# AM 205: lecture 13

- Last time: Numerical solution of ordinary differential equations
- Today: Additional ODE methods, boundary value problems
- Thursday's lecture will be given by Thomas Fai
- Assignment 3 will be posted tonight

# Runge–Kutta Methods

The family of Runge–Kutta methods with two intermediate evaluations is defined by

$$y_{k+1} = y_k + h(ak_1 + bk_2),$$

where $k_1 = f(t_k, y_k)$, $k_2 = f(t_k + \alpha h, y_k + \beta h k_1)$

The Euler method is a member of this family, with $a = 1$ and $b = 0$. By careful analysis of the truncation error, it can be shown that we can choose $a, b, \alpha, \beta$ to obtain a second-order method

# Runge–Kutta Methods

Three such examples are:

- The modified Euler method ($a = 0$, $b = 1$, $\alpha = \beta = 1/2$):

$$y_{k+1} = y_k + hf\left(t_k + \frac{1}{2}h, y_k + \frac{1}{2}hf(t_k, y_k)\right)$$

- The improved Euler method (or Heun's method) ($a = b = 1/2$, $\alpha = \beta = 1$):

$$y_{k+1} = y_k + \frac{1}{2}h[f(t_k, y_k) + f(t_k + h, y_k + hf(t_k, y_k))]$$

- Ralston's method ($a = 1/4$, $b = 3/4$, $\alpha = 2/3$, $\beta = 2/3$)

$$y_{k+1} = y_k + \frac{1}{4}h[f(t_k, y_k) + 3f(t_k + \tfrac{2h}{3}, y_k + \tfrac{2h}{3}f(t_k, y_k))]$$

# Runge–Kutta Methods

The most famous Runge–Kutta method is the "classical fourth-order method", RK4 (used by MATLAB's `ode45`):

$$y_{k+1} = y_k + \frac{1}{6}h(k_1 + 2k_2 + 2k_3 + k_4)$$

where

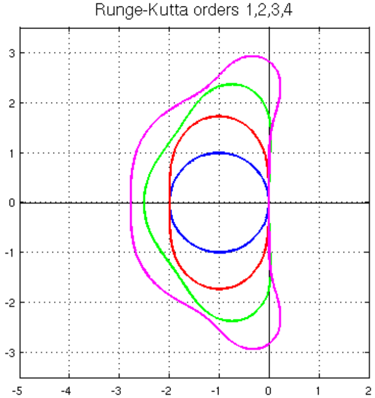
$$\begin{aligned}
k_1 &= f(t_k, y_k) \\
k_2 &= f(t_k + h/2, y_k + hk_1/2) \\
k_3 &= f(t_k + h/2, y_k + hk_2/2) \\
k_4 &= f(t_k + h, y_k + hk_3)
\end{aligned}$$

Analysis of the truncation error in this case (which gets quite messy!) gives $T_k = O(h^4)$

# Runge–Kutta Methods: Stability

We can also examine stability of RK4 methods for $y' = \lambda y$

Figure shows stability regions for four different RK methods (higher order RK methods have larger stability regions here)



# Butcher tableau

Can summarize an $s + 1$ stage Runge–Kutta method using a triangular grid of coefficients

$$\begin{array}{c|ccccc}
\alpha_0 & & & & & \\
\alpha_1 & \beta_{1,0} & & & & \\
\vdots & \vdots & & & & \\
\alpha_s & \beta_{s,0} & \beta_{s,1} & \dots & \beta_{s,s-1} & \\
\hline
 & \gamma_0 & \gamma_1 & \dots & \gamma_{s-1} & \gamma_s
\end{array}$$

The $i$th intermediate step is

$$f(t_k + \alpha_i h, y_k + h \sum_{j=0}^{i-1} k_j).$$

The $(k+1)$th answer for $y$ is

$$y_{k+1} = y_k + h \sum_{j=0}^{s} \gamma_j k_j.$$

# Estimation of error

First approach: Richardson extrapolation.

Suppose that $y_{k+2}$ is the numerical result of two steps with size $h$ of a Runge–Kutta method of order $p$, and $w$ is the result of one big step with step size $2h$. Then the error of $y_2$ can be approximated as

$$y(t_k + 2h) - y_{k+2} = \frac{y_{k+2} - w}{2^p - 1} + O(h^{p+2})$$

and

$$\hat{y}_{k+2} = y_{k+2} + \frac{y_2 - w}{2^p - 1}$$

is an approximation of order $p + 1$ to $y(t_0 + 2h)$.

