BLOM: Berkeley Library for Optimization Modeling

Sergey Vichik, Anthony Kelman and Francesco Borrelli

UC Berkeley
Department of Mechanical Engineering
Berkeley, CA
{sergv, kelman}@berkeley.edu

January, 2013
What is **BLOM**?

- A tool and language of modeling (dynamical) nonlinear systems for optimization problems.
What is BLOM?

- A tool and language of modeling (dynamical) nonlinear systems for optimization problems.

Conventional approach

- Manual formulation and coding
- Time consuming, error prone (especially gradients, Jacobian, Hessian)
- Difficult to change solvers
- Tricky to get good performance

BLOM approach eliminates manual problem coding, eases maintenance and assures that the same model used for optimization as for simulation

- Intuitive block diagram
- Simulink/Matlab based interface.
- Forward simulation for validation of the model
- Developed to handle non trivial problems
  - Explicit evaluation of Jacobian and Hessian
  - Automatic and efficient export of the optimization problem to a solver
What is **BLOM**?

- **A tool and language** of modeling (dynamical) nonlinear systems for optimization problems.

**Conventional approach**
- Manual formulation and coding
- Time consuming, error prone (especially gradients, Jacobian, Hessian)
- Difficult to change solvers
- Tricky to get good performance

**BLOM** approach eliminates manual problem coding, eases maintenance and assures that the same model used for optimization as for simulation
- Intuitive block diagram
- Simulink/Matlab based interface.
- Forward simulation for validation of the model
- Developed to handle non trivial problems
  - Explicit evaluation of Jacobian and Hessian
  - Automatic and efficient export of the optimization problem to a solver
\[
\min_{u_k, x_k} \sum_k 0.5x_k^2 + 2u_k^2
\]

s.t. \(-2 \leq u_k \leq 2 ; 0.5 \leq x_k \leq 1 ; x_{k+1} = 0.9x_k + u_k\)
"Hello World" model example

\[
\min_{u_k, x_k} \sum_k 0.5x_k^2 + 2u_k^2 \\
\text{s.t. } -2 \leq u_k \leq 2 ; 0.5 \leq x_k \leq 1 ; x_{k+1} = 0.9x_k + u_k
\]

- The **Functional** block holds expression of the form \( \frac{f(x)}{g(x)} \)
"Hello World" model example

\[
\min_{u_k, x_k} \sum_k 0.5x_k^2 + 2u_k^2
\]

s.t. : \(-2 \leq u_k \leq 2 ; 0.5 \leq x_k \leq 1 ; x_{k+1} = 0.9x_k + u_k\)

- The **Functional** block holds expression of the form \(\frac{f(x)}{g(x)}\)
- The **Constraint** block marks variable as \(\geq 0\) or \(\leq 0\)
"Hello World" model example

\[
\min_{u_k,x_k} \sum_k 0.5x_k^2 + 2u_k^2
\]

s.t. : \(-2 \leq u_k \leq 2 ; 0.5 \leq x_k \leq 1 ; x_{k+1} = 0.9x_k + u_k\)

- The **Functional** block holds expression of the form \(\frac{f(x)}{g(x)}\)
- The **Constraint** block marks variable as \(\geq 0\) or \(\leq 0\)
- The continuous or discrete **State** block
"Hello World" model example

\[
\min_{u_k, x_k} \sum_k 0.5x_k^2 + 2u_k^2
\]

s.t. \(-2 \leq u_k \leq 2; 0.5 \leq x_k \leq 1; x_{k+1} = 0.9x_k + u_k\)

- The **Functional** block holds expression of the form \(f(x)/g(x)\)
- The **Constraint** block marks variable as \(\geq 0\) or \(\leq 0\)
- The continuous or discrete **State** block
- The **Cost** block accumulates cost variables over horizon
"Hello World" model example

\[
\min_{u_k, x_k} \sum_{k} 0.5x_k^2 + 2u_k^2 \\
\text{s.t.} : -2 \leq u_k \leq 2; \quad 0.5 \leq x_k \leq 1; \quad x_{k+1} = 0.9x_k + u_k
\]

