Executing models in less time—some solver insight

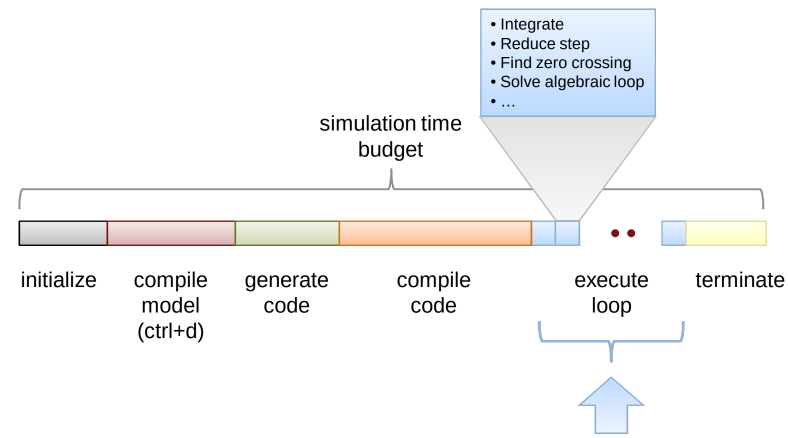
Pieter J. Mosterman

What all may affect simulation performance?

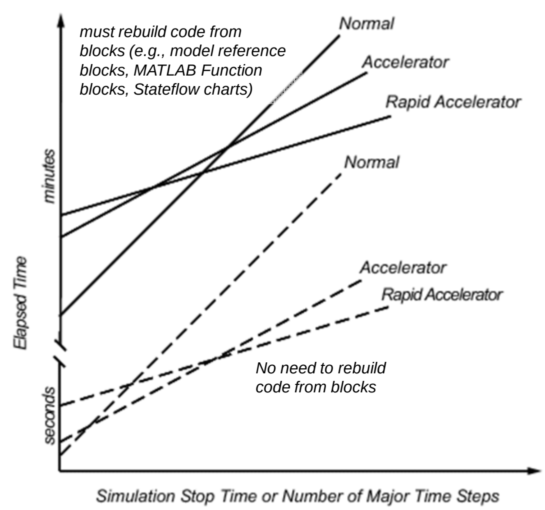
- Model
  - Initialization (images, ML script, set_params, …)
  - Execution
  - Code generation
- **Solver**
- Processing (Engine, Code Generator, Compiler)
  - Optimization
  - Diagnostics
- Interaction
  - Debugging
  - Logging
  - Viewing
- Simulation mode
- Platform
- Use scenarios
  - Simstate

http://www.mathworks.com/company/events/webinars/webinarconf.html?id=54186&language=en

The make up of a simulation



Comparing performance of different simulation modes

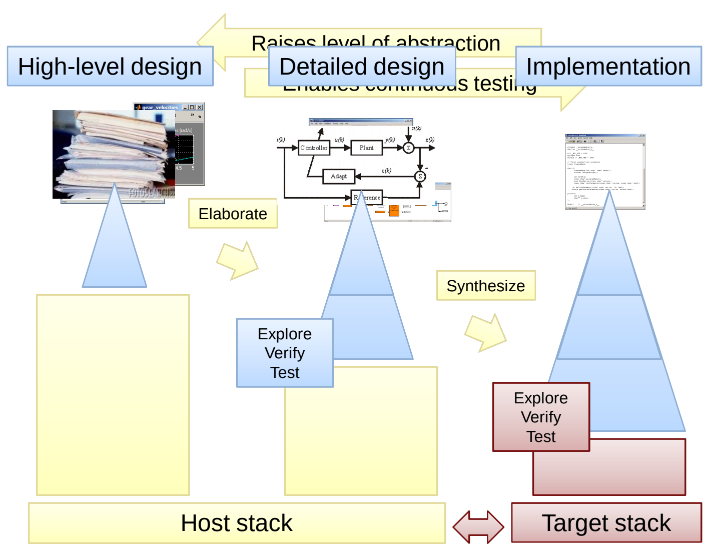


Help Search: comparing performance

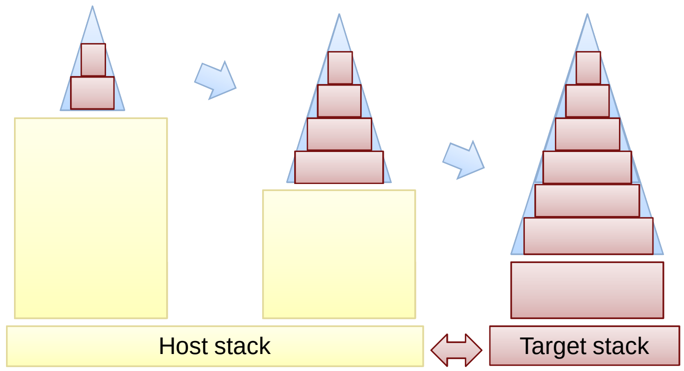
Agenda

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

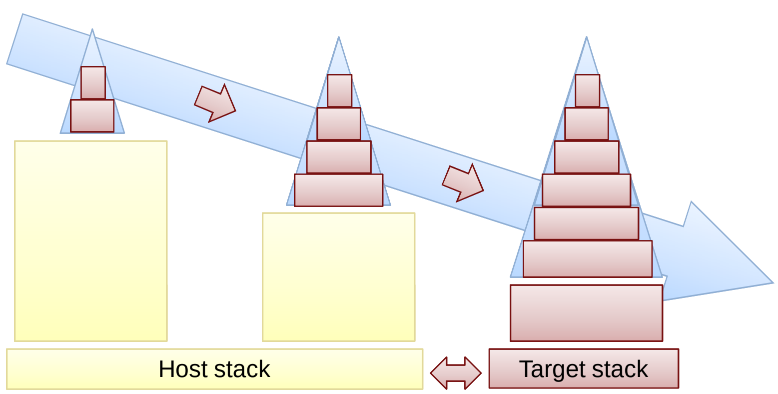
Model-Based Design



Create executable models in all phases



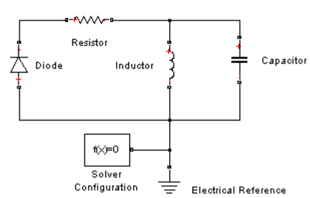
Make the computational approximation the primary design deliverable—the real thing!

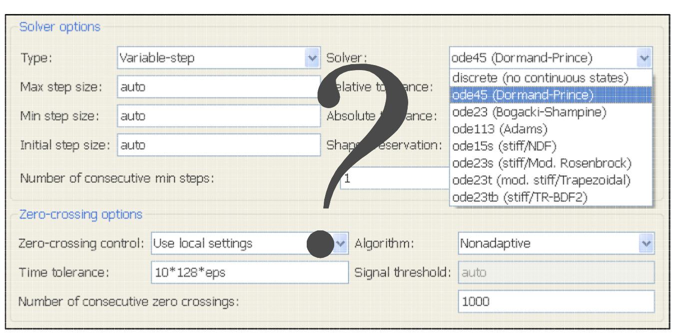


Agenda

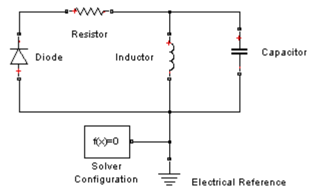
- Constructing models with increasing fidelity
- Execution of models explained
  - A reference model
  - Fixed step solver methods
  - Variable step solver methods
  - Zero crossings
- Conclusions

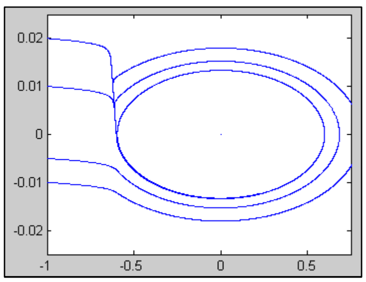
A Simscape model—what solver to choose?

