocate(T==398-t*100)};

% Box constraints
cbox = {298 <= collocate(T) <= 398};

% Boundary constraints
cbnd = initial({x1 == 1; x2 == 0});

% Various constants and expressions
k1 = 4000*exp(-2500./T);
k2 = 620000*exp(-5000./T);

% ODEs and path constraints
ceq = collocate({dot(x1) == -k1.*x1.^2
    dot(x2) == k1.*x1.^2-k2.*x2});

% Objective
objective = -final(x2);
```

13.3 Solve the problem

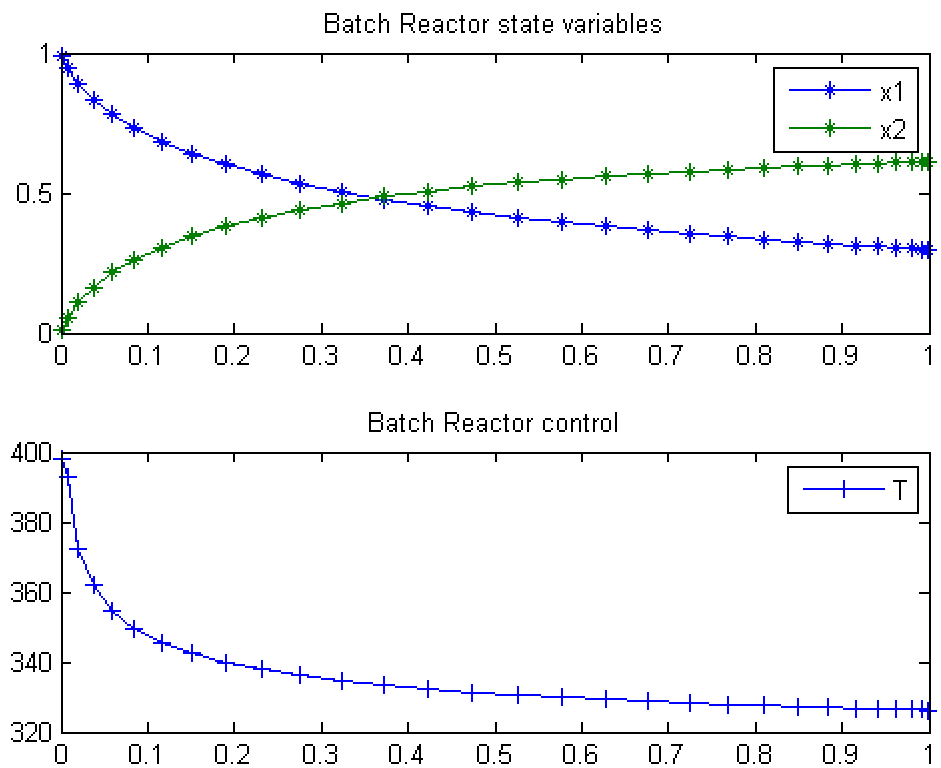
```
options = struct;
options.name = 'Batch Reactor';
solution = ezsolve(objective, {cbox, cbnd, ceq}, x0, options);

% Extract optimal states and controls from solution
t  = subs(collocate(t),solution);
x1 = subs(collocate(x1),solution);
x2 = subs(collocate(x2),solution);
T  = subs(collocate(T),solution);
```

```
Problem type appears to be: lpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Batch Reactor                  f_k      -0.610799380695553730
                                       sum(|constr|)      0.000006007956267540
```

```
                f(x_k) + sum(|constr|)     -0.610793372739286240
                               f(x_0)      0.000000000000000000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv   26 ConJacEv   26 Iter   23 MinorIter   84
CPU time: 0.093750 sec. Elapsed time: 0.094000 sec.
```

13.4 Plot result

```
subplot(2,1,1)
plot(t,x1,'*-',t,x2,'*-');
legend('x1','x2');
title('Batch Reactor state variables');

subplot(2,1,2)
plot(t,T,'+-');
legend('T');
title('Batch Reactor control');
```


Batch Reactor state variables
x1
x2
1
0.5
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Batch Reactor control
T
400
380
360
340
320
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

14 The Brachistochrone Problem

This problem was formulated by Johann Bernoulli, in Acta Eruditorum, June 1696

14.1 Problem description

"Given two points A and B in a vertical plane, what is the curve traced out by a point acted on only by gravity, which starts at A and reaches B in the shortest time."

In this example, we solve the problem numerically for A = (0,0) and B = (10,-3), and an initial speed of zero.

The mechanical system is modelled as follows:

$$
\begin{aligned}
\frac{dx}{dt} &= v \sin(\theta) \\
\frac{dy}{dt} &= -v \cos(\theta) \\
\frac{dv}{dt} &= g \cos(\theta)
\end{aligned}
$$

where (x,y) is the coordinates of the point, v is the velocity, and theta is the angle between the direction of movement and the vertical.

Reference: [6]

14.2 Problem setup

```
toms t
toms t_f
p = tomPhase('p', t, 0, t_f, 20);
setPhase(p);

tomStates x y v
tomControls theta

% Initial guess
% Note: The guess for t_f must appear in the list before expression involving t.
x0 = {t_f == 10
    icollocate({
    v == t
    x == v*t/2
    y == -1
```

```
    })
    collocate(theta==0.1)};

% Box constraints
cbox = {0.1 <= t_f <= 100
    0 <= icollocate(v)
    0 <= collocate(theta) <= pi};

% Boundary constraints
cbnd = {initial({x == 0; y == 0; v == 0})
    final({x == 10; y == -3})};

% ODEs and path constraints
g = 9.81;
ceq = collocate({
    dot(x) == v.*sin(theta)
    dot(y) == -v.*cos(theta)
    dot(v) == g*cos(theta)});

% Objective
objective = t_f;
```

14.3 Solve the problem

```
options = struct;
options.name = 'Brachistochrone';
solution = ezsolve(objective, {cbox, cbnd, ceq}, x0, options);
x     = subs(collocate(x),solution);
y     = subs(collocate(y),solution);
v     = subs(collocate(v),solution);
theta = subs(collocate(theta),solution);
t =     subs(collocate(t),solution);
```

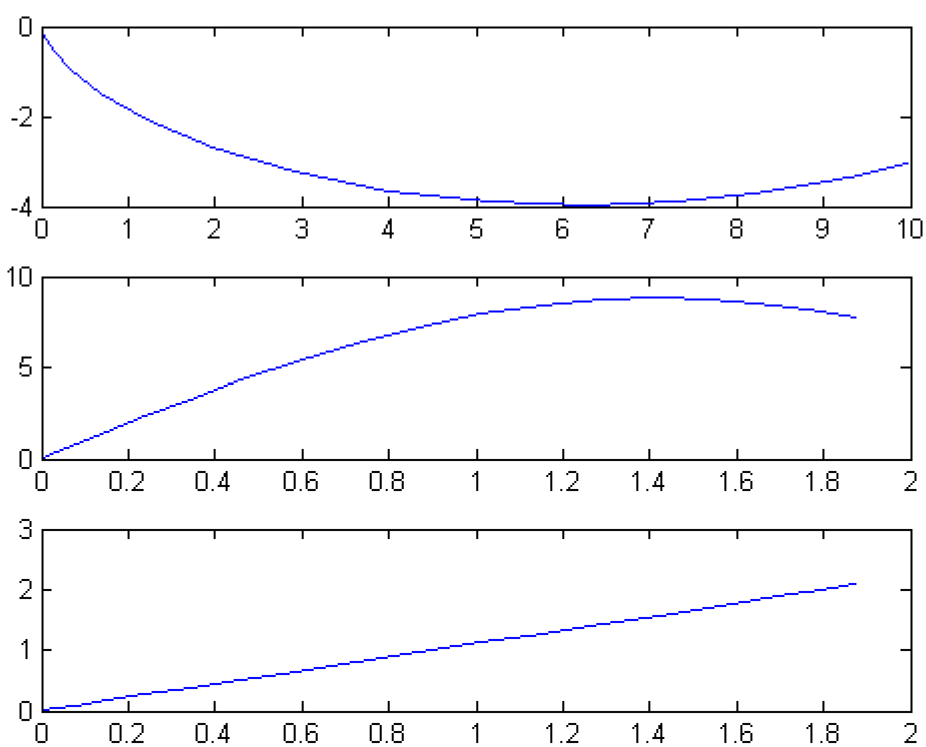
```
Problem type appears to be: lpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Brachistochrone                f_k       1.878940329113843100
                                       sum(|constr|)      0.000000174716635746
                               f(x_k) + sum(|constr|)      1.878940503830478900
                                              f(x_0)     10.000000000000000000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied
```

```
FuncEv    1 ConstrEv  834 ConJacEv  834 Iter  224 MinorIter  826
CPU time: 1.484375 sec. Elapsed time: 1.500000 sec.
```

14.4 Plot the result

To obtain the brachistochrone curve, we plot y versus x.

```
subplot(3,1,1)
plot(x, y);

subplot(3,1,2)
plot(t, v);

% We can also plot theta vs. t.
subplot(3,1,3)
plot(t, theta)
```



15 The Brachistochrone Problem (DAE formulation)

We will now solve the same problem as in brachistochrone.m, but using a DAE formulation for the mechanics.

15.1 DAE formulation

In a DAE formulation we don't need to formulate explicit equations for the time-derivatives of each state. Instead we can, for example, formulate the conservation of energy.

$$E_{kin} = \frac{m}{2}\left(\frac{dx}{dt}^2 + \frac{dx}{dt}^2\right),$$
$$E_{pot} = mgy.$$

The boundary conditions are still A = (0,0), B = (10,-3), and an initial speed of zero, so we have

$$E_{kin} + E_{pot} = 0$$

For complex mechanical systems, this freedom to choose the most convenient formulation can save a lot of effort in modelling the system. On the other hand, computation times may get longer, because the problem can to become more non-linear and the jacobian less sparse.

Reference: [6]

15.2 Problem setup

```
toms t
toms t_f

p = tomPhase('p', t, 0, t_f, 20);
setPhase(p);

tomStates x y

% Initial guess
x0 = {t_f == 10};

% Box constraints
cbox = {0.1 <= t_f <= 100};
```

```
% Boundary constraints
cbnd = {initial({x == 0; y == 0})
    final({x == 10; y == -3})};

% Expressions for kinetic and potential energy
m = 1;
g = 9.81;
Ekin = 0.5*m*(dot(x).^2+dot(y).^2);
Epot = m*g*y;
v = sqrt(2/m*Ekin);

% ODEs and path constraints
ceq = collocate(Ekin + Epot == 0);

% Objective
objective = t_f;
```

15.3 Solve the problem

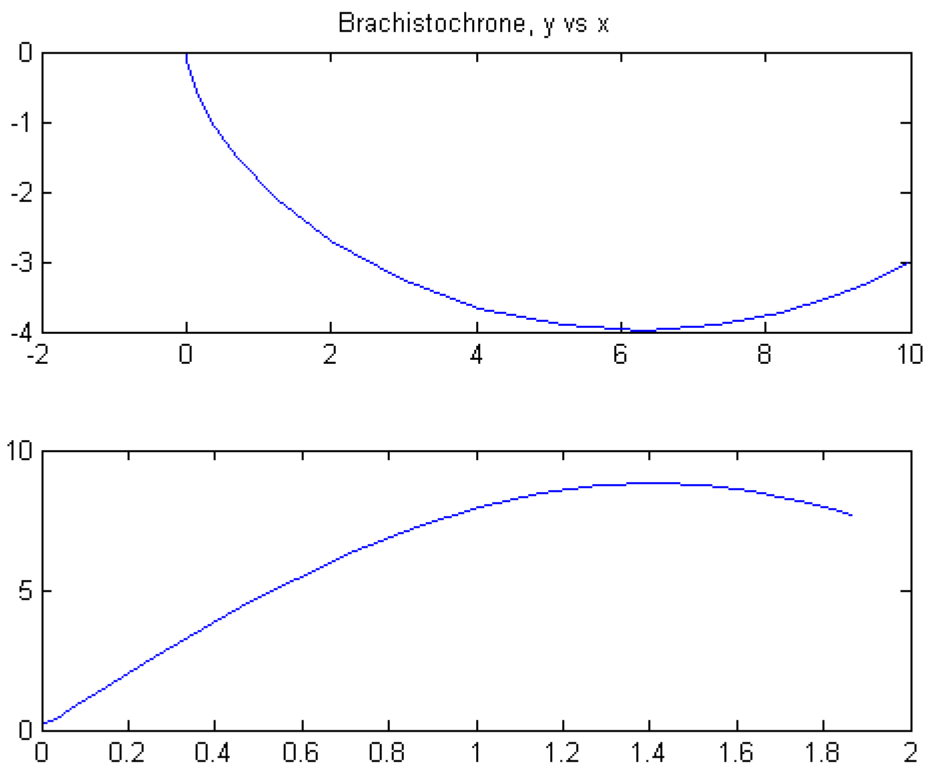
```
options = struct;
options.name = 'Brachistochrone-DAE';
solution = ezsolve(objective, {cbox, cbnd, ceq}, x0, options);
x  = subs(collocate(x),solution);
y  = subs(collocate(y),solution);
v  = subs(collocate(v),solution);
t  = subs(collocate(t),solution);
```

```
Problem type appears to be: lpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Brachistochrone-DAE            f_k       1.869963310229847400
                                      sum(|constr|)      0.000000000158881015
                             f(x_k) + sum(|constr|)      1.869963310388728300
                                             f(x_0)     10.000000000000000000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv  168 ConJacEv  168 Iter   93 MinorIter  154
CPU time: 0.203125 sec. Elapsed time: 0.219000 sec.
```

15.4 Plot the result

To obtain the brachistochrone curve, we plot y versus x.

```
subplot(2,1,1)
plot(x, y);
title('Brachistochrone, y vs x');

subplot(2,1,2)
plot(t, v);
```



16 Bridge Crane System

ITERATIVE DYNAMIC PROGRAMMING, REIN LUUS

12.4.1 Example 1: Bridge crane system

CHAPMAN & HALL/CRC Monographs and Surveys in Pure and Applied Mathematics

16.1 Problem description

Find u over t in [0; t_F] to minimize

$$J = t_F$$

subject to:

$$\frac{dx_1}{dt} = x_2$$
$$\frac{dx_2}{dt} = u$$
$$\frac{dx_3}{dt} = x_4$$
$$\frac{dx_4}{dt} = -0.98 * x_3 + 0.1 * u$$

The initial condition are:

$$x(0) = [0\ 0\ 0\ 0]$$
$$x(t_F) = [15\ 0\ 0\ 0]$$
$$-1 <= u <= 1$$

Reference: [25]

16.2 Problem setup

```
toms t
toms t_f

p = tomPhase('p', t, 0, t_f, 50);
setPhase(p);

tomStates x1 x2 x3 x4
tomControls u

% Initial guess
% Note: The guess for t_f must appear in the list before expression involving t.
x0 = {t_f == 8, ...
    collocate(u==1-2*t/t_f)};

% Box constraints
cbox = {0.1 <= t_f <= 100
    -1  <= collocate(u) <= 1};

% Boundary constraints
cbnd = {initial({x1 == 0; x2 == 0
    x3 == 0; x4 == 0})
    final({x1 == 15; x2 == 0
    x3 == 0; x4 == 0})};

% ODEs and path constraints
ceq = collocate({
    dot(x1) == x2
    dot(x2) == u
    dot(x3) == x4
    dot(x4) == -0.98*x3+0.1*u});

% Objective
objective = t_f;
```

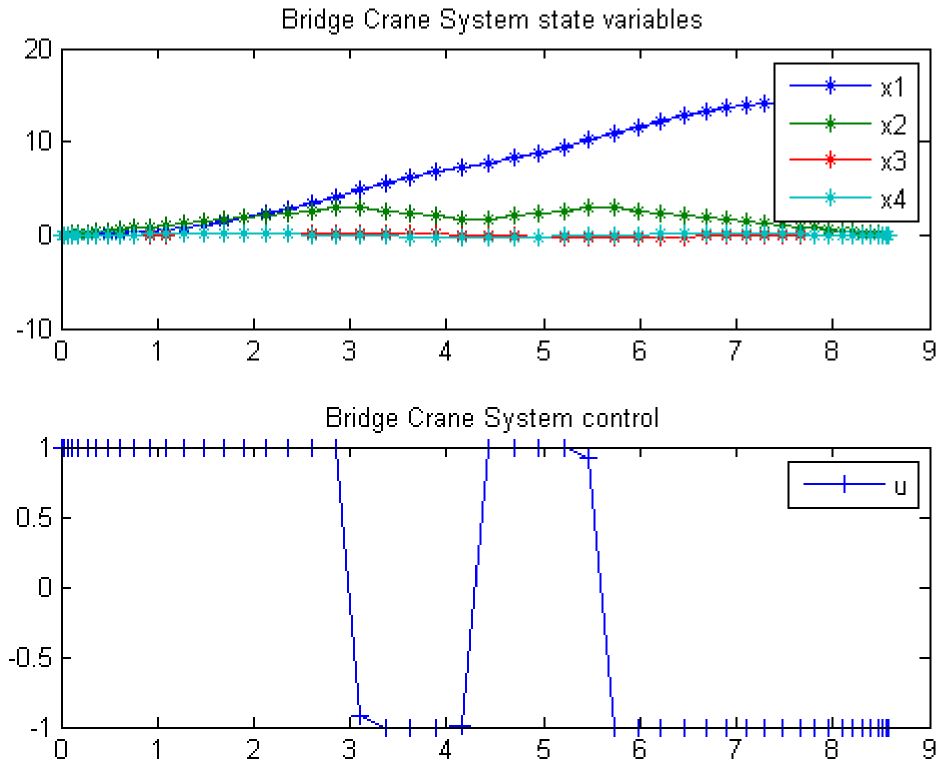
16.3 Solve the problem

```
options = struct;
options.name = 'Bridge Crane System';
solution = ezsolve(objective, {cbox, cbnd, ceq}, x0, options);
t  = subs(collocate(t),solution);
x1 = subs(collocate(x1),solution);
x2 = subs(collocate(x2),solution);
x3 = subs(collocate(x3),solution);
x4 = subs(collocate(x4),solution);
u  = subs(collocate(u),solution);
```

```
Problem type appears to be: lpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Bridge Crane System           f_k       8.578933610367178300
                                       sum(|constr|)      0.000000187955007808
                              f(x_k) + sum(|constr|)      8.578933798322186300
                                              f(x_0)      8.000000000000000000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv   37 ConJacEv   37 Iter   19 MinorIter  501
CPU time: 0.468750 sec. Elapsed time: 0.469000 sec.
```

16.4 Plot result

```
subplot(2,1,1)
plot(t,x1,'*-',t,x2,'*-',t,x3,'*-',t,x4,'*-');
legend('x1','x2','x3','x4');
title('Bridge Crane System state variables');

subplot(2,1,2)
plot(t,u,'+-');
legend('u');
title('Bridge Crane System control');
```


Bridge Crane System state variables
20
10
0
-10
0
1
2
3
4
5
6
7
8
9
x1
x2
x3
x4
Bridge Crane System control
1
0.5
0
-0.5
-1
0
1
2
3
4
5
6
7
8
9
u

17 Bryson-Denham Problem

17.1 Problem description

Standard formulation for Bryson-Denham

Reference: [9]

17.2 Problem setup

```
toms t t_f
p = tomPhase('p', t, 0, t_f, 50);
setPhase(p);

x10 = 0; x20 = 1;
x30 = 0; x1f = 0; x2f = -1;
x1min = -10;   x1max = 10;    x2min = x1min;
x2max = x1max; x3min = x1min; x3max = x1max;

tomStates x1 x2 x3
tomControls u

% Initial guess
x0 = {t_f == 0.5
    icollocate({
    x1 == x10+(x1f-x10)*t/t_f
    x2 == x20+(x2f-x20)*t/t_f
    x3 == x30
    })
    collocate(u==0)};

% Box constraints
cbox = {0.001 <= t_f <= 50
    0      <= mcollocate(x1) <= 1/9
    x2min  <= mcollocate(x2) <= x2max
    x3min  <= mcollocate(x3) <= x3max
    -5000  <= collocate(u)   <= 5000};

% Boundary constraints
cbnd = {initial({x1 == x10; x2 == x20; x3 == x30})
    final({x1 == x1f; x2 == x2f})};

% ODEs and path constraints
ceq = collocate({
```

```
    dot(x1) == x2
    dot(x2) == u
    dot(x3) == u.^2/2});

% Objective
objective = final(x3);
```

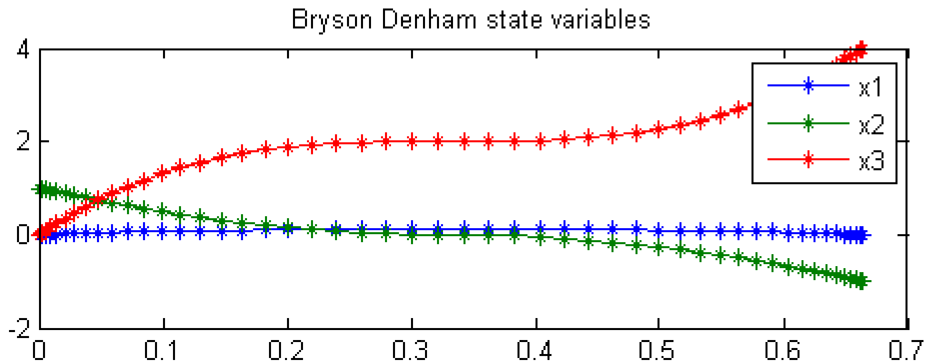
17.3 Solve the problem

```
options = struct;
options.name = 'Bryson Denham';
solution = ezsolve(objective, {cbox, cbnd, ceq}, x0, options);
t  = subs(collocate(t),solution);
x1 = subs(collocate(x1),solution);
x2 = subs(collocate(x2),solution);
x3 = subs(collocate(x3),solution);
u  = subs(collocate(u),solution);
```

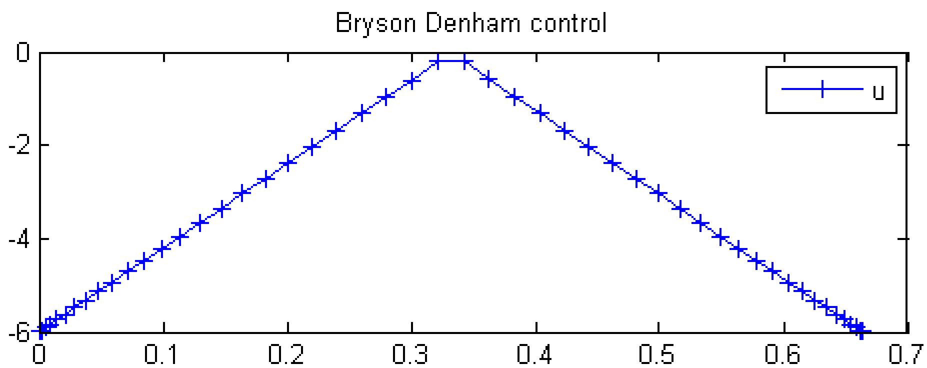
```
Problem type appears to be: lpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Bryson Denham                  f_k       4.000021208621198800
                                       sum(|constr|)      0.000000174861447145
                              f(x_k) + sum(|constr|)      4.000021383482645900
                                              f(x_0)      0.000000000000000000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv   63 ConJacEv   63 Iter   60 MinorIter  260
CPU time: 0.906250 sec. Elapsed time: 0.922000 sec.
```

17.4 Plot result

```
subplot(2,1,1)
plot(t,x1,'*-',t,x2,'*-',t,x3,'*-');
legend('x1','x2','x3');
title('Bryson Denham state variables');

subplot(2,1,2)
plot(t,u,'+-');
legend('u');
title('Bryson Denham control');
```



Bryson Denham state variables
x1
x2
x3
4
2
0
-2
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7



Bryson Denham control
u
0
-2
-4
-6
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7

18 Bryson-Denham Problem (Detailed)

18.1 Problem description

A detailed version of the Bryson-Denham problem

Reference: [9]

18.2 Define the independent variable, and phase:

It is common for the independent variable to be named "t", but any legal tomSym name is possible. The length of the time-interval is variable, so another tomSym symbol is created for that.

```
toms t t_f
p = tomPhase('p', t, 0, t_f, 30);
setPhase(p);
```

18.3 A few constants.

```
% The name on the mathematics:
options = struct;
options.name = 'Bryson Denham Detailed';

x1max = 1/9;
tfmax = 50;
```

18.4 Define a list of states

After a phase has been defined, states can be created Note that the states can be given meaningful names, even though they are simply named x1...x3 in this particular problem.

```
tomStates x1 x2 x3

% Initial guess
% The guess for t_f must appear first in the list
x0 = {t_f == 0.5
    icollocate({
    x1 == 0
    x2 == 1-2*t/t_f
    x3 == 0
    })};
```

```
cbox = {0 <= t_f <= 50
    -10 <= icollocate(x1) <= 10
    -10 <= icollocate(x2) <= 10
    -10 <= icollocate(x3) <= 10};
```

18.5 Adding the control variable and equations

```
tomControls u

x0 = {x0
    collocate(u == 0)};

cbox = {cbox
    -5000 <= collocate(u) <= 5000};
```

18.6 Equations

The equations are on a nonlinear DAE form, i.e. the states and their derivatives can be combined into arbitrary equations. Equations can also be specified point-wise, or using integrals. Integrals can be achieved by using the function integral(). Each equation must contain one or more equals (==) signs, or one or more greater than (>=) signs, or one or more less than (<=) signs.

```
usquared = u^2;

ceq = collocate({
    dot(x1) == x2
    dot(x2) == u
    dot(x3) == 0.5*usquared
    0 <= x1 <= x1max % Path constraint
    });

cbnd = {initial({
    x1 == 0; x2 == 1; x3 == 0})
    final({x1 == 0; x2 == -1})};

% The "objective" function to minimize. Cost can be specified at a point,
% or integrated over time by using the function integral.
% If costs are defined as many parts they should be added together.
objective = final(x3);
```

18.7 Build the .m files and general TOMLAB problem

```
Prob = sym2prob('con',objective,{cbox, ceq, cbnd},x0,options);

% Solve the problem using any TOMLAB solver
```

```
Result = tomRun('snopt', Prob, 1);
```

```
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Bryson Denham Detailed         f_k       3.998132387937467900
                                     sum(|constr|)      0.000000810449850953
                            f(x_k) + sum(|constr|)      3.998133198387318700
                                            f(x_0)      0.000000000000000000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv   35 GradEv   33 ConstrEv   33 ConJacEv   33 Iter   32 MinorIter  170
CPU time: 0.187500 sec. Elapsed time: 0.187000 sec.
```

19 Bryson-Denham Problem (Short version)

19.1 Problem description

The Bryson-Denham Problem but we take advantage of the propt input format to compute the cost function directly, without going via u and x3.

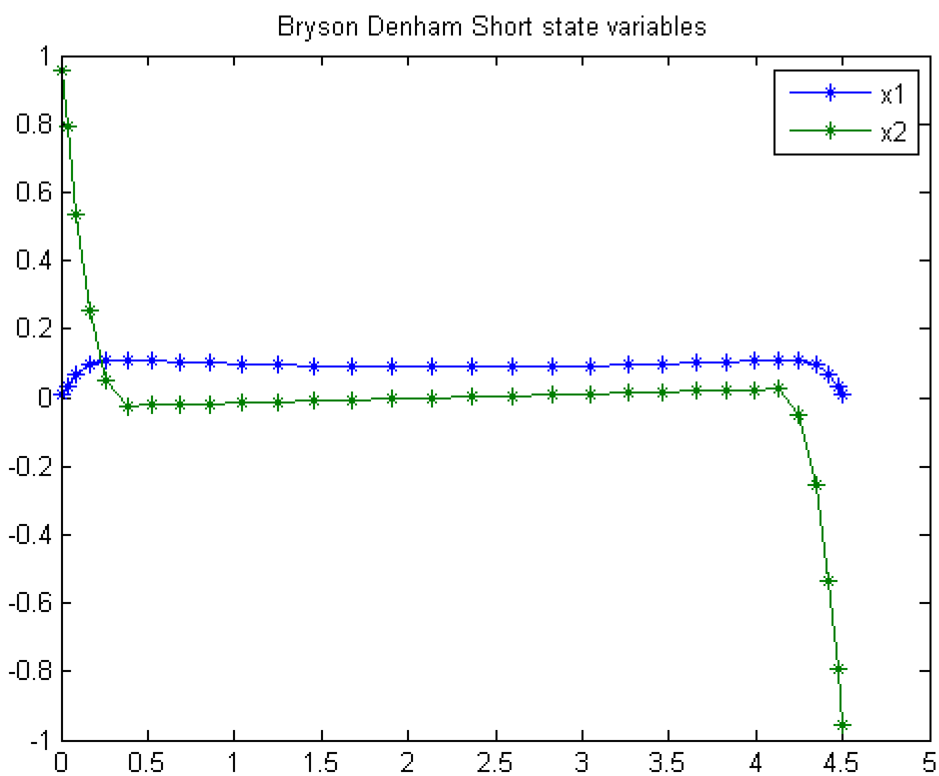
Reference: [9]

19.2 Problem setup

```
toms t t_f
p = tomPhase('p', t, 0, t_f, 30); setPhase(p);
tomStates x1 x2
x1max = 1/9; x0 = {t_f == 0.5};

constr = {0.001 <= t_f <= 50
    collocate({0 <= x1 <= x1max; -10 <= x2 <= 10})
    initial({x1 == 0; x2 == 1}); final({x1 == 0; x2 == -1})
    collocate(dot(x1) == x2)};

options = struct;
options.name = 'Bryson Denham Short';
solution = ezsolve(integrate(0.5*dot(x2).^2), constr, x0, options);
t  = subs(collocate(t),solution);
x1 = subs(collocate(x1),solution); x2 = subs(collocate(x2),solution);
figure(1)
plot(t,x1,'*-',t,x2,'*-');
legend('x1','x2');
title('Bryson Denham Short state variables');
```

```
Problem type appears to be: con
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Bryson Denham Short             f_k       3.975295744665008800
                                       sum(|constr|)      0.000000002213310492
                              f(x_k) + sum(|constr|)      3.975295746878319200
                                              f(x_0)   1859.999999999970900000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
```

```
Optimality conditions satisfied

FuncEv  139 GradEv  137 ConstrEv  137 ConJacEv  137 Iter  136 MinorIter  316
CPU time: 0.375000 sec. Elapsed time: 0.375000 sec.
```



20 Bryson-Denham Two Phase Problem

An example of how the input could look for PROPT.

20.1 Problem description

In this example we also take advantage of the advance knowledge that the solution reaches x1=x1max with x2=0, to introduce an event that divides the time interval into two phases. This increases the accuracy of the solution.

Reference: [9]

20.2 Problem setup

```
for n=[5 21]

    toms t1 tcut
    p1 = tomPhase('p1', t1, 0, tcut, n);
    toms t2 tmax
    p2 = tomPhase('p2', t2, tcut, tmax-tcut, n);

    setPhase(p1);
    tomStates x1p1 x2p1
    tomControls up1
    setPhase(p2);
    tomStates x1p2 x2p2
    tomControls up2

    % Constant
    x1max = 1/9;

    setPhase(p1);
    % Initial guess
    if n==5
        x01 = {tcut == 0.25
            tmax == 0.5
            icollocate({
            x1p1 == 0
            x2p1 == 1-2*t1/tcut
            })
            collocate(up1==0)};
    else
        x01 = {tcut == tcut_opt
            tmax == tmax_opt
            icollocate({
```

```
        x1p1 == x1p1_opt
        x2p1 == x2p1_opt
        })
        collocate(up1==up1_opt)};
end

% Box constraints
cbox1 = {0.001 <= tcut <= tmax-0.01
    tmax <= 50
    collocate({0 <= x1p1 <= x1max
    -10 <= x2p1 <= 10})};

% Set up initial conditions in phase p1
% Initial constraints
cbnd1 = initial({x1p1 == 0; x2p1 == 1});

% ODEs and path constraints
ceq1 = collocate({
    dot(x1p1) == x2p1
    dot(x2p1) == up1});

% We take advantage of the fact that we've determined that a good place to
% shift between phases is when x1 reaches x1max, and that x2 must equal 0
% there (Later, we want the solver to be able to figure this out for
% itself).
% Final constraints
cbnd1 = {cbnd1
    final({x1p1 == x1max; x2p1 == 0})};

% Using integral gives the integral over the phase of an expression -
% in this case 0.5 times the square of u.
% Objective
objective1 = integrate(0.5*up1.^2);

setPhase(p2);
% Initial guess
if n==5
    x02 = {icollocate({
        x1p2 == 0
        x2p2 == 1-2*t2/tmax
        })
        collocate(up2==0)};
else
    x02 = {icollocate({
        x1p2 == x1p2_opt
        x2p2 == x2p2_opt
        })
```

```
        collocate(up2==up2_opt)};
end

% Box constraints
cbox2 = collocate({0 <= x1p2 <= x1max
    -10 <= x2p2 <= 10});

% ODEs and path constraints
ceq2 = collocate({
    dot(x1p2) == x2p2
    dot(x2p2) == up2});

% x2_i of p2 is already linked to x2_f of p1, but linking it to a constant
% helps convergence.
% Final conditions in phase p2.
cbnd2 = {initial(x2p2 == 0)
    final(x1p2 == 0)
    final(x2p2 == -1)};

objective2 = integrate(0.5*up2.^2);

% Link the phases
link = {final(p1,x1p1) == initial(p2,x1p2)
    final(p1,x2p1) == initial(p2,x2p2)};
```

20.3 Solve the problem

```
options = struct;
options.name = 'Bryson Denham Two Phase';
objective = objective1+objective2;
constr = {cbox1, cbnd1, ceq1, cbox2, cbnd2, ceq2, link};
solution = ezsolve(objective, constr, {x01, x02}, options);
x1p1_opt = subs(x1p1, solution);
x2p1_opt = subs(x2p1, solution);
up1_opt  = subs(up1, solution);
x1p2_opt = subs(x1p2, solution);
x2p2_opt = subs(x2p2, solution);
up2_opt  = subs(up2, solution);
tcut_opt = subs(tcut, solution);
tmax_opt = subs(tmax, solution);
```

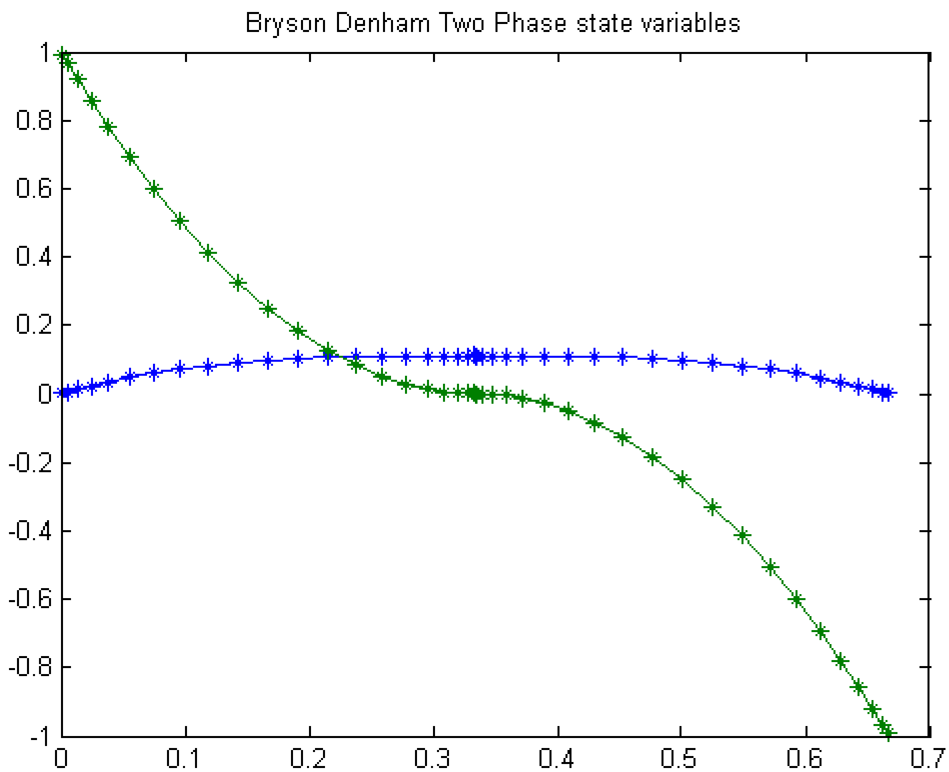
```
Problem type appears to be: con
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
```

```
Problem: ---  1: Bryson Denham Two Phase        f_k       3.999999993412843000
                                       sum(|constr|)      0.000000023184469332
                              f(x_k) + sum(|constr|)      4.000000016597312000
                                              f(x_0)      0.000000000000000000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv   39 GradEv   37 ConstrEv   37 ConJacEv   37 Iter   34 MinorIter   70
CPU time: 0.093750 sec. Elapsed time: 0.094000 sec.


Problem type appears to be: con
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Bryson Denham Two Phase        f_k       3.999999993239177900
                                       sum(|constr|)      0.000000001580905490
                              f(x_k) + sum(|constr|)      3.999999994820083500
                                              f(x_0)      3.999999738758653700

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    4 GradEv    2 ConstrEv    2 ConJacEv    2 Iter    1 MinorIter   76
CPU time: 0.046875 sec. Elapsed time: 0.047000 sec.
```

```
end

t = subs(collocate(p1,t1),solution);
t = [t;subs(collocate(p2,t2),solution)];
x1 = subs(collocate(p1,x1p1),solution);
x1 = [x1;subs(collocate(p2,x1p2),solution)];
x2 = subs(collocate(p1,x2p1),solution);
x2 = [x2;subs(collocate(p2,x2p2),solution)];
```

20.4 Plot the result

```
figure(1)
plot(t,x1,'*-',t,x2,'*-');
title('Bryson Denham Two Phase state variables');
```


Bryson Denham Two Phase state variables
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7

21 Bryson Maxrange

21.1 Problem description

Max range version of Bryson-Denham problem

Reference: [9]

21.2 Problem setup

```
toms t
p = tomPhase('p', t, 0, 2, 50);
setPhase(p);

tomStates x y v
tomControls u1 u2

% Various constants and expressions
xmin = -10; xmax = 10;
ymin = xmin; ymax = xmax;
Vmin = -100; Vmax = 100;
g = 1;
a = 0.5*g;

% Initial guess
x0 = collocate({u1 == 1; u2 == 0});

% Box constraints
cbox = {xmin <= icollocate(x) <= xmax
    ymin <= icollocate(y) <= ymax
    Vmin <= icollocate(v) <= Vmax
    -100 <= collocate(u1) <= 100
    -100 <= collocate(u2) <= 100};

% Boundary constraints
cbnd = {initial({x == 0; y == 0; v == 0})
    final(y == 0.1)};

% ODEs and path constraints
ceq = {collocate({
    dot(x) == v.*u1
    dot(y) == v.*u2
    dot(v) == a-g*u2
    })
```

```
    collocate(u1.^2+u2.^2 == 1)};

% Objective
objective = -final(x);
```

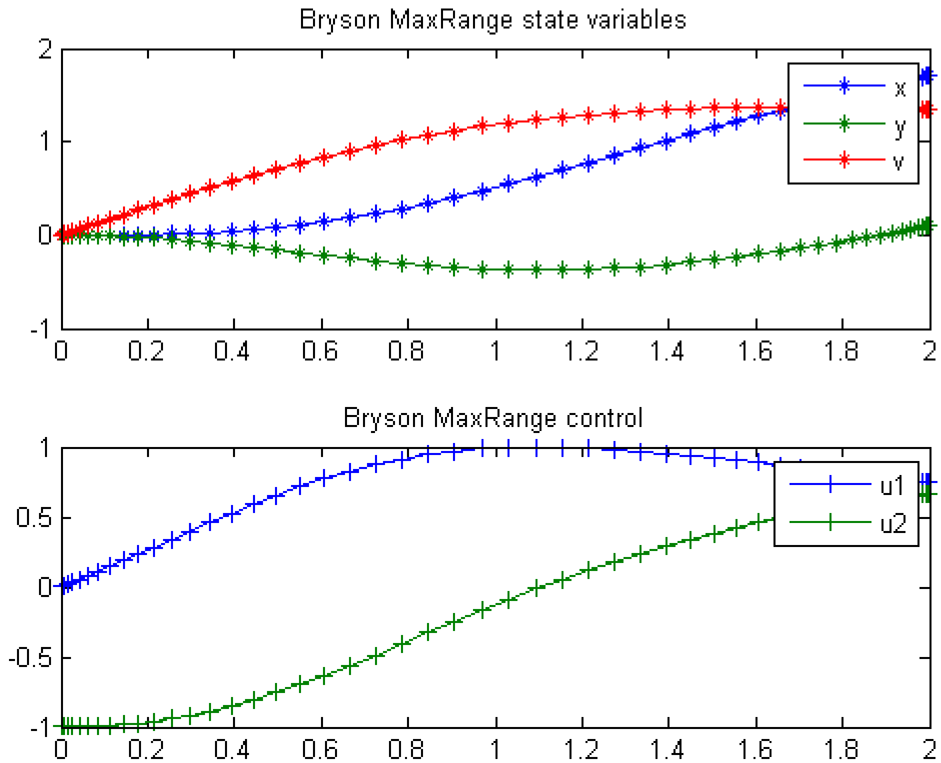
21.3 Solve the problem

```
options = struct;
options.name = 'Bryson MaxRange';
solution = ezsolve(objective, {cbox, cbnd, ceq}, x0, options);
t  = subs(collocate(t),solution);
x  = subs(collocate(x),solution);
y  = subs(collocate(y),solution);
v  = subs(collocate(v),solution);
u1 = subs(collocate(u1),solution);
u2 = subs(collocate(u2),solution);
```

```
Problem type appears to be: lpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Bryson MaxRange                f_k      -1.712314875015309000
                                       sum(|constr|)      0.000000100745600920
                              f(x_k) + sum(|constr|)     -1.712314774269708000
                                              f(x_0)      0.000000000000000000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv   55 ConJacEv   55 Iter   34 MinorIter  185
CPU time: 0.515625 sec. Elapsed time: 0.515000 sec.
```

21.4 Plot result

```
subplot(2,1,1)
plot(t,x,'*-',t,y,'*-',t,v,'*-');
legend('x','y','v');
title('Bryson MaxRange state variables');

subplot(2,1,2)
plot(t,u1,'+-',t,u2,'+-');
legend('u1','u2');
title('Bryson MaxRange control');
```



Bryson MaxRange state variables
2
1
0
-1
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
x
y
v
Bryson MaxRange control
1
0.5
0
-0.5
-1
u1
u2

22 Catalyst Mixing

Second-order sensitivities of general dynamic systems with application to optimal control problems. 1999, Vassilios S. Vassiliadis, Eva Balsa Canto, Julio R. Banga

Case Study 6.2: Catalyst mixing

22.1 Problem formulation

This problem considers a plug-flow reactor, packed with two catalysts, involving the reactions

```
S1 <-> S2 -> S3
```

The optimal mixing policy of the two catalysts has to be determined in order to maximize the production of species S3. This dynamic optimization problem was originally proposed by Gunn and Thomas (1965), and subsequently considered by Logsdon (1990) and Vassiliadis (1993). The mathematical formulation is

Maximize:

$$J = 1 - x_1(t_f) - x_2(t_f)$$

subject to:

$$\frac{dx_1}{dt} = u * (10 * x_2 - x_1)$$
$$\frac{dx_2}{dt} = u * (x_1 - 10 * x_2) - (1 - u) * x_2$$
$$0 <= u <= 1$$
$$x(t_0) = [1\ 0]'$$
$$t_f = 1$$

Reference: [31]

22.2 Problem setup

```
toms t
p = tomPhase('p', t, 0, 1, 30);
```

```
setPhase(p);

tomStates x1 x2
tomControls u

% Initial guess
% Note: The guess for t_f must appear in the list before expression involving t.
x0 = {icollocate({
    x1 == 1-0.085*t
    x2 == 0.05*t
    })
    collocate(u==1-t)};

% Box constraints
cbox = {0.9 <= icollocate(x1) <= 1
    0 <= icollocate(x2) <= 0.1
    0 <= collocate(u)   <= 1};

% Boundary constraints
cbnd = {initial({x1 == 1; x2 == 0})
    final({x1 <= 0.95})};

% ODEs and path constraints
ceq = collocate({
    dot(x1) == u.*(10*x2-x1)
    dot(x2) == u.*(x1-10*x2)-(1-u).*x2});

% Objective
objective = -1+final(x1)+final(x2);
```

22.3 Solve the problem

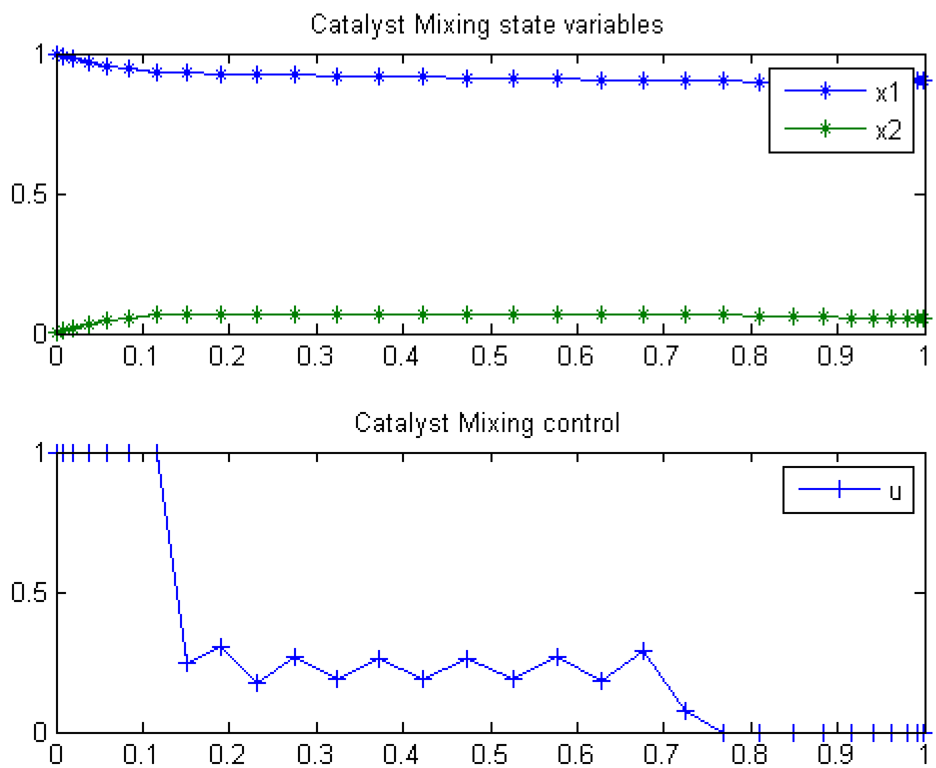
```
options = struct;
options.name = 'Catalyst Mixing';
solution = ezsolve(objective, {cbox, cbnd, ceq}, x0, options);
t  = subs(collocate(t),solution);
x1 = subs(collocate(x1),solution);
x2 = subs(collocate(x2),solution);
u  = subs(collocate(u),solution);
```

```
Problem type appears to be: lpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Catalyst Mixing                f_k      -0.048059280695325390
```

```
                 sum(|constr|)      0.000000452031812690
        f(x_k) + sum(|constr|)     -0.048058828663512701
                        f(x_0)      0.964999999999998080

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv   66 ConJacEv   66 Iter   43 MinorIter  248
CPU time: 0.171875 sec. Elapsed time: 0.188000 sec.
```

22.4 Plot result

```
subplot(2,1,1)
plot(t,x1,'*-',t,x2,'*-');
legend('x1','x2');
title('Catalyst Mixing state variables');

subplot(2,1,2)
plot(t,u,'+-');
legend('u');
title('Catalyst Mixing control');
```


Catalyst Mixing state variables
x1
x2
Catalyst Mixing control
u

23 Catalytic Cracking of Gas Oil

Benchmarking Optimization Software with COPS Elizabeth D. Dolan and Jorge J. More ARGONNE NATIONAL LABORATORY

23.1 Problem Formulation

Find theta over t in [0; 0.95] to minimize

$$J = \sum_{j=1}^{2} \sum_{i=1}^{21} (y_{j,i} - y_{j,i,meas})^2$$

subject to:

$$\frac{dy_1}{dt} = -(theta_1 + theta_3) * y_1^2$$
$$\frac{dy_2}{dt} = theta_1 * y_1^2 - theta_2 * y_2$$
$$theta >= 0$$

Where the data is given in the code.

Reference: [14]

23.2 Problem setup

```
toms t theta1 theta2 theta3
p = tomPhase('p', t, 0, 0.95, 100);
setPhase(p);

tomStates y1 y2

% Initial guess
x0 = icollocate({
    y1 == 1-(1-0.069)*t/0.95
    y2 == 0.01*t/0.95});

% Box constraints
cbox = {0 <= theta1; 0 <= theta2; 0 <= theta3};
```

```
% Various constants and expressions
y1meas = [1.0;0.8105;0.6208;0.5258;0.4345;0.3903;...
    0.3342;0.3034;0.2735;0.2405;0.2283;0.2071;0.1669;...
    0.153;0.1339;0.1265;0.12;0.099;0.087;0.077;0.069];
y2meas = [0;0.2;0.2886;0.301;0.3215;0.3123;0.2716;...
    0.2551;0.2258;0.1959;0.1789;0.1457;0.1198;0.0909...
    ;0.0719;0.0561;0.046;0.028;0.019;0.014;0.010];
tmeas = [0;0.025;0.05;0.075;0.1;0.125;...
    0.15;0.175;0.2;0.225;0.25;0.3;0.35;0.4;...
    0.45;0.5;0.55;0.65;0.75;0.85;0.95];

y1err = atPoints(tmeas,y1) - y1meas;
y2err = atPoints(tmeas,y2) - y2meas;

% ODEs and path constraints
ceq = collocate({
    dot(y1) == -(theta1+theta3)*y1.^2
    dot(y2) == theta1*y1.^2-theta2*y2});

% Objective
objective = sum(y1err.^2)+sum(y2err.^2);
```

23.3 Solve the problem

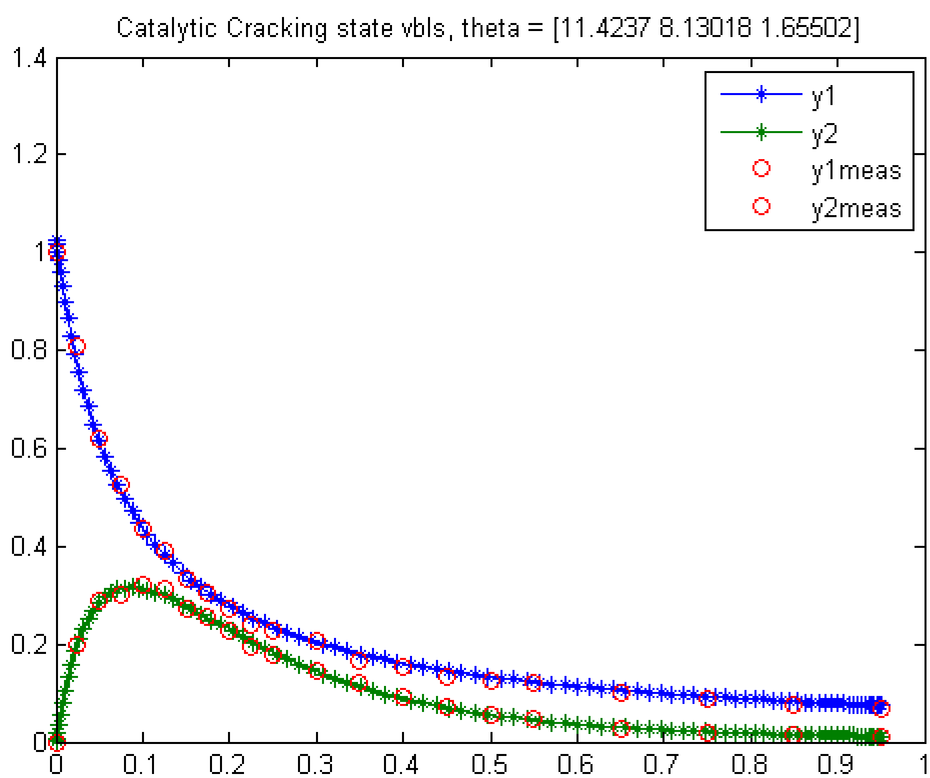
```
options = struct;
options.name = 'Catalytic Cracking';
solution = ezsolve(objective, {cbox, ceq}, x0, options);
t  = subs(collocate(t),solution);
y1 = subs(collocate(y1),solution);
y2 = subs(collocate(y2),solution);
theta1 = subs(theta1,solution);
theta2 = subs(theta2,solution);
theta3 = subs(theta3,solution);
```

```
Problem type appears to be: qpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Catalytic Cracking             f_k       0.004326020490940330
                                       sum(|constr|)      0.000000000508878939
                              f(x_k) + sum(|constr|)      0.004326020999819269
                                              f(x_0)      0.165642328947365690

Solver: snopt.  EXIT=0.  INFORM=1.
```

```
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv   38 ConJacEv   38 Iter   29 MinorIter  235
CPU time: 0.734375 sec. Elapsed time: 0.735000 sec.
```

23.4 Plot result

```
figure(1);
tm  = tmeas;  y1m = y1meas; y2m = y2meas;
t1  = theta1; t2  = theta2; t3  = theta3;
plot(t,y1,'*-',t,y2,'*-',tm,y1m,'ro',tm,y2m,'ro');
legend('y1','y2','y1meas','y2meas');
title(sprintf('Catalytic Cracking state vbls, theta = [%g %g %g]',t1,t2,t3));
```



24 Flow in a Channel

Benchmarking Optimization Software with COPS Elizabeth D. Dolan and Jorge J. More ARGONNE NATIONAL LABORATORY

24.1 Problem Formulation

Find u(t) over t in [0; 1] to minimize

$$J = 0$$

subject to:

$$\frac{d^4u}{dt^4} = R * (\frac{du}{dt} * \frac{d^2u}{dt^2} - u * \frac{d^3u}{dt^3})$$

$$u_0 = 0$$
$$u_1 = 1$$
$$\frac{du}{dt}_0 = 0$$
$$\frac{du}{dt}_1 = 0$$
$$R = 10$$

After some transformation we get this problem:

$$\frac{dx_1}{dt} = x_2$$
$$\frac{dx_2}{dt} = x_3$$
$$\frac{dx_3}{dt} = x_4$$
$$\frac{dx_4}{dt} = R * (x_2 * x_3 - x_1 * x_4)$$
$$x_1(0) = 0$$

$$x_1(1) = 1$$
$$x_2(0) = 0$$
$$x_2(1) = 0$$

Reference: [14]

24.2 Problem setup

```
toms t
p = tomPhase('p', t, 0, 1, 30);
setPhase(p);

tomStates x1 x2 x3 x4

x0 = icollocate({x1 == 3*t.^2 - 2*t.^3
    x2 == 2*t - 6*t.^2
    x3 == t - 12*t
    x4 == -12});

% Boundary constraints
cbnd = {initial({x1 == 0; x2 == 0})
    final({x1 == 1; x2 == 0})};

% Various constants and expressions
R = 10;

% ODEs and path constraints
ceq = collocate({dot(x1) == x2
    dot(x2) == x3; dot(x3) == x4
    dot(x4) == R*(x2.*x3-x1.*x4)});

% Objective
objective = 1; %(feasibility problem)
```

24.3 Solve the problem

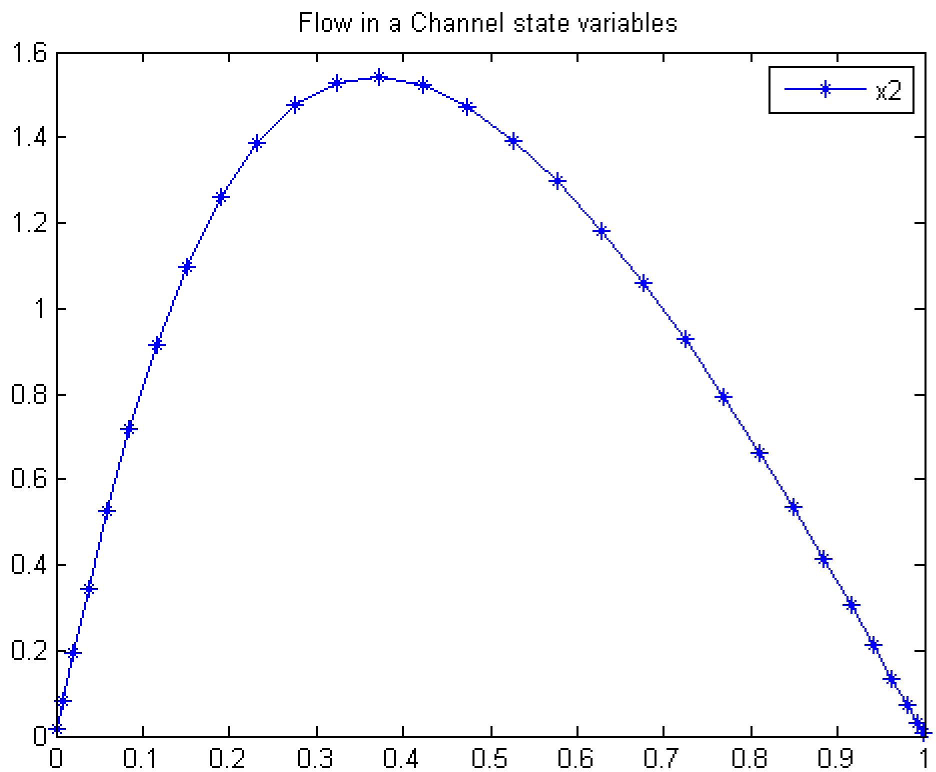
```
options = struct;
options.name = 'Flow in a Channel Steering';
solution = ezsolve(objective, {cbnd, ceq}, x0, options);

% Extract optimal states and controls from solution
t  = subs(collocate(t),solution);
x1 = subs(collocate(x1),solution);
x2 = subs(collocate(x2),solution);
```

```
x3 = subs(collocate(x3),solution);
x4 = subs(collocate(x4),solution);
```

```
Problem type appears to be: lpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Flow in a Channel Steering     f_k       1.000000000000000000
                                       sum(|constr|)      0.000000000018584877
                              f(x_k) + sum(|constr|)      1.000000000018584900
                                              f(x_0)      0.000000000000000000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv   11 ConJacEv   11 Iter    9 MinorIter   91
CPU time: 0.062500 sec. Elapsed time: 0.078000 sec.
```

24.4 Plot result

```
figure(1)
plot(t,x2,'*-');
legend('x2');
title('Flow in a Channel state variables');
```


Flow in a Channel state variables
x2
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

25 Coloumb Friction 1

Minimum-Time Control of Systems With Coloumb Friction: Near Global Optima Via Mixed Integer Linear Programming, Brian J. Driessen, Structural Dynamics Department, Sandia National Labs.

4. Numerical Examples

25.1 Problem Formulation

Find u over t in [0; t_F] to minimize

$$J = t_f$$

subject to:

$$\frac{d^2q}{dt^2} = u - sign(\frac{dq}{dt})$$
$$-2 <= u <= 2$$
$$q_0 = 0$$
$$\frac{dq}{dt}_0 = 1$$
$$q_2 = -1$$
$$\frac{dq}{dt}_2 = 0$$

Reference: [15]

25.2 Problem setup

```
toms t
toms t_f

p = tomPhase('p', t, 0, t_f, 60);
setPhase(p);

tomStates q qdot
tomControls u
```

```
% Initial guess
x0 = {t_f == 1, icollocate(q == -t)};

% Box constraints
cbox = {-2 <= collocate(u) <= 2
    0.001 <= t_f};

% Boundary constraints
cbnd = {initial({q == 0; qdot == 1}), final({q == -1, qdot == 0})};

% ODEs and path constraints
ceq = collocate({
    dot(q)    == qdot
    dot(qdot) == u-sign(qdot)});

objective = t_f;
```

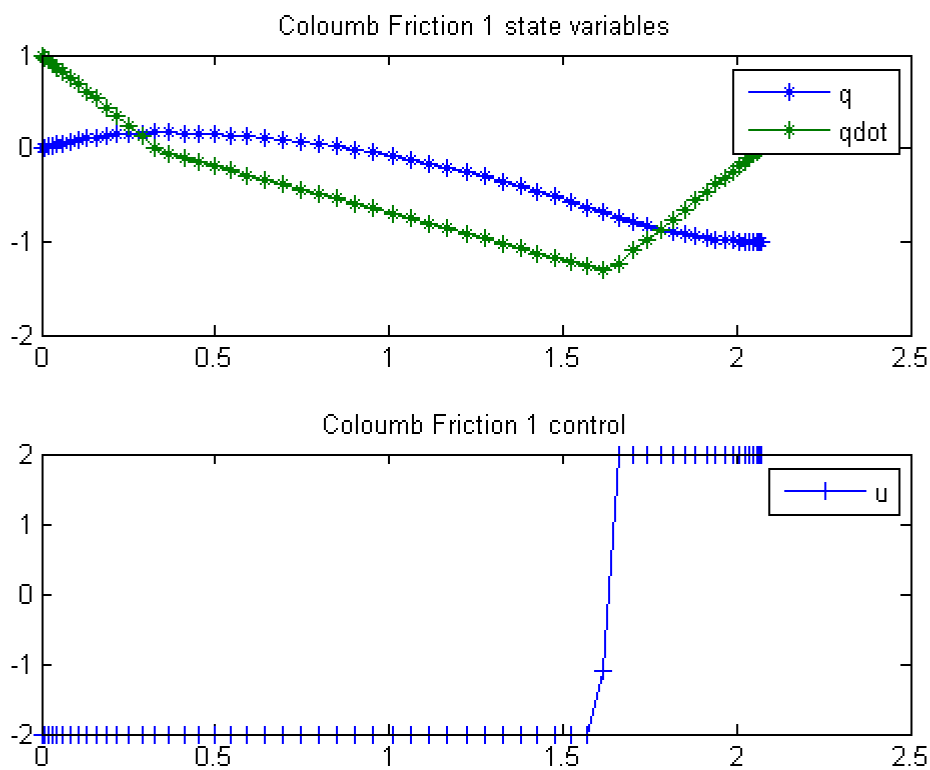
25.3 Solve the problem

```
options = struct;
options.name = 'Coloumb Friction 1';
constr = {cbox, cbnd, ceq};
solution = ezsolve(objective, constr, x0, options);

t    = subs(collocate(p,t),solution);
q    = subs(collocate(p,q),solution);
qdot = subs(collocate(p,qdot),solution);
u    = subs(collocate(p,u),solution);
```

```
Problem type appears to be: lpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Coloumb Friction 1             f_k       2.070229757012032500
                                       sum(|constr|)      0.000000001043060926
                              f(x_k) + sum(|constr|)      2.070229758055093200
                                              f(x_0)      1.000000000000000000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv  130 ConJacEv  130 Iter   23 MinorIter  676
CPU time: 0.609375 sec. Elapsed time: 0.641000 sec.
```

25.4 Plot result

```
subplot(2,1,1)
plot(t,q,'*-',t,qdot,'*-');
legend('q','qdot');
title('Coloumb Friction 1 state variables');

subplot(2,1,2)
plot(t,u,'+-');
legend('u');
title('Coloumb Friction 1 control');
```



26 Coloumb Friction 2

Minimum-Time Control of Systems With Coloumb Friction: Near Global Optima Via Mixed Integer Linear Programming, Brian J. Driessen, Structural Dynamics Department, Sandia National Labs.

4. Numerical Examples

26.1 Problem Formulation

Find u over t in [0; t_F] to minimize

$$J = t_f$$

subject to:

$$m_1 * \frac{d^2 q_1}{dt^2} = (-k_1 - k_2) * q_1 + k_2 * q_2 - mmu * sign(\frac{dq_1}{dt}) + u_1$$
$$m_2 * \frac{d^2 q_2}{dt^2} = k_2 * q_1 - k_2 * q_2 - mmu * sign(\frac{dq_2}{dt}) + u_2$$
$$q_{1:2}(0) = [0\ 0]$$
$$\frac{dq_{1:2}}{dt}_0 = [-1\ \ -2]$$
$$q_{1:2}(t_f) = [1\ 2]$$
$$\frac{dq_{1:2}}{dt}_{t_f} = [0\ 0]$$
$$-4 <= u_{1:2} <= 4$$
$$k_{1:2} = [0.95\ 0.85]$$
$$m_{1:2} = [1.1\ 1.2]$$
$$mmu = 1.0$$

Reference: [15]

26.2 Problem setup

```
toms t
toms t_f

p = tomPhase('p', t, 0, t_f, 40, [], 'gauss');
setPhase(p);

tomStates q1 q1dot q2 q2dot
tomControls u1 u2

% Initial guess
x0 = {t_f == 1};

% Box constraints
cbox = {1.8 <= t_f <= 4
    -4  <= collocate(u1) <= 4
    -4  <= collocate(u2) <= 4};

% Boundary constraints
cbnd = {initial({q1 == 0; q1dot == -1
    q2 == 0; q2dot == -2})
    final({q1 == 1; q1dot == 0
    q2 == 2; q2dot == 0})};

k1 = 0.95; k2 = 0.85;
m1 = 1.1;  m2 = 1.2;
mmu = 1;

% ODEs and path constraints
ceq = collocate({dot(q1) == q1dot
    m1*dot(q1dot) == (-k1-k2)*q1+k2*q2-mmu*sign(q1dot)+u1
    dot(q2)       == q2dot
    m2*dot(q2dot) == k2*q1-k2*q2-mmu*sign(q2dot)+u2});

% Objective
objective = t_f;
```

26.3 Solve the problem

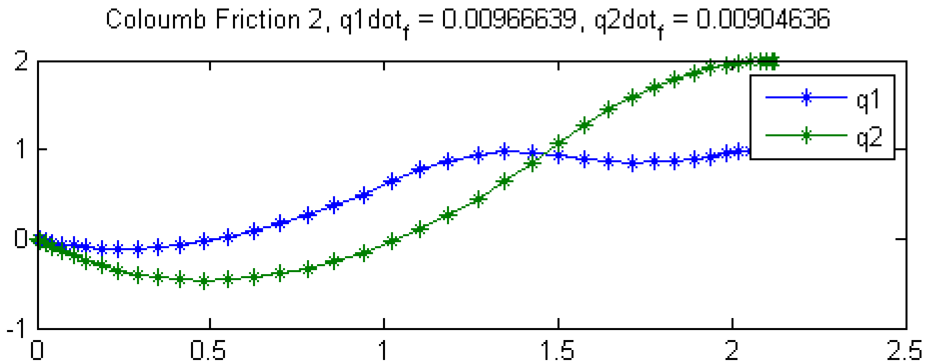
```
options = struct;
options.name = 'Coloumb Friction 2';
solution = ezsolve(objective, {cbox, cbnd, ceq}, x0, options);
t     = subs(collocate(t),solution);
q1    = subs(collocate(q1),solution);
q2    = subs(collocate(q2),solution);
q1dot = subs(collocate(q1dot),solution);
```

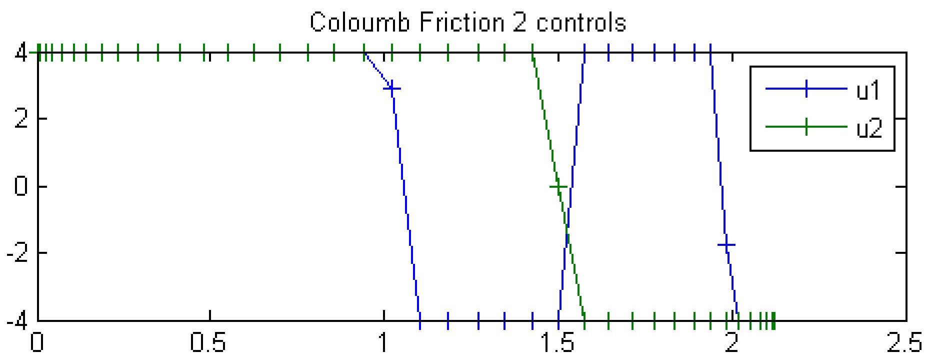
```
q2dot = subs(collocate(q2dot),solution);
u1    = subs(collocate(u1),solution);
u2    = subs(collocate(u2),solution);
q1dot_f = q1dot(end);
q2dot_f = q2dot(end);
```

```
Problem type appears to be: lpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Coloumb Friction 2             f_k       2.125397251986161700
                                        sum(|constr|)      0.000006640472891742
                               f(x_k) + sum(|constr|)      2.125403892459053300
                                               f(x_0)      1.800000000000000000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv   33 ConJacEv   32 Iter   27 MinorIter  388
CPU time: 0.531250 sec. Elapsed time: 0.531000 sec.
```

26.4 Plot result

```
subplot(2,1,1)
plot(t,q1,'*-',t,q2,'*-');
legend('q1','q2');
title(sprintf('Coloumb Friction 2, q1dot_f = %g, q2dot_f = %g',q1dot_f,q2dot_f));

subplot(2,1,2)
plot(t,u1,'+-',t,u2,'+-');
legend('u1','u2');
title('Coloumb Friction 2 controls');
```


Coloumb Friction 2, q1dot$_f$ = 0.00966639, q2dot$_f$ = 0.00904636
q1
q2
2
1
0
-1
0
0.5
1
1.5
2
2.5


Coloumb Friction 2 controls
u1
u2
4
2
0
-2
-4
0
0.5
1
1.5
2
2.5

27 Continuous State Constraint Problem

Problem 2: Miser3 manual

27.1 Problem description

Find u(t) over t in [0; 1] to minimize

$$J = \int_0^1 x_1(t)^2 + x_2(t)^2 + 0.005 * u(t)^2 \mathrm{d}t$$

subject to:

$$\frac{dx_1}{dt} = x_2$$

$$\frac{dx_2}{dt} = -x_2 + u$$

$$x_1(0) = 0$$

$$x_2(0) = -1$$

$$8 * (t - 0.5)^2 - 0.5 - x_2 >= 0$$

Reference: [19]

27.2 Problem setup

```
toms t
p = tomPhase('p', t, 0, 1, 50);
setPhase(p);

tomStates x1 x2
tomControls u

% Initial guess
x0 = {icollocate({x1 == 0; x2 == -1})
    collocate(u==0)};

% Box constraints
cbox = {-10 <= icollocate(x1) <= 10
    -10 <= icollocate(x2) <= 10
```

```
    -20 <= collocate(u)  <= 20};

% Boundary constraints
cbnd = initial({x1 == 0; x2 == -1});

% ODEs and path constraints
ceq = collocate({
    dot(x1) == x2
    dot(x2) == -x2+u
    8*(t-0.5).^2-0.5-x2 >= 0 % Path constr.
    });

% Objective
objective = integrate(x1.^2 + x2.^2 + 0.005*u.^2);
```

27.3 Solve the problem

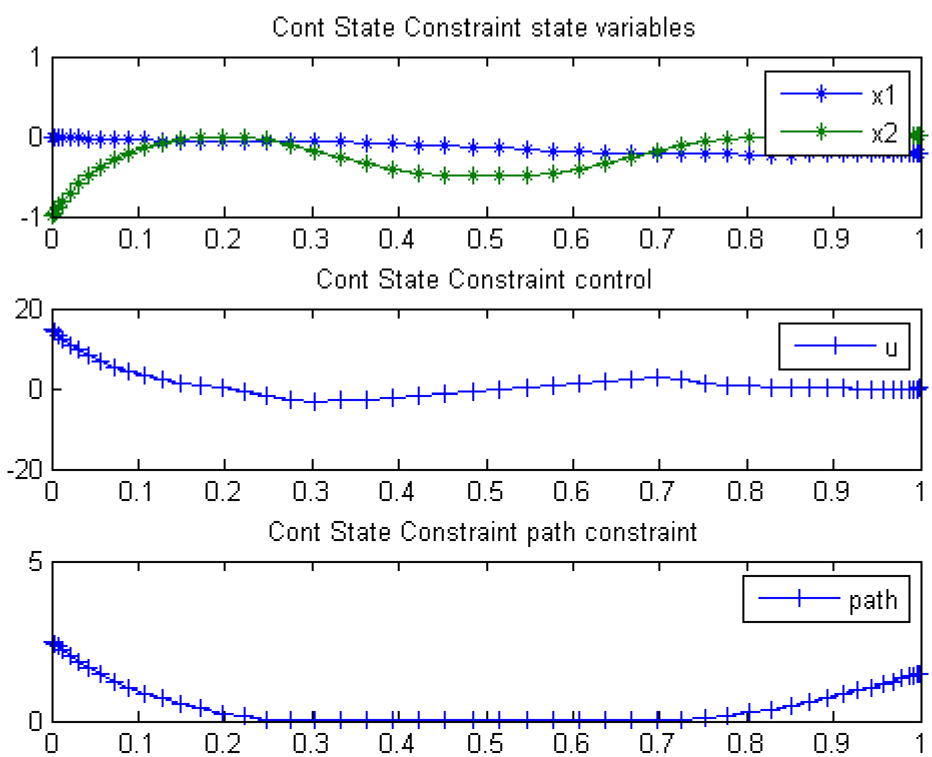
```
options = struct;
options.name = 'Cont State Constraint';
solution = ezsolve(objective, {cbox, cbnd, ceq}, x0, options);
t  = subs(collocate(t),solution);
x1 = subs(collocate(x1),solution);
x2 = subs(collocate(x2),solution);
u  = subs(collocate(u),solution);
```

```
Problem type appears to be: qp
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem:  1: Cont State Constraint              f_k       0.169824305998486440
                                       sum(|constr|)      0.000000000079892583
                              f(x_k) + sum(|constr|)      0.169824306078379030
                                              f(x_0)      0.000000000000000000

Solver: CPLEX.  EXIT=0.  INFORM=1.
CPLEX Barrier QP solver
Optimal solution found

FuncEv   10 GradEv   10 ConstrEv   10 Iter   10
CPU time: 0.531250 sec. Elapsed time: 0.532000 sec.
```

27.4 Plot result

```
subplot(3,1,1)
```

```
plot(t,x1,'*-',t,x2,'*-');
legend('x1','x2');
title('Cont State Constraint state variables');

subplot(3,1,2)
plot(t,u,'+-');
legend('u');
title('Cont State Constraint control');

subplot(3,1,3)
ieq = 8*(t-0.5).^2-0.5-x2;
plot(t,ieq,'+-');
axis([0 1 0 5]);
legend('path');
title('Cont State Constraint path constraint');
```



28 Curve Area Maximization

On smooth optimal control determination, Ilya Ioslovich and Per-Olof Gutman, Technion, Israel Institute of Technology.

Example 3: Maximal area under a curve of given length

28.1 Problem Description

Find u over t in [0; 1] to minimize:

$$J = \int_0^1 x_1 \mathrm{d}t$$

subject to:

$$\frac{dx_1}{dt} = u$$
$$\frac{dx_2}{dt} = \sqrt{1+u^2}$$
$$x(t_0) = [0\ 0]$$
$$x(t_f) = [0\ \frac{pi}{3}]$$

Reference: [18]

28.2 Problem setup

```
toms t
p = tomPhase('p', t, 0, 1, 20);
setPhase(p);

tomStates x1 x2
tomControls u

x0 = {icollocate({x1 == 0.1, x2 == t*pi/3}), collocate(u==0.5-t)};

% Boundary constraints
cbnd = {initial({x1 == 0; x2 == 0})
    final({x1 == 0; x2 == pi/3})};
```

```
% ODEs and path constraints
ceq = collocate({dot(x1) == u
    dot(x2) == sqrt(1+u.^2)});

% Objective
objective = -integrate(x1);
```

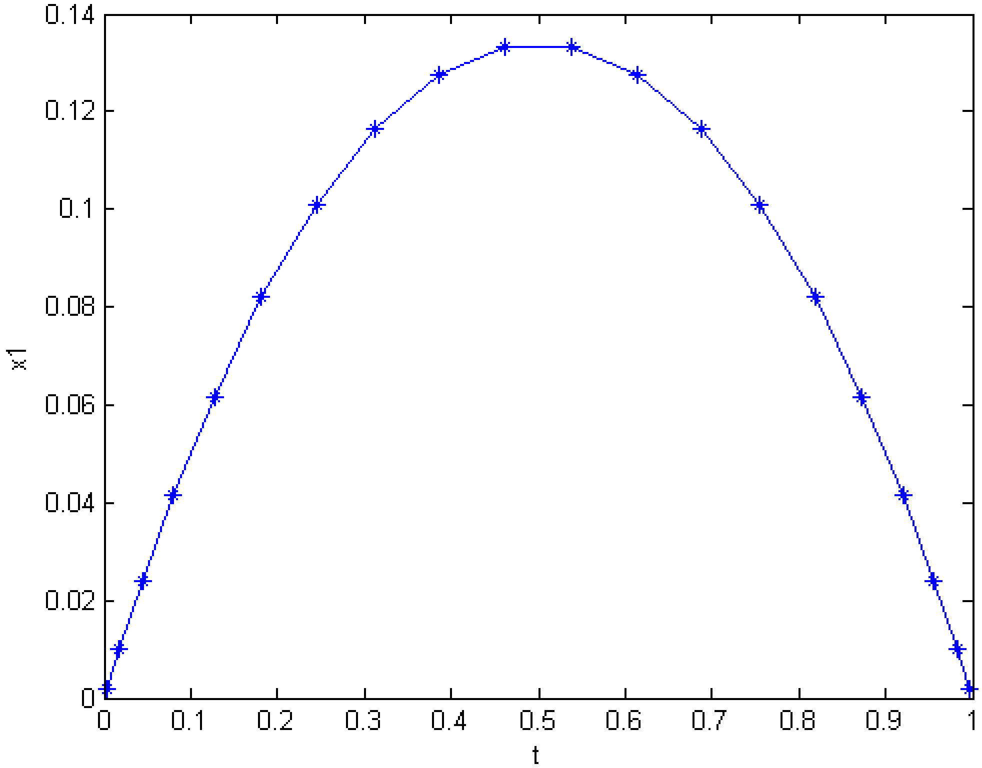
28.3 Solve the problem

```
options = struct;
options.name = 'Curve Area Maximization';
solution = ezsolve(objective, {cbnd, ceq}, x0, options);
```

```
Problem type appears to be: lpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Curve Area Maximization        f_k      -0.090586073472539108
                                       sum(|constr|)      0.000000003581094695
                              f(x_k) + sum(|constr|)     -0.090586069891444410
                                              f(x_0)     -0.099999999999999756

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv  120 ConJacEv  120 Iter   99 MinorIter  137
CPU time: 0.171875 sec. Elapsed time: 0.188000 sec.
```

28.4 Plot result

```
t = subs(collocate(t),solution);
x1 = subs(collocate(x1),solution);
figure(1);
plot(t,x1,'*-');
xlabel('t')
ylabel('x1')
```



0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
x1
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
t

29 Denbigh's System of Reactions

Dynamic Optimization of Batch Reactors Using Adaptive Stochastic Algorithms 1997, Eugenio F. Carrasco, Julio R. Banga

Case Study I: Denbigh's System of Reactions

29.1 Problem description

This optimal control problem is based on the system of chemical reactions initially considered by Denbigh (1958), which was also studied by Aris (1960) and more recently by Luus (1994):

```
A + B -> X

A + X -> P

X -> Y

X -> Q
```

where X is an intermediate, Y is the desired product, and P and Q are waste products. This system is described by the following differential equations:

$$\frac{dx_1}{dt} = -k_1 * x_1 - k_2 * x_1$$
$$\frac{dx_2}{dt} = k_1 * x_1 - k_3 + k_4 * x_2$$
$$\frac{dx_3}{dt} = k_3 * x_2$$

where x1 = [A][B], x2 = [X] and x3 = [Y]. The initial condition is

$$x(t_0) = [1\ 0\ 0]'$$

The rate constants are given by

$$k_i = k_{i0} * exp(-\frac{E_i}{R * T}), i = 1, 2, 3, 4$$

where the values of ki0 and Ei are given by Luus (1994).

The optimal control problem is to find T(t) (the temperature of the reactor as a function of time) so that the yield of Y is maximized at the end of the given batch time t_f. Therefore, the performance index to be maximized is

$$J = x_3(t_f)$$

where the batch time t_f is specified as 1000 s. The constraints on the control variable (reactor temperature) are

$$273 <= T <= 415$$

Reference: [10]

29.2 Problem setup

```
toms t
```

29.3 Solve the problem, using a successively larger number collocation points

```
for n=[25 70]

    p = tomPhase('p', t, 0, 1000, n);
    setPhase(p);
    tomStates x1 x2 x3
    tomControls T

    % Initial guess
    if n==25
        x0 = {icollocate({
            x1 == 1-t/1000;
            x2 == 0.15
            x3 == 0.66*t/1000
            })
            collocate(T==273*(t<100)+415*(t>=100))};
    else
        x0 = {icollocate({
            x1 == x1_init
            x2 == x2_init
            x3 == x3_init
```

```
        })
        collocate(T==T_init)};
end

% Box constraints
cbox = {
    0 <= icollocate(x1) <= 1
    0 <= icollocate(x2) <= 1
    0 <= icollocate(x3) <= 1
    273 <= collocate(T) <= 415};

% Boundary constraints
cbnd = initial({x1 == 1; x2 == 0
    x3 == 0});

% Various constants and expressions
ki0 = [1e3; 1e7; 10; 1e-3];
Ei  = [3000; 6000; 3000; 0];
ki4 = ki0(4)*exp(-Ei(4)./T);
ki3 = ki0(3)*exp(-Ei(3)./T);
ki2 = ki0(2)*exp(-Ei(2)./T);
ki1 = ki0(1)*exp(-Ei(1)./T);

% ODEs and path constraints
ceq = collocate({
    dot(x1) == -ki1.*x1-ki2.*x1
    dot(x2) == ki1.*x1-(ki3+ki4).*x2
    dot(x3) == ki3.*x2});

% Objective
objective = -final(x3);
```

29.4 Solve the problem

```
options = struct;
options.name = 'Denbigh System';
solution = ezsolve(objective, {cbox, cbnd, ceq}, x0, options);
x1_init  = subs(x1,solution);
x2_init  = subs(x2,solution);
x3_init  = subs(x3,solution);
T_init   = subs(T,solution);
```

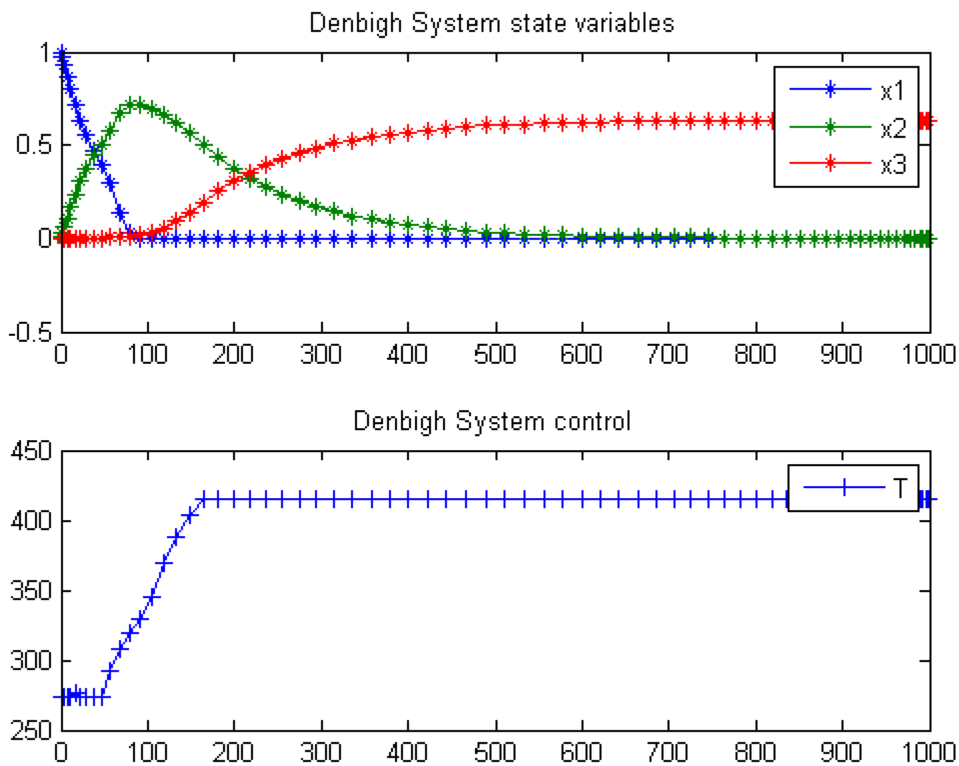
```
Problem type appears to be: lpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
```

```
===================================================================================
Problem: ---  1: Denbigh System                 f_k      -0.633847592419796270
                                         sum(|constr|)      0.000000569070071108
                              f(x_k) + sum(|constr|)     -0.633847023349725200
                                               f(x_0)     -0.660000000000000140

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv   14 ConJacEv   14 Iter   11 MinorIter 1033
CPU time: 0.140625 sec. Elapsed time: 0.140000 sec.


