Extra handout for Nonlinear Control

- *Overview* of Optimica by Johan Åkesson
  - Remark: Optimica will be used in a computer exercise, but you need not to learn any details about Modelica, AMPL, IPOPT etc.
- Extra example
  - Minimum-time problem for double integrator

Bring this hand-out to computer exercise CE5

Dynamic Optimization with Optimica, AMPL and IPOPT

Johan Åkesson

Dept. of Automatic Control
Lund Institute of Technology

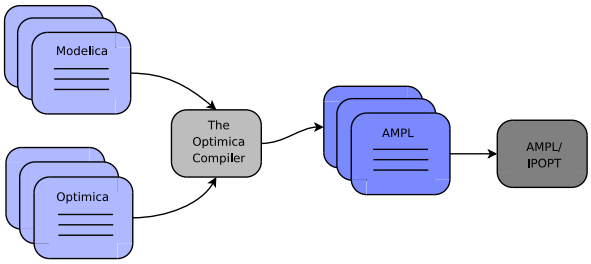
Outline

- Motivation of Optimica
- System overview
- The Optimica language extension
- AMPL
- CSTR example in Optimica
- Working with Dymola and Optimica

Optimica – Motivation

- It is often cumbersome to use numerical optimization algorithms for large-scale systems
  - Code the dynamics in C or FORTRAN: inconvenient modeling formalism
  - Compute and code derivatives
  - Difficult to reuse code
- High-level modeling language Modelica increasingly used
  - Object-oriented component-based modeling
  - Libraries and reuse of code
- Modelica does not have language support for optimization
- The JModelica project and Optimica
  - Develop a prototype Modelica compiler in JastAdd/Java
  - Extend Modelica to support dynamic optimization – Optimica
    - Modelica used to express dynamics
    - Optimica used to express cost functions, constraints etc.

System Overview



- The Optimica compiler reads Modelica and Optimica models and outputs a set of AMPL files
- The AMPL files can be executed by the AMPL tools
- The problem is solved numerically by IPOPT

The Optimica Extension – Scope

$$\min_{u(t),p} \int_0^{t_f} L(x(t),u(t),p)dt + \phi(x(t_f))$$

subject to

$$f(\dot{x}(t),x(t),u(t),p) = 0$$

and

$$\begin{aligned} c_e(x(t),u(t),p) &= 0 & c_i(x(t),u(t),p) &\leq 0 \\ c_{fe}(x(t_f),u(t_f),p) &= 0 & c_{fi}(x(t_f),u(t_f),p) &\leq 0 \\ c_{0e}(x(0),u(0),p) &= 0 & c_{0i}(x(0),u(0),p) &\leq 0 \end{aligned}$$

- Modelica to express dynamic constraint
- Optimica to express everything else

The Optimica Extension – Scope cont'd

$$\min_{u,p} M(x)$$

subject to

$$\begin{aligned} f(\dot{x},x,u,p)|_{\dot{x}=0} &= 0 \\ c_i(x,u,p) &\leq 0 \\ c_e(x,u,p) &= 0 \end{aligned}$$

- Static optimization based on a static or dynamic Modelica model

The Optimica Compiler

- Extension of a Modelica compiler: The JModelica compiler for pure Modelica
- The following steps are performed in the compilation
  - **Flattening:** The original Modelica model is transformed into a flat differential equation. This involves instantiation of components and resolution of inheritance.
  - **Transcription:** The derivative operators in the flat model representation is transcribed using the collocation method based on Lagrange polynomials
  - **Code generation:** The flat transcribed representation is written, in the form of an NLP problem, to AMPL format
- The cumbersome and error-prone transcription procedure is performed automatically
- The user can focus on *formulation* of the optimization problem instead of *encoding* it

The Optimica Extension

- An Optimica model is complementing a Modelica model
  - All declarations in the Modelica model are (implicitly) visible in the corresponding Optimica description
- Language elements
  - Superimpose bounds on variables
  - Mark Modelica parameters as free in optimization
  - Specify initial guesses
  - Mesh
  - Cost function
  - Constraints
- Initial guesses can be read from file

Optimica – Superimpose information

- Variable bounds (path constraints for variables)

```
varName(lowerBound=-1,upperBound=1);
```

- Let a Modelica parameter be free in the optimization

```
oq varName(lowerBound=-1,upperBound=1);
```

- Initial guess

```
varName(lowerBound=-1,
        upperBound=1,
        initialGuess=0);
```