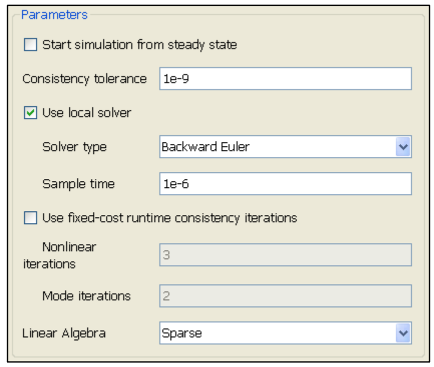




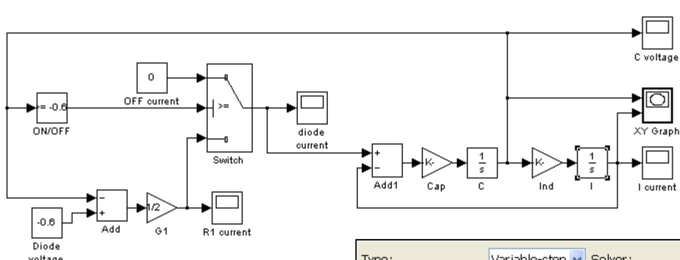
Using a Simscape solver to generate reference behavior

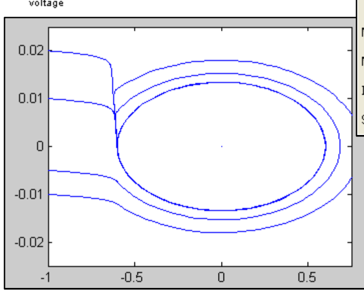


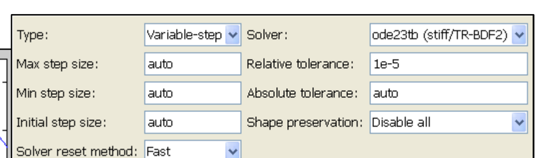




The corresponding Simulink model with an ideal diode







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

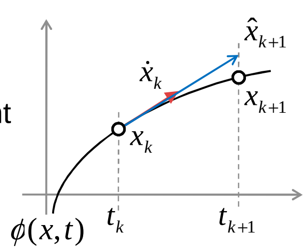
$$\hat{x}_e(t_{k+1}) = x(t_k) + \dot{x}(t_k)h_k$$

