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.

- An ODE of any *order* k can be transformed into a first-order ODE,

Example: Newton's Second Law

- Consider, $F = ma$ for a given force F, which is a second order ODE,
- We can transform it into a first order ODE in two variables:

Initial Value Problems

- Generally, a first order ODE specifies only the derivative, so the solutions are non-unique. An *initial condition* addresses this:
- Given an initial condition, an ODE must satisfy an integral equation for any given point t:

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*:

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

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:

- A constant-coefficient linear ODE has the form $\boldsymbol{y}' = \boldsymbol{A}\boldsymbol{y}$, with stability dependent on the real parts of the eigenvalues of $\boldsymbol{A}$:

***Demo:** Forward Euler stability*

Numerical Solutions to ODEs

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

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

Error in Numerical Methods for ODEs

- Truncation error is typically the main quantity of interest, which can be defined *globally* or *locally*:

- The *order of accuracy* of a given method is one less than than the order of the leading order term in the local error $\boldsymbol{l}_k$:

Accuracy and Taylor Series Methods

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

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

Growth Factors and Stability Regions

- Stability of an ODE method discerns whether local errors are amplified, deamplified, or stay constant:

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

Stability Region for Forward Euler

- The stability region of a general ODE constrains the eigenvalues of $h\boldsymbol{J_f}$

Backward Euler Method

Demo: *Backward Euler stability*
Activity: *Backward Euler Method*

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

- The stability region of the backward Euler method is the left half of the complex plane:

Stiffness

- *Stiff* ODEs are ones that contain components that vary at disparate time-scales:

Trapezoid Method

- A second-order accurate implicit method is the *trapezoid method*

- Generally, methods can be derived from quadrature rules:

Multi-Stage Methods

- *Multi-stage methods* construct $\boldsymbol{y}_{k+1}$ by approximating $\boldsymbol{y}$ between t_k and t_{k+1}:

- The 4th order Runge-Kutta scheme is particularly popular:

  This scheme uses Simpson's rule,

$$\boldsymbol{y}_{k+1} = \boldsymbol{y}_k + (h/6)(\boldsymbol{v}_1 + 2\boldsymbol{v}_2 + 2\boldsymbol{v}_3 + \boldsymbol{v}_4)$$

$$\boldsymbol{v}_1 = \boldsymbol{f}(t_k, \boldsymbol{y}_k), \qquad \boldsymbol{v}_2 = \boldsymbol{f}(t_k + h/2, \boldsymbol{y}_k + (h/2)\boldsymbol{v}_1),$$

$$\boldsymbol{v}_3 = \boldsymbol{f}(t_k + h/2, \boldsymbol{y}_k + (h/2)\boldsymbol{v}_2), \qquad \boldsymbol{v}_4 = \boldsymbol{f}(t_k + h, \boldsymbol{y}_k + h\boldsymbol{v}_3).$$

Runge-Kutta Methods

Demo: Dissipation in Runge-Kutta Methods
Activity: Diagonally Implicit Runge Kutta

- Runge-Kutta methods evaluate $\boldsymbol{f}$ at $t_k + c_i h$ for $c_0, \ldots, c_r \in [0, 1]$,

- A general family of Runge Kutta methods can be defined by

Multistep Methods

- *Multistep methods* employ $\{\boldsymbol{y}_i\}_{i=0}^{k}$ to compute $\boldsymbol{y}_{k+1}$:
- Multistep methods are not self-starting, but have practical advantages: