Numerical PDEs:
Numerical Methods for ODEs

Di Qi

Purdue University [1]
qidi@purdue.edu
[1] MA/CS 615, Spring 2026

Outline

(1) Initial Value Problems
(2) One-step methods for ODEs
(3) Convergence (after LeVeque)
(4) MATLAB ode suite

Initial Value Problems

- We want to numerically approximate the solution to a system of ordinary differential equations

$$\frac{d\mathbf{x}}{dt} = \mathbf{x}'(t) = \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), t),$$

with initial condition $\mathbf{x}(t = 0) = \mathbf{x}(0) = \mathbf{x}_0$.

- This means that we want to generate an approximation to the trajectory $\mathbf{x}(t)$, for example, a sequence $\mathbf{x}(t_k = k\Delta t)$ for $k = 1, 2, \ldots, N = T/\Delta t$, where Δt is the time step used to discretize time.
- If $\mathbf{f}(\mathbf{x})$ is independent of t we call the system autonomous.
- Note that second-order equations can be written as a system of first-order equations:

$$\frac{d^2\mathbf{x}}{dt^2} = \ddot{\mathbf{x}}(t) = \mathbf{f}[\mathbf{x}(t), t] \equiv \begin{cases} \dot{\mathbf{x}}(t) = \mathbf{v}(t) \\ \dot{\mathbf{v}}(t) = \mathbf{f}[\mathbf{x}(t), t] \end{cases}$$

Relation to Numerical Integration

- If $\mathbf{f}$ is independent of $\mathbf{x}$ then the problem is equivalent to numerical integration

$$\mathbf{x}(t) = \mathbf{x}_0 + \int_0^t \mathbf{f}(s)ds$$

- More generally, we cannot compute the integral because it depends on the unknown answer $\mathbf{x}(t)$:

$$\mathbf{x}(t) = \mathbf{x}_0 + \int_0^t \mathbf{f}(\mathbf{x}(s), s)ds$$

- Numerical methods are based on approximations of $\mathbf{f}(\mathbf{x}(s), s)$ into the "future" based on knowledge of $\mathbf{x}(t)$ in the "past" and "present".

Convergence

- Consider a trajectory numerically discretized as a sequence that approximates the exact solution at a discrete set of points:

$$\mathbf{x}^{(k)} \approx \mathbf{x}\left(t_k = k\Delta t\right), \quad k = 1, \ldots, T/\Delta t.$$

- A method is said to converge with order $p > 0$, or to have order of accuracy p, if for any finite T for which the ODE has a solution,

$$\left|\mathbf{x}^{(k)} - \mathbf{x}(k\Delta t)\right| = O\left(\Delta t^p\right) \text{ for all } 0 \leq k \leq T/\Delta t.$$

- All methods are recursions that compute a new $\mathbf{x}^{(k+1)}$ from previous $\mathbf{x}^{(k)}$ by evaluating $\mathbf{f}$ several times.
  E.g., one-step methods have the form

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \Delta t \boldsymbol{\Psi}\left(\mathbf{x}^{(k)}, t = k\Delta t, \Delta t; \mathbf{f}\right) \tag{1}$$

Consistency

- The local trunction error (LTE) tells us how much the exact solution does not satisfy the numerical scheme at the end of the time step if started from the correct solution $\mathbf{x}^{(k)} = \mathbf{x}(k\Delta t)$:

$$\mathbf{e}_k = \frac{\mathbf{x}[(k+1)\Delta t] - \mathbf{x}[k\Delta t]}{\Delta t} - \mathbf{\Psi}[\mathbf{x}(t), \Delta t; \mathbf{f}],$$

  i.e., the error you get when you plug the exact solution into (1).

- Note that the error in $\mathbf{x}^{(k+1)}$ is $\tilde{\mathbf{e}}_k = \mathbf{e}_k \Delta t$; this is an alternative definition of LTE.

- A method is consistent with order $q \geq 1$ if $|\mathbf{e}_k| = O\left(\Delta t^q\right)$.

- The global truncation error is the actual error

$$E^{(k)} = \left|\mathbf{x}^{(k)} - \mathbf{x}(t = k\Delta t)\right|.$$

Propagation of errors

- Can the global error be bounded by $O\left(\Delta t^p\right)$ if the local one is $O\left(\Delta t^q\right)$?