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

$$\varepsilon(t_{k+1}) = O(h_k^2)$$



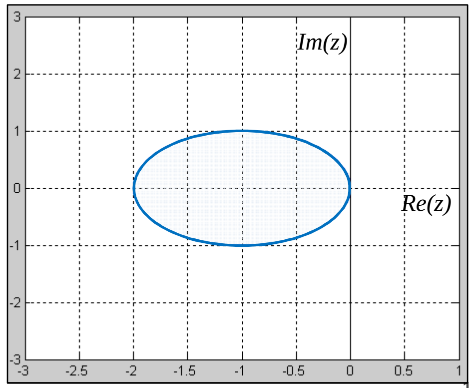
ode1: stability analysis

$$\dot{x} = kx; \quad x(0) = 1 \qquad\qquad x(t) = e^{kt}$$

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

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

$$k \in C, z = kh \Rightarrow \{z \in C \,|\, |1+z| < 1\}$$



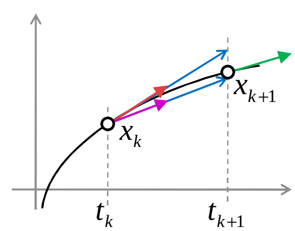
In MATLAB:

```
>> 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

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



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

$$x(t_{k+1}) = \underbrace{\boxed{x(t_k) + \frac{\dot{x}(t_k)}{1!} h_k + \frac{\ddot{x}(t_k)}{2!} h_k^2}}_{\hat{x}_t(t_{k+1})} + \underbrace{\boxed{O(h_k^3)}}_{\varepsilon(t_{k+1})}$$

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

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

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

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

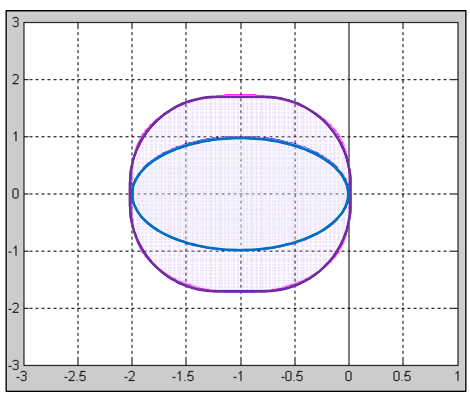
$$\dot{x}(t_{k+1}) = f\left(\boxed{x(t_k) + h\dot{x}(t_k)}, t_{k+1}\right)$$

ode2: (Heun) stability

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

In MATLAB:

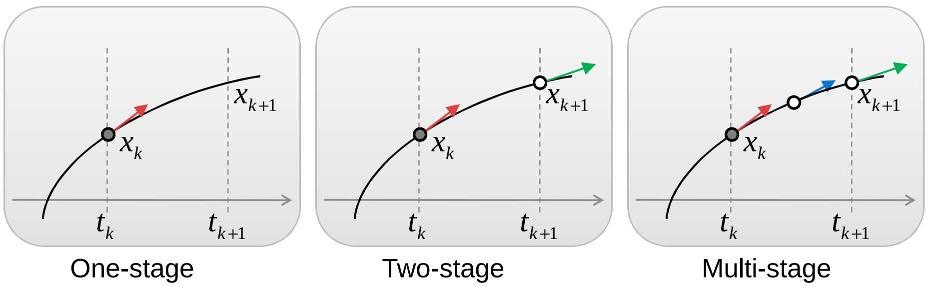
```
>> clear i
>> [X,Y]= meshgrid(-3:0.01:1,-3:0.01:3);
>> Mu = X + i*Y;
>> R = 1 + Mu;
>> Rhat = abs(R);
>> contour(X,Y,Rhat,[1, 1],'b')
>> grid
>> hold
>> plot([0 0],[-3 3],'k')
>> R = 1 + Mu + .5*Mu.^2;
>> Rhat = abs(R);
>> contour(X,Y,Rhat,[1, 1],'m')
```



Stability area of Runge-Kutta methods increases with order

MathWorks

In general, varying order of numerical solvers

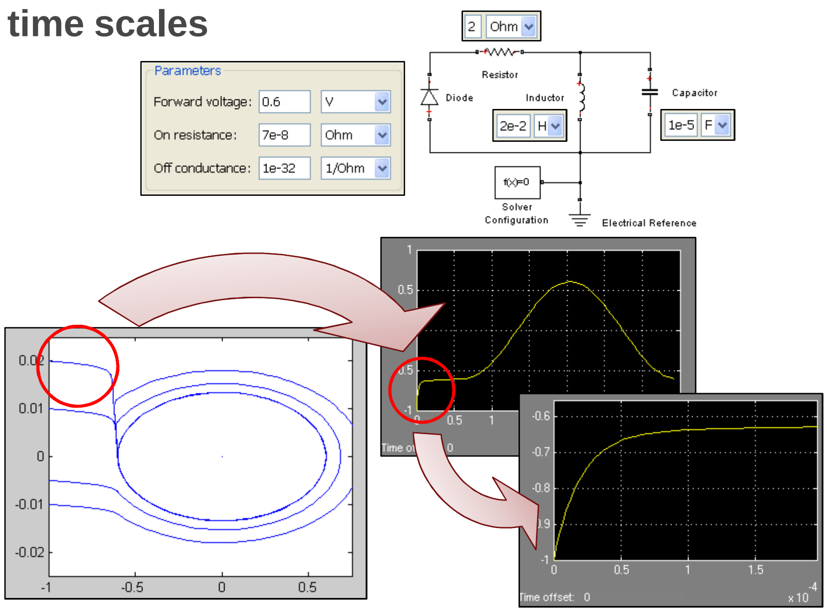


For example: ode1, ode2, ode3, ode4, ode5, ode8

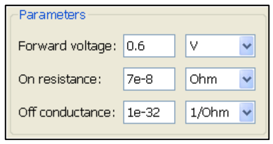
More accurate and stable as order increases

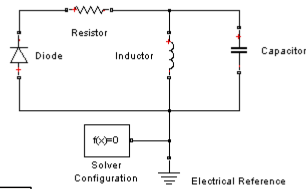
MathWorks

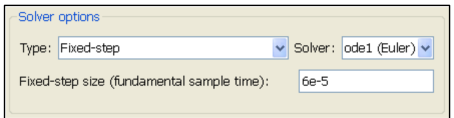
Stiff systems mix behavior at widely differing time scales

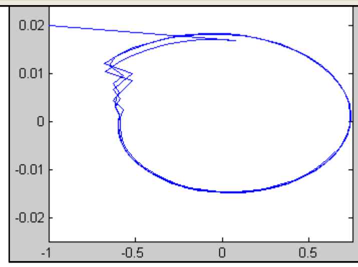


