Native Optimization Features in **OpenModelica**

Vitalij Ruge

January 2, 2015

Abstract

OpenModelica supports native dynamic optimization of models. This allows users to define optimal control problems (OCP) using Modelica and Optimica (only partly supported) language specifications, and solve the underlying model formulation using collocation methods.

Motivation

In Modelica it's easy to formulate models with the causality visualized in figure 1. But in some cases

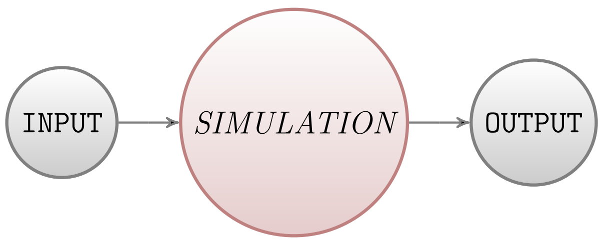

Figure 1: Causality Modelica model

it is really hard to find the right input to get the right output like in figure 2. With the optimization

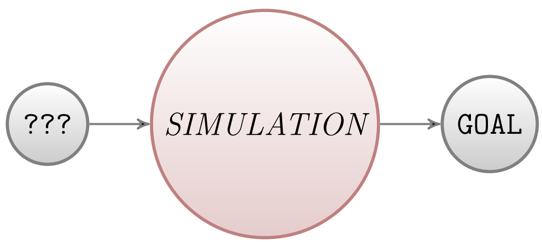

Figure 2: Unknown input, but a goal for the output

in **OpenModelica** we can find the input!

Contents

1 Optimal Control Problem

Mathematical formalization [4]:

$$\min_{u(t)} \ \mathtt{J}(x(t), u(t), t) = \ E(x(t_f), u(t_f), t_f) + \int_{t_0}^{t_f} L(x(t), u(t), t) \ dt \tag{1}$$

s.t.

$$\begin{aligned} x(t_0) &= & x_0 \quad (2)\\ \dot{x}(t) &= & f(x(t), u(t), t) \quad (3)\\ \hat{g}(x(t), u(t), t) &\leq & 0 \quad (4)\\ r(x(t_f)) &= & 0 \quad (5) \end{aligned}$$

where $x(t) = \left[x^{(1)}(t), \ldots, x^{(n_x)}(t)\right]^\top$ and $u(t) = \left[u^{(1)}(t), \ldots, u^{(n_u)}(t)\right]^\top$ are the state vector and control variable vector for $t \in [t_0, t_f]$, respectively. The constraints (2), (3), (4) and (5) represent the initial conditions, the nonlinear dynamic model description based on differential algebraic equations (DAEs), the path constraints $\hat{g}(x(t), u(t), t) \in \mathbb{R}^{n_{\hat{g}}}$ and the terminal constraints [2].
The path constraints can be split in the box constraints from $\hat{g}(x(t), u(t), t) \leq 0$, i.e.

$$\begin{array}{ccccc} x_{\min} & \leq & x(t) & \leq & x_{\max} \\ u_{\min} & \leq & u(t) & \leq & u_{\max} \end{array}$$

which can handling efficiently, it is possible to use the attributes `min` and `max` already available in Modelica for the description(***hint***: use `StateSelect` for transform nonlinear constraints in box constraints)[3]. And general constraints

$$g(x(t), u(t), t) \leq \ 0$$

which can formulate with Optimica [1] or special user annotation.

2 Extension

Extension for the formulation of OCP in **OpenModelica**:

	Optimica	Annotation
Mayer-Term	objective = costM	Real costM annotation(isMayer = true);
Lagrange-Term	objectiveIntegrand = costL	Real costL annotation(isLagrange = true);
constraints	con <= 0	Real con(max=0) annotation(isConstraint = true);
final constraints	not supported in OM	Real fcon(max=0) (isFinalConstraint = true);

Open issues: alias elimination for final constraints.
Note: *OMEdite* support Modelica and not Optimica. It's possible to use *OMNotebook*.

3 OCP Formulation

At the beginning we start with a predator–prey equations in Modelica, which we want optimize!

Listing 1: Modelica forest model

```
model forest
```

```
  parameter Real g_r = 4e-2  "Natural growth rate for rabbits";
  parameter Real g_fr = 1e-1 "Efficient in growing foxes from
     rabbits";

  parameter Real d_rf = 5.0e-3 "Death rate of rabbits due to foxes
     ";
  parameter Real d_rh = 5.0e-2 "Death rate of rabbits due to
     hunters";
  parameter Real d_f = 9.0e-2  "Natural deathrate for foxes";
  parameter Real d_fh = 9.0e-2 "Death rate of foxes due to hunters
     ";


  Real rabbits(start=700)    "Rabbits,(R) with start population
     700";
  Real foxes(start=10)       "Foxes,(F) with start population 10";

  input Real hunter_rabbits;
  input Real hunter_foxes;
equation
  der(rabbits) = g_r*rabbits - d_rf*rabbits*foxes - d_rh*
     hunter_rabbits;
  der(foxes) = g_fr*d_rf*rabbits*foxes -d_f*foxes - d_fh*
     hunter_foxes;
end forest;
```

The forest-model depict a forest with foxes, rabbits and hunters. The hunters should control the population. This example we can extends with our optimization goals.