- Crude estimate: If one makes an error $O\left(\Delta t^{q+1}\right)$ at each time step, the global error after $T/\Delta t$ time steps can become on the order of

$$\left|\mathbf{x}^{(k)} - \mathbf{x}(k\Delta t)\right| = O\left(\Delta t^{q+1} \cdot \frac{T}{\Delta t}\right) = O\left(\Delta t^q\right) = O\left(\Delta t^p\right),$$

  and we must have $p = q \geq 1$ for convergence.

- This result is often the right one, but it has a hidden assumption that errors made at previous steps do not grow but rather stay of the same order so they can be added.

- In practice, errors made in previous time steps will either grow or shrink with time. If they grow "too fast" we are in trouble.

- So we arrive for the first time at a recurring theme: Convergence requires stability in addition to consistency. What does stability mean?

One-step methods for ODEs

Euler's Method

- Assume that we have our approximation $\mathbf{x}^{(k)}$ and want to move by one time step:

$$\mathbf{x}^{(k+1)} \approx \mathbf{x}^{(k)} + \int_{k\Delta t}^{(k+1)\Delta t} \mathbf{f}(\mathbf{x}(s), s) ds$$

- The simplest possible thing is to use a piecewise constant approximation:

$$\mathbf{f}(\mathbf{x}(s), s) \approx \mathbf{f}\left(\mathbf{x}^{(k)}, t^{(k)}\right) = \mathbf{f}^{(k)},$$

which gives the forward Euler method

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \mathbf{f}^{(k)} \Delta t.$$

- This method requires only one function evaluation per time step.

Euler's Method

- The local trunction error is easy to find using a Taylor series expansion:

$$\begin{aligned} \mathbf{e}_k &= \{[\mathbf{x}(k+1)\Delta t] - \mathbf{x}(k\Delta t)\}/\Delta t - \mathbf{f}[\mathbf{x}(k\Delta t)] = \\ &= \{\mathbf{x}[(k+1)\Delta t] - \mathbf{x}(k\Delta t)\}/\Delta t - \mathbf{x}'(k\Delta t) = \frac{\mathbf{x}''(\xi)}{2}\Delta t, \end{aligned}$$

for some $k\Delta t \leq \xi \leq (k+1)\Delta t$.

- Therefore the LTE is $O(\Delta t), q = 1$.
- The global truncation error, is expected to be of order $O(\Delta t)$ (we will prove this shortly), $p = 1$, so this is a first-order accurate method.

Backward Euler

$$\mathbf{x}^{(k+1)} \approx \mathbf{x}^{(k)} + \int_{k\Delta t}^{(k+1)\Delta t} \mathbf{f}(\mathbf{x}(s), s) ds$$

- How about we use a piecewise constant-approximation, but based on the end-point:

$$\mathbf{f}(\mathbf{x}(s), s) \approx \mathbf{f}\left(\mathbf{x}^{(k+1)}, t^{(k+1)}\right) = \mathbf{f}^{(k+1)}$$

which gives the first-order backward Euler method

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \mathbf{f}^{(k+1)}\Delta t = \mathbf{x}^{(k)} + \mathbf{f}\left(\mathbf{x}^{(k+1)}, t^{(k+1)}\right)\Delta t.$$

- This implicit method requires solving a non-linear equation at every time step, which is expensive and hard. We will understand why implicit methods are needed next class.

Runge-Kutta Methods

- Runge-Kutta methods are a powerful class of one-step methods similar to Euler's method, but more accurate.
- As an example, consider using a trapezoidal rule to approximate the integral

$$\mathbf{f}^{(k)} + \int_{k\Delta t}^{(k+1)\Delta t} \mathbf{f}[\mathbf{x}(s), s] ds \approx \mathbf{x}^{(k)} + \frac{\Delta t}{2}\left[\mathbf{f}^{(k)} + \mathbf{f}^{(k+1)}\right]$$
$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \frac{\Delta t}{2}\left[\mathbf{f}\left(\mathbf{x}^{(k)}, t^{(k)}\right) + \mathbf{f}\left(\mathbf{x}^{(k+1)}, t^{(k+1)}\right)\right]$$

which requires solving a nonlinear equation for $x^{(k+1)}$.