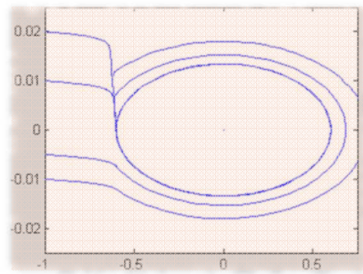
Fixed step and solver order must be chosen carefully—ode1, 6e-5





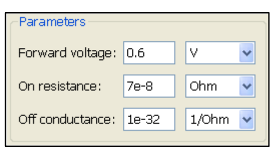


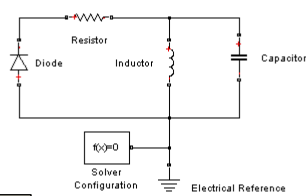


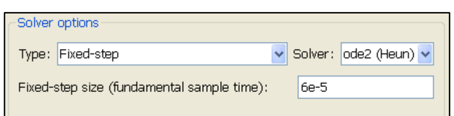


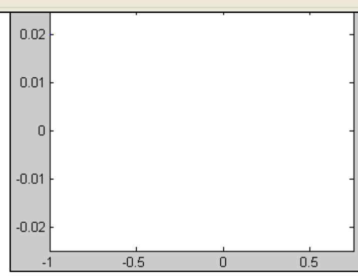
21

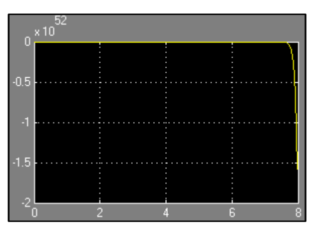
Fixed step and solver order must be chosen carefully—ode2, 6e-5





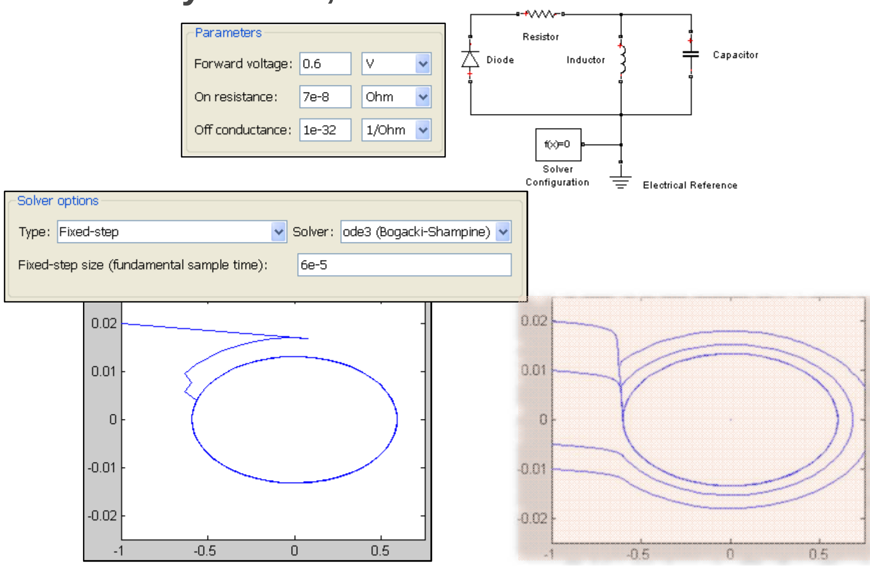




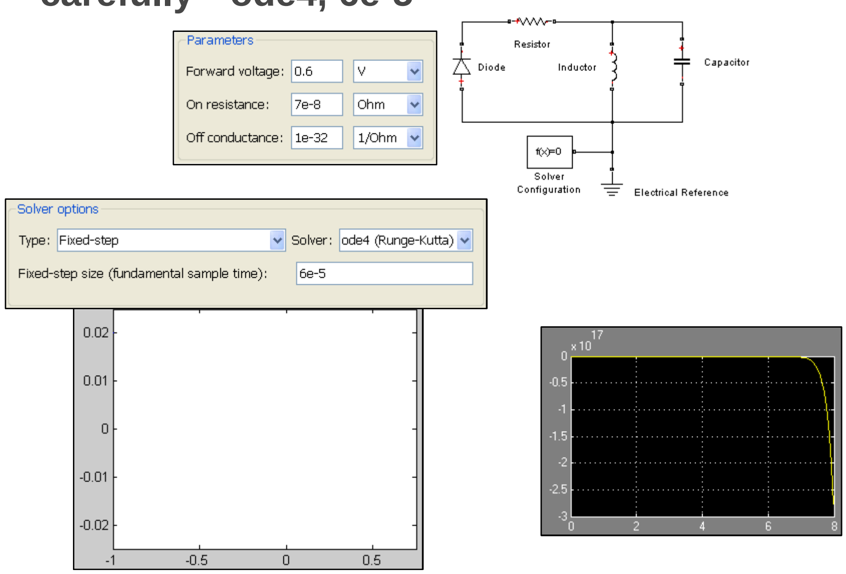


22

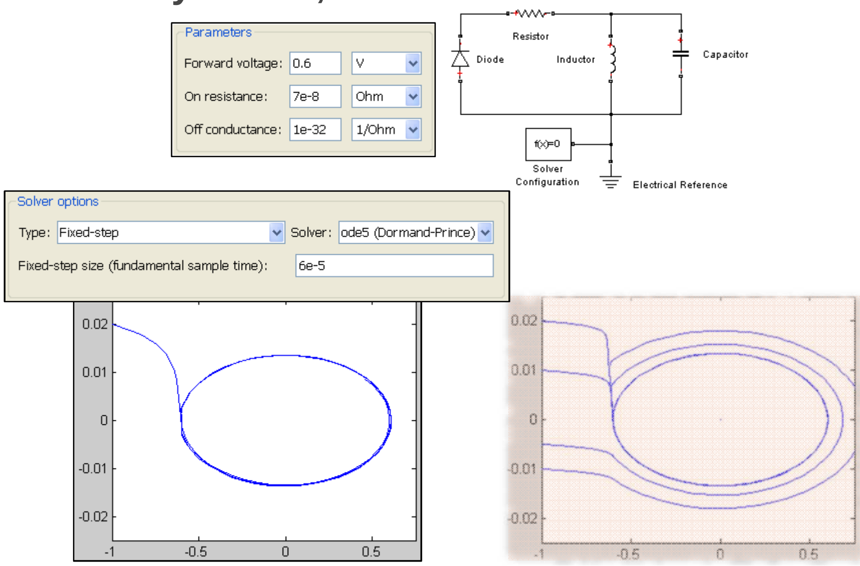
Fixed step and solver order must be chosen carefully—ode3, 6e-5



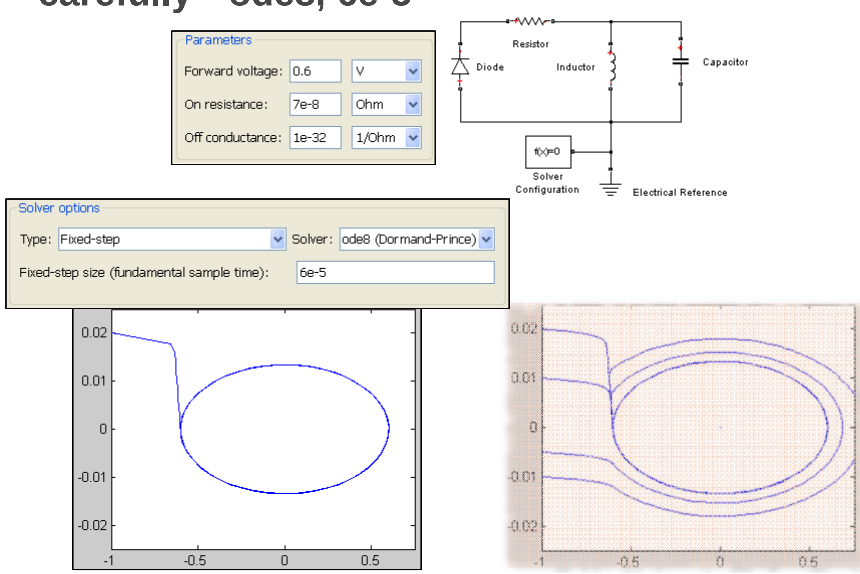
Fixed step and solver order must be chosen carefully—ode4, 6e-5



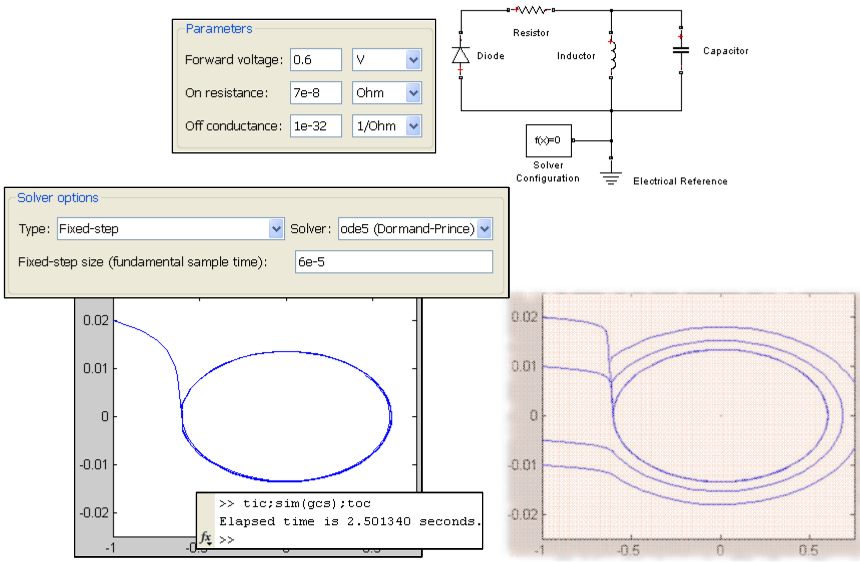
Fixed step and solver order must be chosen carefully—ode5, 6e-5



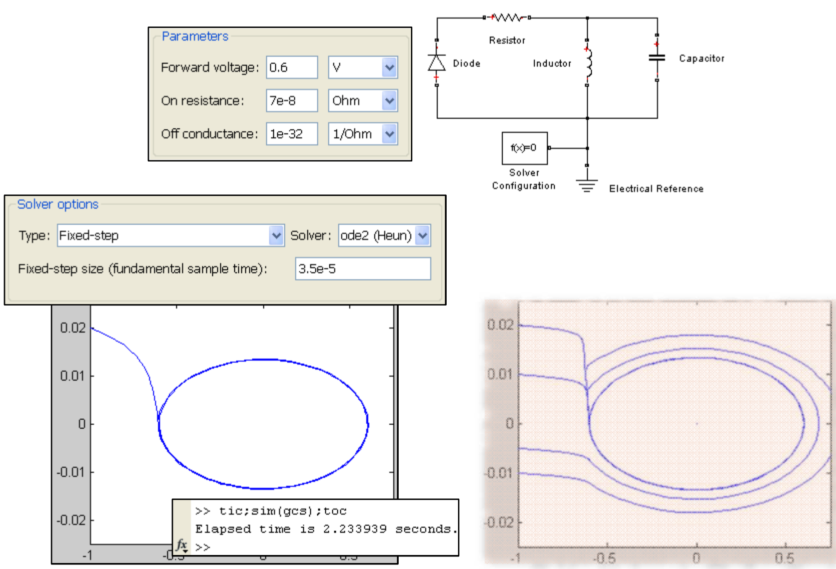
Fixed step and solver order must be chosen carefully—ode8, 6e-5



Step size and order determine simulation time

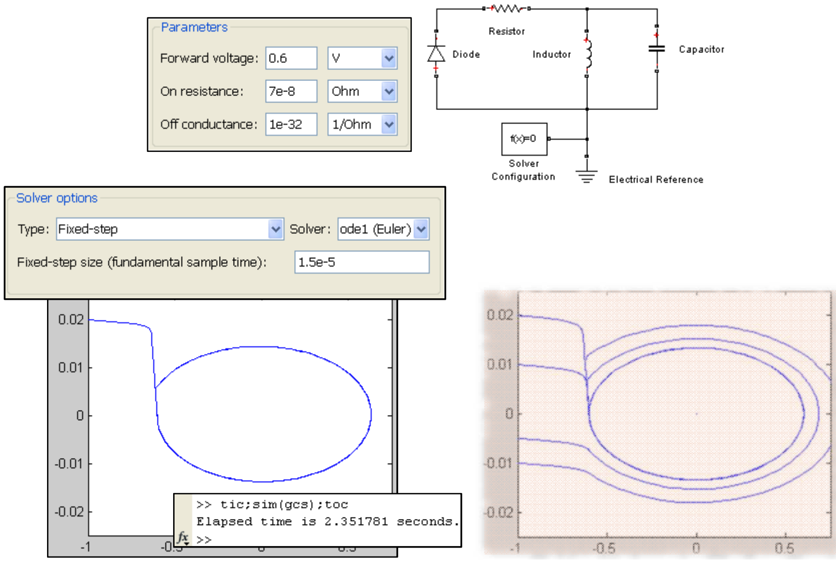


Step size and order determine simulation time



MathWorks

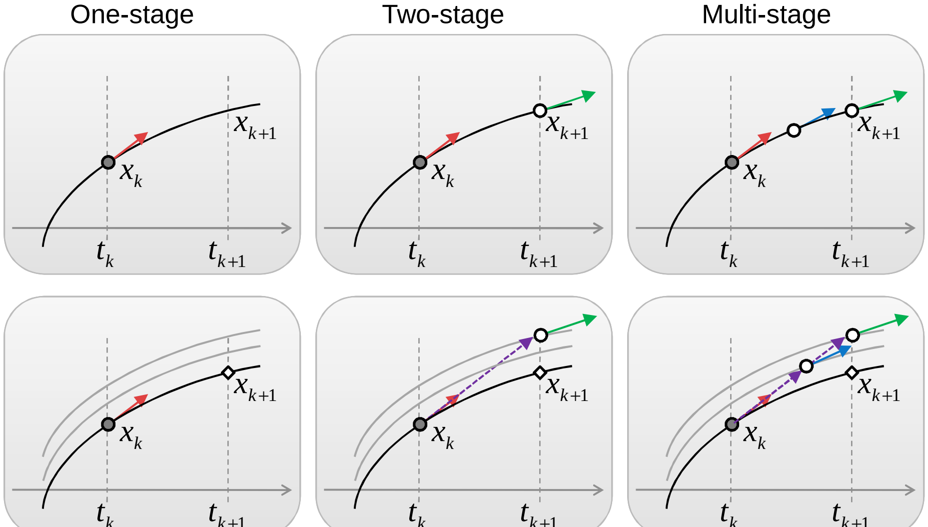
Step size and order determine simulation time



MathWorks

But these solvers really used an explicit formulation

One-stage Two-stage Multi-stage



How about a truly implicit approach?

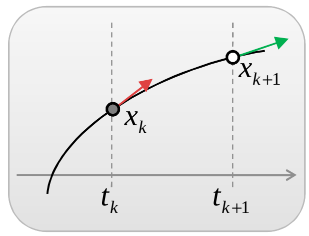
- Trapezoidal integration
  - Requires future points!
  - Impossible?
- Can do, but requires inverting the system

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

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

$$(2I - Ah)\hat{x}_t(t_{k+1}) = (2I + Ah)\hat{x}_t(t_k)$$

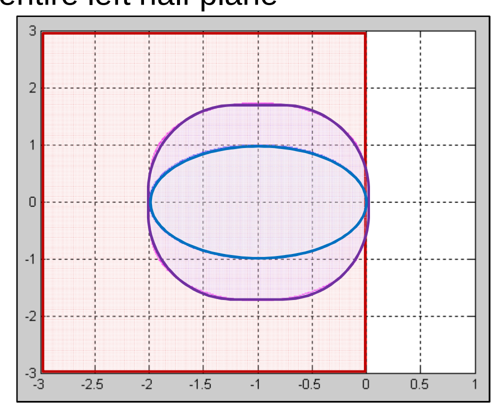
$$\hat{x}_t(t_{k+1}) = (2I - Ah)^{-1}(2I + Ah)\hat{x}_t(t_k)$$



Plusses and deltas?

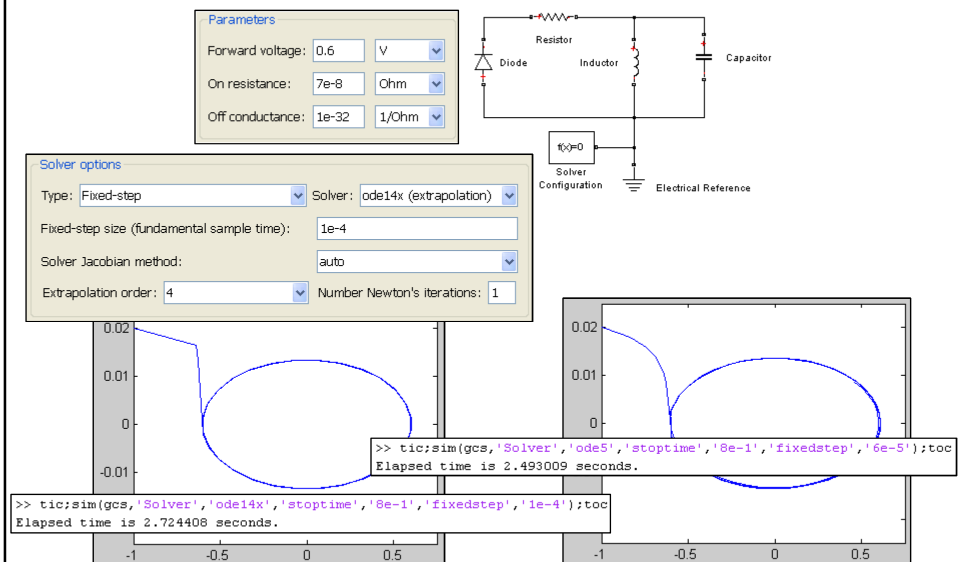
- Trapezoidal is stable in entire left half plane

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')
>> R = 1 + Mu + .5*Mu.^2;
>> Rhat = abs(R);
>> contour(X,Y,Rhat,[1, 1],'m')
>> R = (2 + Mu)./(2-Mu);
>> Rhat = abs(R);
>> contour(X,Y,Rhat,[1, 1],'r')
```

