PHY407: Computational Physics
Fall, 2016

Instructor: Paul Kushner

TAs: Miriam Diamond and Heather Fong

Marker: Chad Gu

Lecture 7: Ordinary differential equations, Part 2

Summary & Status

- ☑ Weeks 1-3: Programming basics, numerical errors, numerical integration and differntiation.
- ☑ Weeks 4-5: Solving linear & nonlinear systems and Fourier transforms.
- ☑ Week 6: ODEs Part 1: RK4, Leapfrog, Verlet, adaptive time stepping; customizing python output
- ❑ Week 7: ODEs Part 2: Bulirsch-Stoer, Boundary Value Problems/shooting
- ❑ Weeks 8-9: PDEs Parts 1 & 2
- ❑ Weeks 10-11: Random numbers & Monte Carlo methods

PHY407: Computational Physics
Fall, 2015

Instructor: Paul Kushner

TAs: Miriam Diamond and Heather Fong

Marker: Chad Gu

Lecture 7: Ordinary differential equations, Part 2

- Burlisch-Stoer – fast leaps to a conservative solution
- Boundary value problems using shooting
- Brief look at stability of numerical methods

Solving ODEs

- Last week we introduced new methods for solving systems of ODEs

One dimension: $\frac{dx}{dt} = f(x,t)$, given $x(t=0) = x_0$

Multiple dimensions: $\frac{dx_i}{dt} = f_i(x_1,\ldots,x_n,t),\ x_i(t=0) = x_{io},\ i = 1,\ldots,n$

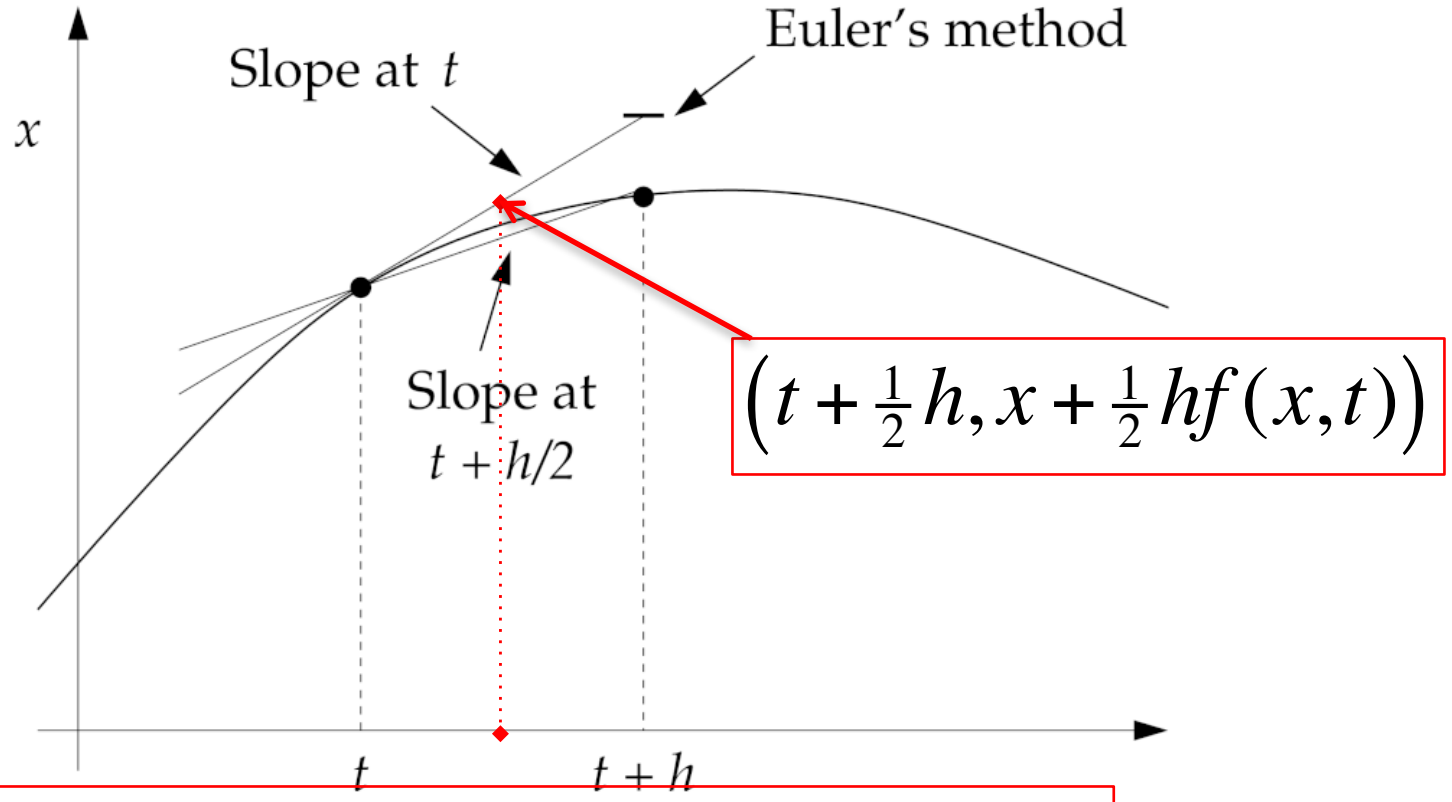
Higher order: e.g. $\frac{d^3x}{dt^3} = g(x,t) \rightarrow$

$$\frac{dx}{dt} = v$$

$$\frac{dv}{dt} = a$$

$$\frac{da}{dt} = g$$

Runge-Kutte Methods



E.g. second order Runge-Kutte:

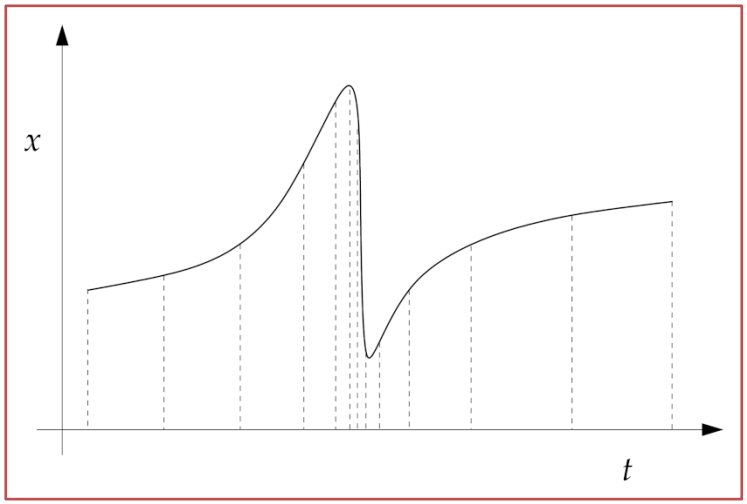
$$x(t+h) = x(t) + hf\left(x + \tfrac{1}{2}hf(x,t), t + \tfrac{1}{2}h\right)$$

Adaptive RK4

- It's worth investing computer time in error estimation as in

$$\rho = \frac{h\delta}{\varepsilon} = \frac{30h\delta}{|x_1 - x_2|} > 1,$$

- Then you can adjust h up or down as required.



