Numerical Methods II
One-Step Methods for ODEs

Aleksandar Donev
Courant Institute, NYU[1]
donev@courant.nyu.edu

[1]MATH-GA.2020-001 / CSCI-GA.2421-001, Spring 2023

Feb 7, 2023

Outline

Initial Value Problems

Initial Value Problems

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

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

  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:

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

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 $\mathbf{x}(t)$:
$$\mathbf{x}(t) = \mathbf{x}_0 + \int_0^t \mathbf{f}\left(\mathbf{x}(s), s\right) ds.$$
- Numerical methods are based on approximations of $\mathbf{f}\left(\mathbf{x}(s), s\right)$ 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:

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

- 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,

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

- 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 $\mathbf{x}^{(k)} = \mathbf{x}\,(k\Delta t)$:

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

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

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

  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:

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

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

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

  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:

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

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

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

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

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

which gives the first-order **backward Euler method**

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

- 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

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

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

  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 $O(\Delta t^3)$, so the global error is **second-order accurate** $O(\Delta t^2)$.

LTE: θ-method (1.4 in Iserles)

$\mathbf{y}'(t) = \mathbf{f}(t, \mathbf{y}(t))$. Scheme $\mathbf{y}_{n+1} = \mathbf{y}_n + h\left[\theta \mathbf{f}(t_n, \mathbf{y}_n) + (1-\theta)\,\mathbf{f}\left(t_{n+1}, \mathbf{y}_{n+}\right.\right.$

$\theta = 1$ is Forward Euler, $\theta = 0$ is Backward Euler, $\theta = 1/2$ is Implicit Trapezoidal

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

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

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

- 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 (LeVeque 5.7)

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

Note that

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

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

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

Using this in (5.31) gives

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

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 $f(x^{\star})$, 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:

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

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

RK4 Method

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

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

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

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

Intro to multistep Methods

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

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

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

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 (after LeVeque)

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

  $$\text{consistency} + \text{(zero) stability} = \text{convergence}.$$

- 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

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

- 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 **x** a neighborhood of the initial condition:

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

  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:
$$\mathbf{x}^{(k)} \approx \mathbf{x}\left(k\Delta t\right).$$
- A method is **convergent** if applying it to any system of ODEs where **f** is Lipshitz over a **finite time interval** $T > 0$ during which the ODE has a solution, the numerical approximation converges to that solution,
$$\lim_{\Delta t \to 0} \mathbf{x}^{(N=T/\Delta t)} = \mathbf{x}\left(T\right).$$
- **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 $\widetilde{L}$,

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

- For example, for explicit midpoint rule,

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

Convergence of One Step Methods

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

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

- 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 $\Delta t\rightarrow 0$.

Error growth factor

- From the definition of the LTE

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

  we get

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

  while our scheme is

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

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

$$\mathbf{E}^{(k+1)} = \mathbf{E}^{(k)} + \Delta t \left(\boldsymbol{\Psi}\left(\mathbf{x}\left(k\Delta t\right), \dots\right) - \boldsymbol{\Psi}\left(\mathbf{x}^{(k)}, \dots\right)\right) + \Delta t \mathbf{e}^{(k)}$$

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

Convergence

- This is the typical relationship we will see many times

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

- 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

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

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

$$\left(1+\Delta t \widetilde{L}\right)^{k}$$

Error bound

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

$$\left(1+\Delta t\widetilde{L}\right) \leq e^{\Delta t\widetilde{L}} \quad \Rightarrow \quad \left(1+\Delta t\widetilde{L}\right)^{k} \leq e^{k\Delta t\widetilde{L}} \leq e^{T\widetilde{L}}.$$

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

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

- 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 $Te^{T\widetilde{L}}$ 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

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

- *ode*23 is a second-order **adaptive explicit** Runge-Kutta method, while *ode*45 is a fourth-order version (try it first).
- *ode*23*tb* is a second-order **implicit** RK method.
- *ode*113 is a **variable-order explicit** multi-step method that can provide very high accuracy.
- *ode*15*s* 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');
```

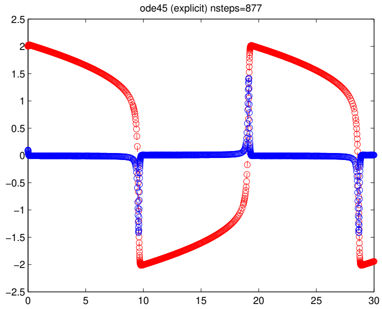
van der Pol equation

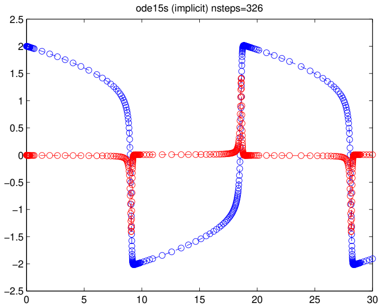
```
r=10; % Try r=100
f = @(t,y) [y(2); r*(1 - y(1)^2)*y(2) - y(1)];

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(T,Y(:,1),'o--b', T,Y(:,2)/r,'o--r')
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).