# Estimation of error

Second approach: can derive Butcher tableaus that contain an additional higher-order formula for estimating error. *e.g.* Fehlberg's order 4(5) method, RKF45

$$
\begin{array}{c|cccccc}
0 & & & & & & \\
\frac{1}{4} & \frac{1}{4} & & & & & \\
\frac{3}{8} & \frac{3}{32} & \frac{9}{32} & & & & \\
\frac{12}{13} & \frac{1932}{2197} & -\frac{7200}{2197} & \frac{7296}{2197} & & & \\
1 & \frac{439}{216} & -8 & \frac{3680}{513} & -\frac{845}{4104} & & \\
\frac{1}{2} & \frac{-8}{27} & 2 & \frac{-3544}{2565} & \frac{1859}{4104} & \frac{-11}{40} & \\
\hline
y_{k+1} & \frac{25}{216} & 0 & \frac{1408}{2565} & \frac{2197}{4104} & -\frac{1}{5} & 0 \\
\hat{y}_{k+1} & \frac{16}{135} & 0 & \frac{6656}{12825} & \frac{28561}{56430} & -\frac{9}{50} & \frac{2}{55}
\end{array}
$$

$y_{k+1}$ is order 4 and $\hat{y}_{k+1}$ is order 5. Use $y_{k+1} - \hat{y}_{k+1}$ as an error estimate.

# Higher-order methods

Fehlberg's 7(8) method[1]

$$
\begin{array}{c|ccccccccccccc}
0 & & & & & & & & & & & & & \\
\frac{2}{27} & \frac{2}{27} & & & & & & & & & & & & \\
\frac{1}{9} & \frac{1}{36} & \frac{1}{12} & & & & & & & & & & & \\
\frac{1}{6} & \frac{1}{24} & 0 & \frac{1}{8} & & & & & & & & & & \\
\frac{5}{12} & \frac{5}{12} & 0 & -\frac{25}{16} & \frac{25}{16} & & & & & & & & & \\
\frac{1}{2} & \frac{1}{20} & 0 & 0 & \frac{1}{4} & \frac{1}{5} & & & & & & & & \\
\frac{5}{6} & -\frac{25}{108} & 0 & 0 & \frac{125}{108} & -\frac{65}{27} & \frac{125}{54} & & & & & & & \\
\frac{1}{6} & \frac{31}{300} & 0 & 0 & 0 & \frac{61}{225} & -\frac{2}{9} & \frac{13}{900} & & & & & & \\
\frac{2}{3} & 2 & 0 & 0 & -\frac{53}{6} & \frac{704}{45} & -\frac{107}{9} & \frac{67}{90} & 3 & & & & & \\
\frac{1}{3} & -\frac{91}{108} & 0 & 0 & \frac{23}{108} & -\frac{976}{135} & \frac{311}{54} & -\frac{19}{60} & \frac{17}{6} & -\frac{1}{12} & & & & \\
1 & \frac{2383}{4100} & 0 & 0 & -\frac{341}{164} & \frac{4496}{1025} & -\frac{301}{82} & \frac{2133}{4100} & \frac{45}{82} & \frac{45}{164} & \frac{18}{41} & & & \\
0 & \frac{3}{205} & 0 & 0 & 0 & 0 & -\frac{6}{41} & -\frac{3}{205} & -\frac{3}{41} & \frac{3}{41} & \frac{6}{41} & 0 & & \\
1 & -\frac{1777}{4100} & 0 & 0 & -\frac{341}{164} & \frac{4496}{1025} & -\frac{289}{82} & \frac{2193}{4100} & \frac{51}{82} & \frac{33}{164} & \frac{12}{41} & 0 & 1 & \\
\hline
y_1 & \frac{41}{840} & 0 & 0 & 0 & 0 & \frac{34}{105} & \frac{9}{35} & \frac{9}{35} & \frac{9}{280} & \frac{9}{280} & \frac{41}{840} & 0 & 0 \\
\hline
\widehat{y}_1 & 0 & 0 & 0 & 0 & 0 & \frac{34}{105} & \frac{9}{35} & \frac{9}{35} & \frac{9}{280} & \frac{9}{280} & 0 & \frac{41}{840} & \frac{41}{840}
\end{array}
$$

[1]From *Solving Ordinary Differential Equations* by Hairer, Nørsett, and Wanner.