Leapfrog and Verlet: Time Reversible

$$x\left(t+h\right)=x\left(t\right)+hf\left(x\left(t+\tfrac{1}{2}h\right),t+\tfrac{1}{2}h\right)$$

$$x\left(t+\tfrac{3}{2}h\right)=x\left(t+\tfrac{1}{2}h\right)+hf\left(x\left(t+h\right),t+h\right)$$

$$x\left(t+\tfrac{1}{2}h\right)=x\left(t+\tfrac{3}{2}h\right)-hf\left(x\left(t+h\right),t+h\right)$$

$$x\left(t\right)=x\left(t+h\right)-hf\left(x\left(t+\tfrac{1}{2}h\right),t+\tfrac{1}{2}h\right)$$

- Leapfrog gives same answer going forward or backward: reversible and energy conserving.
- Verlet Method: special case of leapfrog for Newton's second law, also reversible.

Leapfrog and Verlet: Time Reversible

$$x\left(t+h\right)=x\left(t\right)+hf\left(x\left(t+\tfrac{1}{2}h\right),t+\tfrac{1}{2}h\right)$$

$$x\left(t+\tfrac{3}{2}h\right)=x\left(t+\tfrac{1}{2}h\right)+hf\left(x\left(t+h\right),t+h\right)$$

$$x\left(t+\tfrac{1}{2}h\right)=x\left(t+\tfrac{3}{2}h\right)-hf\left(x\left(t+h\right),t+h\right)$$

$$x\left(t\right)=x\left(t+h\right)-hf\left(x\left(t+\tfrac{1}{2}h\right),t+\tfrac{1}{2}h\right)$$

- Leapfrog gives same answer going forward or backward: reversible and energy conserving.
- Verlet Method: special case of leapfrog for Newton's second law, also reversible.
- New point: Because leapfrog is time reversal symmetric, ε is an odd power of h:

$$\varepsilon(-h)=-\varepsilon\left(h\right)$$

- Cumulative error over integration interval is *even* in h.
- Put another way: this means that we can potentially get two orders of accuracy in h for each improvement.

$$
\begin{aligned}
x_0 &= x\left(t\right) \\
y_1 &= x_0 + \tfrac{1}{2} hf\left(x_0, t\right) \\
x_1 &= x_0 + hf\left(y_1, t + \tfrac{1}{2} h\right) \\
y_2 &= y_1 + hf\left(x_1, t + h\right) \\
&\cdots \\
y_{m+1} &= y_m + hf\left(x_m, t + mh\right) \\
x_{m+1} &= x_m + hf\left(y_{m+1}, t + \left(m + \tfrac{1}{2}\right) h\right) \\
&\cdots \\
y_n &= y_{n-1} + hf\left(x_{n-1}, t + H - h\right) \\
x_n &= x_{n-1} + hf\left(y_n, t + H - \tfrac{1}{2} h\right) \approx x\left(t + H\right)
\end{aligned}
$$

This is new notation for leapfrog over the interval from t to t+H

$$x_0 = x(t)$$

$$y_1 = x_0 + \tfrac{1}{2} hf\left(x_0, t\right)$$

$$x_1 = x_0 + hf\left(y_1, t + \tfrac{1}{2} h\right)$$

$$y_2 = y_1 + hf\left(x_1, t + h\right)$$

$$\cdots$$

$$y_{m+1} = y_m + hf\left(x_m, t + mh\right)$$

$$x_{m+1} = x_m + hf\left(y_{m+1}, t + \left(m + \tfrac{1}{2}\right) h\right)$$

$$\cdots$$

$$y_n = y_{n-1} + hf\left(x_{n-1}, t + H - h\right)$$

$$x_n = x_{n-1} + hf\left(y_n, t + H - \tfrac{1}{2} h\right) \approx x\left(t + H\right)$$

This is new notation for leapfrog over the interval from t to t+H

Leapfrog starts with an Euler half step.

$$x_0 = x(t)$$

$$y_1 = x_0 + \tfrac{1}{2} hf(x_0, t)$$

$$x_1 = x_0 + hf(y_1, t + \tfrac{1}{2} h)$$

$$y_2 = y_1 + hf(x_1, t + h)$$

$$\cdots$$

$$y_{m+1} = y_m + hf(x_m, t + mh)$$

$$x_{m+1} = x_m + hf(y_{m+1}, t + (m + \tfrac{1}{2})h)$$

$$\cdots$$

$$y_n = y_{n-1} + hf(x_{n-1}, t + H - h)$$

$$x_n = x_{n-1} + hf(y_n, t + H - \tfrac{1}{2} h) \approx x(t + H)$$

This is new notation for leapfrog over the interval from t to t+H

Leapfrog starts with an Euler half step.

$$x_0 = x(t)$$

$$y_1 = x_0 + \tfrac{1}{2} h f(x_0, t)$$

This reduces the accuracy by an order of magnitude!

$$x_1 = x_0 + h f(y_1, t + \tfrac{1}{2} h)$$

$$y_2 = y_1 + h f(x_1, t + h)$$

$$\dots$$

$$y_{m+1} = y_m + h f(x_m, t + mh)$$

$$x_{m+1} = x_m + h f(y_{m+1}, t + (m + \tfrac{1}{2}) h)$$

$$\dots$$

$$y_n = y_{n-1} + h f(x_{n-1}, t + H - h)$$

$$x_n = x_{n-1} + h f(y_n, t + H - \tfrac{1}{2} h) \approx x(t + H)$$

This is new notation for leapfrog over the interval from t to t+H

Leapfrog starts with an Euler half step.

