Executing models in less time—some solver insight

Pieter J. Mosterman

What all may affect simulation performance?

- Model
  - Initialization (images, ML script, set_params, …)
  - Execution
  - Code generation
- Solver
- Processing (Engine, Code Generator, Compiler)
  - Optimization
  - Diagnostics
- Interaction
  - Debugging
  - Logging
  - Viewing
- Simulation mode
- Platform
- Use scenarios
  - Simstate

The make up of a simulation

Comparing performance of different simulation modes

Help Search: comparing performance
Agenda

- Constructing models with increasing fidelity
  - Execution of models explained
    - Fixed step solver methods
    - Variable step solver methods
    - Zero crossings
  - Conclusions
Create executable models in all phases

Make the computational approximation the primary design deliverable—the real thing!
Agenda

- Constructing models with increasing fidelity
- Execution of models explained
  - A reference model
  - Fixed step solver methods
  - Variable step solver methods
  - Zero crossings
- Conclusions

A Simscape model—what solver to choose?
Using a Simscape solver to generate reference behavior

The corresponding Simulink model with an ideal diode
Agenda

- Constructing models with increasing fidelity
- Execution of models explained
  - A reference model
  - Fixed step solver methods
  - Variable step solver methods
  - Zero crossings
- Conclusions

ode1: forward Euler numerical integration

Step $h$ in time

$$h_k = t_{k+1} - t_k$$

Along instantaneous vector component

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

Gives the estimate

$$\hat{x}_k(t_{k+1}) = x(t_k) + \dot{x}(t_k)h_k$$

Comparison with Taylor series expansion

$$x(t_{k+1}) = x(t_k) + \frac{\dot{x}(t_k)}{1!}h_k + \frac{\ddot{x}(t_k)}{2!}h_k^2 + O(h_k^3)$$

Gives error of the estimate

$$\epsilon(t_{k+1}) = O(h_k^2)$$
ode1: stability analysis

\[ \dot{x} = kx; \quad x(0) = 1 \quad \Rightarrow x(t) = e^{kt} \]

\[ \hat{x}_e(t_{k+1}) = x(t_k) + \dot{x}(t_k)h_k \]

\[ \hat{x}_e(t_{k+1}) = \hat{x}_e(t_k) + kh\hat{x}_e(t_k) \Rightarrow \hat{x}_e(t_{k+n}) = (1 + kh)^n \hat{x}_e(t_k) \]

\[ k \in C, z = kh \Rightarrow \left\{ z \in C \mid |1 + z| < 1 \right\} \]

In MATLAB:

```
>> clear i
>> [X,Y]= meshgrid(-3:0.01:1,-3:0.01:3);
>> Mu = X + i*Y;
>> R = 1 + Mu;
>> Rhat = abs(R);
>> contour(X,Y ,Rhat,[1, 1],'b')
>> grid
>> hold
>> plot([0 0],[-3 3],'k')
```

Trapezoidal numerical integration

Average beginning and end point

\[ \hat{x}_e(t_{k+1}) = x(t_k) + \frac{\dot{x}(t_{k+1}) + \dot{x}(t_k)}{2} h_k \]

Finite difference …

\[ \ddot{x}(t_k) = \frac{\dot{x}(t_{k+1}) - \dot{x}(t_k)}{h_k} + O(h_k) \]

… to approximate Taylor series expansion

\[ x(t_{k+1}) = x(t_k) + \frac{\dot{x}(t_k)}{1!} h_k + \frac{\ddot{x}(t_k)}{2!} h_k^2 + \frac{\dot{x}(t_{k+1})}{2!} h_k^2 + O(h_k^3) \]
ode2: Heun method

- But, we do not have the value at $t_{k+1}$, because that is what we are trying to compute
- So, combine trapezoidal with Euler approximation

$$\hat{x}_i(t_{k+1}) = x(t_k) + \frac{\dot{x}(t_{k+1}) + \dot{x}(t_k)}{2} h_k$$

$$\dot{x}(t_k) = f(x(t_k), t_k)$$

$$\dot{x}(t_{k+1}) = f(x(t_k) + h\dot{x}(t_k), t_{k+1})$$

ode2: (Heun) stability

$$k \in C, z = kh \implies \left\{ z \in C \mid 1 + z + \frac{1}{2} z^2 < 1 \right\}$$

In MATLAB:

```matlab
>> clear;
>> [X,Y]= meshgrid(-3:0.01:1,-3:0.01:3);
>> Mu = X + i*Y;
>> R = 1 + Mu;
>> Rhat = abs(R);
>> contour(X,Y, Rhat, [1, 1], 'b')
>> grid
>> hold
>> plot([0 0], [-3 3], 'k')
>> R = 1 + Mu + .5*Mu.^2;
>> Rhat = abs(R);
>> contour(X,Y, Rhat, [1, 1], 'm')
```

