CS 450: Numerical Anlaysis[1]

Initial Value Problems for Ordinary Differential Equations

University of Illinois at Urbana-Champaign

[1] *These slides have been drafted by Edgar Solomonik as lecture templates and supplementary material for the book "Scientific Computing: An Introductory Survey" by Michael T. Heath (slides).*

Ordinary Differential Equations

- An *ordinary differential equation (ODE)* usually describes time-varying system by a function $\boldsymbol{y}(t)$ that satisfies a set of equations in its derivatives
  The general implicit form is

$$\boldsymbol{g}(t, \boldsymbol{y}, \boldsymbol{y}', \boldsymbol{y}'', \ldots, \boldsymbol{y}^{(k)}) = \boldsymbol{0},$$

  but we restrict focus on the explicit form, $\boldsymbol{y}^{(k)} = \boldsymbol{f}(t, \boldsymbol{y}, \boldsymbol{y}', \boldsymbol{y}'', \ldots, \boldsymbol{y}^{(k-1)})$.
- An ODE of any *order* k can be transformed into a first-order ODE,

$$\boldsymbol{u}' = \begin{bmatrix} \boldsymbol{u}_1' \\ \vdots \\ \boldsymbol{u}_{k-1}' \\ \boldsymbol{u}_k' \end{bmatrix} = \begin{bmatrix} \boldsymbol{u}_2 \\ \vdots \\ \boldsymbol{u}_k \\ \boldsymbol{f}(t, \boldsymbol{u}_1, \ldots, \boldsymbol{u}_k) \end{bmatrix} = \quad \textit{where} \quad \boldsymbol{u}_i(t) = \boldsymbol{y}^{(i-1)}(t).$$

  Consequently we restrict our focus to systems of first-order ODEs. Of particular importance are linear ODEs, which have the form $\boldsymbol{y}' = \boldsymbol{A}(t)\boldsymbol{y}$*, whose coefficients are said to be constant if* $\boldsymbol{A}(t) = \boldsymbol{A}$ *for all* t.

Example: Newton's Second Law

- $F = ma$ corresponds to a second order ODE

$$F(t, y(t), y'(t)) = my''(t)$$
$$y''(t) = f(t, y, y') = F(t, y(t), y'(t))/m$$

- We can transform it into a first order ODE in two variables

$$\boldsymbol{u} = \begin{bmatrix} y(t) \\ y'(t) \end{bmatrix}$$

$$\begin{bmatrix} u_1' \\ u_2' \end{bmatrix} = \boldsymbol{u}' = \boldsymbol{f}(t, \boldsymbol{u}) = \begin{bmatrix} u_2 \\ F(t, \boldsymbol{u})/m \end{bmatrix}$$

Initial Value Problems

- Generally, a first order ODE specifies only the derivative, so the solutions are non-unique, an *initial condition* addresses this:

$$\boldsymbol{y}(t_0) = \boldsymbol{y}_0$$

This condition yields an initial value problem (IVP), which is the simplest example of a boundary condition.

- Given an initial condition, an ODE must satisfy an integral equation for any given point t:

$$\boldsymbol{y}(t) = \boldsymbol{y}_0 + \int_{t_0}^{t} \boldsymbol{f}(s, \boldsymbol{y}(s))ds$$

In the special case that $\boldsymbol{y}' = \boldsymbol{f}(t)$, the integral can be computed directly by numerical quadrature to solve the ODE.

Existence and Uniqueness of Solutions

- For an ODE to have a unique solution, it must be defined on a closed domain D and be *Lipschitz continuous*:

$$\forall \boldsymbol{y}, \hat{\boldsymbol{y}} \in D, \quad ||\boldsymbol{f}(t, \hat{\boldsymbol{y}}) - \boldsymbol{f}(t, \boldsymbol{y})|| \leq L||\hat{\boldsymbol{y}} - \boldsymbol{y}||,$$

i.e. the rate of change of the ODE should itself change continuously. Any differentiable function $\boldsymbol{f}$ is Lipschitz continuous with

$$L = \max_{(t,\boldsymbol{y}) \in D} ||\boldsymbol{J}_{\boldsymbol{f}}(t, \boldsymbol{y})||.$$

- The solutions of an ODE can be stable, unstable, or asymptotically stable:

  Perturbation to the input causes a perturbation to the solution that has bounded growth for a stable ODE, unbounded for an unstable ODE, and shrinks for an asymptotically stable ODE.