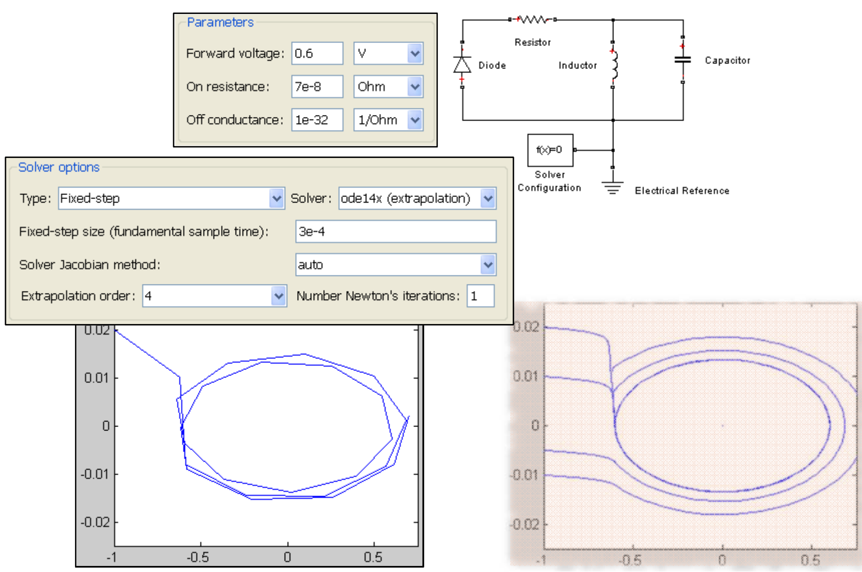


- But, expensive!
  - Linearization
  - Inversion

Advantage: a large step size is possible



Still stable with very large step size

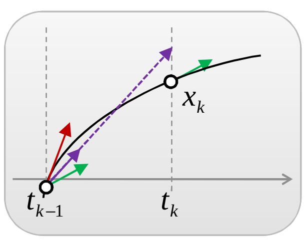


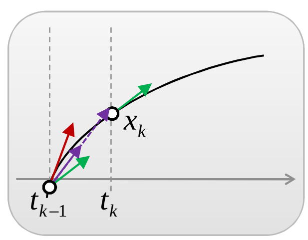
Agenda

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

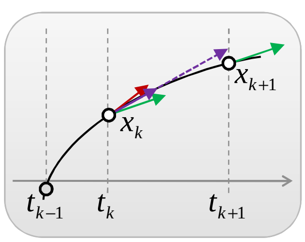
A fixed step size has to be stable for the fastest behavior anywhere in a simulation

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

$$\hat{x}_e(t_{k+1}) = x(t_k) + f\big(x(t_k)\big)h_k$$

- Compute Heun approximation

$$\hat{x}_t(t_{k+1}) = x(t_k) + \frac{f\big(\hat{x}_e(t_{k+1})\big) + f\big(x(t_k)\big)}{2} h_k$$

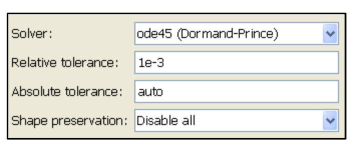
- Compare the results for the error estimate, ε

$$x_e(t_{k+1}) - x_t(t_{k+1}) \approx \frac{\ddot{x}(t_k)}{2!} h_k^2 = \varepsilon_e(t_{k+1})$$

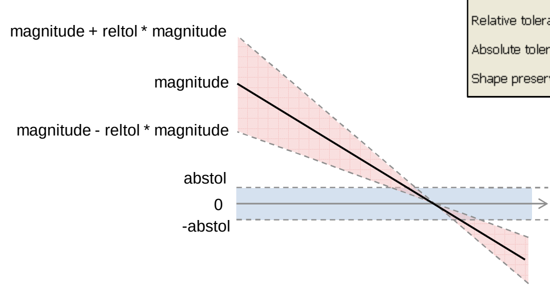
- Reduce step size from h_{max} till $\varepsilon < tol$
  - For example, bisection

Tolerance consists of two components

- Relative tolerance depends on signal magnitude
- Absolute tolerance is constant

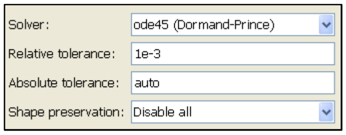


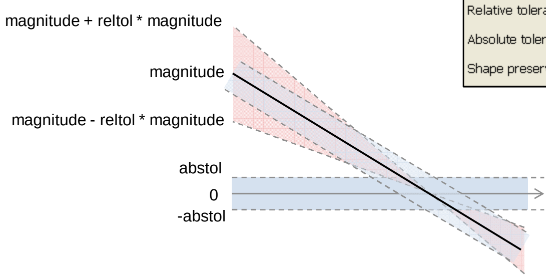
Solver:	ode45 (Dormand-Prince)
Relative tolerance:	1e-3
Absolute tolerance:	auto
Shape preservation:	Disable all



Approximations determined by absolute and relative tolerance

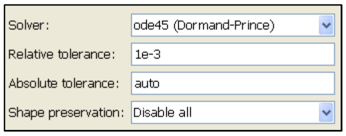
- Absolute tolerance prevents infinitely accurate solution around 0

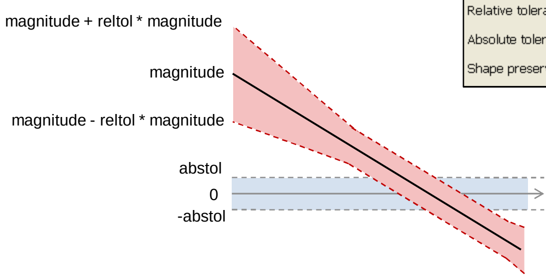




Approximations determined by absolute and relative tolerance

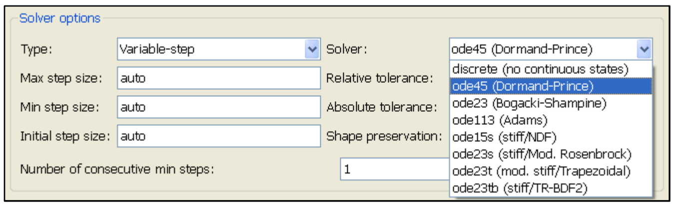
- Combined relative and absolute tolerance drive solver step size selection





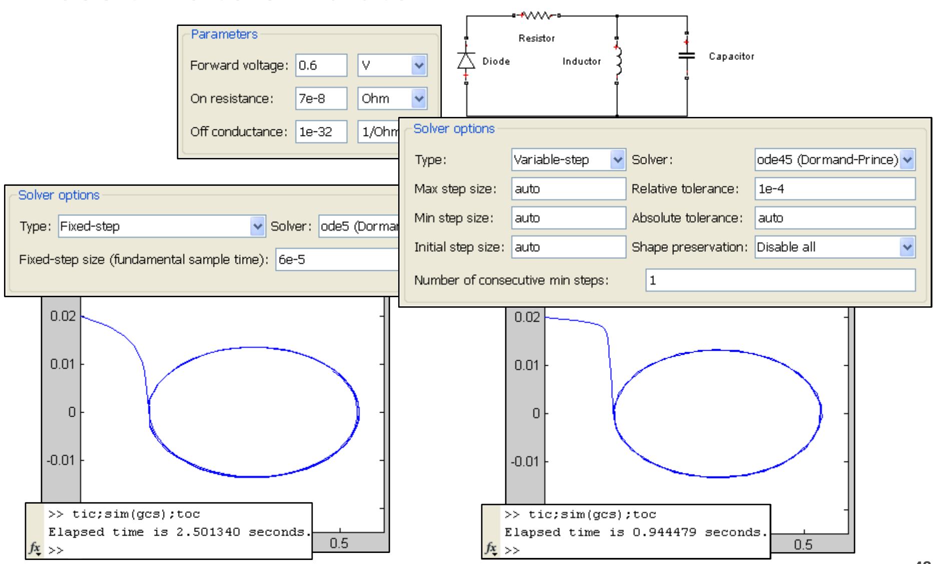
Variable step solvers

- 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

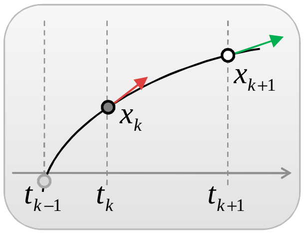


But now we better be careful with our computational complexity?

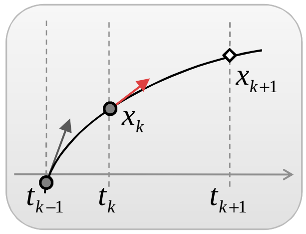
A variable-step solver may actually take much less time to simulate



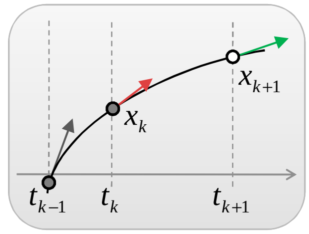
And we can do even better: multi-step solvers reuse effort



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

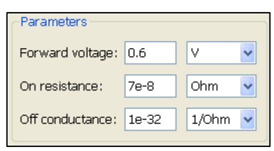


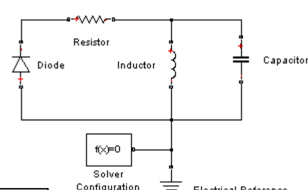
Explicit multi-step
(e.g., ode113)