Stability area of Runge-Kutta methods increases with order
In general, varying order of numerical solvers

One-stage

Two-stage

Multi-stage

For example: ode1, ode2, ode3, ode4, ode5, ode8

More accurate and stable as order increases

Stiff systems mix behavior at widely differing time scales

Parameters
- Forward voltage: 0.6
- On resistance: 7Ω
- Off conductance: 1e-02 1/Ohm

Electrical Reference

Stiffness Configuration

Current 2a

Voltage 2V

2 Ohm

1e-5 F

0.2

0.1

0.0

-0.1

-0.2

0

0.5

1

1e-05
Fixed step and solver order must be chosen carefully—ode1, 6e-5

Parameters:
- Forward voltage: 0.6 V
- On resistance: 7e-9 Ohm
- Off conductance: 1e-02 1.0m

Solver options:
- Type: Fixed-step
- Solver: ode1 (dassl)
- Fixed-step size (fundamental sample time): 6e-5

---

Fixed step and solver order must be chosen carefully—ode2, 6e-5

Parameters:
- Forward voltage: 0.6 V
- On resistance: 7e-9 Ohm
- Off conductance: 1e-02 1.0m

Solver options:
- Type: Fixed-step
- Solver: ode2 (bdf)
- Fixed-step size (fundamental sample time): 6e-5
Fixed step and solver order must be chosen carefully—ode3, 6e-5

- Parameters:
  - Forward voltage: 0.6 V
  - On resistance: 7e-6 Ohm
  - Off conductance: 1e-32 1.0nH

- Solver options:
  - Type: Fixed-step
  - Solver: ode3 (Bogacki-Shampine)
  - Fixed-step size (Fundamental sample time): 6e-5

---

Fixed step and solver order must be chosen carefully—ode4, 6e-5

- Parameters:
  - Forward voltage: 0.6 V
  - On resistance: 7e-6 Ohm
  - Off conductance: 1e-32 1.0nH

- Solver options:
  - Type: Fixed-step
  - Solver: ode4 (Runge-Kutta)
  - Fixed-step size (Fundamental sample time): 6e-5
Fixed step and solver order must be chosen carefully—ode5, 6e-5

**Parameters**
- Forward voltage: 0.6 V
- On resistance: 7e-9 Ohm
- Off conductance: 1e-32 1.0 Ohm

**Solver options**
- Type: Fixed-step
- Solver: ode5 (Dormand-Prince)
- Fixed-step size (fundamental sample time): 6e-5

---

Fixed step and solver order must be chosen carefully—ode8, 6e-5

**Parameters**
- Forward voltage: 0.6 V
- On resistance: 7e-9 Ohm
- Off conductance: 1e-32 1.0 Ohm

**Solver options**
- Type: Fixed-step
- Solver: ode8 (Dormand-Prince)
- Fixed-step size (fundamental sample time): 6e-5
Step size and order determine simulation time

Parameters
Forward voltage: 0.6 V
On resistance: 7e-9 0 Ohm
Off conductance: 1e-32 1 Ohm

Solver options
Type: Fixed-step
Solver: ode5 (Dormand-Prince)
Fixed-step size (fundamental sample time): 3.5e-5

Elapsed time is 2.035940 seconds
Step size and order determine simulation time

Two-stage Multi-stage

But these solvers really used an explicit formulation

One-stage

Two-stage

Multi-stage

Forward voltage: 0.6
On resistance: 7e-9 Ohm
Off conductance: 1e-32 1e-33m

Solver options
Type: Fixed-step
Solver: ode45 (Euler)
Fixed-step size (fundamental sample time): 1e-5
How about a truly implicit approach?

- Trapezoidal integration
  - Requires future points!
  - Impossible?
- Can do, but requires inverting the system
  \[
  \begin{align*}
  \dot{x}_i(t_{k+1}) &= x(t_k) + \frac{\dot{x}(t_{k+1}) + \dot{x}(t_k)}{2} h_k \\
  \dot{\dot{x}}_i(t_{k+1}) &= \dot{x}_i(t_k) + \frac{A \dot{x}_i(t_{k+1}) + A \dot{x}_i(t_k)}{2} h_k \\
  (2I - Ah)\ddot{x}_i(t_{k+1}) &= (2I + Ah)\dot{x}_i(t_k) \\
  \dot{x}_i(t_{k+1}) &= (2I - Ah)^{-1}(2I + Ah)\dot{x}_i(t_k)
  \end{align*}
  \]

Trapezoidal is stable in entire left half plane

But, expensive!
- Linearization
- Inversion

Plusses and deltas?