- This is the simplest implicit Runge-Kutta method, usually called the implicit trapezoidal method.
- The local truncation error is $O\left(\Delta t^3\right)$, so the global error is second-order accurate $O\left(\Delta t^2\right)$.

LTE: θ-method

$\mathbf{y}'(t) = \mathbf{f}(t, \mathbf{y}(t))$. Scheme $\mathbf{y}_{n+1} = \mathbf{y}_n + h\left[\theta \mathbf{f}\left(t_n, \mathbf{y}_n\right) + (1-\theta)\mathbf{f}\left(t_{n+1}, \mathbf{y}_{n+}\right.\right.$ $\theta = 1$ is Forward Euler, $\theta = 0$ is Backward Euler, $\theta = 1/2$ is Implicit Trapezoidal

$$
\begin{aligned}
&\boldsymbol{y}\left(t_{n+1}\right) - \boldsymbol{y}\left(t_n\right) - h\left[\theta \boldsymbol{f}\left(t_n, \boldsymbol{y}\left(t_n\right)\right) + (1-\theta)\boldsymbol{f}\left(t_{n+1}, \boldsymbol{y}\left(t_{n+1}\right)\right)\right] \\
=&\boldsymbol{y}\left(t_{n+1}\right) - \boldsymbol{y}\left(t_n\right) - h\left[\theta \boldsymbol{y}'\left(t_n\right) + (1-\theta)\boldsymbol{y}'\left(t_{n+1}\right)\right] \\
=&\left[\boldsymbol{y}\left(t_n\right) + h\boldsymbol{y}'\left(t_n\right) + \frac{1}{2}h^2\boldsymbol{y}''\left(t_n\right) + \frac{1}{6}h^3\boldsymbol{y}'''\left(t_n\right)\right] - \boldsymbol{y}\left(t_n\right) \\
&- h\left\{\theta \boldsymbol{y}'\left(t_n\right) + (1-\theta)\left[\boldsymbol{y}'\left(t_n\right) + h\boldsymbol{y}''\left(t_n\right) + \frac{1}{2}h^2\boldsymbol{y}'''\left(t_n\right)\right]\right\} + \mathcal{O}\left(h^4\right) \\
=&\left(\theta - \frac{1}{2}\right)h^2\boldsymbol{y}''\left(t_n\right) + \left(\frac{1}{2}\theta - \frac{1}{3}\right)h^3\boldsymbol{y}'''\left(t_n\right) + \mathcal{O}\left(h^4\right).
\end{aligned}
$$

Midpoint/Trapezoidal Methods

- Schemes that treat beginning and end of time step in a symmetric fashion will lead to a cancellation of first-order error terms in Taylor series and will thus be second order (Lesson: second order is easy).
- In addition to trapezoidal one can do implicit midpoint scheme:

$$
\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \Delta t \mathbf{f}\left(\frac{\mathbf{x}^{(k)} + \mathbf{x}^{(k+1)}}{2}, t^{(k)} + \frac{\Delta t}{2}\right)
$$

  Observe this is the same as trapezoidal for linear problems (why?).
- In an explicit method, we can approximate $\mathbf{x}^{(k+1)} \approx \mathbf{x}^{(k+1,\star)}$ to first order only (why?), say using Euler's method.

Explicit Midpoint

- This gives an explicit Runge-Kutta method, usually called Heun's or explicit trapezoidal method

$$
\begin{aligned}
\mathbf{x}^{(k+1,\star)} &= \mathbf{x}^{(k)} + \mathbf{f}\left(\mathbf{x}^{(k)}, t^{(k)}\right)\Delta t \\
\mathbf{x}^{(k+1)} &= \mathbf{x}^{(k)} + \frac{\Delta t}{2}\left[\mathbf{f}\left(\mathbf{x}^{(k)}, t^{(k)}\right) + \mathbf{f}\left(\mathbf{x}^{(k+1,\star)}, t^{(k+1)}\right)\right].
\end{aligned}
$$

- Explicit midpoint rule

$$\mathbf{x}^{\left(k+\frac{1}{2},\star\right)} = \mathbf{x}^{(k)} + \mathbf{f}\left(\mathbf{x}^{(k)}, t^{(k)}\right)\frac{\Delta t}{2}$$
$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \Delta t\mathbf{f}\left(\mathbf{x}^{\left(k+\frac{1}{2},\star\right)}, t^{(k)} + \frac{\Delta t}{2}\right).$$

- Explicit midpoint/trapezoidal are a representative of a powerful class of second-order methods called predictor-corrector methods: Euler (forward or backward) method is the predictor, and then (implicit or explicit) trapezoidal/midpoint method is the corrector.
(Belongs to multi-stage one-step methods.)