Problem type appears to be: lpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
===================================================================================
Problem: ---  1: Denbigh System                 f_k      -0.633520858635351790
                                         sum(|constr|)      0.000526177451453518
                              f(x_k) + sum(|constr|)     -0.632994681183898230
                                               f(x_0)     -0.633848214286008240

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv   20 ConJacEv   20 Iter   13 MinorIter  233
CPU time: 0.437500 sec. Elapsed time: 0.453000 sec.


end

t  = collocate(subs(t,solution));
x1 = collocate(x1_init);
x2 = collocate(x2_init);
x3 = collocate(x3_init);
T  = collocate(T_init);
```

29.5 Plot result

```
subplot(2,1,1)
plot(t,x1,'*-',t,x2,'*-',t,x3,'*-');
legend('x1','x2','x3');
title('Denbigh System state variables');
```

```
subplot(2,1,2)
plot(t,T,'+-');
legend('T');
title('Denbigh System control');
```



30 Dielectrophoresis Particle Control

Time-Optimal Control of a Particle in a Dielectrophoretic System, Dong Eui Chang, Nicolas Petit, and Pierre Rouchon

30.1 Problem Description

Find u over t in [0; t_F] to minimize:

$$J = t_f$$

subject to:

$$\frac{dx}{dt} = y * u + alpha * u^2$$
$$\frac{dy}{dt} = -c * y + u$$
$$|u| <= 1$$
$$alpha = -\frac{3}{4}$$
$$c = 1$$
$$[x_0 \; y_0] = [1 \; 0]$$
$$x_{t_f} = 2$$

Reference: [12]

30.2 Problem setup

```
toms t
toms t_f
p = tomPhase('p', t, 0, t_f, 60);
setPhase(p);

tomStates x y
tomControls u

% Initial guess
x0 = {t_f == 10
```

```
    icollocate({
    x == 1+1*t/t_f
    y == t/t_f
    })
    collocate(u == 1)};

% Box constraints
cbox = {
    sqrt(eps) <= icollocate(x)
    sqrt(eps) <= collocate(y)
    1         <= t_f <= 100
    -1        <= collocate(u) <= 1};

% Boundary constraints
cbnd = {initial({x == 1; y == 0})
    final({x == 2})};

% ODEs and path constraints
ceq = collocate({
    dot(x) == y.*u-3/4*u.^2
    dot(y) == -y+u});

% Objective
objective = t_f;
```

30.3 Solve the problem

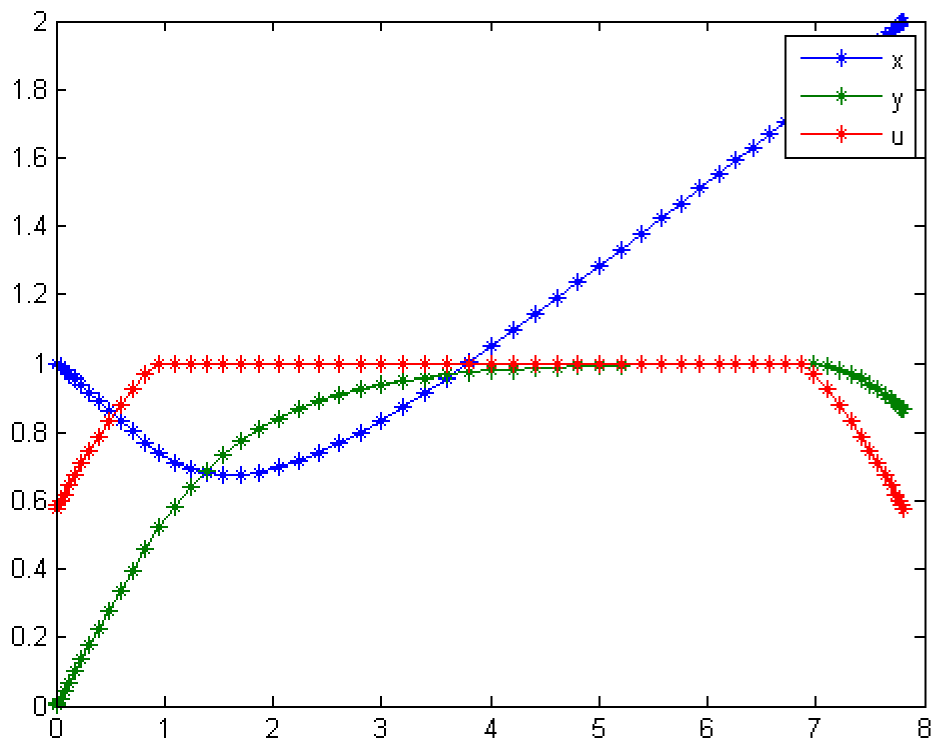
```
options = struct;
options.name = 'Dielectrophoresis Control';
solution = ezsolve(objective, {cbox, cbnd, ceq}, x0, options);
t = subs(collocate(t),solution);
x = subs(collocate(x),solution);
y = subs(collocate(y),solution);
u = subs(collocate(u),solution);
```

```
Problem type appears to be: lpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Dielectrophoresis Control      f_k       7.811292811901784800
                                       sum(|constr|)      0.000001365448751008
                              f(x_k) + sum(|constr|)      7.811294177350536200
                                              f(x_0)     10.000000000000000000

Solver: snopt.  EXIT=0.  INFORM=1.
```

```
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv   26 ConJacEv   26 Iter   25 MinorIter  218
CPU time: 0.281250 sec. Elapsed time: 0.281000 sec.
```

30.4 Plot result

```
figure(1);
plot(t,x,'*-',t,y,'*-',t,u,'*-');
legend('x','y','u');
```



31 Disturbance Control

Optimal On-Line Control and Classical Regulation Problem, Faina M. Kirillova, Institute of Mathematics National Academy of Sciences of Belarus.

Algorithm of Acting Optimal Controller

31.1 Problem Description

Find u over t in [0; 25] to minimize:

$$J = 0$$

subject to:

$$\frac{dx_1}{dt} = x_3$$
$$\frac{dx_2}{dt} = x_4$$
$$\frac{dx_3}{dt} = -x_1 + x_2 + u$$
$$\frac{dx_4}{dt} = 0.1 * x_1 - 1.02 * x_2 + 0.3 * sin(4 * t) * (t < 9.75)$$
$$x(t_0) = [0\ 0\ 2\ 1]$$
$$x(t_f) = [0\ 0\ 0\ 0]$$
$$0 <= u <= 1$$

Reference: [20]

31.2 Problem setup

```
toms t
p = tomPhase('p', t, 0, 25, 80);
setPhase(p);

tomStates x1 x2 x3 x4
tomControls u
```

```
% Box constraints
cbox = {0 <= collocate(u) <= 1};

% Boundary constraints
cbnd = {initial({x1 == 0; x2 == 0
    x3 == 2; x4 == 1})
    final({x1 == 0; x2 == 0
    x3 == 0; x4 == 0})};

% ODEs and path constraints
ceq = collocate({
    dot(x1) == x3
    dot(x2) == x4
    dot(x3) == -x1+x2+u
    dot(x4) == 0.1*x1-1.02*x2+0.3*sin(4*t).*(t<9.75)});

% Objective
objective = 0;
```

31.3 Solve the problem

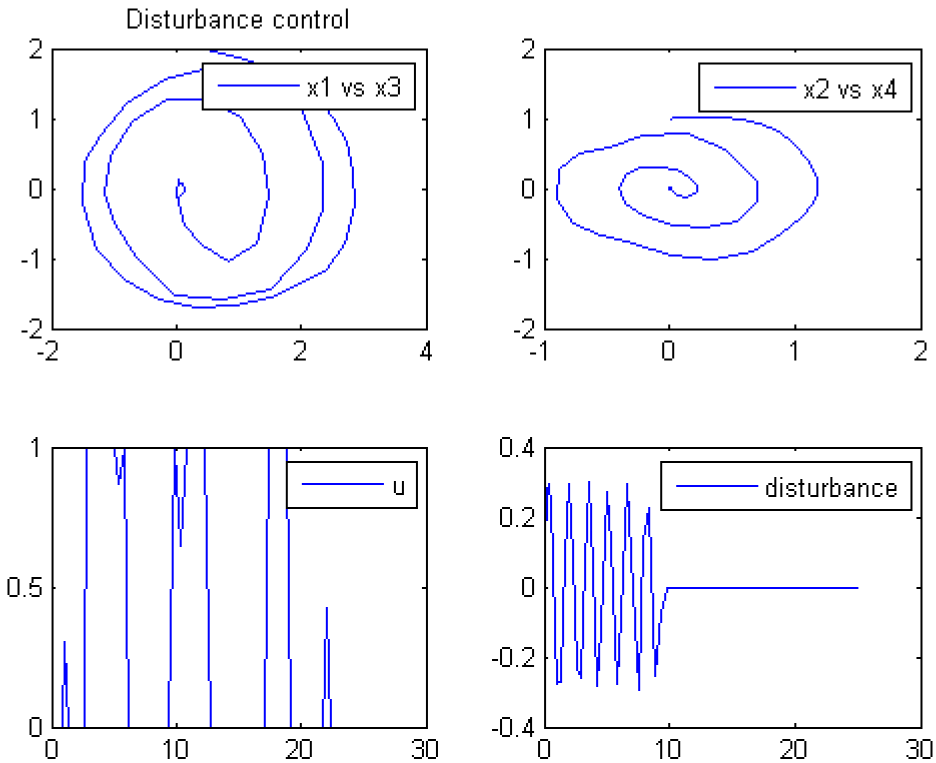
```
options = struct;
options.name = 'Disturbance Control';
solution = ezsolve(objective, {cbox, cbnd, ceq}, [], options);
t  = subs(collocate(t),solution);
x1 = subs(collocate(x1),solution);
x2 = subs(collocate(x2),solution);
x3 = subs(collocate(x3),solution);
x4 = subs(collocate(x4),solution);
u  = subs(collocate(u),solution);
```

```
Problem type appears to be: lp
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Disturbance Control            f_k       0.000000000000000000
                                       sum(|constr|)      0.000000000046160833
                              f(x_k) + sum(|constr|)      0.000000000046160833
                                              f(x_0)      0.000000000000000000

Solver: CPLEX.  EXIT=0.  INFORM=1.
CPLEX Dual Simplex LP solver
Optimal solution found

FuncEv  336 Iter  336
```

```
CPU time: 0.171875 sec. Elapsed time: 0.172000 sec.
```

31.4 Plot result

```
figure(1);
subplot(2,2,1)
plot(x1,x3,'-');
title('Disturbance control');
legend('x1 vs x3');
subplot(2,2,2)
plot(x2,x4,'-');
legend('x2 vs x4');

subplot(2,2,3)
plot(t,u,'-');
legend('u');

subplot(2,2,4)
plot(t,0.3*sin(4*t).*(t<9.75),'-');
legend('disturbance');
```


Disturbance control
x1 vs x3
x2 vs x4
u
disturbance
2
1
0
-1
-2
-2
0
2
4
-1
0.5
10
20
30
0.4
0.2
-0.2
-0.4

32 Drug Displacement Problem

ITERATIVE DYNAMIC PROGRAMMING, REIN LUUS

12.4.3 Example 3: The desired level of two drugs, warfarin and phenylbutazone, must be reached in a patients bloodstream in minimum time.

CHAPMAN & HALL/CRC Monographs and Surveys in Pure and Applied Mathematics

32.1 Problem Formulation

Find u over t in [0; t] to minimize

$$J = t_F$$

subject to:

$$\begin{gathered}
\frac{dx_1}{dt} = g_1 * (g_4 * (0.02 - x_1) + 46.4 * x_1 * (u - 2 * x_2)) \\
\frac{dx_2}{dt} = g_1 * (g_3 * (u - 2 * x_2) + 46.4 * (0.02 - x_1)) \\
g_2 = 1 + 0.2 * (x_1 + x_2) \\
g_3 = g_2^2 + 232 + 46.4 * x_2 \\
g_4 = g_2^2 + 232 + 46.4 * x_1 \\
g_1 = \frac{g_2^2}{g_3 * g_4 - 2152.96 * x_1 * x_2} \\
0 <= u <= 8
\end{gathered}$$

x1 is the concentration of warfarin, and x2 of phenylbutazone. The initial and final condition are:

$$\begin{gathered}
x_0 = [0.02 \ 0] \\
x_{t_f} = [0.02 \ 2.00]
\end{gathered}$$

Reference: [25]

32.2 Problem setup

```
toms t
toms t_f
p = tomPhase('p', t, 0, t_f, 50);
setPhase(p);

tomStates x1 x2
tomControls u

% Initial guess
x0 = {t_f == 300
    icollocate({
    x1 == 0.02; x2 == 2*t/t_f})
    collocate(u == 8-8*t/t_f)};

% Box constraints
cbox = { 1 <= t_f <= 500
    0 <= collocate(u) <= 8};

% Boundary constraints
cbnd = {initial({x1 == 0.02; x2 == 0})
    final({x1 == 0.02; x2 == 2})};

% General variables
g2 = 1+0.2*(x1+x2);
g3 = g2.^2+232+46.4*x2;
g4 = g2.^2+232+46.4*x1;
g1 = g2.^2./(g3.*g4-2152.96*x1.*x2);

% ODEs and path constraints
ceq = collocate({
    dot(x1) == g1.*(g4.*(0.02-x1)+46.4*x1.*(u-2*x2))
    dot(x2) == g1.*(g3.*(u-2*x2)+46.4*(0.02-x1))});
```

32.3 Solve the problem

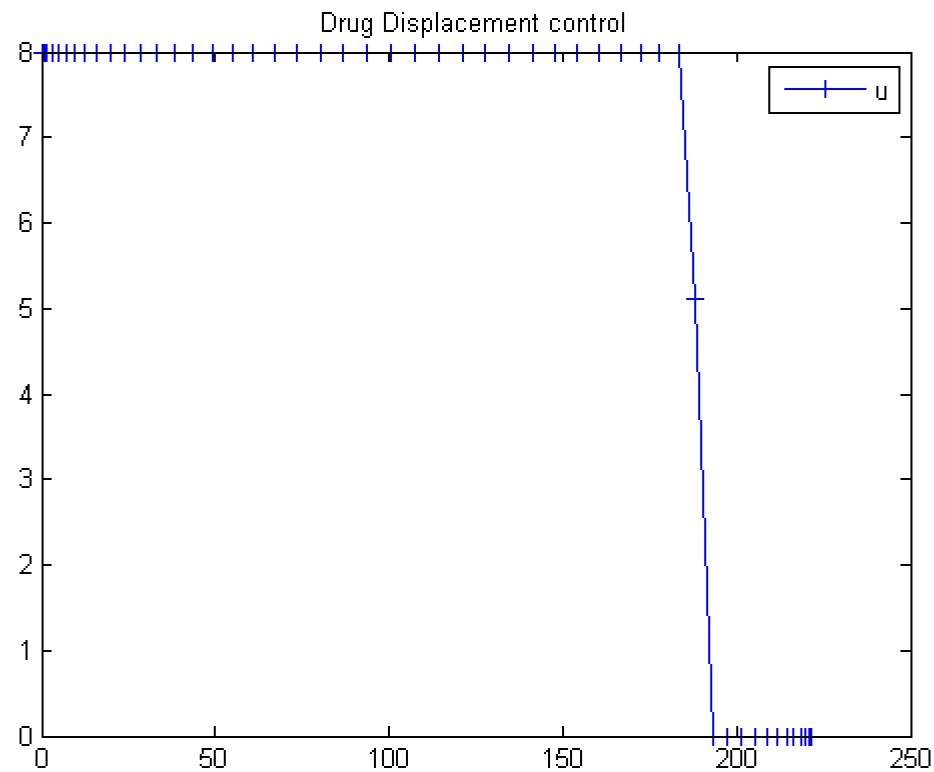
```
options = struct;
options.name = 'Drug Displacement';
% Objective is first parameter
solution = ezsolve(t_f, {cbox, cbnd, ceq}, x0, options);
t = subs(collocate(t),solution);
u = subs(collocate(u),solution);
```

```
Problem type appears to be: lpcon
Starting numeric solver
```

```
===== * * * ================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Drug Displacement              f_k     221.333418113505130000
                                       sum(|constr|)      0.000000061271395437
                              f(x_k) + sum(|constr|)    221.333418174776540000
                                              f(x_0)    300.000000000000000000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv   14 ConJacEv   14 Iter   10 MinorIter  256
CPU time: 0.140625 sec. Elapsed time: 0.140000 sec.
```

32.4 Plot result

```
figure(1)
plot(t,u,'+-');
legend('u');
title('Drug Displacement control');
```



Drug Displacement control
u
8
7
6
5
4
3
2
1
0
0
50
100
150
200
250

33 Optimal Drug Scheduling for Cancer Chemotherapy

Dynamic optimization of bioprocesses: efficient and robust numerical strategies 2003, Julio R. Banga, Eva Balsa-Cantro, Carmen G. Moles and Antonio A. Alonso

Case Study III: Optimal Drug Scheduling for Cancer Chemotherapy

33.1 Problem description

Many researches have devoted their efforts to determine whether current methods for drugs administration during cancer chemotherapy are optimal, and if alternative regimens should be considered. Martin (1992) considered the interesting problem of determining the optimal cancer drug scheduling to decrease the size of a malignant tumor as measured at some particular time in the future. The drug concentration must be kept below some level throughout the treatment period and the cumulative (toxic) effect of the drug must be kept below the ultimate tolerance level. Bojkov et al. (1993) and Luus et al. (1995) also studied this problem using direct search optimization. More recently, Carrasco and Banga (1997) have applied stochastic techniques to solve this problem, obtaining better results (Carrasco & Banga 1998). The mathematical statement of this dynamic optimization problem is: Find u(t) over t in [t0; t_f] to maximize:

$$J = x_1(t_f)$$

subject to:

$$\frac{dx_1}{dt} = -k_1 * x_1 + k_2 * (x_2 - k_3) * H(x_2 - k_3)$$
$$\frac{dx_2}{dt} = u - k_4 * x_2$$
$$\frac{dx_3}{dt} = x_2$$

where the tumor mass is given by N = 10^12 * exp (-x1) cells, x2 is the drug concentration in the body in drug units [D] and x3 is the cumulative effect of the drug. The parameters are taken as k1 = 9.9e-4 days, k2 = 8.4e-3 days-1 [De-1], k3 = 10 [De-1], and k4 = 0.27 days-1. The initial state considered is:

$$x(t_0) = [log(100)\ 0\ 0]'$$

where,

```
H(x2-k3) = 1 if x2 >= k3 or 0 if x2 < k3
```

and the final time t_f = 84 days. The optimization is subject to the following constraints on the drug delivery (control variable):

$$u >= 0$$

There are the following path constraints on the state variables:

$$x_2(t) <= 50$$
$$x_3(t) <= 2.1e3$$

Also, there should be at least a 50% reduction in the size of the tumor every three weeks, so that the following point constraints must be considered:

$$x_1(21) >= log(200)$$
$$x_1(42) >= log(400)$$
$$x_1(63) >= log(800)$$

State number 3 is converted to an integral constraints in the formulation.

Reference: [3]

33.2 Problem setup

```
toms t

nn = [20 40 120];

for i = 1:length(nn)

    n = nn(i);
    p = tomPhase('p', t, 0, 84, n);
    setPhase(p);

    tomStates x1 x2
    tomControls u
```

```
% Initial guess
if i==1
    x0 = {icollocate(x2 == 10)
        collocate(u == 20)};
else
    x0 = {icollocate({x1 == x1opt; x2 == x2opt})
        collocate(u == uopt)};
end

% Box constraints
cbox = {
    0 <= mcollocate(x1)
    0 <= mcollocate(x2) <= 50
    0 <= collocate(u)   <= 80};

% Boundary constraints
cbnd = initial({x1 == log(100); x2 == 0});

% ODEs and path constraints
k1 = 9.9e-4; k2 = 8.4e-3;
k3 = 10;     k4 = 0.27;
ceq = {collocate({
    dot(x1) == -k1*x1+k2*max(x2-k3,0)
    dot(x2) == u-k4*x2})
    % Point-wise conditions
    atPoints([21;42;63],x1) >= log([200;400;800])
    % Integral constr.
    integrate(x2) == 2.1e3};

% Objective
objective = -final(x1);
```

33.3 Solve the problem

```
options = struct;
options.name = 'Drug Scheduling';
options.solver = 'multiMin';
options.xInit = 130-n;
solution = ezsolve(objective, {cbox, cbnd, ceq}, x0, options);

x1opt = subs(x1, solution);
x2opt = subs(x2, solution);
uopt  = subs(u, solution);
```

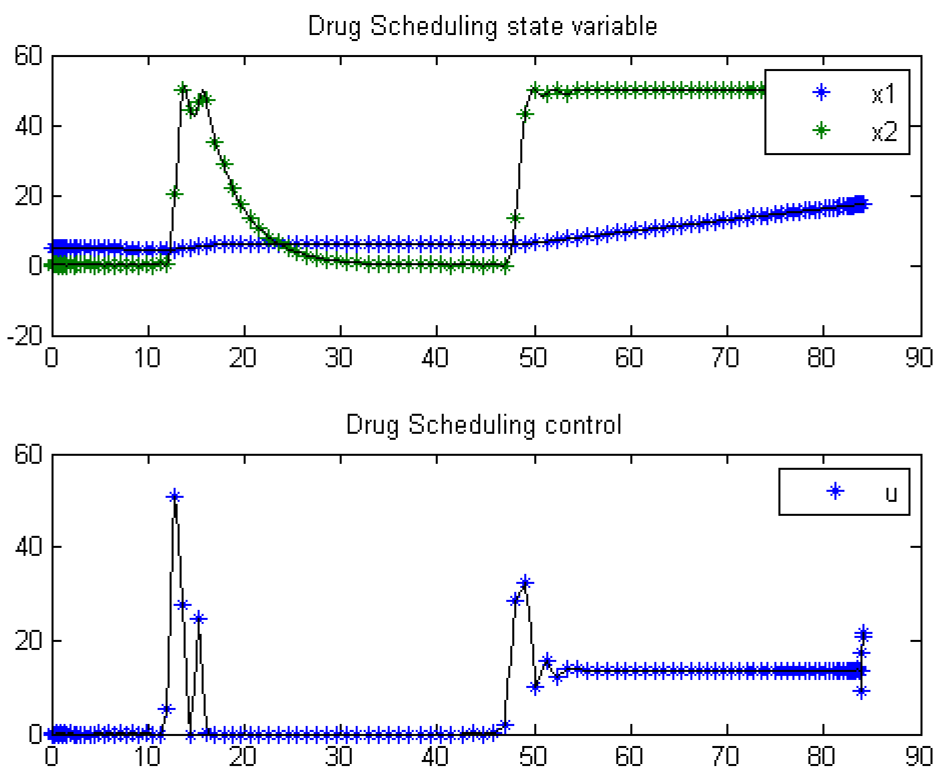
```
Problem type appears to be: lpcon
```

```
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Drug Scheduling - Trial 1      f_k     -16.628828853430683000
                                     sum(|constr|)      0.000000000002606145
                              f(x_k) + sum(|constr|)    -16.628828853428075000

Solver: multiMin with local solver snopt.  EXIT=0.  INFORM=0.
Find local optima using multistart local search
Did 1 local tries. Found 1 global, 1 minima. TotFuncEv 1. TotConstrEv 33

FuncEv    1 ConstrEv   33 ConJacEv   32 Iter   15
CPU time: 0.375000 sec. Elapsed time: 0.375000 sec.


Problem type appears to be: lpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Drug Scheduling - Trial 1      f_k     -16.875588505484753000
                                     sum(|constr|)      0.000000000001656122
                              f(x_k) + sum(|constr|)    -16.875588505483098000

Solver: multiMin with local solver snopt.  EXIT=0.  INFORM=0.
Find local optima using multistart local search
Did 1 local tries. Found 1 global, 1 minima. TotFuncEv 1. TotConstrEv 43

FuncEv    1 ConstrEv   43 ConJacEv   42 Iter   15
CPU time: 0.421875 sec. Elapsed time: 0.422000 sec.


Problem type appears to be: lpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Drug Scheduling - Trial 1      f_k     -17.395852753239591000
                                     sum(|constr|)      0.000000023567180277
                              f(x_k) + sum(|constr|)    -17.395852729672409000

Solver: multiMin with local solver snopt.  EXIT=0.  INFORM=0.
Find local optima using multistart local search
Did 1 local tries. Found 1 global, 1 minima. TotFuncEv 1. TotConstrEv 53

FuncEv    1 ConstrEv   53 ConJacEv   52 Iter   19
CPU time: 1.875000 sec. Elapsed time: 1.907000 sec.
```

```
end
```

33.4 Plot result

```
subplot(2,1,1)
ezplot([x1;x2]);
legend('x1','x2');
title('Drug Scheduling state variable');

subplot(2,1,2)
ezplot(u);
legend('u');
title('Drug Scheduling control');
```



34 Euler Buckling Problem

Problem 4: Miser3 manual

34.1 Problem description

Over t in [0; 1], minimize

$$J = -z_1$$

subject to:

$$\frac{dx_1}{dt} = x_2$$
$$\frac{dx_2}{dt} = \frac{-z_1 * x_1}{x_3^2}$$
$$\int_0^1 x_3(t)\mathrm{d}t - 1 = 0$$
$$x_1(0) = 0$$
$$x_1(1) = 0$$
$$x_2(0) = 1$$
$$x_3(0) >= 0.5$$
$$x_3(t) >= 0.5$$

Reference: [19]

34.2 Problem setup

```
toms t
toms z1
p = tomPhase('p', t, 0, 1, 40);
setPhase(p);

% States
tomStates x1 x2 x3 x4
% We don't need to introduce any control variables.
```

```
% Initial guess
x0 = {icollocate({
    x1 == 0;   x2 == 1
    x3 == 0.5; x4 == t/40})
    z1 == 10};

% Box constraints
cbox = {
    icollocate({-10 <= x1 <= 10
    -10 <= x2 <= 10; 0.5 <= x3 <= 10})
      0 <= z1 <= 500};

% Boundary constraints
cbnd = {initial({x3 >= 0.5; x1 == 0
    x2 == 1; x4 == 0})
    final({x1 == 0; x4 == 1})};

% ODEs and path constraints
ceq = {collocate({
    dot(x1) == x2
    dot(x2) == -z1*x1./x3.^2
    x3 >= 0.5 % Path constr.
     % Integral constr.
    })
    integrate(x3) == 1};

% Objective
objective = z1;
```

34.3 Solve the problem

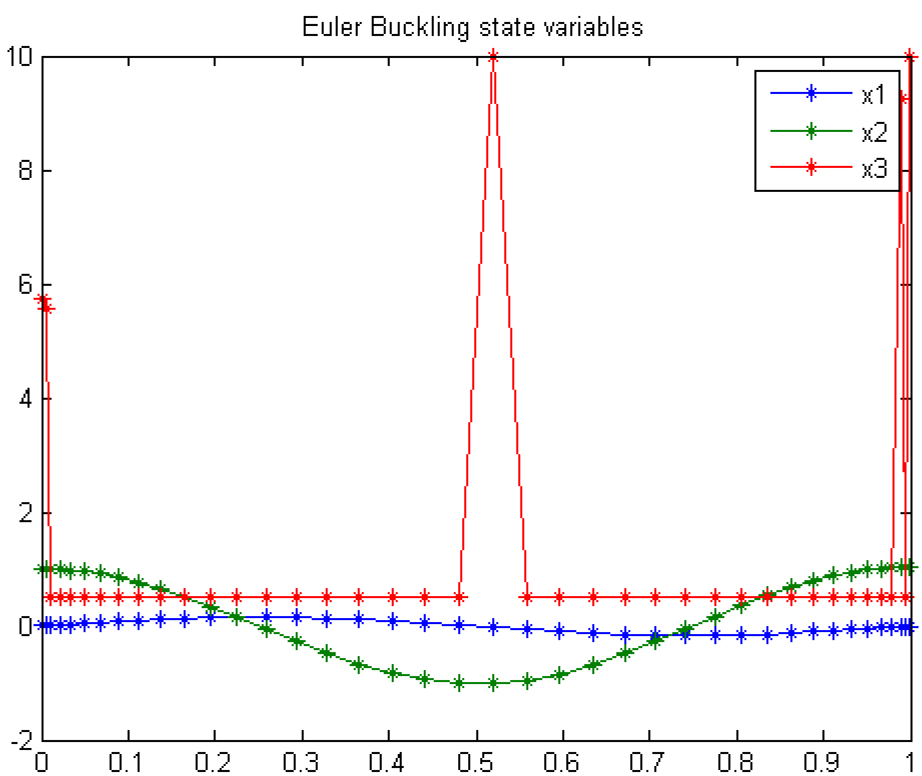
```
options = struct;
options.name = 'Euler Buckling';
solution = ezsolve(objective, {cbox, cbnd, ceq}, x0, options);

% Extract optimal states and controls from solution
t  = subs(collocate(t),solution);
x1 = subs(collocate(x1),solution);
x2 = subs(collocate(x2),solution);
x3 = subs(collocate(x3),solution);
```

```
Problem type appears to be: lpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
```

```
Problem: ---  1: Euler Buckling                  f_k       9.881895688695376400
                                        sum(|constr|)      0.000000016445563739
                               f(x_k) + sum(|constr|)      9.881895705140939500
                                               f(x_0)     10.000000000000000000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv   73 ConJacEv   72 Iter   33 MinorIter  239
CPU time: 0.187500 sec. Elapsed time: 0.188000 sec.
```

34.4 Plot result

```
figure(1)
plot(t,x1,'*-',t,x2,'*-',t,x3,'*-');
legend('x1','x2','x3');
title('Euler Buckling state variables');
```



34.5 Footnote

In the original [Miser3] problem formulation, it is requested to compute "u", equal to x3_t. u is not included in the optimization problem, thereby speeding up the solution process. x3_t can be obtained by simple numeric differentiation of x3.

Note, however, that because there was no constraint on u, and it was not included in the cost function, x3_t looks very strange.

35 MK2 5-Link robot

Singular time-optimal of the MK2 5-Link robot. Implementation without mass matrix inversion.

35.1 Problem description

The dynamic model of the MK2 robot was generated automatically by AUTOLEV that produces Fortran 77 code:

`http://www.autolev.com/`

The transfer to matlab code was performed partly automatically using

```
 1) to_f90: http://users.bigpond.net.au/amiller/
 2) f2matlab.m: http://www.mathworks.com/matlabcentral/fileexchange/5260
```

Programmer: Gerard Van Willigenburg (Wageningen University)

35.2 Problem setup

```
toms t t_f % Free final time

p = tomPhase('p', t, 0, t_f, 20);
setPhase(p);

global AP4AD
AP4AD = true; % Work-around to get more efficient code for this particular case.

% Dimension state and control vector
np = 5; nx = 2*np; nu = np;

% Define the state and control vector
tomStates a1 a2 a3 a4 a5 w1 w2 w3 w4 w5
phi = [a1; a2; a3; a4; a5];
omega = [w1; w2; w3; w4; w5];
tomControls u1 u2 u3 u4 u5
u     = [u1; u2; u3; u4; u5];

% Initial and terminal states
znp    = zeros(np,1);
phif = [0.975; 0.975; 0; 0; 0.975];

% Maximum values controls
```

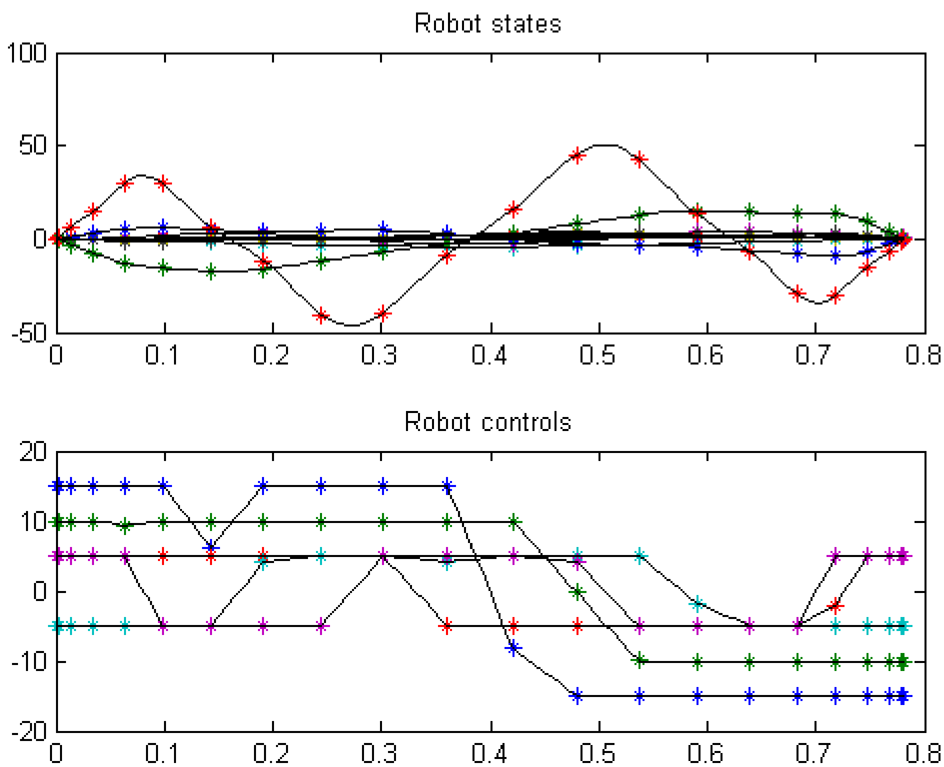
```
umax = [15; 10; 5; 5; 5];

% Initial guess
x0 = {t_f==0.8;
    icollocate({phi == phif; omega == znp})
    collocate({u == 0})};

% Box constraints
cbox = {0.7 <= t_f <= 0.9;
    collocate({-umax <= u <= umax})};

% Boundary constraints
cbnd = {initial({phi == znp; omega == znp})
    final({phi == phif; omega == znp})};

% Compute mass matrix
[mass, rhs] = fiveLinkMK2Robotdyn([phi; omega], u);

% Equality differential equation constraints
ceq = collocate({dot(phi) == omega; mass*dot(omega) == rhs});

% Objective
objective = t_f;
```

35.3 Solve the problem

```
options = struct;
options.use_d2c = 0;
options.use_H   = 0;
options.type = 'lpcon';
options.name = 'Five Link MK2 Robot';
options.derivatives = 'automatic';
solution = ezsolve(objective, {cbox, cbnd, ceq}, x0, options);

% Plot intermediate result
subplot(2,1,1);
ezplot([phi; omega]);
title('Robot states');

subplot(2,1,2);
ezplot(u);
title('Robot controls');

clear global AP4AD
```

```
Starting numeric solver
```

```
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Five Link MK2 Robot            f_k       0.781121381435685990
                                       sum(|constr|)      0.000004003058181095
                              f(x_k) + sum(|constr|)      0.781125384493867040
                                              f(x_0)      0.800000000000000040

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied
MAD TB Automatic Differentiation estimating: gradient and constraint gradient

FuncEv    1 ConstrEv  133 ConJacEv   61 Iter   40 MinorIter 1236
CPU time: 67.109375 sec. Elapsed time: 55.922000 sec.
```



36 Flight Path Tracking

Minimum Cost Optimal Control: An Application to Flight Level Tracking, John Lygeros, Department of Engineering, University of Cambridge, Cambridge, UK.

36.1 Problem Description

Find scalar w over t in [0; t_F] to minimize:

$$J = \int_0^{100} (x_3^2) \mathrm{d}t$$

subject to:

Equations in the code.

Reference: [26]

36.2 Problem setup

```
toms t
p = tomPhase('p', t, 0, 100, 60);
setPhase(p);

tomStates x1s x2 x3s
tomControls u1s u2

x1 = x1s*100;
x3 = x3s*10;
u1 = u1s*10e3;
cr2d = pi/180;

% Box constraints
cbox = {
    92        <= icollocate(x1) <= 170
    -20*cr2d <= icollocate(x2) <= 25*cr2d
    -150      <= icollocate(x3) <= 150
    60e3      <= collocate(u1) <= 120e3
    -150      <= collocate(u2) <= 150};

% Boundary constraints
cbnd = initial({x1 == 153.73
```

```
    x2 == 0; x3 == 0});

L = 65.3;
D = 3.18;
m = 160e3;
g = 9.81;
c = 6;

% ODEs and path constraints
ceq = collocate({
    dot(x1) == (-D/m*x1.^2-g*sin(x2)+u1/m)
    dot(x2) == L/m*x1.*(1-c*x2)-g*cos(x2)./x1+L*c/m*u2
    dot(x3) == (x1.*sin(x2))});

% Objective
objective = integrate(x3.^2);
```

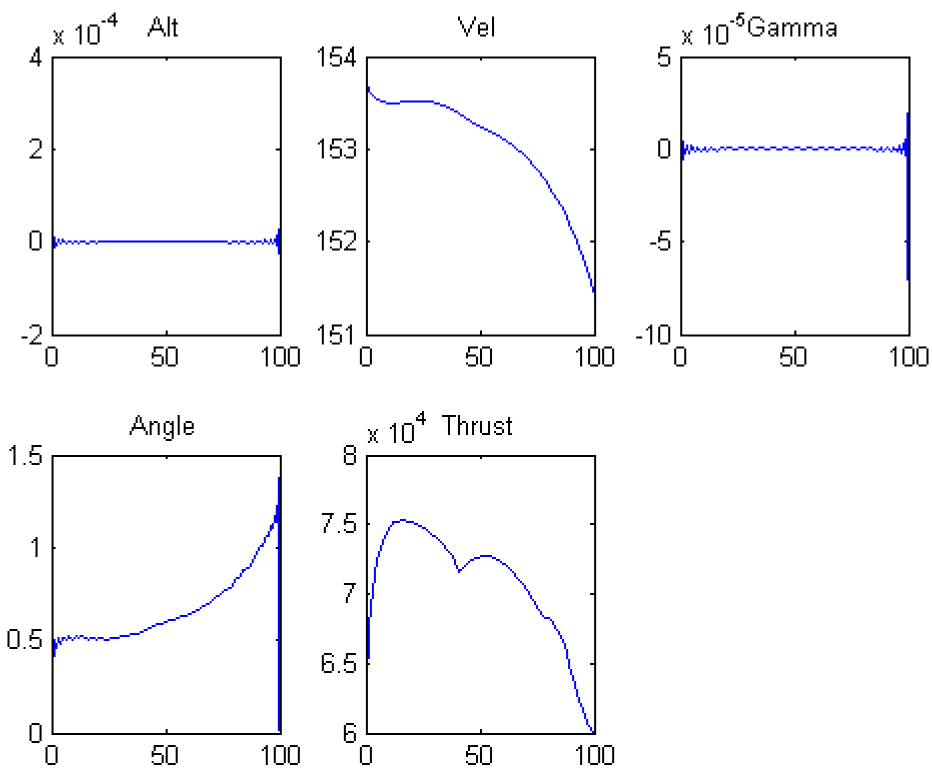
36.3 Solve the problem

```
options = struct;
options.name = 'Flight Path Tracking';
solution = ezsolve(objective, {cbox, cbnd, ceq}, [], options);
t  = subs(collocate(t),solution);
x1 = subs(collocate(x1),solution);
x2 = subs(collocate(x2),solution);
x3 = subs(collocate(x3),solution);
u1 = subs(collocate(u1),solution);
u2 = subs(collocate(u2),solution);
```

```
Problem type appears to be: qpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Flight Path Tracking           f_k       0.000000009789752895
                                       sum(|constr|)      0.000000000014497456
                              f(x_k) + sum(|constr|)      0.000000009804250351
                                              f(x_0)      0.000000000000000000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv  444 ConJacEv  444 Iter  421 MinorIter  720
CPU time: 13.125000 sec. Elapsed time: 13.344000 sec.
```

36.4 Plot result

```
figure(1);
subplot(2,3,1);
plot(t,x3,'-');
title('Alt')
subplot(2,3,2);
plot(t,x1,'-');
title('Vel')
subplot(2,3,3);
plot(t,x2,'-');
title('Gamma')
subplot(2,3,4);
plot(t,u2,'-');
title('Angle')
subplot(2,3,5);
plot(t,u1,'-');
title('Thrust')
```



37 Food Sterilization

37.1 Problem description

Simplified version of the sterilization problem considered in the paper: Z.S. Chalabi, L.G. van Willigenburg, G. van Straten, 1999, Robust optimal receding horizon control of the thermal sterilization of canned food, Journal of Food Engineering, 40, pp. 207-218.

Programmers: Gerard Van Willigenburg (Wageningen University) Willem De Koning (retired from Delft University of Technology)

37.2 Problem setup

```
% Array with consecutive number of collocation points
narr = [20 30 40];

toms t;
t_f = 1500; % Fixed final time

for n=narr

    p = tomPhase('p', t, 0, t_f, n);
    setPhase(p)

    tomStates x1 x2 x3 x4
    tomControls u1

    % Initial & terminal states
    xi = [20; 20;   0; 0];
    xf = [40;  0; 100; 0];

    % Initial guess
    if n==narr(1)
        x0 = {icollocate({x1 == xf(1); x2 == xf(2)
            x3 == xf(3); x4 == xf(4)})
            collocate({u1 == 50})};
    else
        x0 = {icollocate({x1 == xopt1; x2 == xopt2
            x3 == xopt3; x4 == xopt4})
            collocate({u1 == uopt1})};
    end

    % Box constraints
```

```
cbox = {0 <= collocate(u1) <= 50};

% Boundary constraints
cbnd = {initial({x1 == xi(1); x2 == xi(2); x3 == xi(3); x4 == xi(4)})};

% ODEs and path constraints
pv  = [0.01; 0.005; 0.01; 20; 10; 121.11; 25.56; 121.11];
dx1 = pv(1)*(x2-x1);
dx2 = pv(2)*(pv(4)-x2)+pv(3)*u1;
dx3 = exp(log(10)/pv(5)*(x1-pv(6)));
dx4 = exp(log(10)/pv(7)*(x1-pv(8)));

ceq = collocate({
    dot(x1) == dx1; dot(x2) == dx2
    dot(x3) == dx3; dot(x4) == dx4});

% Objective
objective = final(x4)+final((x3-100)^2)+final((x1-40)^2);
```

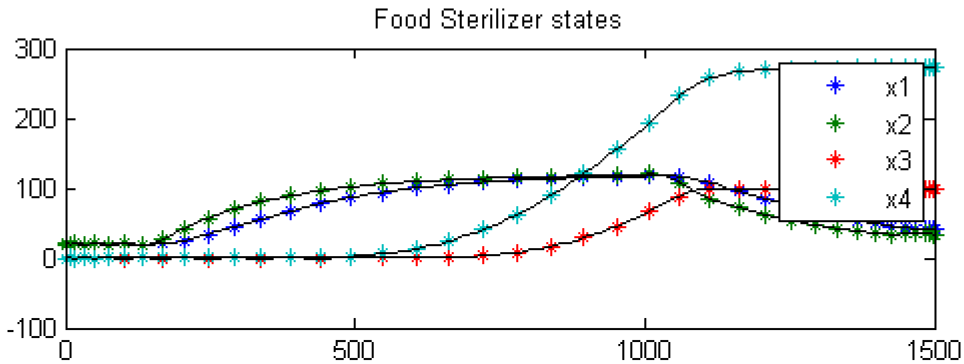
37.3 Solve the problem

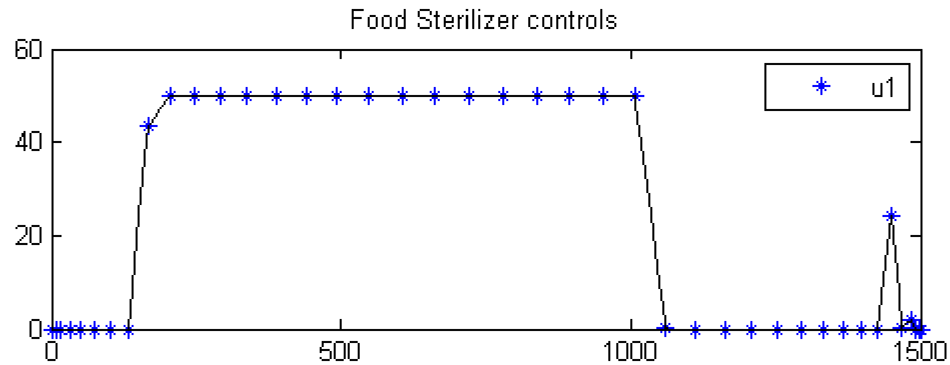
```
options = struct;
options.name = 'Food Sterilization';
solution = ezsolve(objective, {cbox, cbnd, ceq}, x0, options);

xopt1 = subs(x1,solution);
xopt2 = subs(x2,solution);
xopt3 = subs(x3,solution);
xopt4 = subs(x4,solution);
uopt1 = subs(u1,solution);
```

```
Problem type appears to be: qpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Food Sterilization             f_k     280.311120000031220000
                                       sum(|constr|)      0.000000895049191567
                              f(x_k) + sum(|constr|)    280.311120895080420000
                                              f(x_0)  -1200.000000000116400000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv   51 ConJacEv   51 Iter   42 MinorIter  282
```

```
CPU time: 0.125000 sec. Elapsed time: 0.125000 sec.


Problem type appears to be: qpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Food Sterilization             f_k     271.918687099909220000
                                       sum(|constr|)      0.000000037012142340
                              f(x_k) + sum(|constr|)    271.918687136921330000
                                              f(x_0) -11319.688879999987000000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv   27 ConJacEv   27 Iter   25 MinorIter  187
CPU time: 0.140625 sec. Elapsed time: 0.140000 sec.


Problem type appears to be: qpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Food Sterilization             f_k     272.295309093633480000
                                       sum(|constr|)      0.000000000102328574
                              f(x_k) + sum(|constr|)    272.295309093735800000
                                              f(x_0) -11328.081312900060000000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv   32 ConJacEv   32 Iter   31 MinorIter  216
CPU time: 0.234375 sec. Elapsed time: 0.250000 sec.
```

```
end

figure(1)
subplot(2,1,1);
ezplot([x1; x2; x3; x4]); legend('x1','x2','x3','x4');
title('Food Sterilizer states');

subplot(2,1,2);
ezplot(u1); legend('u1');
title('Food Sterilizer controls');
```


Food Sterilizer states
300
200
100
0
-100
0
500
1000
1500
x1
x2
x3
x4


Food Sterilizer controls
60
40
20
0
0
500
1000
1500
u1

38 Free Floating Robot

Users Guide for dyn.Opt, Example 6a, 6b, 6c

A free floating robot

38.1 Problem description

Find u over t in [0; 5] to minimize

6c is free end time

6a:

$$\int_0^5 0.5*(u_1^2+u_2^2+u_3^2+u_4^2)\mathrm{d}t+$$
$$(x_1(t_F)-4.0)^2+(x_3(t_F)-4.0)^2+x_2(t_F)^2+x_4(t_F)^2+x_5(t_F)^2+x_6(t_F)^2$$

6b:

$$\int_0^5 0.5*(u_1^2+u_2^2+u_3^2+u_4^2)\mathrm{d}t$$

6c:

$$J=t_F$$

subject to:

$$M=10.0$$
$$D=5.0$$
$$Le=5.0$$
$$In=12.0$$
$$s5=sin(x_5)$$

$$c5 = cos(x_5)$$

$$\frac{dx_1}{dt} = x_2$$
$$\frac{dx_2}{dt} = \frac{(u_1 + u_3) * c5 - (u_2 + u_4) * s5}{M}$$
$$\frac{dx_3}{dt} = x_4$$
$$\frac{dx_4}{dt} = \frac{(u_1 + u_3) * s5 + (u_2 + u_4) * c5}{M}$$
$$\frac{dx_5}{dt} = x_6$$
$$\frac{dx_6}{dt} = \frac{(u_1 + u_3) * D - (u_2 + u_4) * Le}{In}$$
$$x(0) = [0\ 0\ 0\ 0\ 0\ 0];$$

6b - x(5) = [4 0 4 0 0 0]; 6c - x(5) = [4 0 4 0 pi/4 0]; 6c - -5 <= u <= 5

Reference: [16]

38.2 Problem setup

```
toms t

for i=1:3

    if i==3
        toms t_f
    else
        t_f = 5;
    end
    p1 = tomPhase('p1', t, 0, t_f, 40);
    setPhase(p1);

    tomStates x1 x2 x3 x4 x5 x6
    tomControls u1 u2 u3 u4

    % Initial guess
    if i==1
        x0 = {icollocate({x1 == 0; x2 == 0; x3 == 0
            x4 == 0; x5 == 0; x6 == 0})
            collocate({u1 == 0; u2 == 0
```

```
        u3 == 0; u4 == 0})};
elseif i==2
    x0 = {icollocate({x1 == x1_init; x2 == x2_init
        x3 == x3_init; x4 == x4_init
        x5 == x5_init; x6 == x6_init})
        collocate({u1 == u1_init; u2 == u2_init
        u3 == u3_init; u4 == u4_init})};
else
    x0 = {t_f == tf_init
        icollocate({x1 == x1_init; x2 == x2_init
        x3 == x3_init; x4 == x4_init
        x5 == x5_init; x6 == x6_init})
        collocate({u1 == u1_init; u2 == u2_init
        u3 == u3_init; u4 == u4_init})};
end

% Box constraints
if i<=2
    cbox = {icollocate({
        -100 <= x1 <= 100; -100 <= x2 <= 100
        -100 <= x3 <= 100; -100 <= x4 <= 100
        -100 <= x5 <= 100; -100 <= x6 <= 100})
        collocate({-1000 <= u1 <= 1000; -1000 <= u2 <= 1000
        -1000 <= u3 <= 1000; -1000 <= u4 <= 1000})};
else
    cbox = {
        icollocate({-100 <= x1 <= 100; -100 <= x2 <= 100
        -100 <= x3 <= 100; -100 <= x4 <= 100
        -100 <= x5 <= 100; -100 <= x6 <= 100})
        collocate({-5 <= u1 <= 5; -5 <= u2 <= 5
        -5 <= u3 <= 5; -5 <= u4 <= 5})};
end

% Boundary constraints
cbnd = initial({x1 == 0; x2 == 0; x3 == 0
    x4 == 0; x5 == 0; x6 == 0});
if i==2
    cbnd6b = {cbnd
        final({x1 == 4; x2 == 0
        x3 == 4; x4 == 0
        x5 == 0; x6 == 0})};
elseif i==3
    cbnd6c = {cbnd
        final({x1 == 4; x2 == 0
        x3 == 4;    x4 == 0
        x5 == pi/4; x6 == 0
        1 <= t_f <= 100})};
```

```
end

% ODEs and path constraints
M = 10.0;
D = 5.0;
Le = 5.0;
In = 12.0;
s5 = sin(x5);
c5 = cos(x5);

ceq = collocate({
    dot(x1) == x2
    dot(x2) == ((u1+u3).*c5-(u2+u4).*s5)/M
    dot(x3) == x4
    dot(x4) == ((u1+u3).*s5+(u2+u4).*c5)/M
    dot(x5) == x6
    dot(x6) == ((u1+u3)*D-(u2+u4)*Le)/In});

% Objective
```

38.3 Solve the problem

```
options = struct;
if i==1
    objective = (final(x1)-4)^2+(final(x3)-4)^2+final(x2)^2+ ...
        final(x4)^2+final(x5)^2+final(x6)^2 + ...
        integrate(0.5*(u1.^2+u2.^2+u3.^2+u4.^2));
    options.name = 'Free Floating Robot 6a';
    solution1 = ezsolve(objective, {cbox, cbnd, ceq}, x0, options);
    tp  = subs(collocate(t),solution1);
    x1p = subs(collocate(x1),solution1);
    x2p = subs(collocate(x2),solution1);
    x3p = subs(collocate(x3),solution1);
    x4p = subs(collocate(x4),solution1);
    x5p = subs(collocate(x5),solution1);
    x6p = subs(collocate(x6),solution1);
    u1p = subs(collocate(u1),solution1);
    u2p = subs(collocate(u2),solution1);
    u3p = subs(collocate(u3),solution1);
    u4p = subs(collocate(u4),solution1);
    tf1 = subs(final(t),solution1);
    x1_init = subs(x1,solution1);
    x2_init = subs(x2,solution1);
    x3_init = subs(x3,solution1);
    x4_init = subs(x4,solution1);
    x5_init = subs(x5,solution1);
    x6_init = subs(x6,solution1);
```

```
        u1_init = subs(u1,solution1);
        u2_init = subs(u2,solution1);
        u3_init = subs(u3,solution1);
        u4_init = subs(u4,solution1);
    elseif i==2
        objective = integrate(0.5*(u1.^2+u2.^2+u3.^2+u4.^2));
        options.name = 'Free Floating Robot 6b';
        solution2 = ezsolve(objective, {cbox, cbnd6b, ceq}, x0, options);
        x1_init = subs(x1,solution2);
        x2_init = subs(x2,solution2);
        x3_init = subs(x3,solution2);
        x4_init = subs(x4,solution2);
        x5_init = subs(x5,solution2);
        x6_init = subs(x6,solution2);
        u1_init = subs(u1,solution2);
        u2_init = subs(u2,solution2);
        u3_init = subs(u3,solution2);
        u4_init = subs(u4,solution2);
        tf_init = subs(final(t),solution2);
    else
        objective = t_f;
        options.name = 'Free Floating Robot 6c';
        solution3 = ezsolve(objective, {cbox, cbnd6c, ceq}, x0, options);
    end
```

```
Problem type appears to be: qpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Free Floating Robot 6a         f_k      13.016949152618082000
                                       sum(|constr|)      0.000000000120833713
                              f(x_k) + sum(|constr|)     13.016949152738915000
                                              f(x_0)      0.000000000000000000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv   35 ConJacEv   35 Iter   31 MinorIter  405
CPU time: 1.046875 sec. Elapsed time: 1.156000 sec.


Problem type appears to be: qpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
```

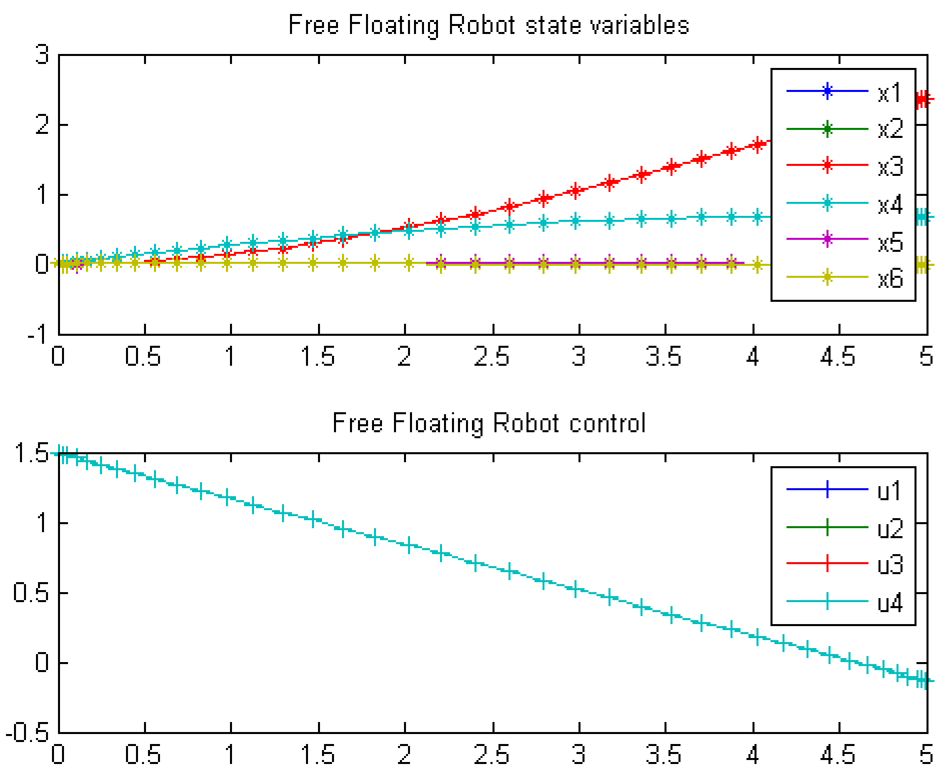
```
===============================================================================
Problem: ---  1: Free Floating Robot 6b         f_k      76.800000142684681000
                                     sum(|constr|)      0.000000241108083137
                              f(x_k) + sum(|constr|)     76.800000383792764000
                                            f(x_0)      6.802639150498469800

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv   35 ConJacEv   35 Iter   25 MinorIter  395
CPU time: 0.921875 sec. Elapsed time: 1.031000 sec.


Problem type appears to be: lpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
===============================================================================
Problem: ---  1: Free Floating Robot 6c         f_k       4.161676034118864100
                                     sum(|constr|)      0.000000000039327186
                              f(x_k) + sum(|constr|)      4.161676034158190900
                                            f(x_0)      5.000000000000000000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv   26 ConJacEv   26 Iter   16 MinorIter  570
CPU time: 0.625000 sec. Elapsed time: 0.656000 sec.


end
```

38.4 Plot result

```
tf2 = tf_init;
tf3 = subs(t_f,solution3);
disp(sprintf('\nFinal time for 6a = %1.4g',tf1));
disp(sprintf('\nFinal time for 6b = %1.4g',tf2));
disp(sprintf('\nFinal time for 6c = %1.4g',tf3));

subplot(2,1,1)
plot(tp,x1p,'*-',tp,x2p,'*-',tp,x3p,'*-',tp,x4p,'*-' ...
    ,tp,x5p,'*-',tp,x6p,'*-');
legend('x1','x2','x3','x4','x5','x6');
title('Free Floating Robot state variables');
```

```
subplot(2,1,2)
plot(tp,u1p,'+-',tp,u2p,'+-',tp,u3p,'+-',tp,u4p,'+-');
legend('u1','u2','u3','u4');
title('Free Floating Robot control');
```

```
Final time for 6a = 5

Final time for 6b = 5

Final time for 6c = 4.162
```



39 Fuller Phenomenon

A Short Introduction to Optimal Control, Ugo Boscain, SISSA, Italy

3.6 Fuller Phenomenon.

39.1 Problem Description

Find u over t in [0; inf] to minimize:

$$J = \int_0^{inf} x_1^2 \mathrm{d}t$$

subject to:

$$\frac{dx_1}{dt} = x_2$$
$$\frac{dx_2}{dt} = u$$
$$x(t_0) = [10\ 0]$$
$$x(t_f) = [0\ 0]$$
$$|u| <= 1$$

Reference: [7]

39.2 Problem setup

```
toms t
toms t_f
p = tomPhase('p', t, 0, t_f, 60);
setPhase(p);

tomStates x1 x2
tomControls u

% Initial guess
x0 = {t_f == 10
    icollocate(x1 == 10-10*t/t_f)
    icollocate(x2 == 0)
```

```
    collocate(u == -1+2*t/t_f)};

% Box constraints
cbox = {1 <= t_f <= 1e4
    -1 <= collocate(u) <= 1};

% Boundary constraints
cbnd = {initial({x1 == 10; x2 == 0})
    final({x1 == 0; x2 == 0})};

% ODEs and path constraints
ceq = collocate({dot(x1) == x2; dot(x2) == u});

% Objective
objective = integrate(x1.^2);
```

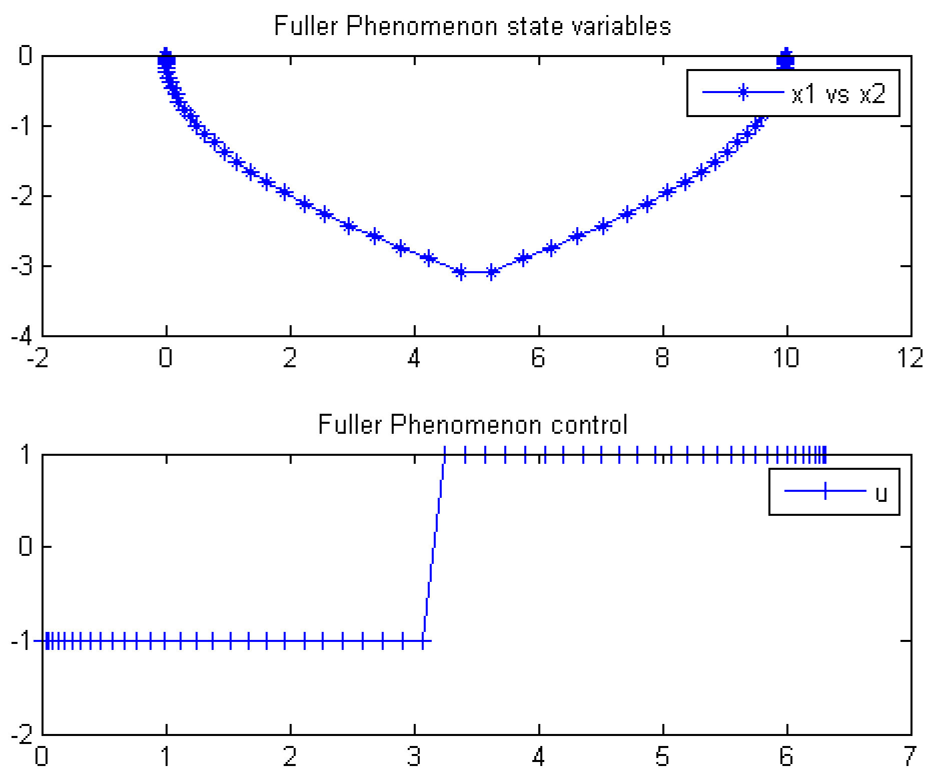
39.3 Solve the problem

```
options = struct;
options.name = 'Fuller Phenomenon';
options.solver = 'snopt';
solution = ezsolve(objective, {cbox, cbnd, ceq}, x0, options);
t  = subs(collocate(t),solution);
x1 = subs(collocate(x1),solution);
x2 = subs(collocate(x2),solution);
u  = subs(collocate(u),solution);
```

```
Problem type appears to be: con
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Fuller Phenomenon              f_k     242.423532418144480000
                                       sum(|constr|)      0.000000063718580492
                              f(x_k) + sum(|constr|)    242.423532481863050000
                                              f(x_0)    333.333333333328770000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv   28 GradEv   26 ConstrEv   27 ConJacEv   26 Iter   14 MinorIter  248
CPU time: 0.218750 sec. Elapsed time: 0.219000 sec.
```

39.4 Plot result

```
subplot(2,1,1)
plot(x1,x2,'*-');
legend('x1 vs x2');
title('Fuller Phenomenon state variables');

subplot(2,1,2)
plot(t,u,'+-');
legend('u');
title('Fuller Phenomenon control');
```



40 Genetic 1

PROCEEDINGS OF WORLD ACADEMY OF SCIENCE, ENGINEERING AND TECHNOLOGY VOLUME 21 JANUARY 2007 ISSN 1307-6884

Optimal Control Problem, Quasi-Assignment Problem and Genetic Algorithm Omid S. Fard and Akbar H. Borzabadi

See paper for failure of GA toolbox algorithm.

Example 1

40.1 Problem Formulation

Find u over t in [0; 1] to minimize

$$J = \int_0^1 u^2 \mathrm{d}t$$

subject to:

$$\frac{dx}{dt} = x^2 + u$$

The initial condition are:

$$x(0) = 0$$
$$x(1) = 0.5$$

Reference: [17]

40.2 Problem setup

```
toms t
p = tomPhase('p', t, 0, 1, 50);
setPhase(p);
```

```
tomStates x
tomControls u

% Initial guess
x0 = {icollocate(x == 0.5*t); collocate(u == 0)};

% Boundary constraints
cbnd = {initial({x == 0}); final({x == 0.5})};

% ODEs and path constraints
ceq = collocate({dot(x) == x.^2+u});

% Objective
objective = integrate(u.^2);
```

40.3 Solve the problem

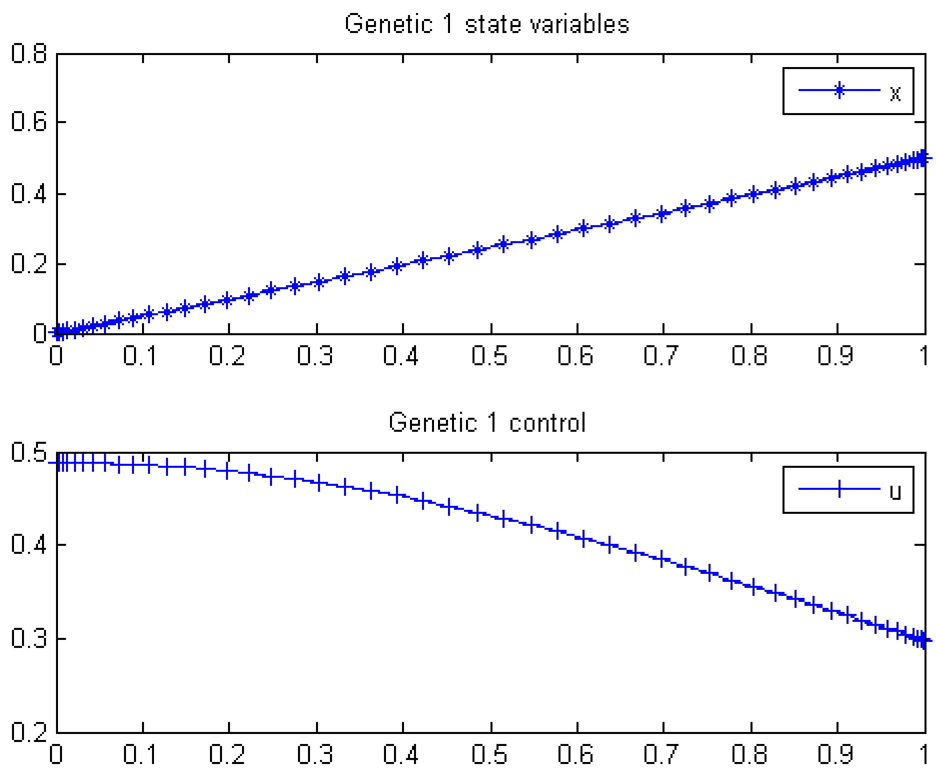
```
options = struct;
options.name = 'Genetic 1';
solution = ezsolve(objective, {cbnd, ceq}, x0, options);
t = subs(collocate(t),solution);
x = subs(collocate(x),solution);
u = subs(collocate(u),solution);
```

```
Problem type appears to be: qpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Genetic 1                         f_k       0.178900993395128240
                                       sum(|constr|)      0.000000000698273828
                              f(x_k) + sum(|constr|)      0.178900994093402070
                                              f(x_0)      0.000000000000000000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv   23 ConJacEv   23 Iter   22 MinorIter   71
CPU time: 0.093750 sec. Elapsed time: 0.094000 sec.
```

40.4 Plot result

```
subplot(2,1,1)
plot(t,x,'*-');
```

```
legend('x');
title('Genetic 1 state variables');

subplot(2,1,2)
plot(t,u,'+-');
legend('u');
title('Genetic 1 control');
```



41 Genetic 2

PROCEEDINGS OF WORLD ACADEMY OF SCIENCE, ENGINEERING AND TECHNOLOGY VOLUME 21 JANUARY 2007 ISSN 1307-6884

Optimal Control Problem, Quasi-Assignment Problem and Genetic Algorithm Omid S. Fard and Akbar H. Borzabadi

See paper for failure of GA toolbox algorithm.

Example 2

41.1 Problem Formulation

Find u over t in [0; 1] to minimize

$$J = \int_0^1 u^2 \mathrm{d}t$$

subject to:

$$\frac{dx}{dt} = \frac{1}{2} * x^2 * sin(x) + u$$

The initial condition are:

$$x(0) = 0$$
$$x(1) = 0.5$$

Reference: [17]

41.2 Problem setup

```
toms t
p = tomPhase('p', t, 0, 1, 50);
setPhase(p);
```

```
tomStates x
tomControls u

% Initial guess
x0 = {icollocate(x == 0.5*t)
    collocate(u == 0)};

% Boundary constraints
cbnd = {initial(x == 0)
    final(x == 0.5)};

% ODEs and path constraints
ceq = collocate(dot(x) == 1/2*x.^2.*sin(x)+u);

% Objective
objective = integrate(u.^2);
```

41.3 Solve the problem

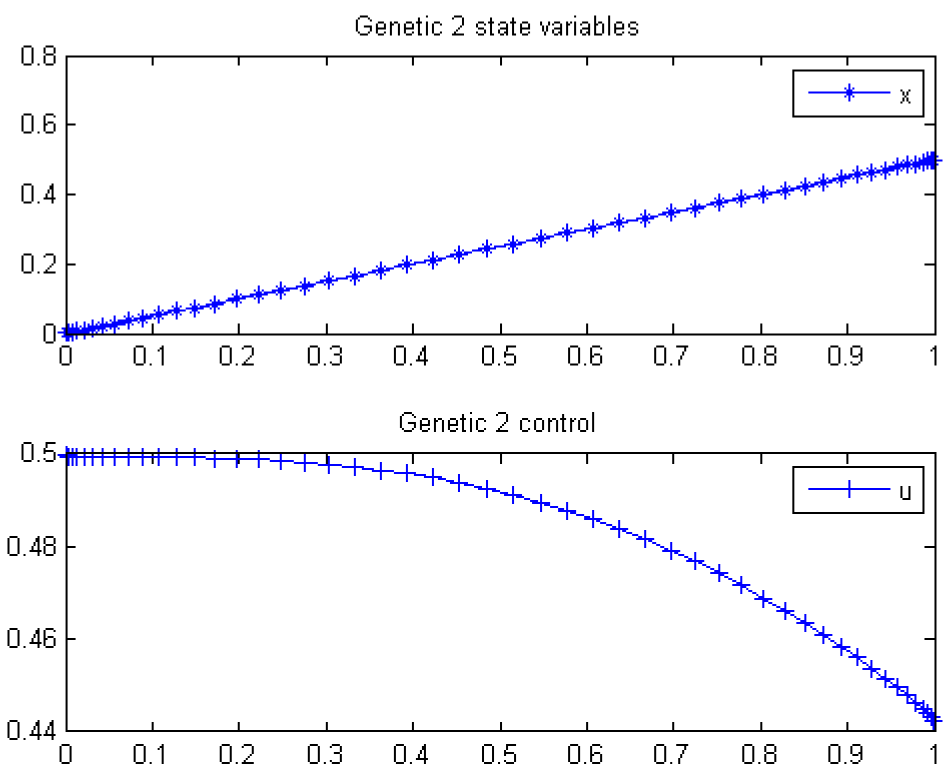
```
options = struct;
options.name = 'Genetic 2';
solution = ezsolve(objective, {cbnd, ceq}, x0, options);
t = subs(collocate(t),solution);
x = subs(collocate(x),solution);
u = subs(collocate(u),solution);
```

```
Problem type appears to be: qpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Genetic 2                       f_k       0.235327080033222360
                                       sum(|constr|)      0.000000001551798634
                              f(x_k) + sum(|constr|)      0.235327081585021000
                                              f(x_0)      0.000000000000000000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv   23 ConJacEv   23 Iter   21 MinorIter   71
CPU time: 0.093750 sec. Elapsed time: 0.093000 sec.
```

41.4 Plot result

```
subplot(2,1,1)
```

```
plot(t,x,'*-');
legend('x');
title('Genetic 2 state variables');

subplot(2,1,2)
plot(t,u,'+-');
legend('u');
title('Genetic 2 control');
```



42 Global Dynamic System

Deterministic Global Optimization of Nonlinear Dynamic Systems, Youdong Lin and Mark A. Stadtherr, Department of Chemical and Biomolecular Engineering, University of Notre Dame

42.1 Problem Description

Find u over t in [0; 1] to minimize:

$$J = -x^2(t_f)$$

subject to:

$$\frac{dx}{dt} = -x^2 + u$$
$$x(t_0) = 9$$
$$-5 <= u <= 5$$

Reference: [36]

42.2 Problem setup

```
toms t
p = tomPhase('p', t, 0, 1, 20);
setPhase(p);

tomStates x
tomControls u

% Box constraints, bounds and odo
c = {-10 <= icollocate(x) <= 10
    -5  <= collocate(u)  <= 5
    initial(x == 9)
    collocate(dot(x) == -x.^2+u)};
```

42.3 Solve the problem

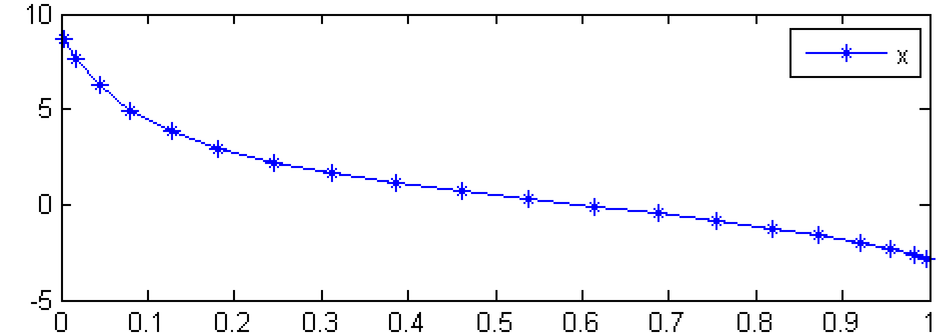
```
options = struct;
```

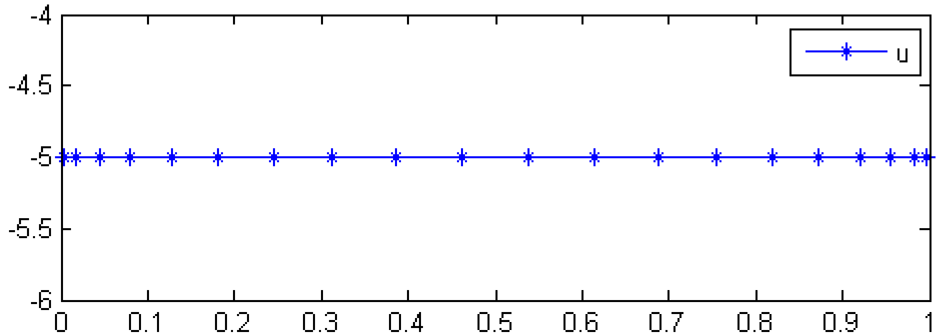
```
options.name = 'Global Dynamic System';
Prob = sym2prob('con',-final(x)^2, c, [], options);
Prob.xInit = 20;
Result = tomRun('multiMin', Prob, 1);
solution = getSolution(Result);
t = subs(collocate(t),solution);
x = subs(collocate(x),solution);
u = subs(collocate(u),solution);
```

```
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Global Dynamic System - Trial 13  f_k      -8.232621699103969300
                                          sum(|constr|)      0.000000001208069200
                                 f(x_k) + sum(|constr|)     -8.232621697895901000

Solver: multiMin with local solver snopt.  EXIT=0.  INFORM=0.
Find local optima using multistart local search
Did 20 local tries. Found 1 global, 2 minima. TotFuncEv 980. TotConstrEv 941

FuncEv  980 GradEv  940 ConstrEv  941 ConJacEv   33 Iter  489
CPU time: 1.453125 sec. Elapsed time: 1.500000 sec.
```

42.4 Plot result

```
figure(1);
subplot(2,1,1);
plot(t,x,'*-');
legend('x');

subplot(2,1,2);
plot(t,u,'*-');
legend('u');
```


x
10
5
0
-5
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


u
-4
-4.5
-5
-5.5
-6
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

43 Goddard Rocket, Maximum Ascent

Benchmarking Optimization Software with COPS Elizabeth D. Dolan and Jorge J. More ARGONNE NATIONAL LABORATORY

43.1 Problem Formulation

Find u(t) over t in [0; T] to minimize

$$J = h(T)$$

subject to:

$$\frac{dv}{dt} = \frac{1}{m} * (T - D) - g$$
$$\frac{dh}{dt} = v$$
$$\frac{dm}{dt} = -\frac{T}{c}$$

$$D = D_0 * v^2 * exp^{-beta * \frac{h - h_0}{h_0}}$$
$$g = g_0 * \frac{h_0}{h}^2$$
$$m(0) = 1$$
$$m(T) = 0.6$$
$$v(0) = 0$$
$$h(0) = 1$$
$$g_0 = 1$$
$$0 <= u <= 3.5$$
$$D_0 = 0.5 * 620$$
$$c = 0.5$$
$$beta = 500$$

Reference: [14]

43.2 Problem setup

```
toms t
toms t_f
```

43.3 Solve the problem, using a successively larger number collocation points

```
for n=[20 50 100]

    p = tomPhase('p', t, 0, t_f, n);
    setPhase(p);
    tomStates v h m
    tomControls T

    % Initial guess
    if n==20
        x0 = {t_f == 1
            icollocate({v == 620; h == 1
            m == 1-0.4*t/t_f})
            collocate(T == 0)};
    else
        x0 = {t_f == tfopt
            icollocate({v == vopt; h == hopt
            m == mopt})
            collocate(T == Topt)};
    end

    % Box constraints
    cbox = {0.1 <= t_f <= 1
        icollocate({
        0 <= v; 1 <= h
        0.6 <= m <= 1
        0   <= T <= 3.5})};

    % Boundary constraints
    cbnd = {initial({v == 0; h == 1; m == 1})
        final({m == 0.6})};

    b    = 500;
    D    = 0.5*620*v.^2.*exp(-b*h);
    g    = 1./h.^2;
    c    = 0.5;

    % ODEs and path constraints
    ceq = collocate({dot(v) == (T-D)./m-g
        dot(h) == v; dot(m) == -T/c});
```

```
% Objective
objective = -final(h);
```

43.4 Solve the problem

```
options = struct;
options.name = 'Goddard Rocket';
solution = ezsolve(objective, {cbox, cbnd, ceq}, x0, options);

% Optimal v and more to use as starting guess
vopt = subs(v, solution);
hopt = subs(h, solution);
mopt = subs(m, solution);
Topt = subs(T, solution);
tfopt = subs(t_f, solution);
```

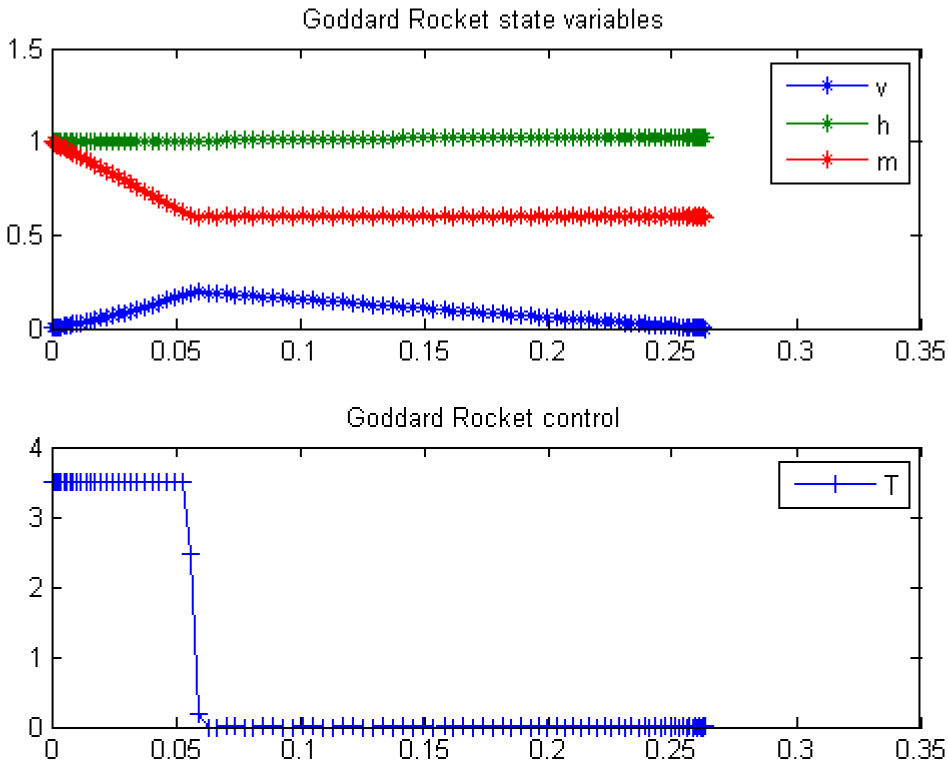
```
Problem type appears to be: lpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Goddard Rocket                 f_k      -1.025133414041158100
                                       sum(|constr|)      0.000002519458500791
                              f(x_k) + sum(|constr|)     -1.025130894582657400
                                              f(x_0)     -0.999999999999998220

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv   41 ConJacEv   41 Iter   23 MinorIter 1196
CPU time: 0.187500 sec. Elapsed time: 0.188000 sec.


Problem type appears to be: lpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Goddard Rocket                 f_k      -1.025311927458321800
                                       sum(|constr|)      0.000016009289221361
                              f(x_k) + sum(|constr|)     -1.025295918169100400
                                              f(x_0)     -1.025133225224282200

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
```

```
Optimality conditions satisfied

FuncEv    1 ConstrEv   23 ConJacEv   23 Iter   14 MinorIter  550
CPU time: 0.312500 sec. Elapsed time: 0.312000 sec.


Problem type appears to be: lpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Goddard Rocket                 f_k      -1.025328777109889600
                                       sum(|constr|)      0.000000000007939354
                              f(x_k) + sum(|constr|)     -1.025328777101950100
                                              f(x_0)     -1.025311927458318500

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv   12 ConJacEv   12 Iter    7 MinorIter  553
CPU time: 0.765625 sec. Elapsed time: 0.797000 sec.
```

```
end

t = subs(collocate(t),solution);
v = subs(collocate(vopt),solution);
h = subs(collocate(hopt),solution);
m = subs(collocate(mopt),solution);
T = subs(collocate(Topt),solution);
```

43.5 Plot result

```
subplot(2,1,1)
plot(t,v,'*-',t,h,'*-',t,m,'*-');
legend('v','h','m');
title('Goddard Rocket state variables');

subplot(2,1,2)
plot(t,T,'+-');
legend('T');
title('Goddard Rocket control');
```


Goddard Rocket state variables
1.5
1
0.5
0
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
v
h
m
Goddard Rocket control
4
3
2
1
0
T

44 Goddard Rocket, Maximum Ascent, Final time free, Singular solution

Example 7.2 (i) from the paper: H. Maurer, "Numerical solution of singular control problems using multiple shooting techniques", Journal of Optimization Theory and Applications, Vol. 18, No. 2, 1976, pp. 235-257

L.G. van Willigenburg, W.L. de Koning

44.1 Problem setup

```
toms t t_f

% Parameters
aalpha = 0.01227; bbeta = 0.145e-3;

c  = 2060;    g0 = 9.81;
r0 = 6.371e6; r02=r0*r0;
m0 = 214.839; mf = 67.9833;
Fm = 9.525515;
```

44.2 Solve the problem, using a successively larger number collocation points

```
for n=[20 40 60]

    p = tomPhase('p', t, 0, t_f, n);
    setPhase(p);
    tomStates h v m
    tomControls F

    % Initial guess
    if n==20
        x0 = {t_f==250
            icollocate({v == 0; h == 0
            m == m0})
            collocate(F == Fm)};
    else
        x0 = {t_f == tfopt
            icollocate({v == vopt; h == hopt
            m == mopt})
            collocate(F == Fopt)};
    end
```

```
% Box constraints
cbox = {100 <= t_f <= 300
    icollocate({0 <= v; 0 <= h
    mf <= m <= m0
    0 <= F <= Fm})};

% Boundary constraints
cbnd = {initial({v == 0; h == 0; m == m0})
    final({v==0; m == mf})};

D = aalpha*v.^2.*exp(-bbeta*h);
g = g0; % or g0*r02./(r0+h).^2;

% ODEs and path constraints
ceq = collocate({dot(h) == v
    m*dot(v) == F*c-D-g*m
    dot(m) == -F});

% Objective
objective = -1e-4*final(h);
```

44.3 Solve the problem

```
options = struct;
options.name = 'Goddard Rocket 1';
options.Prob.SOL.optPar(30) = 30000;
solution = ezsolve(objective, {cbox, cbnd, ceq}, x0, options);

% Optimal v and more to use as starting guess
vopt = subs(v, solution);
hopt = subs(h, solution);
mopt = subs(m, solution);
Fopt = subs(F, solution);
tfopt = subs(t_f, solution);
```

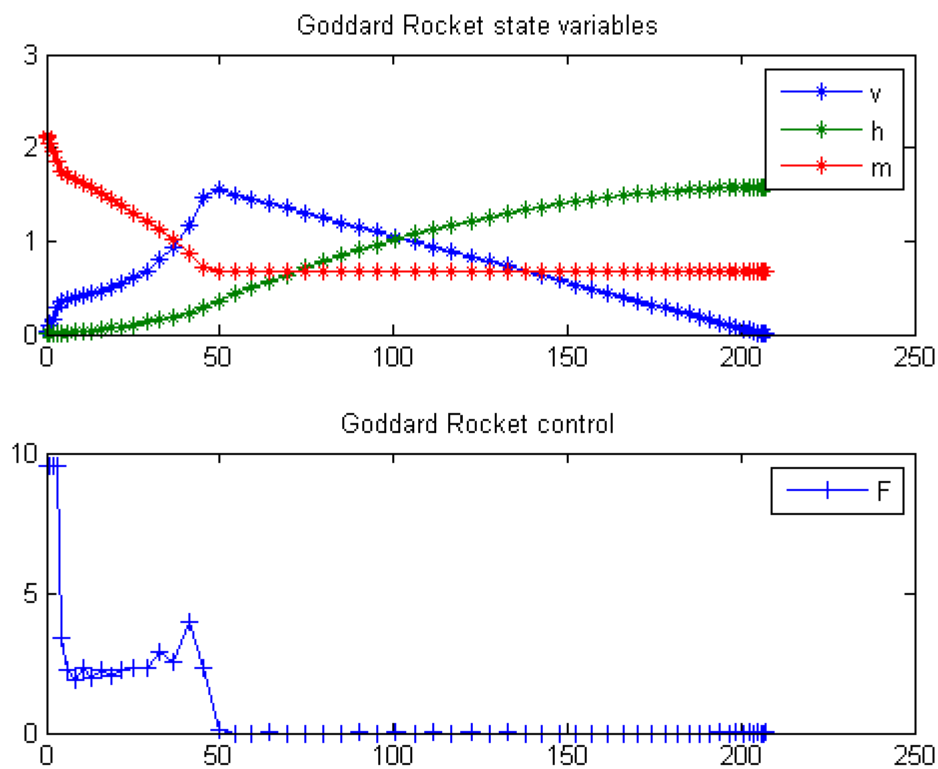
```
Problem type appears to be: lpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Goddard Rocket 1                f_k     -15.580049356479115000
                                       sum(|constr|)      0.000028635866519332
                               f(x_k) + sum(|constr|)    -15.580020720612596000
                                              f(x_0)      0.000000000000000000
```

```
Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv  217 ConJacEv  216 Iter   45 MinorIter 1278
CPU time: 0.484375 sec. Elapsed time: 0.500000 sec.


Problem type appears to be: lpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Goddard Rocket 1               f_k     -15.718139470103905000
                                       sum(|constr|)      0.043635004313635782
                              f(x_k) + sum(|constr|)    -15.674504465790269000
                                              f(x_0)    -15.580049356479037000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv  237 ConJacEv  235 Iter   34 MinorIter  919
CPU time: 1.000000 sec. Elapsed time: 1.000000 sec.


Problem type appears to be: lpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Goddard Rocket 1               f_k     -15.731752553138087000
                                       sum(|constr|)      0.000724004045428859
                              f(x_k) + sum(|constr|)    -15.731028549092658000
                                              f(x_0)    -15.718139470103878000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv  271 ConJacEv  270 Iter   51 MinorIter 4625
CPU time: 3.218750 sec. Elapsed time: 3.265000 sec.


end

t = subs(collocate(t),solution);
v = subs(collocate(vopt),solution);
```

```
h = subs(collocate(hopt),solution);
m = subs(collocate(mopt),solution);
F = subs(collocate(Fopt),solution);
```

44.4 Plot result

```
subplot(2,1,1)
plot(t,v/1e3,'*-',t,h/1e5,'*-',t,m/1e2,'*-');
legend('v','h','m');
title('Goddard Rocket state variables');

subplot(2,1,2)
plot(t,F,'+-');
legend('F');
title('Goddard Rocket control');
```



45 Goddard Rocket, Maximum Ascent, Final time fixed, Singular solution

Example 7.2 (ii) from the paper: H. Maurer, "Numerical solution of singular control problems using multiple shooting techniques", Journal of Optimization Theory and Applications, Vol. 18, No. 2, 1976, pp. 235-257

Remark: You can vary the fixed final time t_f to obtain Fig. 8 in the paper

L.G. van Willigenburg, W.L. de Koning

45.1 Problem setup

```
toms t

% Parameters
aalpha = 0.01227; bbeta = 0.145e-3; c = 2060; g0 = 9.81;
r0 = 6.371e6; r02=r0*r0; m0 = 214.839; mf = 67.9833; Fm = 9.525515;
t_f = 100; %Paper value 206.661;
```

45.2 Solve the problem, using a successively larger number of collocation points

```
nvec = [20 40 60];
for i=1:length(nvec);

    n = nvec(i);

    p = tomPhase('p', t, 0, t_f, n);
    setPhase(p);
    tomStates h v m
    tomControls F

    % Initial guess
    if i==1
        x0 = {icollocate({v == 10*t; h == 10*t^2
            m == m0+(t/t_f)*(mf-m0)})
            collocate(F == Fm)};
    else
        x0 = {icollocate({v == vopt; h == hopt
            m == mopt})
            collocate(F == Fopt)};
    end
```

```
% Box constraints
cbox = {icollocate({0 <= v; 0 <= h
    mf <= m <= m0
    0 <= F <= Fm})};

% Boundary constraints
cbnd = {initial({v == 0; h == 0; m == m0})
    final({m == mf})};

D = aalpha*v.^2.*exp(-bbeta*h);
g = g0; % or g0*r02./(r0+h).^2;

% ODEs and path constraints
ceq = collocate({dot(h) == v
    m*dot(v) == F*c-D-g*m
    dot(m) == -F});

% Objective
objective = -final(h);
```

45.3 Solve the problem

```
options = struct;
options.name = 'Goddard Rocket';
if i==1
    options.solver = 'multiMin';
    options.xInit = 20;
end
%options.scale = 'auto'
solution = ezsolve(objective, {cbox, cbnd, ceq}, x0, options);

% Optimal v and more to use as starting guess
vopt = subs(v, solution);
hopt = subs(h, solution);
mopt = subs(m, solution);
Fopt = subs(F, solution);
```

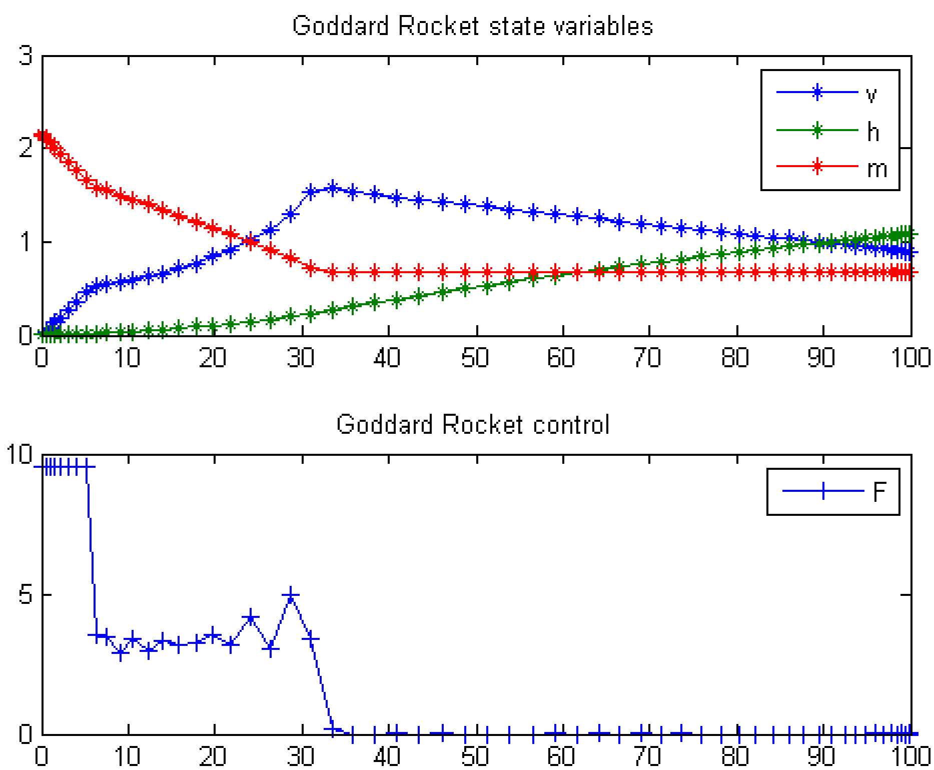
```
Problem type appears to be: lpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Goddard Rocket - Trial 1       f_k -108076.039985832960000000
                                      sum(|constr|)      0.000067118000448545
                             f(x_k) + sum(|constr|)-108076.039918714960000000
```

```
Solver: multiMin with local solver snopt.  EXIT=4.  INFORM=5.
Find local optima using multistart local search
Nonlinear infeasible problem. TotFuncEv 1. TotConstrEv 209

FuncEv    1 ConstrEv  209 ConJacEv  209 Iter   76
CPU time: 0.421875 sec. Elapsed time: 0.453000 sec.
Warning: Solver returned ExitFlag = 4
The returned solution may be incorrect.


Problem type appears to be: lpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Goddard Rocket                 f_k -108220.931724708310000000
                                       sum(|constr|)      0.000369534980940259
                              f(x_k) + sum(|constr|)-108220.931355173320000000
                                              f(x_0)-108076.039985832410000000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv   20 ConJacEv   20 Iter   19 MinorIter  536
CPU time: 0.234375 sec. Elapsed time: 0.235000 sec.