- The **Functional** block holds expression of the form \( \frac{f(x)}{g(x)} \)
- The **Constraint** block marks variable as \( \geq 0 \) or \( \leq 0 \)
- The continuous or discrete **State** block
- The **Cost** block accumulates cost variables over horizon
- The **Input/External** variable modifiers marks the control and the external variables
Efficient problem representation

- Why are LP, QP easy? (besides being convex)
  - Standard format, e.g. for QP:
    \[
    \min_x \frac{1}{2} x^T Q x + c^T x
    \]
    s.t. \(Ax \leq b, \ Ex = d\)
  - Gradient, Jacobian, etc immediate
Efficient problem representation

- Why are LP, QP easy? (besides being convex)
  - Standard format, e.g. for QP:
    \[
    \min_x \frac{1}{2} x^T Q x + c^T x \\
    \text{s.t. } A x \leq b, \quad E x = d
    \]
  - Gradient, Jacobian, etc immediate

- Typical nonlinear approach:
  - Code generation or parsing, algorithmic differentiation
  - Explicit code gen does not scale well to very large problems
Efficient problem representation

- Why are LP, QP easy? (besides being convex)
  - Standard format, e.g. for QP:
    \[
    \min_x \frac{1}{2} x^T Q x + c^T x
    \]
    \[
    \text{s.t. } A x \leq b, \; E x = d
    \]
  - Gradient, Jacobian, etc immediate

- Typical nonlinear approach:
  - Code generation or parsing, algorithmic differentiation
  - Explicit code gen does not scale well to very large problems

- **BLOM** is our proposal for standardized NLP format
  - Represent nonlinear structure of model in sparse matrices
  - Matrix of exponents/functions, matrix of coefficients
  - Cost vector, upper and lower bound vectors

- Key to performance of optimization algorithms
BLOM representation details

\[ f(x) = \sum_{k=1}^{r} K_k \left( \prod_{j=1}^{n} v(x_j, P_{kj}) \right) \]

The parameterized function \( v \) is defined as

\[ v(x, p) = \begin{cases} 
  x^p & \text{if } p \text{ is not an exception code} \\
  \exp(x) & \text{if } p \text{ is the code for } \exp \\
  \sin(x) & \text{if } p \text{ is the code for } \sin \\
  \tanh(x) & \text{if } p \text{ is the code for } \tanh \\
  \text{etc.} & 
\end{cases} \]

- Multivariate polynomial-like structure with sparse \( P, K \)
- Closed-form sparse Jacobian, Hessian
Create model using Simulink with BLOM library
BLOM work flow

- Create model using Simulink with **BLOM** library
- Run and compare the model to reference data
Create model using Simulink with BLOM library
Run and compare the model to reference data
Automatically generate efficient problem representation
Create model using Simulink with BLOM library
Run and compare the model to reference data
Automatically generate efficient problem representation
Solve using compiled interface to optimization solvers
Availability, License and Usage

- Publicly available since September
  Follow links at www.mpc.berkeley.edu, under Software BSD license: free to use, modify, or redistribute
- Please cite us if you find it useful (as academic courtesy, not a condition of license)
- Not application specific, feel free to evaluate for other projects
- Being used in large HVAC MPC setup by UTRC, Master’s level MPC course at Berkeley, HVAC MPC in our lab, wave energy conversion research and more
- Typical computation times from 0.3 sec for ~1000 variables to ~100 sec for ~30K variables
Summary

- BLOM eliminates manual coding of optimization model, facilitates very fast development cycle
- High performance for very large scale models
- Simplifies collaborative development and knowledge transfer between team members - Using someone else’s block diagram vs reading their code
- Develop model once, use with any solver and environment later
Summary

- BLOM eliminates manual coding of optimization model, facilitates very fast development cycle
- High performance for very large scale models
- Simplifies collaborative development and knowledge transfer between team members - Using someone else’s block diagram vs reading their code
- Develop model once, use with any solver and environment later
- DEMO