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

So, yet more efficient integration





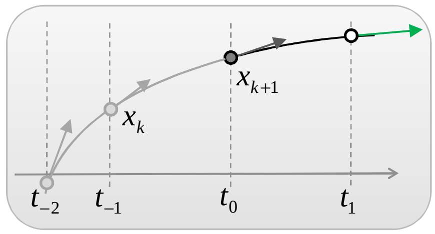

```
>> tic;sim(gcs,'reltol','1e-4','Solver','ode45','stoptime','0.8');toc
Elapsed time is 1.025632 seconds.
>> tic;sim(gcs,'reltol','1e-4','Solver','ode45','stoptime','0.8');toc
Elapsed time is 0.991488 seconds.
>> tic;sim(gcs,'reltol','1e-4','Solver','ode45','stoptime','0.8');toc
Elapsed time is 1.169297 seconds.
>> tic;sim(gcs,'reltol','1e-4','Solver','ode45','stoptime','0.8');toc
Elapsed time is 1.082161 seconds.
>> tic;sim(gcs,'reltol','1e-4','Solver','ode45','stoptime','0.8');toc
Elapsed time is 1.008882 seconds.
>> tic;sim(gcs,'reltol','1e-4','Solver','ode113','stoptime','0.8');toc
Elapsed time is 0.895317 seconds.
>> tic;sim(gcs,'reltol','1e-4','Solver','ode113','stoptime','0.8');toc
Elapsed time is 0.921423 seconds.
>> tic;sim(gcs,'reltol','1e-4','Solver','ode113','stoptime','0.8');toc
Elapsed time is 0.848121 seconds.
>> tic;sim(gcs,'reltol','1e-4','Solver','ode113','stoptime','0.8');toc
Elapsed time is 0.902069 seconds.
>> tic;sim(gcs,'reltol','1e-4','Solver','ode113','stoptime','0.8');toc
Elapsed time is 0.866556 seconds.
>>
```

ode45: 1.055492

ode113: 0.8866972

multi-stage is ~19% slower than multi-step

But the reuse may turn against us

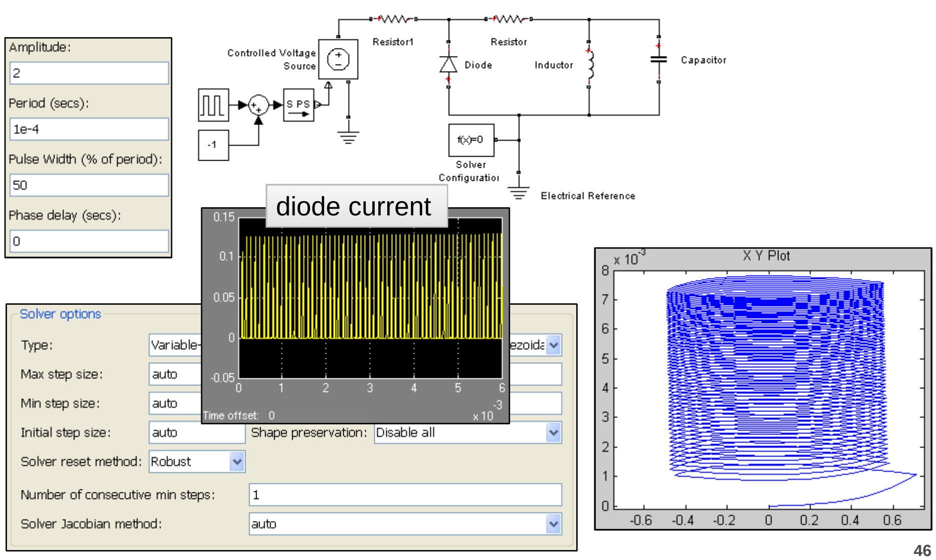


We do not have initial values at t_{-2} and t_{-1}
How do we build this history?
- Single-step integration algorithm
- Either implicit or explicit with very small step size, ε

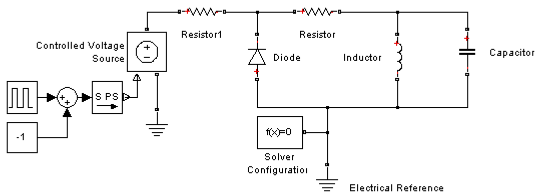
How about t_1 in case of implicit methods?
- Linearize system and invert system matrix

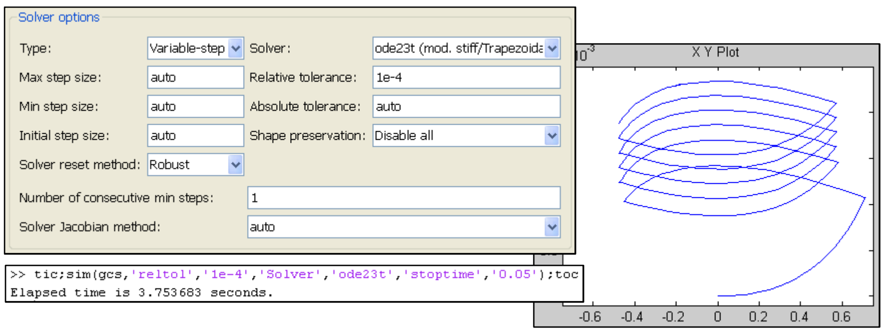
But now when we reach a discontinuity …
… smoothness is violated … solver reset!

Many discontinuities require many solver resets

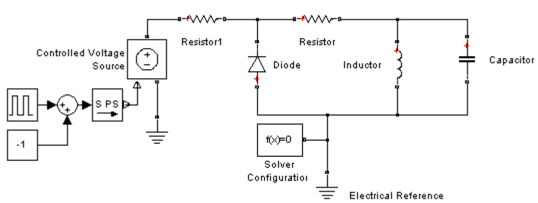


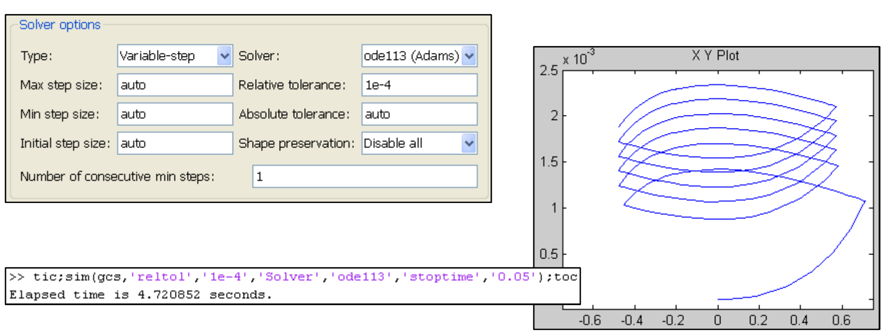
A single-step solver …





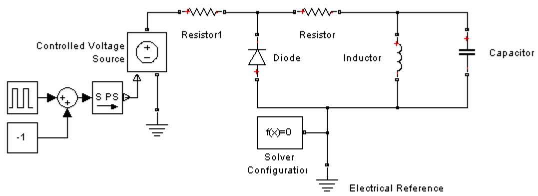
…versus a multi-step solver …

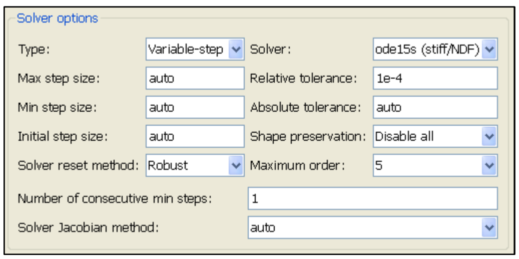




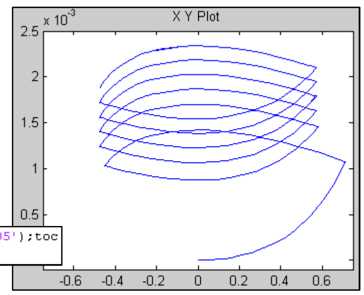
MathWorks

… and another multi-step solver …



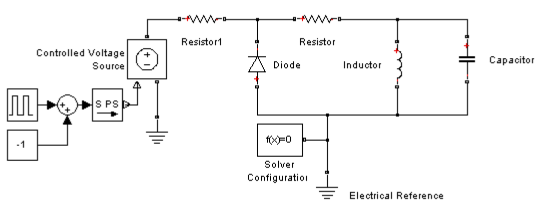


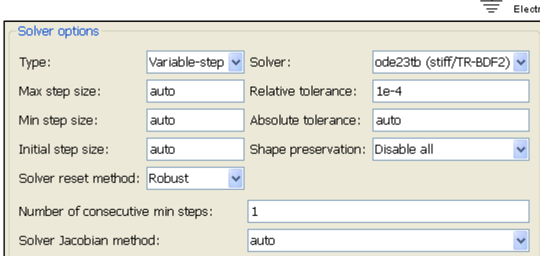

```
>> tic;sim(gcs,'reltol','1e-4','Solver','ode15s','stoptime','0.05');toc
Elapsed time is 4.505691 seconds.
```



