Lecture 6
Solving ODEs
General Stuff

- ODEs occur quite often in physics and astrophysics:
  - Wave Equation in 1-D
  - stellar structure equations
  - hydrostatic equation in atmospheres
  - orbits
- need *workhorse* solvers to deal with them
- PDEs are more common
these methods apply to all ODEs

higher order ODEs are transformed to systems of first order ODEs, e.g.,

\[
\frac{d^2y}{dx^2} + q(x) \frac{dy}{dx} = r(x)
\]

transforms to

\[
\frac{dy_1}{dx} = y_2(x) \tag{1}
\]
\[
\frac{dy_2}{dx} = r(x) - q(x)y_2(x) \tag{2}
\]

with \( y(x) = y_1(x) \) and \( dy/dx = y_2(x) \)
need methods to solve a set of ODEs

\[
\frac{dy_i}{dx} = f_i(x, y_{(1...i)})
\]

equations of more than one independent variable → PDEs
require far more complex methods
next chapter
Boundary Conditions

- need BCs to solve a ODE problem numerically
- analytical solution → integration constants
- type of BCs determine numerical approach, e.g.:
  1. initial value problem: all $y_i$ are given at one starting point $x_s$
  2. two-point boundary value problem: some $y_i$ are given at $x_s$, the others at some point $x_f$
- first: initial value problems
Euler method

- consider only ODE with $i = 1$ for simplicity!
- index $y_n \rightarrow$ value of $y(x)$ at $x = x_n$, i.e., the $n$th point going from $x_s$ to $x_f$
- simplest approach: Euler
- chose stepsize $h$ for $x$ then write

$$y_{n+1} = y_n + hf(x_n, y_n)$$

- uses information only at the beginning of an interval
- expansion in power series $\rightarrow$
- *first order accuracy*:
- error is $O(h^2)$
Euler method

Figure 16.1.1. Euler’s method. In this simplest (and least accurate) method for integrating an ODE, the derivative at the starting point of each interval is extrapolated to find the next function value. The method has first-order accuracy.
Euler method

- not recommended for practical use:
  - Euler Predictor Corrector is better
  - other methods have better accuracy for same step size
  - instability problems (see below)
**Euler Predictor-Corrector**

\[ y_1 = y_0 + \frac{d f}{d x} \bigg|_{x_0} \, h \]  
\[ y_0 = y_1 - \frac{d f}{d x} \bigg|_{x_1} \, h \]  

- then slope of chord is

\[ \frac{y_1 - y_0}{h} = \frac{1}{2} \{ f(x_0, y_0) + f(x_1, y_1) \} \]

- but don’t know \( y_1 \)
- use Euler step to estimate (predictor step)

\[ Y_1 = y_0 + f(x_0, y_0) \, h \]

- then use chord to advance the equation

\[ y_1 = y_0 + \frac{1}{2} \{ f(x_0, y_0) + f(x_1, Y_1) \} \]
Euler PC Method

Ordinary differential equations

Fig. 14. (a) The Euler method. (b) The Euler predictor-corrector method. The slope of the chord is approximately the average of the slopes at $(x_0, y_0)$ and $(x_1, y_1)$. 
Runge-Kutta method

- use Euler to take trial step to $x_n + h/2$
- use Euler estimate to approximate $x$ and $y$ at midpoint
- use these to compute full step
- → midpoint method
- → second order Runge-Kutta method
Figure 16.1.2. Midpoint method. Second-order accuracy is obtained by using the initial derivative at each step to find a point halfway across the interval, then using the midpoint derivative across the full width of the interval. In the figure, filled dots represent final function values, while open dots represent function values that are discarded once their derivatives have been calculated and used.
Runge-Kutta method

- formally:

\[ k_1 = hf(x_n, y_n) \]  
\[ k_2 = hf(x_n + h/2, y_n + k_1/2) \]  
\[ y_{n+1} = y_n + k_2 + O(h^3) \]  

- → second order accurate!
- needs 2 evals of \( f() \) for stepsize \( h \)
- can be extended
- by using several estimates at start/mid/endpoints
- accuracy can be increased
4th order RK method