Problem type appears to be: lpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Goddard Rocket                 f_k -108245.171256515870000000
                                       sum(|constr|)      0.000090842481984530
                              f(x_k) + sum(|constr|)-108245.171165673380000000
                                              f(x_0)-108220.931724708060000000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv   33 ConJacEv   33 Iter   22 MinorIter 1129
CPU time: 0.656250 sec. Elapsed time: 0.672000 sec.


end
```

```
t = subs(collocate(t),solution);
v = subs(collocate(vopt),solution);
h = subs(collocate(hopt),solution);
m = subs(collocate(mopt),solution);
F = subs(collocate(Fopt),solution);
```

45.4 Plot result

```
subplot(2,1,1)
plot(t,v/1e3,'*-',t,h/1e5,'*-',t,m/1e2,'*-');
legend('v','h','m');
title('Goddard Rocket state variables');

subplot(2,1,2)
plot(t,F,'+-');
legend('F');
title('Goddard Rocket control');
```



46 Greenhouse Climate Control

Greenhouse Optimal Climate Control, a problem with external inputs

46.1 Problem description

Taken from the book: Optimal Control of Greenhouse Cultivation G. van Straten, R.J.C. van Ooteghem, L.G. van Willigenburg, E. van Henten

ISBN: 9781420059618 CRC Pr I Llc Books

Programmers: Gerard Van Willigenburg (Wageningen University)

46.2 Problem setup

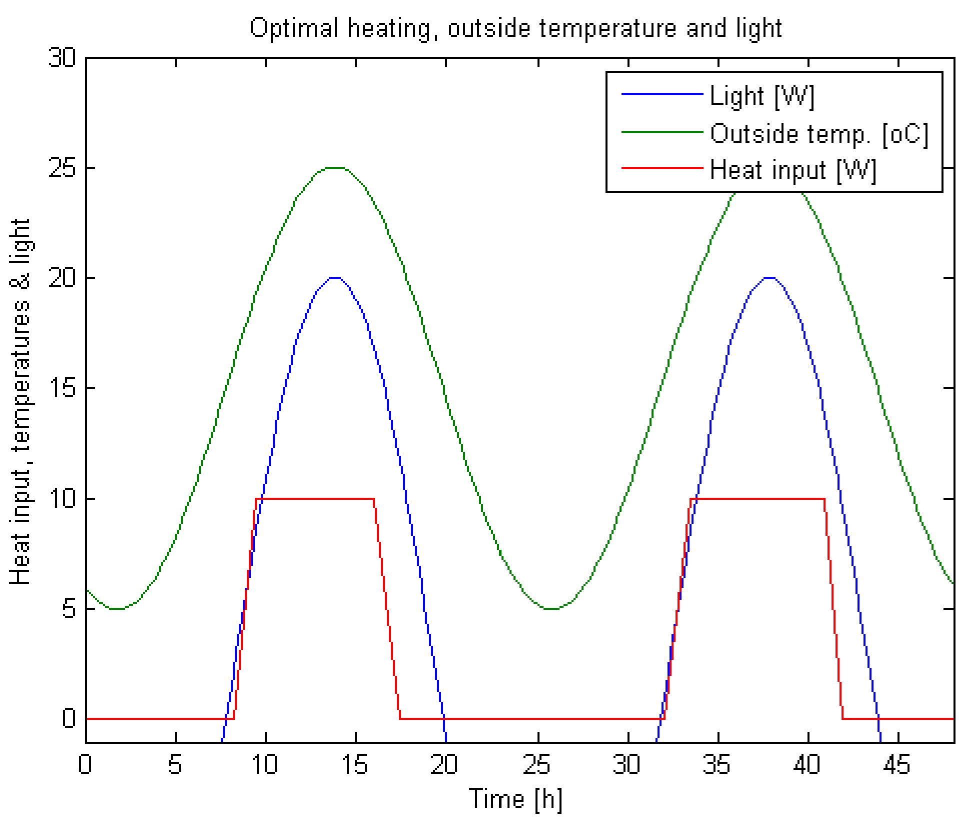
```
% Define tomSym variable t (time) and t_f (fixed final time)
toms t; t_f = 48;

% Define and set time axis
p = tomPhase('p', t, 0, t_f, 50);
setPhase(p);

% Define the state and control variables
tomStates x1 x2 x3
tomControls u
x = [x1; x2; x3];

% Initial state
xi = [0; 10; 0];

% Initial guess
x0 = {icollocate(x == xi); collocate(u == 0)};

% Boundary conditions
cbnd = initial(x == xi);

% Equality constraints: state-space diffenrential equations
pW = 3e-6/40; pT = 1; pH = 0.1;
pHc = 7.5e-2/220; pWc = 3e4/220;

% External inputs: [time, sunlight, outside temperature]
te  = (-1:0.2:49)';
tue = [te 800*sin(4*pi*te/t_f-0.65*pi) 15+10*sin(4*pi*te/t_f-0.65*pi)];
```

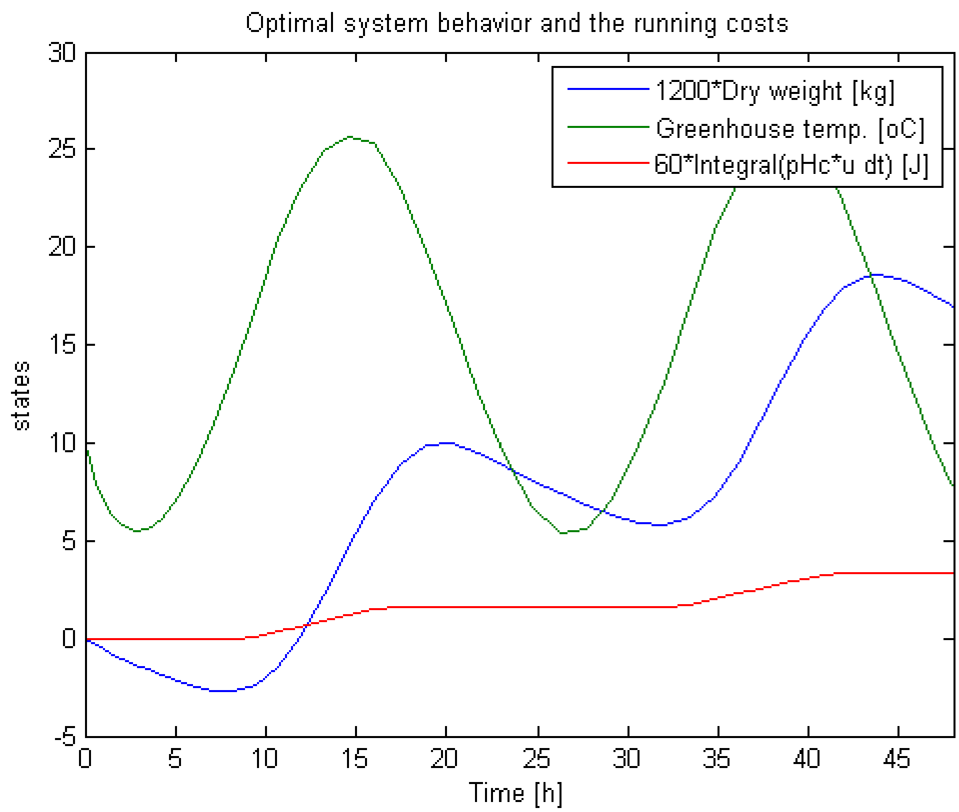
```
% Extract external inputs from table tue through interpolation
ue1 = interp1(tue(:,1),tue(:,2),t);
ue2 = interp1(tue(:,1),tue(:,3),t);

%Differential equations
ceq = collocate({
    dot(x1) == pW*ue1*x2
    dot(x2) == pT*(ue2-x2)+pH*u;
    dot(x3) == pHc*u});

% Control bounds
cbox = {0 <= collocate(u) <= 10};

% Cost function to be minimized
objective = final(x3-pWc*x1);
```

46.3 Solve the problem

```
options = struct;
options.name = 'Greenhouse Problem';
solution = ezsolve(objective, {cbox, cbnd, ceq}, x0, options);

% Obtain final solution t,x1,...,u,..
% that overwrite the associated tomSym variables
t  = subs(collocate(t),solution);
x1 = subs(collocate(x1),solution);
x2 = subs(collocate(x2),solution);
x3 = subs(collocate(x3),solution);
u = subs(collocate(u),solution);

%Plot external inputs and control
figure(1);
plot(tue(:,1),tue(:,2)/40,tue(:,1),tue(:,3),t,u); axis([0 t_f -1 30]);
xlabel('Time [h]');
ylabel('Heat input, temperatures & light');
legend('Light [W]','Outside temp. [oC]','Heat input [W]');
title('Optimal heating, outside temperature and light');

% Plot the optimal state
figure(2)
sf1=1200; sf3=60;
plot(t,[sf1*x1 x2 sf3*x3]); axis([0 t_f -5 30]);
xlabel('Time [h]'); ylabel('states');
legend('1200*Dry weight [kg]','Greenhouse temp. [oC]','60*Integral(pHc*u dt) [J]');
title('Optimal system behavior and the running costs');
```

```
Problem type appears to be: lp
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Greenhouse Problem             f_k      -1.870359095786537500
                                    sum(|constr|)      0.000000000000137357
                           f(x_k) + sum(|constr|)     -1.870359095786400000
                                           f(x_0)      0.000000000000000000

Solver: CPLEX.  EXIT=0.  INFORM=1.
CPLEX Dual Simplex LP solver
Optimal solution found

FuncEv  187 Iter  187
CPU time: 0.031250 sec. Elapsed time: 0.031000 sec.
```




Optimal system behavior and the running costs
states
Time [h]
1200*Dry weight [kg]
Greenhouse temp. [oC]
60*Integral(pHc*u dt) [J]
30
25
20
15
10
5
0
-5
0
5
10
15
20
25
30
35
40
45

47 Grusins Metric

A Short Introduction to Optimal Control, Ugo Boscain, SISSA, Italy

3.4 A Detailed Application: the Grusin's Metric

47.1 Problem Description

Find u over t in [0; 1] to minimize:

$$J = \int_0^1 u_1^2 + u_2^2 \mathrm{d}t$$

subject to:

$$\frac{dx_1}{dt} = u_1$$
$$\frac{dx_2}{dt} = u_2 * x_1$$
$$x(t_0) = [0 \ 0]$$
$$x(t_f) = [-0.001 \ \ -1]$$

Reference: [7]

47.2 Problem setup

```
toms t
p = tomPhase('p', t, 0, 1, 60);
setPhase(p);

tomStates x1 x2
tomControls u1 u2

% Boundary constraints
cbnd = {initial({x1 == 0; x2 == 0})
    final({x1 == -0.01; x2 == -1})};

% ODEs and path constraints
ceq = collocate({dot(x1) == u1
```

```
    dot(x2) == u2.*x1});

% Objective
objective = integrate(u1.^2+u2.^2);
```

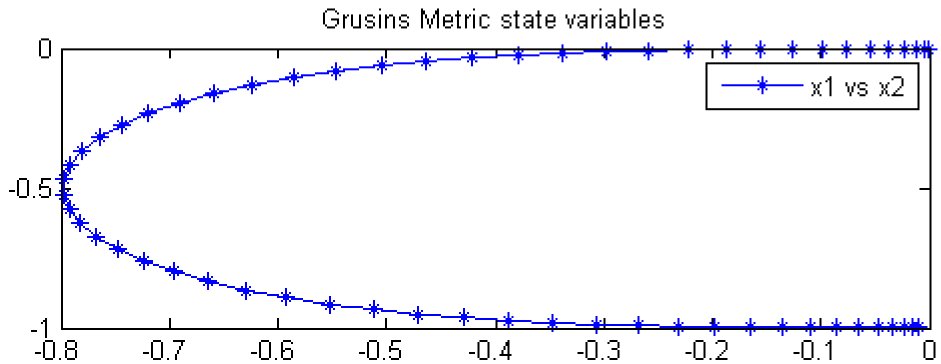
47.3 Solve the problem

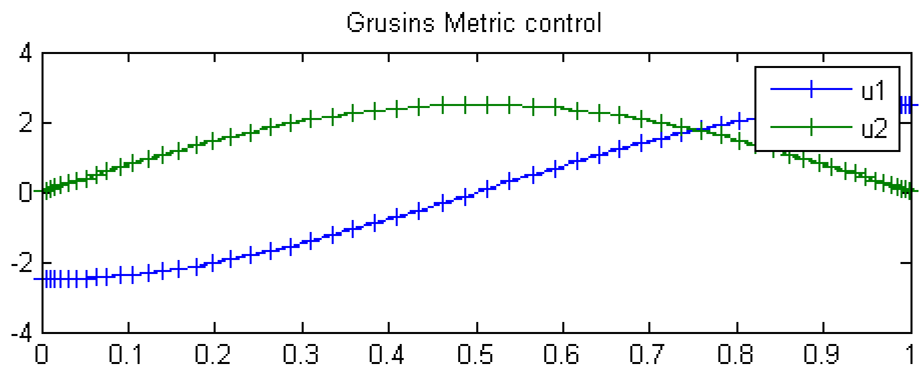
```
options = struct;
options.name = 'Grusins Metric';
solution = ezsolve(objective, {cbnd, ceq}, [], options);
t  = subs(collocate(t),solution);
x1 = subs(collocate(x1),solution);
x2 = subs(collocate(x2),solution);
u1 = subs(collocate(u1),solution);
u2 = subs(collocate(u2),solution);
```

```
Problem type appears to be: qpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Grusins Metric                 f_k       6.233154129251588800
                                      sum(|constr|)      0.000000060297919979
                             f(x_k) + sum(|constr|)      6.233154189549508400
                                             f(x_0)      0.000000000000000000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv   45 ConJacEv   45 Iter   38 MinorIter  271
CPU time: 0.593750 sec. Elapsed time: 0.610000 sec.
```

47.4 Plot result

```
subplot(2,1,1)
plot(x1,x2,'*-');
legend('x1 vs x2');
title('Grusins Metric state variables');

subplot(2,1,2)
plot(t,u1,'+-',t,u2,'+-');
legend('u1','u2');
title('Grusins Metric control');
```


Grusins Metric state variables
x1 vs x2
0
-0.5
-1
-0.8
-0.7
-0.6
-0.5
-0.4
-0.3
-0.2
-0.1
0


Grusins Metric control
u1
u2
4
2
0
-2
-4
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

48 Hang Glider Control

Benchmarking Optimization Software with COPS Elizabeth D. Dolan and Jorge J. More ARGONNE NATIONAL LABORATORY

48.1 Problem Formulation

Find u(t) over t in [0; t_F] to maximize

$$J = x$$

subject to:

$$\frac{d^2x}{dt^2} = \frac{1}{m} * (-L * sin(neta) - D * cos(neta))$$
$$\frac{d^2y}{dt^2} = \frac{1}{m} * (L * cos(neta) - D * sin(neta)) - g$$
$$sin(neta) = \frac{w}{v}$$
$$cos(neta) = \frac{\frac{dx}{dt}}{v}$$
$$v = \sqrt{(\frac{dx}{dt})^2 + w^2}$$
$$w = \frac{dy}{dt} - u$$
$$u = u_0 * (1 - r) * exp^{-r}$$
$$r = (\frac{x}{r_0} - 2.5)^2$$
$$u_0 = 2.5$$
$$r_0 = 100$$

$$D = \frac{1}{2} * (c_0 + c_1 * *c_L^2) * rho * S * v^2$$
$$L = \frac{1}{2} * c_L * rho * S * v^2$$
$$c_0 = 0.034$$
$$c_1 = 0.069662$$

$$S = 14$$

$$rho = 1.13$$

$$0 <= c_L <= c_{max}$$

$$x >= 0$$

$$\frac{dx}{dt} >= 0$$

$$c_{max} = 1.4$$

$$m = 100$$

$$g = 9.81$$

$$[x_0 \ y_0] = [0 \ 1000]$$

$$[y_{t_f}] = 900$$

$$[\frac{dx}{dt}_0 \ \frac{dy}{dt}_0] = [13.23 \ -1.288]$$

$$[\frac{dx}{dt}_{t_f} \ \frac{dy}{dt}_{t_f}] = [13.23 \ -1.288]$$

cL is the control variable.

Reference: [14]

48.2 Problem setup

```
toms t
toms t_f

for n=[10 80]

    p = tomPhase('p', t, 0, t_f, n, [], 'cheb');
    setPhase(p);

    tomStates x dx y dy
    tomControls cL

    % Initial guess
    % Note: The guess for t_f must appear in the list before
    % expression involving t.
    if n == 10
        x0 = {t_f == 105
            icollocate({
            dx == 13.23; x  == dx*t
            dy == -1.288; y  == 1000+dy*t
            })
```

```
        collocate(cL==1.4)};
else
    x0 = {t_f == tf_opt
        icollocate({
        dx == dx_opt; x  == x_opt
        dy == dy_opt; y  == y_opt
        })
        collocate(cL == cL_opt)};
end

% Box constraints
cbox = {
    0.1 <= t_f <= 200
    0   <= icollocate(x)
    0   <= icollocate(dx)
    0   <= collocate(cL) <= 1.4};

% Boundary constraints
cbnd = {initial({x  == 0; dx == 13.23
    y  == 1000; dy == -1.288})
    final({dx == 13.23; y  == 900; dy == -1.288})};

% Various constants and expressions
m = 100;      g = 9.81;
uc = 2.5;     r0 = 100;
c0  = 0.034;  c1  = 0.069662;
S   = 14;     rho = 1.13;

r = (x/r0-2.5).^2;
u = uc*(1-r).*exp(-r);
w = dy-u;
v = sqrt(dx.^2+w.^2);
sinneta = w./v;
cosneta = dx./v;
D = 1/2*(c0+c1*cL.^2).*rho.*S.*v.^2;
L = 1/2*cL.*rho.*S.*v.^2;

% ODEs and path constraints
ceq = collocate({
    dot(x)  == dx
    dot(dx) == 1/m*(-L.*sinneta-D.*cosneta)
    dot(y)  == dy
    dot(dy) == 1/m*(L.*cosneta-D.*sinneta)-g
    dx.^2+w.^2 >= 0.01});

% Objective
objective = -final(x);
```

48.3 Solve the problem

```
options = struct;
options.name = 'Hang Glider';
solution = ezsolve(objective, {cbox, cbnd, ceq}, x0, options);
```

```
Problem type appears to be: lpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Hang Glider                   f_k   -1281.388593956430400000
                                     sum(|constr|)      0.000000000082304738
                              f(x_k) + sum(|constr|)  -1281.388593956348100000
                                            f(x_0)  -1389.149999999999600000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv   55 ConJacEv   55 Iter   37 MinorIter  191
CPU time: 0.250000 sec. Elapsed time: 0.250000 sec.


Problem type appears to be: lpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Hang Glider                   f_k   -1305.253702077266800000
                                     sum(|constr|)      0.000000045646790482
                              f(x_k) + sum(|constr|)  -1305.253702031619900000
                                            f(x_0)  -1281.388593956420700000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv   80 ConJacEv   80 Iter   67 MinorIter  801
CPU time: 4.468750 sec. Elapsed time: 4.547000 sec.
```

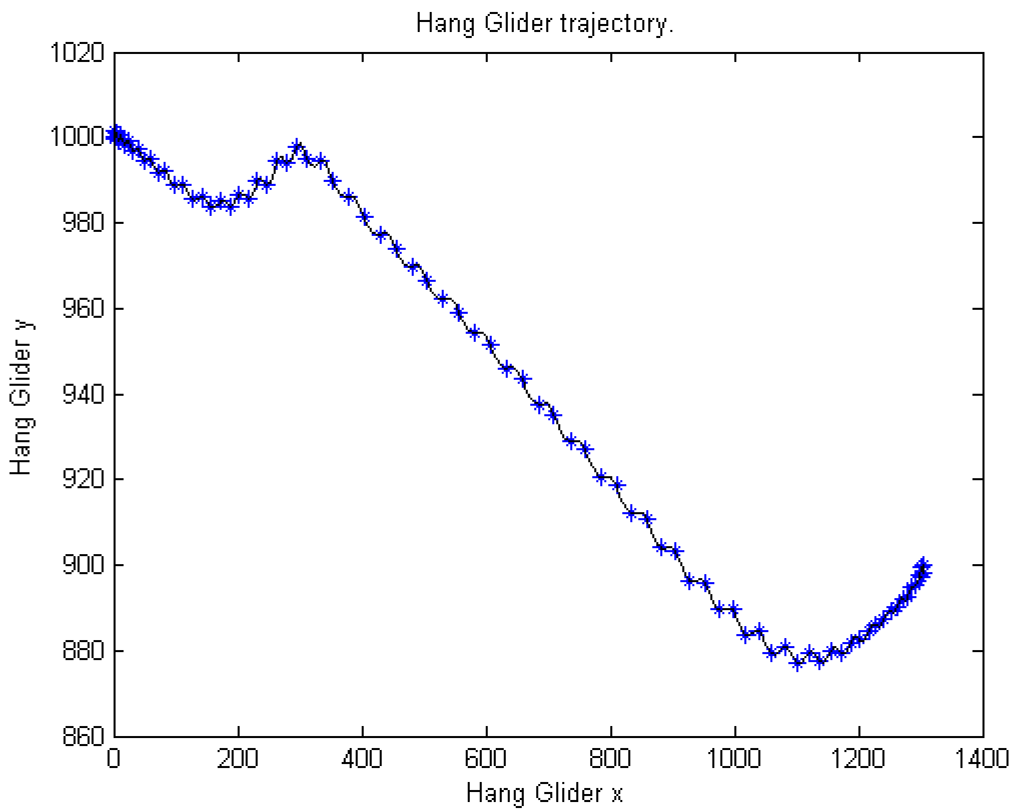
48.4 Extract optimal states and controls from solution

```
x_opt  = subs(x,solution);
dx_opt = subs(dx,solution);
y_opt  = subs(y,solution);
```

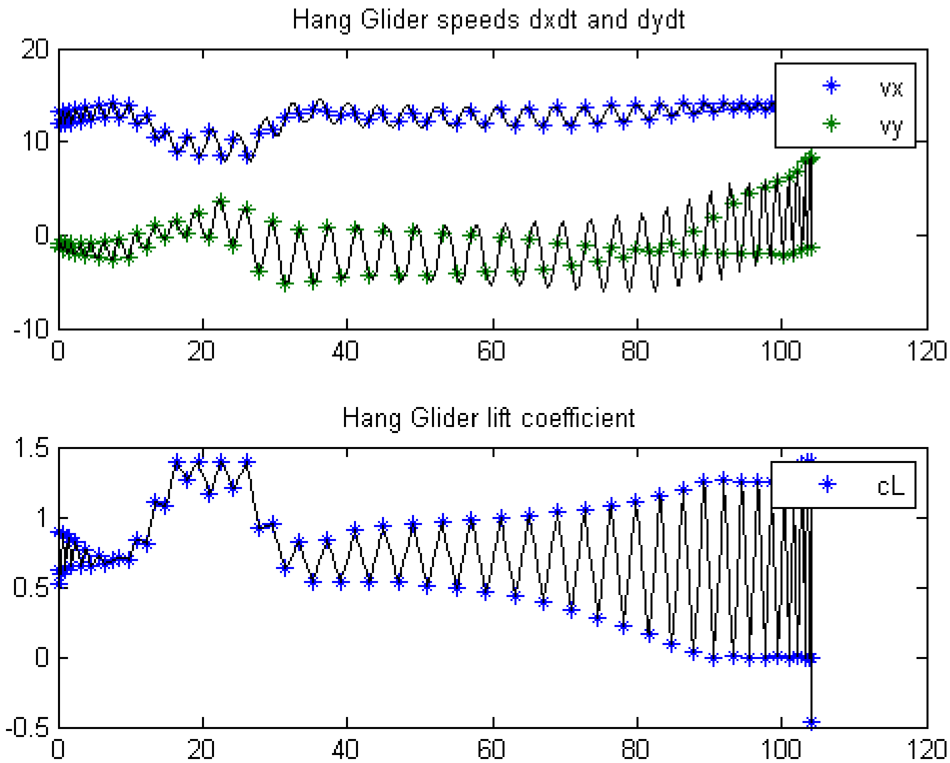
```
    dy_opt = subs(dy,solution);
    cL_opt = subs(cL,solution);
    tf_opt = subs(t_f,solution);

end
```

48.5 Plot result

```
figure(1)
ezplot(x,y);
xlabel('Hang Glider x');
ylabel('Hang Glider y');
title('Hang Glider trajectory.');

figure(2)
subplot(2,1,1)
ezplot([dx; dy]);
legend('vx','vy');
title('Hang Glider speeds dxdt and dydt');

subplot(2,1,2)
ezplot(cL);
legend('cL');
title('Hang Glider lift coefficient');
```



Hang Glider trajectory.
1020
1000
980
960
940
920
900
880
860
Hang Glider y
0
200
400
600
800
1000
1200
1400
Hang Glider x


Hang Glider speeds dxdt and dydt
20
10
0
-10
0
20
40
60
80
100
120
vx
vy
Hang Glider lift coefficient
1.5
1
0.5
0
-0.5
0
20
40
60
80
100
120
cL

49 Hanging Chain

Benchmarking Optimization Software with COPS Elizabeth D. Dolan and Jorge J. More ARGONNE NATIONAL LABORATORY

49.1 Problem Formulation

Find x(t) over t in [0; 1] to minimize

$$J = \int_0^1 x * \sqrt{1 + (\frac{dx}{dt})^2} \mathrm{d}t$$

subject to:

$$\int_0^1 \sqrt{1 + (\frac{dx}{dt})^2} \mathrm{d}t = 4$$
$$x_0 = 1$$
$$x_1 = 3$$

Reference: [14]

49.2 Problem setup

```
toms t
p = tomPhase('p', t, 0, 1, 30);
setPhase(p);

tomStates x

% Initial guess
a = 1; b = 3;
x0 = icollocate(x == 2*abs(b-a)*t.*(t-2*(0.25+(b<a)*0.5))+1);

% Constraints
con = {initial(x) == a
    final(x) == b
    integrate(sqrt(1+dot(x).^2)) == 4};

% Objective
objective = integrate(x.*sqrt(1+dot(x).^2));
```

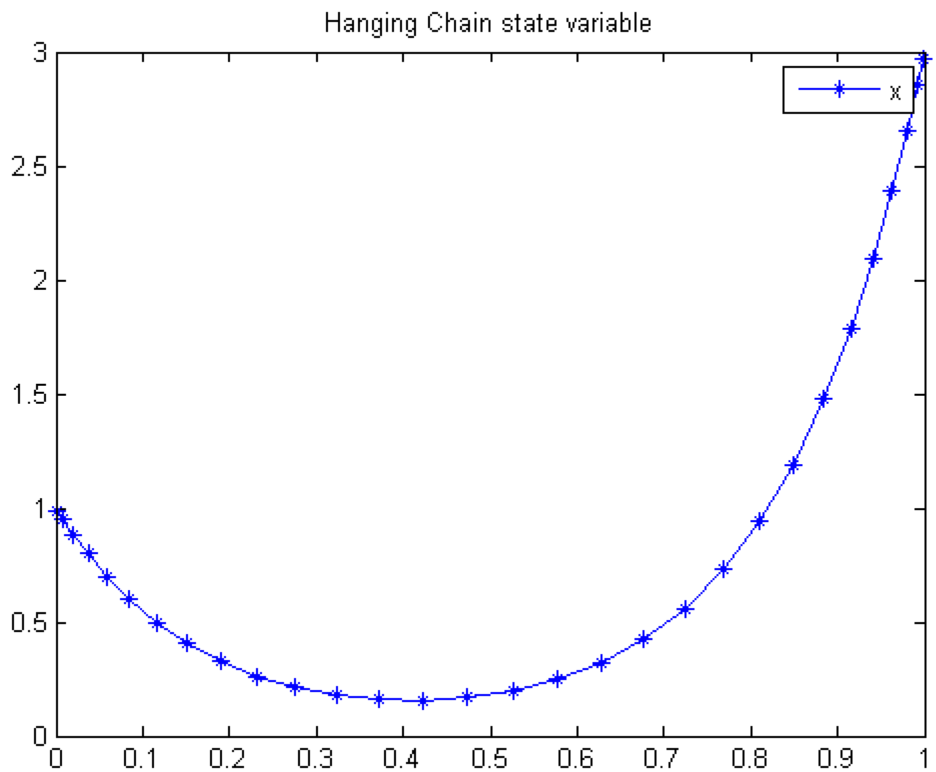
49.3 Solve the problem

```
options = struct;
options.name = 'Hanging Chain';
solution = ezsolve(objective, con, x0, options);
t = subs(collocate(t),solution);
x = subs(collocate(x),solution);
```

```
Problem type appears to be: con
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Hanging Chain                  f_k       5.068480111000088300
                                       sum(|constr|)      0.000000000009249268
                              f(x_k) + sum(|constr|)      5.068480111009337800
                                              f(x_0)      4.742150260697741300

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv  295 GradEv  293 ConstrEv  293 ConJacEv  293 Iter  244 MinorIter  279
CPU time: 0.484375 sec. Elapsed time: 0.484000 sec.
```

49.4 Plot result

```
figure(1)
plot(t,x,'*-');
legend('x');
title('Hanging Chain state variable');
```


Hanging Chain state variable
x
0
0.5
1
1.5
2
2.5
3
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

50 High Dimensional Control

Problem 7: DYNOPT User's Guide version 4.1.0

M. Cizniar, M. Fikar, M. A. Latifi, MATLAB Dynamic Optimisation Code DYNOPT. User's Guide, Technical Report, KIRP FCHPT STU Bratislava, Slovak Republic, 2006.

50.1 Problem description

Find u over t in [0; 0.2] to minimize

$$\int_0^{0.2} 5.8 * (q * x_1 - u_4) - 3.7 * u_1 - 4.1 * u_2 +$$
$$q * (23 * x_4 + 11 * x_5 + 28 * x_6 + 35 * x_7) - 5.0 * u_3^2 - 0.099 \mathrm{d}t$$

subject to:

$$\frac{dx_1}{dt} = u_4 - q * x_1 - 17.6 * x_1 * x_2 - 23 * x_1 * x_6 * u_3$$
$$\frac{dx_2}{dt} = u_1 - q * x_2 - 17.6 * x_1 * x_2 - 146 * x_2 * x_3$$
$$\frac{dx_3}{dt} = u_2 - q * x_3 - 73 * x_2 * x_3$$
$$\frac{dx_4}{dt} = -q * x_4 + 35.2 * x_1 * x_2 - 51.3 * x_4 * x_5$$
$$\frac{dx_5}{dt} = -q * x_5 + 219 * x_2 * x_3 - 51.3 * x_4 * x_5$$
$$\frac{dx_6}{dt} = -q * x_6 + 102.6 * x_4 * x_5 - 23 * x_1 * x_6 * u_3$$
$$\frac{dx_7}{dt} = -q * x_7 + 46 * x_1 * x_6 * u_3$$

where

$$q = u_1 + u_2 + u_4$$
$$x(0) = [0.1883\ 0.2507\ 0.0467\ 0.0899\ 0.1804\ 0.1394\ 0.1046]'$$
$$0 <= u_1 <= 20$$

$$0 <= u_2 <= 6$$

$$0 <= u_3 <= 4$$

$$0 <= u_4 <= 20$$

Reference: [13]

50.2 Problem setup

```
toms t
p = tomPhase('p', t, 0, 0.2, 20);
setPhase(p);

tomStates x1 x2 x3 x4 x5 x6 x7
tomControls u1 u2 u3 u4

x = [x1; x2; x3; x4; x5; x6; x7];
u = [u1; u2; u3; u4];

x0i = [0.1883;0.2507;0.0467;0.0899;0.1804;0.1394;0.1046];
x0 = icollocate({x1==x0i(1),x2==x0i(2),x3==x0i(3),x4==x0i(4),x5==x0i(5),x6==x0i(6),x7==x0i(7)});

% Box constraints and boundary
uL = zeros(4,1); uU = [20;6;4;20];
cbb = {collocate(uL <= u <= uU)
    initial(x == x0i)};

% ODEs and path constraints
q = u(1)+u(2)+u(4);
ceq = collocate({
    dot(x1) == u4-q.*x1-17.6*x1.*x2-23*x1.*x6.*u3;
    dot(x2) == u1-q.*x2-17.6*x1.*x2-146*x2.*x3;
    dot(x3) == u2-q.*x3-73*x2.*x3;
    dot(x4) == -q.*x4+35.2*x1.*x2-51.3*x4.*x5;
    dot(x5) == -q.*x5+219*x2.*x3-51.3*x4.*x5;
    dot(x6) == -q.*x6+102.6*x4.*x5-23*x1.*x6.*u3;
    dot(x7) == -q.*x7+46*x1.*x6.*u3});

% Objective
objective = integrate(-(5.8*(q.*x1-u4)-3.7*u1-4.1*u2+...
    q.*(23*x4+11*x5+28*x6+35*x7)-5.0*u3.^2-0.099));
```

50.3 Solve the problem

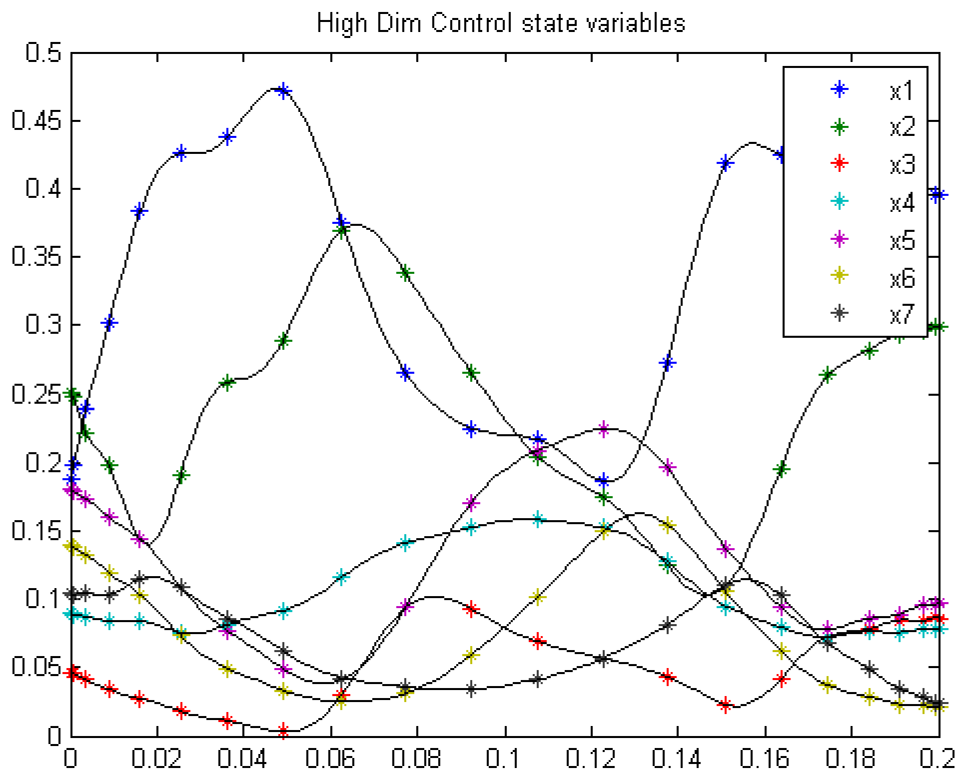
```
options = struct;
```

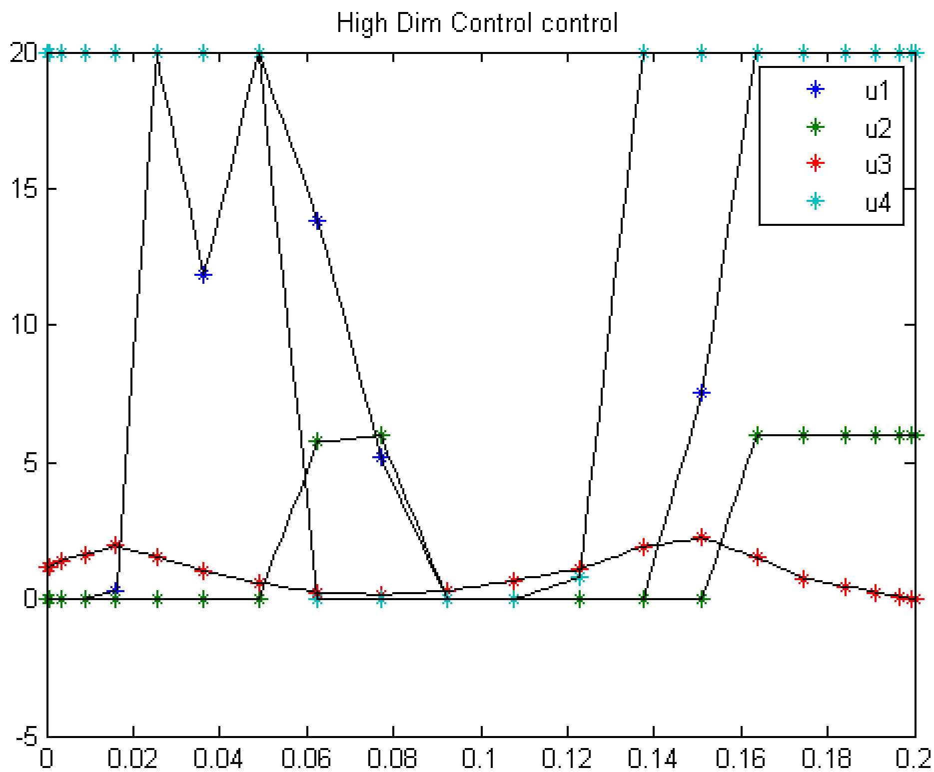
```
options.name = 'High Dim Control';
solution = ezsolve(objective, {cbb, ceq}, x0, options);
```

```
Problem type appears to be: qpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: High Dim Control               f_k     -21.834326989498084000
                                      sum(|constr|)      0.000000000215730497
                             f(x_k) + sum(|constr|)    -21.834326989282353000
                                             f(x_0)      0.000000000000000000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv   95 ConJacEv   95 Iter   88 MinorIter  488
CPU time: 1.234375 sec. Elapsed time: 1.250000 sec.
```

50.4 Plot result

```
figure(1)
ezplot(x);
legend('x1','x2','x3','x4','x5','x6','x7');
title('High Dim Control state variables');

figure(2)
ezplot(u);
legend('u1','u2','u3','u4');
title('High Dim Control control');
```


High Dim Control state variables
0.5
0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
0
0.02
0.04
0.06
0.08
0.1
0.12
0.14
0.16
0.18
0.2
x1
x2
x3
x4
x5
x6
x7


High Dim Control control
u1
u2
u3
u4
20
15
10
5
0
-5
0
0.02
0.04
0.06
0.08
0.1
0.12
0.14
0.16
0.18
0.2

51 Hyper Sensitive Optimal Control

Eigenvector approximate dichotomic basis method for solving hyper-sensitive optimal control problems 2000, Anil V. Rao and Kenneth D. Mease

3.1. Motivating example, a hyper-sensitive HBVP

51.1 Problem Formulation

Find u(t) over t in [0; t_f] to minimize

$$J = \int_0^{t_f} (x^2 + u^2) \mathrm{d}t$$

subject to:

$$\frac{dx}{dt} = -x^3 + u$$
$$x_0 = 1$$
$$x_{t_f} = 1.5$$
$$t_f = 10$$

Reference: [27]

51.2 Problem setup

```
toms t
p = tomPhase('p', t, 0, 10, 50);
setPhase(p);

tomStates x
tomControls u

% Initial guess
x0 = {icollocate(x == 0)
    collocate(u == 0)};

% bounds and ODEs
bceq = {collocate(dot(x) == -x.^3+u)
```

```
    initial(x) == 1; final(x) == 1.5};

% Objective
objective = integrate(x.^2+u.^2);
```

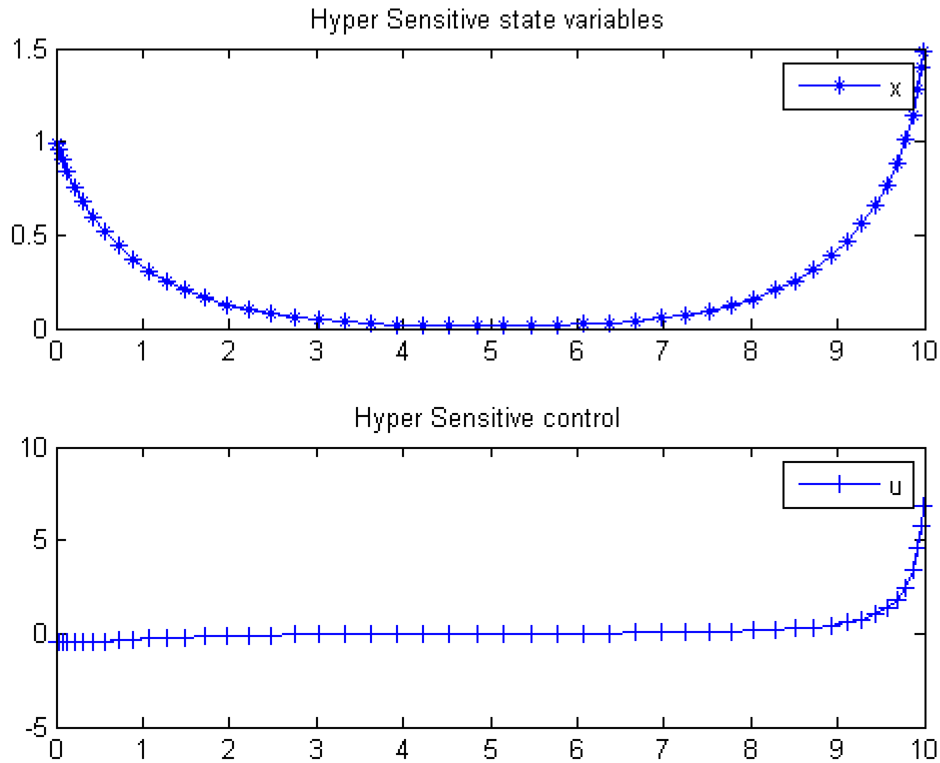
51.3 Solve the problem

```
options = struct;
options.name = 'Hyper Sensitive';
solution = ezsolve(objective, bceq, x0, options);
t = subs(collocate(t),solution);
x = subs(collocate(x),solution);
u = subs(collocate(u),solution);
```

```
Problem type appears to be: qpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Hyper Sensitive                f_k       6.723925391388356800
                                       sum(|constr|)      0.000000002440650080
                              f(x_k) + sum(|constr|)      6.723925393829007100
                                              f(x_0)      0.000000000000000000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv   26 ConJacEv   26 Iter   21 MinorIter   70
CPU time: 0.093750 sec. Elapsed time: 0.093000 sec.
```

51.4 Plot result

```
subplot(2,1,1)
plot(t,x,'*-');
legend('x');
title('Hyper Sensitive state variables');

subplot(2,1,2)
plot(t,u,'+-');
legend('u');
title('Hyper Sensitive control');
```


Hyper Sensitive state variables
x
Hyper Sensitive control
u

52 Initial Value Problem

On some linear-quadratic optimal control problems for descriptor systems. Galina Kurina, Department of Mathematics, Stockholm University, Sweden.

2.5 Necessary control optimality conditions is not valid in general case.

52.1 Problem Description

Find u over t in [0; 1] to minimize:

$$J = \frac{1}{2} * x_1^2(0.5) + \frac{1}{2} * x_1^2(1) + \frac{1}{2} * \int_0^1 u^2 \mathrm{d}t$$

subject to:

$$\frac{dx_1}{dt} = x_3 + u$$
$$\frac{dx_2}{dt} = x_2 - x_3 + u$$
$$x_2 = 0$$
$$x(t_0) = [5\ 0\ NaN]$$

Reference: [21]

52.2 Problem setup

```
toms t1
p1 = tomPhase('p1', t1, 0, 0.5, 20);
setPhase(p1);

tomStates x1p1 x2p1
tomControls x3p1 up1

% Initial guess
x01 = {icollocate({x1p1 == 0; x2p1 == 0})
    collocate({x3p1 == 0; up1 == 0})};

% Boundary constraints
```

```
cbnd1 = initial({x1p1 == 5; x2p1 == 0});

% ODEs and path constraints
ceq1 = collocate({
    dot(x1p1) == x3p1+up1
    dot(x2p1) == x2p1-x3p1+up1
    dot(x2p1) == 0});

% Objective
objective1 = 0.5*final(x1p1)^2+0.5*integrate(up1.^2);

toms t2
p2 = tomPhase('p2', t2, 0.5, 0.5, 20);
setPhase(p2);

tomStates x1p2 x2p2
tomControls x3p2 up2

% Initial guess
x02 = {icollocate({x1p2 == 0; x2p2 == 0})
    collocate({x3p2 == 0; up2 == 0})};

% ODEs and path constraints
ceq2 = collocate({
    dot(x1p2) == x3p2+up2
    dot(x2p2) == x2p2-x3p2+up2
    dot(x2p2) == 0});

% Objective
objective2 = 0.5*final(x1p2)^2+0.5*integrate(up2.^2);
objective = objective1 + objective2;

% Link phase
link = {final(p1,x1p1) == initial(p2,x1p2)
    final(p1,x2p1) == initial(p2,x2p2)
    final(p1,x3p1) == initial(p2,x3p2)};
```

52.3 Solve the problem

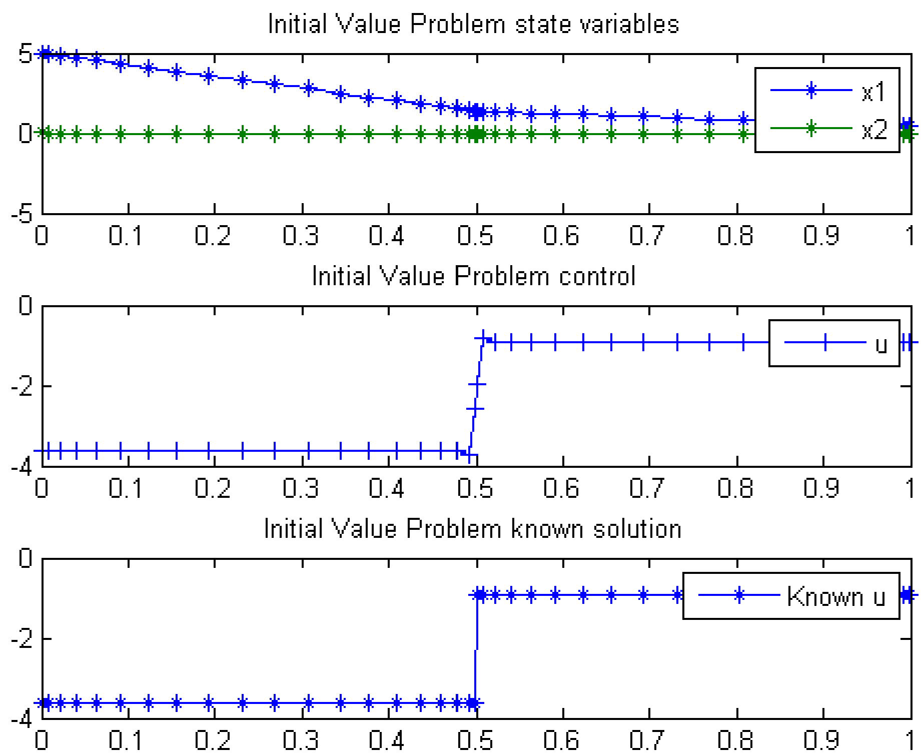
```
options = struct;
options.name = 'Initial Value Problem';
options.solver = 'snopt';
constr = {cbnd1, ceq1, ceq2, link};
solution = ezsolve(objective, constr, {x01, x02}, options);
```

```
Problem type appears to be: qp
```

```
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem:  1: Initial Value Problem              f_k       4.550747663987713100
                                       sum(|constr|)      0.000000000451074017
                              f(x_k) + sum(|constr|)      4.550747664438787000
                                              f(x_0)     12.499999999999893000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 Iter   23 MinorIter  142
CPU time: 0.031250 sec. Elapsed time: 0.031000 sec.
```

52.4 Plot result

```
subplot(3,1,1)
t  = [subs(collocate(p1,t1),solution);subs(collocate(p2,t2),solution)];
x1 = [subs(collocate(p1,x1p1),solution);subs(collocate(p2,x1p2),solution)];
x2 = [subs(collocate(p1,x2p1),solution);subs(collocate(p2,x2p2),solution)];
u  = [subs(collocate(p1,up1),solution);subs(collocate(p2,up2),solution)];

plot(t,x1,'*-',t,x2,'*-');
legend('x1','x2');
title('Initial Value Problem state variables');

subplot(3,1,2)
plot(t,u,'+-');
legend('u');
title('Initial Value Problem control');

subplot(3,1,3)
plot(t,-8/11*5*(t<0.5)-2/11*5*(t>=0.5),'*-');
legend('Known u');
title('Initial Value Problem known solution');
```



Initial Value Problem state variables
x1
x2
Initial Value Problem control
u
Initial Value Problem known solution
Known u

53 Isometrization of alpha pinene

Benchmarking Optimization Software with COPS Elizabeth D. Dolan and Jorge J. More ARGONNE NATIONAL LABORATORY

53.1 Problem Formulation

Find theta over t in [0; 40000] to minimize

$$J = \sum_{i=1}^{4}\sum_{j=1}^{8}(y_{j,i} - ymeas_{j,i})^2$$

subject to:

$$\frac{dy_1}{dt} = -(theta_1 + theta_2) * y_1$$
$$\frac{dy_2}{dt} = theta_1 * y_1$$
$$\frac{dy_3}{dt} = theta_2 * y_1 - (theta_3 + theta_4) * y_3 + theta_5 * y_5$$
$$\frac{dy_4}{dt} = theta_3 * y_3$$
$$\frac{dy_5}{dt} = theta_4 * y_3 - theta_5 * y_5$$

$$theta >= 0$$
$$time_{meas} = [1230\ 3060\ 4920\ 7800\ 10680\ 15030\ 22620\ 36420]$$
$$y1_{meas} = [88.35\ 76.4\ 65.1\ 50.4\ 37.5\ 25.9\ 14.0\ 4.5]$$
$$y2_{meas} = [7.3\ 15.6\ 23.1\ 32.9\ 42.7\ 49.1\ 57.4\ 63.1]$$
$$y3_{meas} = [2.3\ 4.5\ 5.3\ 6.0\ 6.0\ 5.9\ 5.1\ 3.8]$$
$$y4_{meas} = [0.4\ 0.7\ 1.1\ 1.5\ 1.9\ 2.2\ 2.6\ 2.9]$$
$$y5_{meas} = [1.75\ 2.8\ 5.8\ 9.3\ 12.0\ 17.0\ 21.0\ 25.7]$$
$$y_0 = [100\ 0\ 0\ 0\ 0]$$

Reference: [14]

53.2 Problem setup

```
toms t theta1 theta2 theta3 theta4 theta5
```

53.3 Solve the problem, using a successively larger number collocation points

```
for n=[20 50]

    p = tomPhase('p', t, 0, 40000, n);
    setPhase(p);
    tomStates y1 y2 y3 y4 y5

    % Initial guess
    if n == 20
        x0 = {theta1 == 0; theta2 == 0
            theta3 == 0; theta4 == 0
            theta5 == 0; icollocate({
            y1 == 100; y2 == 0
            y3 == 0;   y4 == 0
            y5 == 0})};
    else
        x0 = {theta1 == theta1opt; theta2 == theta2opt
            theta3 == theta3opt; theta4 == theta4opt
            theta5 == theta5opt; icollocate({
            y1 == y1opt; y2 == y2opt
            y3 == y3opt; y4 == y4opt
            y5 == y5opt})};
    end

    % Box constraints
    cbox = {0 <= theta1; 0 <= theta2; 0 <= theta3
        0 <= theta4; 0 <= theta5};

    % Boundary constraints
    cbnd = initial({y1 == 100; y2 == 0
        y3 == 0; y4 == 0; y5 == 0});

    y1meas = [88.35; 76.4; 65.1; 50.4; 37.5; 25.9; 14.0; 4.5];
    y2meas = [7.3; 15.6; 23.1; 32.9; 42.7; 49.1; 57.4; 63.1];
    y3meas = [2.3; 4.5; 5.3; 6.0; 6.0; 5.9; 5.1; 3.8];
    y4meas = [0.4; 0.7; 1.1; 1.5; 1.9; 2.2; 2.6; 2.9];
    y5meas = [1.75; 2.8; 5.8; 9.3; 12.0; 17.0; 21.0; 25.7];
    tmeas  = [1230; 3060; 4920; 7800; 10680; 15030; 22620; 36420];

    y1err = sum((atPoints(tmeas,y1) - y1meas).^2);
    y2err = sum((atPoints(tmeas,y2) - y2meas).^2);
    y3err = sum((atPoints(tmeas,y3) - y3meas).^2);
```

```
y4err = sum((atPoints(tmeas,y4) - y4meas).^2);
y5err = sum((atPoints(tmeas,y5) - y5meas).^2);

% ODEs and path constraints
ceq = collocate({
    dot(y1) == -(theta1+theta2)*y1
    dot(y2) == theta1*y1
    dot(y3) == theta2*y1-(theta3+theta4)*y3+theta5*y5
    dot(y4) == theta3*y3
    dot(y5) == theta4*y3-theta5*y5});

% Objective
objective = y1err+y2err+y3err+y4err+y5err;
```

53.4 Solve the problem

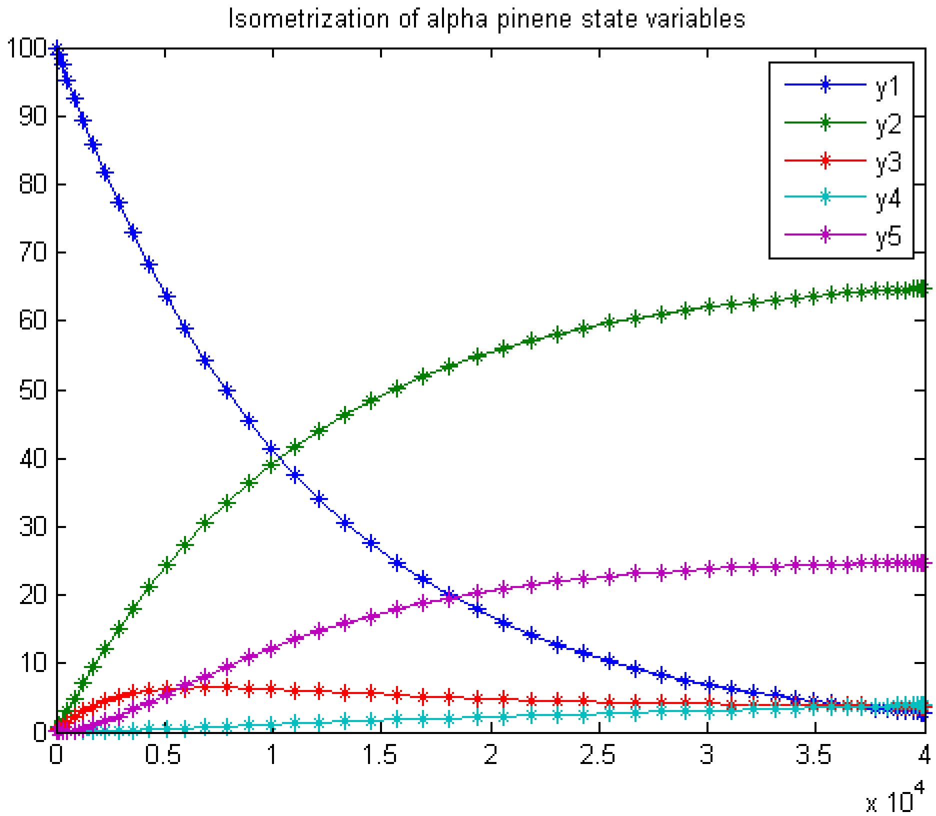
```
options = struct;
options.name = 'Isometrization of alpha pinene';
solution = ezsolve(objective, {cbox, cbnd, ceq}, x0, options);

% Optimal y and theta - starting guess in the next iteration
y1opt = subs(y1, solution);
y2opt = subs(y2, solution);
y3opt = subs(y3, solution);
y4opt = subs(y4, solution);
y5opt = subs(y5, solution);
theta1opt = subs(theta1, solution);
theta2opt = subs(theta2, solution);
theta3opt = subs(theta3, solution);
theta4opt = subs(theta4, solution);
theta5opt = subs(theta5, solution);
```

```
Problem type appears to be: qpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Isometrization of alpha pinene  f_k      19.872166933768312000
                                        sum(|constr|)      0.000000000005482115
                               f(x_k) + sum(|constr|)     19.872166933773794000
                                               f(x_0)   7569.999999999995500000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied
```

```
FuncEv    1 ConstrEv   74 ConJacEv   74 Iter   57 MinorIter  239
CPU time: 0.343750 sec. Elapsed time: 0.391000 sec.


Problem type appears to be: qpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Isometrization of alpha pinene  f_k      19.872166934168490000
                                        sum(|constr|)      0.000000000010589892
                               f(x_k) + sum(|constr|)     19.872166934179081000
                                               f(x_0) -38011.572833066202000000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv   15 ConJacEv   15 Iter   11 MinorIter  263
CPU time: 0.328125 sec. Elapsed time: 0.359000 sec.


end

t  = subs(collocate(t),solution);
y1 = collocate(y1opt);
y2 = collocate(y2opt);
y3 = collocate(y3opt);
y4 = collocate(y4opt);
y5 = collocate(y5opt);
```

53.5 Plot result

```
figure(1)
plot(t,y1,'*-',t,y2,'*-',t,y3,'*-',t,y4,'*-',t,y5,'*-');
legend('y1','y2','y3','y4','y5');
title('Isometrization of alpha pinene state variables');
```


Isometrization of alpha pinene state variables
y1
y2
y3
y4
y5
100
90
80
70
60
50
40
30
20
10
0
0
0.5
1
1.5
2
2.5
3
3.5
4
x 10^4

54 Isoperimetric Constraint Problem

54.1 Problem Formulation

Find u over t in [0; 1] to minimize

$$J = \int_0^1 x^2 \mathrm{d}t$$

subject to:

$$\frac{dx}{dt} = -sin(x) + u$$
$$\int_0^1 u^2 \mathrm{d}t = 10$$

The initial condition are:

$$x(0) = 1$$
$$x(1) = 0$$

54.2 Problem setup

```
toms t
p = tomPhase('p', t, 0, 1, 30);
setPhase(p);

tomStates x
tomControls u

% Initial guess
x0 = {icollocate(x == 0)
    collocate(u == 0)};

% Box constraints
cbox = {icollocate(-10 <= x <= 10)
```

```
    collocate(-4 <= u <= 4)};

% Boundary constraints
cbnd = {initial(x == 1)
    final(x == 0)};

% ODEs and path constraints
ceq = collocate(dot(x) == -sin(x)+u);

% Integral constraint
cint = {integrate(u^2) == 10};

% Objective
objective = integrate(x);
```

54.3 Solve the problem

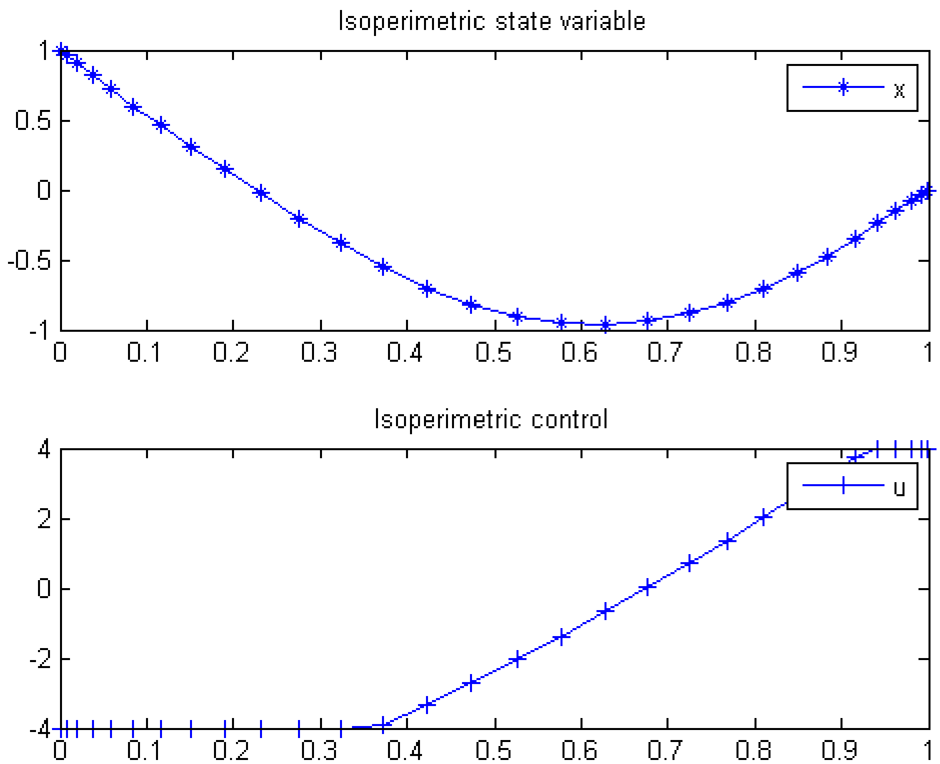
```
options = struct;
options.name = 'Isoperimetric';
solution = ezsolve(objective, {cbox, cbnd, ceq, cint}, x0, options);
t = subs(collocate(t),solution);
x = subs(collocate(x),solution);
u = subs(collocate(u),solution);
```

```
Problem type appears to be: lpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Isoperimetric                  f_k      -0.375495523108184680
                                       sum(|constr|)      0.000000031769415330
                              f(x_k) + sum(|constr|)     -0.375495491338769360
                                              f(x_0)      0.000000000000000000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv  112 ConJacEv  112 Iter   53 MinorIter  216
CPU time: 0.171875 sec. Elapsed time: 0.171000 sec.
```

54.4 Plot result

```
subplot(2,1,1)
plot(t,x,'*-');
```

```
legend('x');
title('Isoperimetric state variable');

subplot(2,1,2)
plot(t,u,'+-');
legend('u');
title('Isoperimetric control');
```



55 Jumbo Crane Container Control

55.1 Problem description

Time-optimal control of a Jumbo Container Crane avoiding an obstacle.

Crane dynamics described in the book: Informatics in control automation and robotics II DOI 10.1007/978-1-4020-5626-0 , Springer, 2007, pp.79-84. T.J.J. van den Boom, J.B. Klaassens, R. Meiland Real-time optimal control for a non linear container crane using a neural network

Optimal control problem by: W.L. De Koning, G Fitie and L.G. Van Willigenburg

Programmers: Gerard Van Willigenburg (Wageningen University) Willem De Koning (retired from Delft University of Technology)

55.2 Problem setup

```
% Array with consecutive number of collocation points
narr = [7 10 40];

toms t t_f % Free final time

toms ho1

for i=1:length(narr)

    p = tomPhase('p', t, 0, t_f, narr(i), [], 'cheb');
    setPhase(p)

    tomStates x1 x2 x3 x4 x5 x6
    tomControls u1 u2

    x = [x1; x2; x3; x4; x5; x6];

    % Crane parameters
    gr = 9.81;  He = 50;    Jh = 35.6;  Jt = 13.5;  ht = 50;
    mc = 47000; mt = 33000; Nh = 26.14; Nt = 16.15; rh = 0.6; rt = 0.5;
    xo_l = 8;    xo_r = 15;   hob = 15;

    % Initial & terminal states
    xi = [0;  0; 0; 0; 50; 0];
    xf = [50; 0; 0; 0; 50; 0];
```

```
% Initial guess
if i==1;
   x0 = {t_f==20.4; ho1==0; icollocate({x1 == 50*t/20; x2 == xi(2)
        x3 == xi(3); x4 == xi(4); x5 == xi(5); x6 == xi(6)})
        collocate({u1 == -2000; u2 == -5000})};
else
    x0 = {t_f==tfopt;
        icollocate({x1 == xopt1; x2 == xopt2
        x3 == xopt3; x4 == xopt4; x5 == xopt5; x6 == xopt6})
        collocate({u1 == uopt1; u2 == uopt2})};
end

% Box constraints
cbox = {15 <= t_f <= 30; -4200 <= collocate(u1) <= 4200
    -11490 <= collocate(u2) <= 11490};

% Boundary constraints
cbnd = {initial(x == xi), final(x == xf)};

Gt  = Jt*Nt*Nt/(rt*rt); Gh = Jh*Nh*Nh/(rh*rh);
st  = sin(x3); ct = cos(x3);
Ft  = (Nt/rt)*u1; Fh = (Nh/rh)*u2;
d2x = (mc+Gh)*Ft-mc*Fh.*st+mc*gr*Gh*st.*ct+mc*Gh*x5.*x4.*x4.*st;
d2x = d2x./((mc+Gh)*(mt+Gt)+mc*Gh*(1-ct.*ct));

% xc is the container x position against time
xc = x1+x5*sin(x3);
% hc is the height of the container against time
hc = ht-x5*cos(x3);
% ho is the height of the obstacle at the container x position.
%ho = ifThenElse(xc<=xo_l,0,ifThenElse(xc>=xo_r,0,hob));
% do is the distance to the obstacle from the container.
do = max(max(xo_l-xc,xc-xo_r),hc-ho1);

% Path constraint - Distance to obstacle should always be >= 0
% and height should always be >= 0.
% Test 300 points, evenly spaced in time.
pth = {atPoints(linspace(0,t_f,300),{do>=0,hc>=0}),ho1>=hob};

% ODEs
ode = collocate({
    dot(x1) == x2
    dot(x2) == d2x
    dot(x3) == x4
    dot(x4) == (-2*x6.*x4-gr*st-d2x.*ct)./x5
    dot(x5) == x6
    dot(x6) == (Fh+mc*x5.*x4.*x4+mc*gr*ct-mc*d2x.*st)/(mc+Gh)
```

```
        });

    % Objective
    objective = t_f;
```

55.3 Solve the problem

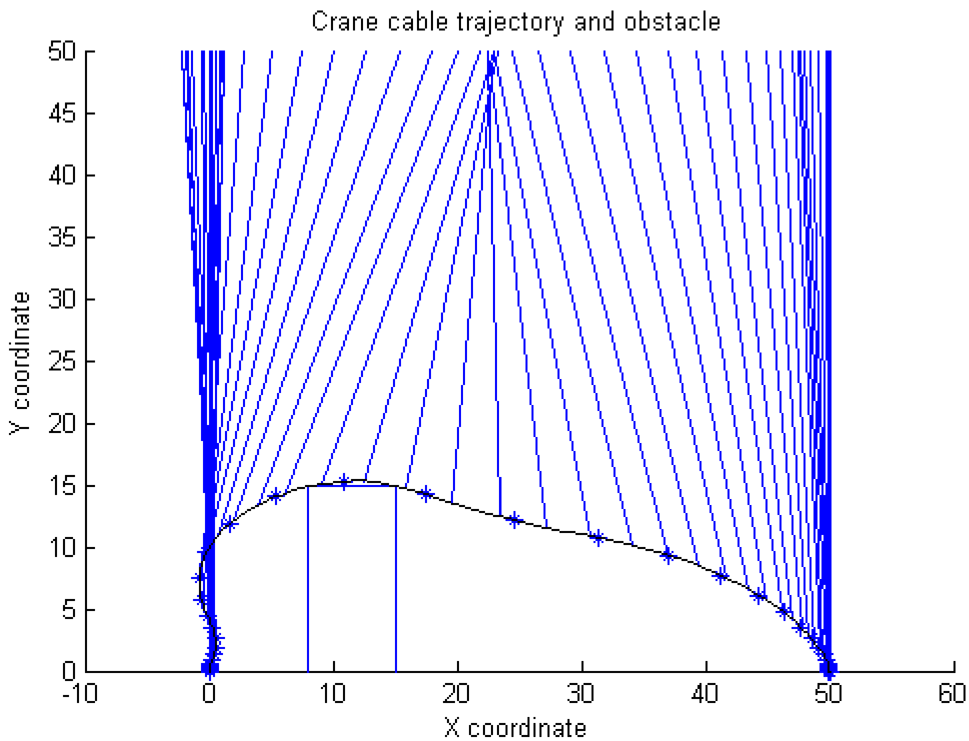
```
    options = struct;
    if i==1
        % To improve convergece, we make the obstacle constraint softer,
% by including it in the ojbective rather than as a hard constraint
% in the first iteration.
% This is necessary because of the very nonlinear properties of this
% constraint.
        pth = pth{1};
        objective = objective - 2*ho1;
    end
    options.name = 'Crane with obstacle';
    options.Prob.SOL.optPar(30) = 20000;
    solution = ezsolve(objective, {cbox, cbnd, pth, ode}, x0, options);

    tfopt = subs(t_f,solution);
    xopt1 = subs(x1,solution);
    xopt2 = subs(x2,solution);
    xopt3 = subs(x3,solution);
    xopt4 = subs(x4,solution);
    xopt5 = subs(x5,solution);
    xopt6 = subs(x6,solution);
    uopt1 = subs(u1,solution);
    uopt2 = subs(u2,solution);
```

```
Problem type appears to be: lpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Crane with obstacle            f_k     -44.290334943951777000
                                       sum(|constr|)      0.000000001680375175
                              f(x_k) + sum(|constr|)    -44.290334942271400000
                                              f(x_0)     20.399999999999999000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv  387 ConJacEv  385 Iter   56 MinorIter 7258
```

```
CPU time: 5.812500 sec. Elapsed time: 5.890000 sec.


Problem type appears to be: lpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Crane with obstacle            f_k      21.780681675931579000
                                       sum(|constr|)      0.000041663897094514
                              f(x_k) + sum(|constr|)     21.780723339828672000
                                              f(x_0)     30.000000000000000000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv  181 ConJacEv  180 Iter   62 MinorIter  768
CPU time: 2.750000 sec. Elapsed time: 2.969000 sec.


Problem type appears to be: lpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Crane with obstacle            f_k      19.487670710700066000
                                       sum(|constr|)      0.000027250297174480
                              f(x_k) + sum(|constr|)     19.487697960997242000
                                              f(x_0)     21.780681675931579000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv  531 ConJacEv  529 Iter   74 MinorIter 1118
CPU time: 20.859375 sec. Elapsed time: 21.062000 sec.
```

```
end

% Get solution for 50 points, evenly distributed in time.
nt = 50;
topt = linspace(0,subs(t_f,solution),nt);
xopt = subs(atPoints(topt,x),solution);

% Plot
[nt,nx]=size(xopt);
```

```
clf
axis([-10 60 0 50]);
axis image;

% Draw Obstacle
line([xo_l xo_l xo_r xo_r],[0 hob hob 0]);
title('Crane cable trajectory and obstacle');

xtop=xopt(:,1); ytop=50*ones(size(topt));
xbottom=xopt(:,1)+xopt(:,5).*sin(xopt(:,3));
ybottom=50-xopt(:,5).*cos(xopt(:,3));

% Draw cable trajectory
tl=5; toptlen=length(topt);
for k=1:toptlen
    line([xtop(k) xbottom(k)],[ytop(k) ybottom(k)]);
    if tl/toptlen>0.01; pause(tl/toptlen); end
end
hold on; ezplot(xc,hc); hold off
xlabel('X coordinate'); ylabel('Y coordinate');
```



Crane cable trajectory and obstacle
Y coordinate
X coordinate
50
45
40
35
30
25
20
15
10
5
0
-10
0
10
20
30
40
50
60

56 Lee-Ramirez Bioreactor

Dynamic optimization of chemical and biochemical processes using restricted second-order information 2001, Eva Balsa-Canto, Julio R. Banga, Antonio A. Alonso Vassilios S. Vassiliadis

Case Study II: Lee-Ramirez bioreactor

56.1 Problem description

This problem considers the optimal control of a fed-batch reactor for induced foreign protein production by recombinant bacteria, as presented by Lee and Ramirez (1994) and considered afterwards by Tholudur and Ramirez (1997) and Carrasco and Banga (1998). The objective is to maximize the profitability of the process using the nutrient and the inducer feeding rates as the control variables. Three different values for the ratio of the cost of inducer to the value of the protein production (Q) were considered.

The mathematical formulation, following the modified parameter function set presented by Tholudur and Ramirez (1997) to increase the sensitivity to the controls, is as follows:

Find u1(t) and u2(t) over t in [t0 t_f] to maximize:

$$J = x_4(t_f) * x_1(t_f) - Q * \int_{t_0}^{t_f} u_2 \mathrm{d}t$$

subject to:

$$\begin{aligned}
\frac{dx_1}{dt} &= u_1 + u_2 \\
\frac{dx_2}{dt} &= g_1. * x_2 - (u_1 + u_2). * \frac{x_2}{x_1} \\
\frac{dx_3}{dt} &= \frac{u_1}{x_1}. * c_1 - (u_1 + u_2). * \frac{x_3}{x_1} - g_1. * \frac{x_2}{c_2} \\
\frac{dx_4}{dt} &= g_2. * x_2 - (u_1 + u_2). * \frac{x_4}{x_1} \\
\frac{dx_5}{dt} &= \frac{u_2 * c_3}{x_1} - (u_1 + u_2). * \frac{x_5}{x_1} \\
\frac{dx_6}{dt} &= -g_3. * x_6 \\
\frac{dx_7}{dt} &= g_3. * (1 - x_7)
\end{aligned}$$

$$t_1 = 14.35 + x_3 + (\frac{x_3^2}{111.5})$$
$$t_2 = 0.22 + x_5$$
$$t_3 = x_6 + \frac{0.22}{t_2} * x_7$$

$$g_1 = \frac{x_3}{t_1} * (x_6 + x_7 * \frac{0.22}{t_2})$$
$$g_2 = 0.233 * \frac{x_3}{t_1} * (\frac{0.0005 + x_5}{0.022 + x_5})$$
$$g_3 = 0.09 * \frac{x_5}{0.034 + x_5}$$

$$c_1 = 100$$
$$c_2 = 0.51$$
$$c_3 = 4$$

where the state variables are the reactor volume (x1), the cell density (x2), the nutrient concentration (x3), the foreign protein concentration (x4), the inducer concentration (x5), the inducer shock factor on cell growth rate (x6) and the inducer recovery factor on cell growth rate (x7). The two control variables are the glucose rate (u1) and the inducer feeding rate (u2). Q is the ratio of the cost of inducer to the value of the protein production, and the final time is considered fixed as t_f = 10 h. The model parameters were described by Lee and Ramirez (1994). The initial conditions are:

$$x(t_0) = [1\ 0.1\ 40\ 0\ 0\ 1\ 0]'$$

The following constraints on the control variables are considered:

$$0 <= u1 <= 1$$
$$0 <= u2 <= 1$$

Reference: [2]

56.2 Problem setup

```
toms t

for n=[20 35 55 85]

    p = tomPhase('p', t, 0, 10, n);
    setPhase(p);

    tomStates z1 z2 z3s z4 z5 z6 z7

    % Declaring u as "states" makes it possible to work with their
    % derivatives.
    tomStates u1 u2

    % Scale z3 by 40
    z3 = z3s*40;

    % Initial guess
    if n == 20
        x0 = {icollocate({z1 == 1; z2 == 0.1
            z3 == 40; z4 == 0; z5 == 0
            z6 == 1; z7 == 0})
            icollocate({u1==t/10; u2==t/10})};
    else
        x0 = {icollocate({z1 == z1opt
            z2 == z2opt; z3 == z3opt
            z4 == z4opt; z5 == z5opt
            z6 == z6opt; z7 == z7opt})
            icollocate({u1 == u1opt
            u2 == u2opt})};
    end

    % Box constraints
    cbox = {mcollocate({0 <= z1; 0 <= z2
        0 <= z3; 0 <= z4; 0 <= z5})
        0 <= collocate(u1) <= 1
        0 <= collocate(u2) <= 1};

    % Boundary constraints
    cbnd = initial({z1 == 1; z2 == 0.1
        z3 == 40; z4 == 0; z5 == 0
        z6 == 1; z7 == 0});

    % Various constants and expressions
    c1 = 100; c2 = 0.51; c3 = 4.0;
    Q  = 0;
```

```
t1 = 14.35+z3+((z3).^2/111.5);
t2 = 0.22+z5;
t3 = z6+0.22./t2.*z7;

g1 = z3./t1.*(z6+z7*0.22./t2);
g2 = 0.233*z3./t1.*((0.0005+z5)./(0.022+z5));
g3 = 0.09*z5./(0.034+z5);

% ODEs and path constraints
ceq = collocate({
    dot(z1) == u1+u2
    dot(z2) == g1.*z2-(u1+u2).*z2./z1
    dot(z3) == u1./z1.*c1-(u1+u2).*z3./z1-g1.*z2/c2
    dot(z4) == g2.*z2-(u1+u2).*z4./z1
    dot(z5) == u2*c3./z1-(u1+u2).*z5./z1
    dot(z6) == -g3.*z6
    dot(z7) == g3.*(1-z7)});

% Objective
J = -final(z1)*final(z4)+Q*integrate(u2);

spenalty = 0.1/n; % penalty term to yield a smoother u.
objective = J + spenalty*integrate(dot(u1)^2+dot(u2)^2);
```

56.3 Solve the problem

```
options = struct;
options.name = 'Lee Bio Reactor';
solution = ezsolve(objective, {cbox, cbnd, ceq}, x0, options);

% Optimal z, u for starting point
z1opt = subs(z1, solution);
z2opt = subs(z2, solution);
z3opt = subs(z3, solution);
z4opt = subs(z4, solution);
z5opt = subs(z5, solution);
z6opt = subs(z6, solution);
z7opt = subs(z7, solution);
u1opt = subs(u1, solution);
u2opt = subs(u2, solution);
```

```
Problem type appears to be: qpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
```

```
===============================================================================
Problem: ---  1: Lee Bio Reactor                f_k      -6.158549510890046500
                                       sum(|constr|)      0.000022911257182501
                              f(x_k) + sum(|constr|)     -6.158526599632864400
                                              f(x_0)      0.001000000000000008

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv  818 ConJacEv  818 Iter  628 MinorIter 3252
CPU time: 8.656250 sec. Elapsed time: 9.078000 sec.


Problem type appears to be: qpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
===============================================================================
Problem: ---  1: Lee Bio Reactor                f_k      -6.149281872815334000
                                       sum(|constr|)      0.000001068521931031
                              f(x_k) + sum(|constr|)     -6.149280804293402600
                                              f(x_0)     -6.161206582021080200

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv  716 ConJacEv  716 Iter  639 MinorIter 2089
CPU time: 19.140625 sec. Elapsed time: 19.593000 sec.