LTE: explicit midpoint

$$\tau^n = \frac{1}{k}\left(u\left(t_{n+1}\right) - u\left(t_n\right)\right) - f\left(u\left(t_n\right) + \frac{1}{2}kf\left(u\left(t_n\right)\right)\right). \tag{5.31}$$

Note that

$$f\left(u\left(t_n\right) + \frac{1}{2}kf\left(u\left(t_n\right)\right)\right) = f\left(u\left(t_n\right) + \frac{1}{2}ku'\left(t_n\right)\right)$$
$$= f\left(u\left(t_n\right)\right) + \frac{1}{2}ku'\left(t_n\right)f'\left(u\left(t_n\right)\right) + \frac{1}{8}k^2\left(u'\left(t_n\right)\right)^2 f''\left(u\left(t_n\right)\right) + \cdots$$

Since $f\left(u\left(t_n\right)\right) = u'\left(t_n\right)$ and differentiating gives $f'(u)u' = u''$, we obtain

$$f\left(u\left(t_n\right) + \frac{1}{2}kf\left(u\left(t_n\right)\right)\right) = u'\left(t_n\right) + \frac{1}{2}ku''\left(t_n\right) + O\left(k^2\right).$$

Using this in (5.31) gives

$$\tau^n = \frac{1}{k}\left(\, ku'\left(t_n\right) + \frac{1}{2}k^2u''\left(t_n\right) + O\left(k^3\right)\right)$$
$$- \left(u'\left(t_n\right) + \frac{1}{2}ku''\left(t_n\right) + O\left(k^2\right)\right)$$

Higher-Order Runge-Kutta Methods

- The idea in RK methods is to evaluate the function $f(x,t)$ several times and then take a time-step based on an average of the values.
- In practice, this is done by performing the calculation in stages: Calculate an intermediate approximation $x^\star$, evaluate $f\left(x^\star\right)$, and go to the next stage.
- The most celebrated Runge-Kutta methods is a four-stage fourth-order accurate RK4 method based on Simpson's rule for the integral: $x^{(k)} + \int_{k\Delta t}^{(k+1)\Delta t} f[x(s),s]ds$

$$\begin{aligned}
&\approx x^{(k)} + \frac{\Delta t}{6}\left[f\left(x^{(k)}\right) + 4f\left(x^{(k+1/2)}\right) + f\left(x^{(k+1)}\right)\right] \\
&= x^{(k)} + \frac{\Delta t}{6}\left[f^{(k)} + 4f^{(k+1/2)} + f^{(k+1)}\right],
\end{aligned}$$

and we approximate $4f^{(k+1/2)} = 2f^{(k+1/2;1)} + 2f^{(k+1/2;2)}$.

RK4 Method

$$\begin{aligned}
f^{(k)} = f\left(x^{(k)}\right), \quad x^{(k+1/2;1)}, &= x^{(k)} + \frac{\Delta t}{2} f^{(k)} \\
f^{(k+1/2;1)} &= f\left(x^{(k+1/2;1)}, t^{(k)} + \Delta t/2\right) \\
x^{(k+1/2;2)} &= x^{(k)} + \frac{\Delta t}{2} f^{(k+1/2;1)} \\
f^{(k+1/2;2)} &= f\left(x^{(k+1/2;2)}, t^{(k)} + \Delta t/2\right) \\
x^{(k+1;1)} &= x^{(k)} + \Delta t f^{(k+1/2;2)} \\
f^{(k+1)} &= f\left(x^{(k+1;1)}, t^{(k)} + \Delta t\right) \\
x^{(k+1)} = x^{(k)} + \frac{\Delta t}{6}&\left[f^{(k)} + 2f^{(k+1/2;1)} + 2f^{(k+1/2;2)} + f^{(k+1)}\right]
\end{aligned}$$

