APMonitor Modeling Language

John Hedengren
Brigham Young University

Advanced Process Solutions, LLC
http://apmonitor.com
Overview of APM

- Software as a service accessible through:
  - MATLAB, Python, Web-browser interface
  - Linux / Windows / Mac OS / Android platforms
- Solvers
  - APOPT\(^1\), BPOPT\(^1\), IPOPT\(^2\), SNOPT\(^3\), MINOS\(^3\)
- Problem characteristics:
  - Large-scale
  - Nonlinear Programming (NLP)
  - Mixed Integer NLP (MINLP)
  - Multi-objective
  - Real-time systems
  - Differential Algebraic Equations (DAEs)

\[
\begin{align*}
\text{min } J(x, y, u, z) \\
\text{s.t. } 0 &= f \left( \frac{\partial x}{\partial t}, x, y, u, z \right) \\
0 &= g(x, y, u, z) \\
0 &< h(x, y, u, z) \\
x, y \in \mathbb{R}^n &\quad u \in \mathbb{R}^m &\quad z \in I^m
\end{align*}
\]

---
1 – APS, LLC
2 – EPL
3 – SBS, Inc.
Overview of APM

- Vector / matrix algebra with set notation
- Automatic Differentiation
  - Exact 1\textsuperscript{st} and 2\textsuperscript{nd} Derivatives
- Large-scale, sparse systems of equations
- Object-oriented access
  - Thermo-physical properties
  - Database of preprogrammed models
- Parallel processing
- Optimization with uncertain parameters
- Custom solver or model connections
Unique Features of APM

- Initialization with nonlinear presolve

\[
\begin{align*}
\min J(x, y, u) \\
\text{s.t.} & \quad 0 = f\left(\frac{\partial x}{\partial t}, x, y, u\right) \\
& \quad 0 = g(x, y, u) \\
& \quad 0 < h(x, y, u)
\end{align*}
\]

- Explicit variable substitution every function call

\[
\begin{align*}
\min J(x, y, u) \\
\text{s.t.} & \quad 0 = f\left(\frac{\partial x}{\partial t}, x, y, u\right) \\
& \quad 0 = g(x, y, u) \\
& \quad 0 < h(x, y, u)
\end{align*}
\]
Unique Features of APM

- Model development workflow

<table>
<thead>
<tr>
<th></th>
<th>Steady State</th>
<th>Dynamic</th>
<th>Sequential</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simulate</td>
<td>1</td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>Estimate</td>
<td>2</td>
<td>5</td>
<td>-</td>
</tr>
<tr>
<td>Optimize</td>
<td>3</td>
<td>6</td>
<td>-</td>
</tr>
</tbody>
</table>

- Solve higher index DAEs (Index 3+ with APM)
  - Index-1 only (e.g. MATLAB ode15s)
  - Index-1 + Index-2 Hessenberg (e.g. DASPK)

- Classes of problems
  - LP, QP, NLP, **DAE**
  - MILP, MIQP, MINLP, **MIDAE**
Solver Benchmarking – Hock-Schittkowski (116)

The graph shows the percentage of problems that are not worse than $2^\tau$ times slower than the best solver, where $\tau$ is indicated on the x-axis. The x-axis represents the slower solver, and the y-axis represents the percentage of problems meeting the criterion.

The solvers compared are:
- APOPT+BPOPT
- APOPT
- BPOPT
- IPOPT
- SNOPT
- MINOS

Each solver is represented by a different line color or marker, allowing for easy comparison of performance across the different solvers.
Solver Benchmarking - Dynamic Optimization (37)

The graph shows the percentage of problems not worse than $2^\tau$ times slower than the best solver, where $\tau$ is the time factor.

- **APOPT+BPOPT**
- **APOPT**
- **BPOPT**
- **IPOPT**
- **SNOPT**
- **MINOS**

Each solver is represented by a different line on the graph, with the percentage (%) on the y-axis and the time factor ($\tau$) on the x-axis.
Solver Benchmarking – SBML (341)

Not worse than $2^\tau$ times slower than the best solver ($\tau$)

- APOPT+BPOPT
- APOPT$_{1.0}$
- BPOPT$_{1.0}$
- IPOPT$_{3.10}$
- IPOPT$_{2.3}$
- SNOPT$_{6.1}$
- MINOS$_{5.5}$
Computational Biology

- Drug treatment and discovery – large-scale models

HIV Virus Simulation

Model

Parameters
- kr1 = 1e5
- kr2 = 0.1
- kr3 = 2e-7
- kr4 = 0.5
- kr5 = 5
- kr6 = 100

End Parameters

Variables
- H = 1e6 ! healthy cells
- V = 1e2 ! virus
- I = 0 ! infected cells

End Variables

Equations
- $H = kr1 - kr2*H - kr3*H*V$
- $I = kr3*H*V - kr4*I$
- $V = -kr3*H*V - kr5*V + kr6*I$

End Equations

End Model
Biological Kinetic Models Modestly Sized

Model sizes from 409 curated models in the Biomodels repository (http://www.ebi.ac.uk/biomodels-main/)
We need better tools (parameter estimation, optimization) to deal with large models!

- Large ErbB signalling model (~504 physical entities)*
- Parameter estimation (simulated annealing) took “24 hours on a 100-node cluster computer”

Smart Grid Energy Systems

Graph showing normalized campus load over time of day, with different colors representing different chillers and a load line.