$$x_0 = x(t)$$

$$y_1 = x_0 + \tfrac{1}{2} hf(x_0, t)$$

This reduces the accuracy by an order of magnitude!

$$x_1 = x_0 + hf(y_1, t + \tfrac{1}{2} h)$$

$$y_2 = y_1 + hf(x_1, t + h)$$

$$\dots$$

$$y_{m+1} = y_m + hf(x_m, t + mh)$$

$$x_{m+1} = x_m + hf(y_{m+1}, t + (m + \tfrac{1}{2})h)$$

$$\dots$$

But if we do an adjustment at the end ….

$$y_n = y_{n-1} + hf(x_{n-1}, t + H - h)$$

$$x_n = x_{n-1} + hf(y_n, t + H - \tfrac{1}{2} h) \approx x(t + H)$$

This is new notation for leapfrog over the interval from t to t+H

Leapfrog starts with an Euler half step.

This reduces the accuracy by an order of magnitude!

$$x_0 = x(t)$$

$$y_1 = x_0 + \tfrac{1}{2} h f(x_0, t)$$

$$x_1 = x_0 + h f(y_1, t + \tfrac{1}{2} h)$$

$$y_2 = y_1 + h f(x_1, t + h)$$

$$\dots$$

$$y_{m+1} = y_m + h f(x_m, t + mh)$$

$$x_{m+1} = x_m + h f(y_{m+1}, t + (m + \tfrac{1}{2}) h)$$

$$\dots$$

But if we do an adjustment at the end ….

$$y_n = y_{n-1} + h f(x_{n-1}, t + H - h)$$

$$x_n = x_{n-1} + h f(y_n, t + H - \tfrac{1}{2} h) \approx x(t + H)$$

$$x_n' = y_n + \tfrac{1}{2} h f(x_n, t + H)$$

$$x(t + H)_{\text{final}} = \tfrac{1}{2}[x_n + x_n']$$

This is new notation for leapfrog over the interval from t to t+H

Leapfrog starts with an Euler half step.

$$x_0 = x(t)$$

$$y_1 = x_0 + \tfrac{1}{2} h f(x_0, t)$$

This reduces the accuracy by an order of magnitude!

$$x_1 = x_0 + h f(y_1, t + \tfrac{1}{2} h)$$

$$y_2 = y_1 + h f(x_1, t + h)$$

$$\dots$$

$$y_{m+1} = y_m + h f(x_m, t + mh)$$

$$x_{m+1} = x_m + h f(y_{m+1}, t + (m + \tfrac{1}{2}) h)$$

$$\dots$$

But if we do an adjustment at the end ….

$$y_n = y_{n-1} + h f(x_{n-1}, t + H - h)$$

$$x_n = x_{n-1} + h f(y_n, t + H - \tfrac{1}{2} h) \approx x(t + H)$$

… This *modified midpoint* method restores the accuracy (Gragg).

$$x_n' = y_n + \tfrac{1}{2} h f(x_n, t + H)$$

$$x(t + H)_{\text{final}} = \tfrac{1}{2}[x_n + x_n']$$

This is what a single MMP step of length H looks like.

$$x_0 = x(t)$$

$$y_1 = x_0 + \tfrac{1}{2} H f(x_0, t)$$

$$x_1 = x_0 + H f(y_1, t + \tfrac{1}{2} H)$$

$$x_1' = y_1 + \tfrac{1}{2} H f(x_1, t + H)$$

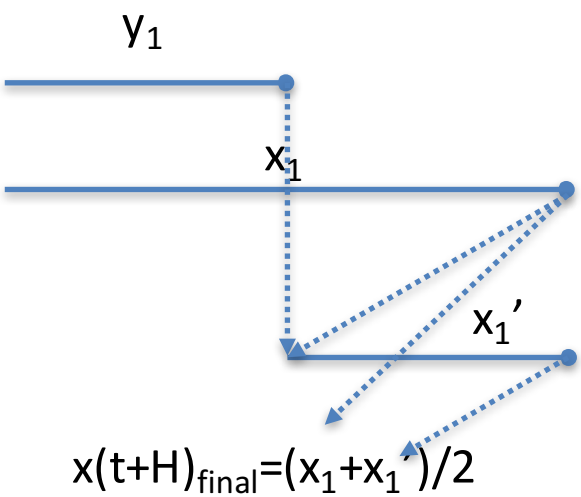
$$x(t+H)_{\text{final}} = \tfrac{1}{2}[x_1 + x_1'] = R_{1,1}$$

$$x(t+H) = R_{1,1} + c_1 H^2 + O(H^4)$$

$$= R_{2,1} + c_1 (H/2)^2 + O\left((H/2)^4\right)$$

$$\text{So } x(t+H) = R_{2,1} + \frac{1}{3}(R_{2,1} - R_{1,1}) + O\left((H/2)^4\right)$$

This is what a single MMP step of length H looks like.



$$x_0 = x(t)$$

$$y_1 = x_0 + \tfrac{1}{2} Hf\left(x_0, t\right)$$

$$x_1 = x_0 + Hf\left(y_1, t + \tfrac{1}{2} H\right)$$

$$x_1' = y_1 + \tfrac{1}{2} Hf\left(x_1, t + H\right)$$

$$x(t+H)_{\text{final}} = \tfrac{1}{2}\left[x_1 + x_1'\right] = R_{1,1}$$

$$x(t+H) = R_{1,1} + c_1 H^2 + O(H^4)$$

$$= R_{2,1} + c_1 \left(H/2\right)^2 + O\left(\left(H/2\right)^4\right)$$

$$\text{So } x(t+H) = R_{2,1} + \frac{1}{3}\left(R_{2,1} - R_{1,1}\right) + O\left(\left(H/2\right)^4\right)$$

This is what a single MMP step of length H looks like.