3.1 Mayer-term

For Example we wish to minimized the difference:

$$\frac{(\mathrm{foxes}(t_f) - 5)^2}{10} + \frac{(\mathrm{rabbits}(t_f) - 500)^2}{100}$$

In order to get a wished population in the forest at the end. In this case we can use the mayer term for the formulation.

3.1.1 Optimica-Mayer-Term

We can use the Optimica-Extension with two new keywords {`optimization,objective` }, which currently not part of Modelica.

Listing 2: Modelica&Optimica forest model, mayert-term

```
optimization forestMayer(objective = goalRabbits + goalFoxes)
"
  goal:
    closing balance rabbits = 500
    closing balance foxes = 5
"
  extends forest;
  Real goalRabbits = (rabbits - 500)^2/100 "goal for rabbits ";
  Real goalFoxes =  (foxes-5)^2/10 "goal for rabbits ";
```

```
end foresMayer;
```

The keyword `optimization` is alternative to `model` and signals this is not typical model for simulation.
The keyword `objective` is an attribute of the class `optimization` and contains the object function for the endpoint.

3.1.2 Annotation-Mayer-Term

Alternative it is possible to use `annotation` to say the optimizer, which variable express the mayer-term.

Listing 3: Modelica forest model, mayert-term

```
model forestMayer
"
  goal:
    closing balance rabbits = 500
    closing balance foxes = 5
"
  extends forest;
  Real goalRabbits(nominal = 1e2) = (rabbits - 500)^2 "goal for
     rabbits " annotation(isMayer = true);
  Real goalFoxes(nominal = 1e1) =  (foxes-5)^2 "goal for rabbits "
     annotation(isMayer = true);

end foresMayer;
```

Note: `nominal` used for scale the object!

In both formulations 2 and 3 **OpenModelica** find the inputs(`hunter_rabbits`,`hunter_foxes`)

3.2 Lagrange-Term

Of the other hand we want minimize the number of the hunters in each time point, because we pay for the guys. In this case we can use the lagrange-term (or include additional state, which affect the mayer-term).

$$\begin{aligned} &\text{min!} && \int \text{cost}(t)dt && \text{or} \\ &\text{min!} && \text{Cost}(t_f) && \text{where } \dot{\text{Cost}} = \text{cost}(t) \text{ and } \text{Cost}(t_0) = 0 \end{aligned}$$

Note: The numeric and convergence of both formulation is not the same.

3.2.1 Optimica-Lagrange-Term

Listing 4: Modelica&Optimica forest model, lagrange-term

```
optimization forestLagrange(objective = goalRabbits + goalFoxes,
objectiveIntegrand = 1e-2*costHuntersFoxes + 1e-2*costHuntersRabbits
)
"
```

```
  goal:
    closing balance rabbits = 500
    closing balance foxes = 5
    minimize cost for hunters
"

  extends forest;
  Real goalRabbits = (rabbits - 500)^2 "goal for rabbits ";
  Real goalFoxes =  (foxes-5)^2 "goal for rabbits ";
  Real costHuntersFoxes = hunter_foxes^2;
  Real costHuntersRabbits = hunter_rabbits^2;
end forestLagrange;
```

The keyword `objectiveIntegrand` is an attribute of the class `optimization` and contains the object function over the time.

3.2.2 Annotation-Lagrange-Term

Listing 5: Modelica forest model, lagrange-term

```
model forestLagrange
"
  goal:
    closing balance rabbits = 500
    closing balance foxes = 5
    minimize cost for hunters
"

  extends forest;
  Real goalRabbits(nominal = 1e2) = (rabbits - 500)^2 "goal for
     rabbits " annotation(isMayer = true);
  Real goalFoxes(nominal = 1e1) =  (foxes-5)^2 "goal for rabbits "
     annotation(isMayer = true);
  Real costHuntersFoxes(nominal = 1e2) = hunter_foxes^2 annotation(
     isLagrange = true);
  Real costHuntersRabbits(nominal = 1e2) = hunter_rabbits^2
     annotation(isLagrange = true);
end forestLagrange;
```

3.3 Box-Constraints

The bounds for the states and inputs will be handling as box constraints.