Figure 16.1.3. Fourth-order Runge-Kutta method. In each step the derivative is evaluated four times: once at the initial point, twice at trial midpoints, and once at a trial endpoint. From these derivatives the final function value (shown as a filled dot) is calculated. (See text for details.)
Runge-Kutta method

formally:

\begin{align*}
    k_1 &= hf(x_n, y_n) \quad (8) \\
    k_2 &= hf(x_n + h/2, y_n + k_1/2) \quad (9) \\
    k_3 &= hf(x_n + h/2, y_n + k_2/2) \quad (10) \\
    k_4 &= hf(x_n + h, y_n + k_3) \quad (11) \\
    y_{n+1} &= y_n + k_1/6 + k_3/3 + k_4/6 + O(h^5) \quad (12)
\end{align*}

fourth order accurate

better than 2nd order if $h$ is at least 2 times larger for same quality
step size control

- very often $f()$ changes significantly over integration interval
- to ensure homogeneous quality we need to adapt $h$
- needs to be really small in domains where $f()$ has nasty features
- $h$ can be larger for smooth, slowly varying functions
step size control

- general ODE solver needs to track accuracy to obtain best performance for given quality
- **step-doubling:**
  - take each step twice as $h$ and two $h/2$ steps
  - total of 3 RK steps with 4 function evals each
  - starting point shared → 11 evals
  - overhead is $11/8 \approx 1.38$
    (we reach accuracy for step size $h/2$!)
step size control

- indicate truncation error $\Delta \longrightarrow$ difference between the two estimates
- use this information to adapt $h$ automatically
- method is 4th order $\longrightarrow$
- $\Delta$ scales as $h^5 \longrightarrow$

$$h_0 = h_1 \left( \frac{\Delta_0}{\Delta_1} \right)^{0.2}$$

given a target truncation error $\Delta_0$, use this to estimate $h_0$ from current $\Delta_1$ and $h_1$
- if $\Delta_1 > \Delta_0 \longrightarrow$ decrease $h$
- otherwise increase it
Fehlberg method

- RK of $M > 4$th order $\rightarrow$ up to $M + 2$ evals
- 4th order most efficient?
- Fehlberg $\rightarrow$
- 5th order method with 6 evals
- these 6 points can also be used to build a 4th order formula
- $\rightarrow$ combine them to compute truncation error with only 6 evals!
- *embedded RK-Fehlberg method*
Embedded RKF method

\[ k_1 = hf(x_n, y_n) \]
\[ k_2 = hf(x_n + a_2h, y_n + b_{21}k_1) \]
\[ \ldots \]
\[ k_6 = hf(x_n + a_6h, y_n + b_{61}k_1 + \cdots + b_{65}k_5) \]
\[ y_{n+1} = y_n + c_1k_1 + c_2k_2 + c_3k_3 + c_4k_4 + c_5k_5 + c_6k_6 + O(h^6) \]

The embedded fourth-order formula is

\[ y_{n+1}^* = y_n + c_1^*k_1 + c_2^*k_2 + c_3^*k_3 + c_4^*k_4 + c_5^*k_5 + c_6^*k_6 + O(h^5) \]

and so the error estimate is

\[ \Delta \equiv y_{n+1} - y_{n+1}^* = \sum_{i=1}^{6} (c_i - c_i^*)k_i \]
### Cash-Karp Parameters for Embedded Runga-Kutta Method

<table>
<thead>
<tr>
<th>$i$</th>
<th>$a_i$</th>
<th>$b_{ij}$</th>
<th>$c_i$</th>
<th>$c_i^*$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$\frac{1}{5}$</td>
<td>$\frac{1}{5}$</td>
<td>$\frac{37}{378}$</td>
<td>$\frac{2825}{27648}$</td>
</tr>
<tr>
<td>2</td>
<td>$\frac{3}{10}$</td>
<td>$\frac{3}{40}$</td>
<td>$\frac{9}{40}$</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>$\frac{3}{5}$</td>
<td>$\frac{3}{10}$</td>
<td>$-\frac{9}{10}$</td>
<td>$\frac{6}{5}$</td>
</tr>
<tr>
<td>4</td>
<td>$\frac{3}{5}$</td>
<td>$\frac{3}{10}$</td>
<td>$-\frac{9}{10}$</td>
<td>$\frac{6}{5}$</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>$-\frac{11}{54}$</td>
<td>$\frac{5}{2}$</td>
<td>$-\frac{70}{27}$</td>
</tr>
<tr>
<td>6</td>
<td>$\frac{7}{8}$</td>
<td>$\frac{1631}{55296}$</td>
<td>$\frac{175}{512}$</td>
<td>$\frac{575}{13824}$</td>
</tr>
</tbody>
</table>