- Trapezoidal is stable in entire left half plane

In MATLAB:
```matlab
>> clear I
>> [X,Y]= meshgrid(-3:0.01:1,-3:0.01:3); >> Mu = X + i*Y; >> R = 1 + Mu; >> Rhat = abs(R); >> contour(X,Y,Rhat,
>> grid
>> hold
>> plot([0 0],[3 3],')
>> Rhat = abs(R); >> contour(X,Y,Rhat,[1, 1],')
>> R = (2 + Mu)/(2-Mu); >> Rhat = abs(R); >> contour(X,Y,Rhat,[1, 1],')
```
Advantage: a large step size is possible

Still stable with very large step size
A fixed step size has to be stable for the fastest behavior anywhere in a simulation

From k-1 to k
Fast behavior, so a small step is necessary

From k to k+1
Slow behavior, so a large step is possible

Is the ‘fast’ behavior holding us hostage?
Step size control

- Compute Euler approximation
  \[ \hat{x}_e(t_{k+1}) = x(t_k) + f(x(t_k))h_k \]
- Compute Heun approximation
  \[ \hat{x}_h(t_{k+1}) = x(t_k) + \frac{f(\hat{x}_e(t_{k+1})) + f(x(t_k))}{2} h_k \]
- Compare the results for the error estimate, \( \varepsilon \)
  \[ x_e(t_{k+1}) - x_h(t_{k+1}) \approx \frac{\hat{x}(t_k)}{2!} h_k^2 = \varepsilon_{e}(t_{k+1}) \]
- Reduce step size from \( h_{max} \) till \( \varepsilon < tol \)
  - For example, bisection

Tolerance consists of two components

- Relative tolerance depends on signal magnitude
- Absolute tolerance is constant
Approximations determined by absolute and relative tolerance

- Absolute tolerance prevents infinitely accurate solution around 0

Combined relative and absolute tolerance drive solver step size selection
Variable step solvers

- **ode45**
  - Compares RK methods of order 4 and 5
- **ode23**
  - Compares RK methods of order 2 and 3
- **ode23tb**
  - Trapezoidal (implicit) integration with BDF error estimate

But now we better be careful with our computational complexity?

A variable-step solver may actually take much less time to simulate
And we can do even better: multi-step solvers reuse effort

\[ x_{k+1} = x_k + k \cdot t^{\text{multi-stage}} \]

Implicit multi-stage (e.g., ode23tb, ode23t)

Explicit multi-step (e.g., ode113)

Implicit multi-step (e.g., ode15s)

So, yet more efficient integration

ode45: 1.055492
ode113: 0.886697

multi-stage is \(-19\%\) slower than multi-step
But the reuse may turn against us

We do not have initial values at $t_2$ and $t_1$

How do we build this history?
- Single-step integration algorithm
- Either implicit or explicit with very small step size, $\varepsilon$

How about $t_1$ in case of implicit methods?
- Linearize system and invert system matrix

But now when we reach a discontinuity …

… smoothness is violated … solver reset!

Many discontinuities require many solver resets
A single-step solver ...

...versus a multi-step solver ...
... and another multi-step solver ...

But there is more to it than just the single step nature
Or even ...

Note: extrapolation becomes unreliable for high order

Stability area of multistep methods (explicit: ode113, implicit: ode15s) decreases with order!
Agenda

- Constructing models with increasing fidelity
- Execution of models explained
  - Fixed step solver methods
  - Variable step solver methods
  - Zero crossings
- Conclusions

Discontinuities

- Variable-step solvers ‘zoom in’ on zero crossings
  - Is solver step size reduction an efficient mechanism?
Discontinuities

- How about we use a dedicated root-finding algorithm?
  - Bisection, Newton-Raphson
- Disregard the discontinuity, then search

\[
\begin{align*}
    x_k - x_{k+1} \\
    t_k & \quad t_{k+1} \\
    x_k & \quad x_{k+1} \\
    t_k & \quad t_{k+1} \\
    x_k & \quad x_{k+1}
\end{align*}
\]

Reduced simulation time for ode23t

Zero-crossing options
- Zero-crossing control: Use local settings
- Algorithm: Nonadaptive
- Time tolerance: $10^{-4}$
- Signal threshold: Auto
- Number of consecutive zero crossings: 1000

```
>> tic;sim(gcs,'reltol','1e-4','Solver','ode23t','stoptime','0.5');toc
Elapsed time is 6.255886 seconds.
```

```
>> tic;sim(gcs,'reltol','1e-4','Solver','ode23t','stoptime','0.5');toc
Elapsed time is 5.634365 seconds.
```
But not for ode113 ... because it resets its history anyway