Let's save the result as $R_{1,1}$. The error on this calculation is $O(H^2)$.

$$x_0 = x(t)$$

$$y_1 = x_0 + \tfrac{1}{2} H f(x_0, t)$$

$$x_1 = x_0 + H f(y_1, t + \tfrac{1}{2} H)$$

$$x_1' = y_1 + \tfrac{1}{2} H f(x_1, t + H)$$

$$x(t+H)_{\text{final}} = \tfrac{1}{2}[x_1 + x_1'] = R_{1,1}$$

$$x(t+H) = R_{1,1} + c_1 H^2 + O(H^4)$$

$$= R_{2,1} + c_1 (H/2)^2 + O\left((H/2)^4\right)$$

$$\text{So } x(t+H) = R_{2,1} + \frac{1}{3}(R_{2,1} - R_{1,1}) + O\left((H/2)^4\right)$$

This is what a single MMP step of length H looks like.

Let's save the result as $R_{1,1}$. The error on this calculation is $O(H^2)$.

Let's do an MMP in two steps of H/2, and save $R_{2,1}$.

$$x_0 = x(t)$$

$$y_1 = x_0 + \tfrac{1}{2} Hf\left(x_0, t\right)$$

$$x_1 = x_0 + Hf\left(y_1, t + \tfrac{1}{2} H\right)$$

$$x_1' = y_1 + \tfrac{1}{2} Hf\left(x_1, t + H\right)$$

$$x(t+H)_{\text{final}} = \tfrac{1}{2}\left[x_1 + x_1'\right] = R_{1,1}$$

$$x(t+H) = R_{1,1} + c_1 H^2 + O(H^4)$$

$$= R_{2,1} + c_1 \left(H/2\right)^2 + O\left(\left(H/2\right)^4\right)$$

$$\text{So } x(t+H) = R_{2,1} + \frac{1}{3}\left(R_{2,1} - R_{1,1}\right) + O\left(\left(H/2\right)^4\right)$$

This is what a single MMP step of length H looks like.

Let's save the result as $R_{1,1}$. The error on this calculation is $O(H^2)$.

Let's do an MMP in two steps of H/2, and save $R_{2,1}$.

Now combine the two to improve the estimate to be fourth order in H/2, instead of second order in H!

This improvement is called *Richardson extrapolation*.

$$x_0 = x(t)$$

$$y_1 = x_0 + \tfrac{1}{2} H f(x_0, t)$$

$$x_1 = x_0 + H f(y_1, t + \tfrac{1}{2} H)$$

$$x_1' = y_1 + \tfrac{1}{2} H f(x_1, t + H)$$

$$x(t+H)_{\text{final}} = \tfrac{1}{2}[x_1 + x_1'] = R_{1,1}$$

$$x(t+H) = R_{1,1} + c_1 H^2 + O(H^4)$$

$$= R_{2,1} + c_1 (H/2)^2 + O\left((H/2)^4\right)$$

$$\text{So } x(t+H) = \underbrace{R_{2,1} + \frac{1}{3}(R_{2,1} - R_{1,1})}_{R_{2,2}} + O\left((H/2)^4\right)$$

This is what a single MMP step of length H looks like.

Let's save the result as $R_{1,1}$. The error on this calculation is $O(H^2)$.

Let's do an MMP in two steps of H/2, and save $R_{2,1}$.

Now combine the two to improve the estimate to be fourth order in H/2, instead of second order in H! $R_{2,1}+(R_{2,1}-R_{1,1})/3=R_{2,2}$

This improvement is called *Richardson extrapolation*.

$$x_0 = x(t)$$

$$y_1 = x_0 + \tfrac{1}{2} H f\left(x_0, t\right)$$

$$x_1 = x_0 + H f\left(y_1, t + \tfrac{1}{2} H\right)$$

$$x_1' = y_1 + \tfrac{1}{2} H f\left(x_1, t + H\right)$$

$$x(t+H)_{\text{final}} = \tfrac{1}{2}\left[x_1 + x_1'\right] = R_{1,1}$$

$$x(t+H) = R_{1,1} + c_1 H^2 + O(H^4)$$

$$= R_{2,1} + c_1 \left(H/2\right)^2 + O\left(\left(H/2\right)^4\right)$$

$$\text{So } x(t+H) = \underbrace{R_{2,1} + \frac{1}{3}\left(R_{2,1} - R_{1,1}\right)}_{R_{2,2}} + O\left(\left(H/2\right)^4\right)$$

$$R_{3,2} = R_{3,1} + \tfrac{4}{5}\left(R_{3,1} - R_{2,1}\right)$$

$$R_{3,3} = R_{3,2} + \tfrac{16}{65}\left(R_{3,2} - R_{2,2}\right)$$

$$x(t+H) = R_{3,3} + O\left(\left(H/3\right)^6\right)$$

This is what a single MMP step of length H looks like.

Let's save the result as $R_{1,1}$. The error on this calculation is $O(H^2)$.

Let's do an MMP in two steps of H/2, and save $R_{2,1}$.

Now combine the two to improve the estimate to be fourth order in H/2, instead of second order in H! $R_{2,1}+(R_{2,1}-R_{1,1})/3=R_{2,2}$

This improvement is called *Richardson extrapolation*.

This procedure can be continued to another MMP with H/3, etc.

$$x_0 = x(t)$$

$$y_1 = x_0 + \tfrac{1}{2} Hf\left(x_0, t\right)$$

$$x_1 = x_0 + Hf\left(y_1, t + \tfrac{1}{2} H\right)$$

$$x_1' = y_1 + \tfrac{1}{2} Hf\left(x_1, t + H\right)$$

$$x(t+H)_{\text{final}} = \tfrac{1}{2}\left[x_1 + x_1'\right] = R_{1,1}$$

$$x(t+H) = R_{1,1} + c_1 H^2 + O(H^4)$$

$$= R_{2,1} + c_1 \left(H/2\right)^2 + O\left(\left(H/2\right)^4\right)$$

$$\text{So } x(t+H) = \underbrace{R_{2,1} + \frac{1}{3}\left(R_{2,1} - R_{1,1}\right)}_{R_{2,2}} + O\left(\left(H/2\right)^4\right)$$

$$R_{3,2} = R_{3,1} + \tfrac{4}{5}\left(R_{3,1} - R_{2,1}\right)$$

$$R_{3,3} = R_{3,2} + \tfrac{16}{65}\left(R_{3,2} - R_{2,2}\right)$$

$$x(t+H) = R_{3,3} + O\left(\left(H/3\right)^6\right)$$

This is what a single MMP step of length H looks like.

Let's save the result as $R_{1,1}$. The error on this calculation is $O(H^2)$.

Let's do an MMP in two steps of H/2, and save $R_{2,1}$.