Intro to multistep Methods

$$\mathbf{x}^{(k+1)} \approx \mathbf{x}^{(k)} + \int_{k\Delta t}^{(k+1)\Delta t} \mathbf{f}[\mathbf{x}(s),s]ds$$

- Euler's method was based on a piecewise constant approximation (extrapolation) of $\mathbf{f}(s) \equiv \mathbf{f}[\mathbf{x}(s), s]$.
- If we instead integrate the linear extrapolation

$$f(s) \approx \mathbf{f}\left(\mathbf{x}^{(k)}, t^{(k)}\right) + \frac{\mathbf{f}\left(\mathbf{x}^{(k)}, t^{(k)}\right) - \mathbf{f}\left(\mathbf{x}^{(k-1)}, t^{(k-1)}\right)}{\Delta t}\left(s - t_k\right),$$

we get the second-order two-step Adams-Bashforth method

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \frac{\Delta t}{2}\left[3\mathbf{f}\left(\mathbf{x}^{(k)}, t^{(k)}\right) - \mathbf{f}\left(\mathbf{x}^{(k-1)}, t^{(k-1)}\right)\right].$$

- This is an example of a multi-step method, which requires keeping memory of previous values of $\mathbf{f}$.

Convergence

Zero Stability

- We must also examine how perturbations grow with time: error propagation.
- A method is called zero stable if for all sufficiently small but finite Δt, introducing perturbations at each step (e.g., roundoff errors, errors in evaluating f) with magnitude less than some small ϵ perturbs the solution by at most $O(\epsilon)$.
- This simply means that errors do not increase but rather decrease from step to step, as we saw with roundoff errors in the first homework.
- A central theorem in numerical methods for differential equations is variants of the Lax equivalence theorem:
  Any consistent method is convergent if and only if it is zero stable, or consistency + (zero) stability = convergence.
- We will show now that one-step methods are zero-stable if f is well-behaved (Lipschitz continuous w.r.t. second argument).

Lipschitz Constants

- Let us consider a system of ODEs

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}, t), \quad \mathbf{x}(t=0) = \mathbf{x_0}.$$

- Standard theory for ODEs shows that the solution exists and is unique over some finite time interval if the r.h.s. is Lipschitz continuous in $\mathbf{x}$ a neighborhood of the initial condition:

$$\|\mathbf{f}(\mathbf{x}, t) - \mathbf{f}(\mathbf{x}^\star, t)\| \leq L \|\mathbf{x} - \mathbf{x}^\star\|,$$

for all $\{\mathbf{x}, \mathbf{x}^\star\}$ within some neighborhood of $\mathbf{x}_0$ over some finite interval $t \geq 0$.

- For differentiable functions we can take

$$L = \max \left\| \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \right\|.$$

Convergence

- Denote our numerical approximation with time step size τ :

$$\mathbf{x}^{(k)} \approx \mathbf{x}(k\Delta t).$$

- A method is convergent if applying it to any system of ODEs where $\mathbf{f}$ is Lipshitz over a finite time interval $T > 0$ during which the ODE has a solution, the numerical approximation converges to that solution,

$$\lim_{\Delta t \to 0} \mathbf{x}^{(N=T/\Delta t)} = \mathbf{x}(T).$$

- Convergence is a statement about a limit, and does not imply a method will give reasonable answers for finite $\Delta t > 0$.
  For that we will later introduce absolute stability.

- Note that we haven't given a precise definition to zero stability because in some sense it is defined as: the extra conditions that are needed to make a consistent method convergent.
  For multistep methods, covered later, we will figure out an explicit condition.

Convergence of One Step Methods

- Let us prove that all consistent one-step methods are convergent (i.e., they are zero stable).