Is computational complexity the only zero crossing issue?
Chattering in our electrical circuit

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Forward voltage</td>
<td>0.6 V</td>
</tr>
<tr>
<td>On resistance</td>
<td>7n-9</td>
</tr>
<tr>
<td>Off conductance</td>
<td>1n-2</td>
</tr>
</tbody>
</table>

Zero-crossing options
- Use local settings: adaptive
- Algorithm: adaptive
- Time tolerance: $10^{-10}$ s
- Signal threshold: auto
- Number of consecutive zero crossings: 1000

Resolve the chattering by adaptive zero crossings

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Forward voltage</td>
<td>0.6 V</td>
</tr>
<tr>
<td>On resistance</td>
<td>7n-9</td>
</tr>
<tr>
<td>Off conductance</td>
<td>1n-2</td>
</tr>
</tbody>
</table>

Zero-crossing options
- Use local settings: adaptive
- Algorithm: adaptive
- Time tolerance: $10^{-10}$ s
- Signal threshold: auto
- Number of consecutive zero crossings: 1000

Example code snippet:
```matlab
tic; sim('myModel', 'variable_step', 'solver', 'ode45', ... 'ZeroCrossingAlgorithm', 'adaptive', 'StopTime', 10); toc
Elapsed time is 3.01231 seconds.
```
Or, employ a Simscape fixed-step solver with fixed cost

But a fixed-cost solver may be less accurate
But a fixed-cost solver may be less accurate

Agenda

- Constructing models with increasing fidelity
- Execution of models explained
  - Fixed step solver methods
  - Variable step solver methods
  - Zero crossings
- Conclusions
Conclusions

- Solvers selection is a trade off
  - Stability
  - Accuracy
  - Time to simulate
- Compare different solvers and solver parameters
  - Use a computationally expensive variable-step solver to generate a reference solution
  - If the model is sensitive, investigate why

Characterization

- My model
  - Must have a predictable execution time (could be long)
  - Must have a short simulation time
  - Has widely varying time constants (stiff)
  - Includes discontinuities
  - Includes many discontinuities
  - Exhibits chattering
  - Should not be dissipative
- My solver
  - Fixed step vs. variable step
  - Explicit vs. implicit
  - Single step vs. multi step
  - Zero crossing location or not

NB: Fixed step solvers do not do root finding
Characterization

- **My model**
  - Must have a predictable execution time (could be long)
  - Must have a short simulation time
  - Has widely varying time constants (stiff)
  - Includes discontinuities
  - Includes many discontinuities
  - Exhibits chattering
  - Should not be dissipative

- **My solver**
  - Fixed-step integration
    - Experiment with integration order, step size, and accuracy
  - Variable-step integration
    - Multi-step methods are preferred
    - Zero-crossing on may shorten simulation time
Characterization

- My model
  - Must have a predictable execution time (could be long)
  - Must have a short simulation time
  - Has widely varying time constants (stiff)
  - Includes discontinuities
  - Includes many discontinuities
  - Exhibits chattering
  - Should not be dissipative

- My solver
  - Preferably implicit
  - Fixed step
    - ode14x
  - Variable step
    - ode15s, ode23s, ode23t, ode23tb

Characterization

- My model
  - Must have a predictable execution time (could be long)
  - Must have a short simulation time
  - Has widely varying time constants (stiff)
  - Includes discontinuities
  - Includes many discontinuities
  - Exhibits chattering
  - Should not be dissipative

- My solver
  - Variable-step integration
    - No integration history (at least single step) with zero crossing location
      - ode23t
Characterization

- **My model**
  - Must have a predictable execution time (could be long)
  - Must have a short simulation time
  - Has widely varying time constants (stiff)
  - Includes discontinuities
  - Includes many discontinuities
  - Exhibits chattering
  - Should not be dissipative

- **My solver**
  - Fixed-step integration
    - If the accuracy suffices
  - Variable-step integration
    - Adaptive zero crossing location

Characterization

- **My model**
  - Must have a predictable execution time (could be long)
  - Must have a short simulation time
  - Has widely varying time constants (stiff)
  - Includes discontinuities
  - Includes many discontinuities
  - Exhibits chattering
  - Should not be dissipative

- **My solver**
  - Fixed-step integration
    - If the accuracy suffices
  - Variable-step integration
    - Adaptive zero crossing location
Characterization

- **My model**
  - Must have a predictable execution time (could be long)
  - Must have a short simulation time
  - Has widely varying time constants (stiff)
  - Includes discontinuities
  - Includes many discontinuities
  - Exhibits chattering
  - Should not be dissipative

- **My solver**
  - Nondissipative integration method
    - ode23t

And ...

Do not fall in love with a model 😊

-- Jacques LeFèvre