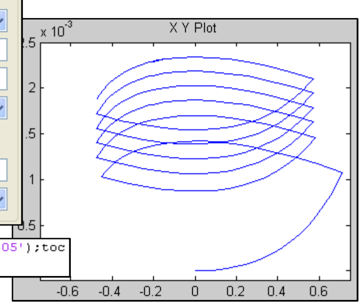
MathWorks

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

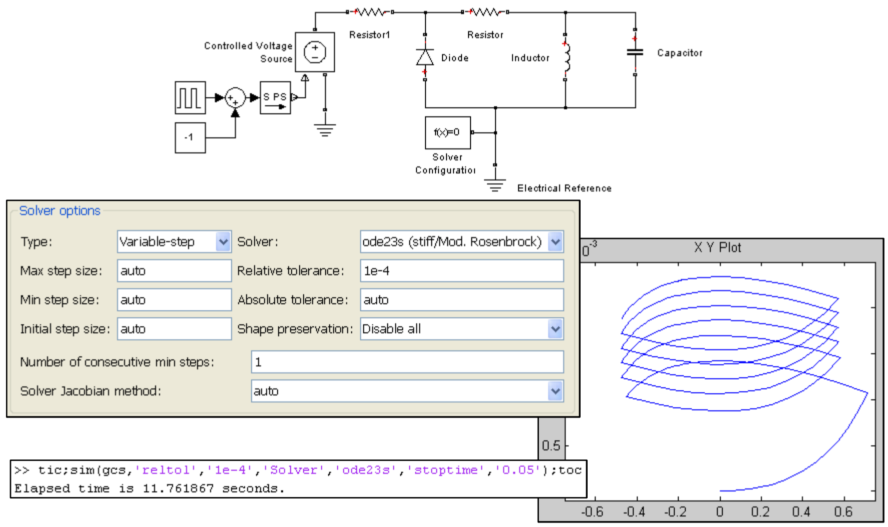




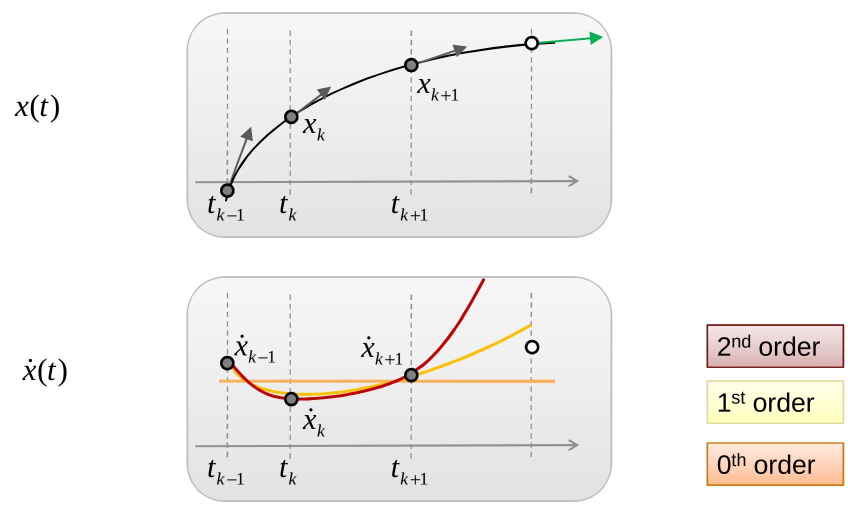

```
>> tic;sim(gcs,'reltol','1e-4','Solver','ode23tb','stoptime','0.05');toc
Elapsed time is 5.529495 seconds.
```



Or even …



Note: extrapolation becomes unreliable for high order



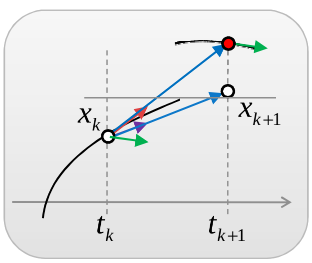
Stability area of multistep methods (explicit: ode113, implicit: ode15s) decreases with order!

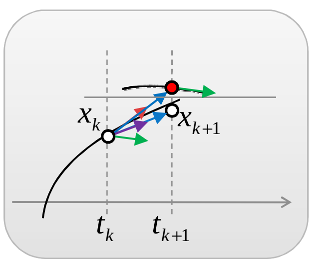
Agenda

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

Discontinuities

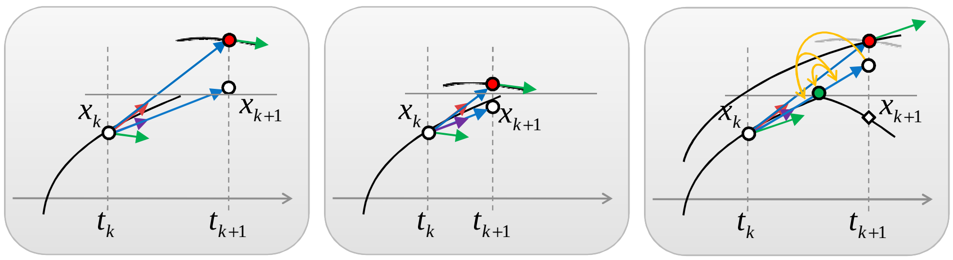
- Variable-step solvers ‘zoom in’ on zero crossings
  - Is solver step size reduction an efficient mechanism?

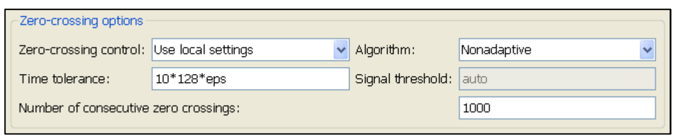




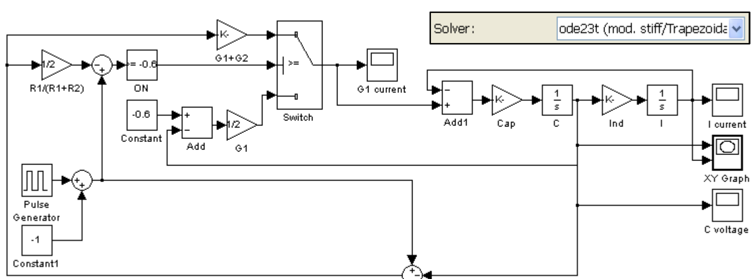
Discontinuities

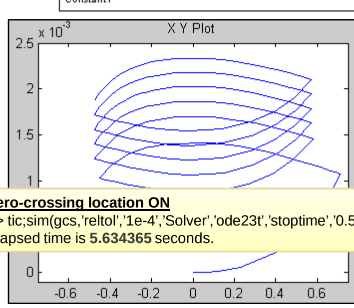
- How about we use a dedicated root-finding algorithm?
  - Bisection, Newton-Raphson
- Disregard the discontinuity, then search



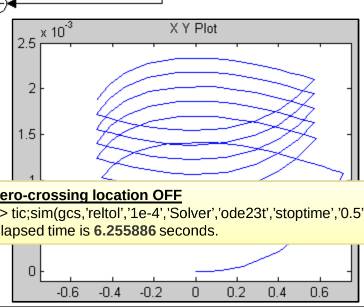


Reduced simulation time for ode23t



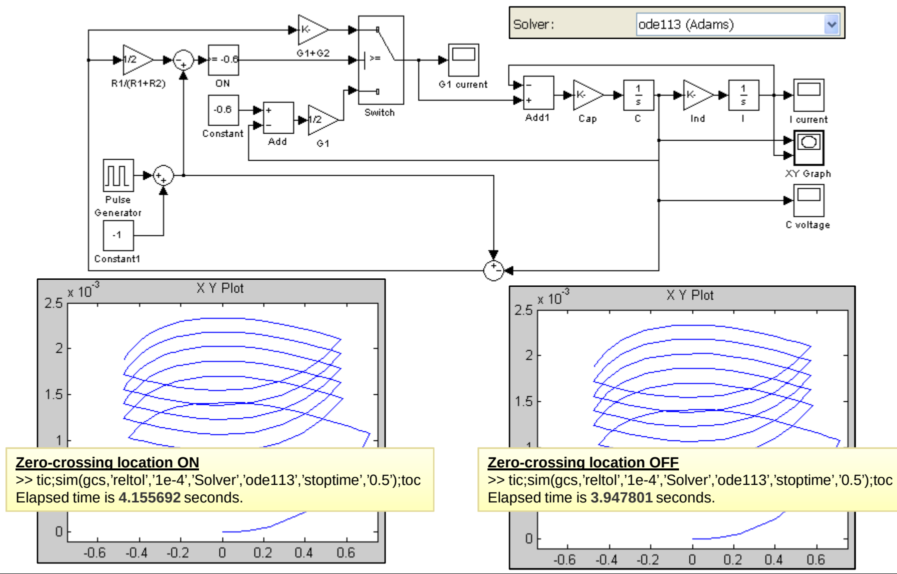


Zero-crossing location ON
>> tic;sim(gcs,'reltol','1e-4','Solver','ode23t','stoptime','0.5');toc
Elapsed time is **5.634365** seconds.