Now combine the two to improve the estimate to be fourth order in H/2, instead of second order in H! $R_{2,1}+(R_{2,1}-R_{1,1})/3=R_{2,2}$

This improvement is called *Richardson extrapolation*.

This procedure can be continued to another MMP with H/3, etc.

$$x_0 = x(t)$$

$$y_1 = x_0 + \tfrac{1}{2} H f(x_0, t)$$

$$x_1 = x_0 + H f(y_1, t + \tfrac{1}{2} H)$$

$$x_1' = y_1 + \tfrac{1}{2} H f(x_1, t + H)$$

$$x(t+H)_{\text{final}} = \tfrac{1}{2}[x_1 + x_1'] = R_{1,1}$$

$$x(t+H) = R_{1,1} + c_1 H^2 + O(H^4)$$

$$= R_{2,1} + c_1 (H/2)^2 + O\left((H/2)^4\right)$$

$$\text{So } x(t+H) = R_{2,1} + \frac{1}{3}(R_{2,1} - R_{1,1}) + O\left((H/2)^4\right)$$

$$R_{3,2} = R_{3,1} + \tfrac{4}{5}(R_{3,1} - R_{2,1})$$

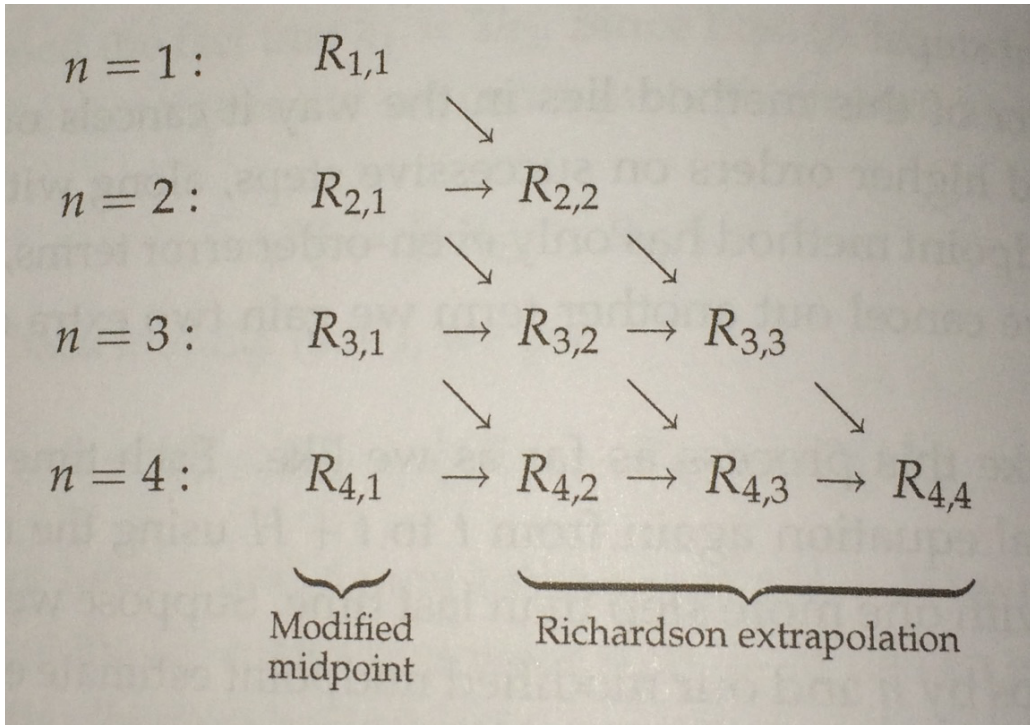
$$R_{3,3} = R_{3,2} + \tfrac{16}{65}(R_{3,2} - R_{2,2})$$

$$x(t+H) = R_{3,3} + O\left((H/3)^6\right)$$

$$\ldots$$

$$x(t+H) = R_{n,m+1} + O\left((H/n)^{2m+2}\right)$$

Bulirsch-Stoer



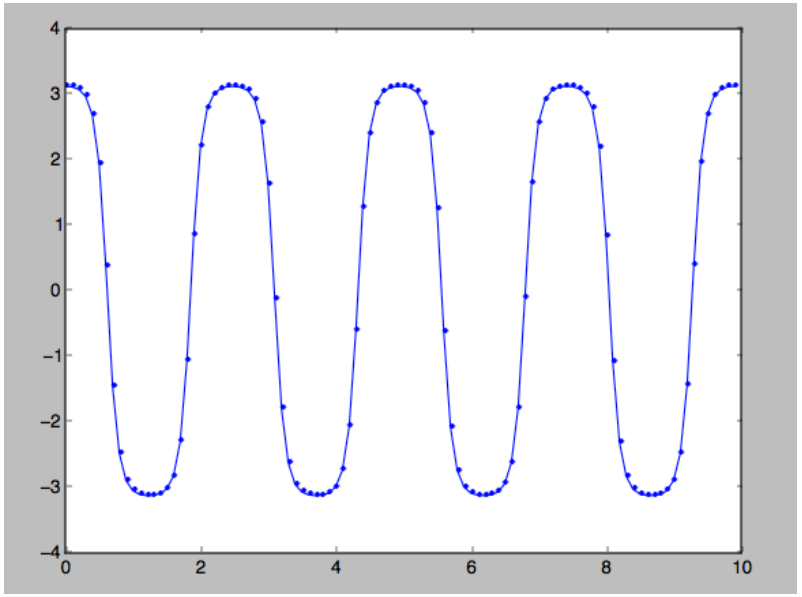
- Iterate over finer intervals to solve over a large “hop” in a conservative and efficient way.
- Your ODES need to be relatively well behaved for this to work well.

Hint for Lab

- To get the provided script `bulirsch.py` to work you need to stick in a line:

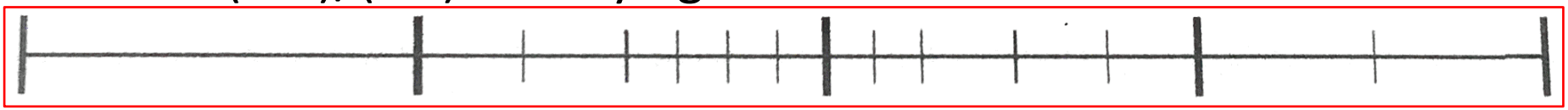
```
from __future__ import division
```

- Let's look at this script.
- Its output is at right.