Listing 6: forest model, box constraints

```
... forestBoxConstarints
"
box constraints
        0 <= foxis <= 700
        0 <= rabbits <= 200
        0 <= hunter_foxes
        0 <= hunter_rabbits
"
```

```
  extends forest(
         foxis(min = 0, max = 700),
         rabbits(min =0, max = 200),
         hunter_foxes(min = 0),
         hunter_rabbits(min = 0)
        );
end forestBoxConstarints;
```

3.4 Constraints

3.4.1 Optimica-Constraints

Listing 7: Modelica&Optimica forest model, constraints

```
optimization forestConstarints
"
box constraints
        0 <= foxis <= 700
        0 <= rabbits <= 200
        0 <= hunter_foxes
        0 <= hunter_rabbits
constraints
  2 <= hunter_foxes + hunter_rabbits <= 30
  foxis <= 7 * rabbits;
"

  extends forest(
         foxis(min = 0, max = 700),
         rabbits(min =0, max = 200),
         hunter_foxes(min = 0),
         hunter_rabbits(min = 0)
        );

constraint
        2 <= hunter_foxes + hunter_rabbits;
        hunter_foxes + hunter_rabbits <= 30;
        foxis <= 7 * rabbits;
end forestConstarints;
```

With `constraint` we have a new section like `equation` where we can formulate constraints.
Note: `constraint` is only support inside `optimization`.

3.4.2 Annotation-Constraints

Listing 8: Modelica forest model, constraints

```
model forestConstarints
"
box constraints
        0 <= foxis <= 700
        0 <= rabbits <= 200
```

```
        0 <= hunter_foxes
        0 <= hunter_rabbits
constraints
  2 <= hunter_foxes + hunter_rabbits <= 30
  foxis <= 7 * rabbits;
"

  extends forest(
         foxis(min = 0, max = 700),
         rabbits(min =0, max = 200),
         hunter_foxes(min = 0),
         hunter_rabbits(min = 0)
        );
        Real con1(min=2, max = 30) annotation(isConstraint = true);
        Real con2(max = 0) = foxis - 7 * rabbits annotation(
           isConstraint = true);
equation
        con1 = hunter_foxes + hunter_rabbits;
end forestConstarints;
```

3.4.3 Annotation-Final-Constraints

Listing 9: Modelica forest model, final constraints

```
model forestFinalConstarints
"
box constraints
        0 <= foxis <= 700
        0 <= rabbits <= 200
        0 <= hunter_foxes
        0 <= hunter_rabbits
constraints
  2 <= hunter_foxes + hunter_rabbits <= 30
  foxis <= 7 * rabbits;
final constraint
  hunter_foxes == 0;
  hunter_rabbits == 0;
"

  extends forest(
         foxis(min = 0, max = 700),
         rabbits(min =0, max = 200),
         hunter_foxes(min = 0),
         hunter_rabbits(min = 0)
        );
        parameter Real noAlias = 1.0;
        Real con1(min=2, max = 30) annotation(isConstraint = true);
        Real con2(max = 0) = foxis - 7 * rabbits annotation(
           isConstraint = true);
        Real fcon1(min=0,max=0) = noAlias*hunter_foxes annotation(
           isFinalConstraint = true);
```

```
        Real fcon2(min=0,max=0) = noAlias*hunter_rabbits annotation(
           isFinalConstraint = true);
equation
        con1 = hunter_foxes + hunter_rabbits;
end forestFinalConstarints;
```

4 Special Options

4.1 Compilier Options

numberOfIntervals	e.g. 50	collocation intervals
startTime, stopTime		time horizon
tolerance	e.g. 1e-8	solver/optimizer tolerance
simflags	(see 4.2)	simulation flags

4.2 Simulation Options

`-lv`	`LOG_IPOPT, LOG_IPOPT_ERROR`	collocation intervals
`-ipopt_hesse`	`CONST,BFGS,NUM`	hessian approximation
`-ipopt_max_iter`	e.g. 100	maximal number of iteration for ipopt
`-exInputFile`	externalInput.csv	input guess
`-optimizerNP`	1 or 3	number of collcation points
`-ipopt_warm_start`	e.g. 8	scale for initial guess (prototype)

5 Examples

Som example can be founded in `https://trac.openmodelica.org/OpenModelica/browser/trunk/testsuite/openmodelica/cruntime/optimization/basic/`.

5.1 OMNotebook and Shell

`https://trac.openmodelica.org/OpenModelica/browser/trunk/doc/SimulationRuntime/DynamicOptimiza`

Initialguess

`https://trac.openmodelica.org/OpenModelica/browser/trunk/testsuite/openmodelica/cruntime/optimization/basic/BRinitialGuess.mos`

Warm Start

`https://trac.openmodelica.org/OpenModelica/browser/trunk/testsuite/openmodelica/cruntime/optimization/basic/DMwarm.mos`

Final Constraints

`https://trac.openmodelica.org/OpenModelica/browser/trunk/testsuite/openmodelica/cruntime/optimization/basic/TFC.mos`

References

[1] J. Åkesson. *Languages and Tools for Optimization of Large-Scale Systems*. PhD thesis, Regler, nov 2007.

[2] B. Bachmann, L. Ochel, V. Ruge, M. Gebremedhin, P. Fritzson, V. Nezhadali, L. Eriksson, and M. Sivertsson. Parallel multiple-shooting and collocation optimization with openmodelica. In *9th International Modelica Conference*, 2012.

[3] R. Franke. Formulation of dynamic optimization problems using modelica and their efficient solution. *9th International Modelica Conference*, pages 315 –323, 2002.

[4] T.L. Friesz. *Dynamic optimization and differential games*. Springer, 2010.