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \Delta t \boldsymbol{\Psi}\left(\mathbf{x}^{(k)}, t = k\Delta t, \Delta t; \mathbf{f}\right)$$

- The method is consistent and we assume that $\boldsymbol{\Psi}$ is continuous in t and Δt, and Lipshitz in $\mathbf{x}$ with some constant $\widetilde{L}$,

$$\boldsymbol{\Psi}(\mathbf{x}(t), t, \Delta t \to 0; \mathbf{f}) = \mathbf{f}(\mathbf{x}, t),$$

- For example, for explicit midpoint rule,

$$\boldsymbol{\Psi}(\mathbf{x}, t, \Delta t; \mathbf{f}) = \mathbf{f}\left(\mathbf{x} + \frac{\Delta t}{2}\mathbf{f}(\mathbf{x}, t), t + \frac{\Delta t}{2}\right) \underset{\Delta t \to 0}{\to} \mathbf{f}(\mathbf{x}, t).$$

Convergence of One Step Methods

- Now use the Lipshitz continuity of $\mathbf{f}$ to bound the Lipshitz constant for $\boldsymbol{\Psi}$:

$$\begin{aligned}
&\|\boldsymbol{\Psi}(\mathbf{x}, t, \Delta t) - \boldsymbol{\Psi}(\mathbf{x}^\star, t, \Delta t)\| \\
&\leq L \left\| \left(\mathbf{x} + \frac{\Delta t}{2}\mathbf{f}(\mathbf{x}, t)\right) - \left(\mathbf{x}^\star + \frac{\Delta t}{2}\mathbf{f}(\mathbf{x}^\star, t)\right) \right\| \\
&\leq L \|\mathbf{x} - \mathbf{x}^\star\| + \frac{L\Delta t}{2} \|\mathbf{f}(\mathbf{x}, t) - \mathbf{f}(\mathbf{x}^\star, t)\| \\
&\leq \left(1 + \frac{L\Delta t}{2}\right) L \|\mathbf{x} - \mathbf{x}^\star\| = \tilde{L} \|\mathbf{x} - \mathbf{x}^\star\|
\end{aligned}$$

- Therefore for explicit midpoint, $\tilde{L} = (1 + L\Delta t/2)L$. More generally, $\tilde{L}$ is L times some rational function of Δt that goes to 1 as $\Delta t \to 0$.

Error growth factor

- From the definition of the LTE

$$\mathbf{e}^{(k)} = \frac{\mathbf{x}((k+1)\Delta t) - \mathbf{x}(k\Delta t)}{\Delta t} - \mathbf{\Psi}(\mathbf{x}(k\Delta t), k\Delta t, \Delta t),$$

we get

$$\mathbf{x}((k+1)\Delta t) = \mathbf{x}(k\Delta t) + \Delta t \mathbf{\Psi}(\mathbf{x}(k\Delta t), k\Delta t, \Delta t) + \Delta t \mathbf{e}^{(k)},$$

while our scheme is

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \Delta t \mathbf{\Psi}\left(\mathbf{x}^{(k)}, t = k\Delta t, \Delta t\right)$$

- Subtracting the two we get the error recursion relation for the global error $\mathrm{E}^{(k)} = \mathrm{x}(k\Delta t) - \mathrm{x}^{(k)}$,

$$\mathbf{E}^{(k+1)} = \mathbf{E}^{(k)} + \Delta t \left(\mathbf{\Psi}(\mathbf{x}(k\Delta t), \ldots) - \mathbf{\Psi}\left(\mathbf{x}^{(k)}, \ldots\right)\right) + \Delta t \mathbf{e}^{(k)}$$
$$\left\|\mathbf{E}^{(k+1)}\right\| \leq \left\|\mathbf{E}^{(k)}\right\| + \Delta t \tilde{L} \left\|\mathbf{E}^{(k)}\right\| + \Delta t \left\|\mathbf{e}^{(k)}\right\|$$

Convergence

- This is the typical relationship we will see many times

$$\left\|\mathbf{E}^{(k+1)}\right\| \leq (1 + \Delta t \tilde{L}) \left\|\mathbf{E}^{(k)}\right\| + \Delta t \left\|\mathbf{e}^{(k)}\right\|$$

- Quite generally, we get recursions of the form:
  global_error (k + 1) <=
  amplification_factor*global_error(k)+local_error(k)

- The recurrence relationship for the error has the explicit solution

$$\left\|\mathbf{E}^{(k)}\right\| \leq (1 + \Delta t \widetilde{L})^k \left\|\mathbf{E}^{(0)}\right\| + \Delta t \sum_{m=1}^{k} (1 + \Delta t \widetilde{L})^{k-m} \left\|\mathbf{e}^{(m-1)}\right\|.$$

