

Mat 3440 Adaptive timestepping for ODEs

Håkon Hoel

Spring 2026

Adaptive timestepping based on local errors

For an IVP $X' = F(X, t)$ given X_0 and solved numerically over $[0, T]$, we have that

$$|X(T) - \bar{X}(T)| \leq C_{\text{stability}} \sum_{n=0}^{N-1} \underbrace{|X(t_{n+1}; X(t_n) = \bar{X}_n) - \bar{X}_{n+1}|}_{=\tau(\bar{X}_n, t_n)}.$$

The local error can also be represented

$$\tau(\bar{X}_n, t_n) = \int_{t_n}^{t_{n+1}} F(X(s; X(t_n) = \bar{X}_n) - F(\bar{X}_n) ds$$

- **Question:** If we know the size of $|\tau(\bar{X}_n, t_n)|$, can we adjust $h_n = h(\bar{X}_n, t_n)$ adaptively to produce a more accurate/efficient numerical method?
- **Important real-life problem:** Normally do not have access to neither the exact solution nor τ itself, so we would have to estimate τ .

Numerical estimation of the local error

Suppose we have two numerical Φ and Φ^* , which respectively produce the solutions

$$\bar{X}_{n+1} = \bar{X}_n + h_n \Phi(\bar{X}_n, t_n, h_n), \quad \text{and} \quad \bar{X}_{n+1}^* = \bar{X}_n^* + h_n \Phi^*(\bar{X}_n^*, t_n, h_n),$$

where h_n is an adaptive time step, $t_{n+1} = t_n + h_n$ and Φ^* is more accurate – a higher order method – than Φ .

1 Let us “view” Φ^* as the exact solver, and let

$$\bar{X}^*(t_{n+1}; \bar{X}^*(t_n) = \bar{X}_n) := \bar{X}_n + h_n \Phi^*(\bar{X}_n, t_n, h_n)$$

2 Approximate the local error by

$$\bar{\tau}(\bar{X}_n, t_n, h_n) := \bar{X}^*(t_{n+1}; \bar{X}^*(t_n) = \bar{X}_n) - \bar{X}_{n+1}$$

3 By the formulas of the schemes,

$$|\bar{\tau}(\bar{X}_n, t_n, h_n)| = |\Phi^*(\bar{X}_n, t_n, h_n) - \Phi(\bar{X}_n, t_n, h_n)| h_n$$

4 Approximation of error (up to a constant):

$$|X(T) - \bar{X}(T)| \approx \sum_{n=0}^{N-1} |\Phi^*(\bar{X}_n, t_n, h_n) - \Phi(\bar{X}_n, t_n, h_n)| h_n = \sum_{n=0}^{N-1} |\bar{\tau}(\bar{X}_n, t_n, h_n)|$$

5 Adaptivity strategy: Given input tolerance $\epsilon > 0$, then

For $n = 0, \dots, N - 1$:

$$\text{if } |\bar{\tau}(\bar{X}_n, t_n, h_n)| > \epsilon/N$$

then halve timestep $h_n = h_n/2$.

6 Recompute numerical solution and local errors and iterate step 5 again until $\max_n |\bar{\tau}(\bar{X}_n, t_n, h_n)| \leq \epsilon/N$.

7 Output: the fine solution $\{\bar{X}_n^*\}_{n=0}^N$ (this not consistent with method, which adapts the mesh for the coarse solution, but it is reasonable to assume the fine solution is even more accurate).

ODE23 – embedded RK methods

Matlab's ODE23 is an adaptive-timestepping explicit Runge-Kutta 2(3) method given by

0		0		
1		1	0	
1/2		1/4	1/4	0
		1/2	1/2	0
		1/6	1/6	2/3

where

$$\Phi = \begin{array}{c|ccc} 0 & 0 & & \\ 1 & 1 & 0 & \\ \hline 1/2 & 1/4 & 1/4 & 0 \\ & 1/2 & 1/2 & 0 \end{array} \quad \text{and} \quad \Phi^* = \begin{array}{c|ccc} 0 & 0 & & \\ 1 & 1 & 0 & \\ \hline 1/2 & 1/4 & 1/4 & 0 \\ & 1/6 & 1/6 & 2/3 \end{array}$$

Here Φ has order 2 (here k_3 is not needed/used) and Φ^* has order 3.

Efficiency benefit: Both methods share the same stages k_1 and k_2 .

ODE45 – Dormand–Prince

Popular adaptive-timestepping explicit Runge-Kutta 4(5) method. The order 5 method Φ^* with b^* uses 6 stages and Φ with b uses 7 stages.