Stability of 1D ODEs

- The solution to the scalar ODE $y' = \lambda y$ is $y(t) = y_0 e^{\lambda t}$, with stability dependent on λ:

$$\lim_{t \to \infty} y(t) = \begin{cases} \infty & : \lambda > 0 \textit{ (unstable)} \\ y_0 & : \lambda = 0 \textit{ (stable)} \\ 0 & : \lambda < 0 \textit{ (asymptotically stable)} \end{cases}$$

- A linear ODE generally has the form $\boldsymbol{y}' = \boldsymbol{A}\boldsymbol{y}$, with stability dependent on the spectral radius (largest eigenvalue) of $\boldsymbol{A}$:

  For general ODEs, stability can be ascertained locally by a considering a linear approximatiaion $\boldsymbol{f}(t, \boldsymbol{y}) \approx \boldsymbol{y} + \boldsymbol{J_f}(t)^{-1}\boldsymbol{y}$ and measuring the spectral radius of $\boldsymbol{J_f}(t)^{-1}$.

Numerical Solutions to ODEs

- Methods for numerical ODEs seek to approximate $\boldsymbol{y}(t)$ at $\{t_k\}_{k=1}^m$.

  Compute $\hat{\boldsymbol{y}}_k$ for $k \in \{1, \ldots, m\}$ so as to approxumate $\boldsymbol{y}(t_k) \approx \hat{\boldsymbol{y}}_k$. For an IVP, typically form $\hat{\boldsymbol{y}}_{k+1}$ using $\hat{\boldsymbol{y}}_k$ or additionally (for multistep methods) $\hat{\boldsymbol{y}}_{k-1}, \ldots$

- Euler's method provides the simplest method (attempt) for obtaining a numerical solution:

  Approximation solution to ODE at $t_k + h$ by linear segment from $(t_k, \boldsymbol{y}_k)$ with slope $\boldsymbol{f}(t_k, \boldsymbol{y}_k)$,

  $$\boldsymbol{y}_{k+1} = \boldsymbol{y}_k + h_k \boldsymbol{f}(t_k, \boldsymbol{y}_k).$$

  Its instructive to observe that this approximation arises as the first order form of various models (Taylor series, finite differences, interpolation, quadrature, undetermined coefficients).

Error in Numerical Methods for ODEs

- Truncation error is typically the main quantity of interest, which can be defined *globally* or *locally*
  Global error is measured at all points

$$\boldsymbol{e}_k = \hat{\boldsymbol{y}}_k - \boldsymbol{y}(t_k),$$

  which local error measures the deviation from the exact solution $\boldsymbol{u}_{k-1}(t)$ *passing through the previous point* $(t_{k-1}, \hat{\boldsymbol{y}}_{k-1})$*,*

$$\boldsymbol{l}_k = \hat{\boldsymbol{y}}_k - \boldsymbol{u}_{k-1}(t_k).$$

- The *order of accuracy* of Euler's method is one less than than the order of the leading order term in $\boldsymbol{l}_k$

  Accuracy is of order p *if* $\boldsymbol{l}_k = O(h_k^{p+1})$*. Euler's method is first order accurate by Taylor expansion analysis.*

Stability of Numerical Methods for ODEs

- Stability can be defined for numerical methods similarly to ODEs themselves.

  A method for an ODE with stable solutions can be unstable. Usually, we will seek to establish a stability region for a method, which describes the step-size conditions necessary for stability in terms of the step size h and (the largest) eigenvalue λ, usually as a function of $h\lambda$.

- Basic stability properties follow fron analysis of linear scalar ODE, which serves as a local approximation to more complex ODEs.

  Consider forward Euler for the ODE $y' = \lambda y$, where

$$y_{k+1} = y_k + h\lambda y_k = (1 + h\lambda)y_k.$$

  Euler's method requires $|1 + h\lambda| \leq 1$ to be stable, which implies $-2 \leq h\lambda \leq 0$ for real λ. For a general ODE, the eigenvalues of $\boldsymbol{J}_f$ must lie within a circle on the complex plain centered at -1 with radius 1.

Implicit Methods

- Implicit methods for ODEs form a sequence of solutions that satisfy conditions on a local approximation to the solution:

  The most basic implicit method is the backward Euler method

  $$\boldsymbol{y}_{k+1} = \boldsymbol{y}_k + \boldsymbol{h}_k \boldsymbol{f}(t_{k+1}, \boldsymbol{y}_{k+1}),$$

  which solves for $\boldsymbol{y}_{k+1}$ so that a linear approximation of the solution at t_{k+1} passed through the point $(t_k, \boldsymbol{y}_k)$. Just like forward Euler, first-order accuracy is achieved by the linear approximation.