Problem type appears to be: qpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
===============================================================================
Problem: ---  1: Lee Bio Reactor                f_k      -6.148479057333524600
                                       sum(|constr|)      0.000000747224617591
                              f(x_k) + sum(|constr|)     -6.148478310108907300
                                              f(x_0)     -6.151340834042628100

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv  457 ConJacEv  457 Iter  400 MinorIter 1890
CPU time: 36.687500 sec. Elapsed time: 37.359000 sec.
```

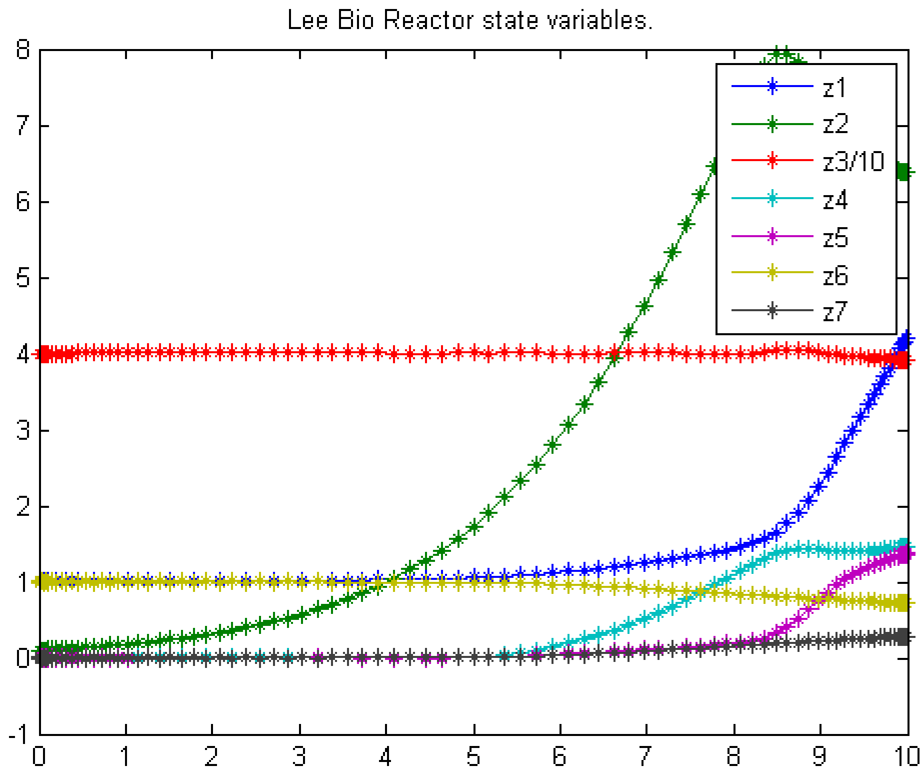
```
Problem type appears to be: qpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Lee Bio Reactor                f_k      -6.149330683460276800
                                       sum(|constr|)      0.000000648177269734
                              f(x_k) + sum(|constr|)     -6.149330035283006700
                                              f(x_0)     -6.149452704290531800

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv  392 ConJacEv  392 Iter  341 MinorIter 2079
CPU time: 144.671875 sec. Elapsed time: 155.766000 sec.
```

```
end

t  = subs(collocate(t),solution);
z1 = subs(collocate(z1),solution);
z2 = subs(collocate(z2),solution);
z3 = subs(collocate(z3),solution);
z4 = subs(collocate(z4),solution);
z5 = subs(collocate(z5),solution);
z6 = subs(collocate(z6),solution);
z7 = subs(collocate(z7),solution);
u1 = subs(collocate(u1),solution);
u2 = subs(collocate(u2),solution);
```

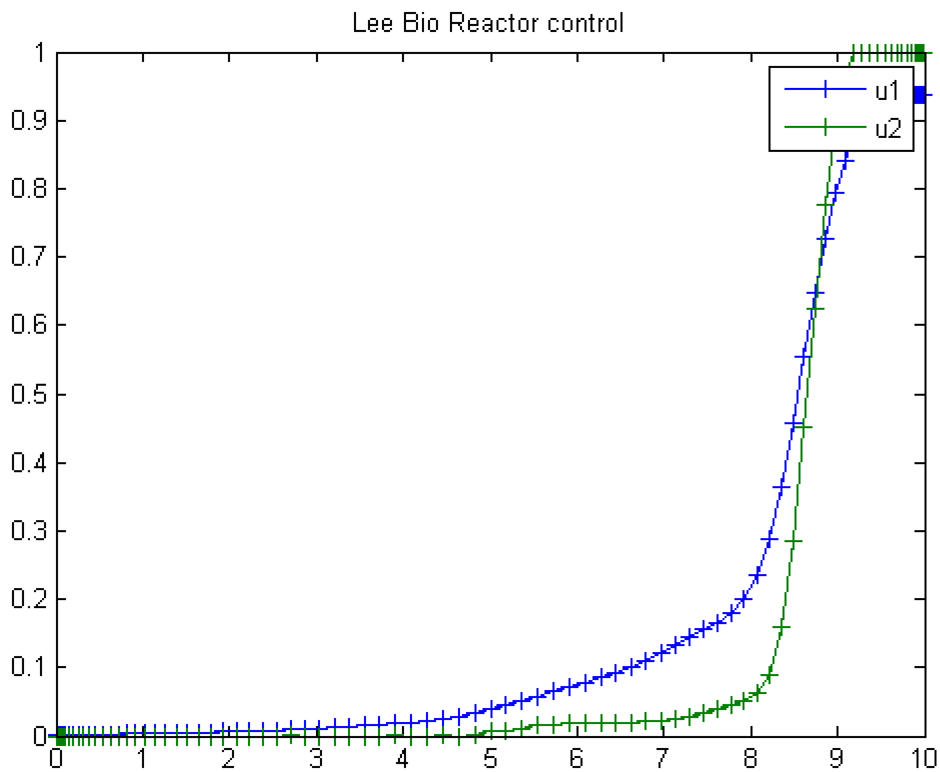
56.4 Plot result

```
figure(1)
plot(t,z1,'*-',t,z2,'*-',t,z3/10,'*-',t,z4,'*-' ...
    ,t,z5,'*-',t,z6,'*-',t,z7,'*-');
legend('z1','z2','z3/10','z4','z5','z6','z7');
title('Lee Bio Reactor state variables.');

figure(2)
plot(t,u1,'+-',t,u2,'+-');
legend('u1','u2');
title('Lee Bio Reactor control');

disp('J = ');
disp(subs(J,solution));
```

```
J =
   -6.1510
```



Lee Bio Reactor state variables.
z1
z2
z3/10
z4
z5
z6
z7
8
7
6
5
4
3
2
1
0
-1
0
1
2
3
4
5
6
7
8
9
10


Lee Bio Reactor control
u1
u2
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
1
2
3
4
5
6
7
8
9
10

57 Linear Tangent Steering Problem

Benchmarking Optimization Software with COPS Elizabeth D. Dolan and Jorge J. More ARGONNE NATIONAL LABORATORY

57.1 Problem Formulation

Find u(t) over t in [0; t_F] to minimize

$$J = t_f$$

subject to:

$$\frac{d^2y_1}{dt^2} = a * cos(u)$$
$$\frac{d^2y_2}{dt^2} = a * sin(u)$$
$$|u| <= \frac{pi}{2}$$

$$y_{1:2}(0) = 0$$
$$\frac{dy_{1:2}}{dt} = 0$$
$$a = 1$$
$$y_2(f) = 5$$
$$\frac{dy_{1:2}}{dt}(f) = [45\ 0]$$

The following transformation gives a new formulation:

$$x_1 = y_1$$
$$x_2 = \frac{dy_1}{dt}$$
$$x_3 = y_2$$
$$x_4 = \frac{dy_2}{dt}$$

$$\frac{dx_1}{dt} = x_2$$

$$\frac{dx_2}{dt} = a * cos(u)$$

$$\frac{dx_3}{dt} = x_4$$

$$\frac{dx_4}{dt} = a * sin(u)$$

Reference: [14]

57.2 Problem setup

```
toms t
toms t_f
p = tomPhase('p', t, 0, t_f, 30);
setPhase(p);

tomStates x1 x2 x3 x4
tomControls u

% Initial guess
x0 = {t_f == 1
    icollocate({
    x1 == 12*t/t_f
    x2 == 45*t/t_f
    x3 == 5*t/t_f
    x4 == 0})};

% Box constraints
cbox = {sqrt(eps) <= t_f
    -pi/2 <= collocate(u) <= pi/2};

% Boundary constraints
cbnd = {initial({x1 == 0; x2 == 0; x3 == 0; x4 == 0})
    final({x2 == 45; x3 == 5; x4 == 0})};

% ODEs and path constraints
a = 100;
ceq = collocate({dot(x1) == x2
    dot(x2) == a*cos(u)
    dot(x3) == x4
    dot(x4) == a*sin(u)});
```

```
% Objective
objective = t_f;
```

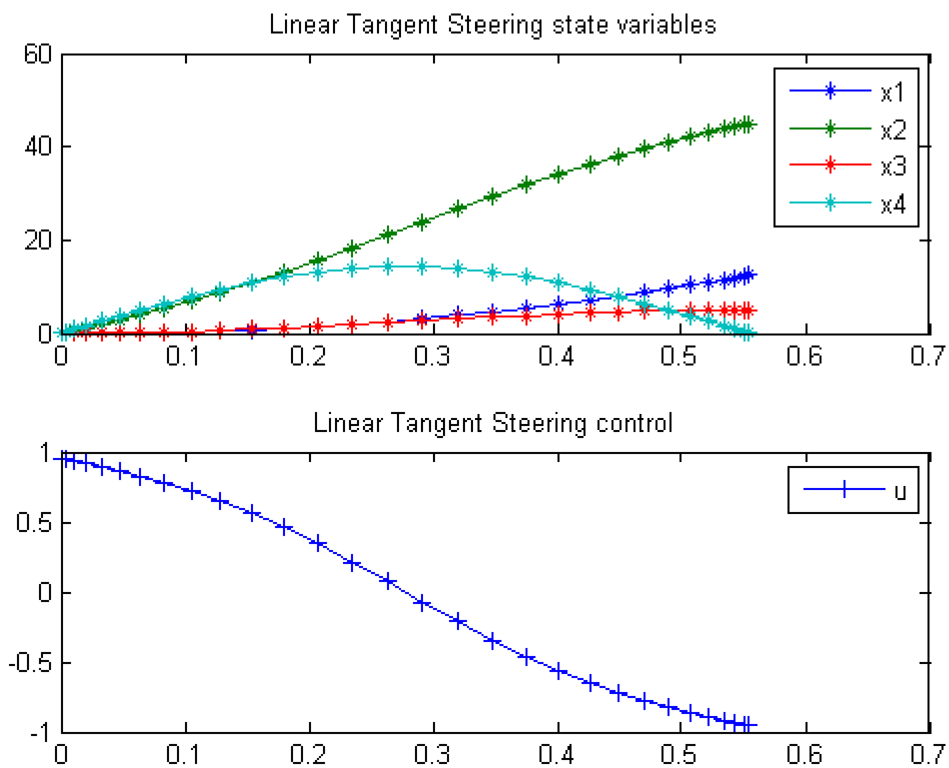
57.3 Solve the problem

```
options = struct;
options.name = 'Linear Tangent Steering';
options.solver = 'knitro';
solution = ezsolve(objective, {cbox, cbnd, ceq}, x0, options);
t  = subs(collocate(t),solution);
x1 = subs(collocate(x1),solution);
x2 = subs(collocate(x2),solution);
x3 = subs(collocate(x3),solution);
x4 = subs(collocate(x4),solution);
u  = subs(collocate(u),solution);
```

```
Problem type appears to be: lpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Linear Tangent Steering        f_k       0.554570876848855420
                                       sum(|constr|)      0.000053410901111122
                              f(x_k) + sum(|constr|)      0.554624287749966530
                                              f(x_0)      1.000000000000000000

Solver: KNITRO.  EXIT=0.  INFORM=0.
Default NLP KNITRO
Locally optimal solution found

FuncEv   14 GradEv  152 ConstrEv   13 ConJacEv  152 Iter   11 MinorIter   12
CPU time: 0.312500 sec. Elapsed time: 0.312000 sec.
```

57.4 Plot result

```
subplot(2,1,1)
plot(t,x1,'*-',t,x2,'*-',t,x3,'*-',t,x4,'*-');
legend('x1','x2','x3','x4');
title('Linear Tangent Steering state variables');

subplot(2,1,2)
plot(t,u,'+-');
legend('u');
title('Linear Tangent Steering control');
```



Linear Tangent Steering state variables
60
40
20
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
x1
x2
x3
x4
Linear Tangent Steering control
1
0.5
0
-0.5
-1
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
u

58 Linear Gas Absorber

ITERATIVE DYNAMIC PROGRAMMING, REIN LUUS

Section 7.4.4 Gas absorber with a large number of plates

CHAPMAN & HALL/CRC Monographs and Surveys in Pure and Applied Mathematics

58.1 Problem description

A general case of an n-plate gas absorber controlled by inlet feed stream concentrations

Find u over t in [0; 10] to minimize

$$J = \int_0^{10} x' * x + u' * u \mathrm{d}t$$

subject to:

$$\frac{dx}{dt} = A * x + B * u$$

A = tridiag(0.538998 -1.173113 0.634115)

B = [0.538998 0 ... 0 0; 0 0 ... 0 0.634115]

The initial condition is chosen as:

xi(0) = -0.0307-(i-1)/(n-1)*(0.1273-0.0307), i= 1, 2, ... , n. where n is the number of stages

$$0 <= u1 <= inf$$

$$0 <= u2 <= inf$$

Reference: [25]

58.2 Problem setup

```
toms t
n   = 10;
t_f  = 10;

p = tomPhase('p', t, 0, t_f, 20);
setPhase(p);

x = tomState(p, 'x', n, 1);
u = tomControl(p, 'u', 2, 1);

i = (1:n)';
x0i = -0.0307-(0.1273-0.0307)/(n-1)*(i-1);

% Initial guess
% Note: The guess for t_f must appear in the list before expression involving t.
guess = icollocate(x == x0i);

% Box constraints
cbox = {0 <= collocate(u)};

% Initial conditions
cinit = initial(x == x0i);

% Various constants and expressions
A = spdiags([0.538998*ones(n,1) ...
    -1.173113*ones(n,1) 0.634115*ones(n,1)],-1:1,n,n);
B = sparse(n,2);
B(1,1)   = 0.538998;
B(end,2) = 0.634115;

% ODEs and path constraints
ceq = collocate(dot(x) == A*x+B*u);

% Objective
objective = integrate(x'*x+u'*u);
```

58.3 Solve the problem

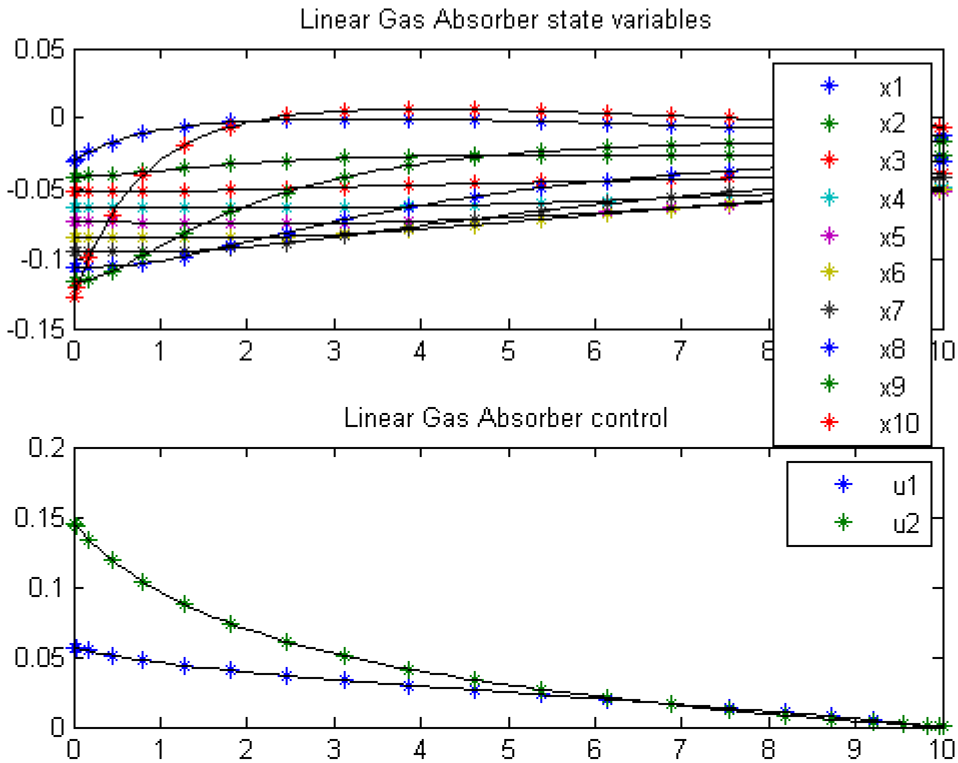
```
options = struct;
options.name = 'Linear Gas Absorber';
solution = ezsolve(objective, {cbox, ceq, cinit}, guess, options);

% Extract optimal states and controls from solution
```

```
Problem type appears to be: qp
```

```
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem:  1: Linear Gas Absorber                f_k       0.328245958214705650
                                      sum(|constr|)      0.000000000000006954
                             f(x_k) + sum(|constr|)      0.328245958214712590
                                             f(x_0)      0.000000000000000000

Solver: CPLEX.  EXIT=0.  INFORM=1.
CPLEX Barrier QP solver
Optimal solution found

FuncEv   15 GradEv   15 ConstrEv   15 Iter   15
CPU time: 0.109375 sec. Elapsed time: 0.062000 sec.
```

58.4 Plot result

```
subplot(2,1,1)
ezplot(x);
legend('x1','x2','x3','x4','x5','x6','x7','x8','x9','x10')
title('Linear Gas Absorber state variables');

subplot(2,1,2)
ezplot(u);
legend('u1', 'u2');
title('Linear Gas Absorber control');
```


Linear Gas Absorber state variables
0.05
0
-0.05
-0.1
-0.15
0
1
2
3
4
5
6
7
8
10
x1
x2
x3
x4
x5
x6
x7
x8
x9
x10
Linear Gas Absorber control
0.2
0.15
0.1
0.05
0
0
1
2
3
4
5
6
7
8
9
10
u1
u2

59 Linear Pendulum

Viscocity Solutions of Hamilton-Jacobi Equations and Optimal Control Problems. Alberto Bressan, S.I.S.S.A, Trieste, Italy.

A linear pendulum problem controlled by an external force.

59.1 Problem Description

Find u over t in [0; 20] to maximize:

$$J = x_1(t_f)$$

subject to:

$$\frac{dx_1}{dt} = x_2$$
$$\frac{dx_2}{dt} = u - x_1$$
$$x(t_0) = [0\ 0]$$
$$|u| <= 1$$

Reference: [8]

59.2 Problem setup

```
toms t
t_f = 20;
p = tomPhase('p', t, 0, t_f, 60);
setPhase(p);

tomStates x1 x2
tomControls u

% Initial guess
x0 = {icollocate({x1 == 0; x2 == 0})
    collocate(u == 0)};
```

```
% Box constraints and bounds
cb = {-1 <= collocate(u) <= 1
    initial(x1 == 0)
    initial(x2 == 0)};

% ODEs and path constraints
ceq = collocate({dot(x1) == x2
    dot(x2) == u-x1});

% Objective
objective = -final(x1);
```

59.3 Solve the problem

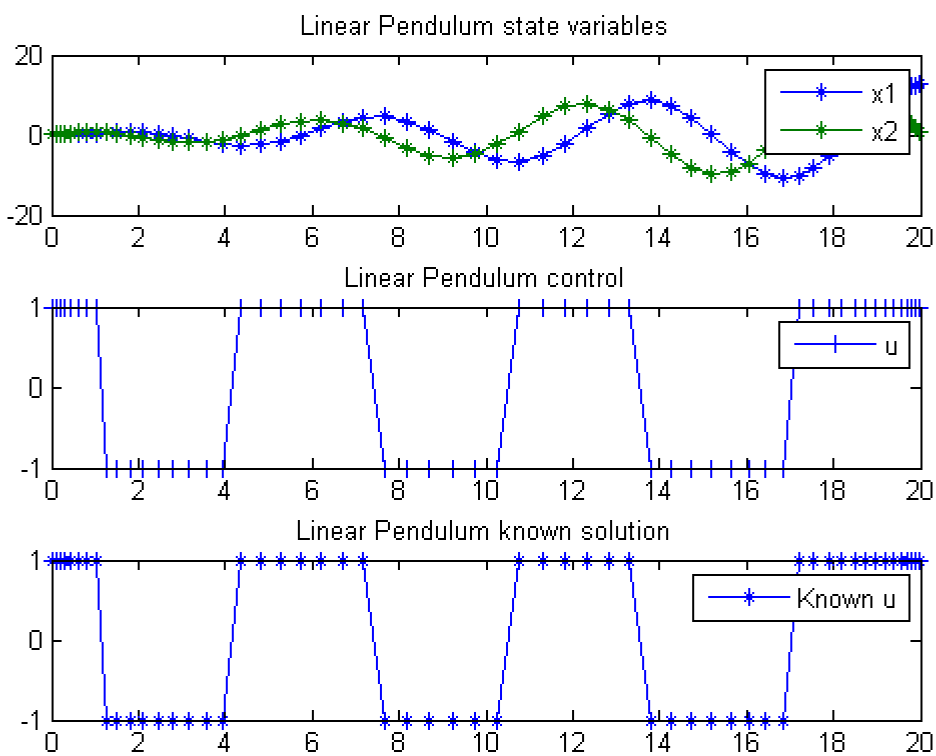
```
options = struct;
options.name = 'Linear Pendulum';
solution = ezsolve(objective, {cb, ceq}, x0, options);
t  = subs(collocate(t),solution);
x1 = subs(collocate(x1),solution);
x2 = subs(collocate(x2),solution);
u  = subs(collocate(u),solution);
```

```
Problem type appears to be: lp
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Linear Pendulum                f_k     -12.612222977985938000
                                       sum(|constr|)      0.000000000003687556
                              f(x_k) + sum(|constr|)    -12.612222977982251000
                                              f(x_0)      0.000000000000000000

Solver: CPLEX.  EXIT=0.  INFORM=1.
CPLEX Dual Simplex LP solver
Optimal solution found

FuncEv  206 Iter  206
CPU time: 0.031250 sec. Elapsed time: 0.031000 sec.
```

59.4 Plot result

```
subplot(3,1,1)
plot(t,x1,'*-',t,x2,'*-');
legend('x1','x2');
title('Linear Pendulum state variables');
```

```
subplot(3,1,2)
plot(t,u,'+-');
legend('u');
title('Linear Pendulum control');

subplot(3,1,3)
plot(t,sign(sin(t_f-t)),'*-');
legend('Known u');
title('Linear Pendulum known solution');
```



60 Linear Problem with Bang Bang Control

Problem 5a: Miser3 manual

60.1 Problem description

Find u over t in [0; 1] to minimize

$$J = \int_0^1 -6 * x_1 - 12 * x_2 + 3 * u_1 + u_2 \mathrm{d}t$$

subject to:

$$\frac{dx_1}{dt} = u_2$$
$$\frac{dx_2}{dt} = -x_1 + u_1$$
$$x_1(0) = 1$$
$$x_2(0) = 0$$
$$|u| <= 10$$

Reference: [19]

60.2 Problem setup

```
toms t
p = tomPhase('p', t, 0, 1, 30);
setPhase(p);

tomStates x1 x2
tomControls u1 u2

% Initial guess
x0 = {icollocate({x1 == 1; x2 == 0})
    collocate({u1 == 0; u2 == 0})};

% Box constraints
cbox = {-10 <= icollocate(x1) <= 10
    -10 <= icollocate(x2) <= 10
```

```
    -10 <= collocate(u1)  <= 10
    -10 <= collocate(u2)  <= 10};

% Boundary constraints
cbnd = initial({x1 == 1; x2 == 0});

% ODEs and path constraints
ceq = collocate({dot(x1) == u2
    dot(x2) == -x1+u1});

% Objective
objective = integrate(-6*x1-12*x2+3*u1+u2);
```

60.3 Solve the problem

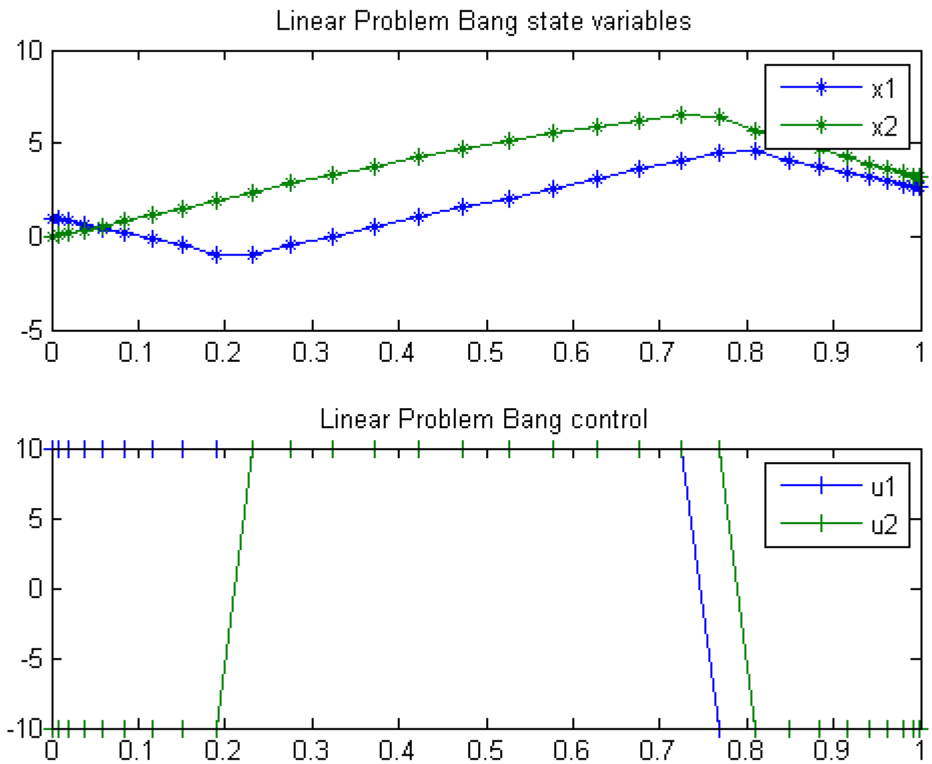
```
options = struct;
options.name = 'Linear Problem Bang';
solution = ezsolve(objective, {cbox, cbnd, ceq}, x0, options);
t  = subs(collocate(t),solution);
x1 = subs(collocate(x1),solution);
x2 = subs(collocate(x2),solution);
u1 = subs(collocate(u1),solution);
u2 = subs(collocate(u2),solution);
```

```
Problem type appears to be: lp
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Linear Problem Bang            f_k     -41.377652213983296000
                                        sum(|constr|)      0.000000000005182483
                              f(x_k) + sum(|constr|)    -41.377652213978116000
                                               f(x_0)      0.000000000000000000

Solver: CPLEX.  EXIT=0.  INFORM=1.
CPLEX Dual Simplex LP solver
Optimal solution found

FuncEv   63 Iter   63
CPU time: 0.015625 sec. Elapsed time: 0.016000 sec.
```

60.4 Plot result

```
subplot(2,1,1)
plot(t,x1,'*-',t,x2,'*-');
```

```
legend('x1','x2');
title('Linear Problem Bang state variables');

subplot(2,1,2)
plot(t,u1,'+-',t,u2,'+-');
legend('u1','u2');
title('Linear Problem Bang control');
```



61 LQR Problem

Problem: LQR: RIOTS 95 Manual

61.1 Problem Description

Find u(t) over t in [0; 1] to minimize

$$J = \int_0^1 (0.625 * x^2 + 0.5 * x * u + 0.5 * u^2)\mathrm{d}t$$

subject to:

$$\frac{dx}{dt} = \frac{1}{2} * x + u$$
$$x(0) = 1$$

Reference: [29]

61.2 Problem setup

```
toms t
p = tomPhase('p', t, 0, 1, 20);
setPhase(p);

tomStates x
tomControls u

% Initial guess
x0 = icollocate(x == 1-t);

% ODEs and constraints
ceq = {collocate(dot(x) == 0.5*x+u)
    initial(x == 1)};

% Objective
objective = integrate(0.625*x.^2+0.5*x.*u+0.5*u.^2);
```

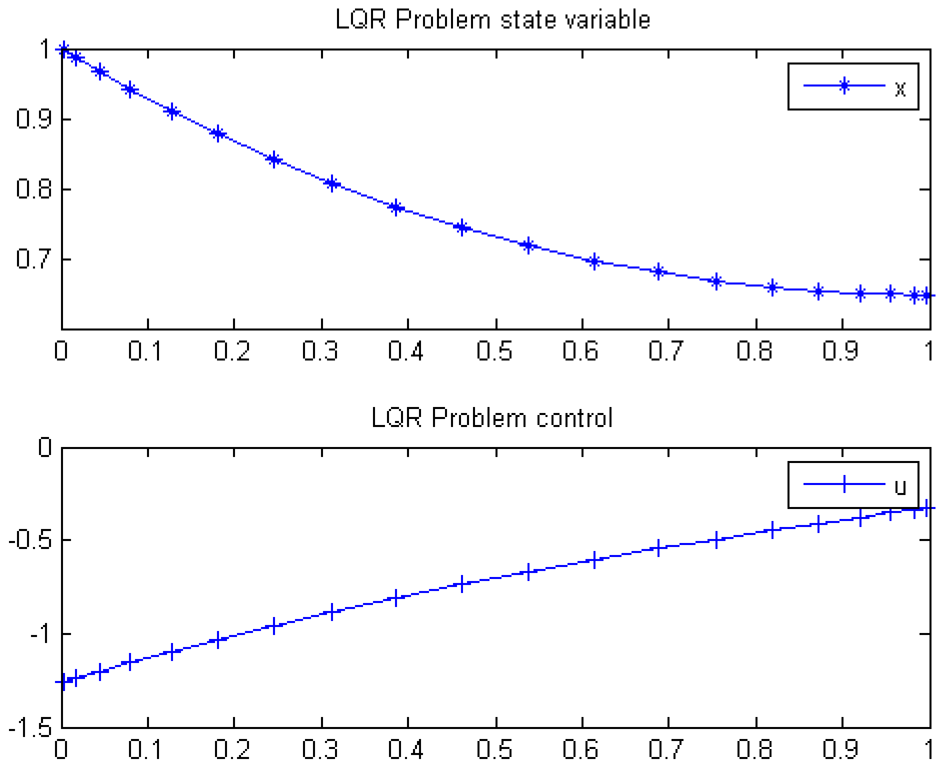
61.3 Solve the problem

```
options = struct;
options.name = 'LQR Problem';
solution = ezsolve(objective, ceq, x0, options);
t = subs(collocate(t),solution);
x = subs(collocate(x),solution);
u = subs(collocate(u),solution);
```

```
Problem type appears to be: qp
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem:  1: LQR Problem                        f_k       0.380797077977577230
                                       sum(|constr|)      0.000000000046131558
                               f(x_k) + sum(|constr|)      0.380797078023708770
                                              f(x_0)      0.000000000000000000

Solver: CPLEX.  EXIT=0.  INFORM=1.
CPLEX Barrier QP solver
Optimal solution found

FuncEv    3 GradEv    3 ConstrEv    3 Iter    3
CPU time: 0.015625 sec. Elapsed time: 0.016000 sec.
```

61.4 Plot result

```
subplot(2,1,1)
plot(t,x,'*-');
legend('x');
title('LQR Problem state variable');

subplot(2,1,2)
plot(t,u,'+-');
legend('u');
title('LQR Problem control');
```


LQR Problem state variable
x
1
0.9
0.8
0.7
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
LQR Problem control
u
0
-0.5
-1
-1.5
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

62 Marine Population Dynamics

Benchmarking Optimization Software with COPS Elizabeth D. Dolan and Jorge J. More ARGONNE NATIONAL LABORATORY

62.1 Problem Formulation

Find m and g over t in [0; 10] to minimize

$$J = \sum_{i=1}^{21} \sum_{j=1}^{8} (y_{j,i} - ymeas_{j,i})^2$$

subject to:

$$\begin{aligned} \frac{dy_1}{dt} &= -(m_1 + g_1) * y_1 \\ \frac{dy_i}{dt} &= g_{i-1} * y_{i-1} - (m_i + g_i) * y_i \\ \frac{dy_8}{dt} &= g_7 * y_7 - (m_8) * y_8 \end{aligned}$$

Where the data is given in the code.

Reference: [14]

62.2 Problem setup

```
t = tom('t');
m = tom('m',8,1);
g = tom('g',7,1);

% Various constants and expressions
ymeas = [...
    20000 17000 10000 15000 12000 9000 7000 3000
    12445 15411 13040 13338 13484 8426 6615 4022
     7705 13074 14623 11976 12453 9272 6891 5020
     4664  8579 12434 12603 11738 9710 6821 5722
     2977  7053 11219 11340 13665 8534 6242 5695
     1769  5054 10065 11232 12112 9600 6647 7034
      943  3907  9473 10334 11115 8826 6842 7348
```

```
      581  2624  7421 10297 12427 8747 7199 7684
      355  1744  5369  7748 10057 8698 6542 7410
      223  1272  4713  6869  9564 8766 6810 6961
      137   821  3451  6050  8671 8291 6827 7525
       87   577  2649  5454  8430 7411 6423 8388
       49   337  2058  4115  7435 7627 6268 7189
       32   228  1440  3790  6474 6658 5859 7467
       17   168  1178  3087  6524 5880 5562 7144
       11    99   919  2596  5360 5762 4480 7256
        7    65   647  1873  4556 5058 4944 7538
        4    44   509  1571  4009 4527 4233 6649
        2    27   345  1227  3677 4229 3805 6378
        1    20   231   934  3197 3695 3159 6454
        1    12   198   707  2562 3163 3232 5566];

tmeas = 0:0.5:10;

% Box constraints
cbox = {
    0 <= m
    0 <= g
    };

p = tomPhase('p', t, tmeas(1), tmeas(end), 2*length(tmeas), [], 'gauss');
setPhase(p);
y = tomState('y',8,1);

% Initial guess - linear interpolation between the data points
x0 = {m==0; g==0;
    icollocate(y == interp1(tmeas,ymeas,t)')};

yerr = sum(sum((atPoints(tmeas,y) - ymeas).^2));

% ODE
ceq = collocate( dot(y) == [0; g].*[0; y(1:7)] - (m+[g;0]).*y );
```

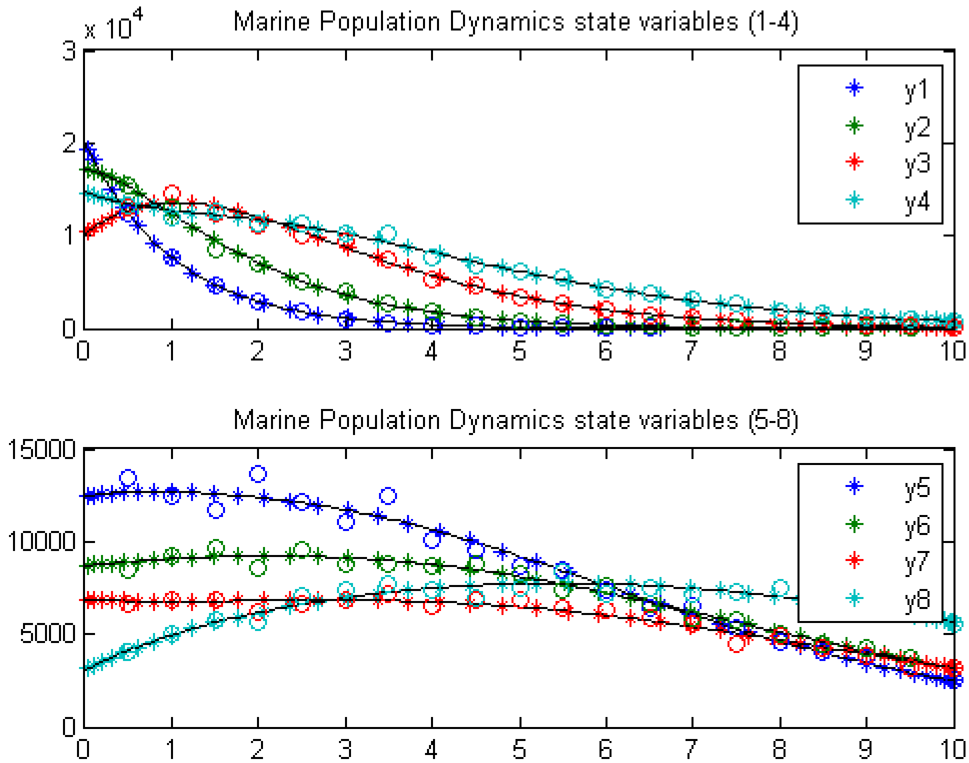
62.3 Solve the problem

```
options = struct;
options.name = 'Marine Population Dynamics';
solution = ezsolve(1e-5*yerr, {cbox, ceq}, x0, options);

% Optimal y, m and g - use as starting guess
yopt = subs(y, solution);
mopt = subs(m, solution);
gopt = subs(g, solution);
```

```
Problem type appears to be: qpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Marine Population Dynamics     f_k     197.465297161281340000
                                       sum(|constr|)      0.000000124069941210
                              f(x_k) + sum(|constr|)    197.465297285351280000
                                              f(x_0) -86874.198350960098000000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv   15 ConJacEv   15 Iter   14 MinorIter  432
CPU time: 0.718750 sec. Elapsed time: 0.407000 sec.
```

62.4 Plot result

```
subplot(2,1,1)
ezplot(y(1:4));
hold on
plot(tmeas,ymeas(:,1:4),'o');
hold off
legend('y1','y2','y3','y4');
title('Marine Population Dynamics state variables (1-4)');

subplot(2,1,2)
ezplot(y(5:8));
legend('y5','y6','y7','y8');
hold on
plot(tmeas,ymeas(:,5:8),'o');
hold off
title('Marine Population Dynamics state variables (5-8)');
```


Marine Population Dynamics state variables (1-4)
x 10^4
y1
y2
y3
y4
Marine Population Dynamics state variables (5-8)
y5
y6
y7
y8

63 Max Radius Orbit Transfer

63.1 Problem description

Maximum radius orbit transfer of a spacecraft.

Applied Optimal Control, Bryson & Ho, 1975. Example on pages 66-69.

Programmers: Gerard Van Willigenburg (Wageningen University) Willem De Koning (retired from Delft University of Technology)

63.2 Problem setup

```
% Array with consecutive number of collocation points
narr = [20 40];

toms t;
t_f = 1; % Fixed final time

for n=narr

    p = tomPhase('p', t, 0, t_f, n);
    setPhase(p)

    tomStates x1 x2 x3
    tomControls u1

    % Parameters
    r0 = 1; mmu = 11; th = 1.55;
    m0 = 1; rm0 = -0.25;

    % Initial state
    xi=[r0; 0; sqrt(mmu/r0)];

    % Initial guess
    if n==narr(1)
        x0 = {icollocate({x1 == xi(1); x2 == xi(2); x3 == xi(3)})
            collocate({u1 == 0})};
    else
        x0 = {icollocate({x1 == xopt1; x2 == xopt2; x3 == xopt3})
            collocate({u1 == uopt1})};
    end
```

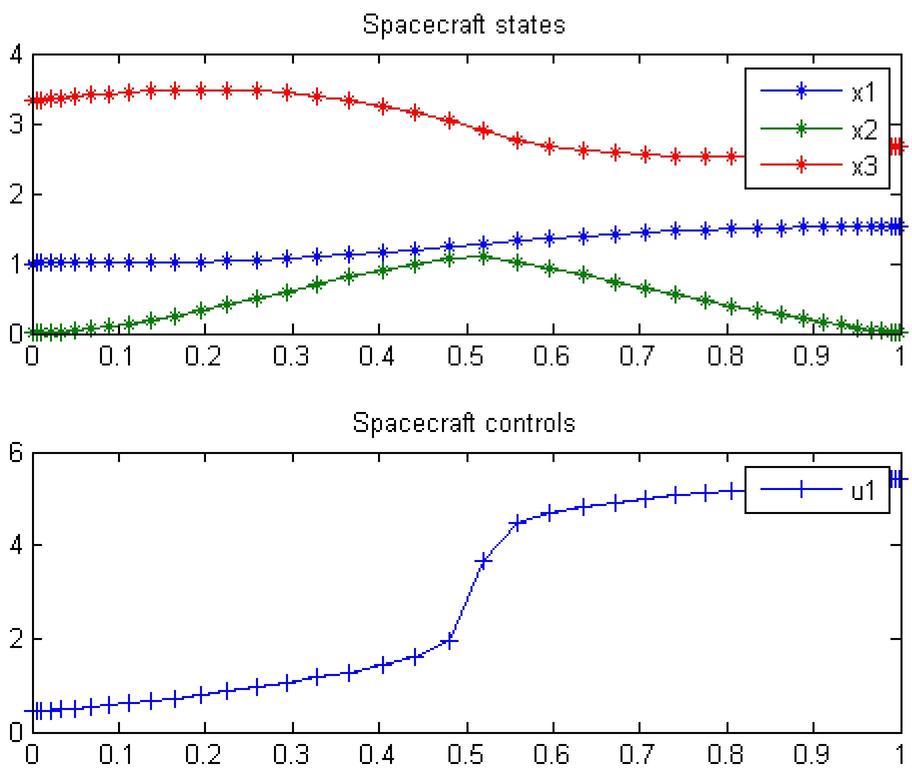
```
% Boundary constraints
cbnd = {initial({x1 == xi(1); x2 == xi(2); x3 == xi(3)})
    final({x3 == sqrt(mmu/x1); x2 == 0})};

% ODEs and path constraints
dx1 = x2;
dx2 = x3.*x3./x1-mmu./(x1.*x1)+th*sin(u1)./(m0+rm0*t);
dx3 = -x2.*x3./x1+th*cos(u1)./(m0+rm0*t);

ceq = collocate({
    dot(x1) == dx1
    dot(x2) == dx2
    dot(x3) == dx3});

% Objective
objective = -final(x1);
```

63.3 Solve the problem

```
options = struct;
options.name = 'Spacecraft';
solution = ezsolve(objective, {cbnd, ceq}, x0, options);

xopt1 = subs(x1,solution);
xopt2 = subs(x2,solution);
xopt3 = subs(x3,solution);
uopt1 = subs(u1,solution);
```

```
Problem type appears to be: lpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Spacecraft                    f_k      -1.526286382793847300
                                      sum(|constr|)      0.000000153662086173
                             f(x_k) + sum(|constr|)     -1.526286229131761200
                                             f(x_0)     -0.999999999999998220

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv   75 ConJacEv   75 Iter   43 MinorIter   96
CPU time: 0.234375 sec. Elapsed time: 0.235000 sec.


Problem type appears to be: lpcon
```

```
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Spacecraft                    f_k      -1.526020732841172800
                                     sum(|constr|)      0.000002928422557971
                            f(x_k) + sum(|constr|)     -1.526017804418614800
                                            f(x_0)     -1.526286382793838200

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv   15 ConJacEv   15 Iter   13 MinorIter  157
CPU time: 0.171875 sec. Elapsed time: 0.172000 sec.


end

% Get final solution
t  = subs(collocate(t),solution);
x1 = subs(collocate(x1),solution);
x2 = subs(collocate(x2),solution);
x3 = subs(collocate(x3),solution);
u1 = subs(collocate(u1),solution);

%Bound u1 to [0,2pi]
u1 = rem(u1,2*pi); u1 = (u1<0)*2*pi+u1;

% Plot final solution
subplot(2,1,1)
plot(t,x1,'*-',t,x2,'*-',t,x3,'*-');
legend('x1','x2','x3');
title('Spacecraft states');

subplot(2,1,2)
plot(t,u1,'+-');
legend('u1');
title('Spacecraft controls');
```


Spacecraft states
x1
x2
x3
Spacecraft controls
u1

64 Sequential Activation of Metabolic Pathways

a Dynamic Optimization Approach 2009, Diego A. Oyarzuna, Brian P. Ingalls, Richard H. Middleton, Dimitrios Kalamatianosa

64.1 Problem description

The problem is described in the paper referenced above.

64.2 Problem setup

```
N = [30 128]; % Number of collocation points
toms t t_f
warning('off', 'tomSym:x0OutOfBounds');

for n = N

    p = tomPhase('p', t, 0, t_f, n);
    setPhase(p);
    tomStates s1 s2 s3 e0 e1 e2 e3
    tomControls u0 u1 u2 u3

    % Initial guess
    if n == N(1)
        x0 = {t_f == 2
            icollocate({
            [s1;s2;s3] == 0
            e0 == t/t_f
            e1 == (t-0.2)/t_f
            e2 == (t-1)/t_f
            e3 == (t-1.2)/t_f
            })
            collocate({u0 == 1
            [u1;u2;u3] == 0})};
    else
        x0 = {t_f == tf_init
            icollocate({
            s1 == s1_init; s2 == s2_init; s3 == s3_init
            e0 == e0_init; e1 == e1_init; e2 == e2_init
            e3 == e3_init})
            collocate({u0 == u0_init
            u1 == u1_init; u2 == u2_init; u3 == u3_init})};
    end
```

```
% Box constraints
cbox = {0.1 <= t_f <= tfmax
    0  <= icollocate([s1;s2;s3]) <= 100
    0  <= icollocate([e0;e1;e2;e3]) <= 1
    0  <= collocate([u0;u1;u2;u3])  <= Umax};

% Boundary constraints
cbnd = {initial({[s1;s2;s3] == 0; [e0;e1;e2;e3] == 0})
    final({s1 == s1f; s2 == s2f; s3 == s3f;
    e0 == e0f; e1 == e1f; e2 == e2f; e3 == e3f})};

% Michaelis-Mentes ODEs and path constraints
ceq = collocate({
    dot(s1) == kcat0*s0*e0/(Km + s0) - kcat1*s1*e1/(Km+s1)
    dot(s2) == kcat1*s1*e1/(Km+s1) - kcat2*s2*e2/(Km+s2)
    dot(s3) == kcat2*s2*e2/(Km+s2) - kcat3*s3*e3/(Km+s3)
    dot(e0) == u0 - lam*e0
    dot(e1) == u1 - lam*e1
    dot(e2) == u2 - lam*e2
    dot(e3) == u3 - lam*e3});

% Objective
objective = integrate(1 + e0 + e1 + e2 + e3);
```

64.3 Solve the problem

```
options = struct;
options.name = 'Metabolic Pathways, Unbranched n=4';
solution = ezsolve(objective, {cbox, cbnd, ceq}, x0, options);

% Collect solution as initial guess
s1_init = subs(s1,solution);
s2_init = subs(s2,solution);
s3_init = subs(s3,solution);
e0_init = subs(e0,solution);
e1_init = subs(e1,solution);
e2_init = subs(e2,solution);
e3_init = subs(e3,solution);
tf_init = subs(t_f,solution);
u0_init = subs(u0,solution);
u1_init = subs(u1,solution);
u2_init = subs(u2,solution);
u3_init = subs(u3,solution);
```

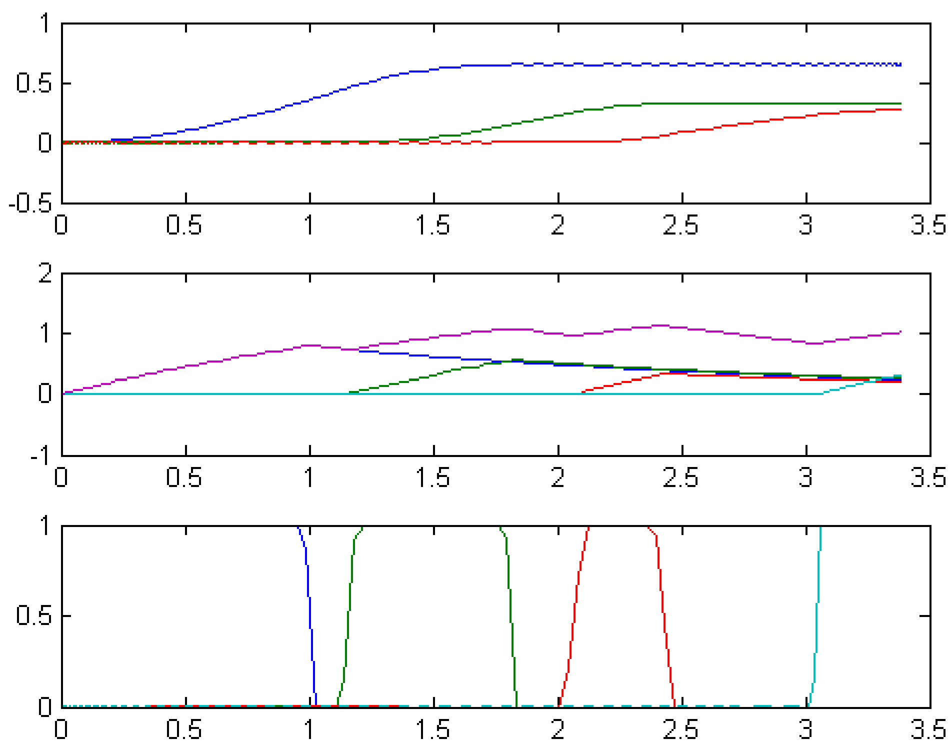
```
Problem type appears to be: qpcon
Starting numeric solver
```

```
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Metabolic Pathways, Unbranched n=4  f_k       6.092698010914444900
                                            sum(|constr|)      0.000001117096701837
                                   f(x_k) + sum(|constr|)      6.092699128011147100
                                                   f(x_0)      4.220507512582271300

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv   34 ConJacEv   34 Iter   22 MinorIter 1806
CPU time: 0.734375 sec. Elapsed time: 0.750000 sec.

Problem type appears to be: qpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Metabolic Pathways, Unbranched n=4  f_k       6.085709091573209100
                                            sum(|constr|)      0.000005873260299532
                                   f(x_k) + sum(|constr|)      6.085714964833508500
                                                   f(x_0)      6.092720075212315400

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv    7 ConJacEv    7 Iter    5 MinorIter 2096
CPU time: 9.343750 sec. Elapsed time: 9.672000 sec.
```

```
end

% Collect data
t  = subs(collocate(t),solution);
s1 = subs(collocate(s1),solution);
s2 = subs(collocate(s2),solution);
s3 = subs(collocate(s3),solution);
e0 = subs(collocate(e0),solution);
e1 = subs(collocate(e1),solution);
e2 = subs(collocate(e2),solution);
e3 = subs(collocate(e3),solution);
u0  = subs(collocate(u0),solution);
u1  = subs(collocate(u1),solution);
u2  = subs(collocate(u2),solution);
u3  = subs(collocate(u3),solution);
```

64.4 Plot result

```
s = [s1 s2 s3];
e = [e0 e1 e2 e3];
r = [u0 u1 u2 u3];
subplot(3,1,1);
plot(t,[s1 s2 s3]);
subplot(3,1,2);
plot(t,[e0 e1 e2 e3 e0+e1+e2+e3]);
subplot(3,1,3);
plot(t,[u0 u1 u2 u3]);
```



65 Methanol to Hydrocarbons

Benchmarking Optimization Software with COPS Elizabeth D. Dolan and Jorge J. More ARGONNE NATIONAL LABORATORY

65.1 Problem Formulation

Find theta over t in [0; 1.122] to minimize

$$J = \sum_{j=1}^{3}\sum_{i=1}^{21}(y_{j,i} - y_{j,i,meas})^2$$

subject to:

$$\frac{dy_1}{dt} = -(2 * theta_2 - \frac{theta_1 * y_2}{(theta_2 + theta_5) * y_1 + y_2} + theta_3 + theta_4) * y_1$$
$$\frac{dy_2}{dt} = \frac{theta_1 * y_1 * (theta_2 * y_1 - y_2)}{(theta_2 + theta_5) * y_1 + y_2} + theta_3 * y_1$$
$$\frac{dy_3}{dt} = \frac{theta_1 * y_1 * (y_2 + theta_5 * y_1)}{(theta_2 + theta_5) * y_1 + y_2} + theta_4 * y_1$$
$$theta >= 0$$

Where the data is given in the code.

Reference: [14]

65.2 Problem setup

```
toms t theta1 theta2 theta3 theta4 theta5

% Various constants and expressions
y1meas = [0.7085;0.5971;0.5537;0.3684;0.1712;...
    0.1198;0.0747;0.0529;0.0415;0.0261;0.0208;...
    0.0085;0.0053;0.0019;0.0018];
y2meas = [0.1621;0.1855;0.1989;0.2845;0.3491;...
    0.3098;0.3576;0.3347;0.3388;0.3557;0.3483;...
    0.3836;0.3611;0.3609;0.3485];
y3meas = [0.0811;0.0965;0.1198;0.1535;0.2097;...
```

```
    0.2628;0.2467;0.2884;0.2757;0.3167;0.2954;...
    0.295;0.2937;0.2831;0.2846];
tmeas = [0.05;0.065;0.08;0.123;0.233;0.273;...
    0.354;0.397;0.418;0.502;0.553;...
    0.681;0.75;0.916;0.937];
```

65.3 Solve the problem, using a successively larger number collocation points

```
for n=[20 80]

    p = tomPhase('p', t, 0, 1.122, n);
    setPhase(p);

    tomStates y1 y2 y3

    % Initial guess
    if n == 20
        x0 = {theta1 == 1; theta2 == 1
            theta3 == 1; theta4 == 1
            theta5 == 1
            icollocate({
            y1 == 1-(1-0.0006)*t/1.122
            y2 == 0.3698*t/1.122
            y3 == 0.2899*t/1.122})};
    else
        x0 = {theta1 == theta1opt; theta2 == theta2opt
            theta3 == theta3opt; theta4 == theta4opt
            theta5 == theta5opt
            icollocate({
            y1 == y1opt
            y2 == y2opt
            y3 == y3opt})};
    end

    % Box constraints
    cbox = {sqrt(eps) <= theta1; sqrt(eps) <= theta2
        sqrt(eps) <= theta3; sqrt(eps) <= theta4
        sqrt(eps) <= theta5};

    y1err = sum((atPoints(tmeas,y1) - y1meas).^2);
    y2err = sum((atPoints(tmeas,y2) - y2meas).^2);
    y3err = sum((atPoints(tmeas,y3) - y3meas).^2);

    % Start and end points cannot be interpolated
    y1end = (1-initial(y1)).^2 + (0.0006-final(y1))^2;
    y2end = (0-initial(y2)).^2 + (0.3698-final(y2))^2;
    y3end = (0-initial(y3)).^2 + (0.2899-final(y3))^2;
```

```
% ODEs and path constraints
ceq = collocate({
    dot(y1) == -(2*theta2-(theta1*y2)./((theta2+theta5)*y1+y2)+theta3+theta4).*y1
    dot(y2) == (theta1*y1.*(theta2*y1-y2))./((theta2+theta5)*y1+y2)+theta3*y1
    dot(y3) == (theta1*y1.*(y2+theta5*y1))./((theta2+theta5)*y1+y2)+theta4*y1});

% Objective
objective = y1err+y2err+y3err+y1end+y2end+y3end;
```

65.4 Solve the problem

```
options = struct;
options.name = 'Methanol to Hydrocarbons';
solution = ezsolve(objective, {cbox, ceq}, x0, options);

% Optimal x, theta for starting point
y1opt = subs(y1, solution);
y2opt = subs(y2, solution);
y3opt = subs(y3, solution);
theta1opt = subs(theta1, solution);
theta2opt = subs(theta2, solution);
theta3opt = subs(theta3, solution);
theta4opt = subs(theta4, solution);
theta5opt = subs(theta5, solution);
```

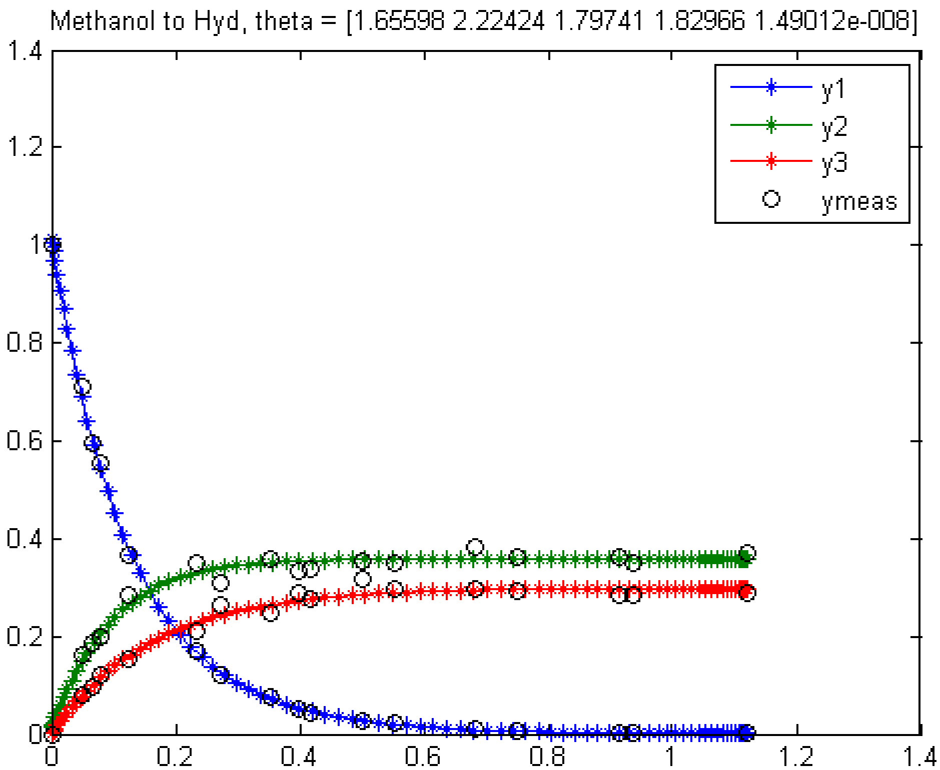
```
Problem type appears to be: qpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Methanol to Hydrocarbons       f_k       0.008301664004164877
                                       sum(|constr|)      0.000000001050742318
                              f(x_k) + sum(|constr|)      0.008301665054907195
                                              f(x_0)     -0.959232294294469990

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv   42 ConJacEv   42 Iter   41 MinorIter   88
CPU time: 0.140625 sec. Elapsed time: 0.157000 sec.


Problem type appears to be: qpcon
Starting numeric solver
```

```
===== * * * ================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Methanol to Hydrocarbons       f_k       0.008301664004168430
                                     sum(|constr|)      0.000000988210502411
                            f(x_k) + sum(|constr|)      0.008302652214670841
                                            f(x_0)     -5.007954925995837100

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv    1 ConJacEv    1 MinorIter  159
CPU time: 0.125000 sec. Elapsed time: 0.125000 sec.
```

```
end

t  = subs(collocate(t),solution);
y1 = collocate(y1opt);
y2 = collocate(y2opt);
y3 = collocate(y3opt);
t1 = subs(theta1,solution);
t2 = subs(theta2,solution);
t3 = subs(theta3,solution);
t4 = subs(theta4,solution);
t5 = subs(theta5,solution);
```

65.5 Plot result

```
figure(1);
tm  = [0;tmeas;1.122];
y1m = [1;y1meas;0.0006];
y2m = [0;y2meas;0.3698];
y3m = [0;y3meas;0.2899];
plot(t,y1,'*-',t,y2,'*-',t,y3,'*-',tm,y1m,'ko',tm,y2m,'ko',tm,y3m,'ko');
legend('y1','y2','y3','ymeas');
title(sprintf('Methanol to Hyd, theta = [%g %g %g %g %g]',t1,t2,t3,t4,t5));
```


Methanol to Hyd, theta = [1.65598 2.22424 1.79741 1.82966 1.49012e-008]
y1
y2
y3
ymeas
1.4
1.2
1
0.8
0.6
0.4
0.2
0
0
0.2
0.4
0.6
0.8
1
1.2
1.4

66 Min Energy Orbit Transfer

66.1 Problem description

Minimum energy orbit transfer of a spacecraft with limited variable thrust.

From the paper: Low thrust, high-accuracy trajectory optimization, I. Michael Ross, Qi Gong and Pooya Sekhavat.

Programmers: Li Jian (Beihang University)

66.2 Problem setup

```
% Array with consecutive number of collocation points
narr = [40 80];

toms t;
toms t_f;
t0 = 0;
tfmax = 57;

for n=narr

    %p = tomPhase('p', t, 0, t_f-t0, n, [], 'cheb');
    p = tomPhase('p', t, 0, t_f-t0, n);
    setPhase(p)

    tomStates r theta vr vt
    tomControls ur ut

    % Parameters
    r0 = 1; theta0 = 0; vr0 = 0; vt0 = 1;
    r_f = 4;              vrf = 0; vtf = 0.5;
    umax = 0.01;

    % Initial state
    xi=[r0; theta0; vr0; vt0];

    % Initial guess
    if n==narr(1)
        x0 = {t_f == 57;
            icollocate({r == xi(1); theta == xi(2); vr == xi(3); vt == xi(4)})
            collocate({ur == 0; ut == umax})};
```

```
else
    x0 = {t_f == tfopt;
        icollocate({r == xopt1; theta == xopt2; vr == xopt3; vt == xopt4});
        collocate({ur == uopt1; ut == uopt2})};
end

% Box constraints
cbox = {10 <= t_f<= tfmax;
    1 <= collocate(r) <= 4;
    0 <= collocate(vr) <= 0.5;
    0 <= collocate(vt) <= 1;
    -umax <= collocate(ur) <= umax;
    -umax <= collocate(ut) <= umax};

% Boundary constraints
cbnd = {initial({r == r0; theta == theta0; vr == vr0; vt == vt0})
    final({r == r_f; vr == 0; vt == vtf})};

% ODEs and path constraints
d_r      = vr;
dtheta  = vt./r;
dvr     = vt.*vt./r - 1.0./r./r + ur;
dvt     = -vr.*vt./r + ut;

ceq = collocate({
    dot(r) == d_r;
    dot(theta) == dtheta;
    dot(vr) == dvr;
    dot(vt) == dvt;
    0<=(ur.*ur+ut.*ut).^0.5<=umax});

% Objective
objective = integrate((ur.^2+ut.^2).^0.5);
```

66.3 Solve the problem

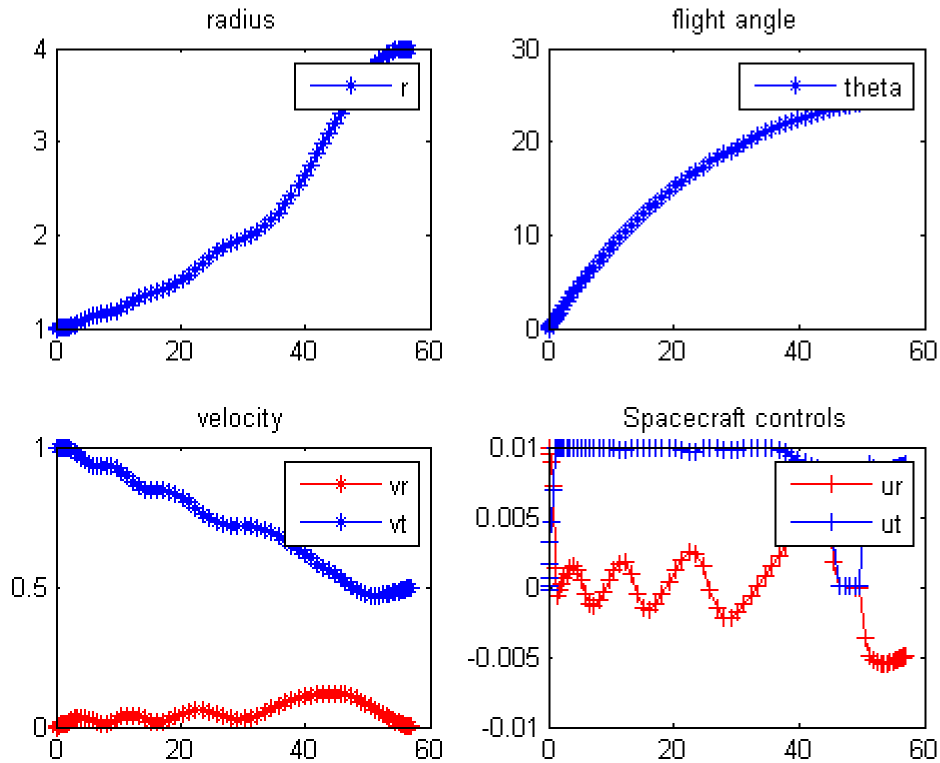
```
options = struct;
options.type = 'con';
options.name = 'Min Energy Transfer';
Prob = sym2prob(objective, {cbox, cbnd, ceq}, x0, options);
Prob.KNITRO.options.ALG = 1;
Prob.KNITRO.options.HONORBNDS = 0;
Result = tomRun('knitro', Prob, 1);
solution = getSolution(Result);

xopt1 = subs(r,solution);
xopt2 = subs(theta,solution);
```

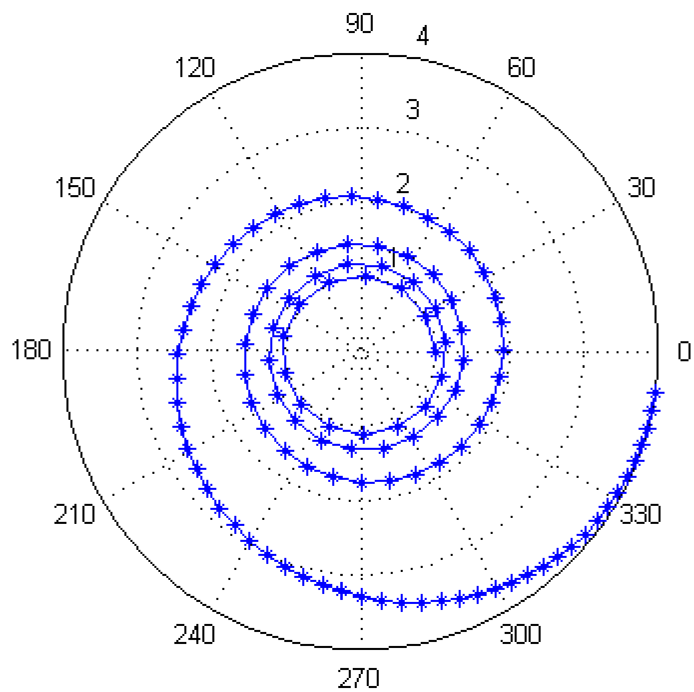
```
    xopt3 = subs(vr,solution);
    xopt4 = subs(vt,solution);
    uopt1 = subs(ur,solution);
    uopt2 = subs(ut,solution);
    tfopt = subs(t_f,solution);
```

```
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Min Energy Transfer            f_k       0.523205792272900430
                                       sum(|constr|)      0.000002310623001201
                              f(x_k) + sum(|constr|)      0.523208102895901580
                                              f(x_0)      0.569999999999999170

Solver: KNITRO.  EXIT=0.  INFORM=0.
Interior/Direct NLP KNITRO
Locally optimal solution found

FuncEv  235 GradEv 1975 ConstrEv  234 ConJacEv 1975 Iter  206 MinorIter  231
CPU time: 9.203125 sec. Elapsed time: 9.203000 sec.


===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Min Energy Transfer            f_k       0.523085806417755370
                                       sum(|constr|)      0.000000004935871044
                              f(x_k) + sum(|constr|)      0.523085811353626420
                                              f(x_0)      0.522037502371971220

Solver: KNITRO.  EXIT=0.  INFORM=0.
Interior/Direct NLP KNITRO
Locally optimal solution found

FuncEv  113 GradEv 1732 ConstrEv  112 ConJacEv 1732 Iter   79 MinorIter  109
CPU time: 19.593750 sec. Elapsed time: 20.375000 sec.
```

```
end

% Get final solution
t = subs(icollocate(t),solution);
r = subs(icollocate(r),solution);
theta = subs(icollocate(theta),solution);
vr = subs(icollocate(vr),solution);
vt = subs(icollocate(vt),solution);
ur = subs(icollocate(ur),solution);
```

```
ut = subs(icollocate(ut),solution);

t1 = 0:0.5:solution.t_f;
r_inter  = interp1p(t,r,t1);
theta_inter  = interp1p(t,theta,t1);
ur_inter = interp1p(t,ur,t1);
ut_inter = interp1p(t,ut,t1);

% Plot final solution
figure(1)
subplot(2,2,1)
plot(t,r,'*-');
legend('r');
title('radius');

subplot(2,2,2)
plot(t,theta,'*-');
legend('theta');
title('flight angle');

subplot(2,2,3)
plot(t,vr,'r*-',t,vt,'b*-');
legend('vr','vt');
title('velocity');

subplot(2,2,4)
plot(t,ur,'r+-',t,ut,'b+-');
legend('ur','ut');
title('Spacecraft controls');

figure(2)
polar(theta_inter,r_inter,'*-')
grid on
axis equal
```


radius
r
flight angle
theta
velocity
vr
vt
Spacecraft controls
ur
ut


90
4
120
60
3
150
2
30
1
180
0
210
330
240
300
270

67 Minimum Climb Time (English Units)

67.1 Problem description

Example about climbing (increase altitude)

Reference: [5]

67.2 Problem setup

```
alt0     = 0;
altf     = 65600;
speed0   = 424.26;
speedf   = 968.148;
fpa0     = 0;
fpaf     = 0;
mass0    = 42000/32.208;

altmin   = 0;
altmax   = 69000;
speedmin = 10;
speedmax = 3000;
fpamin   = -40*pi/180;
fpamax   = -fpamin;
massmin  = 50/32.208;

toms t t_f
p = tomPhase('p',t,0,t_f,50);
setPhase(p);

% Altitude, speed, flight path angle, mass
tomStates h v fpa m

% Angle of Attack
tomControls aalpha

guess = {
    t_f == 300;
    icollocate({
    h == alt0 + t/t_f*(altf-alt0);
    v == speed0 + t/t_f*(speedf-speed0);
    fpa == 10*pi/180;
    m == mass0;
    })};
```

```
cbox = {
    100 <= t_f <= 800;
    collocate(-pi*20/180 <= aalpha <= pi/20*180)
    icollocate(altmin <= h <= altmax)
    icollocate(speedmin <= v <= speedmax)
    icollocate(fpamin <= fpa <= fpamax)
    icollocate(massmin <= m <= mass0)
    };

bnd = {
    initial(h) == alt0;
    initial(v) == speed0;
    initial(fpa) == fpa0;
    initial(m) == mass0;
    final(h) == altf;
    final(v) == speedf;
    final(fpa) == fpaf;
    };

% US1976 data
hTab = (-2000:2000:86000);
rhoTab = [1.478 1.225 1.007 0.8193 0.6601 0.5258 0.4135 0.3119 ...
    0.2279 0.1665 0.1216 0.08891 0.06451 0.04694 0.03426 0.02508 ...
    0.01841 0.01355 0.009887 0.007257 0.005366 0.003995 0.002995 ...
    0.002259 0.001714 0.001317 0.001027 0.0008055 0.0006389 0.0005044 ...
    0.0003962 0.0003096 0.0002407 0.000186 0.0001429 0.0001091 ...
    8.281e-005 6.236e-005 4.637e-005 3.43e-005 2.523e-005 1.845e-005 ...
    1.341e-005 9.69e-006 6.955e-006];
sosTab = [347.9 340.3 332.5 324.6 316.5 308.1 299.5 295.1 295.1 ...
    295.1 295.1 295.1 296.4 297.7 299.1 300.4 301.7 303 306.5 310.1 ...
    313.7 317.2 320.7 324.1 327.5 329.8 329.8 328.8 325.4 322 318.6 ...
    315.1 311.5 308 304.4 300.7 297.1 293.4 290.7 288 285.3 282.5 ...
    279.7 276.9 274.1];

Mtab   = [0; 0.2; 0.4; 0.6; 0.8; 1; 1.2; 1.4; 1.6; 1.8];
alttab = [0 5000 10000 15000 20000 25000 30000 40000 50000 70000];
Ttab = 1000*[24.2 24.0  20.3 17.3 14.5 12.2 10.2 5.7 3.4 0.1;
    28.0 24.6 21.1 18.1 15.2 12.8 10.7 6.5 3.9 0.2;
    28.3 25.2 21.9 18.7 15.9 13.4 11.2 7.3 4.4 0.4;
    30.8 27.2 23.8 20.5 17.3 14.7 12.3 8.1 4.9 0.8;
    34.5 30.3 26.6 23.2 19.8 16.8 14.1 9.4 5.6 1.1;
    37.9 34.3 30.4 26.8 23.3 19.8 16.8 11.2 6.8 1.4;
    36.1 38.0 34.9 31.3 27.3 23.6 20.1 13.4 8.3 1.7;
    36.1 36.6 38.5 36.1 31.6 28.1 24.2 16.2 10.0 2.2;
    36.1 35.2 42.1 38.7 35.7 32.0 28.1 19.3 11.9 2.9;
    36.1 33.8 45.7 41.3 39.8 34.6 31.1 21.7 13.3 3.1];
```

```
M2     = [0 0.4 0.8 0.9 1.0 1.2 1.4 1.6 1.8];
Clalphatab = [3.44 3.44 3.44 3.58 4.44 3.44 3.01 2.86 2.44];
CD0tab    = [0.013 0.013 0.013 0.014 0.031 0.041 0.039 0.036 0.035];
etatab    = [0.54 0.54 0.54 0.75 0.79 0.78 0.89 0.93 0.93];
M         = Mtab;
alt       = alttab;
Re        = 20902900;
mmu        = 0.14076539e17;
S         = 530;
g0        = 32.208;
ISP       = 1600;
H         = 23800;
rho0      = 0.002378;

rho = interp1(hTab,rhoTab,h*0.3048,'pchip')*0.001941;
sos1 = interp1(hTab,sosTab,h*0.3048,'pchip')./0.3048;
Mach = v/sos1;

CD0       = interp1(M2,CD0tab,Mach,'pchip');
Clalpha   = interp1(M2,Clalphatab,Mach,'pchip');
eta       = interp1(M2,etatab,Mach,'pchip');

T = interp2(alttab, Mtab, Ttab, h, Mach, 'spline');
CD      = CD0 + eta.*Clalpha.*aalpha.^2;
CL      = Clalpha.*aalpha;
dynpres = 0.5.*rho.*v.^2;
D       = dynpres.*S.*CD;
L       = dynpres.*S.*CL;

equ = collocate({
    dot(h) == v.*sin(fpa);
    dot(v) == ((T.*cos(aalpha)-D)./m - mmu.*sin(fpa)./(Re+h).^2);
    dot(fpa) == (T.*sin(aalpha)+L)./(m.*v)+cos(fpa).*(v./(Re+h)-mmu./(v.*(Re+h).^2));
    dot(m) == -T./(g0.*ISP);
    });

options = struct;
options.name = 'Minimum Time to Climb (English)';
options.scale = 'auto';

% Strating guess of fpa is in confilct with the boundary conditions, but
% that's ok. (It will give a warning, which we suppress.)
warns = warning('off', 'tomSym:x0OutOfBounds');
```

67.3 Solve the problem

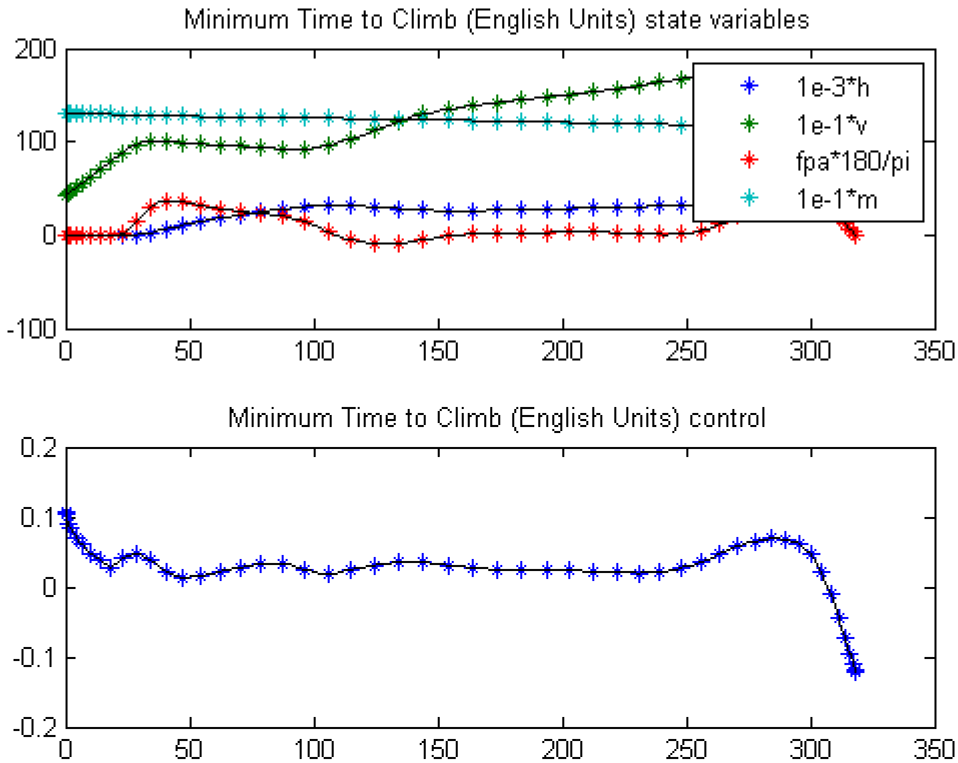
```
ezsolve(t_f,{cbox,bnd,equ},guess,options);
```

```
% Restore warning
warning(warns);
```

```
Problem type appears to be: lpcon
Auto-scaling
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Minimum Time to Climb (English)  f_k     318.348640856079270000
                                         sum(|constr|)      0.000000000036130174
                                f(x_k) + sum(|constr|)    318.348640856115420000
                                                f(x_0)    300.000000000000000000

Solver: snopt.  EXIT=0.  INFORM=3.
SNOPT 7.2-5 NLP code
Requested accuracy could not be achieved

FuncEv    1 ConstrEv   96 ConJacEv   95 Iter   51 MinorIter 2042
CPU time: 4.703125 sec. Elapsed time: 2.969000 sec.
```

67.4 Plot result

```
subplot(2,1,1)
ezplot([1e-3*h, 1e-1*v, fpa*180/pi, 1e-1*m])
legend('1e-3*h', '1e-1*v', 'fpa*180/pi', '1e-1*m');
title('Minimum Time to Climb (English Units) state variables');

subplot(2,1,2)
ezplot(aalpha);
title('Minimum Time to Climb (English Units) control');
```


Minimum Time to Climb (English Units) state variables
1e-3*h
1e-1*v
fpa*180/pi
1e-1*m
Minimum Time to Climb (English Units) control

68 Missile Intercept

Egwald Mathematics: Optimal Control, Intercept Missile, Elmer G. Wiens

68.1 Problem Description

Find scalar w over t in [0; t_F] to minimize:

$$J = 0$$

subject to:

$$\frac{dx_1}{dt} = x_3$$
$$\frac{dx_2}{dt} = x_4$$
$$\frac{dx_3}{dt} = cos(w)$$
$$\frac{dx_4}{dt} = sin(w)$$
$$x(t_0) = [0\ 0\ 0\ 0]$$
$$x(t_f) = [4 + 1.5 * t_f\ \ 1]$$
$$0 <= w <= \frac{pi}{4}$$

Reference: [35]

68.2 Problem setup

```
toms t t_f w
p = tomPhase('p', t, 0, t_f, 10);
setPhase(p);

tomStates x1 x2 x3 x4

% Initial guess
x0 = {t_f == 10; w == 0.2};
```

```
% Box constraints
cbox = {1 <= t_f <= 1e4
    0 <= w  <= pi/4};

% Boundary constraints
cbnd = {initial({x1 == 0; x2 == 0
    x3 == 0; x4 == 0})
    final({x1 == 4+1.5*t_f; x2 == 1})};

% ODEs and path constraints
ceq = collocate({dot(x1) == x3; dot(x2) == x4
    dot(x3) == cos(w); dot(x4) == sin(w)});

% Objective
objective = 0;
```

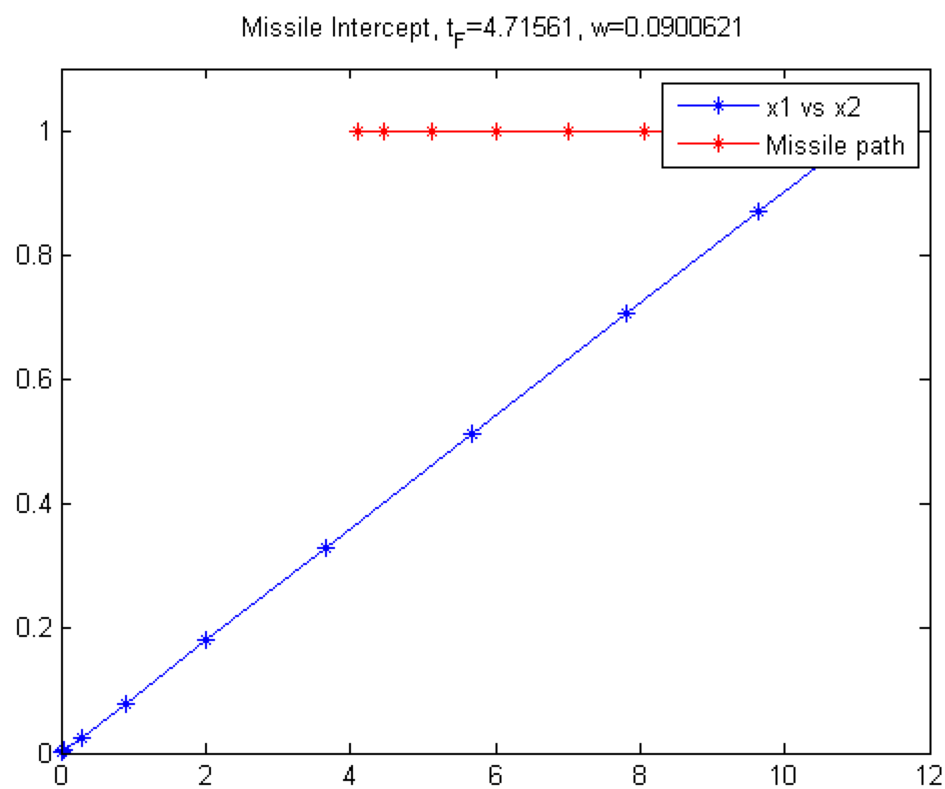
68.3 Solve the problem

```
options = struct;
options.name = 'Missile Intercept';
solution = ezsolve(objective, {cbox, cbnd, ceq}, x0, options);
t  = subs(collocate(t),solution);
x1 = subs(collocate(x1),solution);
x2 = subs(collocate(x2),solution);
w  = subs(w,solution);
t_f = subs(t_f,solution);
```

```
Problem type appears to be: lpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Missile Intercept              f_k       0.000000000000000000
                                       sum(|constr|)      0.000000010885019417
                              f(x_k) + sum(|constr|)      0.000000010885019417
                                              f(x_0)      0.000000000000000000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv   13 ConJacEv   13 Iter   10 MinorIter   43
CPU time: 0.031250 sec. Elapsed time: 0.031000 sec.
```

68.4 Plot result

```
figure(1);
plot(x1,x2,'*-');
hold on
plot(4+1.5*t,ones(length(t)),'-*r');
legend('x1 vs x2','Missile path');
title(sprintf('Missile Intercept, t_F=%g, w=%g',t_f,w));
ylim([0 1.1]);
```



69 Moonlander Example

Arthur Bryson - Dynamic Optimization

69.1 Problem description

Example about landing an object.

Reference: [9]

69.2 Problem setup

```
toms t
toms t_f
p = tomPhase('p', t, 0, t_f, 60);
setPhase(p);

tomStates altitude speed mass
tomControls thrust

% Initial guess
x0 = {t_f == 1.5
    icollocate({
    altitude == 1-t/t_f
    speed == -0.783+0.783*t/t_f
    mass == 1-0.99*t/t_f
    })
    collocate(thrust == 0)};

% Box constraints
cbox = {
    0  <= t_f                   <= 1000
    -20  <= icollocate(altitude) <= 20
    -20  <= icollocate(speed)    <= 20
    0.01  <= icollocate(mass)     <= 1
    0  <= collocate(thrust)    <= 1.227};

% Boundary constraints
cbnd = {initial({altitude == 1; speed == -0.783; mass == 1})
    final({altitude == 0; speed == 0})};

% ODEs and path constraints
exhaustvelocity = 2.349;
gravity         = 1;
```

```
ceq = collocate({
    dot(altitude) == speed
    dot(speed) == -gravity + thrust./mass
    dot(mass) == -thrust./exhaustvelocity});

% Objective
objective = integrate(thrust);
```

69.3 Solve the problem

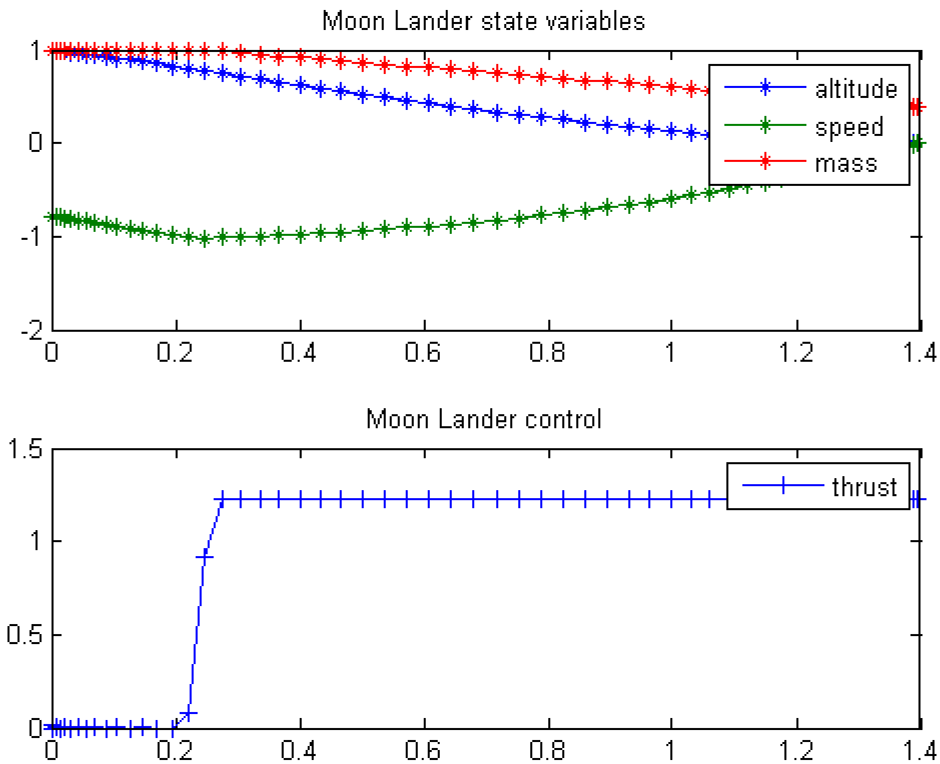
```
options = struct;
options.name = 'Moon Lander';
solution = ezsolve(objective, {cbox, cbnd, ceq}, x0, options);
t  = subs(collocate(t),solution);
altitude = subs(collocate(altitude),solution);
speed  = subs(collocate(speed),solution);
mass   = subs(collocate(mass),solution);
thrust = subs(collocate(thrust),solution);
```

```
Problem type appears to be: qpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Moon Lander                    f_k       1.420346223923628400
                                       sum(|constr|)      0.000000000118734605
                              f(x_k) + sum(|constr|)      1.420346224042363000
                                              f(x_0)      0.000000000000000000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv   70 ConJacEv   70 Iter   21 MinorIter 1434
CPU time: 0.937500 sec. Elapsed time: 0.984000 sec.
```

69.4 Plot result

```
subplot(2,1,1)
plot(t,altitude,'*-',t,speed,'*-',t,mass,'*-');
legend('altitude','speed','mass');
title('Moon Lander state variables');

subplot(2,1,2)
plot(t,thrust,'+-');
```

```
legend('thrust');
title('Moon Lander control');
```



70 Nagurka Problem

ITERATIVE DYNAMIC PROGRAMMING, REIN LUUS

6.4 Further example

CHAPMAN & HALL/CRC Monographs and Surveys in Pure and Applied Mathematics

n'th-order linear time-invariant system.

70.1 Problem description

Find u over t in [0; 1] to minimize

$$J = \int_0^1 x' * x + u' * u \mathrm{d}t + 10 * x_1(t_F)^2$$

subject to:

$$\frac{dx}{dt} = A * x + u$$

```
A = [0 1 0 ... 0
     0 0 1 ... 0
     ... ... ...
     0 0 0 ... 1
     1 -2 3 ... (-1)^(n+1)*n]
```

The initial condition are:

$$x(0) = [1\ 2\ ...\ n],$$

$$-\infty <= u(1:n) <= \infty.$$

Reference: [25]

70.2 Problem setup

```
toms t
n  = 6;
t_F = 1;

p = tomPhase('p', t, 0, t_F, 25);
setPhase(p);

x = tomState('x', n, 1);
u = tomState('u', n, 1);

nvec = (1:n);
A = [sparse(n-1,1), speye(n-1); ...
    sparse(nvec.*(-1).^(nvec+1))];

% Initial guess
guess = icollocate(x == nvec');

% Initial conditions
cinit = (initial(x) == nvec');

% ODEs and path constraints
ceq = collocate(dot(x) == A*x+u);

% Objective
objective = 10*final(x(1))^2 + integrate(x'*x + u'*u);
```

70.3 Solve the problem

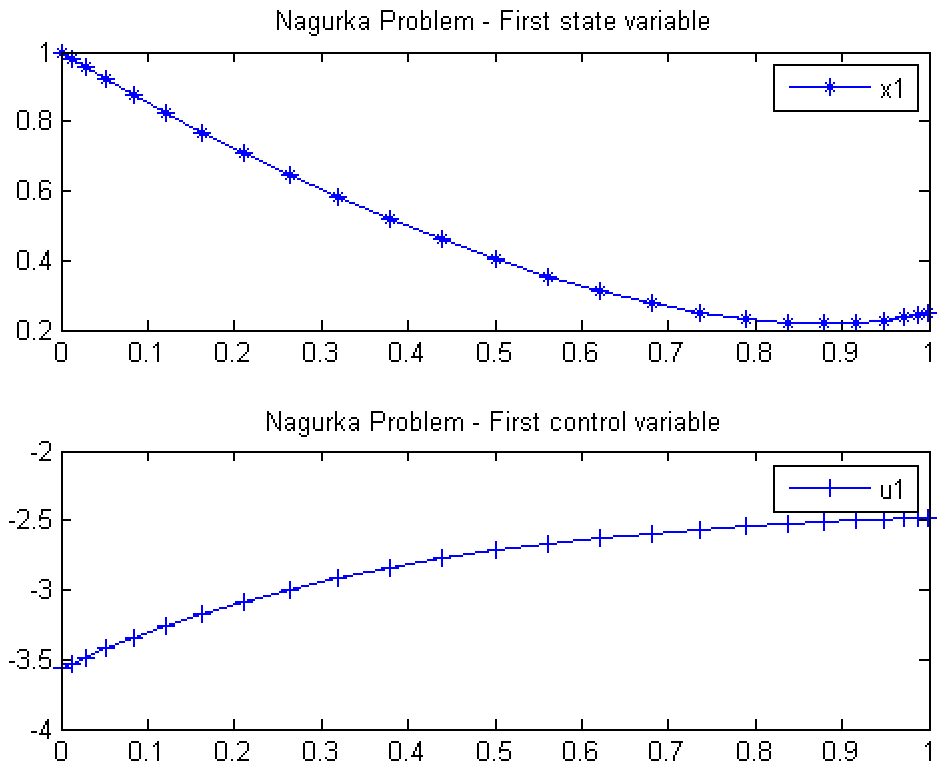
```
options = struct;
options.name = 'Nagurka Problem';
solution = ezsolve(objective, {ceq, cinit}, guess, options);
t = subs(collocate(t),solution);
x = subs(collocate(x),solution);
u = subs(collocate(u),solution);
```

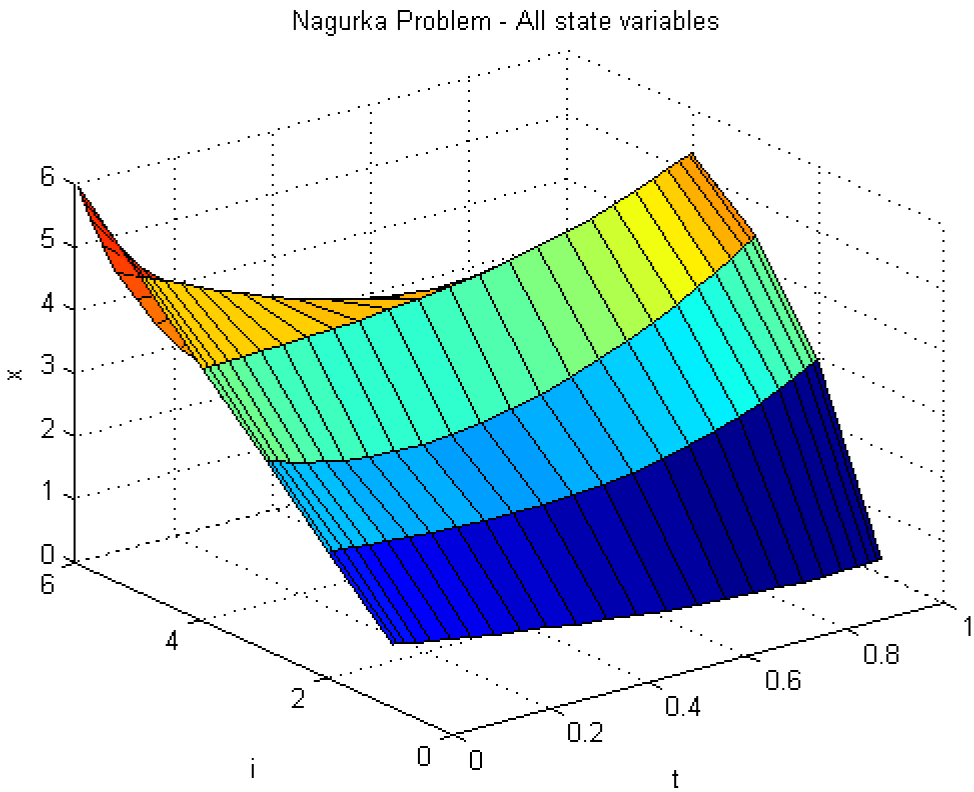
```
Problem type appears to be: qp
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem:  1: Nagurka Problem                     f_k     109.074347751904550000
                                       sum(|constr|)      0.000000000002256926
                              f(x_k) + sum(|constr|)    109.074347751906810000
                                              f(x_0)      0.000000000000000000
```

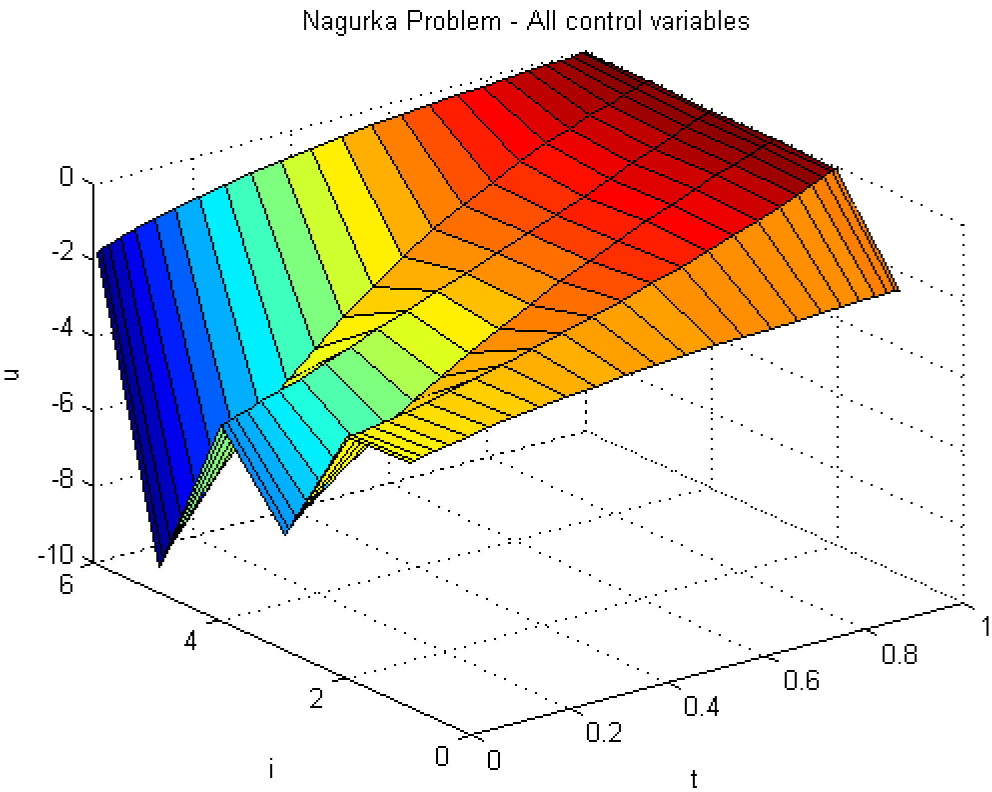
```
Solver: CPLEX.  EXIT=0.  INFORM=1.
CPLEX Barrier QP solver
Optimal solution found

FuncEv    3 GradEv    3 ConstrEv    3 Iter    3
CPU time: 0.046875 sec. Elapsed time: 0.032000 sec.
```

70.4 Plot result

```
subplot(2,1,1)
x1 = x(:,1);
plot(t,x1,'*-');
legend('x1');
title('Nagurka Problem - First state variable');

subplot(2,1,2)
u1 = u(:,1);
plot(t,u1,'+-');
legend('u1');
title('Nagurka Problem - First control variable');

figure(2)
surf(t, 1:n, x')
xlabel('t'); ylabel('i'); zlabel('x');
title('Nagurka Problem - All state variables');

figure(3)
surf(t, 1:n, u')
xlabel('t'); ylabel('i'); zlabel('u');
title('Nagurka Problem - All control variables');
```


Nagurka Problem - First state variable
x1
Nagurka Problem - First control variable
u1


Nagurka Problem - All state variables
x
0
1
2
3
4
5
6
i
0
2
4
6
t
0
0.2
0.4
0.6
0.8
1


Nagurka Problem - All control variables
0
-2
-4
-6
-8
-10
u
6
4
2
0
i
0
0.2
0.4
0.6
0.8
1
t

71 Nishida problem

Second-order sensitivities of general dynamic systems with application to optimal control problems. 1999, Vassilios S. Vassiliadis, Eva Balsa Canto, Julio R. Banga

Case Study 6.3: Nishida problem

71.1 Problem description

This case study was presented by Nishida et al. (1976) and it is posed as follows:

Minimize:

$$J = x_1(t_f)^2 + x_2(t_f)^2 + x_3(t_f)^2 + x_4(t_f)^2$$

subject to:

$$\begin{aligned}
\frac{dx_1}{dt} &= -0.5 * x_1 + 5 * x_2 \\
\frac{dx_2}{dt} &= -5 * x_1 - 0.5 * x_2 + u \\
\frac{dx_3}{dt} &= -0.6 * x_3 + 10 * x_4 \\
\frac{dx_4}{dt} &= -10 * x_3 - 0.6 * x_4 + u \\
-1.0 &<= u <= 1.0
\end{aligned}$$

with the initial conditions: x(i) = 10, i=1,..,4 and with t_f = 4.2.

Reference: [31]

71.2 Problem setup

```
toms t
p = tomPhase('p', t, 0, 4.2, 60);
setPhase(p);

tomStates x1 x2 x3 x4
tomControls u
```

```
% Initial guess
x0 = {icollocate({
    x1 == 10-10*t/4.2; x2 == 10-10*t/4.2
    x3 == 10-10*t/4.2; x4 == 10-10*t/4.2})
    collocate(u == 0)};

% Box constraints
cbox = {icollocate({
    -15 <= x1 <= 15; -15 <= x2 <= 15
    -15 <= x3 <= 15; -15 <= x4 <= 15})
    -1  <= collocate(u) <= 1};

% Boundary constraints
cbnd = initial({x1 == 10; x2 == 10
    x3 == 10; x4 == 10});

% ODEs and path constraints
ceq = collocate({
    dot(x1) == -0.5*x1+5*x2
    dot(x2) == -5*x1-0.5*x2+u
    dot(x3) == -0.6*x3+10*x4
    dot(x4) == -10*x3-0.6*x4+u});

% Objective
objective = final(x1)^2+final(x2)^2+final(x3)^2+final(x4)^2;
```

71.3 Solve the problem

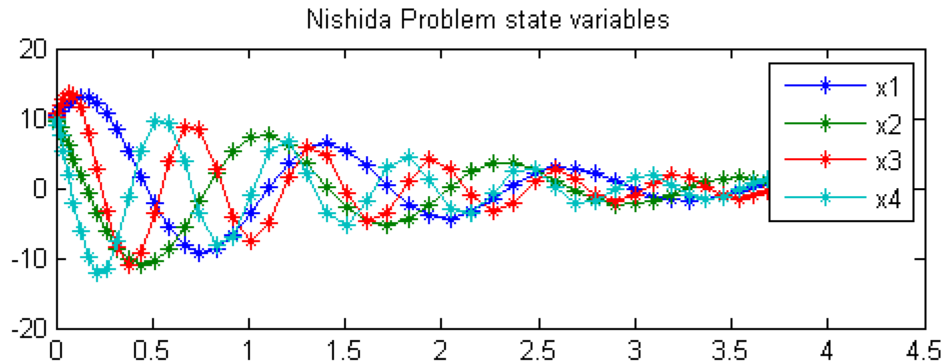
```
options = struct;
options.name = 'Nishida Problem';
solution = ezsolve(objective, {cbox, cbnd, ceq}, x0, options);
t  = subs(collocate(t),solution);
x1 = subs(collocate(x1),solution);
x2 = subs(collocate(x2),solution);
x3 = subs(collocate(x3),solution);
x4 = subs(collocate(x4),solution);
u  = subs(collocate(u),solution);
```

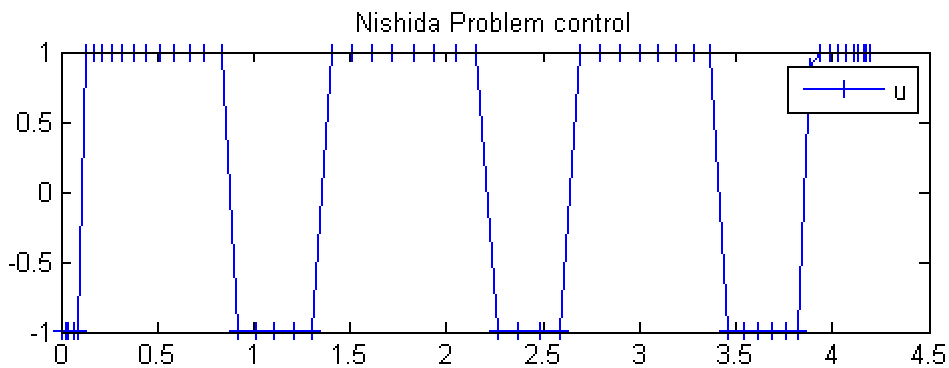
```
Problem type appears to be: qp
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem:  1: Nishida Problem                    f_k       1.004684962685394000
                                       sum(|constr|)      0.000018521427591828
```