- Specify free initial conditions for state variables

```
varName(freeInitial(
  lowerBound=[-0.01;-0.001;-0.01;-0.001],
  upperBound=[0.01;0.001;0.01;0.001],
  initialGuess=[0.001;0;0;0])=true);
```

Optimica – Mesh

- A collocation method use a *mesh* consisting of a number of finite elements
- Declared in section `optimization`
- Mesh with fixed final time

```
grid(finalTime=fixedFinalTime(finalTime=tf),
                              nbrElements=n_el);
```

- Mesh with free final time

```
grid(finalTime=openFinalTime(initialGuess=tf_ig,
                             lowerBound=tf_lb,
                             upperBound=tf_ub),
                             nbrElements=n_el);
```

- Static optimization

```
grid(static=true);
```

Optimica – Cost Function and Constraints

- Cost function specified in section `optimization`
- Lagrange cost function

```
minimize(lagrangeIntegrand=li_exp);
```

- Terminal cost function

```
minimize(terminalCost=tc_exp);
```

- Combining Lagrange and terminal cost

```
minimize(lagrangeIntegrand=li_exp,
         terminalCost=tc_exp);
```

- Constraints declared in section `subject to`

```
y<=x^2; // Path constraint
initial cos(x)>=0.4 // Initial constraint
terminal y=4; // Terminal constraint
```

Example – CSTR: Modelica

```
model CSTR
  parameter Real beta=1/20, J=100, cf=7.6;
  parameter Real alpha=1.95e-4, Tf=300, k=300;
  parameter Real Tc=290, N=5;
  parameter Real yc=Tc/J/cf, yf=Tf/J/cf;
  parameter Real c0=0.1367, T0=0.7293;
  parameter Real cref=0.0944, Tref=0.7766, uref=340;
  parameter Real q1=1, q2=1, r=0.000001;
  Real c(start=c0), T(start=T0),
  input Real u;
equation
  der(c)=beta*(1-c)-k*exp(-N/T)*c;
  der(T)=beta*(yf-T)+k*exp(-N/T)*c-alpha*(T-yc)*u;
end CSTR;
```

Example – CSTR: Optimica

```
class CSTR
optimization
  grid(finalTime=fixedFinalTime(finalTime=60),
                                nbrElements=300);
  minimize(lagrangeIntegrand = q1*(cref-c)^2+
                               q2*(Tref-T)^2+
                               r*(uref-u)^2);
end CSTR;
```

- Mesh with 300 elements and fixed final time 60 s
- Lagrange cost function

Running the Optimica Compiler

- `optimicac` arguments: argument
  - Name of Optimica file
  - Name of Modelica file
  - Name of Modelica class to optimize
  - Name of initial guess file (optional)
- TOC output

```
> optimicac CSTR.op CSTR.mo CSTR
Parsing CSTR.mo...
Checking for errors in Modelica file...
0 errors detected...
Instantiation starts...
Instantiation finished!
Parsing CSTR.op...
Checking for errors in Optimica file...
0 errors detected...
Generating AMPL code...
Code generation finished!
```

Optimica Compiler Output – AMPL Code Files

- Model definition

  `CSTR.mod`
- Data definition

  `CSTR.dat`
- Initial guess file

  `CSTR.InitialGuess.mod`
- Constraints

  `CSTR.Constraint.mod`
- Cost

  `CSTR.Cost.mod`
- Script for generating result file

  `CSTR.GenLogFile.run`
- Script for loading problem definitions and solving problem

  `CSTR.run`

AMPL

- A language for mathematical programming
  - Variable an set declarations
  - Cost function
  - Constraints
  - Model and data separation
  - But no derivative operator...
- The AMPL shell offers a scripting language which enables definition and execution of optimization problems
- Interfaces to a large number of numerical solvers (including IPOPT)
- Offers an API with sparsity patterns, first and second order derivatives to solvers
- Numerical solver invoked from AMPL shell

AMPL – Example

```
ampl: var x >=-10 <=10;
ampl: var y=(x-3)^2+3;
ampl: minimize COST: y;
ampl: solve;
... Ipopt 3.2.0: Optimal Solution Found
ampl: display x;
x = 3
ampl: display y;
y = 3
ampl: subject to CONSTR: x>=5;
ampl: solve;
... Ipopt 3.2.0: Optimal Solution Found
ampl: display x;
x = 5
ampl: display y;
y = 7
```