- The backward Euler method for a simple linear scalar ODE stability region is the left half of the complex plane:

  Such a method is called unconditionally stable. Note that the growth factor can be derived via

  $$y_{k+1} = y_k + h\lambda y_{k+1} = \frac{1}{1 - h\lambda} y_k,$$

  and satisfies $|1/(1 - h\lambda)| \leq 1$ so long as $h\lambda \leq 0$.

Initial Value Problems for ODEs

- We restrict attention to first-order systems of ODEs and pay special attention to linear and constant-coefficient systems:

  An IVP for an ODE usually has the form $\boldsymbol{y}' = \boldsymbol{f}(t, \boldsymbol{y})$ with initial value $(t_0, \boldsymbol{y}(t_0))$. A linear ODE $\boldsymbol{f}$ has the form $\boldsymbol{f}(t, \boldsymbol{y}) = \boldsymbol{A}(t)\boldsymbol{y}(t)$, and is said to have constant coefficients if $\boldsymbol{A}(t)$ does not vary with t.

- Existence and uniqueness of a solution to an IVP is guaranteed over any domain D on which $\boldsymbol{f}$ is Lipschitz continuous:

$$\exists L \in \mathbb{R}, ||\boldsymbol{f}(t, \boldsymbol{y}) - \boldsymbol{f}(t, \hat{\boldsymbol{y}})|| \leq L||\boldsymbol{y} - \hat{\boldsymbol{y}}||, \forall \boldsymbol{y}, \hat{\boldsymbol{y}} \in D$$

  which is stronger than continuity of $\boldsymbol{f}$ (differentiability of $\boldsymbol{y}$), but weaker than differentiability of $\boldsymbol{f}$, i.e. $\boldsymbol{f}$ can suddenly begin to change at a different rate. The constant L bounds the rate at which similar solutions $\boldsymbol{y}$ and $\hat{\boldsymbol{y}}$ can diverge or converge.

Convergence and Stability

- Generally, we seek to approximate $\boldsymbol{y}(t_k)$ for a set of points $t_k = t_0 + kh$ by $\hat{y}_k$: *Would like to converge with decreasing step-size, i.e.* $\lim_{h\to 0} \hat{\boldsymbol{y}}_k = \boldsymbol{y}(t_k)$*. Can measure global error (deviation from true solution) or local error (deviation of* $\hat{\boldsymbol{y}}_k$ *from* $\boldsymbol{u}_{k-1}(t_k)$ *where* $\boldsymbol{u}_{k-1}(t_{k-1}) = \hat{\boldsymbol{y}}_{k-1}$ *and* $\boldsymbol{u}'_{k-1} = \boldsymbol{y}'$*).*
- Stability ascertains behavior as $t \to \infty$ either of the ODE itself or of a numerical method:
  - *For an ODE, stability identifies convergence/divergence of perturbed solutions.*
  - *For a numerical method, it identifies step-size bounds necessary to ensure the method converges along with the ODE.*
  - *If a method is stable, the local error provides a trustworthy metric of global error.*
  - *A stiff ODE is a convergent ODE with rapidly varying small components, which poses a challenge for stability of methods.*

Accuracy and Taylor Series Methods

- By taking a degree-r Taylor expansion of the ODE in t, at each consecutive $(t_k, \hat{\boldsymbol{y}}_k)$, we achieve kth order accuracy.

  We can bound the local approximation error as the error the Taylor expansion,

  $$\boldsymbol{y}(t_k + h) = \boldsymbol{y}(t_k) + \boldsymbol{y}'(t_k)h + \cdots + \boldsymbol{y}^{(r)}(t_k)h^{r-1}/r!$$

  which is $O(h^{r+1})$, which leads to $O(h^r)$ accuracy in the approximation to $\boldsymbol{f}(t, \boldsymbol{y})$. Euler's method is a first-order Taylor series method.

- Taylor series methods require high-order derivatives at each step:

  Analytic differentiation is expensive, so seek to approximate. Can perform a finite-differencing approximation by evaluating at points near $(t_k, \hat{\boldsymbol{y}}_k)$ (multi-stage methods) or simply using previous points, e.g. $(t_{k-1}, \hat{\boldsymbol{y}}_{k-1})$ (multi-step methods).