# Stiff systems

You may have heard of "stiffness" in the context of ODEs: an important, though somewhat fuzzy, concept

Common definition of stiffness for a linear ODE system $y' = Ay$ is that $A$ has eigenvalues that differ greatly in magnitude[2]

The eigenvalues determine the time scales, and hence large differences in $\lambda$'s $\implies$ resolve disparate timescales simultaneously!

[2] Nonlinear case: stiff if the Jacobian, $J_f$, has large differences in eigenvalues, but this defn. isn't always helpful since $J_f$ changes at each time-step

# Stiff systems

Suppose we're primarily interested in the long timescale. Then:

- We'd like to take large time steps and resolve the long timescale accurately
- But we may be forced to take extremely small timesteps to avoid instabilities due to the fast timescale

In this context it can be highly beneficial to use an implicit method since that enforces stability regardless of timestep size

# Stiff systems

From a practical point of view, an ODE is stiff if there is a significant benefit in using an implicit instead of explicit method

*e.g.* this occurs if the time-step size required for stability is much smaller than size required for the accuracy level we want

Example: Consider $y' = Ay$, $y_0 = [1, 0]^T$ where

$$A = \begin{bmatrix} 998 & 1998 \\ -999 & -1999 \end{bmatrix}$$

which has $\lambda_1 = -1$, $\lambda_2 = -1000$ and exact solution

$$y(t) = \begin{bmatrix} 2e^{-t} - e^{-1000t} \\ -e^{-t} + e^{-1000t} \end{bmatrix}$$

# Multistep Methods

So far we have looked at one-step methods, but to improve efficiency why not try to reuse data from earlier time-steps?

This is exactly what multistep methods do:

$$y_{k+1} = \sum_{i=1}^{m} \alpha_i y_{k+1-i} + h \sum_{i=0}^{m} \beta_i f(t_{k+1-i}, y_{k+1-i})$$

If $\beta_0 = 0$ then the method is explicit

We can derive the parameters by interpolating and then integrating the interpolant

# Multistep Methods

The stability of multistep methods, often called "zero stability," is an interesting topic, but not considered here

Question: Multistep methods require data from several earlier time-steps, so how do we initialize?

Answer: The standard approach is to start with a one-step method and move to multistep once there is enough data

Some key advantages of one-step methods:

- They are "self-starting"
- Easier to adapt time-step size

# ODE Boundary Value Problems

# ODE BVPs

Consider the ODE Boundary Value Problem (BVP):[3] find $u \in C^2[a, b]$ such that

$$-\alpha u''(x) + \beta u'(x) + \gamma u(x) = f(x), \quad x \in [a, b]$$

for $\alpha, \beta, \gamma \in \mathbb{R}$ and $f : \mathbb{R} \to \mathbb{R}$

The terms in this ODE have standard names:

| | |
|---|---|
| $-\alpha u''(x)$: | diffusion term |
| $\beta u'(x)$: | convection (or transport) term |
| $\gamma u(x)$: | reaction term |
| $f(x)$: | source term |

[3]Often called a "Two-point boundary value problem"

# ODE BVPs

Also, since this is a BVP $u$ must satisfy some boundary conditions, *e.g.* $u(a) = c_1$, $u(b) = c_2$

$u(a) = c_1$, $u(b) = c_2$ are called Dirichlet boundary conditions

Can also have:

- A Neumann boundary condition: $u'(b) = c_2$
- A Robin (or "mixed") boundary condition:[4]
  $u'(b) + c_2 u(b) = c_3$

[4] With $c_2 = 0$, this is a Neumann condition

# ODE BVPs

This is an ODE, so we could try to use the ODE solvers from $\mathrm{III}.3$ to solve it!

Question: How would we make sure the solution satisfies $u(b) = c_2$?

# ODE BVPs

Answer: Solve the IVP with $u(a) = c_1$ and $u'(a) = s_0$, and then update $s_k$ iteratively for $k = 1, 2, \ldots$ until $u(b) = c_2$ is satisfied

This is called the "shooting method", we picture it as shooting a projectile to hit a target at $x = b$ (just like Angry Birds!)

However, the shooting method does not generalize to PDEs hence it is not broadly useful