```
                    f(x_k) + sum(|constr|)      1.004703484112985800
                                    f(x_0)      0.000000000000000000

Solver: CPLEX.  EXIT=0.  INFORM=1.
CPLEX Barrier QP solver
Optimal solution found

FuncEv   15 GradEv   15 ConstrEv   15 Iter   15
CPU time: 0.140625 sec. Elapsed time: 0.125000 sec.
```

71.4 Plot result

```
subplot(2,1,1)
plot(t,x1,'*-',t,x2,'*-',t,x3,'*-',t,x4,'*-');
legend('x1','x2','x3','x4');
title('Nishida Problem state variables');

subplot(2,1,2)
plot(t,u,'+-');
legend('u');
title('Nishida Problem control');
```


Nishida Problem state variables
20
10
0
-10
-20
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
x1
x2
x3
x4


Nishida Problem control
1
0.5
0
-0.5
-1
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
u

72 Nondifferentiable system

Global Optimization of Chemical Processes using Stochastic Algorithms 1996, Julio R. Banga, Warren D. Seider

Case Study IV: Optimal control of a nondifferentiable system

72.1 Problem description

This problem has been studied by Thomopoulos and Papadakis who report convergence difficulties using several optimization algorithms and by Luus using Iterative Dynamic Programming. The optimal control problem is:

Find u(t) to minimize:

$$J = x_3(t_f)$$

Subject to:

$$\frac{dx_1}{dt} = x_2$$
$$\frac{dx_2}{dt} = -x_1 - x_2 + u + d$$
$$\frac{dx_3}{dt} = 5 * x_1^2 + 2.5 * x_2^2 + 0.5 * u^2$$

with

$$d = 100 * [U(t - 0.5) - U(t - 0.6)]$$

where U(t-alpha) is the unit function such that U = 0 for t - alpha < 0 and U = 1 for t - alpha > 0. Hence d is a rectangular pulse of magnitude 100 from t=0.5 until t=0.6. These authors consider t_f = 2.0s and the initial conditions:

$$x(t_0) = [0\ 0\ 0]'$$

Reference: [4]

72.2 Problem setup

```
toms t1
p1 = tomPhase('p1', t1, 0, 0.5, 30);
toms t2
p2 = tomPhase('p2', t2, 0.5, 0.1, 20);
toms t3
p3 = tomPhase('p3', t3, 0.6, 2-0.6, 50);

x1p1 = tomState(p1,'x1p1');
x2p1 = tomState(p1,'x2p1');
x3p1 = tomState(p1,'x3p1');
up1 = tomControl(p1,'up1');

x1p2 = tomState(p2,'x1p2');
x2p2 = tomState(p2,'x2p2');
x3p2 = tomState(p2,'x3p2');
up2 = tomControl(p2,'up2');

x1p3 = tomState(p3,'x1p3');
x2p3 = tomState(p3,'x2p3');
x3p3 = tomState(p3,'x3p3');
up3 = tomControl(p3,'up3');

% Initial guess
x0 = {icollocate(p1,{x1p1 == 0;x2p1 == 0;x3p1 == 0})
    icollocate(p2,{x1p2 == 0;x2p2 == 0;x3p2 == 0})
    icollocate(p3,{x1p3 == 0;x2p3 == 0;x3p3 == 0})
    collocate(p1,up1==-4-8*t1/0.5)
    collocate(p2,up2==-12)
    collocate(p3,up3==-12+14*t3/2)};

% Box constraints
cbox = {
    icollocate(p1,{-100 <= x1p1 <= 100
    -100 <= x2p1 <= 100
    -100 <= x3p1 <= 100})
    icollocate(p2,{-100 <= x1p2 <= 100
    -100 <= x2p2 <= 100
    -100 <= x3p2 <= 100})
    icollocate(p3,{-100 <= x1p3 <= 100
    -100 <= x2p3 <= 100
    -100 <= x3p3 <= 100})
    collocate(p1,-15 <= up1 <= 2)
    collocate(p2,-15 <= up2 <= 2)
    collocate(p3,-15 <= up3 <= 2)};
```

```
% Boundary constraints
cbnd = {initial(p1,{x1p1 == 0;x2p1 == 0;x3p1 == 0})
    final(p3,x3p3) <= 60};

% ODEs and path constraints
ceq = {collocate(p1,{
    dot(p1,x1p1) == x2p1
    dot(p1,x2p1) == -x1p1-x2p1+up1
    dot(p1,x3p1) == 5*x1p1.^2+2.5*x2p1.^2+0.5*up1.^2})
    collocate(p2,{
    dot(p2,x1p2) == x2p2
    dot(p2,x2p2) == -x1p2-x2p2+up2+100
    dot(p2,x3p2) == 5*x1p2.^2+2.5*x2p2.^2+0.5*up2.^2})
    collocate(p3,{
    dot(p3,x1p3) == x2p3
    dot(p3,x2p3) == -x1p3-x2p3+up3
    dot(p3,x3p3) == 5*x1p3.^2+2.5*x2p3.^2+0.5*up3.^2})};

% Objective
objective = final(p3,x3p3);

% Link phase
link = {final(p1,x1p1) == initial(p2,x1p2)
    final(p1,x2p1) == initial(p2,x2p2)
    final(p1,x3p1) == initial(p2,x3p2)
    final(p2,x1p2) == initial(p3,x1p3)
    final(p2,x2p2) == initial(p3,x2p3)
    final(p2,x3p2) == initial(p3,x3p3)};
```

72.3 Solve the problem

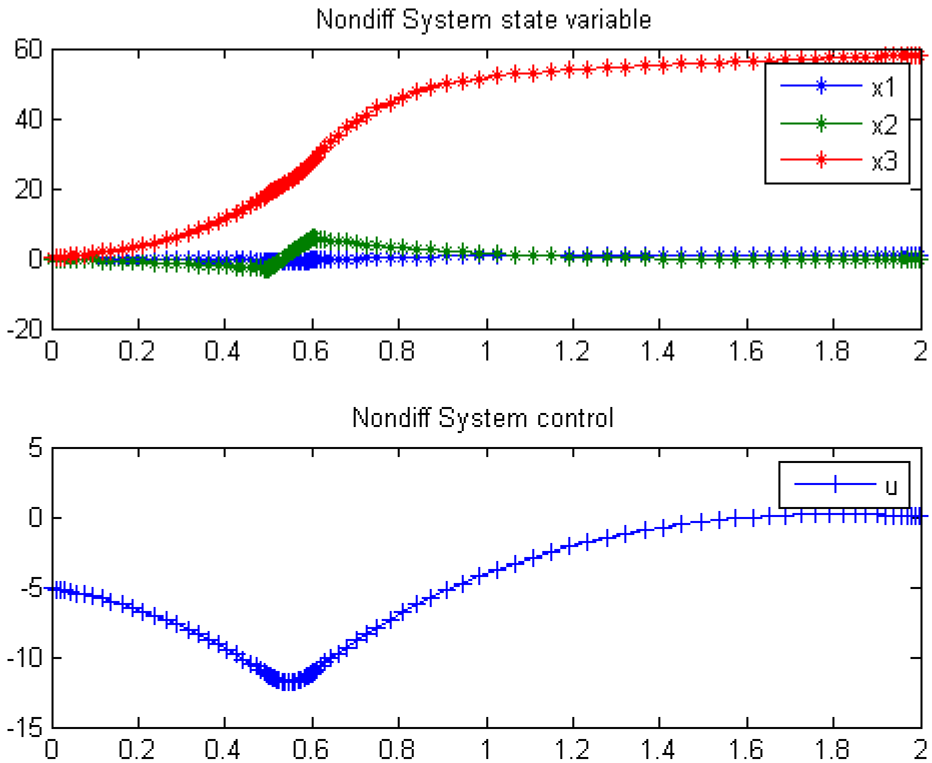
```
options = struct;
options.name = 'Nondiff System';
constr = {cbox, cbnd, ceq, link};
solution = ezsolve(objective, constr, x0, options);

t = subs(collocate(p1,t1),solution);
t = [t;subs(collocate(p2,t2),solution)];
t = [t;subs(collocate(p3,t3),solution)];
x1 = subs(collocate(p1,x1p1),solution);
x1 = [x1;subs(collocate(p2,x1p2),solution)];
x1 = [x1;subs(collocate(p3,x1p3),solution)];
x2 = subs(collocate(p1,x2p1),solution);
x2 = [x2;subs(collocate(p2,x2p2),solution)];
x2 = [x2;subs(collocate(p3,x2p3),solution)];
x3 = subs(collocate(p1,x3p1),solution);
x3 = [x3;subs(collocate(p2,x3p2),solution)];
```

```
x3 = [x3;subs(collocate(p3,x3p3),solution)];
u = subs(collocate(p1,up1),solution);
u = [u;subs(collocate(p2,up2),solution)];
u = [u;subs(collocate(p3,up3),solution)];
```

```
Problem type appears to be: lpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Nondiff System                 f_k      58.065028469582764000
                                       sum(|constr|)      0.000030760875102477
                              f(x_k) + sum(|constr|)     58.065059230457869000
                                              f(x_0)      0.000000000000000000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv   45 ConJacEv   45 Iter   38 MinorIter 1056
CPU time: 1.437500 sec. Elapsed time: 1.516000 sec.
```

72.4 Plot result

```
subplot(2,1,1)
plot(t,x1,'*-',t,x2,'*-',t,x3,'*-');
legend('x1','x2','x3');
title('Nondiff System state variable');

subplot(2,1,2)
plot(t,u,'+-');
legend('u');
title('Nondiff System control');
```


Nondiff System state variable
60
40
20
0
-20
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
x1
x2
x3
Nondiff System control
5
0
-5
-10
-15
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
u

73 Nonlinear CSTR

Dynamic optimization of chemical and biochemical processes using restricted second-order information 2001, Eva Balsa-Canto, Julio R. Banga, Antonio A. Alonso Vassilios S. Vassiliadis

Case Study III: Nonlinear CSTR

73.1 Problem description

The problem was first introduced by Jensen (1964) and consists of determining the four optimal controls of a chemical reactor in order to obtain maximum economic benefit. The system dynamics describe four simultaneous chemical reactions taking place in an isothermal continuous stirred tank reactor. The controls are the flow rates of three feed streams and an electrical energy input used to promote a photochemical reaction. Luus (1990) and Bojkov, Hansel, and Luus (1993) considered two sub-cases using three and four control variables respectively.

The problem is formulated as follows:Find u1(t), u2(t), u3(t) and u4(t) over t in [t0,t_f] to maximize:

$$J = x_8(t_f)$$

Subject to:

$$\frac{dx_1}{dt} = u_4 - q * x_1 - 17.6 * x_1 * x_2 - 23 * x_1 * x_6 * u_3$$

$$\frac{dx_2}{dt} = u_1 - q * x_2 - 17.6 * x_1 * x_2 - 146 * x_2 * x_3$$

$$\frac{dx_3}{dt} = u_2 - q * x_3 - 73 * x_2 * x_3$$

$$\frac{dx_4}{dt} = -q * x_4 + 35.2 * x_1 * x_2 - 51.3 * x_4 * x_5$$

$$\frac{dx_5}{dt} = -q * x_5 + 219 * x_2 * x_3 - 51.3 * x_4 * x_5$$

$$\frac{dx_6}{dt} = -q * x_6 + 102.6 * x_4 . * x_5 - 23 * x_1 * x_6 * u_3$$

$$\frac{dx_7}{dt} = -q * x_7 + 46 * x_1 * x_6 * u_3$$

$$\frac{dx_8}{dt} = 5.8 * (q * x_1 - u_4) - 3.7 * u_1 - 4.1 * u_2 +$$

$$q * (23 * x_4 + 11 * x_5 + 28 * x_6 + 35 * x_7) - 5 * u_3^2 - 0.099$$

where:

$$q = u_1 + u_2 + u_4;$$

with the initial conditions:

$$x(t_0) = [0.1883\ 0.2507\ 0.0467\ 0.0899\ 0.1804\ 0.1394\ 0.1046\ 0.000]'$$

And the following bounds on the control variables:

$$0 <= u1 <= 20$$
$$0 <= u2 <= 6$$
$$0 <= u3 <= 4$$
$$0 <= u4 <= 20$$

The final time is considered fixed as t_f = 0.2.

Reference: [1]

73.2 Problem setup

```
toms t
```

73.3 Solve the problem, using a successively larger number collocation points

```
for n=[5 20 60]

    p = tomPhase('p', t, 0, 0.2, n);
    setPhase(p);

    tomStates x1 x2 x3 x4 x5 x6 x7 x8
    tomControls u1 u2 u3 u4

    % Interpolate an initial guess for the n collocation points
    if n == 5
        x0 = {};
    else
```

```
    x0 = {icollocate({x1 == x1opt; x2 == x2opt
        x3 == x3opt; x4 == x4opt; x5 == x5opt
        x6 == x6opt; x7 == x7opt; x8 == x8opt})
        collocate({u1 == u1opt; u2 == u2opt
        u3 == u3opt; u4 == u4opt})};
end

% Box constraints
cbox = {icollocate({
    0 <= x1; 0 <= x2; 0 <= x3
    0 <= x4; 0 <= x5; 0 <= x6
    0 <= x7; 0 <= x8})
    collocate({
    0 <= u1 <= 20; 0 <= u2 <= 6
    0 <= u3 <= 4;  0 <= u4 <= 20})};

% Boundary constraints
cbnd = initial({x1 == 0.1883; x2 == 0.2507
    x3 == 0.0467; x4 == 0.0899; x5 == 0.1804
    x6 == 0.1394; x7 == 0.1064; x8 == 0});

% ODEs and path constraints
% 4.1*u2+(u1+u2.*u4) in another paper, -0.09 instead of -0.099
q = u1+u2+u4;
ceq = collocate({
    dot(x1) == (u4-q.*x1-17.6*x1.*x2-23*x1.*x6.*u3)
    dot(x2) == (u1-q.*x2-17.6*x1.*x2-146*x2.*x3)
    dot(x3) == (u2-q.*x3-73*x2.*x3)
    dot(x4) == (-q.*x4+35.2*x1.*x2-51.3*x4.*x5)
    dot(x5) == (-q.*x5+219*x2.*x3-51.3*x4.*x5)
    dot(x6) == (-q.*x6+102.6*x4.*x5-23*x1.*x6.*u3)
    dot(x7) == (-q.*x7+46*x1.*x6.*u3)
    dot(x8) == (5.8*(q.*x1-u4)-3.7*u1-4.1*u2+q.*...
    (23*x4+11*x5+28*x6+35*x7)-5*u3.^2-0.099)});

% Objective
objective = -final(x8);
```

73.4 Solve the problem

```
options = struct;
options.name = 'Nonlinear CSTR';
solution = ezsolve(objective, {cbox, cbnd, ceq}, x0, options);

% Optimal x and u as starting point
x1opt = subs(x1, solution);
x2opt = subs(x2, solution);
```

```
    x3opt = subs(x3, solution);
    x4opt = subs(x4, solution);
    x5opt = subs(x5, solution);
    x6opt = subs(x6, solution);
    x7opt = subs(x7, solution);
    x8opt = subs(x8, solution);
    u1opt = subs(u1, solution);
    u2opt = subs(u2, solution);
    u3opt = subs(u3, solution);
    u4opt = subs(u4, solution);
```

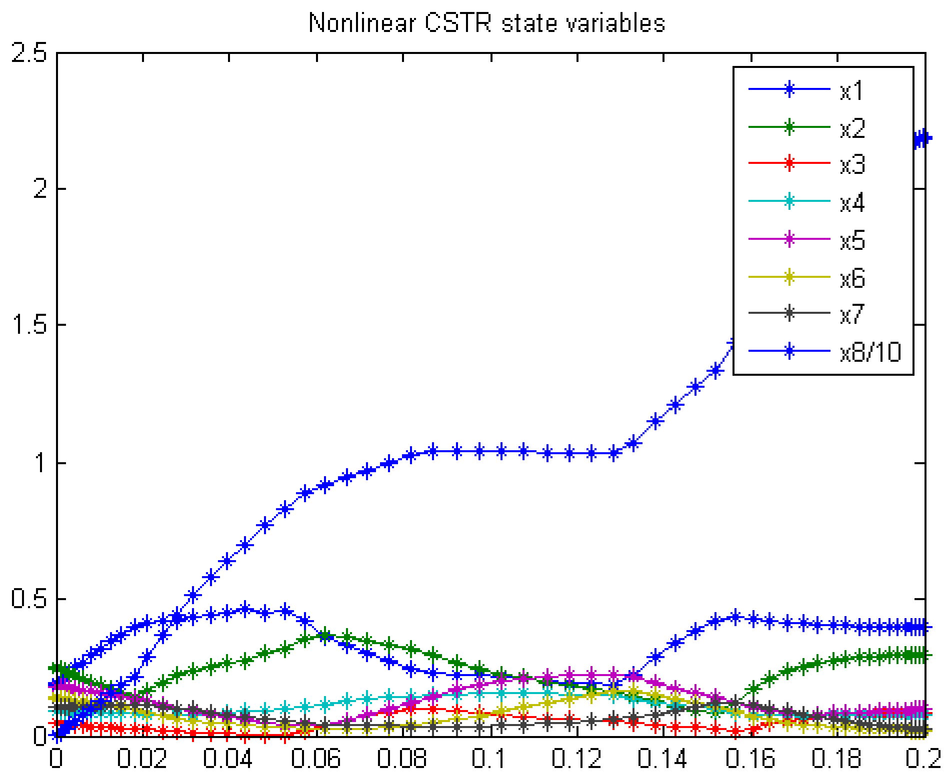
```
Problem type appears to be: lpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Nonlinear CSTR                 f_k     -21.841502289865435000
                                       sum(|constr|)      0.000000000210565355
                              f(x_k) + sum(|constr|)    -21.841502289654869000
                                              f(x_0)      0.000000000000000000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv   53 ConJacEv   53 Iter   41 MinorIter  342
CPU time: 0.453125 sec. Elapsed time: 0.500000 sec.


Problem type appears to be: lpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Nonlinear CSTR                 f_k     -21.896802275281718000
                                       sum(|constr|)      0.000000001587400641
                              f(x_k) + sum(|constr|)    -21.896802273694316000
                                              f(x_0)    -21.841502289865460000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv   96 ConJacEv   96 Iter   91 MinorIter  380
CPU time: 1.500000 sec. Elapsed time: 1.547000 sec.
```

```
Problem type appears to be: lpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Nonlinear CSTR                 f_k     -21.887245712594538000
                                       sum(|constr|)      0.000000000445950436
                              f(x_k) + sum(|constr|)    -21.887245712148587000
                                              f(x_0)    -21.896802275281658000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv  277 ConJacEv  277 Iter  258 MinorIter 1045
CPU time: 40.765625 sec. Elapsed time: 42.203000 sec.
```

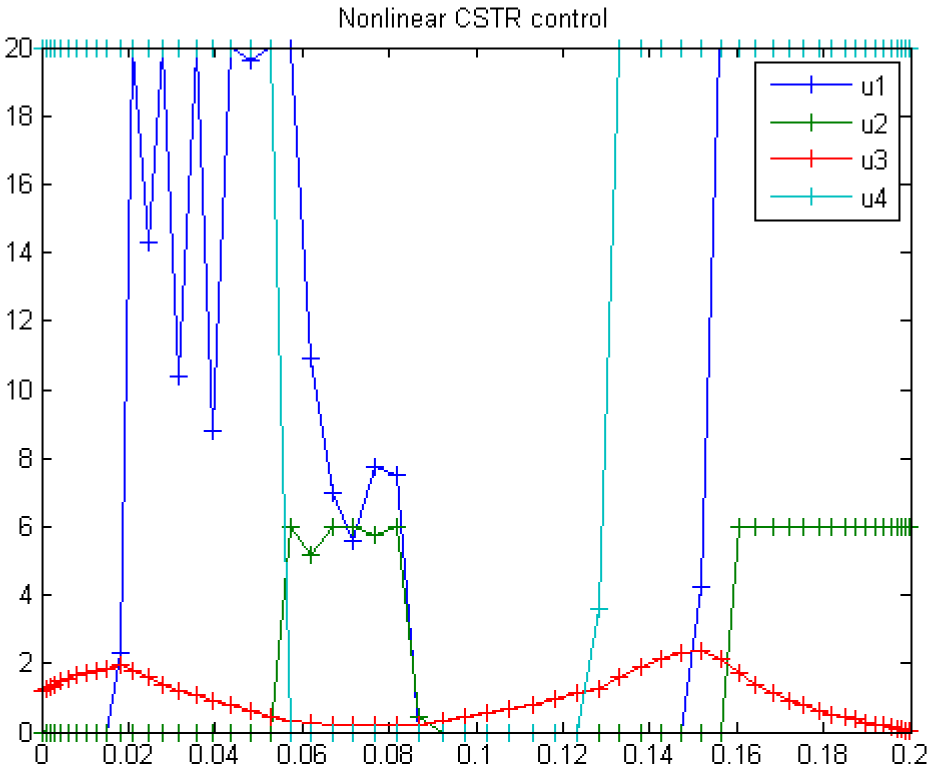
```
end

t  = subs(collocate(t),solution);
x1 = collocate(x1opt);
x2 = collocate(x2opt);
x3 = collocate(x3opt);
x4 = collocate(x4opt);
x5 = collocate(x5opt);
x6 = collocate(x6opt);
x7 = collocate(x7opt);
x8 = collocate(x8opt);
u1 = collocate(u1opt);
u2 = collocate(u2opt);
u3 = collocate(u3opt);
u4 = collocate(u4opt);
```

73.5 Plot result

```
figure(1)
plot(t,x1,'*-',t,x2,'*-',t,x3,'*-',t,x4,'*-' ...
    ,t,x5,'*-',t,x6,'*-',t,x7,'*-',t,x8/10,'*-');
legend('x1','x2','x3','x4','x5','x6','x7','x8/10');
title('Nonlinear CSTR state variables');

figure(2)
plot(t,u1,'+-',t,u2,'+-',t,u3,'+-',t,u4,'+-');
legend('u1','u2','u3','u4');
title('Nonlinear CSTR control');
```



Nonlinear CSTR state variables
2.5
2
1.5
1
0.5
0
0
0.02
0.04
0.06
0.08
0.1
0.12
0.14
0.16
0.18
0.2
x1
x2
x3
x4
x5
x6
x7
x8/10



Nonlinear CSTR control
u1
u2
u3
u4
0
2
4
6
8
10
12
14
16
18
20
0
0.02
0.04
0.06
0.08
0.1
0.12
0.14
0.16
0.18
0.2

74 Obstacle Avoidance

OPTRAGEN 1.0 A MATLAB Toolbox for Optimal Trajectory Generation, Raktim Bhattacharya, Texas A&M University (Note: There is typographical error in the OPTRAGEN documentation. The objective involves second derivatives of x and y.)

A robot with obstacles in 2D space. Travel from point A to B using minimum energy.

74.1 Problem Formulation

Find theta(t) and V over t in [0; 1] to minimize

$$\int_0^1 ((\frac{d^2x}{dt^2})^2 + (\frac{d^2y}{dt^2})^2)\mathrm{d}t$$

subject to:

$$\frac{dx}{dt} = V * cos(theta)$$
$$\frac{dy}{dt} = V * sin(theta)$$
$$[x_0 \ y_0] = [0 \ 0]$$
$$[x_1 \ y_1] = [1.2 \ 1.6]$$
$$(x - 0.4)^2 + (y - 0.5)^2 >= 0.1$$
$$(x - 0.8)^2 + (y - 1.5)^2 >= 0.1$$

Where V is a constant scalar speed.

Reference: [6]

74.2 Solve the problem, using a successively larger number collocation points

```
for n=[4 15 30]

    % Create a new phase and states, using n collocation points
    p = tomPhase('p', t, 0, t_f, n);
    setPhase(p);
    tomStates x y vx vy
```

```
tomControls theta

% Interpolate an initial guess for the n collocation points
x0 = {V == speed
    icollocate({x == xopt; y == yopt; vx == vxopt; vy == vyopt})
    collocate(theta == thetaopt)};

% Box constraints
cbox = {0 <= V <= 100 };

% Boundary constraints
cbnd = {initial({x == 0; y == 0})
    final({x == 1.2; y == 1.6})};

% ODEs and path constraints
ode = collocate({
    dot(x) == vx == V*cos(theta)
    dot(y) == vy == V*sin(theta)
    });

% A 30th order polynomial is more than sufficient to give good
% accuracy. However, that means that mcollocate would only check
% about 60 points. In order to make sure we don't hit an obstacle,
% we check 300 evenly spaced points instead, using atPoints.
obstacles = atPoints(linspace(0,t_f,300), {
    (x-0.4)^2 + (y-0.5)^2 >= 0.1
    (x-0.8)^2 + (y-1.5)^2 >= 0.1});

% Objective: minimum energy.
objective = integrate(dot(vx)^2+dot(vy)^2);
```

74.3 Solve the problem

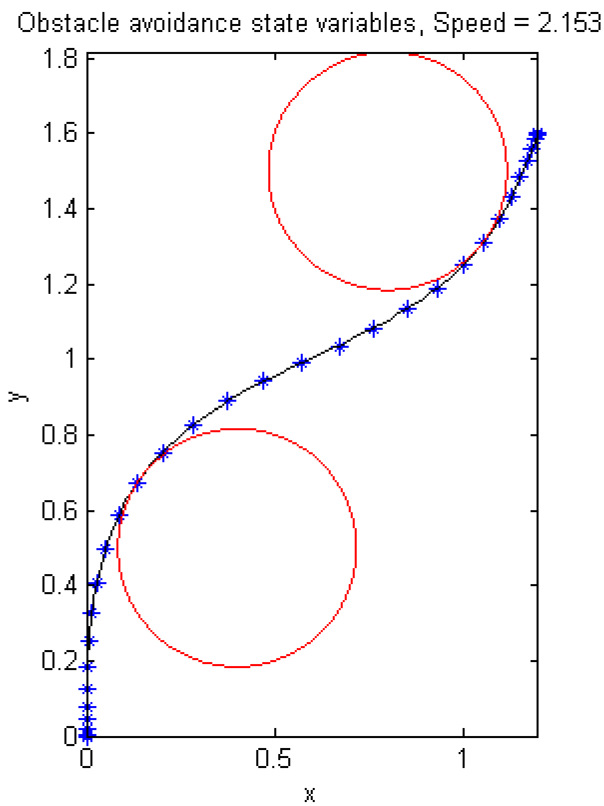
```
options = struct;
options.name = 'Obstacle avoidance';
constr = {cbox, cbnd, ode, obstacles};
solution = ezsolve(objective, constr, x0, options);

% Optimal x, y, and speed, to use as starting guess in the next iteration
xopt = subs(x, solution);
yopt = subs(y, solution);
vxopt = subs(vx, solution);
vyopt = subs(vy, solution);
thetaopt = subs(theta, solution);
speed = subs(V,solution);
```

```
Problem type appears to be: qpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Obstacle avoidance             f_k      29.812856165009947000
                                       sum(|constr|)      0.000000001309307815
                              f(x_k) + sum(|constr|)     29.812856166319254000
                                              f(x_0)      0.000000000000062528

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv   22 ConJacEv   22 Iter   20 MinorIter 2732
CPU time: 0.609375 sec. Elapsed time: 0.688000 sec.


Problem type appears to be: qpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Obstacle avoidance             f_k      22.128728366250083000
                                       sum(|constr|)      0.000000000006744707
                              f(x_k) + sum(|constr|)     22.128728366256826000
                                              f(x_0)     29.812856165010601000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv  151 ConJacEv  151 Iter  136 MinorIter  488
CPU time: 2.437500 sec. Elapsed time: 2.703000 sec.


Problem type appears to be: qpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Obstacle avoidance             f_k      22.091923280888466000
                                       sum(|constr|)      0.000000000011942997
                              f(x_k) + sum(|constr|)     22.091923280900410000
                                              f(x_0)     22.128728366249423000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
```

```
Optimality conditions satisfied

FuncEv    1 ConstrEv  289 ConJacEv  289 Iter  261 MinorIter  697
CPU time: 9.437500 sec. Elapsed time: 9.922000 sec.


end
```

74.4 Plot result

```
figure(1)
th = linspace(0,2*pi,500);
x1 = sqrt(0.1)*cos(th)+0.4;
y1 = sqrt(0.1)*sin(th)+0.5;
x2 = sqrt(0.1)*cos(th)+0.8;
y2 = sqrt(0.1)*sin(th)+1.5;
ezplot(x,y);
hold on
plot(x1,y1,'r',x2,y2,'r');
hold off
xlabel('x');
ylabel('y');
title(sprintf('Obstacle avoidance state variables, Speed = %2.4g',speed));
axis image
```


Obstacle avoidance state variables, Speed = 2.153
y
x
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
0
0.5
1

75 Oil Shale Pyrolysis

Dynamic Optimization of Batch Reactors Using Adaptive Stochastic Algorithms 1997, Eugenio F. Carrasco, Julio R. Banga

Case Study II: Oil Shale Pyrolysis

75.1 Problem description

A very challenging optimal control problem is the oil shale pyrolysis case study, as considered by Luus (1994). The system consists of a series of five chemical reactions:

A1 -> A2

A2 -> A3

A1+A2 -> A2+A2

A1+A2 -> A3+A2

A1+A2 -> A4+A2

This system is described by the differential equations

$$
\begin{aligned}
\frac{dx_1}{dt} &= -k_1 * x_1 - (k_3 + k_4 + k_5) * x_1 * x_2 \\
\frac{dx_2}{dt} &= k_1 * x_1 - k_2 * x_2 + k_3 * x_1 * x_2 \\
\frac{dx_3}{dt} &= k_2 * x_2 + k_4 * x_1 * x_2 \\
\frac{dx_4}{dt} &= k_5 * x_1 * x_2
\end{aligned}
$$

where the state variables are the concentrations of species, Ai, i = 1, ..., 4. The initial condition is

$$x(t_0) = [1\ 0\ 0\ 0]'$$

The rate expressions are given by:

$$k_i = k_{i0} * exp(-\frac{Ei}{R * T}), i = 1, 2, 3, 4, 5$$

where the values of ki0 and Ei are given by Luus (1994). The optimal control problem is to find the residence time t_f and the temperature profile T(t) in the time interval 0 <= t <= t_f so that the production of pyrolytic bitumen, given by x2, is maximized. Therefore, the performance index is

$$J = x_2(t_f)$$

The constraints on the control variable (temperature) are:

$$698.15 <= T <= 748.15$$

Reference: [10]

75.2 Problem setup

```
toms t
toms t_f

ai = [8.86; 24.25; 23.67; 18.75; 20.70];
bi = [20300; 37400; 33800; 28200; 31000]/1.9872;

for n=[4 10 20 30 35]

    p = tomPhase('p', t, 0, t_f, n);
    setPhase(p);

    tomStates x1 x2 x3 x4
    tomControls T

    % Initial guess
    if n == 4
        x0 = {t_f == 9.3
            collocate(T == 725)};
    else
        x0 = {t_f == tfopt
            icollocate({
            x1 == x1opt; x2 == x2opt
            x3 == x3opt; x4 == x4opt
```

```
        })
        collocate(T == Topt)};
end

% Box constraints
cbox = {9.1 <= t_f <= 12
    icollocate({0 <= x1 <= 1; 0 <= x2 <= 1
    0 <= x3 <= 1; 0 <= x4 <= 1})
    698.15 <= collocate(T) <= 748.15};

% Boundary constraints
cbnd = initial({x1 == 1; x2 == 0; x3 == 0; x4 == 0});

% ODEs and path constraints

ki1 = exp(ai(1)-bi(1)./T);
ki2 = exp(ai(2)-bi(2)./T);
ki3 = exp(ai(3)-bi(3)./T);
ki4 = exp(ai(4)-bi(4)./T);
ki5 = exp(ai(5)-bi(5)./T);

ceq = collocate({
    dot(x1) == -ki1.*x1-(ki3+ki4+ki5).*x1.*x2
    dot(x2) == ki1.*x1-ki2.*x2+ki3.*x1.*x2
    dot(x3) == ki2.*x2+ki4.*x1.*x2
    dot(x4) == ki5.*x1.*x2});

% Objective
objective = -final(x2);
```

75.3 Solve the problem

```
options = struct;
options.name = 'Oil Pyrolysis';
solution = ezsolve(objective, {cbox, cbnd, ceq}, x0, options);

x1opt = subs(x1, solution);
x2opt = subs(x2, solution);
x3opt = subs(x3, solution);
x4opt = subs(x4, solution);
Topt = subs(T, solution);
tfopt = subs(final(t), solution);
```

```
Problem type appears to be: lpcon
Starting numeric solver
===== * * * =================================================================== * * *
```

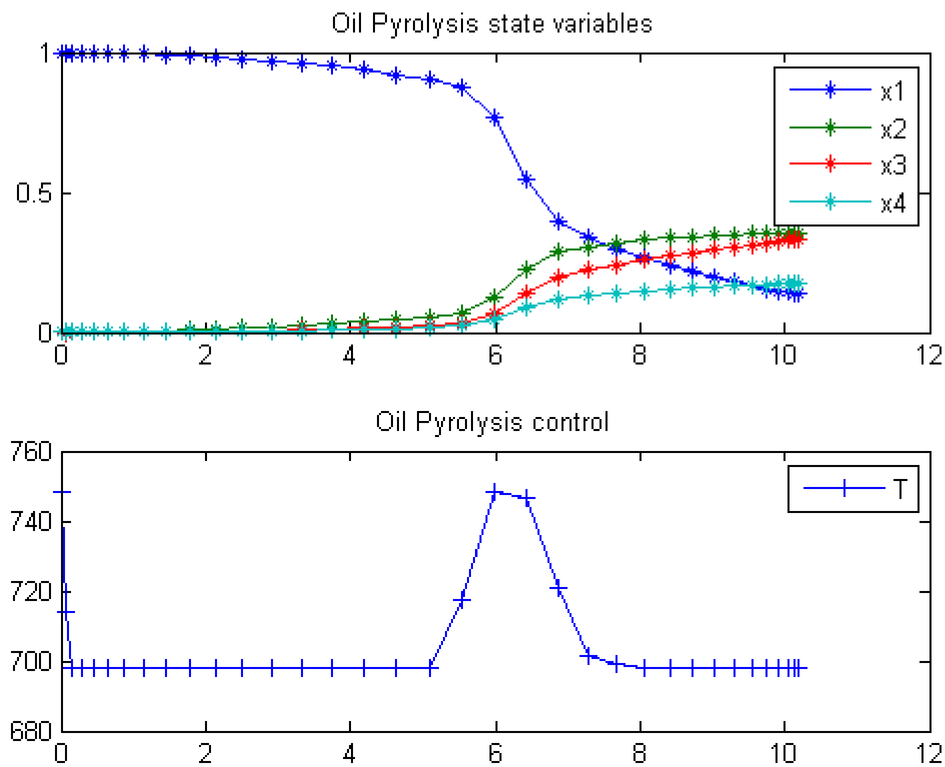
```
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Oil Pyrolysis                  f_k      -0.357327805323273570
                                       sum(|constr|)      0.000000000957551901
                              f(x_k) + sum(|constr|)     -0.357327804365721650
                                              f(x_0)      0.000000000000000000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv   93 ConJacEv   93 Iter   50 MinorIter  196
CPU time: 0.250000 sec. Elapsed time: 0.250000 sec.


Problem type appears to be: lpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Oil Pyrolysis                  f_k      -0.354368525552954730
                                       sum(|constr|)      0.000011503449213919
                              f(x_k) + sum(|constr|)     -0.354357022103740820
                                              f(x_0)     -0.357327805323273630

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv  207 ConJacEv  207 Iter  111 MinorIter  310
CPU time: 0.640625 sec. Elapsed time: 0.594000 sec.


Problem type appears to be: lpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Oil Pyrolysis                  f_k      -0.351747595241774460
                                       sum(|constr|)      0.000001734189479164
                              f(x_k) + sum(|constr|)     -0.351745861052295270
                                              f(x_0)     -0.354368525552955170

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv  145 ConJacEv  145 Iter   72 MinorIter  335
```

```
CPU time: 0.562500 sec. Elapsed time: 0.578000 sec.


Problem type appears to be: lpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Oil Pyrolysis                  f_k      -0.352833701459578820
                                      sum(|constr|)      0.000000002646417003
                              f(x_k) + sum(|constr|)     -0.352833698813161790
                                              f(x_0)     -0.351747595241774570

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv  209 ConJacEv  209 Iter  134 MinorIter  506
CPU time: 1.265625 sec. Elapsed time: 1.281000 sec.


Problem type appears to be: lpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Oil Pyrolysis                  f_k      -0.352618613056547010
                                      sum(|constr|)      0.000031914554407910
                              f(x_k) + sum(|constr|)     -0.352586698502139080
                                              f(x_0)     -0.352833701459578600

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv   64 ConJacEv   64 Iter   46 MinorIter  367
CPU time: 0.500000 sec. Elapsed time: 0.532000 sec.
```

```
end

t  = subs(collocate(t),solution);
x1 = subs(collocate(x1opt),solution);
x2 = subs(collocate(x2opt),solution);
x3 = subs(collocate(x3opt),solution);
x4 = subs(collocate(x4opt),solution);
T  = subs(collocate(Topt),solution);
```

75.4 Plot result

```
subplot(2,1,1)
plot(t,x1,'*-',t,x2,'*-',t,x3,'*-',t,x4,'*-');
legend('x1','x2','x3','x4');
title('Oil Pyrolysis state variables');

subplot(2,1,2)
plot(t,T,'+-');
legend('T');
title('Oil Pyrolysis control');
```



76 One Dimensional Rocket Ascent

User's Guide for DIRCOL

Problem 2.3 One-dimensional ascent of a rocket

76.1 Problem Formulation

Find tCut over t in [0; t_F] to minimize

$$J = tCut$$

subject to:

$$\frac{dx_1}{dt} = x2$$
$$\frac{dx_2}{dt} = a - g \ (0 < t < tCut)$$
$$\frac{dx_2}{dt} = -g \ (tCut < t < t_F)$$
$$[x_1 \ x_2] = 0$$
$$g = 1$$
$$a = 2$$
$$x_1(t_f) = 100$$

Reference: [33]

76.2 Problem setup

```
toms t
toms tCut tp2
p1 = tomPhase('p1', t, 0, tCut, 20);
p2 = tomPhase('p2', t, tCut, tp2, 20);

t_f = tCut+tp2;

x1p1 = tomState(p1,'x1p1');
```

```
x2p1 = tomState(p1,'x2p1');
x1p2 = tomState(p2,'x1p2');
x2p2 = tomState(p2,'x2p2');

% Initial guess
x0 = {tCut==10
    t_f==15
    icollocate(p1,{x1p1 == 50*tCut/10;x2p1 == 0;})
    icollocate(p2,{x1p2 == 50+50*t/100;x2p2 == 0;})};

% Box constraints
cbox = {
    1  <= tCut <= t_f-0.00001
    t_f <= 100
    0  <= icollocate(p1,x1p1)
    0  <= icollocate(p1,x2p1)
    0  <= icollocate(p2,x1p2)
    0  <= icollocate(p2,x2p2)};

% Boundary constraints
cbnd = {initial(p1,{x1p1 == 0;x2p1 == 0;})
    final(p2,x1p2 == 100)};

% ODEs and path constraints
a = 2; g = 1;
ceq = {collocate(p1,{
    dot(p1,x1p1) == x2p1
    dot(p1,x2p1) == a-g})
    collocate(p2,{
    dot(p2,x1p2) == x2p2
    dot(p2,x2p2) == -g})};

% Objective
objective = tCut;

% Link phase
link = {final(p1,x1p1) == initial(p2,x1p2)
    final(p1,x2p1) == initial(p2,x2p2)};
```

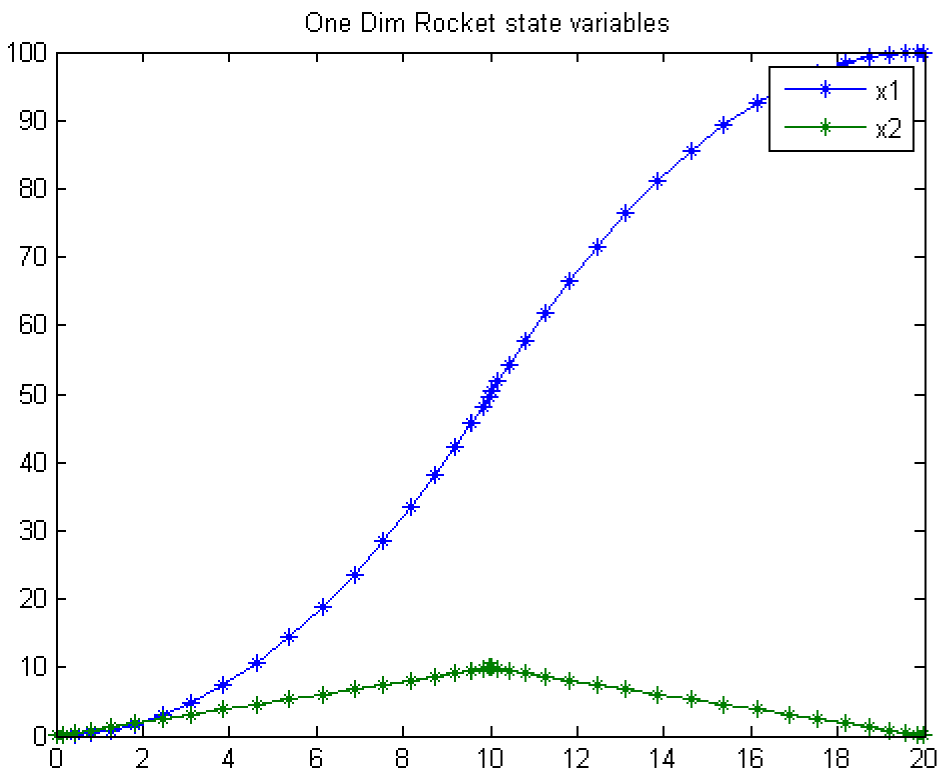
76.3 Solve the problem

```
options = struct;
options.name = 'One Dim Rocket';
constr = {cbox, cbnd, ceq, link};
solution = ezsolve(objective, constr, x0, options);
```

```
Problem type appears to be: lpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: One Dim Rocket                  f_k       9.999998166162907200
                                       sum(|constr|)      0.000733735151552596
                              f(x_k) + sum(|constr|)     10.000731901314460000
                                              f(x_0)     10.000000000000000000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv   10 ConJacEv   10 Iter    8 MinorIter   91
CPU time: 0.046875 sec. Elapsed time: 0.047000 sec.
```

76.4 Plot result

```
t = [subs(collocate(p1,t),solution);subs(collocate(p2,t),solution)];
x1 = subs(collocate(p1,x1p1),solution);
x1 = [x1;subs(collocate(p2,x1p2),solution)];
x2 = subs(collocate(p1,x2p1),solution);
x2 = [x2;subs(collocate(p2,x2p2),solution)];

figure(1)
plot(t,x1,'*-',t,x2,'*-');
legend('x1','x2');
title('One Dim Rocket state variables');
```


One Dim Rocket state variables
100
90
80
70
60
50
40
30
20
10
0
0
2
4
6
8
10
12
14
16
18
20
x1
x2

77 Parametric Sensitivity Control

Optimal Parametric Sensitivity control of a fed-batch reactor

77.1 Problem description

From the paper: J.D. Stigter, K.J. Keesman, 2004, "Optimal Parametric Sensitivity control of a fed-batch reactor", Automatica, 40, 4, pp. 1459-1464.

Programmer: Gerard Van Willigenburg (Wageningen University)

77.2 Problem setup

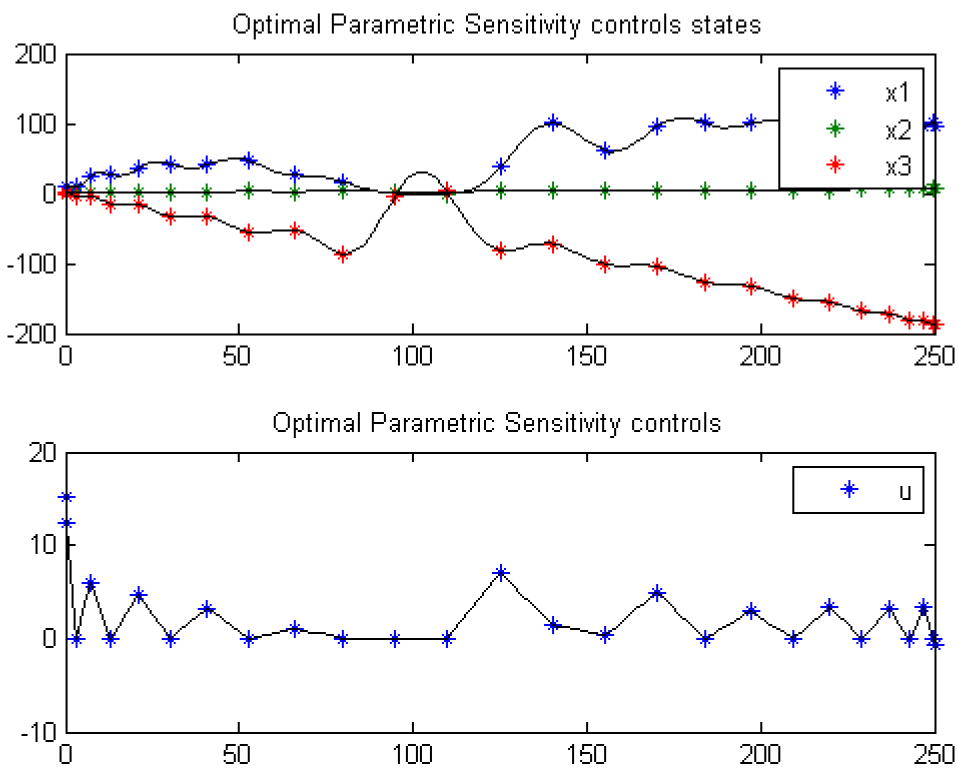
```
toms t

t_f = 250; % Fixed final time
p = tomPhase('p', t, 0, t_f, 25);
setPhase(p)

tomStates x1 x2 x3 x4
tomControls u

% Initial state amd maximum control
xi   = [0; 0; 0; 0];
umax = 20;
x    = [x1; x2; x3; x4];

% Initial guess
x0 = {icollocate(x == xi)
    collocate(u == umax)};

% Box constraints
cbox = {collocate({0 <= u <= umax; 0 <= x1 <= 100})};

% Boundary constraints
cbnd = initial(x == xi);

% Bio kinectic parameters
mu_m = 2.62e-4; Y = 0.64; K_S = 1.0;
X    = 4e3; muXY = mu_m*X/Y;

% Sensitivity parameters
q = [1 3e-2]/250;
```

```
% Odes: state and state sensitivity dynamics
Kx1 = K_S+x1; Kx12 = Kx1*Kx1;

ceq = collocate({
    dot(x1) == -muXY*x1/Kx1 + u
    dot(x2) == muXY*(x1-K_S*x2)/Kx12
    dot(x3) == -muXY*K_S*x3/Kx12-x1/Kx1
    dot(x4) == q(1)*x2*x2+q(2)*x3*x3});

% Objective
objective = -final(x4);
```

77.3 Solve the problem

```
options = struct;
options.name = 'Optimal Parametric Sensitivity';
solution = ezsolve(objective, {cbox, cbnd, ceq}, x0, options);

% Plot intermediate results
subplot(2,1,1);
ezplot([x1; x2; x3]); legend('x1','x2','x3');
title('Optimal Parametric Sensitivity controls states');

subplot(2,1,2);
ezplot(u); legend('u');
title('Optimal Parametric Sensitivity controls'); drawnow;
```

```
Problem type appears to be: lpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Optimal Parametric Sensitivity  f_k    -306.096141697006890000
                                        sum(|constr|)      0.000000004516783062
                               f(x_k) + sum(|constr|)   -306.096141692490110000
                                               f(x_0)      0.000000000000000000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv  186 ConJacEv  186 Iter   83 MinorIter 2388
CPU time: 0.843750 sec. Elapsed time: 0.844000 sec.
```


Optimal Parametric Sensitivity controls states
x1
x2
x3
200
100
0
-100
-200
0
50
100
150
200
250
Optimal Parametric Sensitivity controls
u
20
10
0
-10
0
50
100
150
200
250

78 Orbit Raising Maximum Radius

78.1 Problem description

Maximize:

$$J = r(t_f)$$

subject to the dynamic constraints

$$\frac{d_r}{dt} = u$$
$$\frac{du}{dt} = \frac{v^2}{r} - \frac{mmu}{r^2} + T * \frac{w1}{m}$$
$$\frac{dv}{dt} = -u * \frac{v}{r} + T * \frac{w2}{m}$$
$$\frac{dm}{dt} = -\frac{T}{g0 * ISP}$$

the boundary conditions

$$r(t_0) = 1$$
$$u(t_0) = 1$$
$$u(t_f) = 0$$
$$v(t_0) = (\frac{mmu}{r(t_0)})^{0.5}$$
$$v(t_f) = (\frac{mmu}{r(t_f)})^{0.5}$$
$$m(t_0) = 1$$

and the path constraint

$$w_1^2 + w_2^2 = 1$$

The control pitch angel is not being used in this formulation. Instead two control variables (w1,w2) are used to for the thrust direction. A path constraint ensures that (w1,w2) is a unit vector.

Reference: [5]

78.2 Problem setup

```
t_F    = 3.32;
mmu    = 1;
m_0   = 1;   r_0     = 1;            u_0         = 0;
u_f   = 0;   v_0     = sqrt(mmu/r_0); rmin        = 0.9;
rmax  = 5;   umin    = -5;           umax        = 5;
vmin  = -5;  vmax    = 5;            mmax        = m_0;
mmin  = 0.1;

T  = 0.1405;
Ve = 1.8758;

toms t
p1 = tomPhase('p1', t, 0, t_F, 40);
setPhase(p1);

tomStates r u v m
tomControls w1 w2

% Initial guess
x0 = {icollocate({
    r == r_0
    u == u_0 + (u_f-u_0)*t/t_F
    v == v_0
    m == m_0})
    collocate({w1 == 0; w2 == 1})};

% Boundary constraints
cbnd = {initial({
    r == r_0
    u == u_0
    v == v_0
    m == m_0
    })
    final({u == u_f
    v - sqrt(mmu/r) == 0})};

% Box constraints
cbox = {
    rmin <= icollocate(r) <= rmax
```

```
    umin <= icollocate(u) <= umax
    vmin <= icollocate(v) <= vmax
    mmin <= icollocate(m) <= mmax
    -1 <= collocate(w1) <= 1
    -1 <= collocate(w2) <= 1};

% ODEs and path constraints
ceq = collocate({
    dot(r) == u
    dot(u) == v^2/r-mmu/r^2+T*w1/m
    dot(v) == -u*v/r+T*w2/m
    dot(m) == -T/Ve
    w1^2+w2^2 == 1});

% Objective
objective = -final(r);
```

78.3 Solve the problem

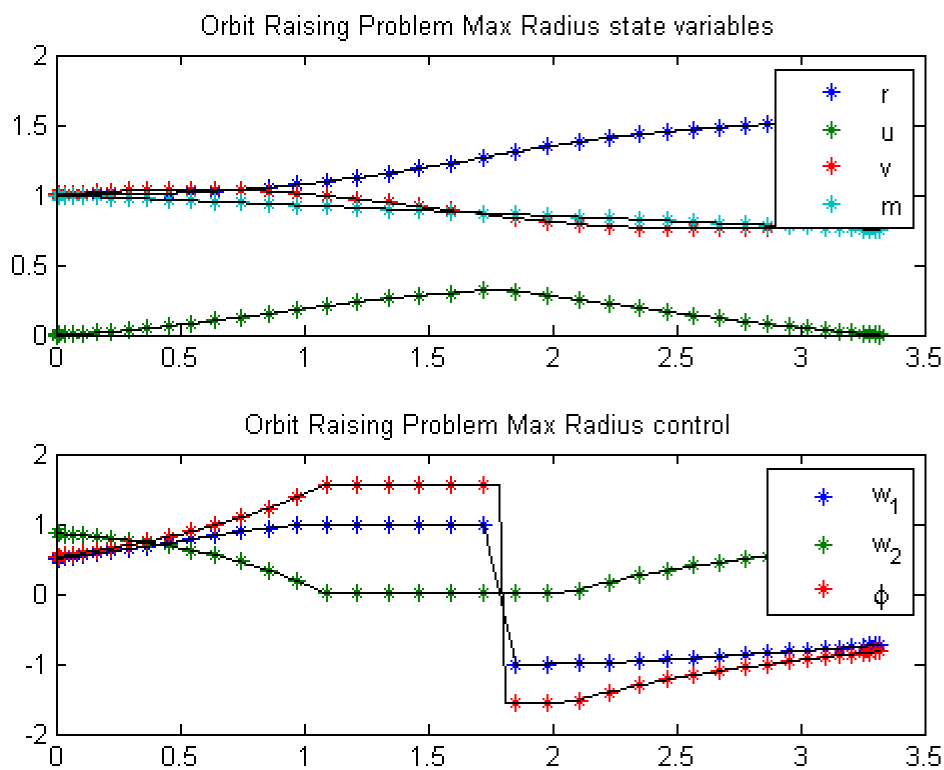
```
options = struct;
options.name = 'Orbit Raising Problem Max Radius';
options.solver = 'snopt';
solution = ezsolve(objective, {cbox, cbnd, ceq}, x0, options);
```

```
Problem type appears to be: lpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Orbit Raising Problem Max Radius  f_k      -1.518744202740336600
                                          sum(|constr|)      0.000017265841651215
                                  f(x_k) + sum(|constr|)     -1.518726936898685300
                                                 f(x_0)     -0.999999999999991120

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv   59 ConJacEv   59 Iter   36 MinorIter  346
CPU time: 0.656250 sec. Elapsed time: 0.656000 sec.
```

78.4 Plot result

```
subplot(2,1,1)
ezplot([r u v m]);
```

```
legend('r','u','v','m');
title('Orbit Raising Problem Max Radius state variables');

subplot(2,1,2)
ezplot([w1 w2 atan2(w1,w2)])
legend('w_1', 'w_2', '\phi');
title('Orbit Raising Problem Max Radius control');
```



79 Orbit Raising Minimum Time

79.1 Problem description

Minimize:

$$J = t_f$$

subject to the dynamic constraints

$$\begin{aligned}
\frac{d_r}{dt} &= u \\
\frac{du}{dt} &= \frac{v^2}{r} - \frac{mmu}{r^2} + T * \frac{w1}{m} \\
\frac{dv}{dt} &= -u * \frac{v}{r} + T * \frac{w2}{m} \\
\frac{dm}{dt} &= -\frac{T}{g0 * ISP}
\end{aligned}$$

the boundary conditions

$$\begin{aligned}
r(t_0) &= 1 \\
u(t_0) &= 1 \\
u(t_f) &= 0 \\
v(t_0) &= (\frac{mmu}{r(t_0)})^{0.5} \\
v(t_f) &= (\frac{mmu}{r(t_f)})^{0.5} \\
m(t_0) &= 1
\end{aligned}$$

where w1 = sin(phi) and w2 = cos(phi)

At t_f, r and m are free.

Reference: [5]

79.2 Problem setup

```
mmu     = 1;
t_f    = 3.32;  m_0    = 1;            r_0    = 1;   u_0    = 0;
u_f    = 0;     v_0    = sqrt(mmu/r_0); rmin   = 0.9; rmax   = 5;
umin   = -5;    umax   = 5;            vmin   = -5;  vmax   = 5;
mmax   = m_0;   mmin   = 0.1;          tf_min = 0.5; tf_max = 10;
r_f     = 1.5;

T  = 0.1405;
Ve = 1.8758;

toms t t_f
p1 = tomPhase('p1', t, 0, t_f, 50);
setPhase(p1);

tomStates r u v m

% The problem becomes less nonlinear if w1 and w2 are control variables
% (with the constraints w1^2+w2^2==1) than if phi is the control varialbe
% (with w1 and w2 being nonlinear functions of phi).
tomControls w1 w2
phi = atan2(w1,w2);

% Initial guess
x0 = {t_f == 3.32
    icollocate({
    r == r_0+(r_f-r_0)*t/t_f
    u == 0.1
    v == v_0
    m == m_0-(T/Ve)*t})
    collocate({
    w1 == -0.7*sign(t-t_f/2)
    w2 == 0.4
    })};

% Boundary constraints
cbnd = {initial({
    r == r_0
    u == u_0
    v == v_0
    m == m_0
    })
    final({
    r == r_f
    u == u_f
    v == sqrt(mmu/r)})};
```

```
% Box constraints
cbox = {0.5 <= t_f <= 10
    rmin <= icollocate(r) <= rmax
    umin <= icollocate(u) <= umax
    vmin <= icollocate(v) <= vmax
    };

% ODEs and path constraints
ceq = collocate({
    dot(r) == u
    dot(u) == v^2/r-mmu/r^2+T*w1/m
    dot(v) == -u*v/r+T*w2/m
    dot(m) == -T/Ve
    w1^2+w2^2 == 1
    });

% Objective
objective = t_f;
```

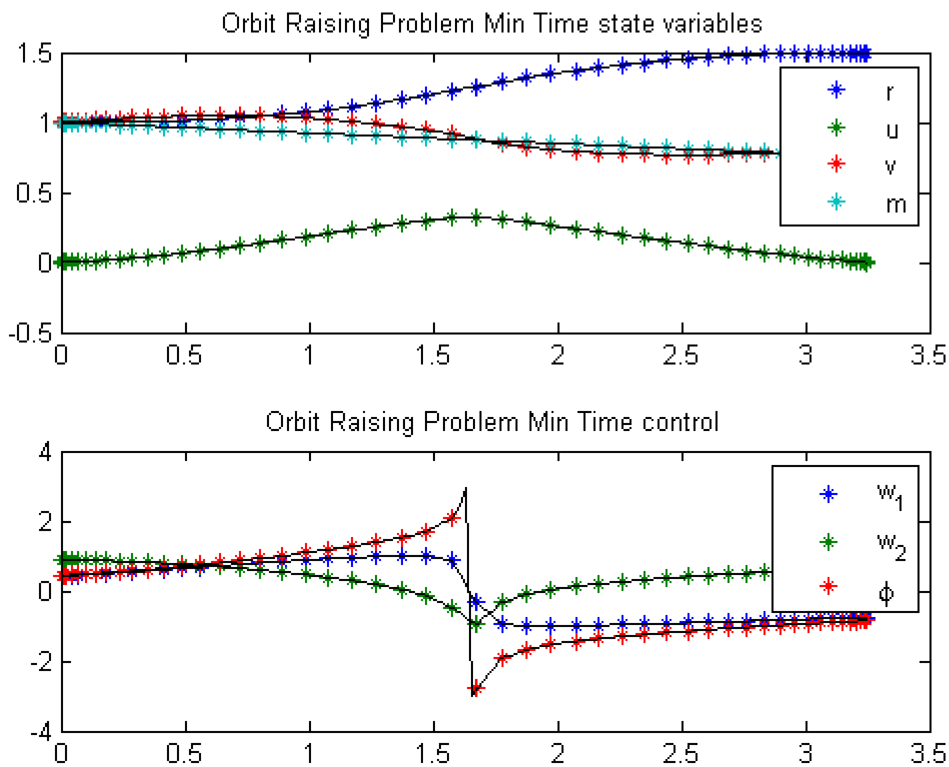
79.3 Solve the problem

```
options = struct;
options.name = 'Orbit Raising Problem Min Time';
options.scale = 'manual'; % Auto-scaling is not really needed as all variables are already reasonably scale
solution = ezsolve(objective, {cbox, cbnd, ceq}, x0, options);
```

```
Problem type appears to be: lpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Orbit Raising Problem Min Time  f_k       3.248079535630944200
                                       sum(|constr|)      0.000032253415551923
                              f(x_k) + sum(|constr|)      3.248111789046496300
                                              f(x_0)      3.319999999999999800

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv   68 ConJacEv   68 Iter   35 MinorIter  283
CPU time: 2.140625 sec. Elapsed time: 1.187000 sec.
```

79.4 Plot result

```
subplot(2,1,1)
ezplot([r u v m]);
legend('r','u','v','m');
title('Orbit Raising Problem Min Time state variables');

subplot(2,1,2)
ezplot([w1 w2 phi])
legend('w_1', 'w_2', '\phi');
title('Orbit Raising Problem Min Time control');
```



80 Parallel Reactions in Tubular Reactor

Problem 4: DYNOPT User's Guide version 4.1.0

Batch reactor with reactions: A -> B and A -> C.

M. Cizniar, M. Fikar, M. A. Latifi, MATLAB Dynamic Optimisation Code DYNOPT. User's Guide, Technical Report, KIRP FCHPT STU Bratislava, Slovak Republic, 2006.

80.1 Problem description

Find T over t in [0; 1] to maximize

$$J = x_2(t_f)$$

subject to:

$$\frac{dx_1}{dt} = -(u + 0.5 * u^2) * x_1$$
$$\frac{dx_2}{dt} = u * x_1$$

where

$$x(0) = [1\ 0]$$
$$0 <= u <= 5$$

Reference: [13]

80.2 Problem setup

```
toms t
p = tomPhase('p', t, 0, 1, 30);
setPhase(p);

tomStates x1 x2
```

```
tomControls u

% Initial guess
x0 = {icollocate({x1 == 1; x2 == 0})
    collocate(u == 5*t)};

% Box constraints
cbox = {0 <= collocate(u) <= 5};

% Boundary constraints
cbnd = initial({x1 == 1; x2 == 0});

% ODEs and path constraints
ceq = collocate({
    dot(x1) == -(u+0.5*u.^2).*x1
    dot(x2) == u.*x1});

% Objective
objective = -final(x2);
```

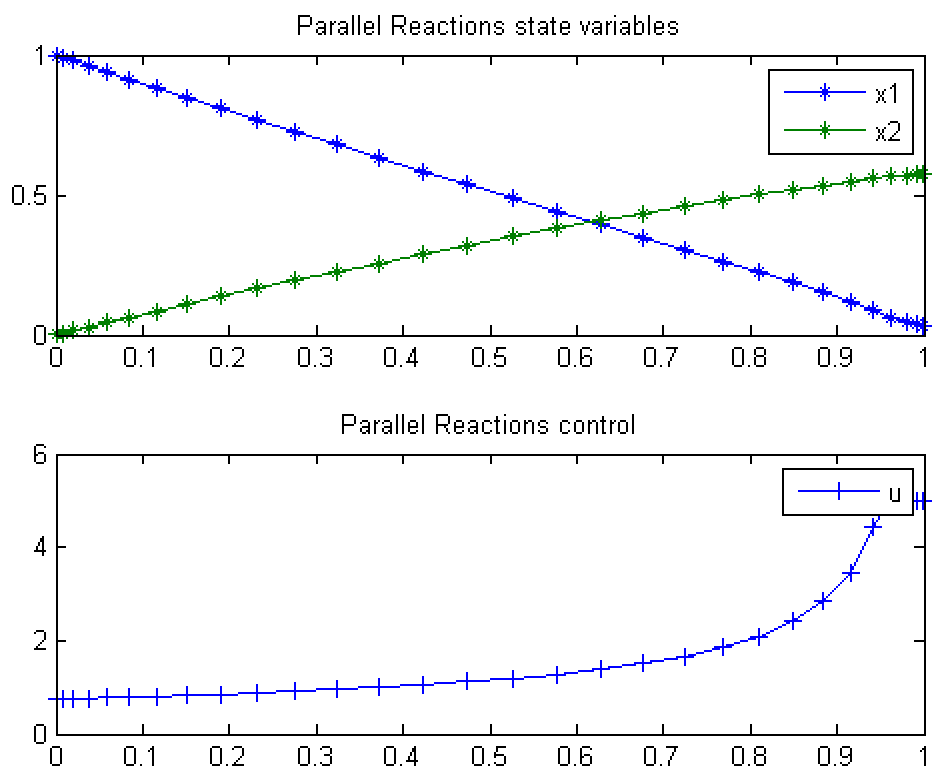
80.3 Solve the problem

```
options = struct;
options.name = 'Parallel Reactions';
solution = ezsolve(objective, {cbox, cbnd, ceq}, x0, options);
t  = subs(collocate(t),solution);
x1 = subs(collocate(x1),solution);
x2 = subs(collocate(x2),solution);
u  = subs(collocate(u),solution);
```

```
Problem type appears to be: lpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Parallel Reactions             f_k      -0.573540787113988930
                                       sum(|constr|)      0.000000228705850230
                              f(x_k) + sum(|constr|)     -0.573540558408138670
                                              f(x_0)      0.000000000000000000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv   35 ConJacEv   35 Iter   34 MinorIter  118
CPU time: 0.125000 sec. Elapsed time: 0.125000 sec.
```

80.4 Plot result

```
subplot(2,1,1)
plot(t,x1,'*-',t,x2,'*-');
legend('x1','x2');
title('Parallel Reactions state variables');

subplot(2,1,2)
plot(t,u,'+-');
legend('u');
title('Parallel Reactions control');
```



81 Parameter Estimation Problem

Example 5: DYNOPT User's Guide version 4.1.0

M. Cizniar, M. Fikar, M. A. Latifi, MATLAB Dynamic Optimisation Code DYNOPT. User's Guide, Technical Report, KIRP FCHPT STU Bratislava, Slovak Republic, 2006.

81.1 Problem description

Find p1 and p2 over t in [0; 6] to minimize

$$J = \sum_{i=1,2,3,5} (x_1(t_i) - x_1^m(t_i))^2$$

subject to:

$$\frac{dx_1}{dt} = x_2$$
$$\frac{dx_2}{dt} = 1 - 2 * x_2 - x_1$$

where

$$x_0 = [p_1 \ p_2]$$
$$t_i = [1 \ 2 \ 3 \ 5]$$
$$x_1^m(t_i) = [0.264 \ 0.594 \ 0.801 \ 0.959]$$
$$-1.5 <= p_{1:2} <= 1.5$$

Reference: [13]

81.2 Problem setup

```
toms t p1 p2
x1meas = [0.264;0.594;0.801;0.959];
tmeas  = [1;2;3;5];
```

```
% Box constraints
cbox = {-1.5 <= p1 <= 1.5
    -1.5 <= p2 <= 1.5};
```

81.3 Solve the problem, using a successively larger number collocation points

```
for n=[10 40]

    p = tomPhase('p', t, 0, 6, n);
    setPhase(p);
    tomStates x1 x2

    % Initial guess
    if n == 10
        x0 = {p1 == 0; p2 == 0};
    else
        x0 = {p1 == p1opt; p2 == p2opt
            icollocate({x1 == x1opt; x2 == x2opt})};
    end

    % Boundary constraints
    cbnd = initial({x1 == p1; x2 == p2});

    % ODEs and path constraints
    x1err = sum((atPoints(tmeas,x1) - x1meas).^2);
    ceq = collocate({dot(x1) == x2; dot(x2) == 1-2*x2-x1});

    % Objective
    objective = x1err;
```

81.4 Solve the problem

```
    options = struct;
    options.name   = 'Parameter Estimation';
    options.solver = 'snopt';
    solution = ezsolve(objective, {cbox, cbnd, ceq}, x0, options);

    % Optimal x, p for starting point
    x1opt = subs(x1, solution);
    x2opt = subs(x2, solution);
    p1opt = subs(p1, solution);
    p2opt = subs(p2, solution);
```

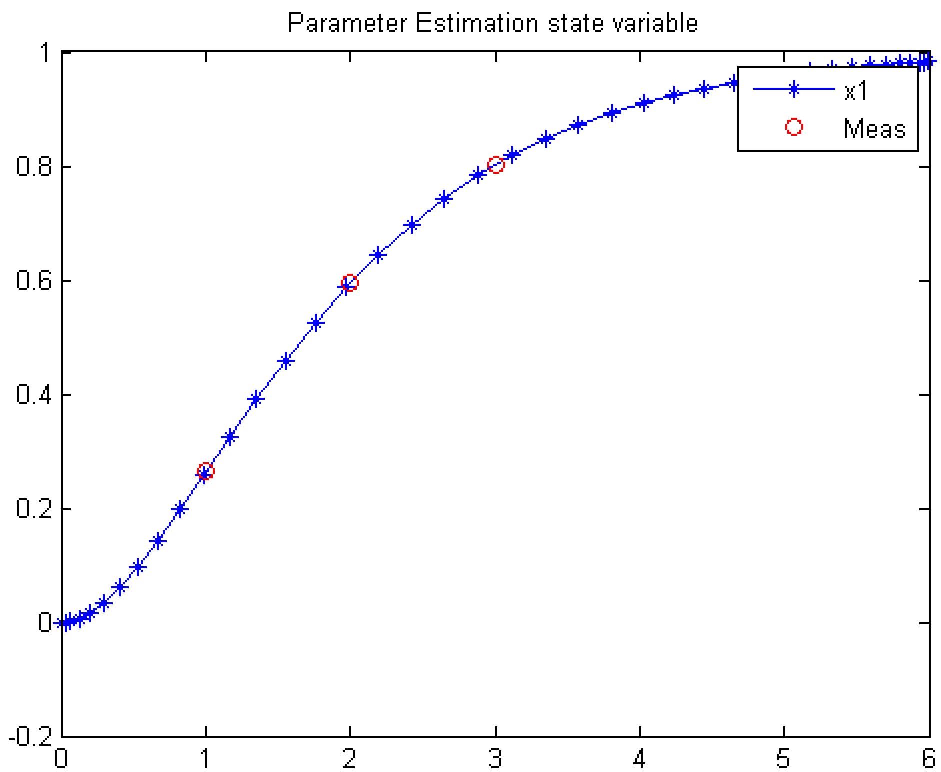
```
Problem type appears to be: qp
Starting numeric solver
===== * * * =================================================================== * * *
```

```
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem:  1: Parameter Estimation               f_k       0.000000352979271145
                                       sum(|constr|)      0.000000000000054134
                              f(x_k) + sum(|constr|)      0.000000352979325279
                                              f(x_0)      0.000000000000000000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 Iter    5 MinorIter   17
CPU time: 0.015625 sec. Elapsed time: 0.016000 sec.


Problem type appears to be: qp
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem:  1: Parameter Estimation               f_k       0.000000355693079657
                                       sum(|constr|)      0.000000000000071929
                              f(x_k) + sum(|constr|)      0.000000355693151586
                                              f(x_0)     -1.983813647020728800

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 MinorIter   40
CPU time: 0.015625 sec. Elapsed time: 0.016000 sec.
```

```
end

t  = subs(collocate(t),solution);
x1 = collocate(x1opt);
```

81.5 Plot result

```
figure(1)
plot(t,x1,'*-',tmeas,x1meas,'ro');
legend('x1','Meas');
title('Parameter Estimation state variable');
```



Parameter Estimation state variable
x1
Meas
1
0.8
0.6
0.4
0.2
0
-0.2
0
1
2
3
4
5
6

82 Park-Ramirez bioreactor

Dynamic optimization of chemical and biochemical processes using restricted second-order information 2001, Eva Balsa-Canto, Julio R. Banga, Antonio A. Alonso Vassilios S. Vassiliadis

Case Study I: Park-Ramirez bioreactor

82.1 Problem description

This case study deals with the optimal production of secreted protein in a fed-batch reactor. It was originally formulated by Park and Ramirez (Park & Ramirez, 1988) and it has also been considered by other researchers (Vassiliadis, 1993; Yang & Wu, 1994; Banga, Irizarry & Seider, 1995; Luus, 1995; Tholudur & Ramirez, 1997). The objective is to maximize the secreted heterologous protein by a yeast strain in a fed-batch culture. The dynamic model accounts for host-cell growth, gene expression, and the secretion of expressed polypeptides. The mathematical statement is as follows:

Find u(t) over t in [t0,t_f] to maximize:

$$J = x_1(t_f) * x_5(t_f)$$

subject to:

$$\begin{aligned}
\frac{dx_1}{dt} &= g_1 * (x_2 - x_1) - \frac{u}{x_5} * x_1, \\
\frac{dx_2}{dt} &= g_2 * x_3 - \frac{u}{x_5} * x_2, \\
\frac{dx_3}{dt} &= g_3 * x_3 - \frac{u}{x_5} * x_3, \\
\frac{dx_4}{dt} &= -7.3 * g_3 * x_3 + \frac{u}{x_5} * (20 - x_4), \\
\frac{dx_5}{dt} &= u,
\end{aligned}$$

with:

$$\begin{aligned}
g_1 &= 4.75 * \frac{g_3}{(0.12 + g_3)}, \\
g_2 &= \frac{x_4}{(0.1 + x4)} * exp(-5 * x_4),
\end{aligned}$$

$$g_3 = 21.87 * \frac{x_4}{(x_4 + 0.4) * (x_4 + 62.5)},$$

where x1 and x2 are, respectively, the concentration of the secreted protein and the total protein (l-1), x3 is the culture cell density (g l-1), x4 is the substrate concentration (g l-1), x5 is the holdup volume (l), u is the nutrient (glucose) feed rate (l h-1), and J is the mass of protein produced (in arbitrary units). The initial conditions are:

$$x(t_0) = [0\ 0\ 1\ 5\ 1]'$$

For final time t_f = 15 h, and the following constraints on the control variable:

$$0 <= u <= 2$$

Reference: [2]

82.2 Problem setup

```
toms t
```

82.3 Solve the problem, using a successively larger number collocation points

```
for n=[20 40 80 120]

    p = tomPhase('p', t, 0, 15, n);
    setPhase(p);

    tomStates z1 z2 z3 z4 z5
    tomControls u

    if n>20
        % Interpolate an initial guess for the n collocation points
        x0 = {icollocate({z1 == z1opt; z2 == z2opt
            z3 == z3opt; z4 == z4opt; z5 == z5opt})
            collocate(u == uopt)};
    else
        x0 = {};
    end

    % Box constraints
    cbox = {icollocate({z1 <= 3
```

```
    z2 <= 3; 0   <= z3 <= 4
    0 <= z4 <= 10; 0.5 <= z5 <= 25})
    0 <= collocate(u) <= 2.5};

% Boundary constraints
cbnd = initial({z1 == 0; z2 == 0; z3 == 1
    z4 == 5; z5 == 1});

% Various constants and expressions
g3 = 21.87*z4./(z4+.4)./(z4+62.5);
g1 = 4.75*g3./(0.12+g3);
g2 = z4./(0.1+z4).*exp(-5*z4);

% ODEs and path constraints
ceq = collocate({
    dot(z1) == g1.*(z2-z1)-u./z5.*z1
    dot(z2) == g2.*z3-u./z5.*z2
    dot(z3) == g3.*z3-u./z5.*z3
    dot(z4) == -7.3*g3.*z3+u./z5.*(20-z4)
    dot(z5) == u});

% Secreted protein must be less than total protein
% proteinlimit = {z1 <= z2};

% Objective
if n == 120
    objective = -final(z1)*final(z5)+var(diff(collocate(u)));
else
    objective = -final(z1)*final(z5);
end
```

82.4 Solve the problem

```
options = struct;
options.name = 'Park Bio Reactor';
solution = ezsolve(objective, {cbox, cbnd, ceq}, x0, options);

% Optimal z and u, to use as starting guess in the
% next iteration
z1opt = subs(z1, solution);
z2opt = subs(z2, solution);
z3opt = subs(z3, solution);
z4opt = subs(z4, solution);
z5opt = subs(z5, solution);
uopt  = subs(u, solution);
```

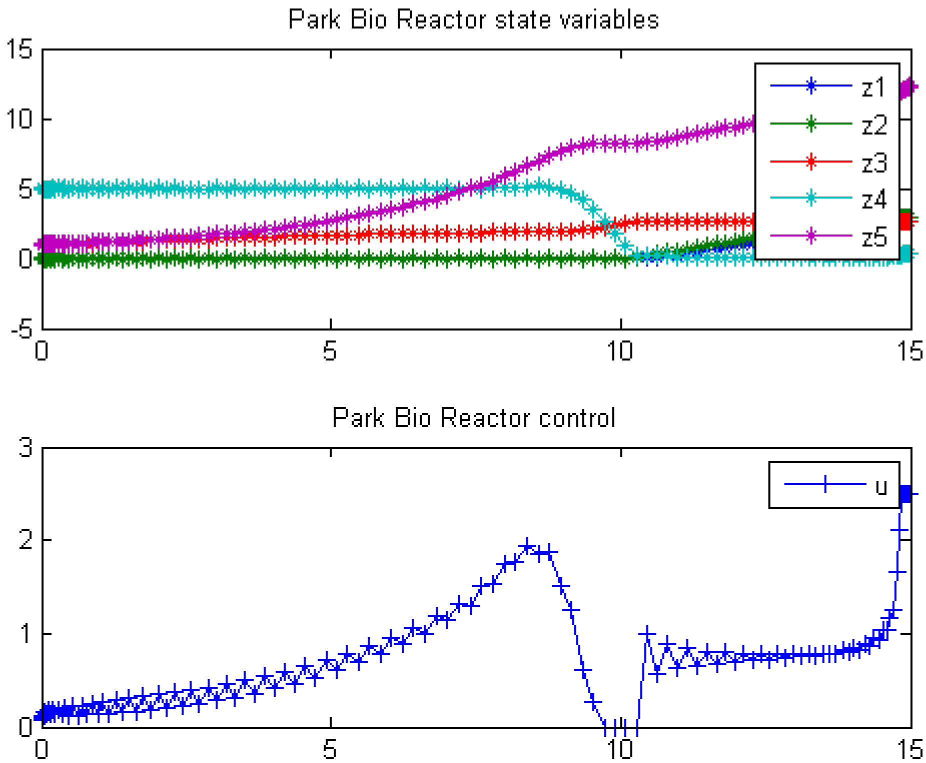
```
Problem type appears to be: qpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Park Bio Reactor               f_k     -31.891604942090215000
                                       sum(|constr|)      0.000000000222834078
                              f(x_k) + sum(|constr|)    -31.891604941867381000
                                              f(x_0)      0.000000000000000000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv  102 ConJacEv  102 Iter   61 MinorIter  915
CPU time: 0.546875 sec. Elapsed time: 0.640000 sec.


Problem type appears to be: qpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Park Bio Reactor               f_k     -29.523702508277239000
                                       sum(|constr|)      0.000000000532314958
                              f(x_k) + sum(|constr|)    -29.523702507744925000
                                              f(x_0)    -31.891604942090005000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv   75 ConJacEv   75 Iter   58 MinorIter  585
CPU time: 1.203125 sec. Elapsed time: 1.235000 sec.


Problem type appears to be: qpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Park Bio Reactor               f_k     -31.866762161406502000
                                       sum(|constr|)      0.000000000725606662
                              f(x_k) + sum(|constr|)    -31.866762160680896000
                                              f(x_0)    -29.523702508277385000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
```

```
Optimality conditions satisfied

FuncEv    1 ConstrEv  137 ConJacEv  137 Iter  120 MinorIter  778
CPU time: 10.796875 sec. Elapsed time: 11.063000 sec.


Problem type appears to be: qpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Park Bio Reactor               f_k     -32.616144466384597000
                                       sum(|constr|)      0.000000000325372392
                              f(x_k) + sum(|constr|)    -32.616144466059225000
                                              f(x_0)    -31.477810608046273000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv  134 ConJacEv  134 Iter  116 MinorIter  926
CPU time: 39.937500 sec. Elapsed time: 41.422000 sec.
```

```
end

t  = subs(collocate(t),solution);
z1 = collocate(z1opt);
z2 = collocate(z2opt);
z3 = collocate(z3opt);
z4 = collocate(z4opt);
z5 = collocate(z5opt);
u  = collocate(uopt);
```

82.5 Plot result

```
subplot(2,1,1)
plot(t,z1,'*-',t,z2,'*-',t,z3,'*-',t,z4,'*-',t,z5,'*-');
legend('z1','z2','z3','z4','z5');
title('Park Bio Reactor state variables');

subplot(2,1,2)
plot(t,u,'+-');
legend('u');
title('Park Bio Reactor control');
```



Park Bio Reactor state variables
15
10
5
0
-5
0
5
10
15
z1
z2
z3
z4
z5
Park Bio Reactor control
3
2
1
0
0
5
10
15
u

83 Path Tracking Robot

User's Guide for DIRCOL

2.7 Optimal path tracking for a simple robot. A robot with two rotational joints and simplified equations of motion has to move along a prescribed path with constant velocity.

83.1 Problem Formulation

Find u over t in [0; 2] to minimize

$$J = \int_0^2 (\sum_{i=1}^{2} (w_i * (q_i(t) - q_{i,ref})^2) + \sum_{i=1}^{2} (w_{2+i} * (\frac{dq}{dt}_i(t) - \frac{dq}{dt}_{i,ref})^2) \mathrm{d}t$$

subject to:

$$\frac{d^2 q_1}{dt^2} = u_1$$
$$\frac{d^2 q_2}{dt^2} = u_2$$

A transformation gives:

$$\frac{dx_1}{dt} = x_3$$
$$\frac{dx_2}{dt} = x_4$$
$$\frac{dx_3}{dt} = u_1$$
$$\frac{dx_4}{dt} = u_2$$

$$x_{1:4}(0) = [0\ 0\ 0.5\ 0]$$
$$x_{1:4}(2) = [0.5\ 0.5\ 0\ 0.5]$$
$$w_{1:4} = [100\ 100\ 500\ 500]$$

$$x_1 1, ref = \frac{t}{2}\ (0 < t < 1),\ \frac{1}{2}\ (1 < t < 2)$$

$$x_2 1, ref = 0\ (0 < t < 1),\ \frac{t-1}{2}\ (1 < t < 2)$$

$$x_3 1, ref = \frac{1}{2}\ (0 < t < 1),\ 0\ (1 < t < 2)$$

$$x_4 1, ref = 0\ (0 < t < 1),\ \frac{1}{2}\ (1 < t < 2)$$

$$|u| < 10$$

Reference: [33]

83.2 Problem setup

```
toms t
p = tomPhase('p', t, 0, 2, 100, [], 'fem1s'); % Use splines with FEM constraints
%p = tomPhase('p', t, 0, 2, 100, [], 'fem1');  % Use linear finite elements
%p = tomPhase('p', t, 0, 2, 100); % Use Gauss point collocation
setPhase(p);

tomStates x1 x2 x3 x4
tomControls u1 u2

% Box constraints
cbox = {
    -10 <= collocate(u1) <= 10
    -10 <= collocate(u2) <= 10};

% Boundary constraints
cbnd = {initial({x1 == 0; x2 == 0
    x3 == 0.5; x4 == 0})
    final({x1 == 0.5; x2 == 0.5
    x3 == 0; x4 == 0.5})};

% ODEs and path constraints
w1 = 100; w2 = 100;
w3 = 500; w4 = 500;

err1 = w1*(x1-t/2.*(t<1)-1/2*(t>=1)).^2;
err2 = w2*(x2-(t-1)/2.*(t>=1)).^2;
err3 = w3*(x3-1/2*(t<1)).^2;
```

```
err4 = w4*(x4-1/2*(t>=1)).^2;

toterr = integrate(err1+err2+err3+err4);

ceq = collocate({
    dot(x1) == x3
    dot(x2) == x4
    dot(x3) == u1
    dot(x4) == u2});

% Objective
objective = toterr;
```

83.3 Solve the problem

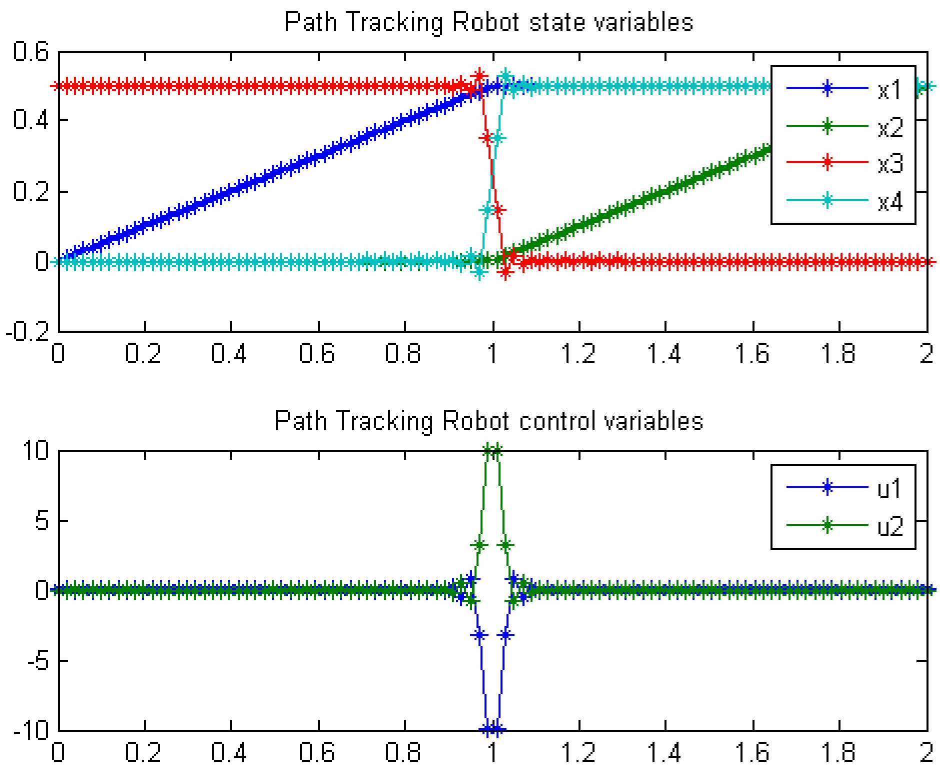
```
options = struct;
options.name = 'Path Tracking Robot';
solution = ezsolve(objective, {cbox, cbnd, ceq}, [], options);
t  = subs(icollocate(t),solution);
x1 = subs(icollocate(x1),solution);
x2 = subs(icollocate(x2),solution);
x3 = subs(icollocate(x3),solution);
x4 = subs(icollocate(x4),solution);
u1 = subs(icollocate(u1),solution);
u2 = subs(icollocate(u2),solution);
```

```
Problem type appears to be: qp
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem:  1: Path Tracking Robot                f_k       1.031157513483037700
                                       sum(|constr|)      0.000000051263199492
                              f(x_k) + sum(|constr|)      1.031157564746237200
                                              f(x_0)      0.000000000000000000

Solver: CPLEX.  EXIT=0.  INFORM=1.
CPLEX Barrier QP solver
Optimal solution found

FuncEv   10 GradEv   10 ConstrEv   10 Iter   10
CPU time: 0.343750 sec. Elapsed time: 0.235000 sec.
```

83.4 Plot result

```
subplot(2,1,1);
```

```
plot(t,x1,'*-',t,x2,'*-',t,x3,'*-',t,x4,'*-');
legend('x1','x2','x3','x4');
title('Path Tracking Robot state variables');

subplot(2,1,2);
plot(t,u1,'*-',t,u2,'*-');
legend('u1','u2');
title('Path Tracking Robot control variables');
```



84 Path Tracking Robot (Two-Phase)

User's Guide for DIRCOL

2.7 Optimal path tracking for a simple robot. A robot with two rotational joints and simplified equations of motion has to move along a prescribed path with constant velocity.

84.1 Problem Formulation

Find u over t in [0; 2] to minimize

$$J = \int_0^2 (\sum_{i=1}^{2} (w_i * (q_i(t) - q_{i,ref})^2) + \sum_{i=1}^{2} (w_{2+i} * (\frac{dq}{dt}_i(t) - \frac{dq}{dt}_{i,ref})^2) \mathrm{d}t$$

subject to:

$$\frac{d^2 q_1}{dt^2} = u_1$$
$$\frac{d^2 q_2}{dt^2} = u_2$$

A transformation gives:

$$\frac{dx_1}{dt} = x_3$$
$$\frac{dx_2}{dt} = x_4$$
$$\frac{dx_3}{dt} = u_1$$
$$\frac{dx_4}{dt} = u_2$$

$$x_{1:4}(0) = [0 \ 0 \ 0.5 \ 0]$$
$$x_{1:4}(2) = [0.5 \ 0.5 \ 0 \ 0.5]$$
$$w_{1:4} = [100 \ 100 \ 500 \ 500]$$

$$x_1 1, ref = \frac{t}{2}\ (0 < t < 1),\ \frac{1}{2}\ (1 < t < 2)$$

$$x_2 1, ref = 0\ (0 < t < 1),\ \frac{t-1}{2}\ (1 < t < 2)$$

$$x_3 1, ref = \frac{1}{2}\ (0 < t < 1),\ 0\ (1 < t < 2)$$

$$x_4 1, ref = 0\ (0 < t < 1),\ \frac{1}{2}\ (1 < t < 2)$$

$$|u| < 10$$

Reference: [33]

84.2 Problem setup

```
t_F = 2;
toms t1
p1 = tomPhase('p1', t1, 0, t_F/2, 25);
toms t2
p2 = tomPhase('p2', t2, t_F/2, t_F/2, 25);
setPhase(p1);

tomStates x1p1 x2p1 x3p1 x4p1
tomControls u1p1 u2p1

% Box constraints
cbox1 = {-10 <= collocate(u1p1) <= 10
    -10 <= collocate(u2p1) <= 10};

% Boundary constraints
w1 = 100; w2 = 100;
w3 = 500; w4 = 500;
err1p1 = w1*(x1p1-t1/2).^2;
err2p1 = w2*(x2p1).^2;
err3p1 = w3*(x3p1-1/2).^2;
err4p1 = w4*(x4p1).^2;

cbnd1 = initial({x1p1 == 0; x2p1 == 0
    x3p1 == 0.5; x4p1 == 0});

% ODEs and path constraints
ceq1 = collocate({dot(x1p1) == x3p1
```

```
    dot(x2p1) == x4p1; dot(x3p1) == u1p1
    dot(x4p1) == u2p1});

% Objective
objective1 = integrate(err1p1+err2p1+err3p1+err4p1);

% Phase 2
setPhase(p2);
tomStates x1p2 x2p2 x3p2 x4p2
tomControls u1p2 u2p2

% Box constraints
cbox2 = {-10 <= collocate(u1p2) <= 10
    -10 <= collocate(u2p2) <= 10};

% Boundary constraints
err1p2 = w1*(x1p2-1/2).^2;
err2p2 = w2*(x2p2-(t2-1)/2).^2;
err3p2 = w3*(x3p2).^2;
err4p2 = w4*(x4p2-1/2).^2;

cbnd2 = final({x1p2 == 0.5
    x2p2 == 0.5
    x3p2 == 0
    x4p2 == 0.5});

% ODEs and path constraints
ceq2 = collocate({
    dot(x1p2) == x3p2
    dot(x2p2) == x4p2
    dot(x3p2) == u1p2
    dot(x4p2) == u2p2});

% Objective
objective2 = integrate(err1p2+err2p2+err3p2+err4p2);

% Objective
objective = objective1 + objective2;

% Link phase
link = {final(p1,x1p1) == initial(p2,x1p2)
    final(p1,x2p1) == initial(p2,x2p2)
    final(p1,x3p1) == initial(p2,x3p2)
    final(p1,x4p1) == initial(p2,x4p2)};
```

84.3 Solve the problem

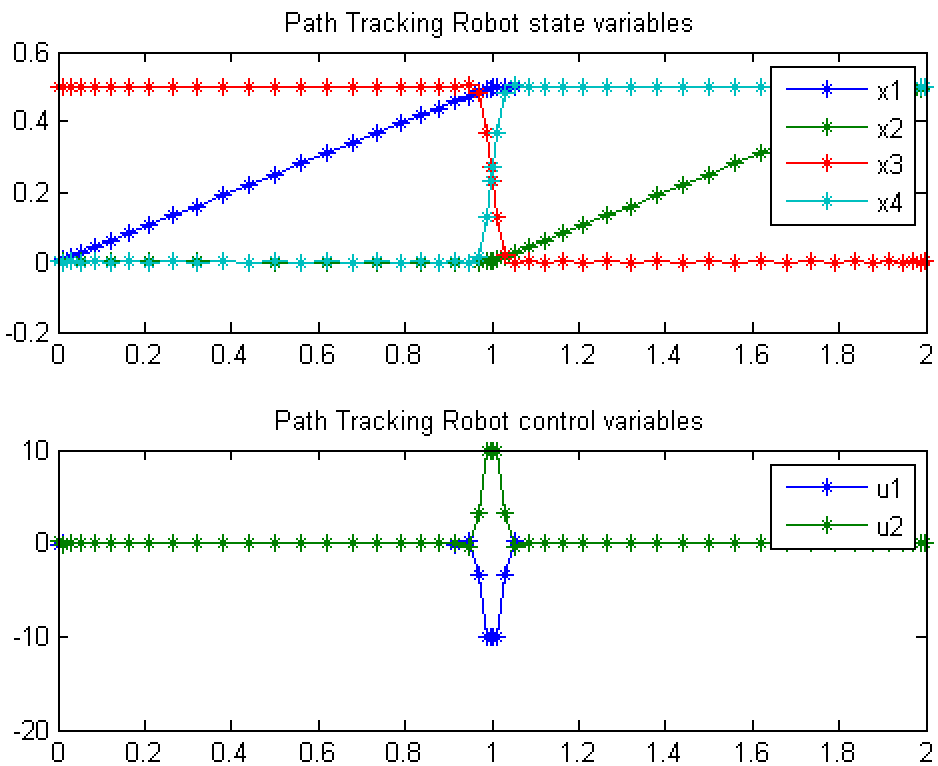
```
options = struct;
options.name = 'Path Tracking Robot (Two-Phase)';
options.solver = 'sqopt7';
constr = {cbox1, cbnd1, ceq1, cbox2, cbnd2, ceq2, link};
solution = ezsolve(objective, constr, [], options);
t = subs(collocate(p1,t1),solution);
t = [t;subs(collocate(p2,t2),solution)];
x1 = subs(collocate(p1,x1p1),solution);
x1 = [x1;subs(collocate(p2,x1p2),solution)];
x2 = subs(collocate(p1,x2p1),solution);
x2 = [x2;subs(collocate(p2,x2p2),solution)];
x3 = subs(collocate(p1,x3p1),solution);
x3 = [x3;subs(collocate(p2,x3p2),solution)];
x4 = subs(collocate(p1,x4p1),solution);
x4 = [x4;subs(collocate(p2,x4p2),solution)];
u1 = subs(collocate(p1,u1p1),solution);
u1 = [u1;subs(collocate(p2,u1p2),solution)];
u2 = subs(collocate(p1,u2p1),solution);
u2 = [u2;subs(collocate(p2,u2p2),solution)];
```

```
Problem type appears to be: qp
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem:  1: Path Tracking Robot (Two-Phase)    f_k       1.049952991377210800
                                      sum(|constr|)      0.000000009563757529
                             f(x_k) + sum(|constr|)      1.049953000940968300
                                             f(x_0)      0.000000000000000000

Solver: SQOPT.  EXIT=0.  INFORM=1.
SQOPT 7.2-5 QP solver
Optimality conditions satisfied

Iter  294
CPU time: 0.046875 sec. Elapsed time: 0.047000 sec.
```

84.4 Plot result

```
subplot(2,1,1);
plot(t,x1,'*-',t,x2,'*-',t,x3,'*-',t,x4,'*-');
legend('x1','x2','x3','x4');
title('Path Tracking Robot state variables');
```

```
subplot(2,1,2);
plot(t,u1,'*-',t,u2,'*-');
legend('u1','u2');
title('Path Tracking Robot control variables');
```



85 Pendulum Gravity Estimation

User's Guide for DIRCOL

Problem 2.5 Pendulum

85.1 Problem Formulation

Find u over t in [0; 2] to minimize

$$J = \frac{1}{2} * ((x_{t_f} - x_{f_{meas}})^2 + (y_{t_f} - y_{f_{meas}})^2)$$

subject to:

$$\frac{dx}{dt} = u$$
$$\frac{dy}{dt} = v$$
$$\frac{du}{dt} = lambda * x/m$$
$$\frac{dv}{dt} = lambda * y/m - g$$
$$x^2 + y^2 - L^2 = 0$$
$$[x_0 \ y_0 \ u_0 \ v_0] = [0.4 \ -0.3 \ 0 \ 0]$$
$$[x_{f_{meas}} \ y_{f_{meas}}] = [-0.231625 \ -0.443109]$$
$$L = 0.5$$
$$m = 0.3$$

Reference: [33]

85.2 Problem setup

```
toms t g

% Initial guess
gopt = 20;
xopt = 0.4-(0.4+0.231625)*t/2;
```

```
yopt = -0.3-(-0.3+0.443109)*t/2;
uopt = 0;
vopt = 0;
lambdaopt = -5;

for n=[20 51]

    p = tomPhase('p', t, 0, 2, n);
    setPhase(p);
    tomStates x y u v
    tomControls lambda

    % Initial guess
    x0 = {g == gopt
        icollocate({
        x == xopt
        y == yopt
        u == uopt
        v == vopt})
        collocate(lambda == lambdaopt)};

    % Box constraints
    cbox = {1 <= g <= 100};

    % Boundary constraints
    cbnd = initial({x == 0.4; y == -0.3
        u == 0; v == 0});

    L   = 0.5;
    m   = 0.3;
    xmeas = -0.231625;
    ymeas = -0.443109;

    % ODEs and path constraints
    ceq = collocate({
        dot(x) == u
        dot(y) == v
        dot(u) == lambda.*x/m
        dot(v) == lambda.*y/m-g
        x.^2 + y.^2 - L^2 == 0});

    % Objective
    objective = 1/2*((final(x)-xmeas)^2+(final(y)-ymeas)^2);
```

85.3 Solve the problem

```
    options = struct;
```

```
    options.name = 'Pendulum Gravity';
    solution = ezsolve(objective, {cbox, cbnd, ceq}, x0, options);
    gopt = subs(g, solution);
    xopt = subs(x, solution);
    yopt = subs(y, solution);
    uopt = subs(u, solution);
    vopt = subs(v, solution);
    lambdaopt = subs(lambda, solution);
```

```
Problem type appears to be: qpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Pendulum Gravity               f_k       0.000000017178935982
                                      sum(|constr|)      0.000001384358714611
                             f(x_k) + sum(|constr|)      0.000001401537650593
                                             f(x_0)     -0.124997863253000020

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv   98 ConJacEv   98 Iter   43 MinorIter   96
CPU time: 0.265625 sec. Elapsed time: 0.266000 sec.