- We can take $\left\|\mathbf{E}^{(0)}\right\| = \mathbf{0}$ if we know the initial condition "exactly", leaving us to bound

$$(1 + \Delta t\tilde{L})^k$$

Error bound

- Very generally we will bound error growth factors by exponentials (here simple scalars but more generally matrix powers and matrix exponentials):

$$(1 + \Delta t\tilde{L}) \leq e^{\Delta t\tilde{L}} \Rightarrow (1 + \Delta t\tilde{L})^k \leq e^{k\Delta t\tilde{L}} \leq e^{T\tilde{L}}.$$

$$\left\|\mathbf{E}^{(k)}\right\| \leq \Delta t \sum_{m=1}^{k} (1 + \Delta t\tilde{L})^{k-m} \left\|\mathbf{e}^{(m-1)}\right\| \leq \Delta t e^{T\tilde{L}} \left(\sum_{m=1}^{k} \left\|\mathbf{e}^{(m-1)}\right\|\right)$$

$$\left\|\mathbf{E}^{(k)}\right\| \leq T e^{T\tilde{L}} \max_{1 \leq m < k} \left\|\mathbf{e}^{(m-1)}\right\|$$

- This now proves that if the local error (defined in LeVeque's way!) is of $O\left(\Delta t^p\right)$ then so is the global error.

- The factor $Te^{T\tilde{L}}$ is a constant for the purpose of zero stability as we are taking the limit $\Delta t \to 0$, but in practice it is extremely important as it controls how small Δt has to be for the method to be useful...

MATLAB ode suite

In MATLAB

- In MATLAB, there are several functions whose names begin with

$$[\mathbf{t}, \mathbf{x}] = \text{ode}\left(f, [t_0, t_e], x_0, \text{odeset}(\ldots)\right).$$

- ode 23 is a second-order adaptive explicit Runge-Kutta method, while ode45 is a fourth-order version (try it first).

- ode23tb is a second-order implicit RK method.

- ode113 is a variable-order explicit multi-step method that can provide very high accuracy.
- ode15s is a variable-order implicit multi-step method.
- For implicit methods the Jacobian can be provided using the odeset routine - very important!