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

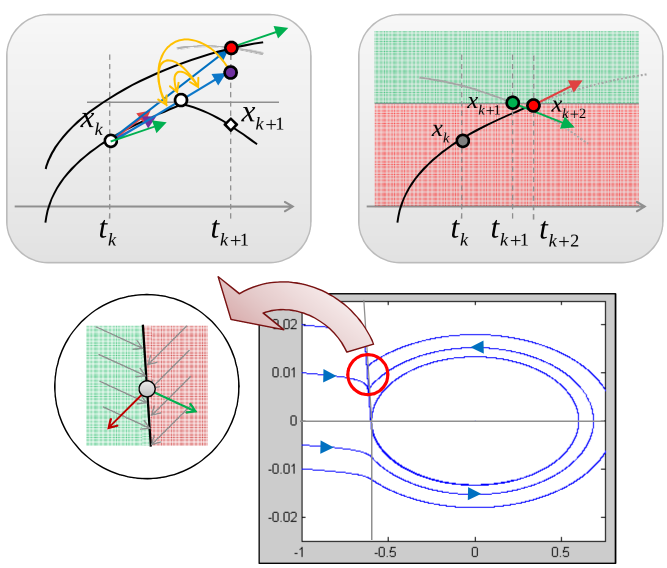
But not for ode113 … because it resets its history anyway



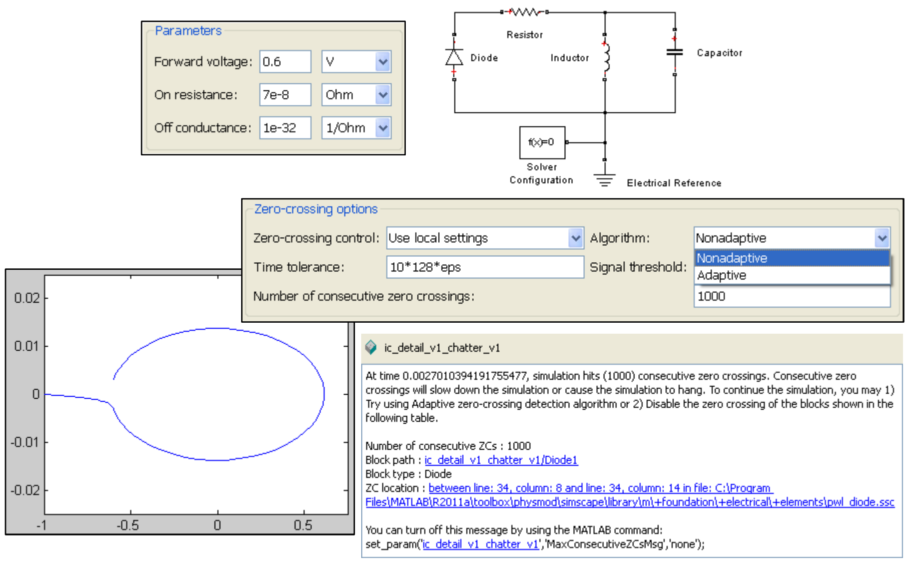
Zero-crossing location ON
>> tic;sim(gcs,'reltol','1e-4','Solver','ode113','stoptime','0.5');toc
Elapsed time is **4.155692** seconds.

Zero-crossing location OFF
>> tic;sim(gcs,'reltol','1e-4','Solver','ode113','stoptime','0.5');toc
Elapsed time is **3.947801** seconds.

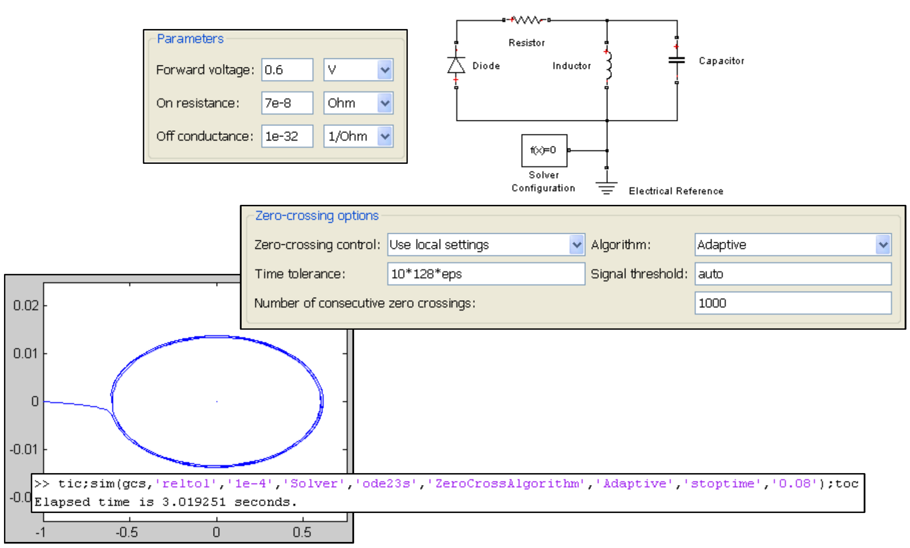
Is computational complexity the only zero crossing issue?



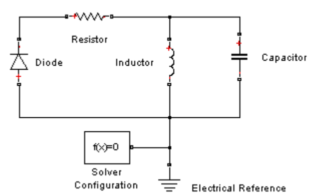
Chattering in our electrical circuit

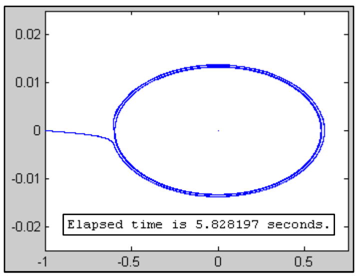


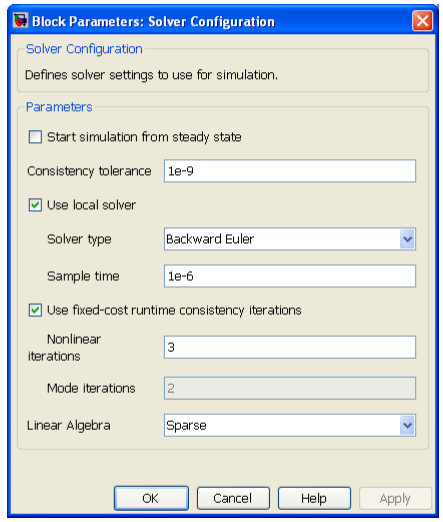
Resolve the chattering by adaptive zero crossings



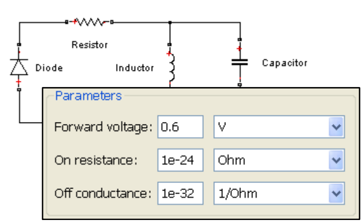
Or, employ a Simscape fixed-step solver with fixed cost

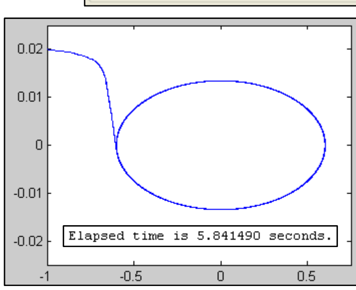


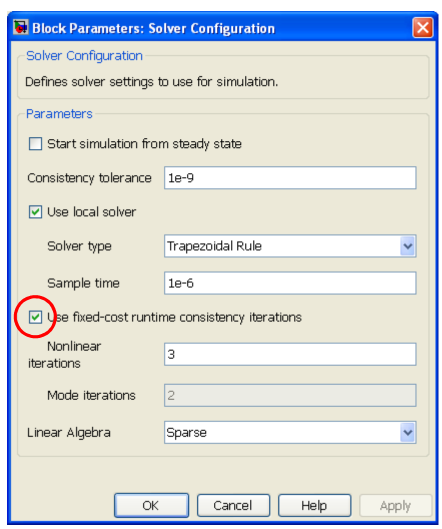




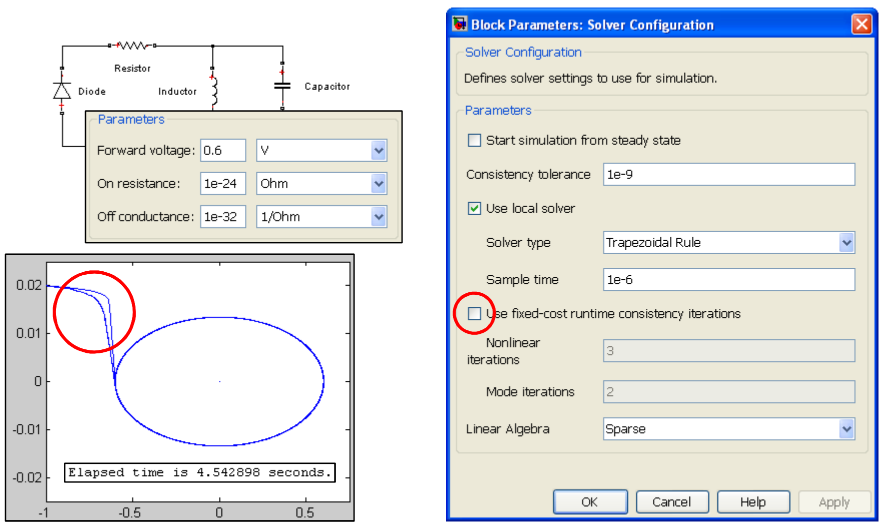
But a fixed-cost solver may be less accurate







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

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
    - 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
    - Choose the fundamental sample time as the step size

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

MathWorks®

Accelerating the pace of engineering and science