$\hat{j} = 1, 2, 3, 4, 5$
Modified Midpoint Method

- go through a (large) step $H$ with $h/n$ sub-steps
- use an algorithm of the form

\[
\begin{align*}
    z_0 & = y(x) \\
    z_1 & = z_0 + hf(x, z_0) \\
    z_{m+1} & = z_{m-1} + 2hf(x + mh, z_m) \\
    y(z + H) & = \frac{1}{2} [z_n + z_{n-1} + hf(x + H, z_n)]
\end{align*}
\]

- same as midpoint method but first and last points
- → modified midpoint method
Modified Midpoint Method

- truncation error \( y_n - y(x + H) = \sum_i \alpha_i h^{2i} \)

- only even powers in \( h \)
- combining steps will deliver two orders better accuracy per doubling
- usually adaptive RKF’s are better
- but modified midpoint is very useful for . . .
Figure 16.4.1. Richardson extrapolation as used in the Bulirsch-Stoer method. A large interval $H$ is spanned by different sequences of finer and finer substeps. Their results are extrapolated to an answer that is supposed to correspond to infinitely fine substeps. In the Bulirsch-Stoer method, the integrations are done by the modified midpoint method, and the extrapolation technique is rational function or polynomial extrapolation.
Bulirsch-Stoer method

- original Richardson $\rightarrow$ power series
- limited convergence radius!
- Bulirsch-Stoer:
  - use modified midpoint method
  - use rational functions in $h^2$
- $\rightarrow$ can take rather large steps $H$
Bulirsch-Stoer method

- how to subdivide steps best?
- $H/n_j$
- Bulirsch-Stoer: $n_j = 2n_{j-2}$
- more efficient: $n_j = 2j$ (Deuflhard)
Bulirsch-Stoer method

- $j_{\text{max}}$ is not determined a priori
- extrapolation delivers error limits
- use them to limit steps
- variation: polynomial extrapolation
Bulirsch-Stoer method

- Method works extremely well for well-behaved ODEs
- Smooth functions

It does not work well for
- Function evaluations through table lookup or interpolation
- Discontinuous functions
- Internal singular points
- Steep changes of $f$ through an interval
Predictor-Corrector Methods

- write the solution of the ODE as
  \[ y_{n+1} = y_n + \int_{x_n}^{x_{n+1}} f(t, y) \, dt \]

- approximate \( f(t, y) \) by a polynomial through a number of previous points \( x_n, x_{n-1}, x_{n-2} \)
  \[ \longrightarrow \text{multistep method} \]

- result of \( \int \) has the form
  \[ y_{n+1} = y_n + h(\beta_0 f(x_{n+1}, y_{n+1}) + \beta_1 f(x_n, y_n) + \beta_2 f(x_{n-1}, y_{n-1}) + \cdots) \]
Predictor-Corrector Methods

- $\beta_0 = 0 \longrightarrow$ explicit, otherwise implicit
- how to solve?
- $\longrightarrow$ Newton’s method $\longrightarrow$ multistep solver for stiff problems
Predictor-Corrector Methods

- functional iteration
- predictor-corrector methods
- idea: use explicit method to get estimate $y_{n+1}$
- predictor step
- next: use estimate to correct $y_{n+1}$
- corrector step
- can be iterated
- if number of iterations is fixed at the beginning
- explicit method (why?)
Adams-Bashforth-Moulton method

- Overall good stability (explicit!)
- 3rd order version:
  - Predictor step: Adams-Bashforth
    \[
    y_{n+1} = y_n + \frac{h}{12} \left( 23y_n' - 16y_{n-1}' + 5y_{n-2}' \right) + O(h^4)
    \]
  - Corrector step: Adams-Moulton
    \[
    y_{n+1} = y_n + \frac{h}{12} \left( 5y_{n+1}' + 8y_n' - y_{n-1}' \right) + O(h^4)
    \]
predictor-corrector methods

- have starting/stopping problems (use RK)
- stepsize control hard
- → use canned routines whenever possible
- historically very frequently used
- very good for very smooth functions that are expensive to evaluate
My favorite method

- Hammings Modified Predictor–Corrector Method
  - Adaptive Stepsize control
  - Written at IBM in the 1960s
  - 4th order method
  - Uses RK4 to start