Problem type appears to be: qpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Pendulum Gravity               f_k       0.000000000017699633
                                      sum(|constr|)      0.000000032137877804
                             f(x_k) + sum(|constr|)      0.000000032155577437
                                             f(x_0)     -0.124997846074063950

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv   13 ConJacEv   13 Iter   12 MinorIter  190
CPU time: 0.437500 sec. Elapsed time: 0.484000 sec.
```

```
end
```

85.4 Show result

```
disp(sprintf('Gravity estimated to %g',gopt));
```

```
Gravity estimated to 9.82655
```

86 Penicillin Plant

Fed-batch Fermentor Control: Dynamic Optimization of Batch Processes II. Role of Measurements in Handling Uncertainty 2001, B. Srinivasan, D. Bonvin, E. Visser, S. Palanki

Illustrative example: Nominal Optimization of a Fed-Batch Fermentor for Penicillin Production.

86.1 Problem description

This particular example was featured in the work of B. Srinivasan et al. 2001. The optimal trajectories for the problem was provided in the work.

In this problem, the objective is to maximize the concentration of penicillin, P, produced in a fed-batch bioreactor, given a finite amount of time.

```
Reactions: S -> X, S -> P
Conditions: Fed-batch, isothermal.
Objective: Maximize the concentration of product P at a given final time.
Manipulated variable: Feed rate of S.
Constraints: Input bounds; upper limit on the biomass concentration,
which is motivated by oxygen-transfer limitation typically occurring at
large biomass concentration.
```

$$J = -P(t_f)$$

subject to:

$$\begin{aligned}
\frac{X}{dt} &= my * X - \frac{u}{V} * X \\
\frac{S}{dt} &= -\frac{my * X}{Yx} - \frac{v * X}{Yp} + \frac{u}{V} * (Sin - S) \\
\frac{P}{dt} &= v * X - \frac{u}{V} * P \\
\frac{V}{dt} &= u
\end{aligned}$$

$$my = \frac{um * S}{Km + S + S^2/Ki}$$

Programmer: Wee Kiat Lim (Nanyang Technological University)

86.2 Problem setup

Penalty for variations in u

```
penalty_constant = 0.001;

% Various constants
miu_m = 0.02;  Km   = 0.05;  Ki = 5;
Yx    = 0.5;   Yp   = 1.2;   v = 0.004;
Sin   = 200;   umin = 0;     umax = 1;
Xmin  = 0;     Xmax = 3.7;   Smin = 0;

% no. of collocation points to use
narr = [20 80];
for n=narr
    toms t1
    toms tcut
    p1 = tomPhase('p1', t1, 0, tcut, n);

    setPhase(p1);
    tomStates X1 S1 P1 V1 %Vs %Scaling is disabled here
    tomControls u1

    % Initial guess
    if n == narr(1)
        x01 = {tcut == 75
            icollocate({X1 == 1+2.7*t1/tcut; S1 == 0.5;
            P1 == 0.6*t1/tcut; V1 == 150})
            collocate(u1 == 0.03+0.06*t1/tcut)};
    else
        x01 = {tcut == tcutg
            icollocate({X1 == Xg1; S1 == Sg1; P1 == Pg1; V1 == Vg1})
            collocate(u1 == ug1)};
    end

    % Box constraints
    cbox1 = {75 <= tcut <= 85
        0 <= icollocate(X1) <= Xmax
        Smin <= icollocate(S1) <= 100
        0 <= icollocate(P1) <= 5
        1 <= icollocate(V1) <= 300
```

```
    umin <= collocate(u1) <= umax};

% Boundary constraints
cbnd1 = initial({X1 == 1; S1 == 0.5;
    P1 == 0; V1 == 150});

miu1 = (miu_m*S1)/(Km + S1 + S1^2/Ki);

% ODEs and path constraints
temp11 = miu1*X1;
temp21 = u1/V1;
temp31 = v*X1;
ceq1 = collocate ({
    dot(X1) == temp11 - u1/V1*X1
    dot(S1) == -temp11/Yx - temp31/Yp + temp21*(Sin - S1)
    dot(P1) == temp31 - temp21*P1
    dot(V1) == u1});

if n == narr(1)
    % No objective in first phase
    objective = 0;
else
    % Variation penalty
    objective = penalty_constant*integrate(dot(u1)^2);
end

toms t2
p2 = tomPhase('p2', t2, tcut, 150-tcut, n);

setPhase(p2);
tomStates X2 S2 P2 V2 %Vs %Scaling is disabled here
tomControls u2

% Initial guess
if n == narr(1)
    x02 = {
        icollocate({X2 == Xmax; S2 == 0; P2 == 0.6+t2/150; V2 == 150});
        collocate(u2 == 0.01);
        };
else
    x02 = {
        icollocate({X2 == Xg2; S2 == Sg2; P2 == Pg2; V2 == Vg2})
        collocate(u2 == ug2)
        };
end

% Box constraints
```

```
umax2 = 0.03;
cbox2 = {0 <= icollocate(X2) <= Xmax
    Smin <= icollocate(S2) <= 100
    0 <= icollocate(P2) <= 5
    1 <= icollocate(V2) <= 300
    umin <= collocate(u2) <= umax2
    initial(S2) <= 0.2};

miu2 = (miu_m*S2)/(Km + S2 + S2^2/Ki);

% ODEs and path constraints
temp12 = miu2*X2;
temp22 = u2/V2;
temp32 = v*X2;
ceq2 = collocate ({
    dot(X2) == temp12 - u2/V2*X2
    dot(S2) == -temp12/Yx - temp32/Yp + temp22*(Sin - S2)
    dot(P2) == temp32 - temp22*P2
    dot(V2) == u2});

% Phase links
links = {initial(X2) == final(p1,X1)
    initial(S2) == final(p1,S1)
    initial(P2) == final(p1,P1)
    initial(V2) == final(p1,V1)};

if n == narr(1)
    % Objective (Negative sign is added to 'maximize' P)
    objective = -final(P2);
    ptype = 'lpcon';
    solver = 'snopt';
else
    objective = objective-final(P2)+penalty_constant*integrate(dot(u2)^2);
    ptype = 'con';
    solver = 'snopt';
end

% Solve the problem
options = struct;
options.name = 'Penicillin Plant';
Prob = sym2prob(ptype, objective, {cbox1, cbnd1, ceq1, cbox2, ceq2, links}, {x01, x02}, options);
Result = tomRun(solver, Prob, 1);
solution = getSolution(Result);

ug1 = subs(u1,solution);
Xg1 = subs(X1,solution);
Sg1 = subs(S1,solution);
```

```
    Pg1 = subs(P1,solution);
    Vg1 = subs(V1,solution);
    ug2 = subs(u2,solution);
    Xg2 = subs(X2,solution);
    Sg2 = subs(S2,solution);
    Pg2 = subs(P2,solution);
    Vg2 = subs(V2,solution);
    tcutg = solution.tcut;
end
```

```
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Penicillin Plant               f_k      -1.682729746070069400
                                       sum(|constr|)      0.000000953950677808
                              f(x_k) + sum(|constr|)     -1.682728792119391600
                                              f(x_0)     -1.599999999999999600

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv  950 ConJacEv  950 Iter  282 MinorIter 3095
CPU time: 6.875000 sec. Elapsed time: 7.172000 sec.
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Penicillin Plant               f_k      -1.682693889838489600
                                       sum(|constr|)      0.000001548138130567
                              f(x_k) + sum(|constr|)     -1.682692341700359000
                                              f(x_0)     -1.682727190779594400

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv   18 GradEv   16 ConstrEv   16 ConJacEv   16 Iter   15 MinorIter  955
CPU time: 3.656250 sec. Elapsed time: 3.687000 sec.
```

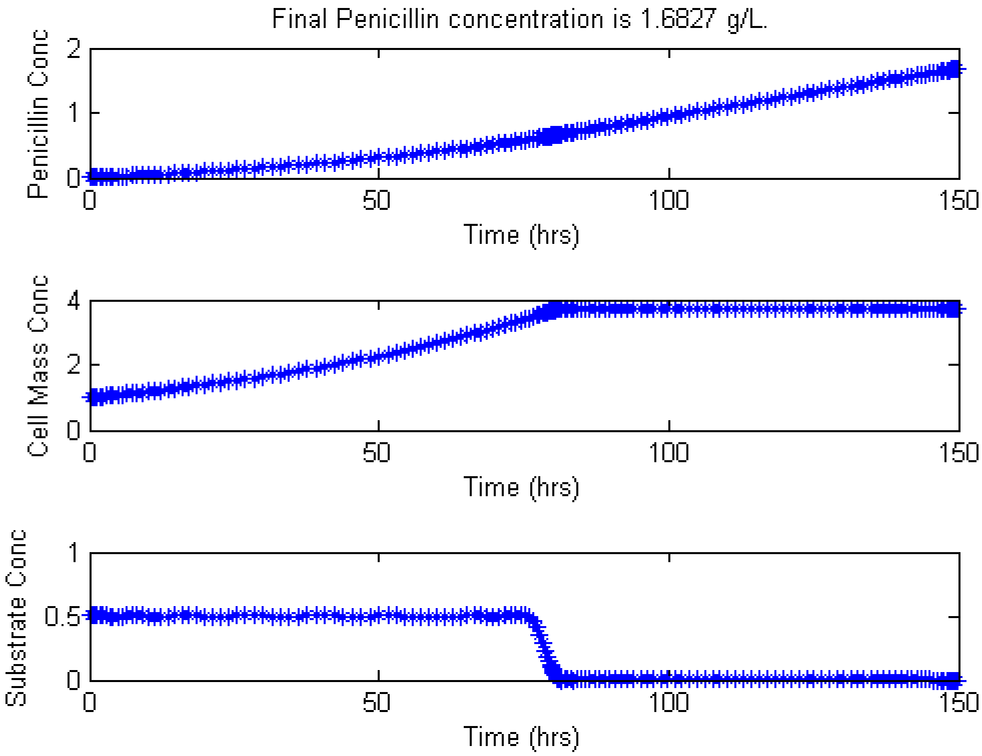
86.3 Plot result

Optimal states and control trajectories

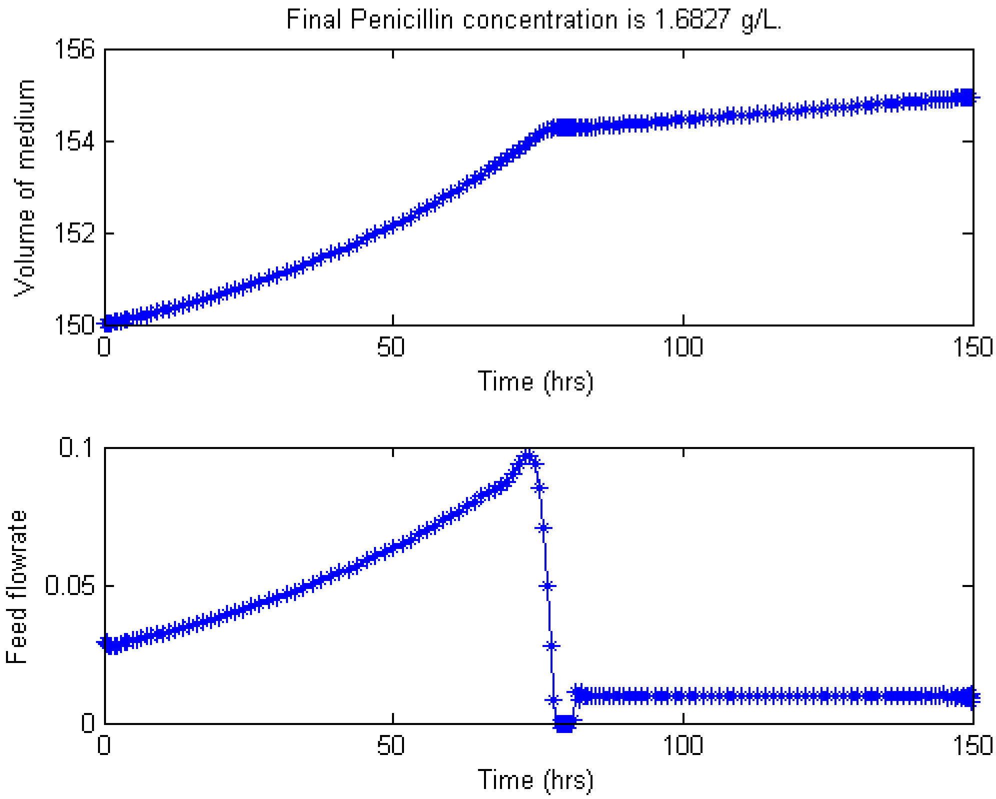
```
uopt = subs([collocate(p1,u1);collocate(p2,u2)],solution);
Xopt = subs([collocate(p1,X1);collocate(p2,X2)],solution);
```

```
Sopt = subs([collocate(p1,S1);collocate(p2,S2)],solution);
Popt = subs([collocate(p1,P1);collocate(p2,P2)],solution);
Vopt = subs([collocate(p1,V1);collocate(p2,V2)],solution);

t = subs([collocate(p1,t1);collocate(p2,t2)],solution);
np = length(t);

Pfinal=subs(final(p2,P2),solution);

% Plots of the trajectories
figure(1)
subplot(3,1,1);
plot(t,Popt,'*-');
title(['Final Penicillin concentration is ',num2str(Pfinal),' g/L.'])
ylabel('Penicillin Conc')
xlabel('Time (hrs)')
subplot(3,1,2);
plot(t,Xopt,'*-');
ylabel('Cell Mass Conc')
xlabel('Time (hrs)')
subplot(3,1,3);
plot(t,Sopt,'*-');
ylabel('Substrate Conc')
xlabel('Time (hrs)')

figure(2)
subplot(2,1,1);
plot(t,Vopt,'*-');
title(['Final Penicillin concentration is ',num2str(Pfinal),' g/L.'])
ylabel('Volume of medium')
xlabel('Time (hrs)')
subplot(2,1,2);
plot(t,uopt,'*-');
ylabel('Feed flowrate')
xlabel('Time (hrs)')

fprintf('\n')
fprintf('Optimization completed... \n')
fprintf('Final Penicillin concentration is %5.4f g/L.\n',Pfinal)
```

```
Optimization completed...
Final Penicillin concentration is 1.6827 g/L.
```



Final Penicillin concentration is 1.6827 g/L.
Penicillin Conc
Time (hrs)
Cell Mass Conc
Time (hrs)
Substrate Conc
Time (hrs)



Final Penicillin concentration is 1.6827 g/L.
156
154
152
150
Volume of medium
0
50
100
150
Time (hrs)
0.1
0.05
0
Feed flowrate
0
50
100
150
Time (hrs)

87 Plug-Flow Tubular Reactor

A HYBRID METHOD FOR THE OPTIMAL CONTROL OF CHEMICAL PROCESSES 1998, E F Carrasco, J R Banga

Case Study II: Plug-Flow Tubular Reactor

87.1 Problem description

This case study considers a plug-flow reactor as studied by Reddy and Husain, Luus and Mekarapiruk and Luus. The objective is to maximize the normalized concentration of the desired product.

Find u(t) to maximize

$$J = x_1(t_f)$$

subject to:

$$\frac{dx_1}{dt} = (1 - x_1) * k_1 - x_1 * k_2$$
$$\frac{dx_2}{dt} = 300 * ((1 - x_1) * k_1 - x_1 * k_2) - u * (x_2 - 290)$$

where x1 denotes the normalized concentration of de desired product, and x2 is the temperature. The initial conditions are:

$$x(t_0) = [0 \; 380]'$$

The rate constants are given by:

$$k_1 = 1.7536e5 * exp(-\frac{1.1374e4}{1.9872 * x_2})$$
$$k_2 = 2.4885e10 * exp(-\frac{2.2748e4}{1.9872 * x_2})$$

where the final time t_f = 5 min. The constraint on the control variable (the normalized coolant flow rate) is:

$$0 <= u <= 0.5$$

In addition, there is an upper path constraint on the temperature:

$$x_2(t) <= 460$$

Reference: [11]

87.2 Problem setup

```
toms t
p = tomPhase('p', t, 0, 5, 30);
setPhase(p);

tomStates x1 x2
tomControls u

% Initial guess
x0 = {icollocate({x1 == 0.6*t/5
    x2 == 380})
    collocate(u == 0.25)};

% Box constraints
cbox = {0 <= icollocate(x1) <= 10
    100 <= icollocate(x2) <= 460
    0   <= collocate(u)   <= 0.5};

% Boundary constraints
cbnd = initial({x1 == 0; x2 == 380});

% ODEs and path constraints
k1 = 1.7536e5*exp(-1.1374e4/1.9872./x2);
k2 = 2.4885e10*exp(-2.2748e4/1.9872./x2);
ceq = collocate({
    dot(x1) == (1-x1).*k1-x1.*k2
    dot(x2) == 300*((1-x1).*k1-x1.*k2)-u.*(x2-290)});

% Objective
objective = -final(x1);
```

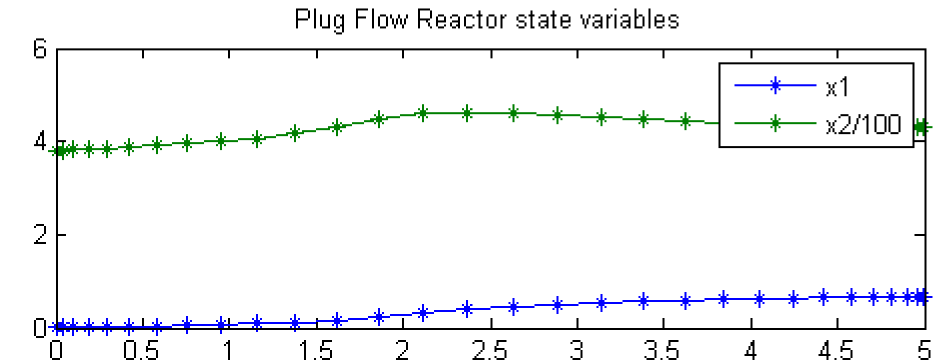
87.3 Solve the problem

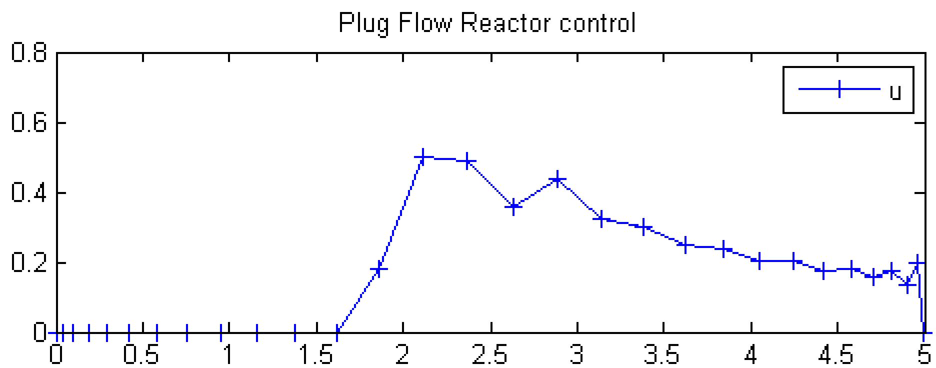
```
options = struct;
options.name = 'Plug Flow Reactor';
solution = ezsolve(objective, {cbox, cbnd, ceq}, x0, options);
t  = subs(collocate(t),solution);
x1 = subs(collocate(x1),solution);
x2 = subs(collocate(x2),solution);
u  = subs(collocate(u),solution);
```

```
Problem type appears to be: lpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Plug Flow Reactor              f_k      -0.677125737913556680
                                       sum(|constr|)      0.000031724693791993
                              f(x_k) + sum(|constr|)     -0.677094013219764700
                                              f(x_0)     -0.599999999999999200

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv  346 ConJacEv  346 Iter  112 MinorIter  325
CPU time: 0.796875 sec. Elapsed time: 0.813000 sec.
```

87.4 Plot result

```
subplot(2,1,1)
plot(t,x1,'*-',t,x2/100,'*-');
legend('x1','x2/100');
title('Plug Flow Reactor state variables');

subplot(2,1,2)
plot(t,u,'+-');
legend('u');
title('Plug Flow Reactor control');
```


Plug Flow Reactor state variables
x1
x2/100
6
4
2
0
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5


Plug Flow Reactor control
u
0.8
0.6
0.4
0.2
0
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5

88 Quadratic constraint problem

Paper: LINEAR-QUADRATIC OPTIMAL CONTROL WITH INTEGRAL QUADRATIC CONSTRAINTS. OPTIMAL CONTROL APPLICATIONS AND METHODS Optim. Control Appl. Meth., 20, 79-92 (1999)

E. B. LIM(1), Y. Q. LIU(2), K. L. TEO(2) AND J. B. MOORE(1)

(1) Department of Systems Engineering, Research School of Information Sciences and Engineering, Australian National University, Canberra ACT 0200, Australia

(2) School of Mathematics and Statistics, Curtin University of Technology, Perth, WA 6845, Australia

88.1 Problem Formulation

Find u(t) over t in [0; 1] to minimize

$$J = 0.5 * x_1(1)^2 + 0.5 * \int_0^1 (x_1^2 + u_1^2 + u_2^2)\mathrm{d}t$$

subject to:

$$\frac{dx_1}{dt} = 3 * x_1 + x_2 + u_1$$
$$\frac{dx_2}{dt} = -x_1 + 2 * x_2 + u_2$$
$$x_1(0) = 4$$
$$x_2(0) = -4$$
$$0.5 * x_2(1)^2 + 0.5 * \int_0^1 (x_1^2 + u_1^2 + u_2^2) <= 80$$

Introduce a new variable to remove integral in constraint:

$$\frac{dx_3}{dt} = 0.5 * (x_1.^2 + u_1.^2 + u_2.^2)$$

resulting in event constraint:

$$0.5 * x_2(1)^2 + x_3(1) <= 8$$

Reference: [24]

88.2 Problem setup

```
toms t
p = tomPhase('p', t, 0, 1, 50);
setPhase(p);

tomStates x1 x2 x3
tomControls u1 u2

% Initial guess
x0 = {icollocate({
    x1 == 4-5*t
    x2 == -4-1*t
    x3 == 50*t
    })
    collocate({
    u1 == -10+10*t
    u2 == 14-12*t})};

% Boundary constraints
cbnd = {
    initial({
    x1 == 4
    x2 == -4
    x3 == 0
    })
    final(x2)^2/2+final(x3) <= 80};

% ODEs and path constraints
ceq = collocate({
    dot(x1) == 3*x1+x2 + u1
    dot(x2) == -x1+2*x2 + u2
    dot(x3) == 1/2 * (x2.^2 + u1.^2 + u2.^2)
    });

% Objective
objective = final(x1)^2/2 + final(x3);
```

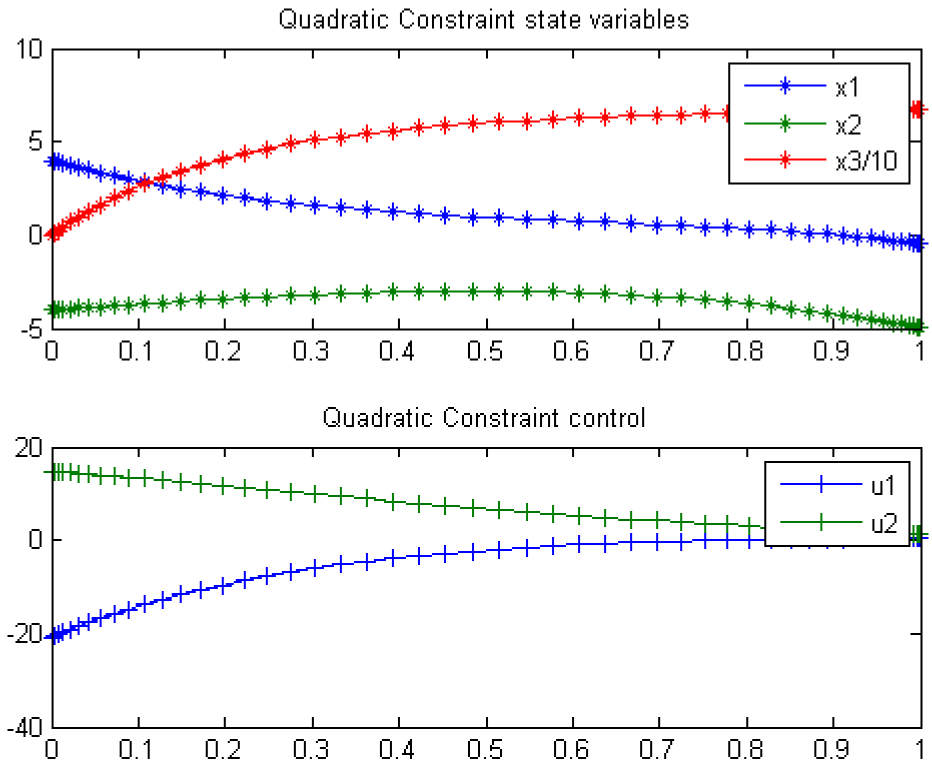
88.3 Solve the problem

```
options = struct;
options.name = 'Quadratic Constraint';
solution = ezsolve(objective, {cbnd, ceq}, x0, options);
t  = subs(collocate(t),solution);
x1 = subs(collocate(x1),solution);
x2 = subs(collocate(x2),solution);
x3 = subs(collocate(x3),solution);
u1 = subs(collocate(u1),solution);
u2 = subs(collocate(u2),solution);
```

```
Problem type appears to be: qpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Quadratic Constraint           f_k      67.888740121887395000
                                       sum(|constr|)      0.000000192425266868
                              f(x_k) + sum(|constr|)     67.888740314312656000
                                              f(x_0)     50.499999999999915000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv   32 ConJacEv   32 Iter   31 MinorIter  280
CPU time: 0.593750 sec. Elapsed time: 0.578000 sec.
```

88.4 Plot result

```
subplot(2,1,1)
plot(t,x1,'*-',t,x2,'*-',t,x3/10,'*-');
legend('x1','x2','x3/10');
title('Quadratic Constraint state variables');

subplot(2,1,2)
plot(t,u1,'+-',t,u2,'+-');
legend('u1','u2');
title('Quadratic Constraint control');
```


Quadratic Constraint state variables
x1
x2
x3/10
10
5
0
-5
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Quadratic Constraint control
u1
u2
20
0
-20
-40
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

89 Quadruple Integral

ITERATIVE DYNAMIC PROGRAMMING, REIN LUUS

12.4.5 Example 5

CHAPMAN & HALL/CRC Monographs and Surveys in Pure and Applied Mathematics

89.1 Problem Formulation

Find u over t in [0; t_F] to minimize

$$J = t_F$$

subject to:

$$\frac{dx_1}{dt} = x_2$$
$$\frac{dx_2}{dt} = x_3$$
$$\frac{dx_3}{dt} = x_4$$
$$\frac{dx_4}{dt} = u$$

The initial condition are:

$$x(0) = [0.1\ 0.2\ 0.3\ 0]$$
$$x(t_F) = [0\ 0\ 0\ 0]$$
$$-1 <= u <= 1$$

Reference: [25]

89.2 Problem setup

```
toms t
toms t_f
p = tomPhase('p', t, 0, t_f, 30);
setPhase(p);

tomStates x1 x2 x3 x4
tomControls u

% Initial guess
x0 = {t_f == 5
    icollocate({x1 == 0.1-0.1*t/t_f
    x2 == 0.2-0.2*t/t_f; x3 == 0.3-0.3*t/t_f})
    collocate(u == -1)};

% Box constraints
cbox = {0.1 <= t_f <= 100
    -1 <= collocate(u) <= 1};

% Boundary constraints
cbnd = {initial({x1 == 0.1; x2 == 0.2; x3 == 0.3; x4 == 0})
    final({x1 == 0; x2 == 0; x3 == 0; x4 == 0})};

% ODEs and path constraints
ceq = collocate({
    dot(x1) == x2; dot(x2) == x3
    dot(x3) == x4; dot(x4) == u});

% Objective
objective = t_f;
```

89.3 Solve the problem

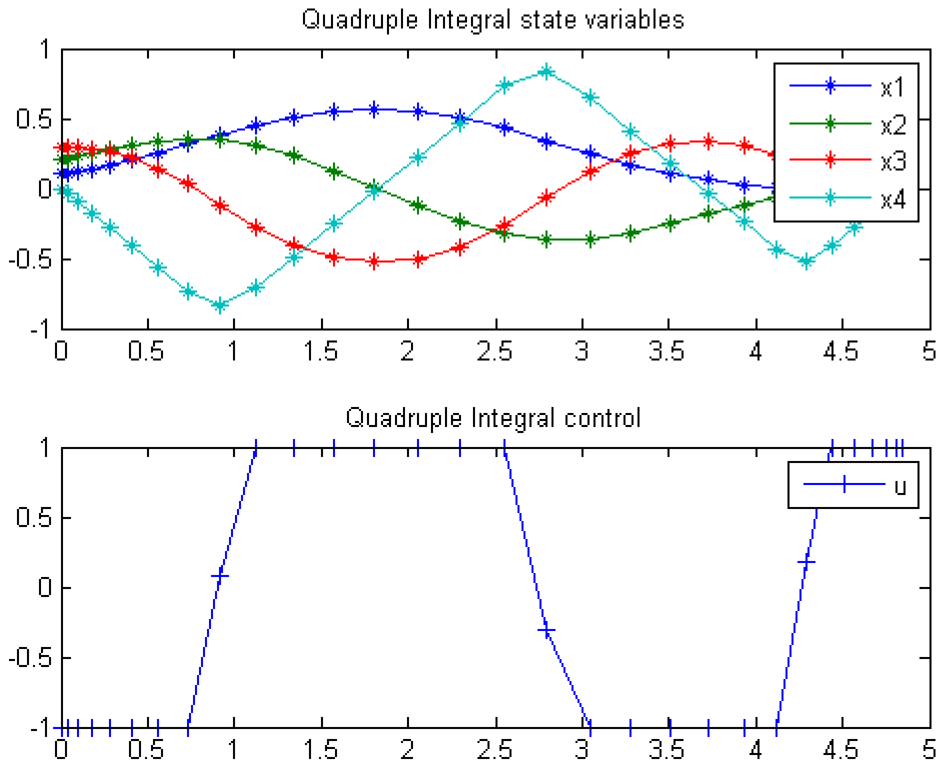
```
options = struct;
options.name = 'Quadruple Integral';
solution = ezsolve(objective, {cbox, cbnd, ceq}, x0, options);
t  = subs(collocate(t),solution);
x1 = subs(collocate(x1),solution);
x2 = subs(collocate(x2),solution);
x3 = subs(collocate(x3),solution);
x4 = subs(collocate(x4),solution);
u  = subs(collocate(u),solution);
```

```
Problem type appears to be: lpcon
Starting numeric solver
```

```
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Quadruple Integral             f_k       4.849598187644263100
                                       sum(|constr|)      0.000000078353877264
                              f(x_k) + sum(|constr|)      4.849598265998140300
                                              f(x_0)      5.000000000000000000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv   58 ConJacEv   58 Iter   31 MinorIter  474
CPU time: 0.250000 sec. Elapsed time: 0.282000 sec.
```

89.4 Plot result

```
subplot(2,1,1)
plot(t,x1,'*-',t,x2,'*-',t,x3,'*-',t,x4,'*-');
legend('x1','x2','x3','x4');
title('Quadruple Integral state variables');

subplot(2,1,2)
plot(t,u,'+-');
legend('u');
title('Quadruple Integral control');
```


Quadruple Integral state variables
x1
x2
x3
x4
Quadruple Integral control
u

90 Radio telescope

90.1 Problem description

Time-optimal positioning of a radio telescope with bounded control.

Programmers: Gerard Van Willigenburg (Wageningen University) Willem De Koning (retired from Delft University of Technology)

90.2 Problem setup

```
% Array with consecutive number of collocation points
narr = [20 40];

% Free final time
toms t t_f

for n=narr

    p = tomPhase('p', t, 0, t_f, n);
    setPhase(p)

    tomStates x1 x2 x3
    tomControls u1

    % Initial & terminal states
    xi = [0; 0; 0];
    xf = [1; 0; 0];

    % Initial guess
    if n==narr(1)
        x0 = {t_f == 5; icollocate({x1 == xi(1); x2 == xi(2)
            x3 == xi(3)})
            collocate({u1 == 0})};
    else
        x0 = {t_f == tfopt; icollocate({x1 == xopt1; x2 == xopt2
            x3 == xopt3})
            collocate({u1 == uopt1})};
    end

    % Box constraints
    cbox = {-1 <= collocate(u1) <= 1, 0.1 <= t_f <= 10};
```

```
% Boundary constraints
cbnd = {initial({x1 == xi(1); x2 == xi(2); x3 == xi(3)})
    final({x1 == xf(1); x2 == xf(2); x3 == xf(3)})};

% ODEs and path constraints
dx1 = x2;
dx2 = -0.5*x2-0.1*x2./sqrt(x2.*x2+1e-4)+x3;
dx3 = -2*x3+2*u1;

ceq = collocate({
    dot(x1) == dx1
    dot(x2) == dx2
    dot(x3) == dx3});

% Objective
objective = t_f;
```

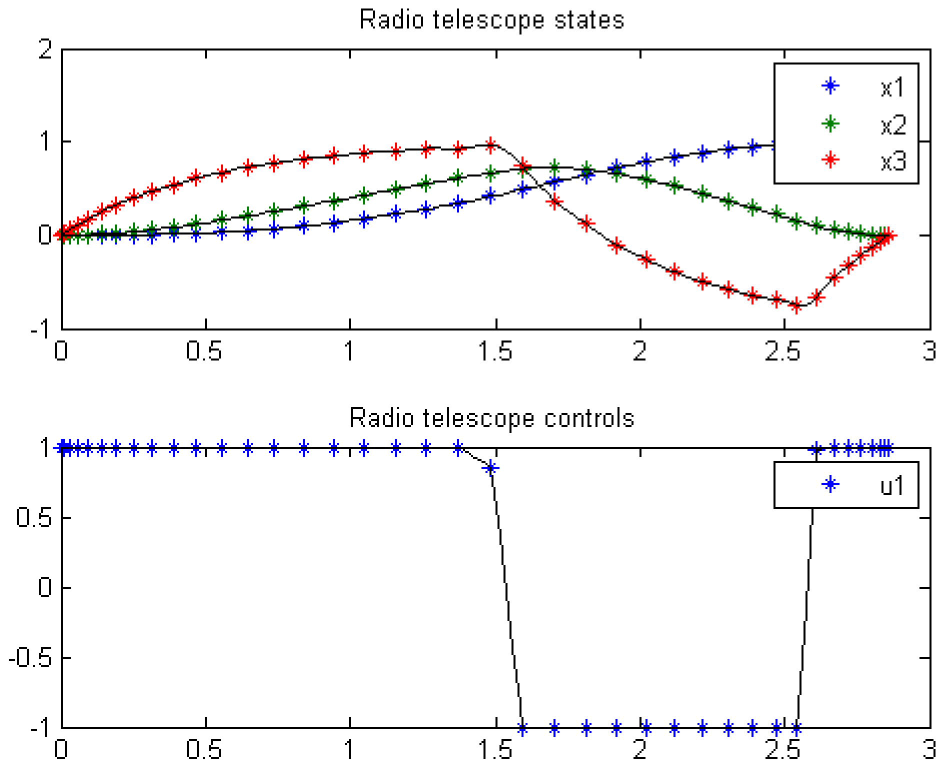
90.3 Solve the problem

```
options = struct;
options.name = 'Radio telescope';
solution = ezsolve(objective, {cbox, cbnd, ceq}, x0, options);

tfopt = subs(t_f,solution);
xopt1 = subs(x1,solution);
xopt2 = subs(x2,solution);
xopt3 = subs(x3,solution);
uopt1 = subs(u1,solution);
```

```
Problem type appears to be: lpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Radio telescope                f_k       2.866900592275516900
                                       sum(|constr|)      0.000001022940510087
                              f(x_k) + sum(|constr|)      2.866901615216026900
                                              f(x_0)      5.000000000000000000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv   41 ConJacEv   41 Iter   22 MinorIter  169
CPU time: 0.109375 sec. Elapsed time: 0.110000 sec.
```

```
Problem type appears to be: lpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Radio telescope               f_k       2.859991999288976800
                                       sum(|constr|)      0.000002602280303807
                              f(x_k) + sum(|constr|)      2.859994601569280500
                                              f(x_0)      2.866900592275516900

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv    5 ConJacEv    5 Iter    4 MinorIter  133
CPU time: 0.062500 sec. Elapsed time: 0.062000 sec.
```

```
end

figure(1)
subplot(2,1,1);
ezplot([x1; x2; x3]); legend('x1','x2','x3');
title('Radio telescope states');

subplot(2,1,2);
ezplot(u1); legend('u1');
title('Radio telescope controls');
```


Radio telescope states
x1
x2
x3
Radio telescope controls
u1

91 Rayleigh Unconstrained

Lecture Notes for ECE/MAE 7360, Robust and Optimal Control (part 2) Fall 2003, Jinsong Liang, Nov. 20, 2003 Utah State University at Logan

91.1 Problem Formulation

Find u over t in [0; t_F] to minimize

$$J = \int_0^{2.5} x_1^2 + u^2 \mathrm{d}t$$

subject to:

$$\frac{dx_1}{dt} = x_2$$

$$\frac{dx_2}{dt} = -x_1 + (1.4 - 0.14 * x_2^2) * x_2 + 4 * u$$

$$x_1(0) = -5$$

$$x_2(0) = -5$$

Reference: [22]

91.2 Problem setup

```
toms t
p = tomPhase('p', t, 0, 2.5, 50);
setPhase(p);

tomStates x1 x2
tomControls u

% Initial guess
x0 = {icollocate({x1 == -5; x2 == -5})
    collocate(u == 0)};

% Box constraints
cbox = {-100 <= icollocate(x1) <= 100
    -100 <= icollocate(x2) <= 100
```

```
    -100 <= collocate(u) <= 100};

% Boundary constraints
cbnd = initial({x1 == -5; x2 == -5});

% ODEs and path constraints
ceq = collocate({dot(x1) == x2
    dot(x2) == -x1+(1.4-0.14*x2.^2).*x2+4*u});

% Objective
objective = integrate(x1.^2+u.^2);
```

91.3 Solve the problem

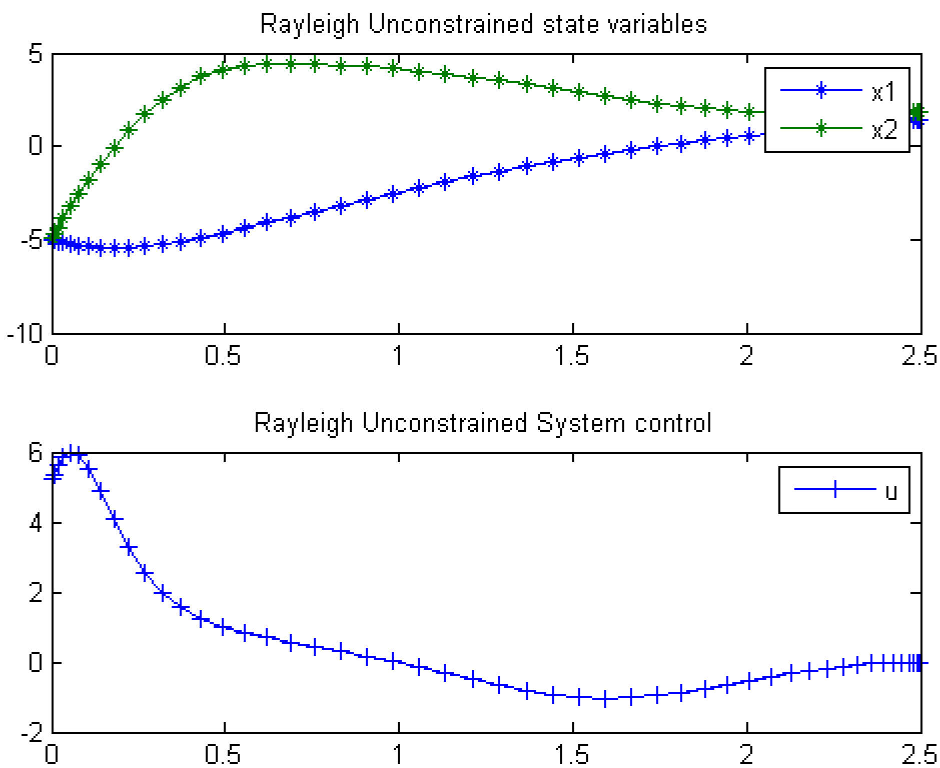
```
options = struct;
options.name = 'Rayleigh Unconstrained';
solution = ezsolve(objective, {cbox, cbnd, ceq}, x0, options);
t  = subs(collocate(t),solution);
x1 = subs(collocate(x1),solution);
x2 = subs(collocate(x2),solution);
u  = subs(collocate(u),solution);
```

```
Problem type appears to be: qpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Rayleigh Unconstrained         f_k      29.376079656023496000
                                       sum(|constr|)      0.000000003740997838
                              f(x_k) + sum(|constr|)     29.376079659764493000
                                              f(x_0)     62.499999999999986000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv   47 ConJacEv   47 Iter   36 MinorIter  144
CPU time: 0.250000 sec. Elapsed time: 0.266000 sec.
```

91.4 Plot result

```
subplot(2,1,1)
plot(t,x1,'*-',t,x2,'*-');
legend('x1','x2');
title('Rayleigh Unconstrained state variables');
```

```
subplot(2,1,2)
plot(t,u,'+-');
legend('u');
title('Rayleigh Unconstrained System control');
```



92 Rigid Body Rotation

On smooth optimal control determination, Ilya Ioslovich and Per-Olof Gutman, Technion, Israel Institute of Technology.

Example 1: Rigid body rotation

92.1 Problem Description

Find u over t in [0; 1] to minimize:

$$J = \frac{1}{4} * \int_0^1 (u_1^2 + u_2^2)^2 \mathrm{d}t$$

subject to:

$$\frac{dx}{dt} = a * y + u_1$$
$$\frac{dy}{dt} = -a * x + u_2$$
$$\frac{du_1}{dt} = a * u_2$$
$$\frac{du_2}{dt} = -a * u_1$$
$$x(t_0) = [0.9 \; 0.75]$$
$$x(t_f) = [0 \; 0]$$
$$a = 2$$

Reference: [18]

92.2 Problem setup

```
toms t
p = tomPhase('p', t, 0, 1, 20);
setPhase(p);

tomStates x y u1 u2
```

```
% Boundary constraints
cbnd = {initial({x == 0.9; y == 0.75})
    final({x == 0; y == 0})};

% ODEs and path constraints
a = 2;
ceq = collocate({dot(x)  == a*y+u1; dot(y)  == -a*x+u2
    dot(u1) == a*u2; dot(u2) == -a*u1});

% Objective
objective = 0.25*integrate((u1.^2+u2.^2).^2);
```

92.3 Solve the problem

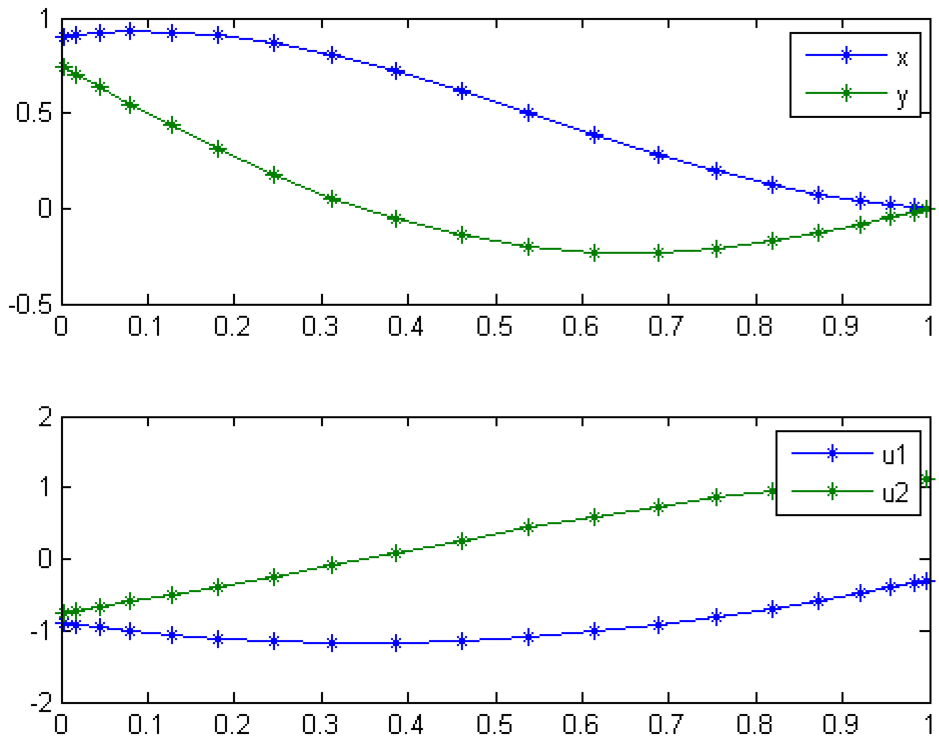
```
options = struct;
options.name = 'Rigid Body Rotation';
solution = ezsolve(objective, {cbnd, ceq}, [], options);
t  = subs(collocate(t),solution);
x = subs(collocate(x),solution);
y = subs(collocate(y),solution);
u1 = subs(collocate(u1),solution);
u2 = subs(collocate(u2),solution);
```

```
Problem type appears to be: con
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Rigid Body Rotation            f_k       0.470939062500256130
                                       sum(|constr|)      0.000000000003070916
                              f(x_k) + sum(|constr|)      0.470939062503327070
                                              f(x_0)      0.000000000000000000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    3 GradEv    1 MinorIter   39
CPU time: 0.031250 sec. Elapsed time: 0.032000 sec.
```

92.4 Plot result

```
figure(1);
subplot(2,1,1);
plot(t,x,'*-',t,y,'*-');
```

```
legend('x','y');

subplot(2,1,2);
plot(t,u1,'*-',t,u2,'*-');
legend('u1','u2');
```



93 Robot Arm Movement

Benchmarking Optimization Software with COPS Elizabeth D. Dolan and Jorge J. More ARGONNE NATIONAL LABORATORY

93.1 Problem Formulation

Find u(t) over t in [0; 1] to minimize

$$J = t_f$$

subject to:

$$L * \frac{d^2 rho}{dt^2} = u_1$$
$$I_1 * \frac{d^2 theta}{dt^2} = u_2$$
$$I_2 * \frac{d^2 phi}{dt^2} = u_3$$

$$0 <= rho <= L$$
$$|theta| <= pi$$
$$0 <= phi <= pi$$
$$|u_{1:3}| <= 1$$

$$I_1 = \frac{((L - rho)^3 + rho^3)}{3} * sin(phi)^2$$
$$I_2 = \frac{((L - rho)^3 + rho^3)}{3}$$

The boundary conditions are:

$$[rho_0 \ theta_0 \ phi_0] = [4.5 \ 0 \ \frac{pi}{4}]$$

$$[rho_1\ theta_1\ phi_1] = [4.5\ \frac{2*pi}{3}\ \frac{pi}{4}]$$

$$L = 5$$

All first order derivatives are 0 at boundaries.

Reference: [14]

93.2 Problem setup

```
toms t
toms t_f

% Initial guess
tfopt = 1;
x1opt = 4.5;
x2opt = 0;
x3opt = 2*pi/3*t.^2;
x4opt = 0;
x5opt = pi/4;
x6opt = 0;
u1opt = 0;
u2opt = 0;
u3opt = 0;

for n=[20 100]

    %rho d(rho)dt theta d(theta)dt phi d(phi)dt
    p = tomPhase('p', t, 0, t_f, n);
    setPhase(p);
    tomStates x1 x2 x3 x4 x5 x6
    tomControls u1 u2 u3

    % Initial guess
    x0 = {t_f == tfopt
        icollocate({x1 == x1opt
        x2 == x2opt; x3 == x3opt
        x4 == x4opt; x5 == x5opt
        x6 == x6opt})
        collocate({u1 == u1opt
        u2 == u2opt; u3 == u3opt})};

    % Box constraints
    L = 5;
    cbox = {
```

```
    0.1 <= t_f <= 10
    0   <= icollocate(x1) <= L
    -pi <= icollocate(x3) <= pi
    0   <= icollocate(x5) <= pi
    -1  <= collocate(u1)  <= 1
    -1  <= collocate(u2)  <= 1
    -1  <= collocate(u3)  <= 1};

% Boundary constraints
cbnd = {initial({x1 == 4.5; x2 == 0
    x3 == 0; x4 == 0
    x5 == pi/4; x6 == 0})
    final({x1 == 4.5; x2 == 0
    x3 == 2*pi/3
    x4 == 0
    x5 == pi/4
    x6 == 0
    })};

I1 = ((L-x1).^3+x1.^3)./3.*sin(x5).^2;
I2 = ((L-x1).^3+x1.^3)/3;

% ODEs and path constraints
ceq = collocate({dot(x1) == x2
    dot(x2) == u1/L;   dot(x3) == x4
    dot(x4) == u2./I1; dot(x5) == x6
    dot(x6) == u3./I2});

% Objective
objective = t_f;
```

93.3 Solve the problem

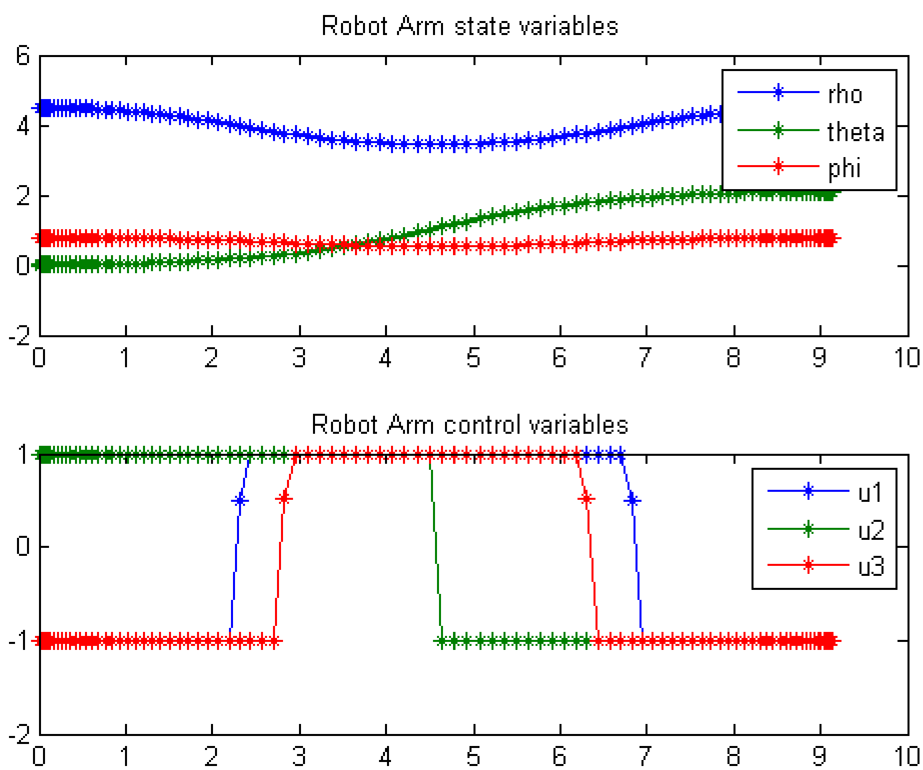
```
options = struct;
options.name = 'Robot Arm';
solution = ezsolve(objective, {cbox, cbnd, ceq}, x0, options);

% Optimal x, y, and speed, to use as starting guess in the next iteration
x1opt = subs(x1, solution);
x2opt = subs(x2, solution);
x3opt = subs(x3, solution);
x4opt = subs(x4, solution);
x5opt = subs(x5, solution);
u1opt = subs(u1, solution);
u2opt = subs(u2, solution);
u3opt = subs(u3, solution);
tfopt = subs(final(t), solution);
```

```
Problem type appears to be: lpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Robot Arm                      f_k       9.146367545948111300
                                      sum(|constr|)      0.000000273543612574
                              f(x_k) + sum(|constr|)      9.146367819491723900
                                              f(x_0)      1.000000000000000000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv   17 ConJacEv   17 Iter   11 MinorIter  225
CPU time: 0.125000 sec. Elapsed time: 0.125000 sec.


Problem type appears to be: lpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Robot Arm                      f_k       9.140854009735486200
                                      sum(|constr|)      0.000002639183837389
                              f(x_k) + sum(|constr|)      9.140856648919323000
                                              f(x_0)      9.146367545948111300

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv    7 ConJacEv    7 Iter    4 MinorIter  729
CPU time: 1.812500 sec. Elapsed time: 1.875000 sec.
```

```
end

t  = subs(collocate(t),solution);
x1 = subs(collocate(x1),solution);
x3 = subs(collocate(x3),solution);
x5 = subs(collocate(x5),solution);
u1 = subs(collocate(u1),solution);
u2 = subs(collocate(u2),solution);
u3 = subs(collocate(u3),solution);
```

93.4 Plot result

```
subplot(2,1,1)
plot(t,x1,'*-',t,x3,'*-',t,x5,'*-');
legend('rho','theta','phi');
title('Robot Arm state variables');

subplot(2,1,2)
plot(t,u1,'*-',t,u2,'*-',t,u3,'*-');
legend('u1','u2','u3');
title('Robot Arm control variables');
```



94 Time-optimal Trajectories for Robot Manipulators

Users Guide for dyn.Opt, Example 2

Dissanayake, M., Goh, C. J., & Phan-Thien, N., Time-optimal Trajectories for Robot Manipulators, Robotica, Vol. 9, pp. 131-138, 1991.

94.1 Problem Formulation

Find u over t in [0; t_F] to minimize

$$J = t_F$$

subject to:

$$x(0) = [0 \ -2 \ 0 \ 0]$$
$$x(t_F) = [1 \ -1 \ 0 \ 0]$$

$$L_1 = 0.4;$$
$$L_2 = 0.4;$$
$$m_1 = 0.5;$$
$$m_2 = 0.5;$$
$$Eye_1 = 0.1;$$
$$Eye_2 = 0.1;$$
$$el_1 = 0.2;$$
$$el_2 = 0.2;$$

$$cos(x2) = cos(x_2);$$
$$H_{11} = Eye_1 + Eye_2 + m_1 * el_1^2 + m_2 * (L_1^2 + el_2^2 + 2.0 * L_1 * el_2 * cos(x2));$$
$$H_{12} = Eye_2 + m_2 * el_2^2 + m_2 * L_1 * el_2 * cos(x2);$$
$$H_{22} = Eye_2 + m_2 * el_2^2;$$

$$h = m_2 * L_1 * el_2 * sin(x_2);$$
$$delta = \frac{1.0}{H_{11} * H_{22} - H_{12} * H_{12}};$$

$$\frac{dx_1}{dt} = x_3$$
$$\frac{dx_2}{dt} = x_4$$
$$\frac{dx_3}{dt} = delta * (2.0 * h * H_{22} * x_3 * x_4 + h * H_{22} * x_4^2 + h * H_{12} * x_3^2 + H_{22} * u_1 - H_{12} * u_2);$$
$$\frac{dx_4}{dt} = delta * (-2.0 * h * H_{12} * x_3 * x_4 - h * H_{11} * x_3^2 - h * H_{12} * x_4^2 + H_{11} * u_2 - H_{12} * u_1);$$

$$-10 <= u <= 10$$

Reference: [16]

94.2 Problem setup

```
toms t
toms t_f

tfopt = 7;
x1opt = 1*t/t_f;
x2opt = -2+1*t/t_f;
x3opt = 2;
x4opt = 4;
u1opt = 10-20*t/t_f;
u2opt = -10+20*t/t_f;
```

94.3 Solve the problem, using a successively larger number collocation points

```
for n=[30 60]

    p = tomPhase('p', t, 0, t_f, n);
    setPhase(p);

    tomStates x1 x2 x3 x4
    tomControls u1 u2
```

```
% Initial guess
x0 = {t_f == tfopt
    icollocate({
    x1 == x1opt
    x2 == x2opt
    x3 == x3opt
    x4 == x4opt})
    collocate({
    u1 == u1opt
    u2 == u2opt})};

% Box constraints
cbox = {
    0.1 <= t_f <= 50
    -10 <= collocate(u1) <= 10
    -10 <= collocate(u2) <= 10};

% Boundary constraints
cbnd = {initial({x1 == 0; x2 == -2
    x3 == 0; x4 == 0})
    final({x1 == 1; x2 == -1
    x3 == 0; x4 == 0})};

% ODEs and path constraints
L_1 = 0.4;   L_2 = 0.4;
m_1 = 0.5;   m_2 = 0.5;
Eye_1 = 0.1; Eye_2 = 0.1;
el_1 = 0.2;  el_2 = 0.2;

H_11  = Eye_1 + Eye_2 + m_1*el_1^2+ ...
    m_2*(L_1^2+el_2^2+2.0*L_1*el_2*cos(x2));
H_12  = Eye_2 + m_2*el_2^2 + m_2*L_1*el_2*cos(x2);
H_22  = Eye_2 + m_2*el_2^2;
h     = m_2*L_1*el_2*sin(x2);
delta = 1.0./(H_11.*H_22-H_12.^2);

ceq = collocate({
    dot(x1) == x3
    dot(x2) == x4
    dot(x3) == delta.*(2.0*h.*H_22.*x3.*x4 ...
    +h.*H_22.*x4.^2+h.*H_12.*x3.^2+H_22.*u1-H_12.*u2)
    dot(x4) == delta.*(-2.0*h.*H_12.*x3.*x4 ...
    -h.*H_11.*x3.^2-h.*H_12.*x4.^2+H_11.*u2-H_12.*u1)});

% Objective
objective = t_f;
```

94.4 Solve the problem

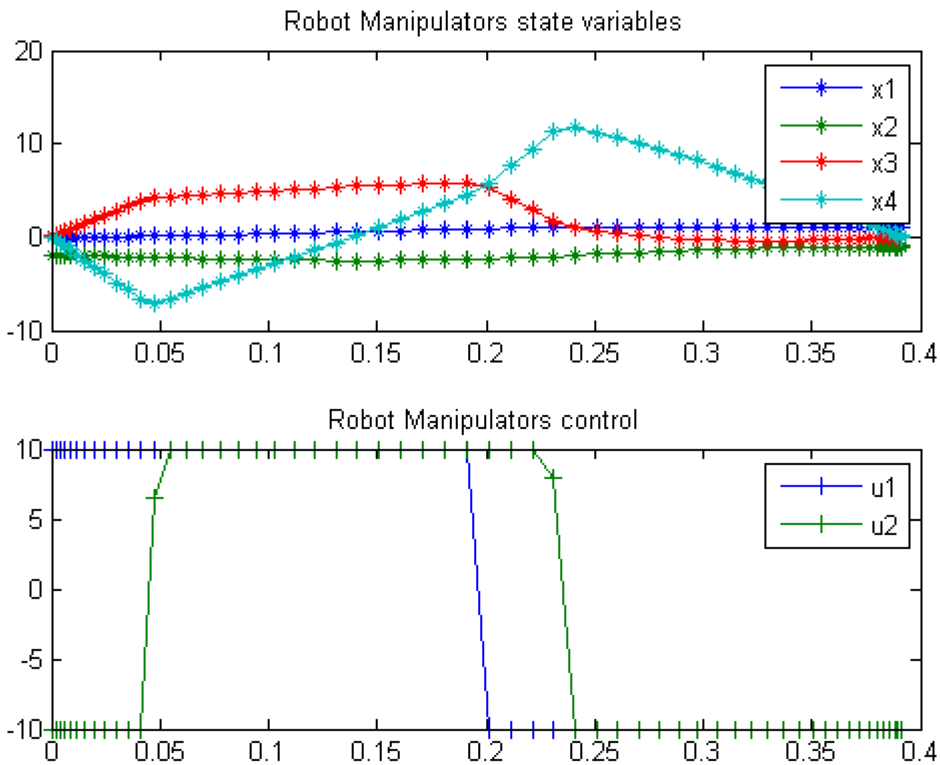
```
options = struct;
options.name = 'Robot Manipulators';
solution = ezsolve(objective, {cbox, cbnd, ceq}, x0, options);

% Optimal x, y, and speed, to use as starting guess
% in the next iteration
tfopt = subs(final(t), solution);
x1opt = subs(x1, solution);
x2opt = subs(x2, solution);
x3opt = subs(x3, solution);
x4opt = subs(x4, solution);
u1opt = subs(u1, solution);
u2opt = subs(u2, solution);
```

```
Problem type appears to be: lpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Robot Manipulators             f_k       0.391698237386178260
                                       sum(|constr|)      0.000063099634834817
                              f(x_k) + sum(|constr|)      0.391761337021013070
                                              f(x_0)      7.000000000000000000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv   68 ConJacEv   68 Iter   26 MinorIter  437
CPU time: 0.421875 sec. Elapsed time: 0.437000 sec.


Problem type appears to be: lpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Robot Manipulators             f_k       0.391820155673056890
                                       sum(|constr|)      0.000000000017635038
                              f(x_k) + sum(|constr|)      0.391820155690691950
                                              f(x_0)      0.391698237386178260

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied
```

```
FuncEv    1 ConstrEv   16 ConJacEv   16 Iter   12 MinorIter  440
CPU time: 0.515625 sec. Elapsed time: 0.547000 sec.


end

t  = subs(collocate(t),solution);
x1 = subs(collocate(x1opt),solution);
x2 = subs(collocate(x2opt),solution);
x3 = subs(collocate(x3opt),solution);
x4 = subs(collocate(x4opt),solution);
u1 = subs(collocate(u1opt),solution);
u2 = subs(collocate(u2opt),solution);
```

94.5 Plot result

```
subplot(2,1,1)
plot(t,x1,'*-',t,x2,'*-',t,x3,'*-',t,x4,'*-');
legend('x1','x2','x3','x4');
title('Robot Manipulators state variables');

subplot(2,1,2)
plot(t,u1,'+-',t,u2,'+-');
legend('u1','u2');
title('Robot Manipulators control');
```


Robot Manipulators state variables
x1
x2
x3
x4
20
10
0
-10
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
Robot Manipulators control
u1
u2
10
5
0
-5
-10
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4

95 Satellite Control

Users Guide for dyn.Opt, Example 7a, 7b, 7c

95.1 A satellite control problem

Find T (controls) over t in [0; 100] to minimize

7c is free end time

7a:

$$J = \int_0^{100} ((T1^2 + T2^2 + T3^2) * 0.5) \mathrm{d}t +$$
$$w_1^2 + w_2^2 + w_3^2 + (e_1 - 0.70106)^2 + (e_2 - 0.0923)^2 + (e_3 - 0.56098)^2 + (e_4 - 0.43047)^2$$

7b:

$$J = \int_0^{100} ((T1^2 + T2^2 + T3^2) * 0.5) \mathrm{d}t$$

7c:

$$J = t_F$$

subject to:

$$I1 = 1.0e6$$
$$I2 = 833333.0$$
$$I3 = 916667.0$$
$$T1S = 550$$
$$T2S = 50$$
$$T3S = 550$$

$$\frac{de_1}{dt} = 0.5*(w_1*e_4 - w_2*e_3 + w_3*e_2)$$
$$\frac{de_2}{dt} = 0.5*(w_1*e_3 + w_2*e_4 - w_3*e_1)$$
$$\frac{de_3}{dt} = 0.5*(-w_1*e_2 + w_2*e_1 + w_3*e_4)$$
$$\frac{de_4}{dt} = -0.5*(w_1*e_1 + w_2*e_2 + w_3*e_3)$$
$$\frac{dw_1}{dt} = \frac{(I2 - I3)*w_2*w_3 + T1*T1S}{I1}$$
$$\frac{dw_2}{dt} = \frac{(I3 - I1)*w3*w1 + T2*T2S}{I2}$$
$$\frac{dw_3}{dt} = \frac{(I1 - I2)*w1*w2 + T3*T3S}{I3}$$

$$e(0) = [0\ 0\ 0\ 1]$$
$$w(0) = [0.01\ 0.005\ 0.001]$$

7b, 7c - x(100) = [0.70106 0.0923 0.56098 NaN 0 0 0]; 7c - free time 7c - -1 <= T <= 1

Reference: [16]

95.2 Problem setup

```
toms t

% Starting guess
e1opt = 0;
e2opt = 0;
e3opt = 0;
e4opt = 0;
w1opt = 0;
w2opt = 0;
w3opt = 0;
T1opt = 0;
T2opt = 0;
T3opt = 0;

% Final times
```

```
tfs = zeros(3,1);

for i=1:3
    if i == 3
        toms t_f
        runs = [10 20 101];
    else
        runs = [10 40];
    end
    if i == 2
        e1opt = 0; e2opt = 0;
        e3opt = 0; e4opt = 0;
        w1opt = 0; w2opt = 0;
        w3opt = 0; T1opt = 0;
        T2opt = 0; T3opt = 0;
    end
    for n=runs

        if i == 3
            p = tomPhase('p', t, 0, t_f, n);
        else
            p = tomPhase('p', t, 0, 100, n);
        end

        setPhase(p);
        tomStates e1 e2 e3 e4 w1 w2 w3
        tomControls T1 T2 T3

        if i == 3
            x0 = {t_f == 100};
        else
            x0 = {};
        end
        % Initial guess
        x0 = {x0; collocate({T1 == T1opt
            T2 == T2opt; T3 == T3opt})
            icollocate({
            e1 == e1opt; e2 == e2opt
            e3 == e3opt; e4 == e4opt
            w1 == w1opt; w2 == w2opt
            w3 == w3opt})};

        if i == 3
            cbox = {
                1 <= t_f <= 1000
                -1 <= collocate(T1) <= 1
                -1 <= collocate(T2) <= 1
```

```
        -1 <= collocate(T3) <= 1};
else
    cbox = {};
end

% Boundary constraints
cbnd = initial({e1 == 0
    e2 == 0;     e3 == 0
    e4 == 1;     w1 == 0.01
    w2 == 0.005; w3 == 0.001});
```

95.3 Problem 7b and 7c modifications

```
if i ~= 1
    cbnd = {cbnd
        final({
        e1 == 0.70106
        e2 == 0.0923
        e3 == 0.56098
        w1 == 0
        w2 == 0
        w3 == 0
        })};
end

% ODEs and path constraints
I1 = 1.0e6;
I2 = 833333.0;
I3 = 916667.0;
T1Sc = 550;
T2Sc = 50;
T3Sc = 550;
ceq = collocate({
    dot(e1) == 0.5*(w1.*e4-w2.*e3+w3.*e2)
    dot(e2) == 0.5*(w1.*e3+w2.*e4-w3.*e1)
    dot(e3) == 0.5*(-w1.*e2+w2.*e1+w3.*e4)
    dot(e4) == -0.5*(w1.*e1+w2.*e2+w3.*e3)
    dot(w1) == ((I2-I3)*w2.*w3+T1*T1Sc)/I1
    dot(w2) == ((I3-I1)*w3.*w1+T2*T2Sc)/I2
    dot(w3) == ((I1-I2)*w1.*w2+T3*T3Sc)/I3});

% Objective
if i == 1
    objective = final(w1)^2 + final(w2)^2 + final(w3)^2 +...
        (final(e1) - 0.70106)^2 + (final(e2) - 0.0923)^2 + ...
        (final(e3) - 0.56098)^2 + (final(e4) - 0.43047)^2 ...
        + integrate((T1.^2+T2.^2+T3.^2)*0.5);
```

```
    elseif i == 2
        objective = integrate((T1.^2+T2.^2+T3.^2)*0.5);
    else
        objective = t_f;
    end
```

95.4 Solve the problem

```
    options = struct;
    if i == 1
        options.name = 'Satellite Control 7a';
    elseif i == 2
        options.name = 'Satellite Control 7b';
    else
        options.name = 'Satellite Control 7c';
    end
    solution = ezsolve(objective, {cbox, cbnd, ceq}, x0, options);

    e1opt = subs(e1, solution);
    e2opt = subs(e2, solution);
    e3opt = subs(e3, solution);
    e4opt = subs(e4, solution);
    w1opt = subs(w1, solution);
    w2opt = subs(w2, solution);
    w3opt = subs(w3, solution);
    T1opt = subs(T1, solution);
    T2opt = subs(T2, solution);
    T3opt = subs(T3, solution);
```

```
Problem type appears to be: qpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Satellite Control 7a            f_k       0.463944669252504550
                                       sum(|constr|)      0.000000135115457319
                              f(x_k) + sum(|constr|)      0.463944804367961870
                                              f(x_0)      0.139185999999999230

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv   27 ConJacEv   27 Iter   15 MinorIter  105
CPU time: 0.140625 sec. Elapsed time: 0.140000 sec.
```

```
Problem type appears to be: qpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Satellite Control 7a           f_k       0.463944366974706650
                                       sum(|constr|)      0.000000000342695392
                              f(x_k) + sum(|constr|)      0.463944367317402020
                                              f(x_0)     -0.536077645550793180

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv    3 ConJacEv    3 Iter    2 MinorIter  255
CPU time: 0.218750 sec. Elapsed time: 0.250000 sec.


Problem type appears to be: qpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Satellite Control 7b           f_k      71.411903807059261000
                                       sum(|constr|)      0.000000066784327430
                              f(x_k) + sum(|constr|)     71.411903873843585000
                                              f(x_0)      0.000000000000000000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv   20 ConJacEv   20 Iter   17 MinorIter  205
CPU time: 0.140625 sec. Elapsed time: 0.141000 sec.


Problem type appears to be: qpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Satellite Control 7b           f_k      71.411593978634926000
                                       sum(|constr|)      0.000000015292225816
                              f(x_k) + sum(|constr|)     71.411593993927156000
                                              f(x_0)     71.236662445283258000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
```

```
Optimality conditions satisfied

FuncEv    1 ConstrEv    5 ConJacEv    5 Iter    3 MinorIter  346
CPU time: 0.421875 sec. Elapsed time: 0.422000 sec.


Problem type appears to be: lpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Satellite Control 7c           f_k      99.185331597188878000
                                       sum(|constr|)      0.000010903612751909
                              f(x_k) + sum(|constr|)     99.185342500801625000
                                              f(x_0)    100.000000000000000000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv    8 ConJacEv    8 Iter    6 MinorIter   83
CPU time: 0.078125 sec. Elapsed time: 0.078000 sec.


Problem type appears to be: lpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Satellite Control 7c           f_k      98.945461990937687000
                                       sum(|constr|)      0.000017743233884652
                              f(x_k) + sum(|constr|)     98.945479734171570000
                                              f(x_0)    100.000000000000000000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv    5 ConJacEv    5 Iter    4 MinorIter  154
CPU time: 0.093750 sec. Elapsed time: 0.109000 sec.