Multi-Stage Methods

- Multi-stage methods construct $\hat{\boldsymbol{y}}_{k+1}$ by approximating $\boldsymbol{y}$ between t_k and t_{k+1}:
  Runge-Kutta methods are the most well-known family of these, simple example is Heun's method,

$$\hat{\boldsymbol{y}}_{k+1} = \hat{\boldsymbol{y}}_k + h\left[\underbrace{\boldsymbol{f}(t_k, \boldsymbol{y}_k)}_{\boldsymbol{v}_1}/2 + \boldsymbol{f}\Big(t_k + h, \boldsymbol{y}_k + h\underbrace{\boldsymbol{f}(t_k, \boldsymbol{y}_k)}_{\boldsymbol{v}_1}\Big)/2\right].$$

  We can think of the above method as employing the trapezoid quadrature rule. The difference between Heun's method and the (implicit) trapezoid method is that we evaluate at $\boldsymbol{f}(t_k + h, \boldsymbol{y}_k + h\boldsymbol{v}_1)$ rather than working with the implicit value of $\boldsymbol{f}(t_k + h, \boldsymbol{y}_{k+1})$.
- The 4th order Runge-Kutta scheme is particularly popular:
  This scheme uses Simpson's rule

$$\begin{aligned}
\hat{\boldsymbol{y}}_{k+1} &= \hat{\boldsymbol{y}}_k + (h/6)(\boldsymbol{v}_1 + 2\boldsymbol{v}_2 + 2\boldsymbol{v}_3 + \boldsymbol{v}_4) \\
\boldsymbol{v}_1 &= \boldsymbol{f}(t_k, \boldsymbol{y}_k), & \boldsymbol{v}_2 &= \boldsymbol{f}(t_k + h/2, \boldsymbol{y}_k + (h/2)\boldsymbol{v}_1), \\
\boldsymbol{v}_3 &= \boldsymbol{f}(t_k + h/2, \boldsymbol{y}_k + (h/2)\boldsymbol{v}_2), & \boldsymbol{v}_4 &= \boldsymbol{f}(t_k + h, \boldsymbol{y}_k + h\boldsymbol{v}_3)
\end{aligned}$$

Runge-Kutta Methods

- Runge-Kutta methods evaluate $\boldsymbol{f}$ at $t_k + c_i h$ for $c_0, \ldots, c_r \in [0, 1]$,
  A Runge-Kutta method can be derived as a quadrature rule

$$\boldsymbol{u}_k(t_{k+1}) = \hat{\boldsymbol{y}}_k + \int_{t_k}^{t_k+h} \boldsymbol{f}(s, \boldsymbol{y}(s))ds \quad \approx \quad \hat{\boldsymbol{y}}_k + h \sum_{i=0}^{r-1} w_i \boldsymbol{f}(t_k + c_i h, \hat{\boldsymbol{y}}_{ki}),$$

where $\{(c_i, w_i)\}_{i=0}^{r}$ are quadrature node, weight pairs, but we still have flexibility in choosing $\hat{\boldsymbol{y}}_{ki}$. One good choice is to successively construct

$$\hat{\boldsymbol{y}}_{ki} = \hat{\boldsymbol{y}}_k + h \sum_j a_{ij} \boldsymbol{f}(t_k + c_i h, \hat{\boldsymbol{y}}_{kj}) = \hat{\boldsymbol{y}}_k + h c_i \boldsymbol{f}(t_k + c_i h, \hat{\boldsymbol{y}}_{k,i-1}).$$

More general choices for a_{ij} are often represented by a Butcher tableau,

$\boldsymbol{c}$	$\boldsymbol{A}$
	$\boldsymbol{w}^T$

e.g. for RK4,

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

If $a_{ij} = 0$ for $j \geq i$, the scheme is explicit, if for $j > i$ then diagonally implicit, and otherwise implicit.

Properties of Runge-Kutta and Extrapolation Methods

- Runge-Kutta methods are self-starting, but are harder to use to obtain error estimates.

  Self-starting means that we only need $\hat{\boldsymbol{y}}_k$ to form $\hat{\boldsymbol{y}}_{k+1}$. Embedded Runge-Kutta schemes provides 4th + 5th order results, yielding an error estimate.

- Extrapolation methods achieve high accuracy by successively reducing step-size.

  Use single-step method with step sizes $h, h/2, h/4, \ldots$ to approximate solution at $t_k + h$.

Multistep Methods

- Multistep methods employ $\{\hat{\boldsymbol{y}}_k\}_{i=0}^{k}$ to compute $\hat{\boldsymbol{y}}_{k+1}$:

  Linear multistep methods have the form,

$$\hat{\boldsymbol{y}}_{k+1} = \sum_{i=1}^{m} \alpha_i \hat{\boldsymbol{y}}_{k+1-i} + h \sum_{i=0}^{m} \beta_i \boldsymbol{f}(t_{k+1-i}, \boldsymbol{y}_{k+1-i}).$$

  Interpolation is used to determine each α_i and β_i, method is explicit if $\beta_0 = 0$.

- Multistep methods are not self-starting

  However, they require few function evaluations. Multivalue methods generalize multistep methods to non-uniformly-spaced points.