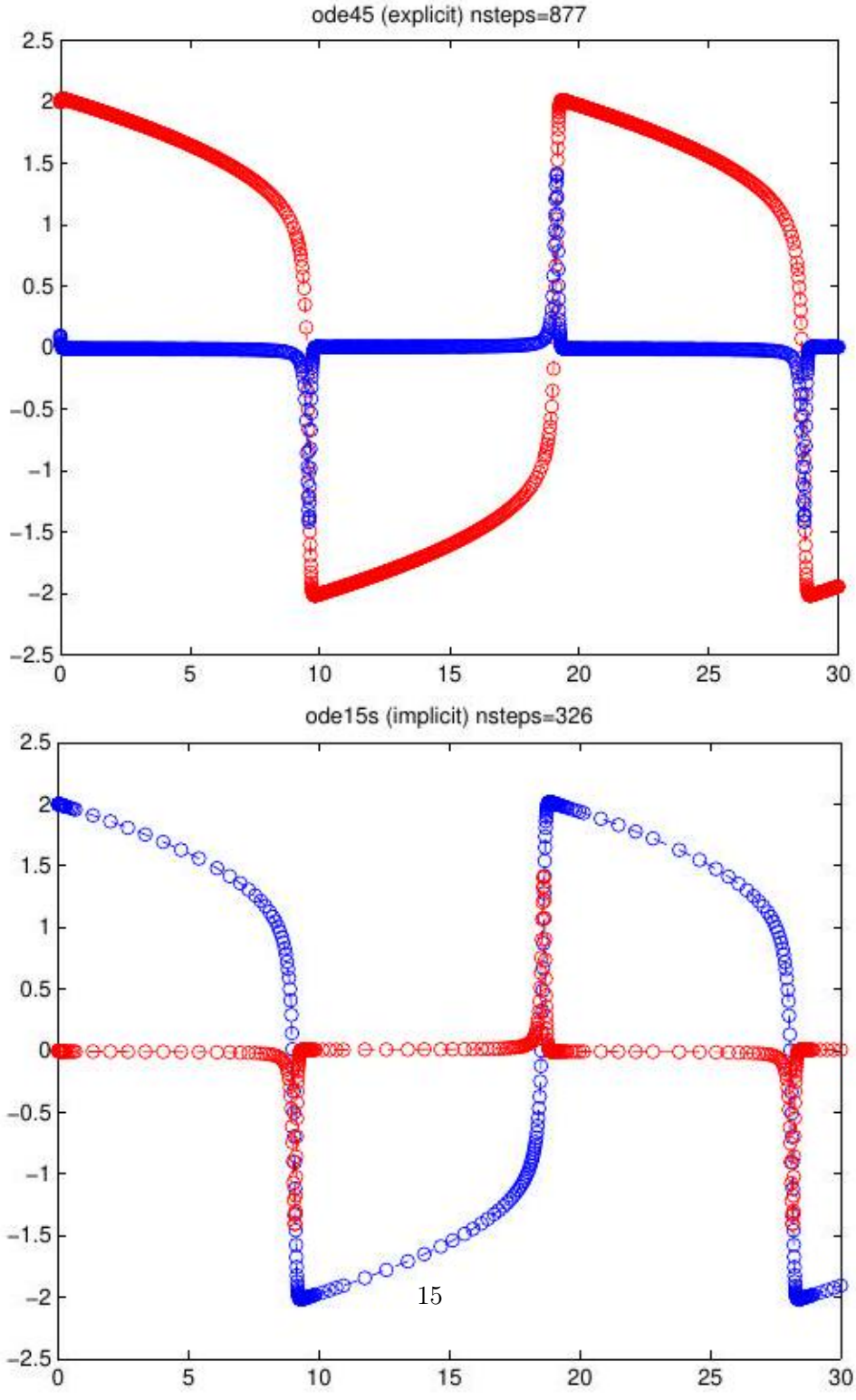
Rigid body motion

```
function dy = rigid(t,y)
dy = zeros(3,1); % a column vector
dy(1) = y(2) * y(3);
dy(2) = -y(1) * y(3);
dy(3) = -0.51 * y(1) * y(2);
%
opts = odeset('RelTol',1e-3, 'AbsTol',[1e-4 1e-4 1e-5
[T,Y] = ode45(@rigid, [0 12], [0 1 1], opts);
plot(T,Y(:,1),'o-r', T,Y(:,2),'s-b', T,Y(:,3),'d-g'
xlabel('t'); ylabel('y'); title('RelTol=1e-3');
```

van der Pol equation

```
$\mathrm{r}=10$; \% Try $\mathrm{r}=100$
$\mathrm{f}=@(\mathrm{t}, \mathrm{y})\left[\mathrm{y}(2) ; \mathrm{r} *\left(1-\mathrm
figure (2); clf
[T,Y] = ode45(f,[0 3*r],[2 1]);
plot(T,Y(:,1), 'o-r', T,Y(:,2)/ r,'o-b')
title(['ode45 (explicit) nsteps=', int2str(size(T,1))]
figure (3); clf
[T,Y] = ode15s(f,[0 3*r],[2 0]);
plot ( $\mathrm{T}, \mathrm{Y}(:, 1),{ }^{\prime}$ o-b', T, $\mathrm{Y}(:, 2) / \mathrm
title(['ode15s (implicit) nsteps=', int2str(size(T,1))
```

Stiff van der Pol system ($r = 10$)



Conclusions/Summary

- Time stepping methods for ODEs are convergent if and only if they are consistent and zero-stable.
- All one-step methods are zero-stable, therefore, there are generic methods that work for any (finite-dimensional) system of ODEs (not true of PDEs).
- We distinguish methods based on their order of accuracy and on whether they are explicit (forward Euler, Heun, RK4, Adams-Bashforth), or implicit (backward Euler, Crank-Nicolson), and whether they are adaptive.
- Runge-Kutta methods require more evaluations of f but are more robust, especially if adaptive (e.g., they can deal with sharp changes in f). Generally the recommended first-try (ode45 or ode23 in MATLAB).