CSTR Example cont'd

- Execution in AMPL

  `ampl: include CSTR.run;`
- Result is written to file `CSTR_res.txt` in Dymola (textual) format
- Load results into Matlab

  `>> d=loaddym('CSTR_res.txt');`
- Retrieve trajectories

```
>> [t,c]=getvar(d,'c');
>> [t,T]=getvar(d,'T');
>> [t,u]=getvar(d,'u');
```

Static Optimization Example

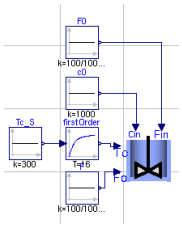
- Modelica class

```
class StaticOpt
  Real x[4];
end StaticOpt;
```

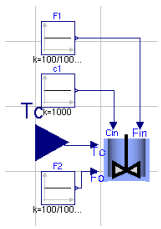
- Optimica class

```
class StaticOpt
        x(lowerBound={1,1,1,1},upperBound={5,5,5,5},
           initialGuess={1,5,5,1});
optimization
  grid(static=true);
  minimize(terminalCost=x[1]*x[4]*(x[1]+x[2]+x[3])+x[3]);
subject to
  x[1]*x[2]*x[3]*x[4]>=25;
  x[1]^2+x[2]^2+x[3]^2+x[4]^2=40;
end StaticOpt;
```

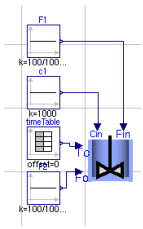
Working with Dymola



Generation of initial guess



Optimization model



Verify optimal solution

Summary

- Modelica and Optimica for high-level dynamic optimization problems
- Transcription automated by the Optimica compiler
- AMPL – A language (and tool) for mathematical programming
- Numerical solver IPOPT used to solve the resulting NLP
- Results can be loaded into Matlab or Dymola

Extra example—Minimum Time Control

Problem: Use bounded control, $u \in [-1, 1]$, and bring the states of the double integrator to the origin as fast as possible.

Free end-time t_f

$$\min_{u:[0,t_f]\to[-1,1]} t_f = \min_{u:[0,t_f]\to[-1,1]} \int_0^{t_f} 1\, dt$$
$$\dot{x}_1(t) = x_2(t)$$
$$\dot{x}_2(t) = u(t)$$
$$\psi(x(t_f)) = (x_1(t_f), x_2(t_f))^T = (0,0)^T$$

Adjoint equations $\dot{\lambda}_1(t) = 0, \quad \dot{\lambda}_2(t) = -\lambda_1(t)$ gives

$$\lambda_1(t) = c_1, \quad \lambda_2(t) = c_2 - c_1 t$$

With $u(t) = \zeta = \pm 1$, we have

$$x_1(t) = x_1(0) + x_2(0)t + \zeta t^2/2$$
$$x_2(t) = x_2(0) + \zeta t$$

Eliminating t gives curves (parabolas)

$$x_1(t) \pm x_2(t)^2/2 = \text{const}$$

These define the *switch curves*, where the optimal control switch.

Remark: See also solution to exam March 8, 2005 for more details

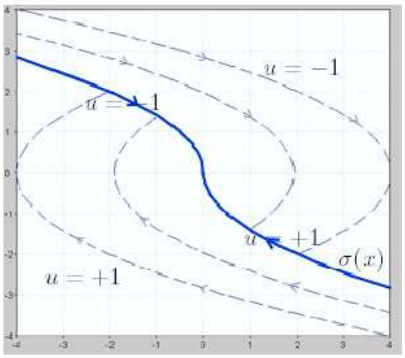
Hamiltonian

$$H = L + \lambda^T f = 1 + [\lambda_1 \lambda_2] \begin{bmatrix} x_2 \\ u \end{bmatrix}$$

The optimal control is the control which minimizes

$$u^*(t) = \arg\min_{u\in[-1,1]} H = \arg\min_{u\in[-1,1]} 1 + \lambda_1(t)x_2^*(t) + \lambda_2(t)u$$
$$= \begin{cases} 1, & \lambda_2(t) < 0 \\ -1, & \lambda_2(t) \geq 0 \end{cases}$$

Note: This is "bang-bang control".



Combination of solution curves: Follow one parabola using maximum or minimum control signal until you hit the switch curve σ, switch to minimum/maximum control and then follow that parabola to the origin.