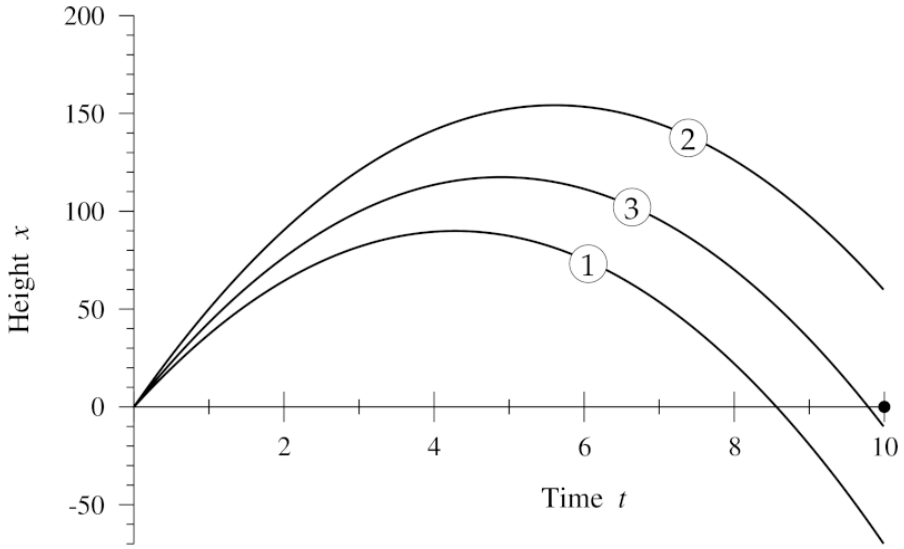
Choosing the Bulirsch-Stoer Timestep

- B-S works well if the number of modified midpoint steps is reasonable (8-10).
- Adaptive time stepping helps ensure this, for a given desired accuracy. Here's the approach:
- Choose an N. For each subinterval of size h=(b-a)/N
  - Calculate error up to 8 or so steps (Lab07Q3).
  - If we haven't met our accuracy goal, subdivide this interval to (b-a)/(2N) and try again on each subinterval.



- Although there are apparently wasted calculations, this focuses accuracy where it is necessary.

Shooting Method for Solving BVPs

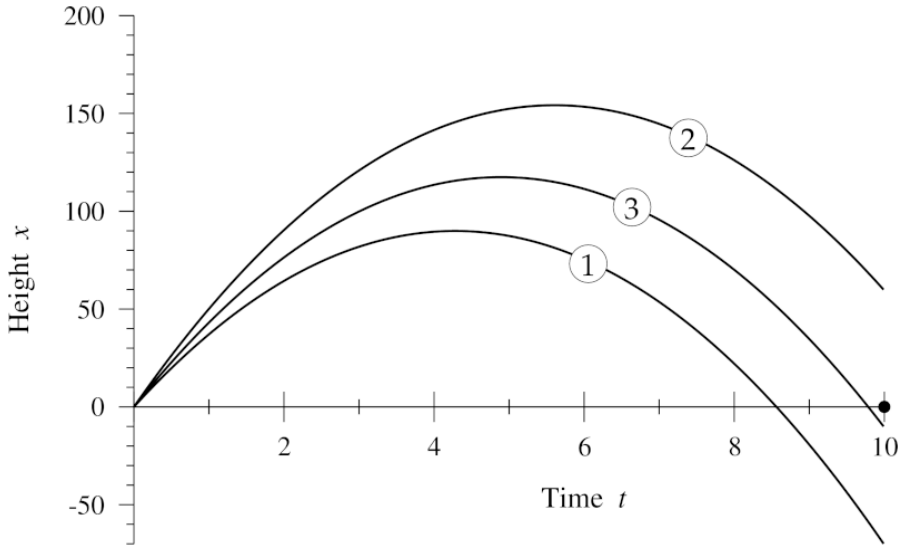


$$\frac{d^2x}{dt^2} = -g,$$

$$x(t=0) = 0,\ v(t=0) = v_0$$

- Suppose we wanted to choose an initial velocity for a projectile so it could land on the ground after t=10s.

Shooting Method for Solving BVPs

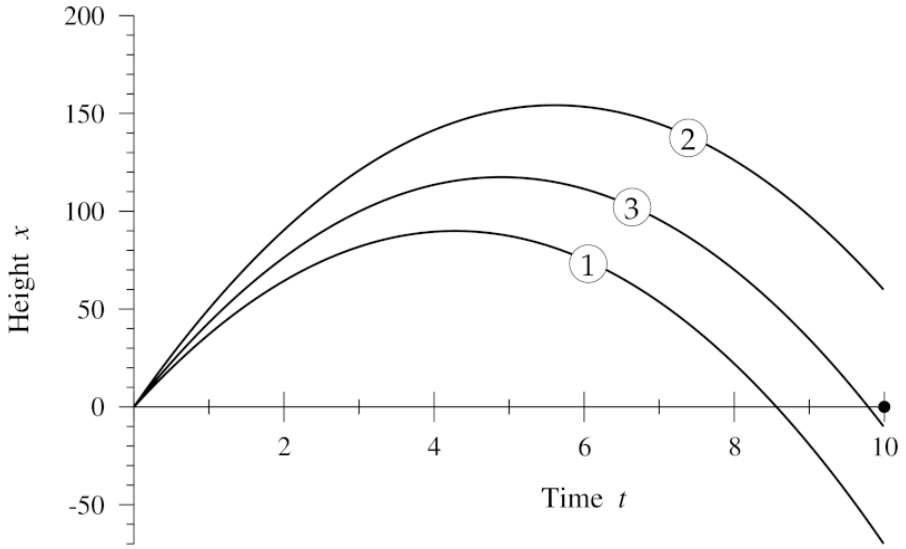


$$\frac{d^2x}{dt^2} = -g,$$

$$x(t=0) = 0,\ v(t=0) = v_0$$

- Suppose we wanted to choose an initial velocity for a projectile so it could land on the ground after t=10s.
- $x(v_0,t)$ is a nonlinear function of v_0, and finding $x(v_0,10s)=0$ can be done using a root finding method.