0							
$\frac{1}{5}$	$\frac{1}{5}$						
$\frac{3}{10}$	$\frac{3}{40}$	$\frac{9}{40}$					
$\frac{4}{5}$	$\frac{44}{45}$	$-\frac{56}{15}$	$\frac{32}{9}$				
$\frac{8}{9}$	$\frac{19372}{6561}$	$-\frac{25360}{2187}$	$\frac{64448}{6561}$	$-\frac{212}{729}$			
1	$\frac{9017}{3168}$	$-\frac{355}{33}$	$\frac{46732}{5247}$	$\frac{49}{176}$	$-\frac{5103}{18656}$		
1	$\frac{35}{384}$	0	$\frac{500}{1113}$	$\frac{125}{192}$	$-\frac{2187}{6784}$	$\frac{11}{84}$	
b^* (5th order)	$\frac{35}{384}$	0	$\frac{500}{1113}$	$\frac{125}{192}$	$-\frac{2187}{6784}$	$\frac{11}{84}$	0
b (4th order)	$\frac{5179}{57600}$	0	$\frac{7571}{16695}$	$\frac{393}{640}$	$-\frac{92097}{339200}$	$\frac{187}{2100}$	$\frac{1}{40}$

Adaptive timestepping

For ill-behaved nonlinear problems, adaptive timestepping can improve efficiency.

Consider the Van der Pol oscillator

$$\left. \begin{aligned} x_1' &= x_2 \\ x_2' &= \mu(1 - x_1^2)x_2 - x_1 \end{aligned} \right\} \quad t \in [0, 100] \quad \text{with} \quad x(0) = (2, 0)$$

for $\mu > 0$. This is an oscillator with nonlinear damping/amplification (a limit-cycle thermostat):

$$x_1'' = \underbrace{\mu(1 - x_1^2)x_1'}_{\text{nonlinear "damping"}} - \underbrace{x_1}_{\text{oscillation}}$$

- ode45 works well for μ close to 1 (stable problem)
- ode23 works well for μ around 10
- for $\mu \gg 10$, use an implicit solver, like ode15s

Using ode23 or ode45

```
mu = 1
options = odeset('RelTol', 1e-10, 'AbsTol', 1e-11, 'Stats', 'on');
F = @(t,x) [x(2); mu*(1-x(1)^2)*x(2)-x(1)];
X0 = [2;0];
t0 = 0, T = 100 % time interval [t0,T]
[t,X] = ode23(F,[t0 T],x0,options); %numerical integrator
%output: [t,X] adaptive mesh t and solution X(t_n) for n = 0,...,N

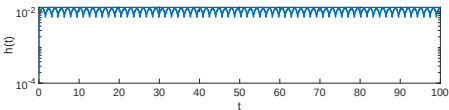
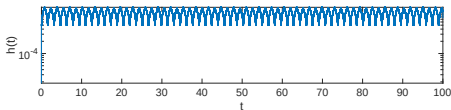
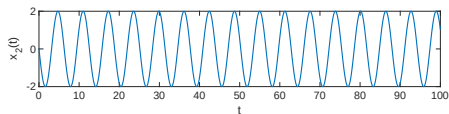
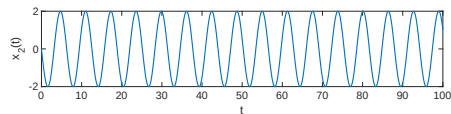
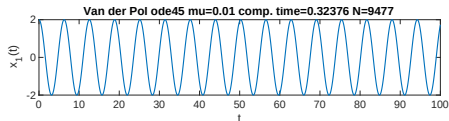
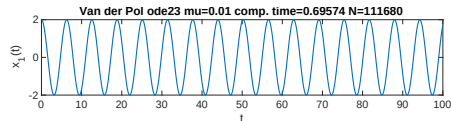
%Alternative call [t,X] = ode45(F,[t0 T],x0,options);

% Then plot results
```

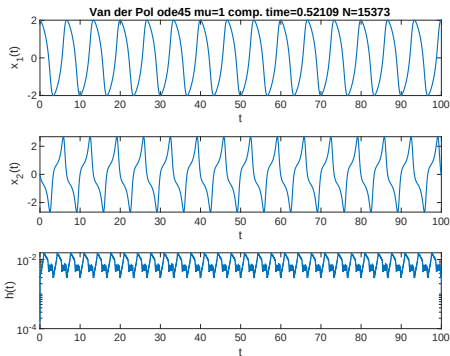
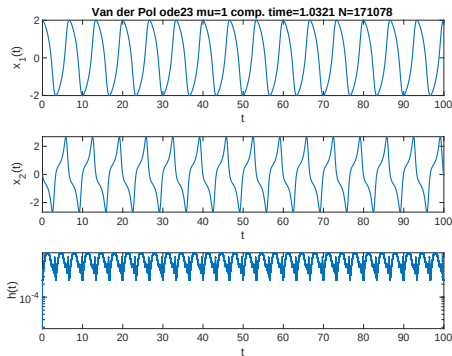
Numerical tests Van der Pol $\mu = 0.01$