Diagram illustrating the integration of buildings, chilling stations, CHP stations, thermal storage, and power grid connections.
Solid Oxide Fuel Cells

Fuel

Power to Grid
Flow Assurance for Oil and Gas Industry

- Fouling and Plugging largest loss category
  - Billions $$$ per year in lost revenue
- Predictive Analytics
  - Real-time or Off-line Monitoring Solution
  - Empirical and First Principles Models

Safe Operations
Reliability Targets
Regulatory Reports
Maximize Economics
Training Simulators
# Engineering in Remote Locations

**Pressure Overview**
- **FLMT #2 (18 miles)**
  - Pressure (psig) last 24 hrs
  - Pressure (psig) last month

**Temperature Overview**
- **FLMT #2 (18 miles)**
  - Temperature (°F) last 24 hrs
  - Temperature (°F) last month

**Pressure Overview**
- **FLMT #1 (36 miles)**
  - Pressure (psig) last 24 hrs
  - Pressure (psig) last month

**Temperature Overview**
- **FLMT #1 (36 miles)**
  - Temperature (°F) last 24 hrs
  - Temperature (°F) last month

**Pressure Overview**
- **FLET (57 miles)**
  - Pressure (psig) last 24 hrs
  - Pressure (psig) last month

**Temperature Overview**
- **FLET (57 miles)**
  - Temperature (°F) last 24 hrs
  - Temperature (°F) last month
Safe, environmentally friendly, and economic operations

- Safety: Velocity of Inlet Waste Gas
- Safety: LEL of Waste Gas
- Economic: Fuel Costs
- Economic: Size / Insulation
- Environmental: Emission Levels
Unmanned Aerial Systems

Mothership

Actuated drogue with small autopilot and homing beacon

Top-down view of system trajectory

North (m)

East (m)
UAS System Dynamics

- Cable-drogue dynamics using Newton 2nd law

\[ m_N \ddot{p}_N = T_N + \Omega_N \]
\[ \Omega_N = G_N + D_N + L_N, \]
\[ m_{j-1} \ddot{p}_{j-1} = T_{j-1} + \Omega_{j-1} - T_j \]
\[ \Omega_{j-1} = G_{j-1} + D_{j-1} + L_{j-1} \]
\[ j = 2, 3, \ldots, N, \]

\[ T_j = \frac{EA}{\ell_0} (\|p_{j-1} - p_j\| - \ell_0) \frac{p_{j-1} - p_j}{\|p_{j-1} - p_j\|}, \]
\[ j = 1, 2, \ldots, N, \]
Dynamic System Example

Model

Parameters

! time constant
tau = 5

! gain
K = 2

! manipulated variable
u = 1

End Parameters

Variables

! output or controlled variable
x = 1

End Variables

Equations

! first order differential equation
tau * $x = -x + K * u

End Equations

End Model
Optimization Under Uncertainty

Conservative movement based on worst case CV

Upper Limit
Selecting a Model for Predictive Control

- Many model forms
  - Linear vs. Non-linear
  - Steady state vs. Dynamic
  - Empirical vs. First Principles
- Select the simplest model
- Accuracy requirements
  - Steady State Gain
  - Dynamics – Time to Steady State
- Speed requirements
  - PID < Linear MPC < Nonlinear MPC

Continuous Form (SS\(_c\))
\[
\dot{x} = Ax + Bu
\]
\[
y = Cx + Du
\]

Discrete Form (SS\(_d\))
\[
x[k+1] = A_d x[k] + B_d u[k]
\]
\[
y[k] = C_d x[k] + D_d u[k]
\]

Nonlinear Model
\[
0 = f(\dot{x}, x, u, p, d)
\]
\[
0 = g(x, u, p, d)
\]
\[
0 \leq h(x, u, p, d)
\]
Friction Stir Welding

- A rotating tool creates heat and plasticizes the metal. This allows the metal to be “stirred” together.
Getting Started with APM

Download Software at APMonitor.com

Bi-weekly Webinars

Symposium on Modeling and Optimization

Webinars are held every two weeks at 9 AM Mountain Time / 10 AM Central Time (USA). These seminars consist of applications and tutorials in mathematical modeling, estimation, and optimization.

<table>
<thead>
<tr>
<th>Topic</th>
<th>Registration Date/Time</th>
<th>Presenter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBD</td>
<td>Oct. 16, 2012 9AM MST</td>
<td>Michael Baldea, UT Austin</td>
<td>Join Webinar Password apm2012</td>
</tr>
<tr>
<td>TBD</td>
<td>Oct. 23, 2012 9AM MST</td>
<td>Selin Cremaschi, Univ of Turba</td>
<td></td>
</tr>
</tbody>
</table>

APMonitor Modeling Language

The APMonitor Modeling Language is optimization software for differential and algebraic equations. It is coupled with large-scale nonlinear programming solvers for data reconciliation, real-time optimization, dynamic simulation, and nonlinear predictive control. It is available as a free web service or for commercial licensing.

Try Example Optimization Problems
Browse or modify example problems to start solving nonlinear programming problems with up to 10 million variables through a web interface.

Documentation
APMonitor Documentation Wiki gives details of the modeling language and example applications. Compare to other popular modeling languages.

Discussion Forum AM Webinars
Users share experiences and collaborate through an online discussion forum and regularly scheduled webinars.

Premium Account Login
Registered users manage applications, view optimization results, and collaborate with other users.

APM Python Interface
Python gives users an open-source option for solving nonlinear programming problems with a growing community of users.

APM MATLAB Interface
MATLAB provides a powerful mathematical scripting language to improve the capability of optimization solutions.
Applications Deployed for Real-time Systems
Future Development Plans

- APM Modeling Language
  - MI-DAE systems
- Active Development Efforts
  - Mixed Integer solvers that exploit DAE structure
  - Interfaces to other scripting languages
- Industrial and Academic Collaborators
- APOPT and BPOPT MINLP solver development
  - Additional information at INFORMS session WC04