Problem type appears to be: lpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Satellite Control 7c           f_k      98.834204968061798000
```

```
                    sum(|constr|)      0.000012503587541902
          f(x_k) + sum(|constr|)      98.834217471649339000
                           f(x_0)    100.000000000000000000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv    7 ConJacEv    7 Iter    6 MinorIter  972
CPU time: 8.406250 sec. Elapsed time: 8.813000 sec.
```

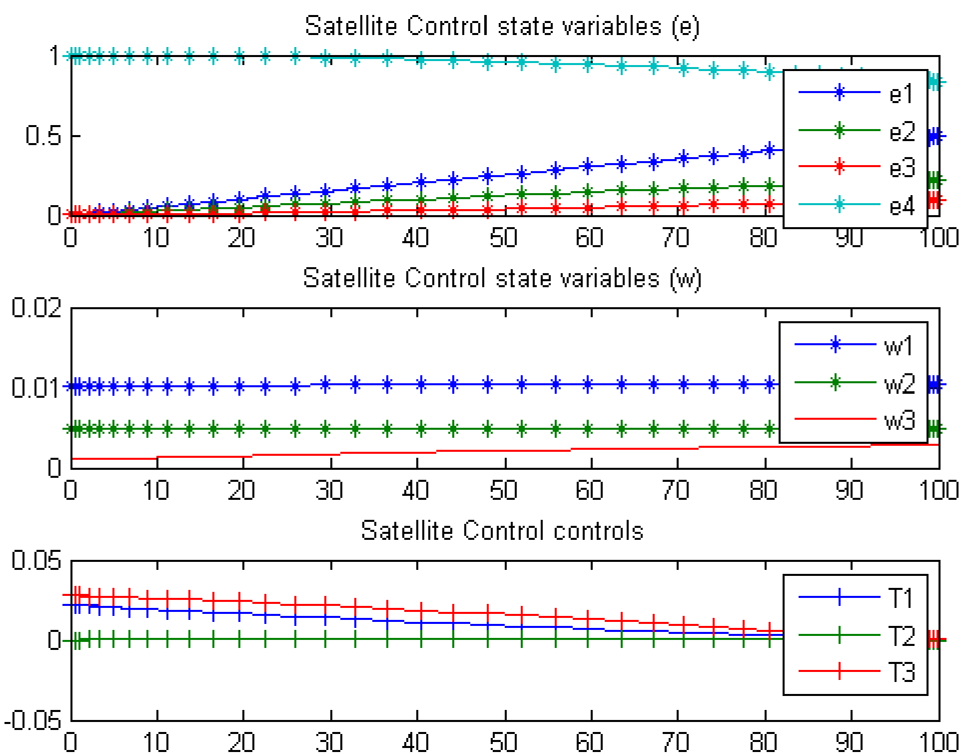
```
    end
    tfs(i) = subs(final(t),solution);
    % We only want to plot the solution from the first problem
    if i == 1
        tp  = subs(collocate(t),solution);
        e1p = subs(collocate(e1),solution);
        e2p = subs(collocate(e2),solution);
        e3p = subs(collocate(e3),solution);
        e4p = subs(collocate(e4),solution);
        w1p = subs(collocate(w1),solution);
        w2p = subs(collocate(w2),solution);
        w3p = subs(collocate(w3),solution);
        T1p = subs(collocate(T1),solution);
        T2p = subs(collocate(T2),solution);
        T3p = subs(collocate(T3),solution);
    end
end

disp(sprintf('\nFinal time for 7a = %1.4g',tfs(1)));
disp(sprintf('\nFinal time for 7b = %1.4g',tfs(2)));
disp(sprintf('\nFinal time for 7c = %1.4g',tfs(3)));
```

```
Final time for 7a = 100

Final time for 7b = 100

Final time for 7c = 98.83
```

95.5 Plot result

```
subplot(3,1,1)
plot(tp,e1p,'*-',tp,e2p,'*-',tp,e3p,'*-',tp,e4p,'*-');
legend('e1','e2','e3','e4');
title('Satellite Control state variables (e)');
```

```
subplot(3,1,2)
plot(tp,w1p,'*-',tp,w2p,'*-',tp,w3p);
legend('w1','w2','w3');
title('Satellite Control state variables (w)');

subplot(3,1,3)
plot(tp,T1p,'+-',tp,T2p,'+-',tp,T3p,'+-');
legend('T1','T2','T3');
title('Satellite Control controls');
```



96 Second Order System

Users Guide for dyn.Opt, Example 1

Optimal control of a second order system

End time says 1 in problem text.

96.1 Problem Formulation

Find u over t in [0; 2] to minimize

$$J = \int_0^2 u^2/2 \mathrm{d}t$$

subject to:

$$\frac{dx_1}{dt} = x_2$$
$$\frac{dx_2}{dt} = u$$
$$x_1(0) = 1$$
$$x_1(2) = 0$$
$$x_2(0) = 1$$
$$x_2(2) = 0$$
$$-100 <= u <= 100$$

Reference: [16]

96.2 Problem setup

```
toms t
p = tomPhase('p', t, 0, 2, 30);
setPhase(p);

tomStates x1 x2
tomControls u
```

```
% Initial guess
x0 = {icollocate({x1 == 1-t/2; x2 == -1+t/2})
    collocate(u == -3.5+6*t/2)};

% Box constraints
cbox = {-100 <= icollocate(x1) <= 100
    -100 <= icollocate(x2) <= 100
    -100 <= collocate(u)   <= 100};

% Boundary constraints
cbnd = {initial({x1 == 1; x2 == 1})
    final({x1 == 0; x2 == 0})};

% ODEs and path constraints
ceq = collocate({dot(x1) == x2; dot(x2) == u});

% Objective
objective = integrate(u.^2/2);
```

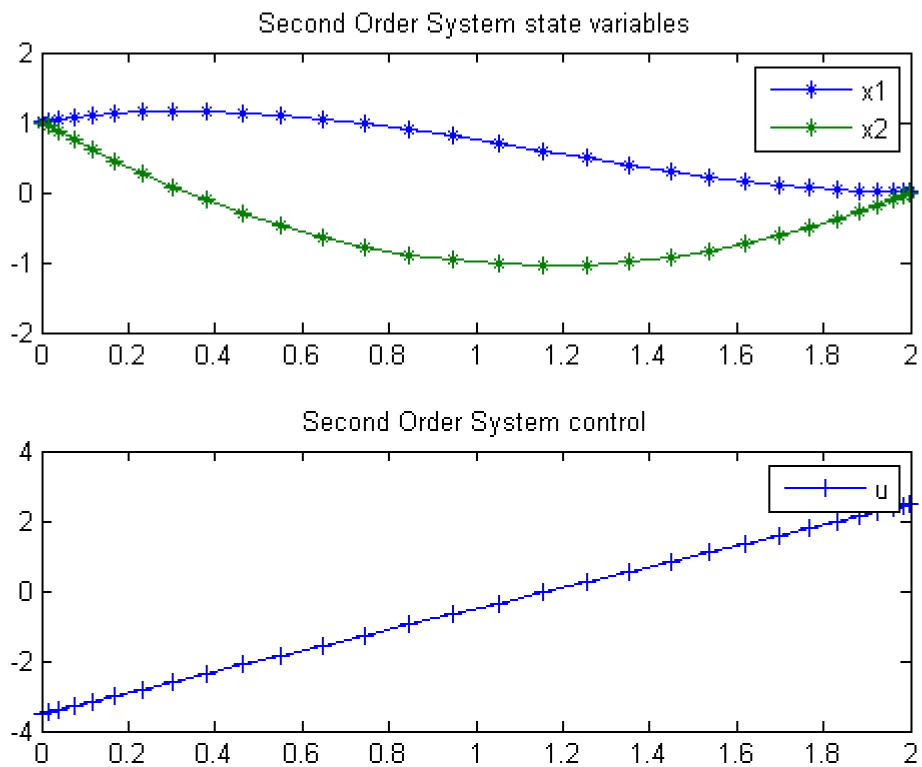
96.3 Solve the problem

```
options = struct;
options.name = 'Second Order System';
solution = ezsolve(objective, {cbox, cbnd, ceq}, x0, options);
t  = subs(collocate(t),solution);
x1 = subs(collocate(x1),solution);
x2 = subs(collocate(x2),solution);
u  = subs(collocate(u),solution);
```

```
Problem type appears to be: qp
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem:  1: Second Order System                f_k       3.249999999996386900
                                     sum(|constr|)      0.000000000432709878
                            f(x_k) + sum(|constr|)      3.250000000429096800
                                            f(x_0)      0.000000000000000000

Solver: CPLEX.  EXIT=0.  INFORM=1.
CPLEX Barrier QP solver
Optimal solution found

FuncEv    6 GradEv    6 ConstrEv    6 Iter    6
CPU time: 0.015625 sec. Elapsed time: 0.016000 sec.
```

96.4 Plot result

```
subplot(2,1,1)
plot(t,x1,'*-',t,x2,'*-');
legend('x1','x2');
title('Second Order System state variables');

subplot(2,1,2)
plot(t,u,'+-');
legend('u');
title('Second Order System control');
```



97 Space Shuttle Reentry

SOCS 6.5.0 Manual

7.4.4 Maximum Crossrange Space Shuttle Reentry Problem.

97.1 Problem Formulation

Find u over t in [0; t] to maximize

$$J = lat$$

subject to:

The equations given in the code below.

Reference: [5]

97.2 Problem setup

```
toms t t_f

% Scaled time
p1 = tomPhase('p1', t, 0, t_f, 30);
setPhase(p1);

tomStates alt long lat vel ggamma azi
tomControls aalpha bbeta

% Constants
tGuess = 2000;
tMax   = 4000;
tMin   = 100;
cr2d    = 180/pi;
betalim = 90;
weight  = 203000;
cm2w    = 32.174;
cea     = 20902900;
mmu      = 0.14076539e17;
rho0    = 0.002378;
href    = 23800;
```

```
cl0     = -0.20704;
cl1     = 0.029244;
cd0     = 0.07854;
cd1     = -6.1592e-3;
cd2     = 6.21408e-4;
sref    = 2690;

alt0 = 260000;
altf = 80000;

vel0 = 25600;
velf = 2500;

% Initial guess
x0 = {
    t_f == 1000
    icollocate({
    alt   == alt0-(alt0-altf)*t/t_f
    long  == -0.5*90/cr2d
    lat   == -89/cr2d
    vel   == vel0-(vel0-velf)*t/t_f
    ggamma == -1/cr2d-4/cr2d*t/t_f
    azi   == pi/2-pi*t/t_f
    })
    collocate({
    aalpha == 0
    bbeta  == 1/cr2d
    })
    };

% Boundary constraints
cbnd = {
    initial({
    alt   == alt0
    long  == -0.5*75.3153/cr2d
    lat   == 0
    vel   == 25600
    ggamma == -1/cr2d
    azi   == 90/cr2d
    aalpha == 17/cr2d
    bbeta  == -betalim/cr2d
    })
    final({
    alt   == altf
    vel   == velf
    ggamma == -5/cr2d
    })};
```

```
% Box constraints
cbox = {
    100 <= t_f <= 5000
    0              <= icollocate(alt)    <= 300000
    -0.5*90/cr2d   <= icollocate(long)   <= 0.5*90/cr2d
    -89/cr2d       <= icollocate(lat)    <= 89/cr2d
    1000           <= icollocate(vel)    <= 40000
    -89/cr2d       <= icollocate(ggamma) <= 89/cr2d
    -pi            <= icollocate(azi)    <= pi
    -89/cr2d       <= collocate(aalpha)  <= 89/cr2d
    -betalim/cr2d  <= collocate(bbeta)   <= 1/cr2d
    };

mass   = weight/cm2w;
alphad = cr2d*aalpha;
radius = cea+alt;
grav   = mmu./radius.^2;

rhodns = rho0*exp(-alt/href);
dynp   = 0.5*rhodns.*vel.^2;
subl   = cl0+cl1*alphad;
subd   = cd0+cd1+cd2*alphad.*alphad;
drag   = dynp.*subd*sref;
lift   = dynp.*subl*sref;
vrelg  = vel./radius-grav./vel;

% ODEs and path constraints
ceq = collocate({
    dot(alt) == vel.*sin(ggamma)
    dot(long) == vel.*cos(ggamma).*sin(azi)./(radius.*cos(lat))
    dot(lat) == vel.*cos(ggamma).*cos(azi)./radius
    dot(vel) == -drag./mass-grav.*sin(ggamma)
    dot(ggamma) == lift.*cos(bbeta)./(mass.*vel)+cos(ggamma).*vrelg
    dot(azi) == lift.*sin(bbeta)./(mass.*vel.*cos(ggamma))+...
    vel.*cos(ggamma).*sin(azi).*sin(lat)./(radius.*cos(lat))
    });

% Objective
objective = -final(lat)*180/pi;
```

97.3 Solve the problem

```
options = struct;
options.name = 'Shuttle Reentry';
options.Prob.SOL.optPar(30) = 100000;
options.scale = 'auto';
```

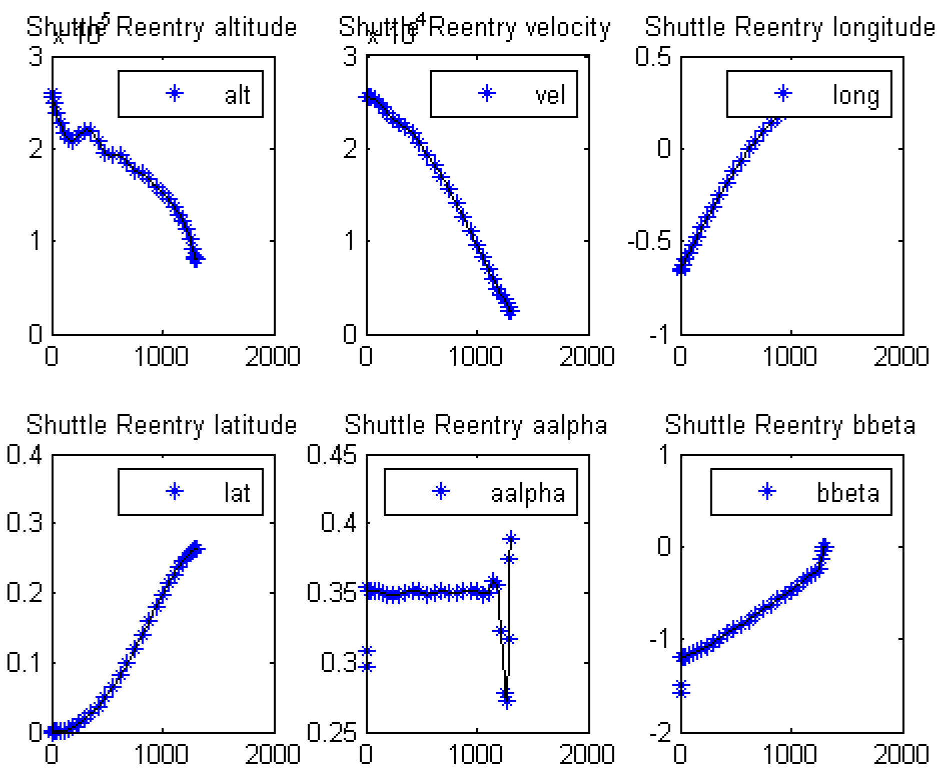
```
solution = ezsolve(objective, {cbox, cbnd, ceq}, x0, options);
```

```
Problem type appears to be: lpcon
Auto-scaling
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Shuttle Reentry                f_k     -15.142919504993619000
                                       sum(|constr|)      0.000000001777377229
                              f(x_k) + sum(|constr|)    -15.142919503216243000
                                              f(x_0)     -0.000000000000795808

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv 1366 ConJacEv 1364 Iter  207 MinorIter 6563
CPU time: 26.031250 sec. Elapsed time: 14.609000 sec.
```

97.4 Plot result

```
subplot(2,3,1)
ezplot(alt)
legend('alt');
title('Shuttle Reentry altitude');

subplot(2,3,2)
ezplot(vel)
legend('vel');
title('Shuttle Reentry velocity');

subplot(2,3,3)
ezplot(long)
legend('long');
title('Shuttle Reentry longitude');

subplot(2,3,4)
ezplot(lat)
legend('lat');
title('Shuttle Reentry latitude');

subplot(2,3,5)
ezplot(aalpha)
legend('aalpha');
```

```
title('Shuttle Reentry aalpha');

subplot(2,3,6)
ezplot(bbeta)
legend('bbeta');
title('Shuttle Reentry bbeta');
```



98 Simple Bang Bang Problem

Function Space Complementarity Methods for Optimal Control Problems, Dissertation, Martin Weiser

98.1 Problem Description

Find u over t in [-0.5; 0.5] to minimize:

$$J = \int_{-\frac{1}{2}}^{\frac{1}{2}} t * u \mathrm{d}t$$

subject to:

$$|u| <= 1$$

Reference: [34]

98.2 Problem setup

```
toms t
p = tomPhase('p', t, -0.5, 1, 20);
setPhase(p);
tomStates x
tomControls u

% Initial guess
x0 = {collocate(u == 1-2*(t+0.5))
    icollocate(x == 1-2*(t+0.5))};

% Box constraints
cbox = {-1 <= icollocate(x) <= 1
    -1 <= collocate(u)  <= 1};

% ODEs and path constraints
ceq = collocate(dot(x) == 0);

% Objective
objective = integrate(t.*u);
```

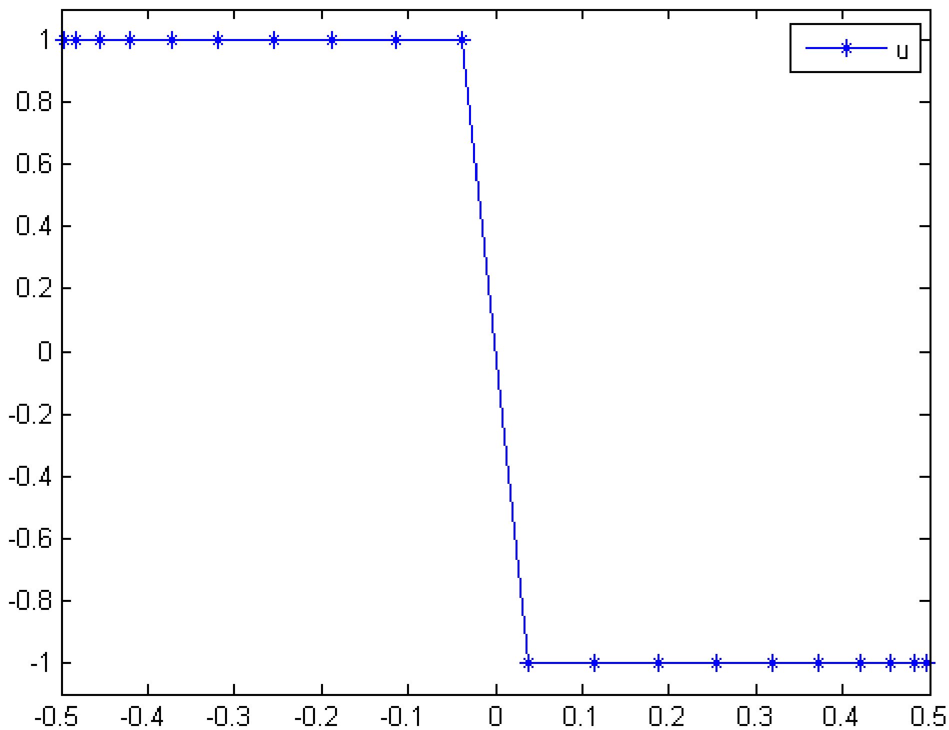
98.3 Solve the problem

```
options = struct;
options.name = 'Simple Bang Bang Problem';
solution = ezsolve(objective, {cbox, ceq}, x0, options);
t = subs(collocate(t),solution);
u = subs(collocate(u),solution);
```

```
Problem type appears to be: lp
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Simple Bang Bang Problem       f_k      -0.250490325030179710
                                       sum(|constr|)      0.000000000000810402
                              f(x_k) + sum(|constr|)     -0.250490325029369300
                                              f(x_0)      0.000000000000000000

Solver: CPLEX.  EXIT=0.  INFORM=1.
CPLEX Dual Simplex LP solver
Optimal solution found
```

98.4 Plot result

```
figure(1);
plot(t,u,'*-');
legend('u');
ylim([-1.1,1.1]);
```


u
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-0.5
-0.4
-0.3
-0.2
-0.1
0
0.1
0.2
0.3
0.4
0.5

99 Singular Arc Problem

Problem 3: Miser3 manual

99.1 Problem Formulation

Find u(t) over t in [0; t_f] to minimize

$$J = t_f$$

subject to:

$$\frac{dx_1}{dt} = u$$
$$\frac{dx_2}{dt} = cos(x_1)$$
$$\frac{dx_3}{dt} = sin(x_1)$$
$$x_2(t_f) = x_3(t_f) = 0$$
$$|u| <= 2$$
$$x(0) = [\frac{pi}{2} \ 4 \ 0]$$

Reference: [19]

99.2 Problem setup

```
toms t
toms t_f
p = tomPhase('p', t, 0, t_f, 60);
setPhase(p);

tomStates x1 x2 x3
tomControls u

% Initial guess
x0 = {t_f == 20
    icollocate({
    x1 == pi/2+pi/2*t/t_f
```

```
    x2 == 4-4*t/t_f; x3 == 0})
    collocate(u == 0)};

% Box constraints
cbox = {2 <= t_f <= 1000
    -2 <= collocate(u) <= 2};

% Boundary constraints
cbnd = {initial({x1 == pi/2; x2 == 4; x3 == 0})
    final({x2 == 0; x3 == 0})};

% ODEs and path constraints
ceq = collocate({dot(x1) == u
    dot(x2) == cos(x1); dot(x3) == sin(x1)});

% Objective
objective = t_f;
```

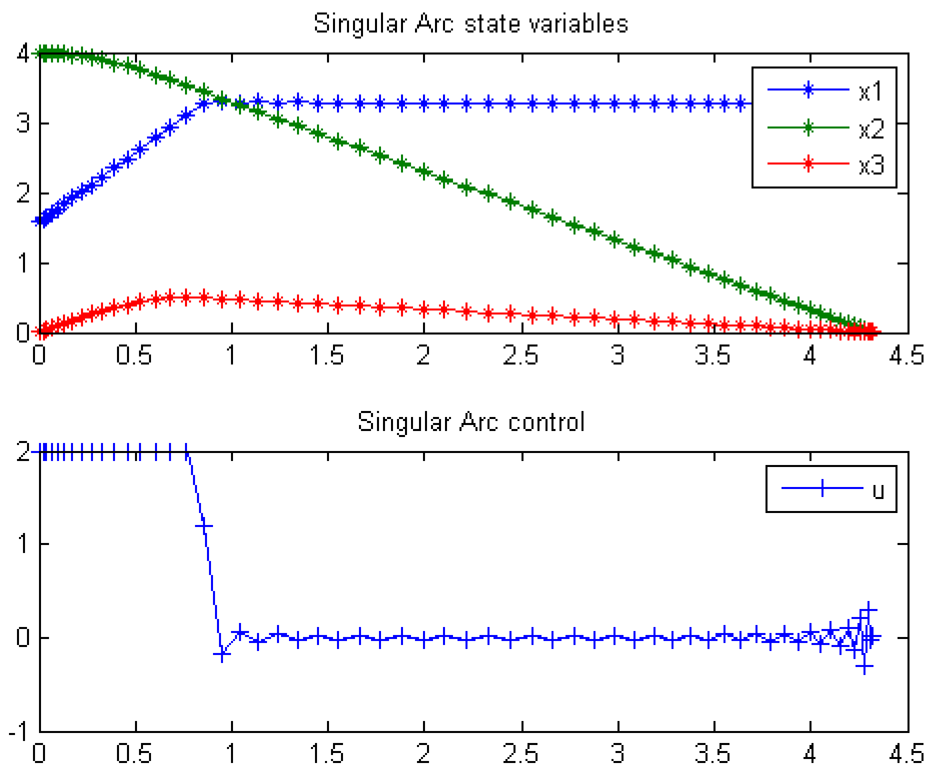
99.3 Solve the problem

```
options = struct;
options.name = 'Singular Arc';
solution = ezsolve(objective, {cbox, cbnd, ceq}, x0, options);
t  = subs(collocate(t),solution);
x1 = subs(collocate(x1),solution);
x2 = subs(collocate(x2),solution);
x3 = subs(collocate(x3),solution);
u  = subs(collocate(u),solution);
```

```
Problem type appears to be: lpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Singular Arc                   f_k       4.321198387073171600
                                       sum(|constr|)      0.000000179336713690
                              f(x_k) + sum(|constr|)      4.321198566409885100
                                              f(x_0)     20.000000000000000000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv   77 ConJacEv   77 Iter   70 MinorIter  377
CPU time: 1.234375 sec. Elapsed time: 1.281000 sec.
```

99.4 Plot result

```
subplot(2,1,1)
plot(t,x1,'*-',t,x2,'*-',t,x3,'*-');
legend('x1','x2','x3');
title('Singular Arc state variables');

subplot(2,1,2)
plot(t,u,'+-');
legend('u');
title('Singular Arc control');
```



100 Singular CSTR

ITERATIVE DYNAMIC PROGRAMMING, REIN LUUS

10.4 Nonlinear two-stage CSTR problem

CHAPMAN & HALL/CRC Monographs and Surveys in Pure and Applied Mathematics

100.1 Problem Formulation

Find u over t in [0; t_F] to minimize:

$$J = x(t_F)' * x(t_F) + t_F$$

(the state variables are moved to bounds)

subject to:

$$\frac{dx_1}{dt} = -3 * x_1 + g_1$$
$$\frac{dx_2}{dt} = -11.1558 * x_2 + g_1 - 8.1558 * (x_2 + 0.1592) * u_1$$
$$\frac{dx_3}{dt} = 1.5 * (0.5 * x_1 - x_3) + g_2$$
$$\frac{dx_4}{dt} = 0.75 * x_2 - 4.9385 * x_4 + g_2 - 3.4385 * (x_4 + 0.122) * u_2$$

$$g_1 = 1.5e7 * (0.5251 - x_1) * exp(-\frac{10}{x_2 + 0.6932}) -$$
$$1.5e10 * (0.4748 + x_1) * exp(-\frac{15}{x_2 + 0.6932}) - 1.4280$$
$$g_2 = 1.5e7 * (0.4236 - x_2) * exp(-\frac{10}{x_4 + 0.6560}) -$$
$$1.5e10 * (0.5764 + x_3) * exp(-\frac{15}{x4 + 0.6560}) - 0.5086$$

The initial condition are:

$$x(0) = [0.1962 \ -0.0372 \ 0.0946 \ 0]$$

$$-1 <= u(1:2) <= 1$$

Reference: [25]

100.2 Problem setup

```
toms t t_f
p = tomPhase('p', t, 0, t_f, 30);
setPhase(p)

tomStates x1 x2 x3 x4
tomControls u1 u2

% Initial guess
x0 = {t_f == 0.3
    icollocate({x1 == 0.1962; x2 == -0.0372
    x3 == 0.0946; x4 == 0})
    collocate({u1 == 0; u2 == 0})};

% Box constraints
cbox = {0.1 <= t_f <= 100
    -1 <= collocate(u1) <= 1
    -1 <= collocate(u2) <= 1};

% Boundary constraints
cbnd = {initial({x1 == 0.1962; x2 == -0.0372
    x3 == 0.0946; x4 == 0})
    final({x1 == 0; x2 == 0
    x3 == 0; x4 == 0})};

% ODEs and path constraints
g1 = 1.5e7*(0.5251-x1).*exp(-10./(x2+0.6932)) ...
    - 1.5e10*(0.4748+x1).*exp(-15./(x2+0.6932)) - 1.4280;
g2 = 1.5e7*(0.4236-x2).*exp(-10./(x4+0.6560)) ...
    - 1.5e10*(0.5764+x3).*exp(-15./(x4+0.6560)) - 0.5086;

ceq = collocate({
    dot(x1) == -3*x1+g1
    dot(x2) == -11.1558*x2+g1-8.1558*(x2+0.1592).*u1
    dot(x3) == 1.5*(0.5*x1-x3)+g2
    dot(x4) == 0.75*x2-4.9385*x4+g2-3.4385*(x4+0.122).*u2});

% Objective
```

```
objective = t_f;
```

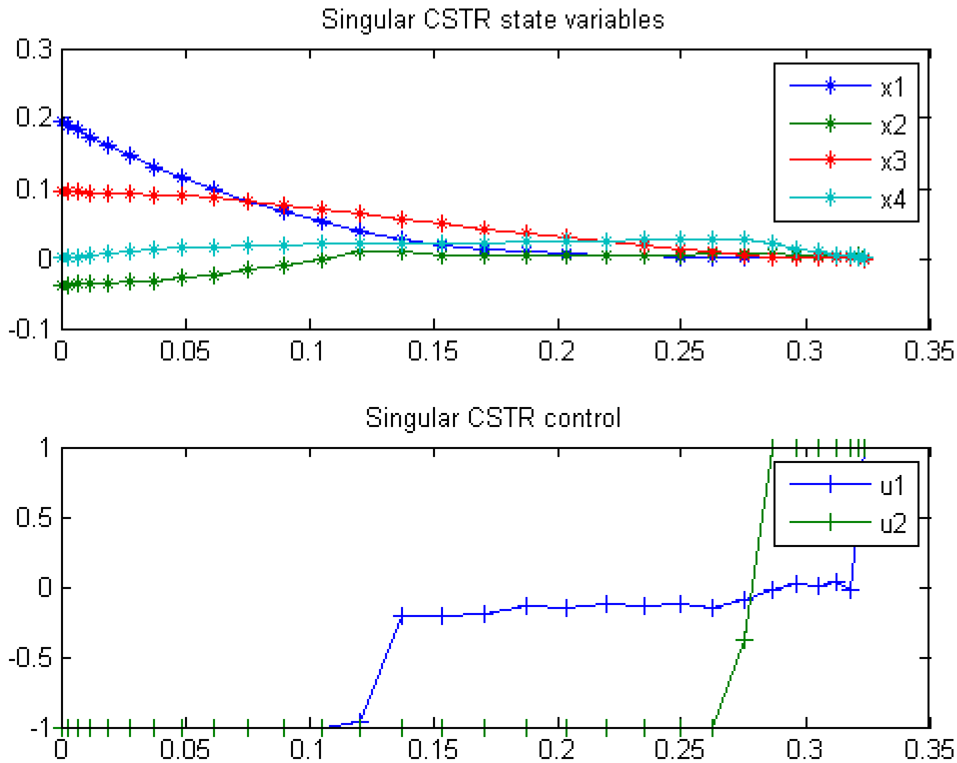
100.3 Solve the problem

```
options = struct;
options.name = 'Singular CSTR';
solution = ezsolve(objective, {cbox, cbnd, ceq}, x0, options);
t  = subs(collocate(t),solution);
x1 = subs(collocate(x1),solution);
x2 = subs(collocate(x2),solution);
x3 = subs(collocate(x3),solution);
x4 = subs(collocate(x4),solution);
u1 = subs(collocate(u1),solution);
u2 = subs(collocate(u2),solution);
```

```
Problem type appears to be: lpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Singular CSTR                  f_k      0.324402684069356410
                                       sum(|constr|)      0.000000010809098017
                              f(x_k) + sum(|constr|)      0.324402694878454410
                                              f(x_0)      0.299999999999999990

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv   75 ConJacEv   75 Iter   42 MinorIter  427
CPU time: 0.562500 sec. Elapsed time: 0.594000 sec.
```

100.4 Plot result

```
subplot(2,1,1)
plot(t,x1,'*-',t,x2,'*-',t,x3,'*-',t,x4,'*-');
legend('x1','x2','x3','x4');
title('Singular CSTR state variables');

subplot(2,1,2)
plot(t,u1,'+-',t,u2,'+-');
legend('u1','u2');
title('Singular CSTR control');
```


Singular CSTR state variables
x1
x2
x3
x4
0.3
0.2
0.1
0
-0.1
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
Singular CSTR control
u1
u2
1
0.5
0
-0.5
-1

101 Singular Control 1

ITERATIVE DYNAMIC PROGRAMMING, REIN LUUS

10.2.1 Example 1

CHAPMAN & HALL/CRC Monographs and Surveys in Pure and Applied Mathematics

101.1 Problem Formulation

Find u over t in [0; 2] to minimize

$$J = x_2(t_F)$$

subject to:

$$\frac{dx_1}{dt} = u$$
$$\frac{dx_2}{dt} = 0.5 * x_1^2$$

The initial condition are:

$$x(0) = [1\ 0]$$
$$-1 <= u <= 1$$

Reference: [25]

101.2 Problem setup

```
toms t
p = tomPhase('p', t, 0, 2, 50);
setPhase(p);

tomStates x1 x2
tomControls u
```

```
% Initial guess
x0 = {icollocate({x1 == 1; x2 == 0})
    collocate(u == 0)};

% Box constraints
cbox = {-1 <= collocate(u) <= 1};

% Boundary constraints
cbnd = initial({x1 == 1; x2 == 0});

% ODEs and path constraints
ceq = collocate({dot(x1) == u
    dot(x2) == 0.5*x1.^2});

% Objective
objective = final(x2);
```

101.3 Solve the problem

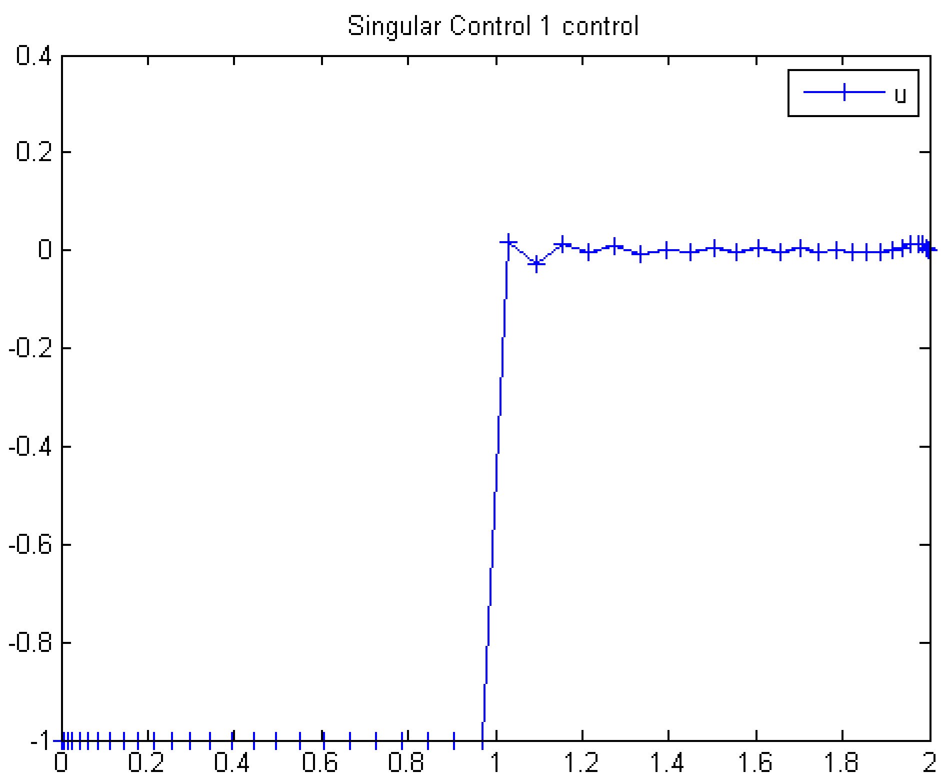
```
options = struct;
options.name = 'Singular Control 1';
solution = ezsolve(objective, {cbox, cbnd, ceq}, x0, options);
t = subs(collocate(t),solution);
u = subs(collocate(u),solution);
```

```
Problem type appears to be: lpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Singular Control 1             f_k       0.166665695130345510
                                       sum(|constr|)      0.000000330654862346
                              f(x_k) + sum(|constr|)      0.166666025785207870
                                              f(x_0)      0.000000000000000000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv   41 ConJacEv   41 Iter   39 MinorIter  164
CPU time: 0.218750 sec. Elapsed time: 0.219000 sec.
```

101.4 Plot result

```
figure(1)
```

```
plot(t,u,'+-');
legend('u');
title('Singular Control 1 control');
```



102 Singular Control 2

ITERATIVE DYNAMIC PROGRAMMING, REIN LUUS

10.2.2 Example 2

CHAPMAN & HALL/CRC Monographs and Surveys in Pure and Applied Mathematics

102.1 Problem Formulation

Find u over t in [0; 5] to minimize

$$J = x_3(t_F)$$

subject to:

$$\frac{dx_1}{dt} = x_2$$
$$\frac{dx_2}{dt} = u$$
$$\frac{dx_3}{dt} = x_1^2$$

The initial condition are:

$$x(0) = [0\ 1\ 0]$$
$$-1 <= u <= 1$$

Reference: [25]

102.2 Problem setup

```
toms t
p = tomPhase('p', t, 0, 5, 50);
setPhase(p);
```

```
tomStates x1 x2 x3
tomControls u

% Initial guess
x0 = {icollocate({x1 == 0; x2 == 1; x3 == 0})
    collocate(u == 0)};

% Box constraints
cbox = {-1 <= collocate(u) <= 1};

% Boundary constraints
cbnd = initial({x1 == 0; x2 == 1; x3 == 0});

% ODEs and path constraints
ceq = collocate({dot(x1) == x2
    dot(x2) == u; dot(x3) == x1.^2});

% Objective
objective = final(x3);
```

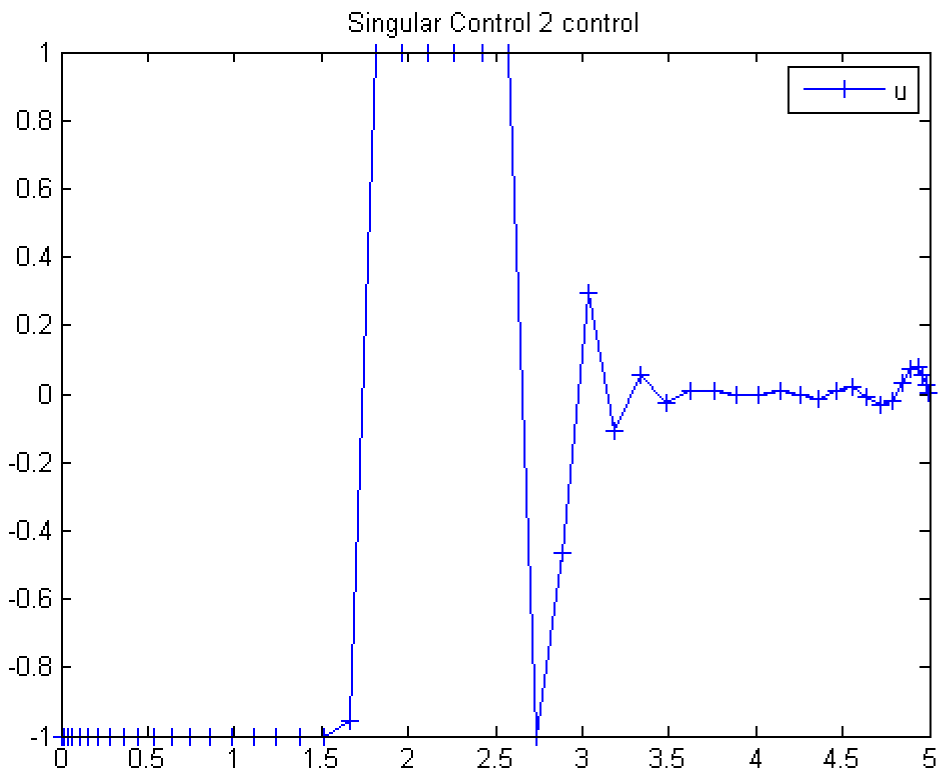
102.3 Solve the problem

```
options = struct;
options.name = 'Singular Control 2';
solution = ezsolve(objective, {cbox, cbnd, ceq}, x0, options);
t  = subs(collocate(t),solution);
u  = subs(collocate(u),solution);
```

```
Problem type appears to be: lpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Singular Control 2             f_k       0.268336059478540890
                                       sum(|constr|)      0.000000004483254193
                              f(x_k) + sum(|constr|)      0.268336063961795100
                                              f(x_0)      0.000000000000000000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv  107 ConJacEv  107 Iter  100 MinorIter  353
CPU time: 0.843750 sec. Elapsed time: 0.875000 sec.
```

102.4 Plot result

```
figure(1)
plot(t,u,'+-');
legend('u');
title('Singular Control 2 control');
```



103 Singular Control 3

ITERATIVE DYNAMIC PROGRAMMING, REIN LUUS

10.2.3 Example 3

CHAPMAN & HALL/CRC Monographs and Surveys in Pure and Applied Mathematics

103.1 Problem Formulation

Find u over t in [0; 5] to minimize

$$J = x_3(t_F)$$

subject to:

$$\frac{dx_1}{dt} = x_2$$
$$\frac{dx_2}{dt} = u$$
$$\frac{dx_3}{dt} = x_1^2 + x_2^2$$

The initial condition are:

$$x(0) = [0\ 1\ 0]$$
$$-1 <= u <= 1$$

Reference: [25]

103.2 Problem setup

```
toms t
p = tomPhase('p', t, 0, 5, 60);
setPhase(p);
```

```
tomStates x1 x2 x3
tomControls u

% Initial guess
x0 = {icollocate({x1 == 0; x2 == 1; x3 == 0})
    collocate(u == 0)};

% Box constraints
cbox = {-1 <= collocate(u) <= 1};

% Boundary constraints
cbnd = initial({x1 == 0; x2 == 1; x3 == 0});

% ODEs and path constraints
ceq = collocate({dot(x1) == x2
    dot(x2) == u; dot(x3) == x1.^2 + x2.^2});

% Objective
objective = final(x3);
```

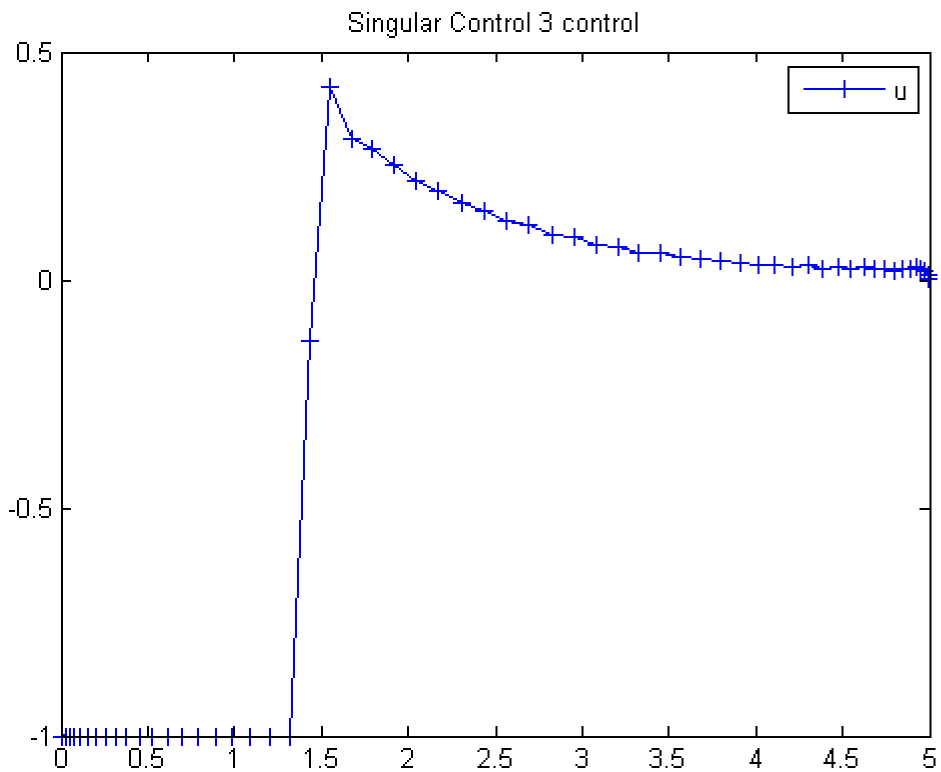
103.3 Solve the problem

```
options = struct;
options.name = 'Singular Control 3';
solution = ezsolve(objective, {cbox, cbnd, ceq}, x0, options);
t = subs(collocate(t),solution);
u = subs(collocate(u),solution);
```

```
Problem type appears to be: lpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Singular Control 3             f_k       0.753994561590098700
                                       sum(|constr|)      0.000000015978054113
                              f(x_k) + sum(|constr|)      0.753994577568152800
                                              f(x_0)      0.000000000000000000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv   43 ConJacEv   43 Iter   34 MinorIter  366
CPU time: 0.671875 sec. Elapsed time: 0.688000 sec.
```

103.4 Plot result

```
figure(1)
plot(t,u,'+-');
legend('u');
title('Singular Control 3 control');
```



104 Singular Control 4

ITERATIVE DYNAMIC PROGRAMMING, REIN LUUS

10.2.3 Example 4

CHAPMAN & HALL/CRC Monographs and Surveys in Pure and Applied Mathematics

104.1 Problem Formulation

Find u over t in [0; 5] to minimize

$$J = x_4(t_F)$$

subject to:

$$\frac{dx_1}{dt} = x_2$$
$$\frac{dx_2}{dt} = x_3$$
$$\frac{dx_3}{dt} = u$$
$$\frac{dx_4}{dt} = x_1^2$$

The initial condition are:

$$x(0) = [1\ 0\ 0\ 0]$$
$$-1 <= u <= 1$$

Reference: [25]

104.2 Problem setup

```
toms t
p = tomPhase('p', t, 0, 5, 100);
```

```
setPhase(p)

tomStates x1 x2 x3 x4
tomControls u

% Initial guess
x0 = {icollocate({x1 == 1; x2 == 0
    x3 == 0; x4 == 0})
    collocate(u == 0)};

% Box constraints
cbox = {-1 <= collocate(u) <= 1};

% Boundary constraints
cbnd = initial({x1 == 1; x2 == 0
    x3 == 0; x4 == 0});

% ODEs and path constraints
ceq = collocate({dot(x1) == x2; dot(x2) == x3
    dot(x3) == u; dot(x4) == x1.^2});

% Objective
objective = final(x4);
```

104.3 Solve the problem

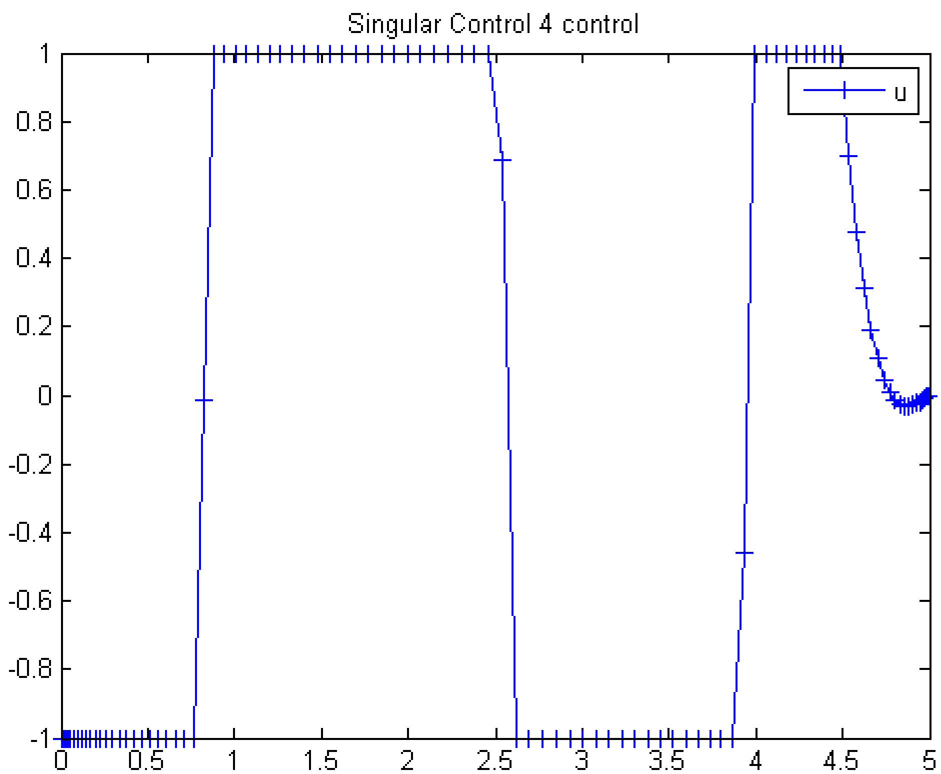
```
options = struct;
options.name = 'Singular Control 4';
solution = ezsolve(objective, {cbox, cbnd, ceq}, x0, options);
t = subs(collocate(t),solution);
u = subs(collocate(u),solution);
```

```
Problem type appears to be: lpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Singular Control 4             f_k       1.252389645383043400
                                       sum(|constr|)      0.000000063932046493
                              f(x_k) + sum(|constr|)      1.252389709315090000
                                              f(x_0)      0.000000000000000000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied
```

```
FuncEv    1 ConstrEv   92 ConJacEv   92 Iter   89 MinorIter  652
CPU time: 5.484375 sec. Elapsed time: 5.672000 sec.
```

104.4 Plot result

```
figure(1)
plot(t,u,'+-');
legend('u');
title('Singular Control 4 control');
```



105 Singular Control 5

ITERATIVE DYNAMIC PROGRAMMING, REIN LUUS

10.3 Yeo's singular control problem

CHAPMAN & HALL/CRC Monographs and Surveys in Pure and Applied Mathematics

105.1 Problem Formulation

Find u over t in [0; 1] to minimize:

$$J = x_4(t_F)$$

subject to:

$$\frac{dx_1}{dt} = x_2$$
$$\frac{dx_2}{dt} = -x_3 * u + 16 * x_5 - 8$$
$$\frac{dx_3}{dt} = u$$
$$\frac{dx_4}{dt} = x_1^2 + x_2^2 + 0.0005 * (x_2 + 16 * x_5 - 8 - 0.1 * x_3 * u^2)^2$$
$$\frac{dx_5}{dt} = 1$$

The initial condition are:

$$x(0) = [0 \ -1 \ -sqrt(5) \ 0 \ 0]$$
$$-4 <= u <= 10$$

The state x4 is implemented as a cost directly. x4 in the implementation is x5. u has a low limit of 9 in the code.

Reference: [25]

105.2 Problem setup

```
toms t
p = tomPhase('p', t, 0, 1, 80);
setPhase(p)

tomStates x1 x2 x3 x4
tomControls u

% Initial guess
x0 = {icollocate({x1 == 0; x2 == -1
    x3 == -sqrt(5); x4 == 0})
    collocate(u == 3)};

% Box constraints
cbox = {0 <= collocate(u) <= 10};

% Boundary constraints
cbnd = initial({x1 == 0; x2 == -1
    x3 == -sqrt(5); x4 == 0});

% ODEs and path constraints
ceq = collocate({dot(x1) == x2
    dot(x2) == -x3.*u + 16*x4 - 8
    dot(x3) == u; dot(x4) == 1});

% Objective
objective = integrate(x1.^2 + x2.^2 + ...
    0.0005*(x2+16*x4-8-0.1*x3.*u.^2).^2);
```

105.3 Solve the problem

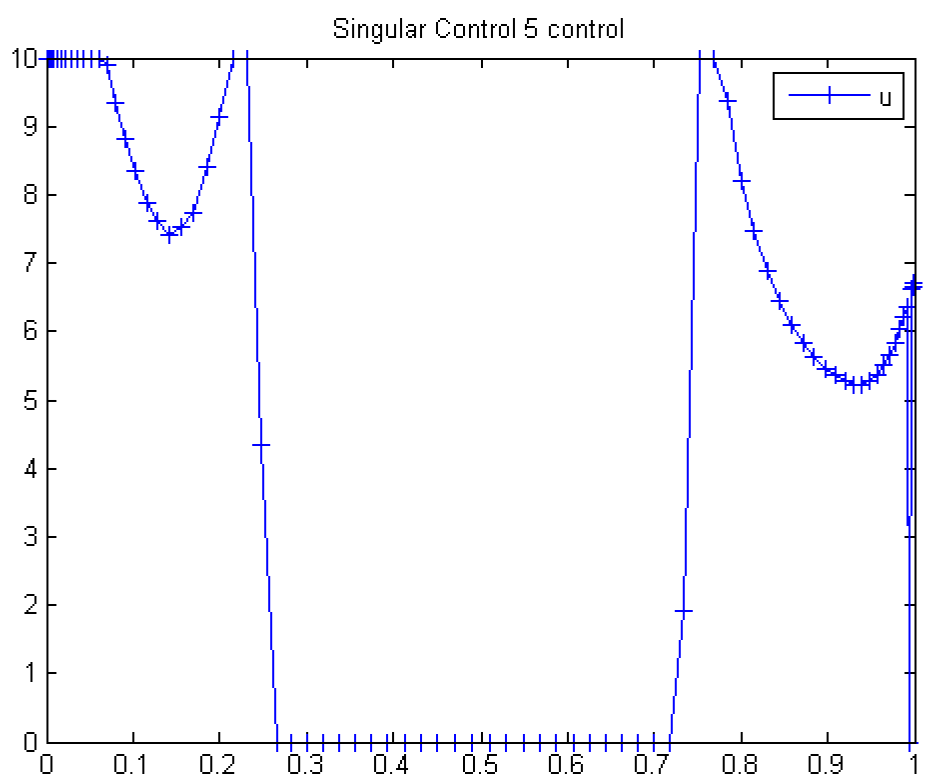
```
options = struct;
options.name = 'Singular Control 5';
solution = ezsolve(objective, {cbox, cbnd, ceq}, x0, options);
t = subs(collocate(t),solution);
u = subs(collocate(u),solution);
```

```
Problem type appears to be: con
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Singular Control 5             f_k       0.119318612949793070
                                       sum(|constr|)      0.000000070783551966
                               f(x_k) + sum(|constr|)      0.119318683733345030
```

```
                                               f(x_0)      1.024412849382205600

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv  372 GradEv  370 ConstrEv  370 ConJacEv  370 Iter  346 MinorIter  939
CPU time: 10.265625 sec. Elapsed time: 10.609000 sec.
```

105.4 Plot result

```
figure(1)
plot(t,u,'+-');
legend('u');
title('Singular Control 5 control');
```



106 Singular Control 6

Viscocity Solutions of Hamilton-Jacobi Equations and Optimal Control Problems. Alberto Bressan, S.I.S.S.A, Trieste, Italy.

A singular control example.

106.1 Problem Description

Find u over t in [0; 10] to maximize:

$$J = x_3(t_f)$$

subject to:

$$\frac{dx_1}{dt} = u$$
$$\frac{dx_2}{dt} = -x_1$$
$$\frac{dx_2}{dt} = x_2 - x_1^2$$
$$x(t_0) = [0\ 0\ 0]$$
$$|u| <= 1$$

Reference: [8]

106.2 Problem setup

```
toms t
t_F = 10;
p = tomPhase('p', t, 0, t_F, 80);
setPhase(p);

tomStates x1 x2 x3
tomControls u

x = [x1; x2; x3];
```

```
% Initial guess
x0 = {icollocate({x1 == 0, x2 == 0, x3 == 0})
    collocate(u==0)};

% Box constraints
cbox = {-1 <= collocate(u) <= 1};

% Boundary constraints
cbnd = initial(x == [0;0;0]);

% ODEs and path constraints
ceq = collocate({dot(x1) == u; dot(x2) == -x(1)
    dot(x3) == x(2)-x(1).^2});

% Objective
objective = -final(x(3));
```

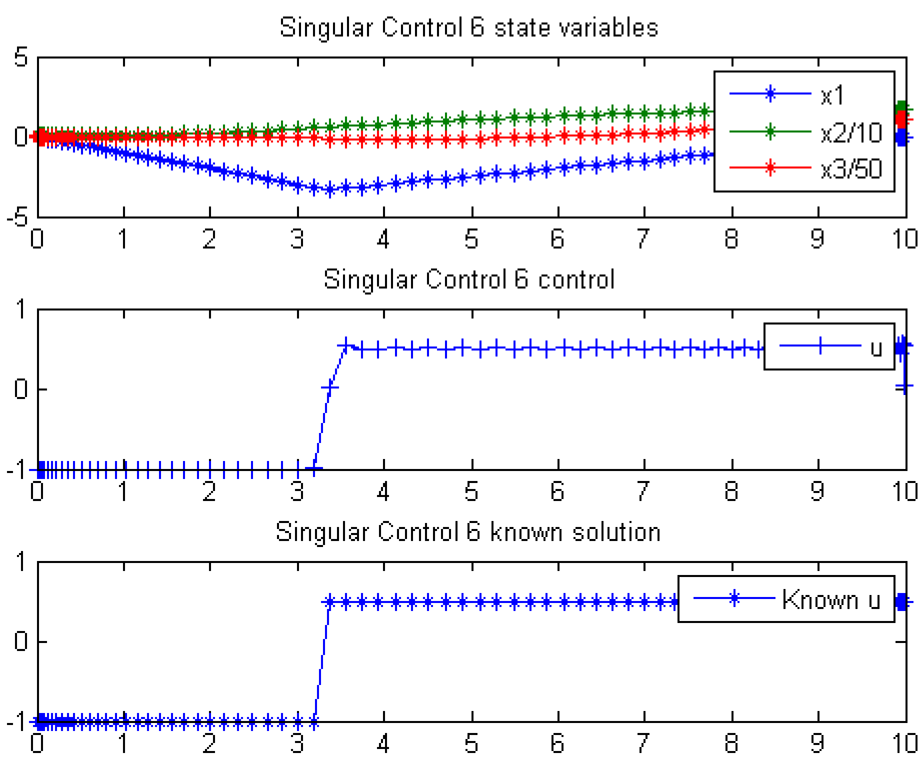
106.3 Solve the problem

```
options = struct;
options.name = 'Singular Control 6';
solution = ezsolve(objective, {cbox, cbnd, ceq}, x0, options);

% Extract optimal states and controls from solution
t  = collocate(subs(t,solution));
u  = collocate(subs(u,solution));
x1 = collocate(subs(x1,solution));
x2 = collocate(subs(x2,solution));
x3 = collocate(subs(x3,solution));
```

```
Problem type appears to be: lpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Singular Control 6             f_k     -55.555568442322624000
                                       sum(|constr|)      0.000000011396340832
                              f(x_k) + sum(|constr|)    -55.555568430926286000
                                              f(x_0)      0.000000000000000000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv   58 ConJacEv   58 Iter   49 MinorIter  678
CPU time: 1.750000 sec. Elapsed time: 1.781000 sec.
```

106.4 Plot result

```
subplot(3,1,1)
plot(t,x1,'*-',t,x2/10,'*-',t,x3/50,'*-');
legend('x1','x2/10','x3/50');
title('Singular Control 6 state variables');

subplot(3,1,2)
plot(t,u,'+-');
legend('u');
title('Singular Control 6 control');

subplot(3,1,3)
plot(t,-1*(t<t_F/3)+1/2*(t>=t_F/3),'*-');
legend('Known u');
title('Singular Control 6 known solution');
```



107 Spring Mass Damper (2 Degree Freedom)

The Direct Approach of General Dynamic Optimal Control: Application on General Software

Tawiwat Veeraklaew, Ph.D. and Settapong Malisuwan, Ph.D. Chulachomklao Royal Military Academy Nakhon-Nayok, Thailand

107.1 Problem Formulation

Find u over t in [0; 2] to minimize

$$J = u_1 + u_2$$

subject to:

$$\frac{dx}{dt} = A * x + B * u$$

Reference: [32]

107.2 Problem setup

```
toms t
p = tomPhase('p', t, 0, 2, 60);
setPhase(p);

tomStates x1 x2 x3 x4
tomControls u1 u2
x = [x1;x2;x3;x4];
u = [u1;u2];

m1 = 1.0; m2 = 1.0; c1 = 1.0; c3 = 1.0;
c2 = 2.0; k1 = 3.0; k2 = 3.0; k3 = 3.0;

B = [0 0; 1/m1 0;
    0 0; 0 1/m2];

A = [0 1 0 0;...
    1/m1*[-(k1+k2) -(c1+c2) k2 c2];...
    0 0 0 1;...
```

```
    1/m2*[ k2 c2 -(k2+k3) -(c2+c3)]];

x0i = [5; 0; 10; 0];
xfi = [0; 0;  0; 0];

% Box constraints
cbox = {0 <= collocate(u) <= 9};

% Boundary constraints
cbnd = {initial(x == x0i)
    final(x == xfi)};

% ODEs and path constraints
ceq = collocate(dot(x) == A*x+B*u);

% Objective
objective = integrate(u1+u2);
```

107.3 Solve the problem

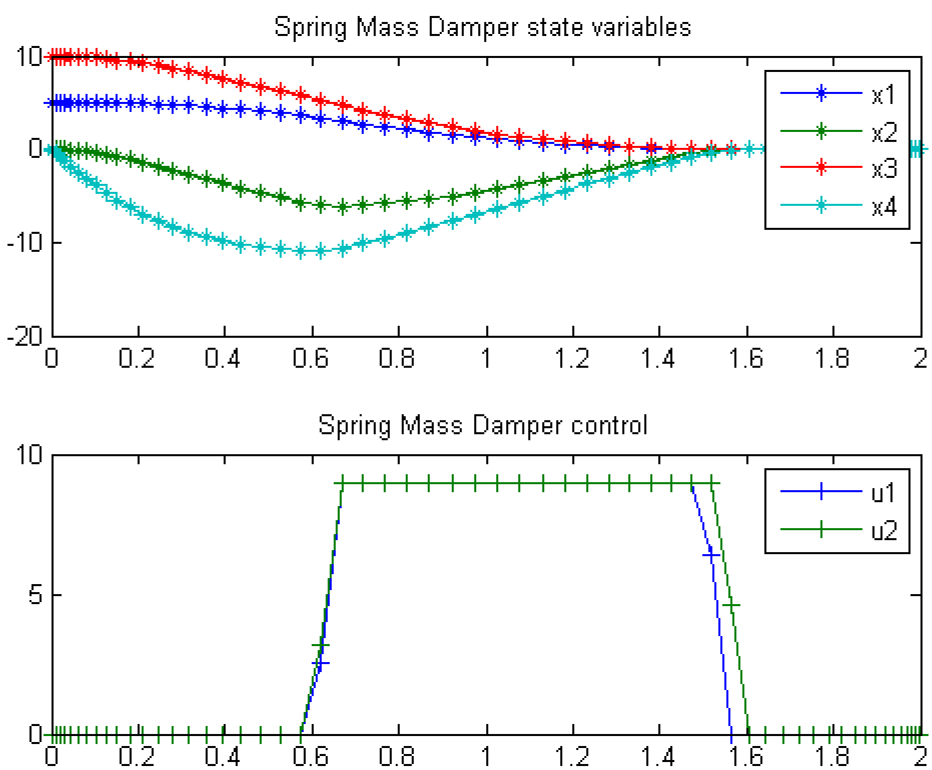
```
options = struct;
options.name = 'Spring Mass Damper';
solution = ezsolve(objective, {cbox, cbnd, ceq}, [], options);
t  = subs(collocate(t),solution);
x1 = subs(collocate(x1),solution);
x2 = subs(collocate(x2),solution);
x3 = subs(collocate(x3),solution);
x4 = subs(collocate(x4),solution);
u1 = subs(collocate(u1),solution);
u2 = subs(collocate(u2),solution);
```

```
Problem type appears to be: lp
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Spring Mass Damper             f_k      16.485256203068737000
                                       sum(|constr|)      0.000000008199340966
                              f(x_k) + sum(|constr|)     16.485256211268076000
                                              f(x_0)      0.000000000000000000

Solver: CPLEX.  EXIT=0.  INFORM=1.
CPLEX Dual Simplex LP solver
Optimal solution found

FuncEv  243 Iter  243
```

```
CPU time: 0.093750 sec. Elapsed time: 0.094000 sec.
```

107.4 Plot result

```
subplot(2,1,1)
plot(t,x1,'*-',t,x2,'*-',t,x3,'*-',t,x4,'*-');
legend('x1','x2','x3','x4');
title('Spring Mass Damper state variables');

subplot(2,1,2)
plot(t,u1,'+-',t,u2,'+-');
legend('u1','u2');
title('Spring Mass Damper control');
```



108 Stirred Tank

Users Guide for dyn.Opt, Example 5a, 5b, 5c

Stirred-Tank Chemical Reactor - Kirk, D. E., Optimal control theory: An introduction, Prentice-Hall, 1970.

5a - unconstrained with terminal penalty 5b - unconstrained 5c - control constraint

108.1 Problem Description

Find u over t in [0; 0.78] to minimize

Does not say u^2 in text

5a:

$$J = \int_0^{0.78} (x_1^2 + x_2^2 + 0.1 * u^2)/2\mathrm{d}t + x_1(t_F)^2 + x_2(t_F)^2$$

5b:

$$J = \int_0^{0.78} (x_1^2 + x_2^2 + 0.1 * u^2)/2\mathrm{d}t$$

5c:

$$J = \int_0^{0.78} (x_1^2 + x_2^2)/2\mathrm{d}t$$

subject to:

$$\begin{aligned} a_1 &= x_1 + 0.25 \\ a_2 &= x_2 + 0.5 \\ a_3 &= x_1 + 2.0 \\ a_4 &= a_2 * exp(25.0 * \frac{x_1}{a_3}) \end{aligned}$$

$$\frac{dx_1}{dt} = -2.0 * a_1 + a_4 - a_1 * u$$

$$\frac{dx_2}{dt} = 0.5 - x_2 - a_4$$

$$x(0) = [0.05\ 0]$$

5b, 5c - x(t_F) = [0 0];

5c - u <= 1

Reference: [16]

108.2 Problem setup

```
toms t

for i=1:3

    p = tomPhase('p', t, 0, 0.78, 40);
    setPhase(p);

    tomStates x1 x2
    tomControls u

    % Initial guess
    x0 = {icollocate({x1 == 0.05; x2 == 0})
        collocate(u == 0)};

    % Box constraints
    cbox = {-1.99 <= icollocate(x1) <= 100
        -100  <= icollocate(x2) <= 100
        -1000 <= collocate(u)  <= 1000};
    % x1 cannot be equal to -2, setting to greater
    % to avoid singularity in a2*exp(25.0*x1/a3)

    % Boundary constraints
    cbnd = initial({x1 == 0.05; x2 == 0});

    % ODEs and path constraints
    a1 = x1 + 0.25; a2 = x2 + 0.5;
    a3 = x1 + 2.0;  a4 = a2.*exp(25.0*x1./a3);

    ceq = collocate({
        dot(x1) == -2.0*a1 + a4 - a1.*u
        dot(x2) == 0.5 - x2 - a4});
```

108.3 Solve the problem

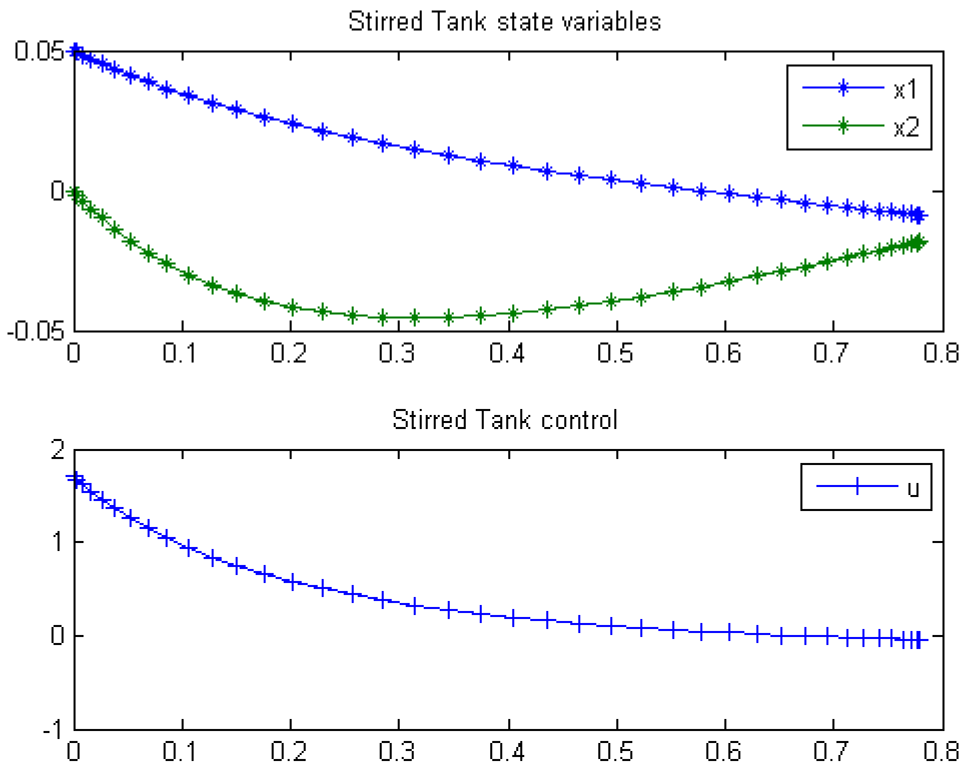
```
options = struct;
if i==1
    objective = final(x1)^2+final(x2)^2+...
        integrate((x1.^2+x2.^2+0.1*u.^2)/2);
    options.name = 'Stirred Tank 5a';
    solution1 = ezsolve(objective, {cbox, cbnd, ceq}, x0, options);
    t1  = subs(collocate(t),solution1);
    x11 = subs(collocate(x1),solution1);
    x21 = subs(collocate(x2),solution1);
    u1  = subs(collocate(u),solution1);
elseif i == 2
    cbnd = {cbnd; final({x1 == 0; x2 == 0})};
    objective = integrate((x1.^2+x2.^2+0.1*u.^2)/2);
    options.name = 'Stirred Tank 5b';
    solution2 = ezsolve(objective, {cbox, cbnd, ceq}, x0, options);
else
    cbnd = {cbnd; final({x1 == 0; x2 == 0})};
    cbox = {-1.99 <= icollocate(x1) <= 100
        -100  <= icollocate(x2) <= 100
        -1    <= collocate(u)   <= 1};
    objective = integrate((x1.^2+x2.^2)/2);
    options.name = 'Stirred Tank 5c';
    solution3 = ezsolve(objective, {cbox, cbnd, ceq}, x0, options);
end
```

```
Problem type appears to be: qpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Stirred Tank 5a                f_k       0.014213969120012267
                                       sum(|constr|)      0.000000005238899986
                              f(x_k) + sum(|constr|)      0.014213974358912253
                                              f(x_0)      0.003474999999999964

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv   30 ConJacEv   30 Iter   27 MinorIter  113
CPU time: 0.171875 sec. Elapsed time: 0.172000 sec.


Problem type appears to be: qpcon
Starting numeric solver
```

```
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Stirred Tank 5b                f_k       0.016702811155814266
                                       sum(|constr|)      0.000000899223593776
                              f(x_k) + sum(|constr|)      0.016703710379408040
                                              f(x_0)      0.000974999999999999

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv   18 ConJacEv   18 Iter   16 MinorIter  118
CPU time: 0.125000 sec. Elapsed time: 0.125000 sec.


Problem type appears to be: qpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Stirred Tank 5c                f_k       0.000989922252663805
                                       sum(|constr|)      0.000000035597664481
                              f(x_k) + sum(|constr|)      0.000989957850328286
                                              f(x_0)      0.000974999999999999

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv   14 ConJacEv   13 Iter   10 MinorIter  139
CPU time: 0.078125 sec. Elapsed time: 0.078000 sec.


end
```

108.4 Plot result

```
subplot(2,1,1)
plot(t1,x11,'*-',t1,x21,'*-');
legend('x1','x2');
title('Stirred Tank state variables');

subplot(2,1,2)
plot(t1,u1,'+-');
legend('u');
title('Stirred Tank control');
```


Stirred Tank state variables
0.05
0
-0.05
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
x1
x2
Stirred Tank control
2
1
0
-1
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
u

109 Temperature Control

Optimal Control CY3H2, Lecture notes by Victor M. Becerra, School of Systems Engineering, University of Reading

Heating a room using the least possible energy.

109.1 Problem Description

Find u over t in [0; 1] to minimize:

$$J = \frac{1}{2}\int_0^1 u^2 \mathrm{d}t$$

subject to:

$$\frac{dx}{dt} = -2 * x + u$$

$$x(0) = 0,$$
$$x(1) = 10$$

109.2 Problem setup

```
toms t
p = tomPhase('p', t, 0, 1, 20);
setPhase(p);

tomStates x
tomControls u

% Initial guess
x0 = {icollocate(x == 10*t)
    collocate(u == 1)};

% Box constraints
cbox = collocate(0 <= u);
```

```
% Boundary constraints
cbnd = {initial(x == 0)
    final(x == 10)};

% ODEs and path constraints
ceq = collocate(dot(x) == -2*x+u);

% Objective
objective = 0.5*integrate(u^2);
```

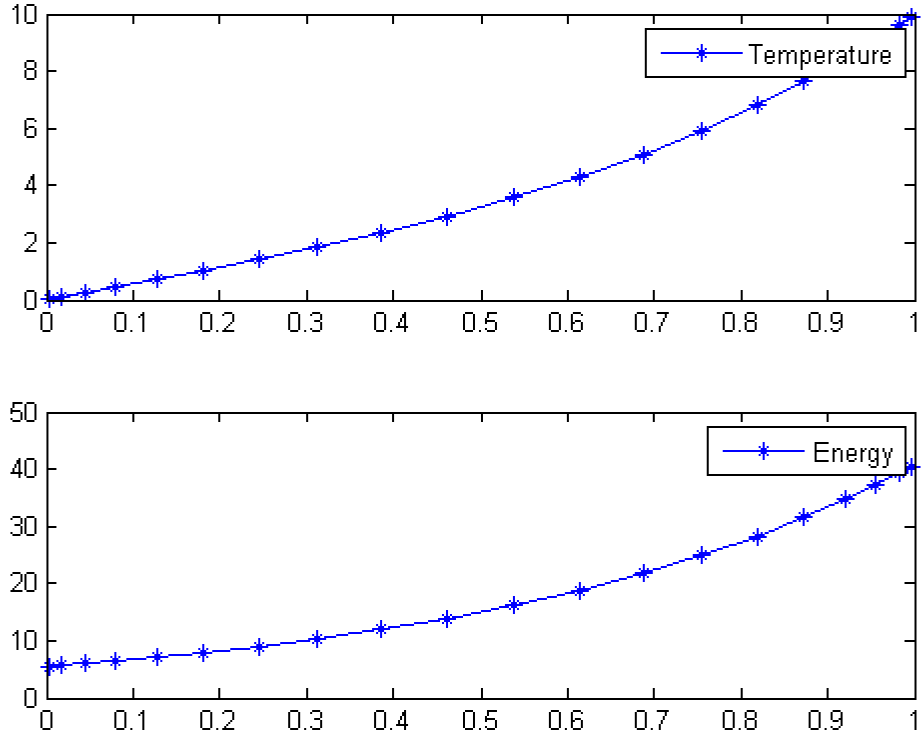
109.3 Solve the problem

```
options = struct;
options.name = 'Temperature Control';
solution = ezsolve(objective, {cbox, cbnd, ceq}, x0, options);
t = subs(collocate(t),solution);
x = subs(collocate(x),solution);
u = subs(collocate(u),solution);
```

```
Problem type appears to be: qp
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem:  1: Temperature Control                f_k     203.731472072763980000
                                       sum(|constr|)      0.000000000046672914
                              f(x_k) + sum(|constr|)    203.731472072810650000
                                              f(x_0)      0.000000000000000000

Solver: CPLEX.  EXIT=0.  INFORM=1.
CPLEX Barrier QP solver
Optimal solution found

FuncEv    9 GradEv    9 ConstrEv    9 Iter    9
CPU time: 0.093750 sec. Elapsed time: 0.063000 sec.
```

109.4 Plot result

```
figure(1);
subplot(2,1,1)
plot(t,x,'*-');
legend('Temperature');
subplot(2,1,2)
plot(t,u,'*-');
legend('Energy');
```


Temperature
10
8
6
4
2
0
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1
Energy
50
40
30
20
10
0
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1

110 Room temperature control

110.1 Problem Description

Temperature control from 0.00 to 7.00 hours at night. Finds best heating policy at night that brings the temperature back to 20 [oC] in the morning irrespective of night temperatures.

Programmers: Gerard Van Willigenburg (Wageningen University)

110.2 Problem setup

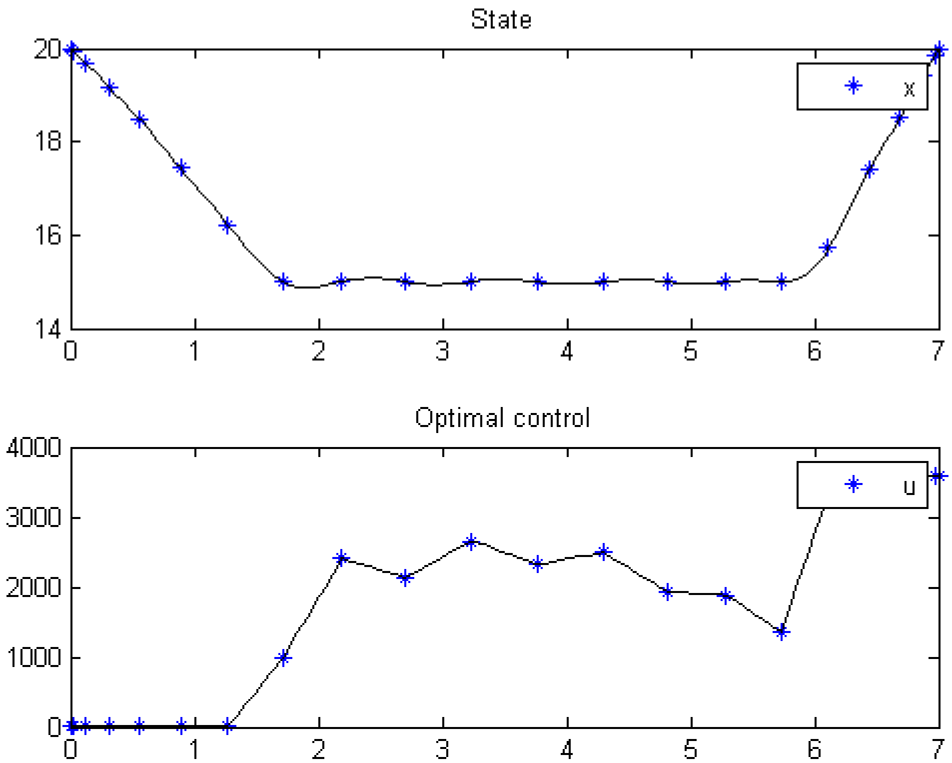
Define tomSym variable t (time) and t_f (final time) if the final time is free

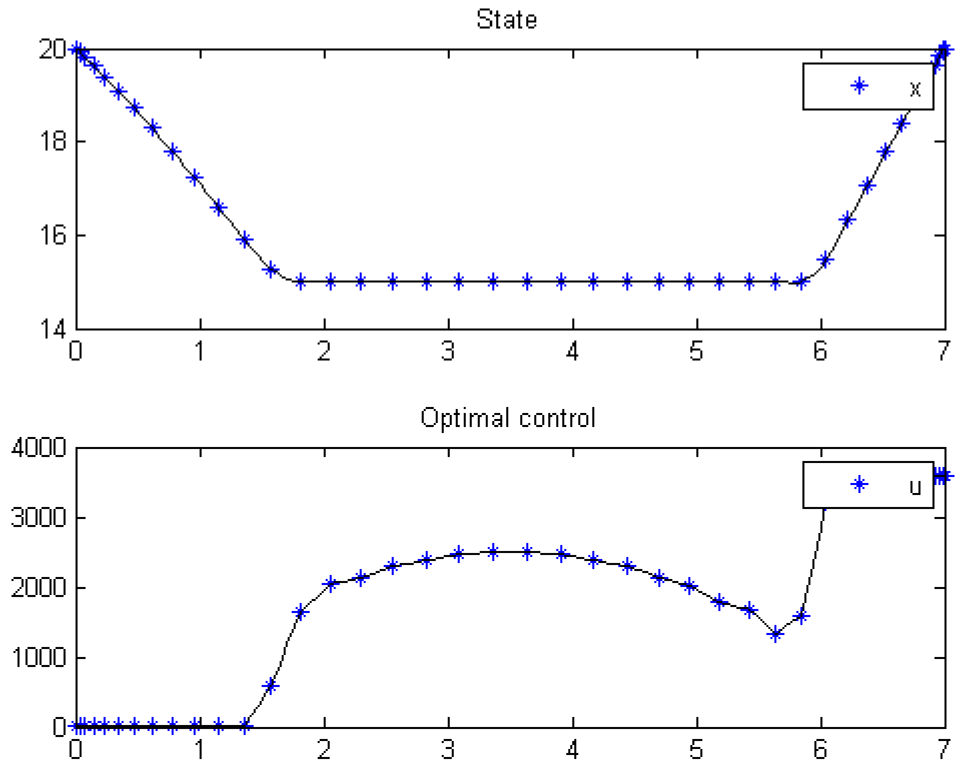
```
toms t; t_f=7; % Fixed final time

for n=[20 40]
    % Define & set time axis
    p = tomPhase('p', t, 0, t_f, n);
    setPhase(p);

    % Define the state and control variables
    tomStates x
    tomControls u

    % Initial state
    xi=20;

    % Initial guess
    if n==20
        x0 = {icollocate({x == xi(1)})
            collocate({u == 0})};
    else
        x0 = {icollocate({x == xopt})
            collocate({u == uopt})};
    end

    % Boundary conditions
    cbnd = {initial({x == xi}); final({x == xi})};

    % Equality constraints: state-space differential equations
    tau=2; pH=0.002; % Parameters

    % External input d1
```

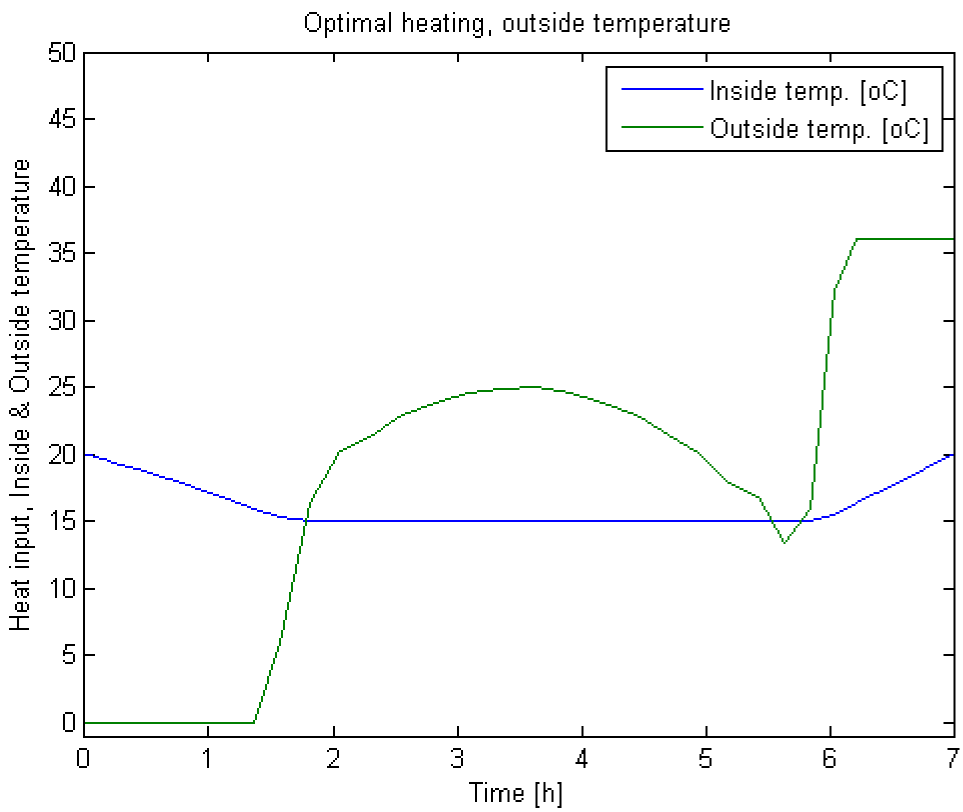
```
    d1=15-10*sin(pi*t/t_f);

    %Differential equation
    ceq = collocate({dot(x) == 1/tau*(d1-x) + pH*u});

    % Inequality constraints
    cbox = {0 <= collocate(u) <= 3600; icollocate(x) >= 15};

    % Cost function to be minimized
    objective = integrate(u+1e-6*dot(u)^2);

    % Solve the problem after specifying its name
    options = struct;
    options.name = 'Temperature control at night';
    solution = ezsolve(objective, {cbox, cbnd, ceq}, x0, options);

    % Plot intermediate solution
    figure; subplot(2,1,1);
    ezplot(x); legend('x');
    title('State');

    subplot(2,1,2);
    ezplot(u); legend('u');
    title('Optimal control'); drawnow;

    % Obtain intermediate solution to initialize the next
    xopt = subs(x,solution);
    uopt = subs(u,solution);
end

% Obtain final solution t,x,...,u,..
% that overwrite the associated tomSym variables
t = subs(collocate(t),solution);
x = subs(collocate(x),solution);
u = subs(collocate(u),solution);

%Plot results
figure; plot(t,x,t,u/100); axis([0 t_f -1 50]);
xlabel('Time [h]'); ylabel('Heat input, Inside & Outside temperature');
title('Optimal heating, outside temperature');
legend('Inside temp. [oC]','Outside temp. [oC]');
```

```
Problem type appears to be: qp
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
```

```
Problem:  1: Temperature control at night       f_k   12892.776351482709000000
                                      sum(|constr|)      0.000000000002029896
                              f(x_k) + sum(|constr|)  12892.776351482711000000
                                             f(x_0)      0.000000000000000000

Solver: CPLEX.  EXIT=0.  INFORM=1.
CPLEX Barrier QP solver
Optimal solution found

FuncEv   11 GradEv   11 ConstrEv   11 Iter   11
CPU time: 0.062500 sec. Elapsed time: 0.031000 sec.
Problem type appears to be: qp
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem:  1: Temperature control at night       f_k   12882.457401725025000000
                                      sum(|constr|)      0.000000006475484391
                              f(x_k) + sum(|constr|)  12882.457401731501000000
                                             f(x_0)      0.000000000000000000

Solver: CPLEX.  EXIT=0.  INFORM=1.
CPLEX Barrier QP solver
Optimal solution found

FuncEv   13 GradEv   13 ConstrEv   13 Iter   13
CPU time: 0.062500 sec. Elapsed time: 0.047000 sec.
```


State
x
Optimal control
u


State
x
20
18
16
14
0
1
2
3
4
5
6
7
Optimal control
u
4000
3000
2000
1000
0


Optimal heating, outside temperature
Inside temp. [oC]
Outside temp. [oC]
Heat input, Inside & Outside temperature
Time [h]
0
1
2
3
4
5
6
7
0
5
10
15
20
25
30
35
40
45
50

111 A Simple Terminal Constraint Problem

Problem 1: Miser3 manual

111.1 Problem Description

Find u(t) over t in [0; 1] to minimize

$$J = \int_0^1 (x^2 + u^2)\mathrm{d}t$$

subject to:

$$\frac{dx}{dt} = u$$

$$x(0) = 1$$

$$x(1) = 0.75$$

$$x(0.75) = 0.9$$

111.2 Problem setup

```
toms t1
p1 = tomPhase('p1', t1, 0, 0.75, 20);
toms t2
p2 = tomPhase('p2', t2, 0.75, 0.25, 20);

setPhase(p1);
tomStates x1p1
tomControls up1
setPhase(p2);
tomStates x1p2
tomControls up2
setPhase(p1);

% Initial guess
x01 = {icollocate({x1p1 == 1-0.1*t1/0.75})
    collocate(up1==0.9*t1/0.75)};
```

```
% Box constraints
cbox1 = {-10 <= icollocate(p1,x1p1) <= 10
    -10 <= collocate(p1,up1) <= 10};

% Boundary constraints
cbnd1 = initial(x1p1 == 1);

% ODEs and path constraints
ceq1 = collocate(dot(x1p1) == up1);

% Objective
objective1 = integrate(x1p1.^2+up1.^2);

setPhase(p2);
% Initial guess
x02 = {icollocate({x1p2 == 1-0.1*t2})
    collocate(up2==0.9+0.1*t2)};

% Box constraints
cbox2 = {-10 <= icollocate(p2,x1p2) <= 10
    -10 <= collocate(p2,up2) <= 10};

% Boundary constraints
cbnd2 = {initial(x1p2 == 0.9)
    final(x1p2 == 0.75)};

% ODEs and path constraints
ceq2 = collocate(dot(x1p2) == up2);

% Objective
objective2 = integrate(x1p2.^2+up2.^2);

% Objective
objective = objective1 + objective2;

% Link phase
link = {final(p1,x1p1) == initial(p2,x1p2)};
```

111.3 Solve the problem

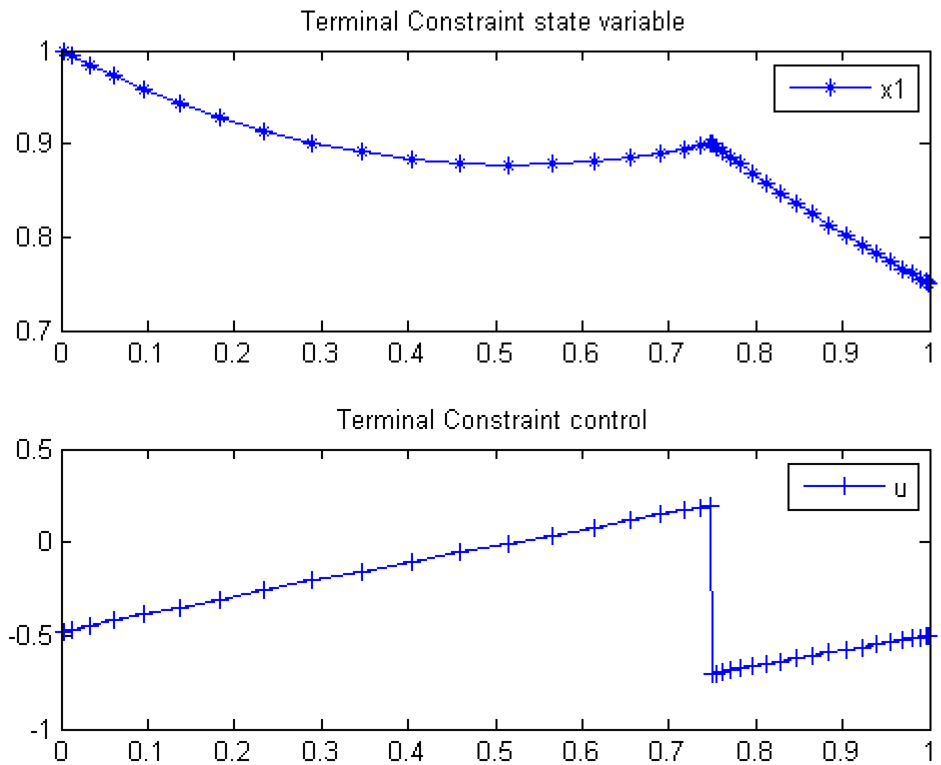
```
options = struct;
options.name = 'Terminal Constraint 2';
constr = {cbox1, cbnd1, ceq1, cbox2, cbnd2, ceq2, link};
solution = ezsolve(objective, constr, {x01, x02}, options);

t = subs(collocate(p1,t1),solution);
t = [t;subs(collocate(p2,t2),solution)];
```

```
x1 = subs(collocate(p1,x1p1),solution);
x1 = [x1;subs(collocate(p2,x1p2),solution)];
u = subs(collocate(p1,up1),solution);
u = [u;subs(collocate(p2,up2),solution)];
```

```
Problem type appears to be: qp
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem:  1: Terminal Constraint 2               f_k       0.920531441472477340
                                       sum(|constr|)      0.000000000559899399
                              f(x_k) + sum(|constr|)      0.920531442032376690
                                              f(x_0)      0.000000000000000000

Solver: CPLEX.  EXIT=0.  INFORM=1.
CPLEX Barrier QP solver
Optimal solution found

FuncEv    8 GradEv    8 ConstrEv    8 Iter    8
CPU time: 0.031250 sec. Elapsed time: 0.031000 sec.
```

111.4 Plot result

```
subplot(2,1,1)
plot(t,x1,'*-');
legend('x1');
title('Terminal Constraint state variable');

subplot(2,1,2)
plot(t,u,'+-');
legend('u');
title('Terminal Constraint control');
```


Terminal Constraint state variable
x1
1
0.9
0.8
0.7
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1
Terminal Constraint control
u
0.5
0
-0.5
-1
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1

112 Third order system

112.1 Problem description

Time-optimal control of a third order system with bounded control.

Programmers: Gerard Van Willigenburg (Wageningen University) Willem De Koning (retired from Delft University of Technology)

112.2 Problem setup

```
% Array with consecutive number of collocation points
narr = [20 40];

toms t t_f % Free final time

for n=narr

    p = tomPhase('p', t, 0, t_f, n);
    setPhase(p)

    tomStates x1 x2 x3
    tomControls u1

    % Initial & terminal states
    xi = [0; 0.931; 0.9];
    xf = [2;     0;   0];

    % Initial guess
    if n==narr(1)
        x0 = {t_f == 5; icollocate({x1 == xi(1); x2 == xi(2)
            x3 == xi(3)})
            collocate({u1 == 0})};
    else
        x0 = {t_f == tfopt; icollocate({x1 == xopt1; x2 == xopt2
            x3 == xopt3})
            collocate({u1 == uopt1})};
    end

    % Box constraints
    cbox = {-1 <= collocate(u1) <= 1};

    % Boundary constraints
```

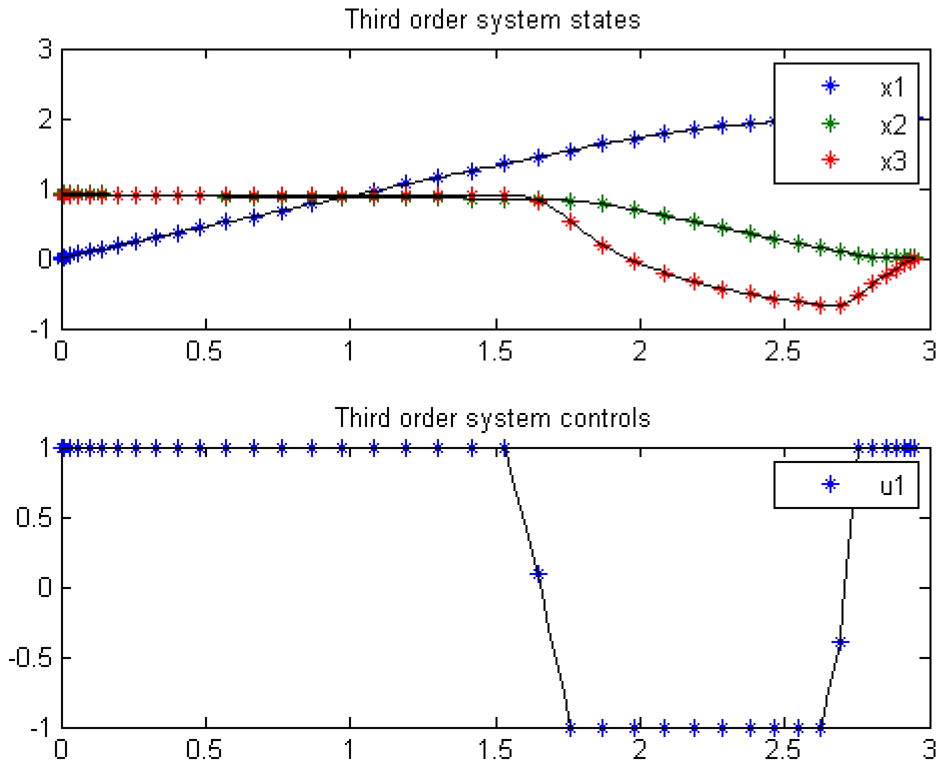
```
cbnd = {initial({x1 == xi(1); x2 == xi(2); x3 == xi(3)})
    final({x1 == xf(1); x2 == xf(2); x3 == xf(3)})};

% ODEs and path constraints
dx1 = x2;
dx2 = -x2-0.1*x2.*x2.*x2+x3;
dx3 = -2*x3+-0.2*x3./sqrt(x3.*x3+1e-4)+2*u1;

ceq = collocate({
    dot(x1) == dx1
    dot(x2) == dx2
    dot(x3) == dx3});

% Objective
objective = t_f;
```

112.3 Solve the problem

```
options = struct;
options.name = 'Third order system';
solution = ezsolve(objective, {cbox, cbnd, ceq}, x0, options);

tfopt = subs(t_f,solution);
xopt1 = subs(x1,solution);
xopt2 = subs(x2,solution);
xopt3 = subs(x3,solution);
uopt1 = subs(u1,solution);
```

```
Problem type appears to be: lpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Third order system             f_k       2.956507317430983900
                                       sum(|constr|)      0.000000000033334328
                              f(x_k) + sum(|constr|)      2.956507317464318200
                                              f(x_0)      5.000000000000000000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv   12 ConJacEv   12 Iter    9 MinorIter  101
CPU time: 0.046875 sec. Elapsed time: 0.047000 sec.