Shooting Method for Solving BVPs

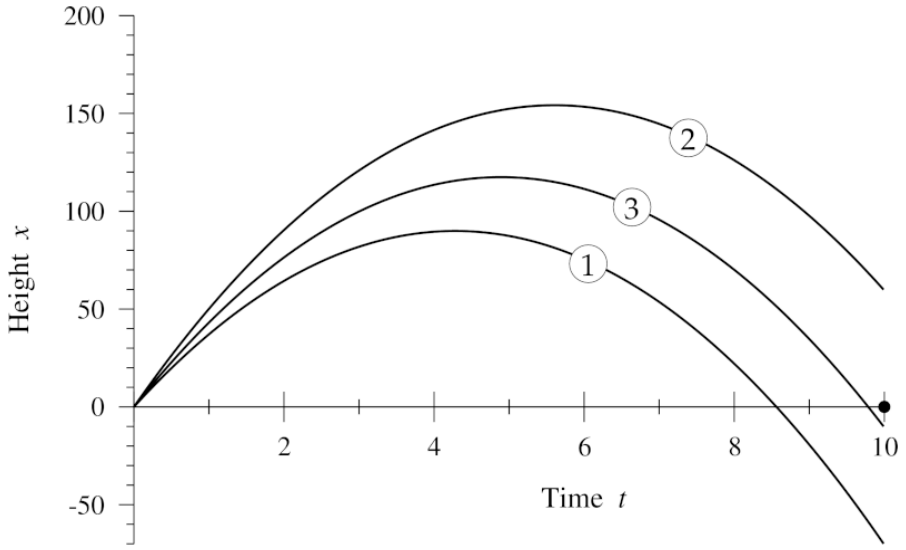


$$\frac{d^2x}{dt^2} = -g,$$

$$x(t=0) = 0,\ v(t=0) = v_0$$

- Suppose we wanted to choose an initial velocity for a projectile so it could land on the ground after t=10s.
- $x(v_0,t)$ is a nonlinear function of v_0, and finding $x(v_0,10s)=0$ can be done using a root finding method.
- *Shooting method:* integrate the equations and adjust v_0 until root located.

Shooting Method for Solving BVPs



$$\frac{d^2x}{dt^2} = -g,$$

$$x(t=0) = 0,\ v(t=0) = v_0$$

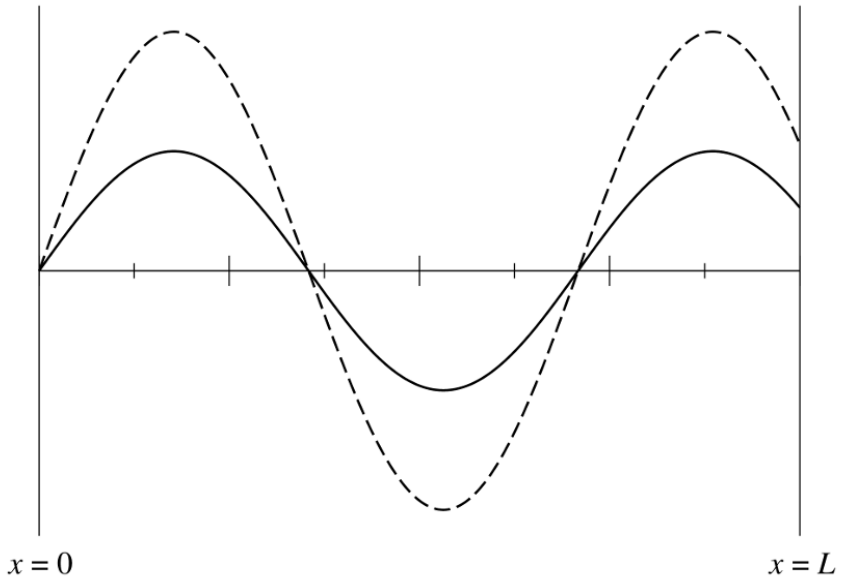
Let's look at `throw.py`

- Suppose we wanted to choose an initial velocity for a projectile so it could land on the ground after t=10s.
- $x(v_0,t)$ is a nonlinear function of v_0, and finding $x(v_0,10s)=0$ can be done using a root finding method.
- *Shooting method:* integrate the equations and adjust v_0 until root located.