Matlab's variable-stepsize integrators ode23 and ode45. Note that "options" in code above aims to achieve

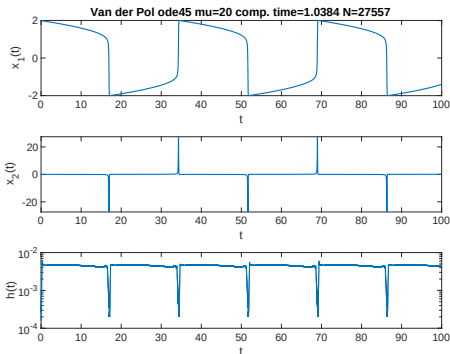
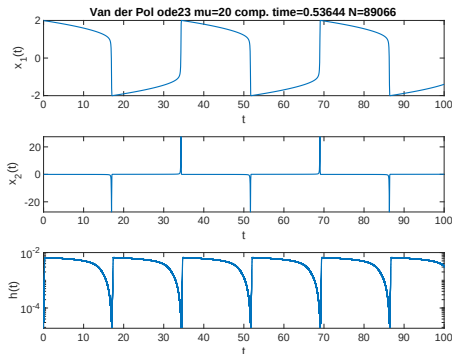
$$|X_i(t_n) - \bar{X}_i(t_n)| \leq \text{RelTol}|X_i(t_n)| + \text{AbsTol}.$$



Numerical tests Van der Pol $\mu = 1$

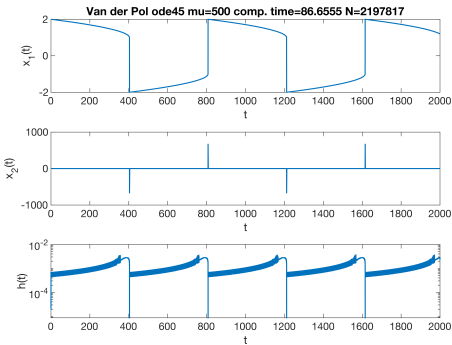
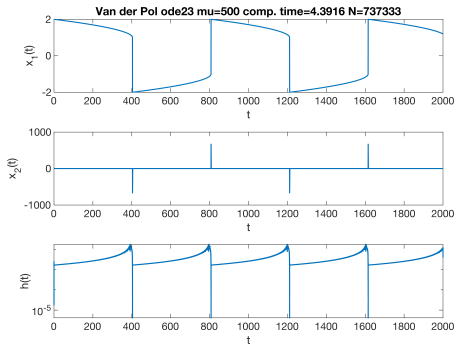


Numerical tests Van der Pol $\mu = 20$



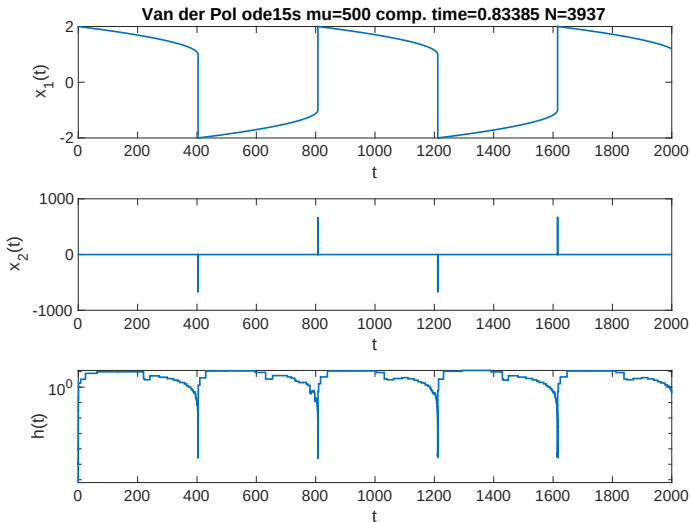
Numerical tests Van der Pol $\mu = 500$

For $\mu = 500$ ode23 and ode45 are inefficient:



Numerical tests Van der Pol $\mu = 500$

A better alternative is ode15s (called the same way):



List of workhorse adaptive ODE solvers in MATLAB

Solver	Type	Order	Best Use Case
ode45	Explicit	Fixed (5th)	General purpose; smooth.
ode23	Explicit	Fixed (3rd)	Crude tolerances; lower regularity.
ode113	Explicit	Adaptive (1–13)	Expensive functions; precision.
ode15s	Implicit	Adaptive (1–5)	Stiff systems.

ode15s is a multistep method (not Runge-Kutta) using up to 5 steps:

$$\bar{X}_n, \bar{X}_{n-1}, \dots, \bar{X}_{n-4} \text{ to compute } \bar{X}_{n+1}.$$