# CS 450: Numerical Anlaysis

## Lecture 24

## Chapter 9 Initial Value Problems for Ordinary Differential Equations

### Numerical Methods for Solving ODE IVPs

Edgar Solomonik

Department of Computer Science
University of Illinois at Urbana-Champaign

April 18, 2018

# 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 $k$th 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 $\alpha_i$ and $\beta_i$, 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.*