Problem type appears to be: lpcon
```

```
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Third order system             f_k       2.949634637798719300
                                     sum(|constr|)      0.000001841572751425
                              f(x_k) + sum(|constr|)      2.949636479371470900
                                            f(x_0)      2.956507317430983900

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv    5 ConJacEv    5 Iter    4 MinorIter  136
CPU time: 0.062500 sec. Elapsed time: 0.063000 sec.


end

figure(1)
subplot(2,1,1);
ezplot([x1; x2; x3]); legend('x1','x2','x3');
title('Third order system states');

subplot(2,1,2);
ezplot(u1); legend('u1');
title('Third order system controls');
```


Third order system states
x1
x2
x3
3
2
1
0
-1
0
0.5
1
1.5
2
2.5
3
Third order system controls
u1
1
0.5
0
-0.5
-1
0
0.5
1
1.5
2
2.5
3

113 Time Delay 1

ITERATIVE DYNAMIC PROGRAMMING, REIN LUUS

8.3.1 Example 1

CHAPMAN & HALL/CRC Monographs and Surveys in Pure and Applied Mathematics

Linear time-delay system used for optimal control studies by Chan and Perkins

113.1 Problem Formulation

Find u over t in [0; 5] to minimize

$$J = x_3(t_F)$$

subject to:

$$\frac{dx_1}{dt} = x_2$$
$$\frac{dx_2}{dt} = -10 * x_1 - 5 * x_2 - 2 * x_1(t - tau) - x_2(t - tau) + u$$
$$\frac{dx_3}{dt} = 0.5 * (10 * x_1^2 + x_2^2 + u^2)$$
$$tau = 0.25$$

The initial condition are:

$$x(t <= 0) = [1\ 1\ 0]$$
$$-inf <= u <= inf$$

Reference: [25]

113.2 Problem setup

```
toms t
p1 = tomPhase('p1', t, 0, 5, 50);
setPhase(p1);

tomStates x1 x2 x3
tomControls u

% Initial guess
x0 = {icollocate({x1 == 1
    x2 == 1; x3 == 0})
    collocate(u == 0)};

% Boundary constraints
cbnd = initial({x1 == 1; x2 == 1; x3 == 0});

% Expressions for x1(t-tau) and x2(t-tau)
tau = 0.25;
x1delayed = ifThenElse(t<tau, 1, subs(x1,t,t-tau));
x2delayed = ifThenElse(t<tau, 1, subs(x2,t,t-tau));

% ODEs and path constraints
ceq = collocate({dot(x1) == x2
    dot(x2) == -10*x1 - 5*x2 - 2*x1delayed - x2delayed + u
    dot(x3) == 0.5*(10*x1.^2+x2.^2+u.^2)});

% Objective
objective = final(x3);
```

113.3 Solve the problem

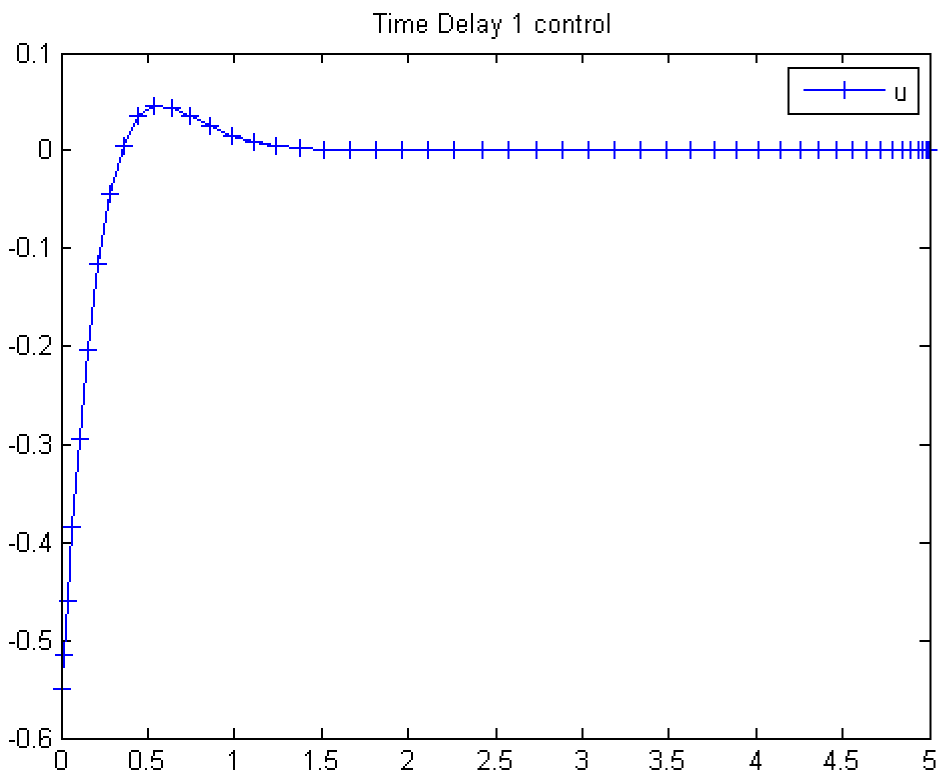
```
options = struct;
options.name = 'Time Delay 1';
solution = ezsolve(objective, {cbnd, ceq}, x0, options);
t  = subs(collocate(t),solution);
u  = subs(collocate(u),solution);
```

```
Problem type appears to be: lpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Time Delay 1                   f_k       2.525970860473679000
                                       sum(|constr|)      0.000000011182005725
                              f(x_k) + sum(|constr|)      2.525970871655684600
```

```
                                                    f(x_0)      0.000000000000000000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv   70 ConJacEv   70 Iter   49 MinorIter  205
CPU time: 0.671875 sec. Elapsed time: 0.687000 sec.
```

113.4 Plot result

```
figure(1)
plot(t,u,'+-');
legend('u');
title('Time Delay 1 control');
```



114 Time Delay 1 (Approximate)

ITERATIVE DYNAMIC PROGRAMMING, REIN LUUS

8.3.1 Example 1

CHAPMAN & HALL/CRC Monographs and Surveys in Pure and Applied Mathematics

Linear time-delay system used for optimal control studies by Chan and Perkins

114.1 Problem Formulation

Find u over t in [0; 5] to minimize

$$J = x_3(t_F)$$

subject to:

$$\frac{dx_1}{dt} = x_2$$
$$\frac{dx_2}{dt} = -10 * x_1 - 5 * x_2 - 2 * x_1(t - tau) - x_2(t - tau) + u$$
$$\frac{dx_3}{dt} = 0.5 * (10 * x_1^2 + x_2^2 + u^2)$$
$$tau = 0.25$$

A Taylor series expansion gives:

$$\frac{dx_2}{dt} \approx (-12 * x_1 + (2 * tau - 6) * x_2 + u)/(1 - tau)$$

The initial condition are:

$$x(0) = [1\ 1\ 0]$$
$$-inf <= u <= inf$$

Reference: [25]

114.2 Problem setup

```
toms t
p = tomPhase('p', t, 0, 5, 50);
setPhase(p);

tomStates x1 x2 x3
tomControls u

% Initial guess
x0 = {icollocate({x1 == 1; x2 == 1; x3 == 0})
    collocate(u == 0)};

% Boundary constraints
cbnd = initial({x1 == 1; x2 == 1; x3 == 0});

% ODEs and path constraints
tau = 0.25;
ceq = collocate({dot(x1) == x2
    dot(x2) == (-12*x1+(2*tau-6)*x2 + u)/(1-tau)
    dot(x3) == 0.5*(10*x1.^2+x2.^2+u.^2)});

% Objective
objective = final(x3);
```

114.3 Solve the problem

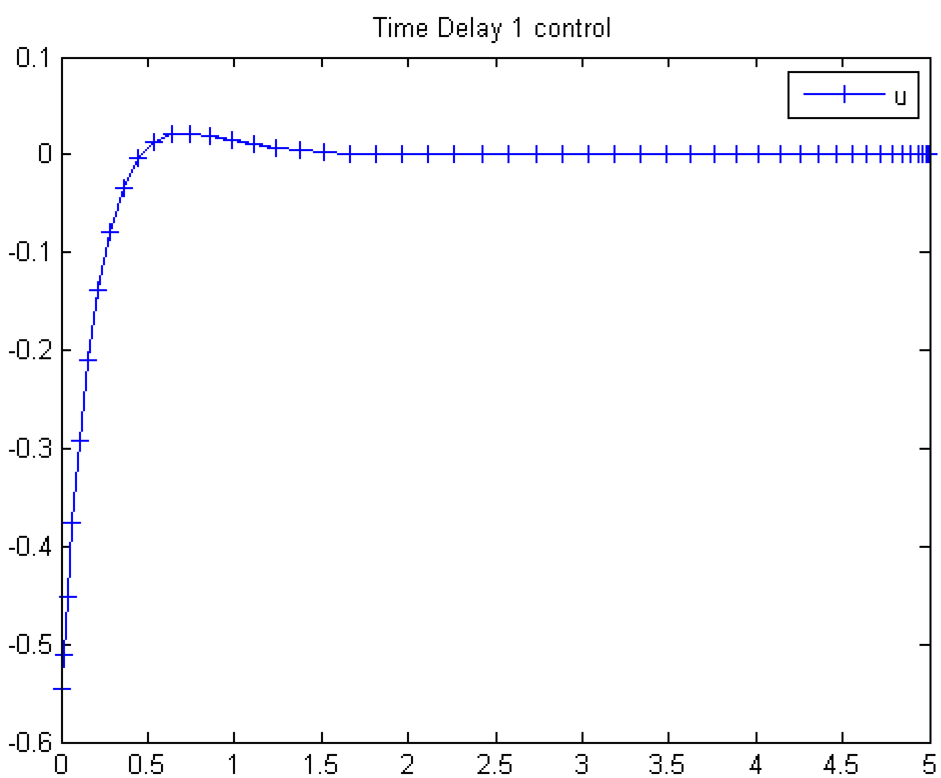
```
options = struct;
options.name = 'Time Delay 1 Appr.';
solution = ezsolve(objective, {cbnd, ceq}, x0, options);
t = subs(collocate(t),solution);
u = subs(collocate(u),solution);
```

```
Problem type appears to be: lpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Time Delay 1 Appr.              f_k       2.387051416916649200
                                       sum(|constr|)      0.000000035059442522
                              f(x_k) + sum(|constr|)      2.387051451976091700
                                              f(x_0)      0.000000000000000000
```

```
Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv   67 ConJacEv   67 Iter   48 MinorIter  214
CPU time: 0.671875 sec. Elapsed time: 0.734000 sec.
```

114.4 Plot result

```
figure(1)
plot(t,u,'+-');
legend('u');
title('Time Delay 1 control');
```



115 Time Delay 2

ITERATIVE DYNAMIC PROGRAMMING, REIN LUUS

8.3.2 Example 2

CHAPMAN & HALL/CRC Monographs and Surveys in Pure and Applied Mathematics

Linear time-delay system considered by Palanisamy et al.

115.1 Problem Formulation

Find u over t in [0; 2] to minimize

$$J = x_2(t_F)$$

subject to:

$$\frac{dx_1}{dt} = t * x_1 + x_1(t - tau) + u$$
$$\frac{dx_2}{dt} = x_1^2 + u^2$$
$$tau = 1$$

The initial condition are:

$$x(t <= 0) = [1 \ 0]$$
$$-inf <= u <= inf$$

Reference: [25]

115.2 Problem setup

```
toms t
p1 = tomPhase('p1', t, 0, 2, 50);
setPhase(p1);
```

```
tomStates x1 x2
tomControls u

% Initial guess
x0 = {icollocate({x1 == 1; x2 == 0})
    collocate(u == 0)};

% Boundary constraints
cbnd = initial({x1 == 1; x2 == 0});

% Expression for x1(t-tau)
tau = 1;
x1delayed = ifThenElse(t<tau, 1, subs(x1,t,t-tau));

% ODEs and path constraints
ceq = collocate({
    dot(x1) == t.*x1 + x1delayed + u
    dot(x2) == x1.^2 + u.^2});

% Objective
objective = final(x2);
```

115.3 Solve the problem

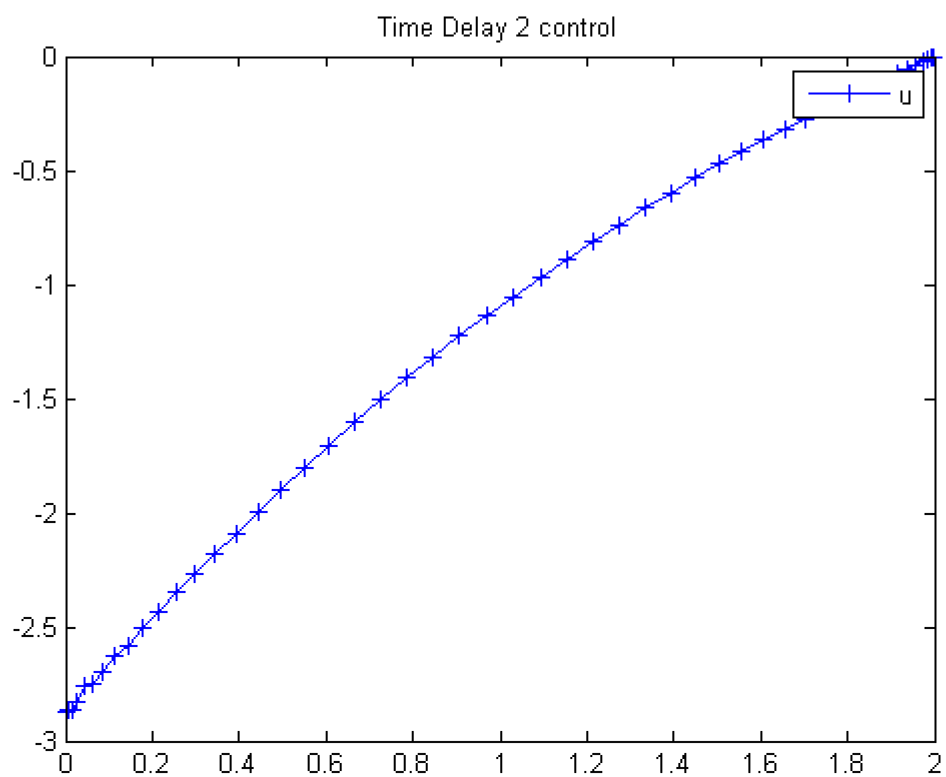
```
options = struct;
options.name = 'Time Delay 2';
solution = ezsolve(objective, {cbnd, ceq}, x0, options);
t  = subs(collocate(t),solution);
u  = subs(collocate(u),solution);
```

```
Problem type appears to be: lpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Time Delay 2                  f_k       4.796108536142883200
                                    sum(|constr|)      0.000000305572156922
                              f(x_k) + sum(|constr|)      4.796108841715040100
                                           f(x_0)      0.000000000000000000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv   32 ConJacEv   32 Iter   27 MinorIter  137
```

```
CPU time: 0.171875 sec. Elapsed time: 0.204000 sec.
```

115.4 Plot result

```
figure(1)
plot(t,u,'+-');
legend('u');
title('Time Delay 2 control');
```



116 Time Delay 2 (Approximate)

ITERATIVE DYNAMIC PROGRAMMING, REIN LUUS

8.3.2 Example 2

CHAPMAN & HALL/CRC Monographs and Surveys in Pure and Applied Mathematics

Linear time-delay system considered by Palanisamy et al.

116.1 Problem Formulation

Find u over t in [0; 2] to minimize

$$J = x_2(t_F)$$

subject to:

$$\begin{gathered}\frac{dx_1}{dt} = t * x_1 + x_1(t - tau) + u \\ \frac{dx_2}{dt} = x_1^2 + u^2 \\ tau = 1\end{gathered}$$

A Taylor series expansion gives:

$$\frac{dx_1}{dt} \approx \frac{(t+1) * x_1 + u}{1 + tau}$$

The initial condition are:

$$\begin{gathered}x(0) = [1\ 0] \\ -inf <= u <= inf\end{gathered}$$

Reference: [25]

116.2 Problem setup

```
toms t
p = tomPhase('p', t, 0, 2, 50);
setPhase(p);

tomStates x1 x2
tomControls u

% Initial guess
x0 = {icollocate({x1 == 1; x2 == 0})
    collocate(u == 0)};

% Boundary constraints
cbnd = initial({x1 == 1; x2 == 0});

% ODEs and path constraints
tau = 1;
ceq = collocate({
    dot(x1) == ((t+1).*x1+u)/(1+tau)
    dot(x2) == x1.^2+u.^2});

% Objective
objective = final(x2);
```

116.3 Solve the problem

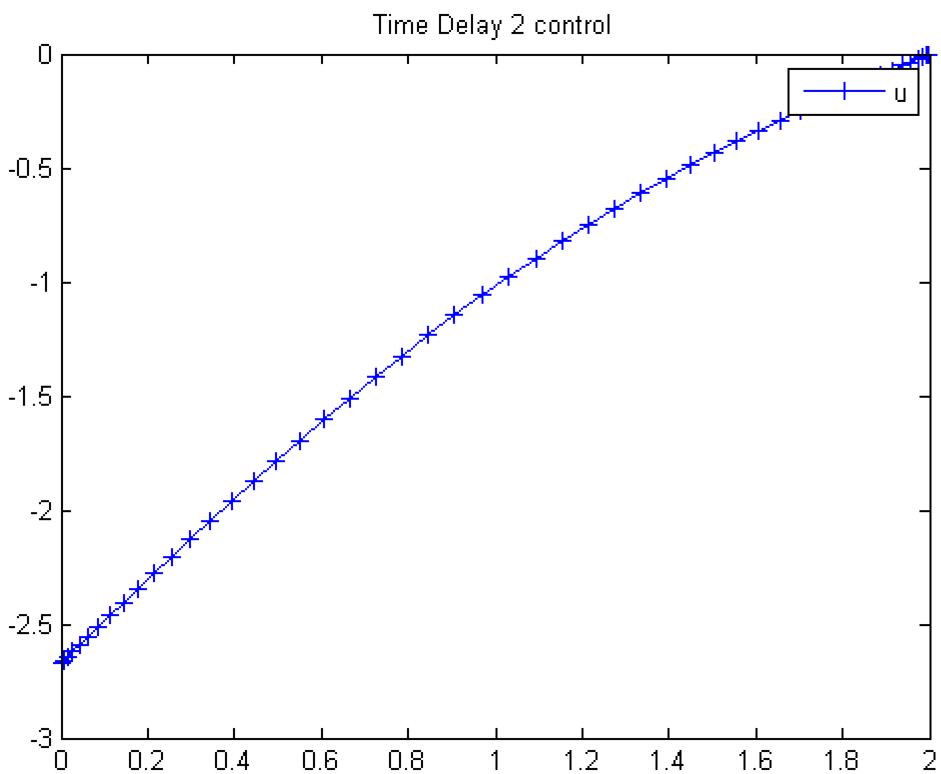
```
options = struct;
options.name = 'Time Delay 2 Appr.';
solution = ezsolve(objective, {cbnd, ceq}, x0, options);
t = subs(collocate(t),solution);
u = subs(collocate(u),solution);
```

```
Problem type appears to be: lpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Time Delay 2 Appr.             f_k       5.340734691399960700
                                       sum(|constr|)      0.000000236393505091
                              f(x_k) + sum(|constr|)      5.340734927793465500
                                              f(x_0)      0.000000000000000000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied
```

```
FuncEv    1 ConstrEv   31 ConJacEv   31 Iter   26 MinorIter  138
CPU time: 0.218750 sec. Elapsed time: 0.218000 sec.
```

116.4 Plot result

```
figure(1)
plot(t,u,'+-');
legend('u');
title('Time Delay 2 control');
```



117 Transfer Min Swing

Example 7.1: K.L. Teo, K. K. Leong, G.J. Goh

117.1 Problem Formulation

Find u over t in [0; 1] to minimize

$$J = \int_0^1 4.5(x_3^2 + x_6^2)\mathrm{d}t$$

subject to:

$$\frac{dx_1}{dt} = 9 * x_4$$
$$\frac{dx_2}{dt} = 9 * x_5$$
$$\frac{dx_3}{dt} = 9 * x_6$$
$$\frac{dx_4}{dt} = 9 * (x_7 + 17.2656 * x_3)$$
$$\frac{dx_5}{dt} = 9 * x_8$$
$$\frac{dx_6}{dt} = -9 * \frac{x_7 + 27.0756 * x_3 + 2 * x_5 * x_6}{x_2}$$
$$\frac{dx_7}{dt} = 9 * u_1$$
$$\frac{dx_8}{dt} = 9 * u_2$$

$$x(0) = [0\ 22\ 0\ 0\ -1\ 0\ NaN\ NaN]$$
$$x(1) = [10\ 14\ 0\ 2.5\ 0\ 0\ NaN\ NaN]$$
$$|x_4| <= 2.5$$
$$|x_5| <= 1.0$$
$$|x_7| <= 2.83374$$
$$-0.80865 <= x_8 <= 0.71265$$

$$|u| <= 10$$

Reference: [30]

117.2 Problem setup

```
toms t phi1 phi2

% Starting guess
speed = 5;
xopt = 1.2*t;
yopt = 1.6*t;
thetaopt = pi/4;

phi1opt = 1;
phi2opt = 1;
x1opt = 10*t;
x2opt = 22-8*t;
x3opt = 0;
x4opt = 2.5*t;
x5opt = -1+t;
x6opt = 0;
x7opt = 0;
x8opt = 0;
u1opt = 0;
u2opt = 0;
```

117.3 Solve the problem, using a successively larger number collocation points

```
for n=[20 40]

    % Create a new phase and states, using n collocation points
    p = tomPhase('p', t, 0, 1, n);
    setPhase(p);
    tomStates x1 x2 x3 x4 x5 x6 x7 x8
    tomControls u1 u2

    % Initial guess
    x0 = {phi1 == phi1opt; phi2 == phi2opt
        icollocate({
        x1 == x1opt; x2 == x2opt
        x3 == x3opt; x4 == x4opt
        x5 == x5opt; x6 == x6opt
        x7 == x7opt; x8 == x8opt})
        collocate({
```

```
    u1 == u1opt; u2 == u2opt})};

% Box constraints
cbox = {-10  <= phi1 <= 10
    -10      <= phi2 <= 10
    -2.5     <= icollocate(x4) <= 2.5
    -1       <= icollocate(x5) <= 1
    -2.83374 <= icollocate(x7) <= 2.83374
    -0.80865 <= icollocate(x8) <= 0.71265
    -10      <= collocate(u1) <= 10
    -10      <= collocate(u2) <= 10};

% Boundary constraints
cbnd = {initial({x1 == 0
    x2 == 22; x3 == 0
    x4 == 0;  x5 == -1
    x6 == 0;  x7 == phi1
    x8 == phi2
    })
    final({x1 == 10
    x2 == 14;  x3 == 0
    x4 == 2.5; x5 == 0
    x6 == 0})};

% ODEs and path constraints
ceq = collocate({dot(x1) == 9*x4
    dot(x2) == 9*x5; dot(x3) == 9*x6
    dot(x4) == 9*(x7+17.2656*x3)
    dot(x5) == 9*x8
    dot(x6) == -9*(x7+27.0756*x3+2*x5.*x6)./x2
    dot(x7) == 9*u1; dot(x8) == 9*u2});

% Objective
objective = integrate(4.5*(x3.^2 + x6.^2));
```

117.4 Solve the problem

```
options = struct;
options.name = 'Transfer Min Swing';
solution = ezsolve(objective, {cbox, cbnd, ceq}, x0, options);

% Optimal x and u to use as starting guess in the next iteration
x1opt = subs(x1, solution);
x2opt = subs(x2, solution);
x3opt = subs(x3, solution);
x4opt = subs(x4, solution);
x5opt = subs(x5, solution);
```

```
    x6opt = subs(x6, solution);
    x7opt = subs(x7, solution);
    x8opt = subs(x8, solution);
    u1opt = subs(u1, solution);
    u2opt = subs(u2, solution);
    phi1opt = subs(phi1, solution);
    phi2opt = subs(phi2, solution);
```

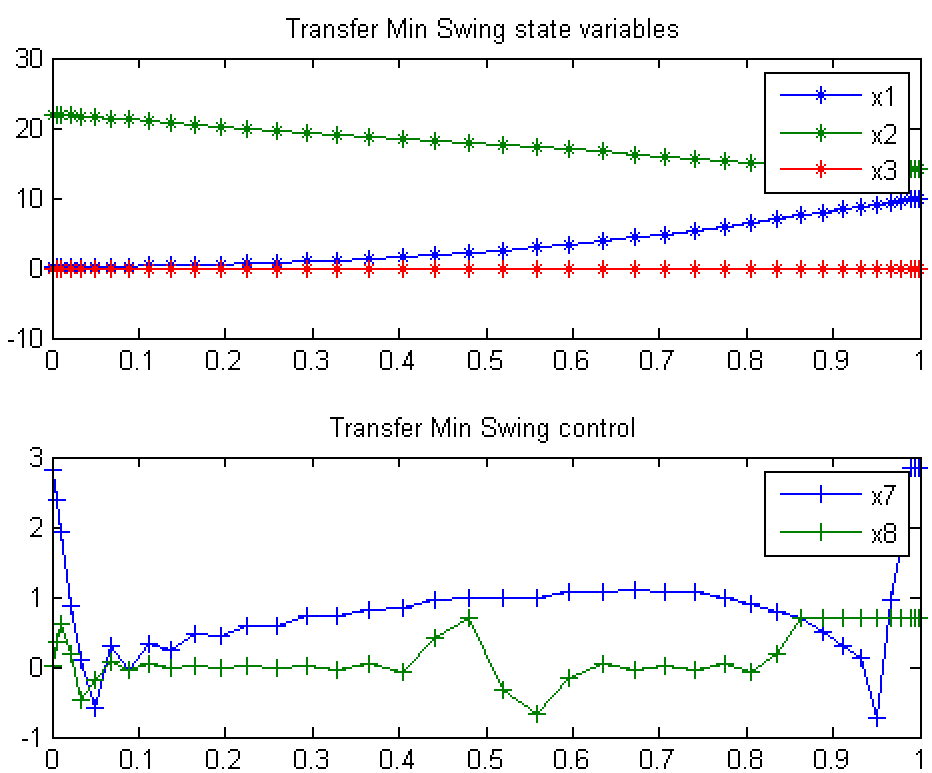
```
Problem type appears to be: qpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Transfer Min Swing             f_k       0.005155677076381509
                                       sum(|constr|)      0.000000000219058091
                              f(x_k) + sum(|constr|)      0.005155677295439600
                                              f(x_0)      0.000000000000000000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv  107 ConJacEv  107 Iter  100 MinorIter  375
CPU time: 0.515625 sec. Elapsed time: 0.547000 sec.


Problem type appears to be: qpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Transfer Min Swing             f_k       0.005157874717791312
                                       sum(|constr|)      0.000000001283862156
                              f(x_k) + sum(|constr|)      0.005157876001653469
                                              f(x_0)      0.005155729096774477

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv  126 ConJacEv  126 Iter  123 MinorIter  594
CPU time: 2.453125 sec. Elapsed time: 2.578000 sec.
```

```
end

t  = subs(collocate(t),solution);
```

```
x1 = collocate(x1opt);
x2 = collocate(x2opt);
x3 = collocate(x3opt);
x7 = collocate(x7opt);
x8 = collocate(x8opt);
```

117.5 Plot result

```
subplot(2,1,1)
plot(t,x1,'*-',t,x2,'*-',t,x3,'*-');
legend('x1','x2','x3');
title('Transfer Min Swing state variables');

subplot(2,1,2)
plot(t,x7,'+-',t,x8,'+-');
legend('x7','x8');
title('Transfer Min Swing control');
```



118 Tubular Reactor

Global Optimization of Chemical Processes using Stochastic Algorithms 1996, Julio R. Banga, Warren D. Seider

Case Study V: Global optimization of a bifunctional catalyst blend in a tubular reactor

118.1 Problem Description

Luus et al and Luus and Bojkov consider the optimization of a tubular reactor in which methylcyclopentane is converted into benzene. The blend of two catalysts, for hydrogenation and isomerization is described by the mass fraction u of the hydrogenation catalyst. The optimal control problem is to find the catalyst blend along the length of the reactor defined using a characteristic reaction time t in the interval 0 <= t <= t_f where t_f = 2000 gh/mol corresponding to the reactor exit such that the benzene concentration at the exit of the reactor is maximized.

Find u(t) to maximize:

$$J = x_7(t_f)$$

Subject to:

$$\frac{dx_1}{dt} = -k_1 * x_1$$

$$\frac{dx_2}{dt} = k_1 * x_1 - (k_2 + k_3) * x_2 + k_4 * x_5$$

$$\frac{dx_3}{dt} = k_2 * x_2$$

$$\frac{dx_4}{dt} = -k_6 * x_4 + k_5 * x_5$$

$$\frac{dx_5}{dt} = k_3 * x_2 + k_6 * x_4 - (k_4 + k_5 + k_8 + k_9) * x_5 + k_7 * x_6 + k_{10} * x_7$$

$$\frac{dx_6}{dt} = k_8 * x_5 - k_7 * x_6$$

$$\frac{dx_7}{dt} = k_9 * x_5 - k_{10} * x_7$$

where xi, i = 1,..,7 are the mole fractions of the chemical species (i = 1 for methylcyclopentane, i = 7 for benzene), the rate constants are functions of the catalyst blend u(t):

$$k_i = c(i,1) + c(i,2) * u + c(i,3) * u^2 + c(i,4) * u^3$$

and the coefficients cij are given by Luus et al. The upper and lower bounds on the mass fraction of the hydrogenation catalyst are.

$$0.6 <= u <= 0.9$$

The initial vector of mole fractions is:

$$x(t_0) = [1\ 0\ 0\ 0\ 0\ 0\ 0]'$$

Reference: [4]

118.2 Problem setup

```
p = tomPhase('p', t, 0, 2000, n);
setPhase(p);

tomStates x1 x2 x3 x4 x5 x6 x7
tomControls u

% Collocate initial guess
x0 = {icollocate({x1 == x1opt; x2 == x2opt
    x3 == x3opt; x4 == x4opt; x5 == x5opt
    x6 == x6opt; x7 == x7opt})
    collocate(u == uopt)};

% Box constraints
cbox = {icollocate({
    0 <= x1 <= 1; 0 <= x2 <= 1
    0 <= x3 <= 1; 0 <= x4 <= 1
    0 <= x5 <= 1; 0 <= x6 <= 1
    0 <= x7 <= 1})
    0.6 <= collocate(u) <= 0.9};

% Boundary constraints
cbnd = initial({x1 == 1; x2 == 0
    x3 == 0; x4 == 0; x5 == 0
    x6 == 0; x7 == 0});

% ODEs and path constraints
```

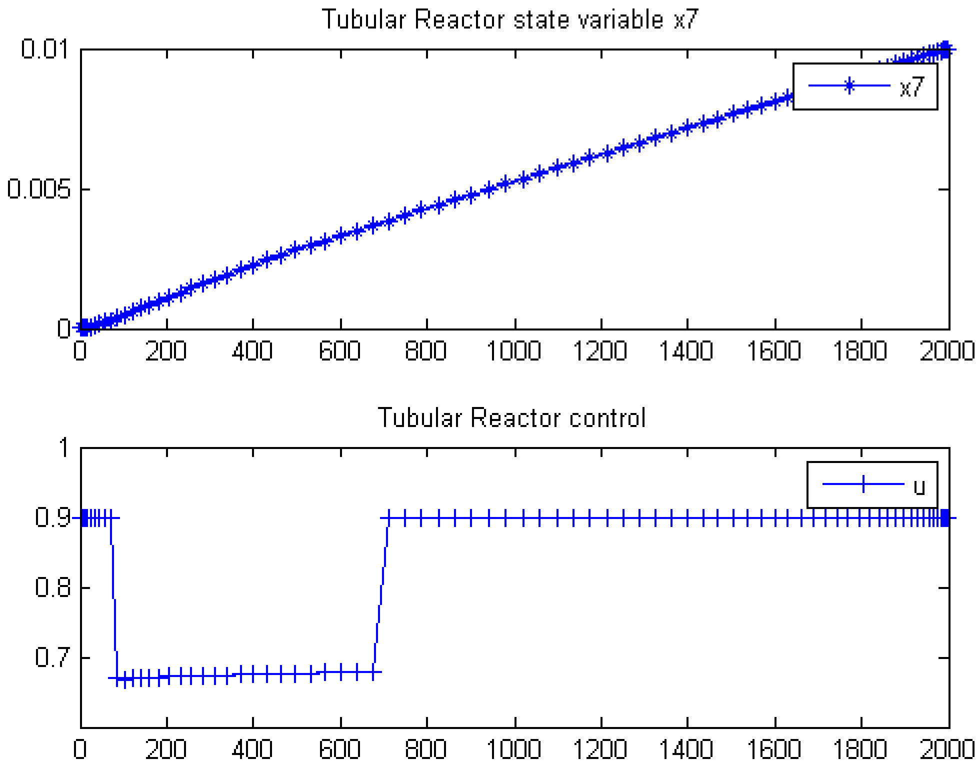
```
    uvec = [1, u, u.^2, u.^3];
    k    = (c'*uvec')';
    ceq = collocate({
        dot(x1) == -k(1)*x1
        dot(x2) ==  k(1)*x1-(k(2)+k(3))*x2+k(4)*x5
        dot(x3) ==  k(2)*x2
        dot(x4) == -k(6)*x4+k(5)*x5
        dot(x5) ==  k(3)*x2+k(6)*x4-(k(4)+k(5)+k(8)+k(9))*x5+...
        k(7)*x6+k(10).*x7
        dot(x6) == k(8)*x5-k(7)*x6
        dot(x7) == k(9)*x5-k(10)*x7});

    % Objective
    objective = -final(x7);
    options = struct;
    options.name = 'Tubular Reactor';
    options.d2c = false;
    Prob = sym2prob('con',objective,{cbox, cbnd, ceq}, x0, options);
    Prob.xInit = 35; % Use 35 random starting points.
    Prob.GO.localSolver = 'snopt';
    if n<=10
        Result = tomRun('multiMin', Prob, 1);
    else
        Result = tomRun('snopt', Prob, 1);
    end
    solution = getSolution(Result);

    % Store optimum for use in initial guess
    x1opt = subs(x1,solution);
    x2opt = subs(x2,solution);
    x3opt = subs(x3,solution);
    x4opt = subs(x4,solution);
    x5opt = subs(x5,solution);
    x6opt = subs(x6,solution);
    x7opt = subs(x7,solution);
    uopt  = subs(u,solution);
```

```
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Tubular Reactor - Trial 7      f_k      -0.010036498851799764
                                      sum(|constr|)      0.000000000086184773
                              f(x_k) + sum(|constr|)     -0.010036498765614991

Solver: multiMin with local solver snopt.  EXIT=0.  INFORM=0.
Find local optima using multistart local search
Did 35 local tries. Found 1 global, 33 minima. TotFuncEv 1546. TotConstrEv 1477
```

```
FuncEv 1546 GradEv 1476 ConstrEv 1477 ConJacEv   37 Iter  921
CPU time: 11.156250 sec. Elapsed time: 11.485000 sec.


===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Tubular Reactor                f_k      -0.009997531214886855
                                      sum(|constr|)      0.000000159449045791
                             f(x_k) + sum(|constr|)     -0.009997371765841064
                                             f(x_0)     -0.010036498851799764

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv   28 GradEv   26 ConstrEv   26 ConJacEv   26 Iter   18 MinorIter  548
CPU time: 3.906250 sec. Elapsed time: 4.000000 sec.


end
```

118.3 Plot result

```
t  = collocate(t);
x7 = collocate(x7opt);
u  = collocate(uopt);

subplot(2,1,1)
plot(t,x7,'*-');
legend('x7');
title('Tubular Reactor state variable x7');

subplot(2,1,2)
plot(t,u,'+-');
legend('u');
title('Tubular Reactor control');
```


Tubular Reactor state variable x7
0.01
0.005
0
0
200
400
600
800
1000
1200
1400
1600
1800
2000
x7
Tubular Reactor control
1
0.9
0.8
0.7
0
200
400
600
800
1000
1200
1400
1600
1800
2000
u

119 Turbo Generator

OCCAL - A Mixed symbolic-numeric Optimal Control CALcluator

Section 4 Example 1

119.1 Problem Formulation

Find u over t in [0; t] to minimize

$$J = \int_0^t (alpha_1 * ((x_1 - x_1^s)^2 + (x_4 - x_4^s)^2) +$$
$$alpha_2 * x_2^2 + alpha_3 * (x_3 - x_3^s)^2 +$$
$$beta_1 * (u_1 - u_1^s)^2 + beta_2 * (u_2 - u_2^s)^2) \mathrm{d}t$$

subject to:

$$\frac{dx_1}{dt} = x_2 * x_4$$
$$\frac{dx_2}{dt} = \frac{1}{M} * (u_1 - s_4 * x_1 * x_4 - s_5 * x_1 * x_3 - kappa_d * x_2$$
$$\frac{dx_3}{dt} = u_2 - A * x_3 + c * x_4$$
$$\frac{dx_4}{dt} = -x_1 * x_2$$

The initial condition are:

$$x(0) = [x_1^s \ x_2^s \ x_3^s \ x_4^s]$$
$$x_{1:4}^s = [0.60295 \ 0.0 \ 1.87243 \ 0.79778]$$
$$alpha = [2.5 \ 1.0 \ 0.1]$$
$$beta = [1.0 \ 1.0]$$
$$M = 0.04225$$
$$s_{4:5} = [0.0 \ 0.0]$$
$$c = 0$$

$$A = 0.17$$

$$u^s_{1:2} = [0.80 \ 0.73962]$$

$$kappa_d = 0.02535$$

Reference: [28]

119.2 Problem setup

```
toms t
p = tomPhase('p', t, 0, 20, 30);
setPhase(p);

tomStates x1 x2 x3 x4
tomControls u1 u2

% Initial guess
x0i = [0.60295;0;1.87243;0.79778];
x0 = {icollocate({x1 == x0i(1); x2 == x0i(2)
    x3 == x0i(3); x4 == x0i(4)})
    collocate({u1 == 0; u2 == 0})};

% Boundary constraints
cbnd = initial({x1 == x0i(1); x2 == x0i(2)
    x3 == x0i(3); x4 == x0i(4)});

% ODEs and path constraints
u1s     = 0.80;    u2s     = 0.73962;
A       = 0.17;    c       = 0;
s4      = 0;       s5      = 0;
M       = 0.04225; alpha1  = 2.5;
alpha2  = 1.0;     alpha3  = 0.1;
beta1   = 1.0;     beta2   = 1.0;
kappa_d = 0.02535;

ceq = collocate({dot(x1) == x2.*x4
    dot(x2) == 1/M.*(u1-s4*x1.*x4-s5*x1.*x3-kappa_d*x2)
    dot(x3) == u2-A*x3+c*x4; dot(x4) == -x1.*x2});

% Objective
objective = integrate(alpha1*( (x1-x0i(1)).^2 +  ...
    (x4-x0i(4)).^2) + alpha2*x2.^2 + alpha3*(x3-x0i(3)).^2 + ...
    beta1*(u1-u1s).^2 + beta2*(u2-u2s).^2);
```

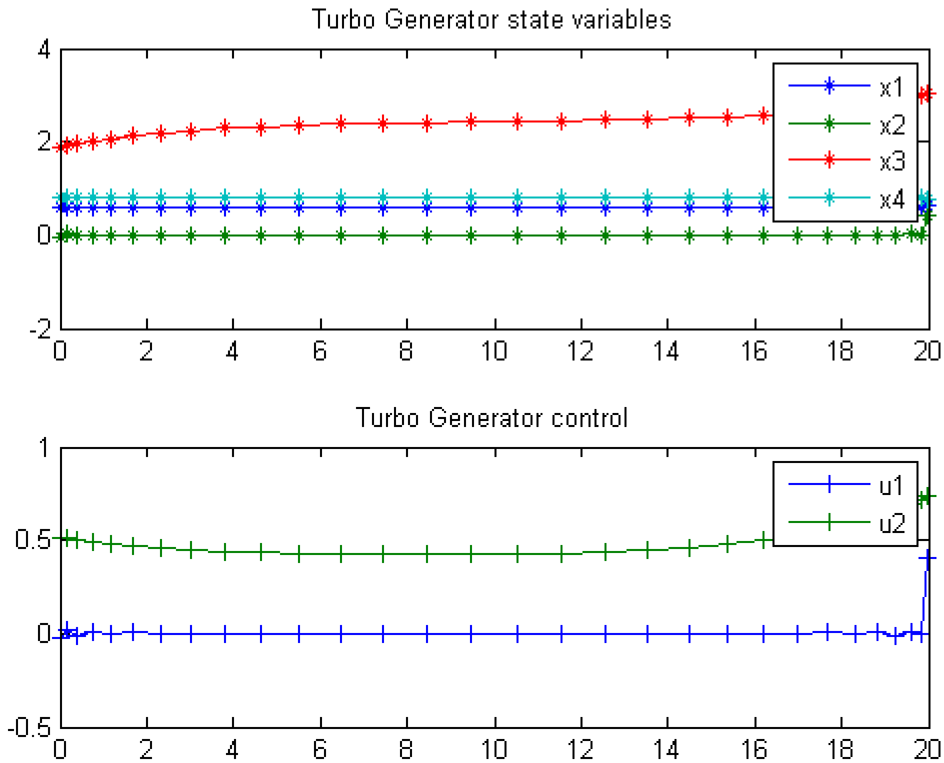
119.3 Solve the problem

```
options = struct;
options.name = 'Turbo Generator';
solution = ezsolve(objective, {cbnd, ceq}, x0, options);
t  = subs(collocate(t),solution);
x1 = subs(collocate(x1),solution);
x2 = subs(collocate(x2),solution);
x3 = subs(collocate(x3),solution);
x4 = subs(collocate(x4),solution);
u1 = subs(collocate(u1),solution);
u2 = subs(collocate(u2),solution);
```

```
Problem type appears to be: qpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Turbo Generator                f_k      15.019841547670836000
                                       sum(|constr|)      0.000000000046598417
                              f(x_k) + sum(|constr|)     15.019841547717435000
                                              f(x_0)    -57.012069754799995000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv   35 ConJacEv   35 Iter   23 MinorIter  117
CPU time: 0.125000 sec. Elapsed time: 0.141000 sec.
```

119.4 Plot result

```
subplot(2,1,1)
plot(t,x1,'*-',t,x2,'*-',t,x3,'*-',t,x4,'*-');
legend('x1','x2','x3','x4');
title('Turbo Generator state variables');

subplot(2,1,2)
plot(t,u1,'+-',t,u2,'+-');
legend('u1','u2');
title('Turbo Generator control');
```


Turbo Generator state variables
x1
x2
x3
x4
Turbo Generator control
u1
u2

120 Two-Link Robot

120.1 Problem description

Singular time-optimal 2 Link robot control

From the paper: L.G. Van Willigenburg, 1991, Computation of time-optimal controls applied to rigid manipulators with friction, Int. J. Contr., Vol. 54, no 5, pp. 1097-1117

Programmers: Gerard Van Willigenburg (Wageningen University) Willem De Koning (retired from Delft University of Technology)

120.2 Problem setup

```
% Array with consecutive number of collocation points
narr = [20 40];

toms t t_f % Free final time

for n=narr

    p = tomPhase('p', t, 0, t_f, n);
    setPhase(p)

    tomStates x1 x2 x3 x4
    tomControls u1 u2

    % Initial & terminal states
    xi = [0;   0; 0; 0];
    xf = [1.5; 0; 0; 0];

    % Initial guess
    if n==narr(1)
        x0 = {t_f==1; icollocate({x1 == xf(1); x2 == xf(2)
            x3 == xf(3); x4 == xf(4)})
            collocate({u1 == 0; u2 == 0})};
    else
        x0 = {t_f==tfopt; icollocate({x1 == xopt1; x2 == xopt2
            x3 == xopt3; x4 == xopt4})
            collocate({u1 == uopt1; u2 == uopt2})};
    end

    % Box constraints
```

```
cbox = {0.75 <= t_f <= 1.5; -25 <= collocate(u1) <= 25
    -9 <= collocate(u2) <= 9};

% Boundary constraints
cbnd = {initial({x1 == xi(1); x2 == xi(2)
    x3 == xi(3); x4 == xi(4)})
    final({x1 == xf(1); x2 == xf(2)
    x3 == xf(3); x4 == xf(4)})};

% ODEs and path constraints
% Robot parameters
mm11 = 5.775; mm12 = 0.815; mm22 = 0.815;
hm11 = 1.35;  m1   = 30.0;  m2   = 15;

% Variables for dynamics
c1  = cos(x1);        c2  = cos(x2);
s2  = sin(x2);        c12 = cos(x1+x2);
ms1 = mm11+2*hm11*c2; ms2 = mm12+hm11*c2;

mdet  = ms1.*mm22-ms2.*ms2;
ms11 = mm22./mdet; ms12=-ms2./mdet; ms22=ms1./mdet;

qg1 = -hm11*s2.*(x4.*x4+2*x3.*x4);
qg2 = hm11*s2.*x3.*x3;

dx1 = x3; dx2=x4;
dx3 = ms11.*(u1-qg1)+ms12.*(u2-qg2);
dx4 = ms12.*(u1-qg1)+ms22.*(u2-qg2);

ceq = collocate({
    dot(x1) == dx1
    dot(x2) == dx2
    dot(x3) == dx3
    dot(x4) == dx4});

% Objective
objective = t_f;
```

120.3 Solve the problem

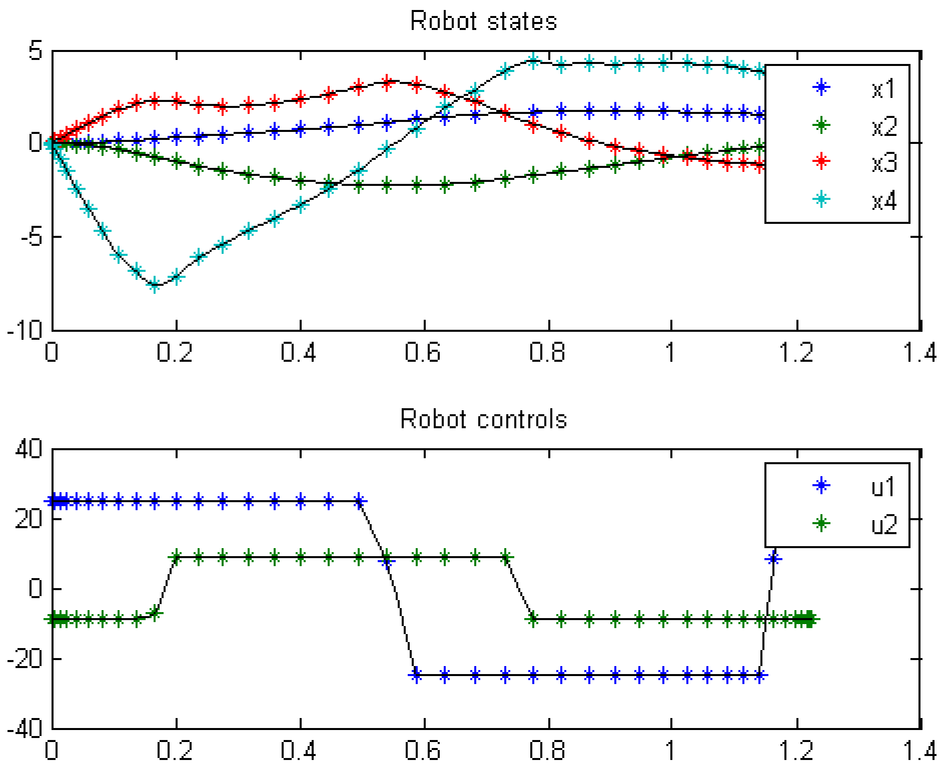
```
options = struct;
options.name = '2-Link-Robot';
solution = ezsolve(objective, {cbox, cbnd, ceq}, x0, options);

tfopt = subs(t_f,solution);
xopt1 = subs(x1,solution);
xopt2 = subs(x2,solution);
```

```
    xopt3 = subs(x3,solution);
    xopt4 = subs(x4,solution);
    uopt1 = subs(u1,solution);
    uopt2 = subs(u2,solution);
```

```
Problem type appears to be: lpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: 2-Link-Robot                  f_k       1.225664453973471300
                                       sum(|constr|)      0.000003477368351302
                              f(x_k) + sum(|constr|)      1.225667931341822600
                                              f(x_0)      1.000000000000000000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv 2552 ConJacEv 2552 Iter  568 MinorIter 4959
CPU time: 7.765625 sec. Elapsed time: 7.953000 sec.


Problem type appears to be: lpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: 2-Link-Robot                  f_k       1.223303478413072100
                                       sum(|constr|)      0.000000031552847509
                              f(x_k) + sum(|constr|)      1.223303509965919500
                                              f(x_0)      1.225664453973471300

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv   44 ConJacEv   44 Iter   16 MinorIter  358
CPU time: 0.406250 sec. Elapsed time: 0.406000 sec.
```

```
end

figure(1)
subplot(2,1,1);
ezplot([x1; x2; x3; x4]); legend('x1','x2','x3','x4');
title('Robot states');
```

```
subplot(2,1,2);
ezplot([u1; u2]); legend('u1','u2');
title('Robot controls');
```



121 Two-Link Robotic Arm

ITERATIVE DYNAMIC PROGRAMMING, REIN LUUS

12.4.2 Example 2: Two-link robotic arm

CHAPMAN & HALL/CRC Monographs and Surveys in Pure and Applied Mathematics

121.1 Problem Formulation

Find u over t in [0; t_F] to minimize

$$J = t_F$$

subject to:

$$\frac{dx_1}{dt} = \frac{sin(x_3) * (\frac{9}{4} * cos(x_3) * x_1^2 + 2 * x_2^2) + \frac{4}{3} * (u_1 - u_2) - \frac{3}{2} * cos(x_3) * u_2}{\frac{31}{36} + \frac{9}{4} * sin(x_3)^2}$$
$$\frac{dx_2}{dt} = -\frac{sin(x_3) * (\frac{7}{2} * x_1^2 + \frac{9}{4} * cos(x3) * x_2^2) - \frac{7}{3} * u_2 + \frac{3}{2} * cos(x_3) * (u_1 - u_2)}{\frac{31}{36} + \frac{9}{4} * sin(x_3)^2}$$
$$\frac{dx_3}{dt} = x_2 - x_1$$
$$\frac{dx_4}{dt} = x_1$$

The initial condition are:

$$x(0) = [0\ 0\ 0.5\ 0]$$
$$x(t_F) = [0\ 0\ 0.5\ 0.522]$$
$$-1 <= u(1:2) <= 1$$

Reference: [25]

121.2 Problem setup

```
toms t t_f
p = tomPhase('p', t, 0, t_f, 30);
setPhase(p);

tomStates x1 x2 x3 x4
tomControls u1 u2

% Initial guess
x0 = {t_f == 3
    icollocate({x1 == 0; x2 == 0
    x3 == 0.5; x4 == 0.522})
    collocate({u1 == 1-2*t/t_f
    u2 == 1-2*t/t_f})};

% Box constraints
cbox = {2.6 <= t_f <= 100
    -1  <= collocate(u1) <= 1
    -1  <= collocate(u2) <= 1};

% Boundary constraints
cbnd = {initial({x1 == 0; x2 == 0
    x3 == 0.5; x4 == 0})
    final({x1 == 0; x2 == 0
    x3 == 0.5; x4 == 0.522})};

% ODEs and path constraints
ceq = collocate({
    dot(x1) == ( sin(x3).*(9/4*cos(x3).*x1.^2+2*x2.^2) ...
    +4/3*(u1-u2) - 3/2*cos(x3).*u2 )./ (31/36 + 9/4*sin(x3).^2)
    dot(x2) == -( sin(x3).*(7/2*x1.^2 + 9/4*cos(x3).*x2.^2) ...
    - 7/3*u2 + 3/2*cos(x3).*(u1-u2) )./ (31/36 + 9/4*sin(x3).^2)
    dot(x3) == x2-x1
    dot(x4) == x1});

% Objective
objective = t_f;
```

121.3 Solve the problem

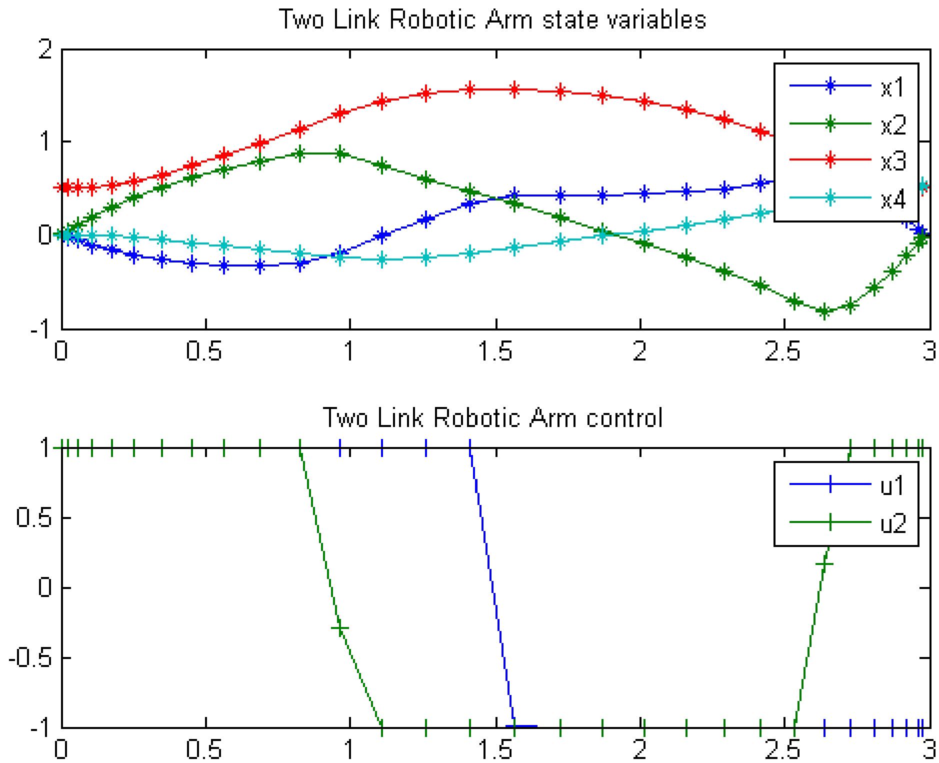
```
options = struct;
options.name = 'Two Link Robotic Arm';
solution = ezsolve(objective, {cbox, cbnd, ceq}, x0, options);
t  = subs(collocate(t),solution);
x1 = subs(collocate(x1),solution);
x2 = subs(collocate(x2),solution);
```

```
x3 = subs(collocate(x3),solution);
x4 = subs(collocate(x4),solution);
u1 = subs(collocate(u1),solution);
u2 = subs(collocate(u2),solution);
```

```
Problem type appears to be: lpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Two Link Robotic Arm           f_k       2.983364855223869000
                                       sum(|constr|)      0.000000154455635731
                              f(x_k) + sum(|constr|)      2.983365009679504800
                                              f(x_0)      3.000000000000000000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv   20 ConJacEv   20 Iter   16 MinorIter  278
CPU time: 0.203125 sec. Elapsed time: 0.219000 sec.
```

121.4 Plot result

```
subplot(2,1,1)
plot(t,x1,'*-',t,x2,'*-',t,x3,'*-',t,x4,'*-');
legend('x1','x2','x3','x4');
title('Two Link Robotic Arm state variables');

subplot(2,1,2)
plot(t,u1,'+-',t,u2,'+-');
legend('u1','u2');
title('Two Link Robotic Arm control');
```



Two Link Robotic Arm state variables
2
1
0
-1
0
0.5
1
1.5
2
2.5
3
x1
x2
x3
x4
Two Link Robotic Arm control
1
0.5
0
-0.5
-1
0
0.5
1
1.5
2
2.5
3
u1
u2

122 Two-Phase Schwartz

Users Guide for dyn.Opt, Example 4

Schwartz, A. L., Theory and Implementation of Numerical Methods based on Runge-Kutta Integration for Solving Optimal Control Problems. Ph.D. Dissertation, University of California, Berkeley, 1989

122.1 Problem Formulation

Find u over t in [0; 2.9] to minimize

$$J = 5 * (x_1(t_F)^2 + x_2(t_F)^2)$$

subject to:

$$\frac{dx_1}{dt} = x_2$$
$$\frac{dx_2}{dt} = u - 0.1 * (1 + 2 * x_1^2) * x_2$$
$$x(0) = [1\ 1]$$

and path constraints for t<1:

$$1 - 9 * (x_1 - 1)^2 - (\frac{x_2 - 0.4}{0.3})^2 <= 0$$
$$-0.8 - x_2 <= 0 \ \ --> \ \ -0.8 <= x_2$$
$$-1 <= u <= 1, (t < 1)$$

Reference: [16]

122.2 Problem setup

```
toms t1
p1 = tomPhase('p1', t1, 0, 1, 25);
toms t2
p2 = tomPhase('p2', t2, 1, 1.9, 25);
```

```
setPhase(p1);
tomStates x1p1 x2p1
tomControls up1

setPhase(p2);
tomStates x1p2 x2p2
tomControls up2

setPhase(p1);
% Initial guess
x01 = {icollocate({x1p1 == 1; x2p1 == 1})
    collocate(up1==0)};

% Box constraints
cbox1 = {-0.8 <= icollocate(x2p1)
    -1 <= collocate(up1) <= 1};

% Boundary constraints
cbnd1 = initial({x1p1 == 1; x2p1 == 1});

% ODEs and path constraints
ceq1 = collocate({
    dot(x1p1) == x2p1
    dot(x2p1) == up1 - 0.1*(1+2*x1p1.^2).*x2p1
    1-9*(x1p1-1).^2-((x2p1-0.4)/0.3).^2 <= 0});

setPhase(p2);
% Initial guess
x02 = {icollocate({x1p2 == 1; x2p2 == 1})
    collocate(up2==0)};

% Box constraints
cbox2 = {-50 <= collocate(up2) <= 50};

% ODEs and path constraints
ceq2 = collocate({
    dot(x1p2) == x2p2
    dot(x2p2) == up2-0.1*(1+2*x1p2.^2).*x2p2});

% Link phase
link = {final(p1,x1p1) == initial(p2,x1p2)
    final(p1,x2p1) == initial(p2,x2p2)};

% Objective
objective = 5*(final(p2,x1p2)^2+final(p2,x2p2)^2);
```

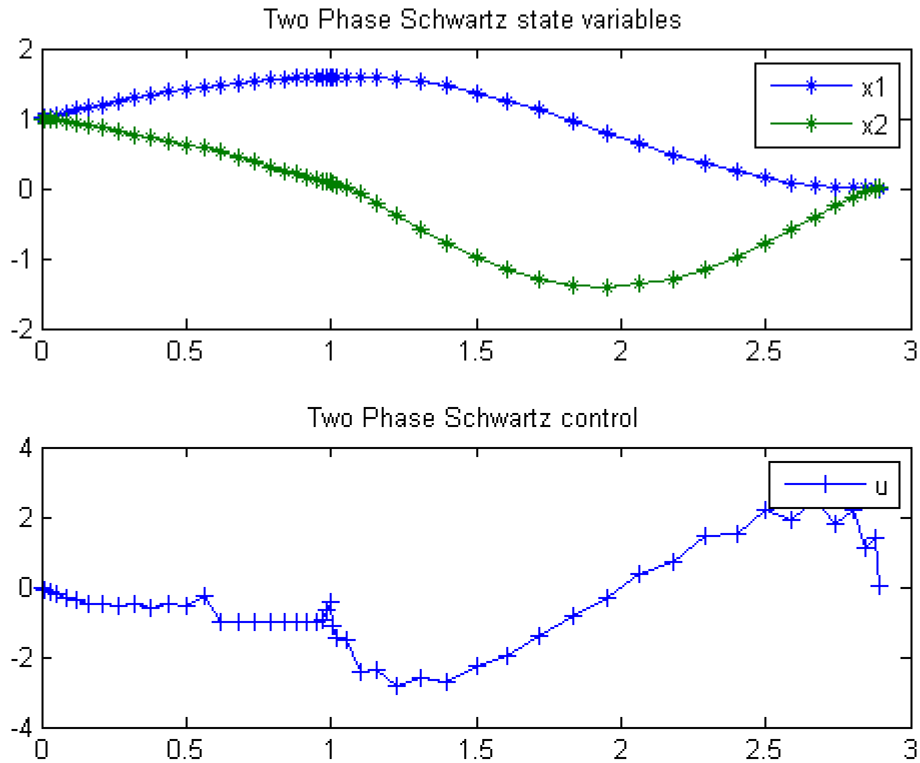
122.3 Solve the problem

```
options = struct;
options.name = 'Two Phase Schwartz';
constr = {cbox1, cbnd1, ceq1, cbox2, ceq2, link};
solution = ezsolve(objective, constr, {x01, x02}, options);

t = subs(collocate(p1,t1),solution);
t = [t;subs(collocate(p2,t2),solution)];
x1 = subs(collocate(p1,x1p1),solution);
x1 = [x1;subs(collocate(p2,x1p2),solution)];
x2 = subs(collocate(p1,x2p1),solution);
x2 = [x2;subs(collocate(p2,x2p2),solution)];
u = subs(collocate(p1,up1),solution);
u = [u;subs(collocate(p2,up2),solution)];
```

```
Problem type appears to be: qpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Two Phase Schwartz             f_k      -0.000000000000002541
                                       sum(|constr|)      0.000000000002833381
                              f(x_k) + sum(|constr|)      0.000000000002830840
                                              f(x_0)     10.000000000000014000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv   24 ConJacEv   24 Iter   18 MinorIter  361
CPU time: 0.156250 sec. Elapsed time: 0.156000 sec.
```

122.4 Plot result

```
subplot(2,1,1)
plot(t,x1,'*-',t,x2,'*-');
legend('x1','x2');
title('Two Phase Schwartz state variables');

subplot(2,1,2)
plot(t,u,'+-');
legend('u');
title('Two Phase Schwartz control');
```


Two Phase Schwartz state variables
x1
x2
2
1
0
-1
-2
0
0.5
1
1.5
2
2.5
3
Two Phase Schwartz control
u
4
2
0
-2
-4
0
0.5
1
1.5
2
2.5
3

123 Two Stage CSTR

ITERATIVE DYNAMIC PROGRAMMING, REIN LUUS

Section 6.3.1 Nonlinear two-stage CSTR system

CHAPMAN & HALL/CRC Monographs and Surveys in Pure and Applied Mathematics

123.1 Problem Description

The system consists of a series of two CSTRs, where there is a transportation delay tau = 0.1 from the first tank to the second. A truncated Taylor series expansion for the time delay.

Find u over t in [0; 2] to minimize

$$J = x_5(t_F)$$

subject to:

$$\frac{dx_1}{dt} = f_1$$
$$\frac{dx_2}{dt} = f_2$$
$$\frac{dx_3}{dt} = x_1 - x_3 - tau * f_1 - R_2 + 0.25$$
$$\frac{dx_4}{dt} = x_2 - 2 * x_4 - u_2 * (x_4 + 0.25) - tau * f_2 + R_2 - 0.25$$
$$\frac{dx_5}{dt} = x_1^2 + x_2^2 + x_3^2 + x_4^2 + 0.1 * (u_1^2 + u_2^2)$$

$$f_1 = 0.5 - x_1 - R_1$$
$$f_2 = -2 * (x_2 + 0.25) - u_1 * (x_2 + 0.25) + R_1$$
$$R_1 = (x_1 + 0.5) * exp(25 * \frac{x_2}{x_2 + 2})$$
$$R_2 = (x_3 + 0.25) * exp(25 * \frac{x_4}{x_4 + 2})$$

The state variables x1 and x3 are normalized concentration variables in tanks 1 and 2, respectively, and x2 and x4 are normalized temperature variables in tanks 1 and 2, respectively. The variable x5 is introduced to provide the performance index to be minimized.

The initial condition are:

$$x(0) = [0.15 \;\; -0.03 \; 0.10 \; 0 \; 0]$$

$$-0.5 <= u1 <= 0.5$$

$$-0.5 <= u2 <= 0.5$$

Reference: [25]

123.2 Problem setup

```
toms t
p = tomPhase('p', t, 0, 2, 20);
setPhase(p);

tomStates x1 x2 x3 x4 x5
tomControls u1 u2

xi = [0.15;-0.03;0.10;0;0];

% Initial guess
x0 = {icollocate({x1 == xi(1); x2 == xi(2)
    x3 == xi(3); x4 == xi(4); x5 == xi(5)})
    collocate({u1 == 0; u2 == 0})};

% Box constraints
cbox = collocate({-0.5 <= u1 <= 0.5
    -0.5 <= u2 <= 0.5});

% Boundary constraints
cbnd = initial({x1 == xi(1); x2 == xi(2)
    x3 == xi(3); x4 == xi(4); x5 == xi(5)});

% ODEs and path constraints
R1 = (x1 + 0.5).*exp(25*x2./(x2 + 2));
R2 = (x3 + 0.25).*exp(25*x4./(x4 + 2));
f1 = 0.5 - x1 - R1;
f2 = -2*(x2 + 0.25) - u1.*(x2 + 0.25) + R1;
tau = 0.1;
ceq = collocate({
```

```
    dot(x1) == f1; dot(x2) == f2
    dot(x3) == x1-x3-tau*f1-R2+0.25
    dot(x4) == x2-2*x4-u2.*(x4+0.25)-tau*f2+R2-0.25
    dot(x5) == x1.^2+ x2.^2+x3.^2+x4.^2+0.1*(u1.^2+u2.^2)});

% Objective
objective = final(x5);
```

123.3 Solve the problem

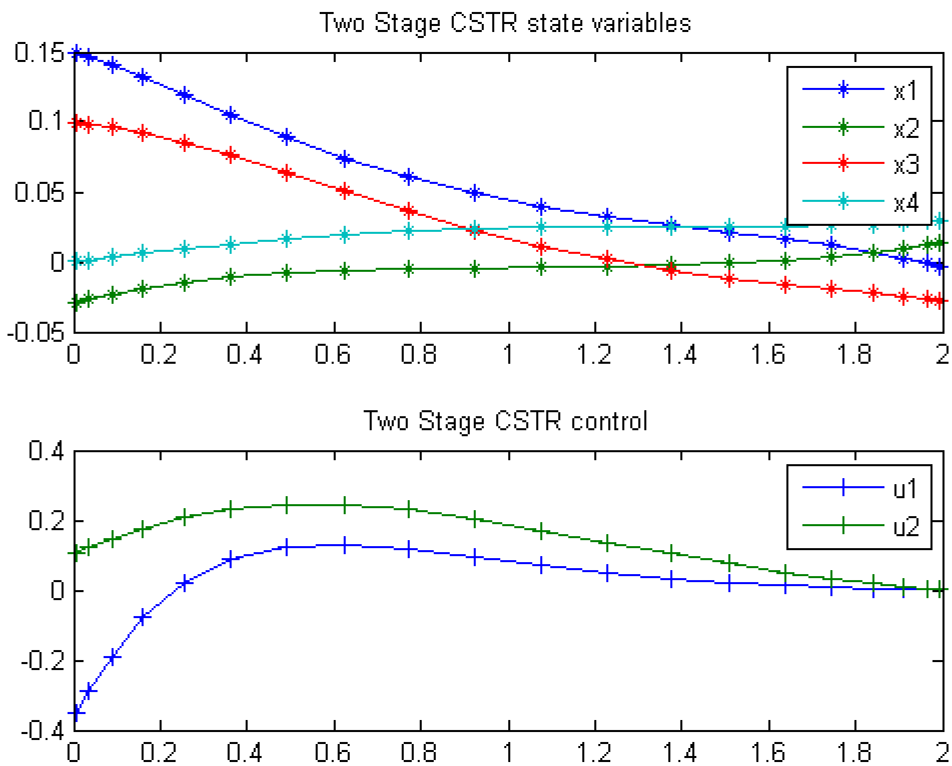
```
options = struct;
options.name = 'Two Stage CSTR';
solution = ezsolve(objective, {cbox, cbnd, ceq}, x0, options);
t  = subs(collocate(t),solution);
x1 = subs(collocate(x1),solution);
x2 = subs(collocate(x2),solution);
x3 = subs(collocate(x3),solution);
x4 = subs(collocate(x4),solution);
u1 = subs(collocate(u1),solution);
u2 = subs(collocate(u2),solution);
```

```
Problem type appears to be: lpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Two Stage CSTR                 f_k       0.023238023992802687
                                       sum(|constr|)      0.000000172957105729
                              f(x_k) + sum(|constr|)      0.023238196949908415
                                              f(x_0)      0.000000000000000000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv   26 ConJacEv   26 Iter   22 MinorIter  123
CPU time: 0.218750 sec. Elapsed time: 0.234000 sec.
```

123.4 Plot result

```
subplot(2,1,1)
plot(t,x1,'*-',t,x2,'*-',t,x3,'*-',t,x4,'*-');
legend('x1','x2','x3','x4');
title('Two Stage CSTR state variables');
```

```
subplot(2,1,2)
plot(t,u1,'+-',t,u2,'+-');
legend('u1','u2');
title('Two Stage CSTR control');
```



124 Van der Pol Oscillator

Restricted second order information for the solution of optimal control problems using control vector parameterization. 2002, Eva Balsa Canto, Julio R. Banga, Antonio A. Alonso Vassilios S. Vassiliadis

Case Study 6.1: van der Pol oscillator

This case study has been studied by several authors, for example Morison, Gritsis, Vassiliadis and Tanartkit and Biegler.

124.1 Problem Description

The dynamic optimization problem is to minimize:

$$J = x_3(t_f)$$

subject to:

$$\frac{dx_1}{dt} = (1 - x_2^2) * x_1 - x_2 + u$$
$$\frac{dx_2}{dt} = x_1$$
$$\frac{dx_3}{dt} = x_1^2 + x_2^2 + u^2$$
$$-0.3 <= u <= 1.0$$
$$x(t_0) = [0\ 1\ 0]'$$
$$t_f = 5$$

Reference: [31]

124.2 Problem setup

```
toms t
p = tomPhase('p', t, 0, 5, 60);
setPhase(p);

tomStates x1 x2 x3
tomControls u
```

```
% Initial guess
x0 = {icollocate({x1 == 0; x2 == 1; x3 == 0})
    collocate(u == -0.01)};

% Box constraints
cbox = {-10  <= icollocate(x1) <= 10
    -10  <= icollocate(x2) <= 10
    -10  <= icollocate(x3) <= 10
    -0.3 <= collocate(u)   <= 1};

% Boundary constraints
cbnd = initial({x1 == 0; x2 == 1; x3 == 0});

% ODEs and path constraints
ceq = collocate({dot(x1) == (1-x2.^2).*x1-x2+u
    dot(x2) == x1; dot(x3) == x1.^2+x2.^2+u.^2});

% Objective
objective = final(x3);
```

124.3 Solve the problem

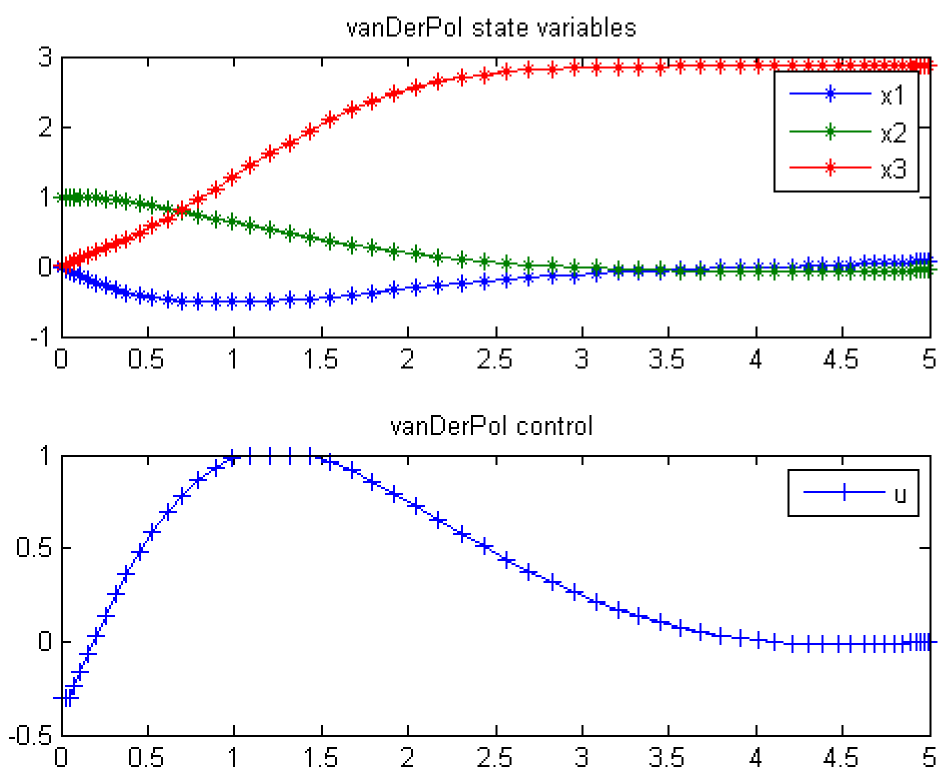
```
options = struct;
options.name = 'Van Der Pol';
solution = ezsolve(objective, {cbox, cbnd, ceq}, x0, options);
t  = subs(collocate(t),solution);
x1 = subs(collocate(x1),solution);
x2 = subs(collocate(x2),solution);
x3 = subs(collocate(x3),solution);
u  = subs(collocate(u),solution);
```

```
Problem type appears to be: lpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Van Der Pol                      f_k       2.867259538084708100
                                       sum(|constr|)      0.000000020744545091
                              f(x_k) + sum(|constr|)      2.867259558829253300
                                              f(x_0)      0.000000000000000000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied
```

```
FuncEv    1 ConstrEv   26 ConJacEv   26 Iter   23 MinorIter  348
CPU time: 0.593750 sec. Elapsed time: 0.594000 sec.
```

124.4 Plot result

```
subplot(2,1,1)
plot(t,x1,'*-',t,x2,'*-',t,x3,'*-');
legend('x1','x2','x3');
title('vanDerPol state variables');

subplot(2,1,2)
plot(t,u,'+-');
legend('u');
title('vanDerPol control');
```



125 Zermelos problem (version 1)

125.1 Problem description

Time-optimal aircraft heading through air in motion

Applied Optimal Control, Bryson & Ho, 1975. Problem 1 on page 77.

Programmers: Gerard Van Willigenburg (Wageningen University) Willem De Koning (retired from Delft University of Technology)

125.2 Problem setup

```
% Array with consecutive number of collocation points
narr = [20 40];

toms t t_f % Free final time

for n=narr

    p = tomPhase('p', t, 0, t_f, n);
    setPhase(p)

    tomStates x1 x2
    tomControls u1

    % Initial & terminal states
    xi = [-2;     0];
    xf = [0.5; -1.6];

    % Initial guess
    if n==narr(1)
        x0 = {t_f == 2; icollocate({x1 == xi(1); x2 == xi(2)})
            collocate({u1 == pi})};
    else
        x0 = {t_f == tfopt; icollocate({x1 == xopt1; x2 == xopt2})
            collocate({u1 == uopt1})};
    end

    % Box constraints
    cbox = {1 <= t_f <= 10};

    % Boundary constraints
```

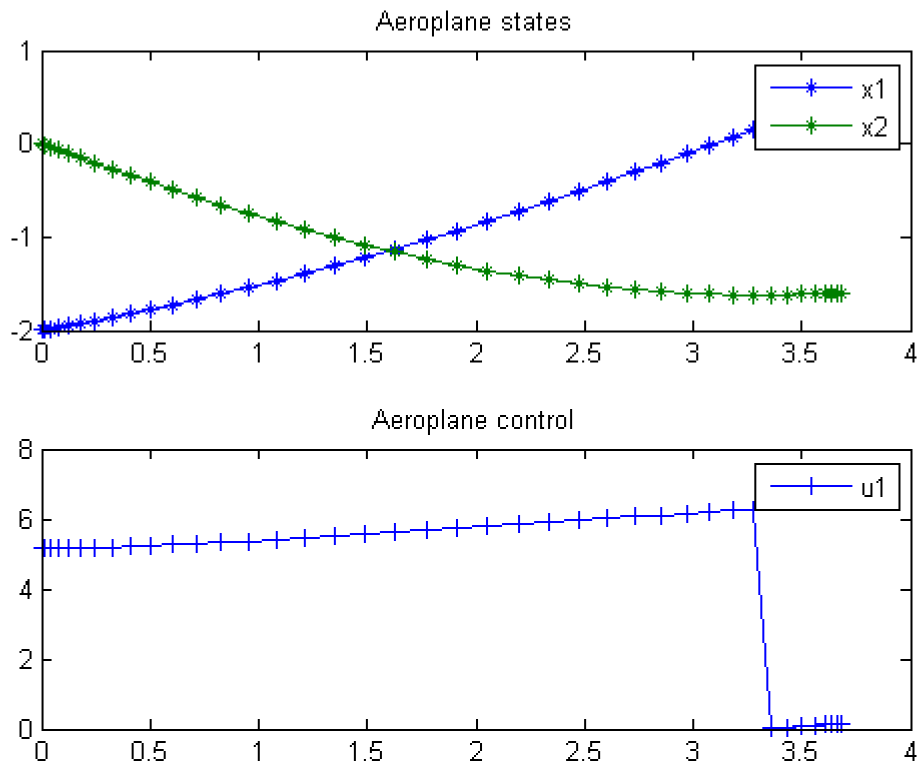
```
cbnd = {initial({x1 == xi(1); x2 == xi(2)});
    final({x1 == xf(1); x2 == xf(2)})};

% ODEs and path constraints
wh  = exp(-x1.*x1-x2.*x2+0.25); v=1;
dx1 = v*cos(u1)+x2.*wh; %  x2*wh: motion of air in x1 direction
dx2 = v*sin(u1)-x1.*wh; % -x1*wh: motion of air in x2 direction

ceq = collocate({
    dot(x1) == dx1
    dot(x2) == dx2});

% Objective
objective = t_f;
```

125.3 Solve the problem

```
options = struct;
options.name = 'Zermelo Flight Trajectory';
solution = ezsolve(objective, {cbox, cbnd, ceq}, x0, options);

tfopt = subs(t_f,solution);
xopt1 = subs(x1,solution);
xopt2 = subs(x2,solution);
uopt1 = subs(u1,solution);
```

```
Problem type appears to be: lpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Zermelo Flight Trajectory      f_k       3.682008465510111100
                                       sum(|constr|)      0.000000351412865501
                              f(x_k) + sum(|constr|)      3.682008816922976500
                                              f(x_0)      2.000000000000000000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv  163 ConJacEv  163 Iter   70 MinorIter  120
CPU time: 0.328125 sec. Elapsed time: 0.343000 sec.


Problem type appears to be: lpcon
Starting numeric solver
```

```
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Zermelo Flight Trajectory      f_k       3.682008477493101200
                                     sum(|constr|)      0.000000024815118568
                            f(x_k) + sum(|constr|)      3.682008502308219600
                                            f(x_0)      3.682008465510111100

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv   34 ConJacEv   34 Iter   31 MinorIter  116
CPU time: 0.234375 sec. Elapsed time: 0.235000 sec.


end

% Get solution
t  = subs(collocate(t),solution);
x1 = subs(collocate(x1),solution);
x2 = subs(collocate(x2),solution);
u1 = subs(collocate(u1),solution);

%Bound u1 to [0,2pi]
u1 = rem(u1,2*pi); u1 = (u1<0)*2*pi+u1;

% Plot final solution
figure(1); subplot(2,1,1)
plot(t,x1,'*-',t,x2,'*-');
legend('x1','x2');
title('Aeroplane states');

subplot(2,1,2)
plot(t,u1,'+-');
legend('u1');
title('Aeroplane control');
```


Aeroplane states
x1
x2
Aeroplane control
u1

126 Zermelos problem (version 2)

126.1 Problem description

Time-optimal crossing by boat of a river with a position dependent current stream.

Applied Optimal Control, Bryson & Ho, 1975. Example 1 on page 77.

Programmers: Gerard Van Willigenburg (Wageningen University) Willem De Koning (retired from Delft University of Technology)

126.2 Problem setup

```
% Array with consecutive number of collocation points
narr = [20 40];

toms t t_f % Free final time

for n=narr

    p = tomPhase('p', t, 0, t_f, n);
    setPhase(p)

    tomStates x1 x2
    tomControls u1

    % Initial & terminal states
    xi = [0;  0];
    xf = [31; 0];

    % Initial guess
    if n==narr(1)
        x0 = {t_f == 2; icollocate({x1 == xi(1); x2 == xi(2)})
            collocate({u1 == 0})};
    else
        x0 = {t_f == tfopt; icollocate({x1 == xopt1; x2 == xopt2})
            collocate({u1 == uopt1})};
    end

    % Box constraints
    cbox = {1 <= t_f <= 10};

    % Boundary constraints
```

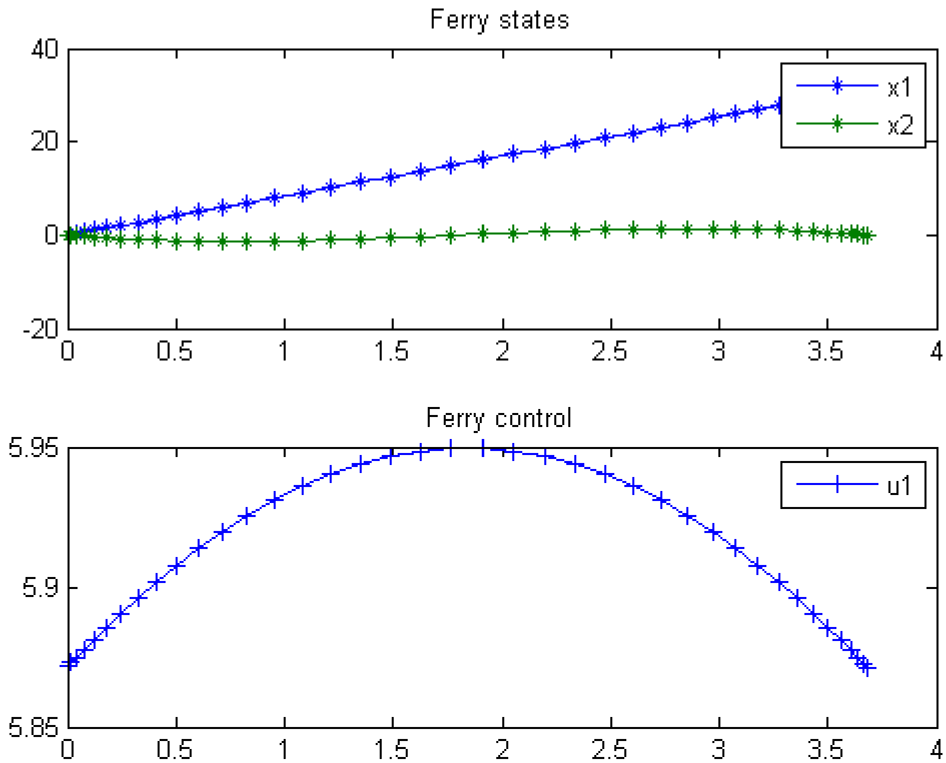
```
cbnd = {initial({x1 == xi(1); x2 == xi(2)});
    final({x1 == xf(1); x2 == xf(2)})};

% ODEs and path constraints
v = 9;
% No water motion in x1 direction
dx1 = v*cos(u1);
% Water motion in x2 direction: 5*sin(pi*x1/31)
dx2 = v*sin(u1)+5*sin(pi*x1/31);

ceq = collocate({
    dot(x1) == dx1
    dot(x2) == dx2});

% Objective
objective = t_f;
```

126.3 Solve the problem

```
options = struct;
options.name = 'Ferry trajectory optimization';
solution = ezsolve(objective, {cbox, cbnd, ceq}, x0, options);

tfopt = subs(t_f,solution);
xopt1 = subs(x1,solution);
xopt2 = subs(x2,solution);
uopt1 = subs(u1,solution);
```

```
Problem type appears to be: lpcon
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Ferry trajectory optimization  f_k       3.681324334091522500
                                       sum(|constr|)      0.000000334501118908
                              f(x_k) + sum(|constr|)      3.681324668592641300
                                              f(x_0)      2.000000000000000000

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv   84 ConJacEv   84 Iter   51 MinorIter   99
CPU time: 0.156250 sec. Elapsed time: 0.156000 sec.


Problem type appears to be: lpcon
```

```
Starting numeric solver
===== * * * =================================================================== * * *
TOMLAB - Tomlab Optimization Inc. Development license  999001. Valid to 2011-02-05
=====================================================================================
Problem: ---  1: Ferry trajectory optimization  f_k       3.681324335373935800
                                       sum(|constr|)      0.000002472599377322
                              f(x_k) + sum(|constr|)      3.681326807973313000
                                              f(x_0)      3.681324334091522500

Solver: snopt.  EXIT=0.  INFORM=1.
SNOPT 7.2-5 NLP code
Optimality conditions satisfied

FuncEv    1 ConstrEv   40 ConJacEv   40 Iter   32 MinorIter  112
CPU time: 0.203125 sec. Elapsed time: 0.203000 sec.


end

% Get solution
t  = subs(collocate(t),solution);
x1 = subs(collocate(x1),solution);
x2 = subs(collocate(x2),solution);
u1 = subs(collocate(u1),solution);

%Bound u1 to [0,2pi]
u1 = rem(u1,2*pi); u1 = (u1<0)*2*pi+u1;

% Plot final solution
figure(1)
subplot(2,1,1)
plot(t,x1,'*-',t,x2,'*-');
legend('x1','x2');
title('Ferry states');

subplot(2,1,2)
plot(t,u1,'+-');
legend('u1');
title('Ferry control');
```


Ferry states
40
20
0
-20
0
0.5
1
1.5
2
2.5
3
3.5
4
x1
x2
Ferry control
5.95
5.9
5.85
0
0.5
1
1.5
2
2.5
3
3.5
4
u1

References

[1] E. Balsa-Canto, J. R. Banga, A. A. Alonso, and V. S. Vassiliadis. Restricted second order information for the solution of optimal control problems using control vector parameterization. *Journal of Process Control*, 12:243–255, 2002.

[2] E. Balsa-Canto, J. R. Banga, V. S. Vassiliadis, and A. A. Alonso. Dynamic optimization of chemical and biochemical processes using restricted second-order information. *Computers and Chemical Engineering*, 25:539–546, 2001.

[3] J. R. Banga, E. Balsa-Canto, C. G. Moles, and A. A. Alonso. Dynamic optimization of bioprocesses: efficient and robust numerical strategies. *Journal of Biotechnology*, 2003.

[4] J. R. Banga and W. D. Seider. Global optimization of chemical processes using stochastic algorithms. *State of the Art in Global Optimization: Computational Methods and Applications*, pages 563–583, 1996.

[5] J. Betts. *SOCS Release 6.5.0*, 2007.

[6] R. Bhattacharya. *OPTRAGEN 1.0: A MATLAB Toolbox for Optimal Trajectory Generation.* College Station, TX 77843-3141, USA, March 2006.

[7] U. Boscain. A short introduction to optimal control. Technical report, S.I.S.S.A., Trieste, Italy, 2004.

[8] A. Bressan. Viscocity solutions of hamilton-jacobi equations and optimal control problems. Technical report, S.I.S.S.A., Trieste, Italy, 2003.

[9] A. E. Bryson. *Dynamic Optimization.* Addison Wesley Longman, Menlo Park, CA, USA, 1999.

[10] E. F. Carrasco and J. R. Banga. Dynamic optimization of batch reactors using adaptive stochastic algorithms. *Ind. Eng. Chem. Res.*, 36:2252–2261, 1997.

[11] E. F. Carrasco and J. R. Banga. A hybrid method for the optimal control of chemical processes. In *UKACC International Conference on CONTROL 98*, University of Wales, Swansea, UK, September 1998.

[12] D. E. Chang, Petit N., and Rouchon P. Time-optimal control of a particle in a dielectrophoretic system. *IEEE TRANSACTIONS ON AUTOMATIC CONTROL*, 51, 2006.

[13] M. Cizniar, M. Fikar, and M. A. Latifi. *MATLAB Dynamic Optimisation Code DYNOPT. User's Guide.* Bratislava, Slovak Republic, 2006.

[14] E. D. Dolan and J. J. More. Benchmarking optimization software with cops. Technical report, ARGONNE NATIONAL LABORATORY, 9700 South Cass Avenue, Argonne, Illinois 60439, January 2001.

[15] B. J. Driessen and Sadegh N. Minimum-time control of systems with coloumb friction: Near global optima via mixed integer linear programming. Technical report, Structural Dynamics Department, Sandia National Labs, June 2000.

[16] B. C. Fabien. *A Java Application for the Solution of Optimal Control Problems.* Stevens Way, Box 352600 Seattle, WA 98195, USA, 1998.

[17] O. S. Fard and A. H. Borzabadi. Optimal control problem, quasi-assignment problem and genetic algorithm. In *Proceedings of world academcy of science, engineering and technology*, January 2007.

[18] I. Ioslovich and P. Gutman. On smooth optimal control determination. Technical report, Technion, Israel Institute of Technology, February 2004.

[19] L. S. Jennings and M. E. Fisher. *MISER3: Optimal Control Toolbox User Manual, Matlab Beta Version 2.0*. Nedlands, WA 6907, Australia, 2002.

[20] F. M. Kirillova. Optimal on-line control and classical regulation problem. Technical report, Institute of Mathematics National Academy of Sciences of Belarus, October 2006.

[21] G. Kurina. On some linear-quadratic optimal control problems for descriptor systems. Technical report, Department of Mathematics, Stockholm University, February 2006.

[22] J. Liang. Lecture notes for ece/mae 7360, robust and optimal control (part 2). Technical report, Utah State University at Logan, November 2003.

[23] J. Liang, M. Meng, Y. Chen, and R. Fullmer. Solving tough optimal control problems by network enabled optimization server (neos). Technical report, School of Engineering, Utah State University USA, Chinene University of Hong Kong China, 2003.

[24] A. E. B. Lim, Y. Q. Liu, K. L. Teo, and Moore J. B. Linear-quadratic optimal control with integral quadratic constraints. *Optimal contorl and applications and methods*, 20:79–92, 1999.

[25] R. Luus. *Iterative dynamic programming*. Chapman and Hall/CRC, 2002.

[26] J. Lygeros. Minimum cost optimal control: An application to flight level tracking. Technical report, Department of Engineering, University of Cambridge, Cambridge, UK, 2003.

[27] A. V. Rao and K. D. Mease. Eigenvector approximate dichotomic basis method for solving hyper-sensitive optimal control problems. *Optimal Control Applications and Methods*, 21:1–19, 2000.

[28] R. Schöpf and P. Deuflhard. *OCCAL: A mixed symbolic-numeric Optimal Control CALculator*. Heilbronner Str. 10, W-1000 Berlin 31, 1991.

[29] A. Schwartz, E. Polak, and Y. Chen. *RIOTS95: AMatlab Toolbox for Solving Optimal Control Problems*, May 1997.

[30] K. L. Teo, K. K. Leong, and Goh G. J. Nonlinearly constrained optimal control problems involving piecewise smooth controls. Technical report, Australian Research Council, 1990.

[31] V. S. Vassiliadis, J. R. Banga, and E. Balsa-Canto. Second-order sensitivities of general dynamic systems with application to optimal control problems. *Chemical Engineering Science*, 54:3851–3860, 1999.

[32] T. Veeraklaew and S. Malisuwan. The direct approach of general dynamic optimal control: Application on general software. *International Journal of The Computer, the Internet and Management*, 14:2:82–87, August 2006.

[33] O. von Stryk. *User's Guide for DIRCOL*, 1999.

[34] M. Weiser. Function space complementarity methods for optimal control problems. Technical report, Fachbereich Mathematik und Informatik der Freien Universitäat Berlin, February 2001.

[35] E. G. Wiens. Egwald mathematics: Optimal control, intercept missile. Technical report, Web, April 2003.

[36] L. Youdong and M. A. Stadtherr. Deterministic global optimization of nonlinear dynamic systems. Technical report, Department of Chemical and Biomolecular Engineering, University of Notre Dame, August 2006.