# Manual: using optimization modeling library.

# Berkeley Library for Optimization Modeling: BLOM

# Revision 0.1

Sergey Vichik

January 23, 2012

## 1 Introduction

The library and the associated tools are designed to ease process of modeling dynamical nonlinear systems for optimization problems, especially MPC. The library allows intuitive block diagram definition of the system, simulation, validation, and an automatic generation of an optimization problem to be solved. This process eliminates errors introduced by manual conversion and allows conversion of large models, too large to be converted manually. The library consists of Simulink blocks library and Matlab software for converting model to an optimization problem.

This library supports the following work flow for defining an optimization problems. First the model of the system is drawn using Simulink and the supplied blocks. The models includes

1. All equations that describe the system
2. Inequality constraints
3. Cost function
4. Which variable is the free variable to optimize, and what variable is externally determined.

In the second stage the model is validated by running forward mode simulation. The validation includes a check whether constraints are violated. After the validation phase, the model is converted to optimization problem, thus

1. Cost function.
2. Linear and nonlinear equality constraints (from model equations)
3. Linear and nonlinear inequality constraints.
4. List of optimization variables.
5. Externally specified variables using user supplied data.

If the model includes dynamics, like MPC problems, the optimization problem includes the requested number of time steps.

Finally, the optimization problem is exported to one of the following optimization engines, including automatic Matlab or C code generation

1. Matalb's fmincont
2. TOMLAB - TBD
3. CPLEX - TBD
4. IPOPT - TBD

## 2 Modeling blocks

For model definition only the blocks listed in the library are allowed. All other Simulink blocks can be used outside of the modeled subsystem.

### 2.1 PolyBlock

PolyBlock is a key element of the library.

**Responsibility:** - Defines output as function of input. The function is defined using ....

**Input:** input variables, as many as width of A matrix.

**Output** the calculated value.

#### 2.1.1 Remarks

If the expression is short, it will be shown on the block. If the expression is too long, only y=f(x)/g(x) will be shown.

More versions of the block exist in the library, for convenience only:

**Sum operator** sum of two variables,

**Difference operator** - difference of two variables,

**Product** - product of two variables.

**Constant gain**

**Bias**

**General form PolyBlock** - can be used to define any relation between input and output, including non-functions, like solution of quadratic equation. In this case, the behavior in forward simulation is undefined.

### 2.1.2 Usage

Basic building block is a sum of terms of the following form $\sum_i \prod_j v_{i,j}(x_i)$, where $v(x_i)$ can be $x^p$ for any $p \in R$, $\exp(x)$ or $\log(x)$. Functions are defined using two matrices, called here $A$ and $C$. Where $A_{i,j} = p$ where $p$ is a power of $v_{i,j}$ function. Two special values are reserved for $p$ to mark the exp and the log functions. $p = \infty$ marks exp and $p = -\infty$ marks log. Although $p$ can be any real number, only positive integers and exp must be used for most cases, because their provide the fastest computation time. Library supports other options for completeness and to accommodate very special cases that cannot be handled with the fast functions. Example: TBD

The PolyBlock enables definition of rational function of the form $\frac{f(x)}{g(x)}$ where $f$ and $g$ are functions as defined above.

Following special cases of PolyBlock are important:

**Constant** $A = [0 \ldots 0]$ , output $= C$.

**Linear algebra** If $A = I$, then the output of the block is $Cx$ where $x$ is the input and $C$ is any matrix of compatible width.

**Vector functions** Vector functions can be defined. The number of rows of C matrix defines the dimension of output vector.

**Selector function** each row of A holds 1 at requested position, and $C = I$.

The dimensions of the A and C matrices of $f$ and $g$ functions, must follow the following rules. If:

|  |  |
|---|---|
| $n$ | input dimension |
| $m$ | output dimension |
| $r$ | number of term in f function |
| $k$ | number of terms in g function |

Then:

$$fA \in \mathbb{R}^{rxn}$$

$$fC \in \mathbb{R}^{mxr}$$

$$gA \in \mathbb{R}^{kxn}$$

$$gC \in \mathbb{R}^{1xk}$$

note that $g$ must be scalar function, because division by vector is not defined.

## 2.2 Constraint

**Responsibility** - Marks variable as a constrained to be $> 0$ or $< 0$. Turns red if the constraint is violated during the simulation.

**Input** The variable to be constrained.

**Output** none.

### 2.3 Input

**Responsibility** - Marks a variable as an input variable. Can specify the number of steps to freeze the input.

**Input** a variable

**Output** - equal to input

### 2.4 External

**Responsibility** - Marks a variable as an external variable.

**Input** a variable

**Output** - equal to input

### 2.5 State

**Responsibility** - Holds value for the next step.

**Input** a variable

**Output** value from the previous step.

### 2.6 Cost

**Responsibility** defines cost function. Sums input variable on every time step to calculate the cost.

**Input** variable

**Output** none

### 2.7 Save to workspace

**Responsibility** copy of the regular Simulink block. Required for storing variables for initial optimization solution.

**Output** none

## 3 Generation of an optimization problem

TBD