Shooting Method for Solving BVPs

$$-\frac{\hbar^2}{2m}\frac{d^2\psi}{dx^2}+V(x)\psi=E\psi,$$

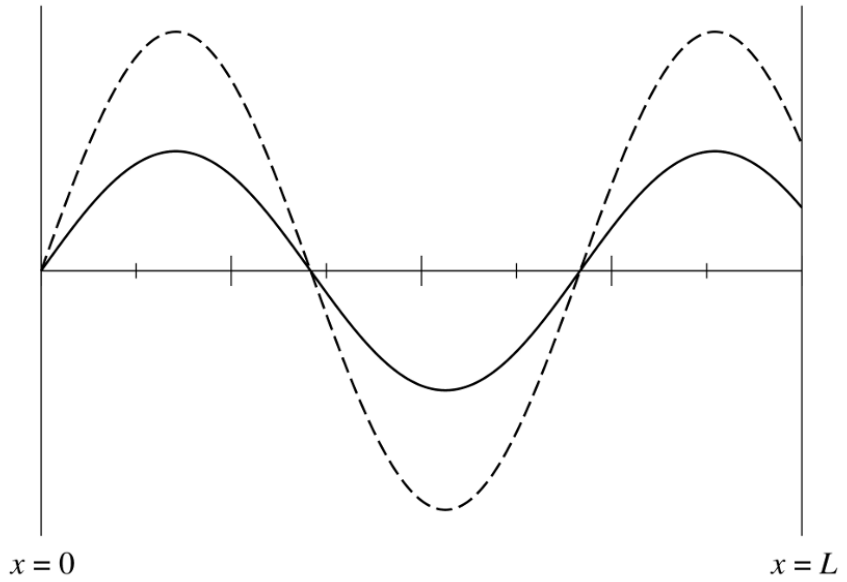
$$\psi(x=0)=\psi(x=L)=0$$



Shooting Method for Solving BVPs

$$-\frac{\hbar^2}{2m}\frac{d^2\psi}{dx^2}+V(x)\psi=E\psi,$$

$$\psi(x=0)=\psi(x=L)=0$$

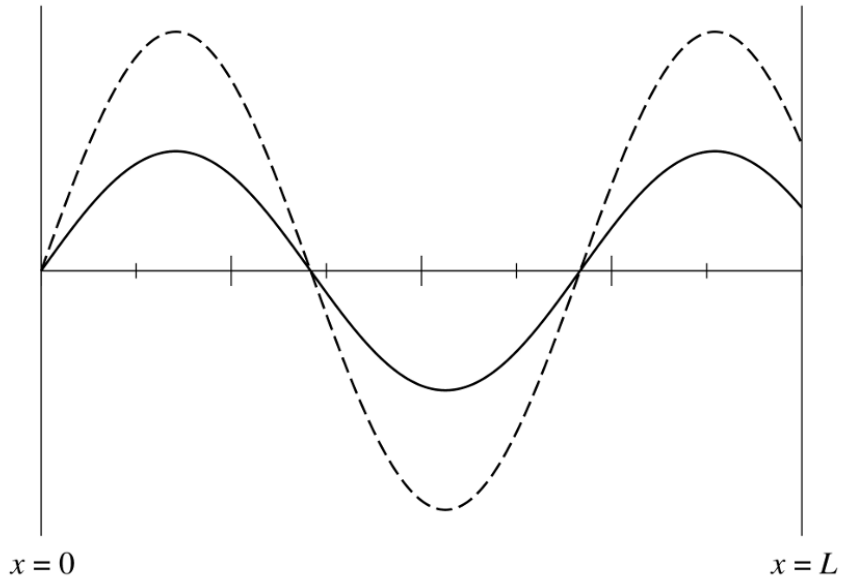


- But this approach does not work for finding wavefunctions that satisfy two boundary conditions, as in QM square well, except for valid eigenvalues E.

Shooting Method for Solving BVPs

$$-\frac{\hbar^2}{2m}\frac{d^2\psi}{dx^2}+V(x)\psi=E\psi,$$

$$\psi(x=0)=\psi(x=L)=0$$



- But this approach does not work for finding wavefunctions that satisfy two boundary conditions, as in QM square well, except for valid eigenvalues E.
- So for these problems, E is the parameter that must be varied instead of the leftmost slope of wavefunction.

Functional and Numerical Stability

- We have focused on accuracy and speed in investigating our solutions to ODEs.
- But **stability** is also important. The stability of solutions tells us how fast initially close solutions diverge from each other.
- Some systems are inherently unstable and so will always be challenging to simulate.
- We will investigate this in some exercises in the coming labs.

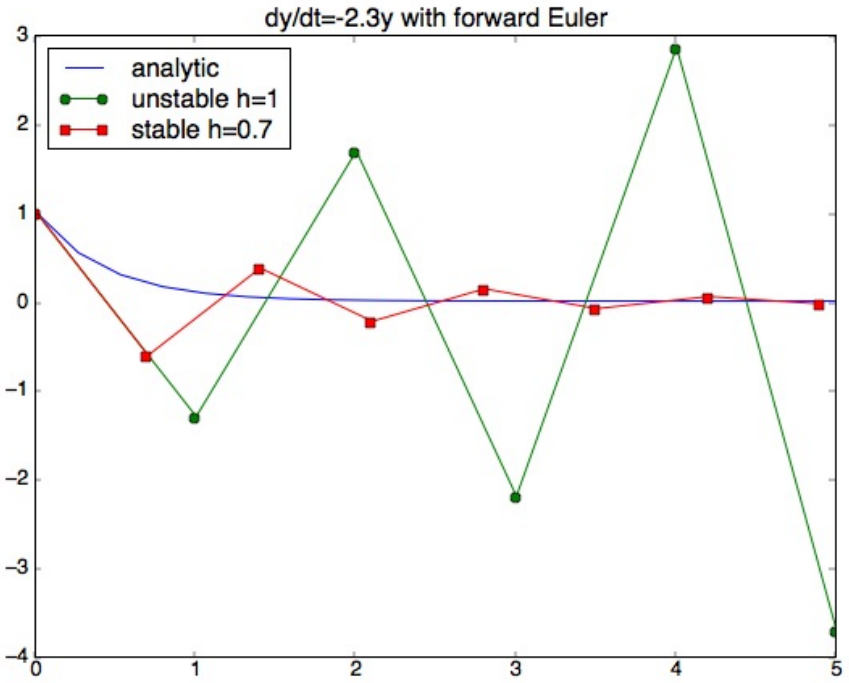
Method Stability

- Stability or instability of a system can be determined from small perturbations to a solution of the ODE.
- But even for stable systems, numerical methods can be unstable and give truncation errors that grow.

Method Stability

- Stability or instability of a system can be determined from small perturbations to a solution of the ODE.
- But even for stable systems, numerical methods can be unstable and give truncation errors that grow.

- Example: dy/dt = -2.3 y is a stable system.
- Forward Euler unstable with h =1 but stable with h=0.7
- Need to determine stability of METHODS. Typically this depends on the ODE and the step size



Method Stability

- Why is Forward Euler unstable for this system?

Method Stability

- Why is Forward Euler unstable for this system?
- Explicitly write the solution: for each timestep

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

Method Stability

- Why is Forward Euler unstable for this system?
- Explicitly write the solution: for each timestep

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

- And for k timesteps

$$y_k = (1 + h_k \lambda)^k y_0$$

Method Stability

- Why is Forward Euler unstable for this system?
- Explicitly write the solution: for each timestep

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

- And for k timesteps

$$y_k = (1 + h_k \lambda)^k y_0$$

- For the method to be stable, the magnitude of the *growth factor*

$$|1 + h_k \lambda| \leq 1$$

$$\Rightarrow \lambda < 0, h_k \leq |2 / \lambda|$$

Method Stability

- Why is Forward Euler unstable for this system?
- Explicitly write the solution: for each timestep

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

- And for k timesteps

$$y_k = (1 + h_k \lambda)^k y_0$$

- For the method to be stable, the magnitude of the *growth factor*

$$|1 + h_k \lambda| \leq 1$$

$$\Rightarrow \lambda < 0, \; h_k \leq \left| 2 / \lambda \right|$$

This is consistent with numerically determined results.

PHY407: Computational Physics
Fall, 2015

Lecture 7: Ordinary differential equations, Part 2

- Burlisch-Stoer – fast leaps to a conservative solution
- Boundary value problems using shooting
- Brief look at stability of numerical methods