计算物理

Lecture 13

傅子文

fuziwen@scu.edu.cn

Numerical solutions of ordinary differential equations (ODEs)

- Runge-Kutta
- Adaptive step size selection

Modified Euler Method ("extrapolate-integrate")

- The Euler method is flawed because we use the beginning of the interval to predict to the end
- If we could use the midpoint derivative that would usually be expected to be a better approximation
  - If the slope is changing rapidly across an interval using the midpoint slope usually gives us something closer to the average over the interval (of course it doesn't always have to be better, just on average)
- However since $y'_{mid} = f(x_{mid}, y_{mid})$ and we don't know y_{mid} (only y_0) & we have to estimate it

Modified Euler Method

- Let $y'(x) = f(x, y)$ and $y(x_0)=y_0$
- Let the step size be given by $h=x_1-x_0$, $f_0=f(x_0,y_0)$, $x_{1/2}=x_0+h/2$, $x_{3/2}=x_1+h/2$ *etc.*

Define $\tilde{y}_{1/2} = y_0 + \frac{h}{2} f_0$ *i.e.* extrapolate using Euler method to $x_{1/2}$

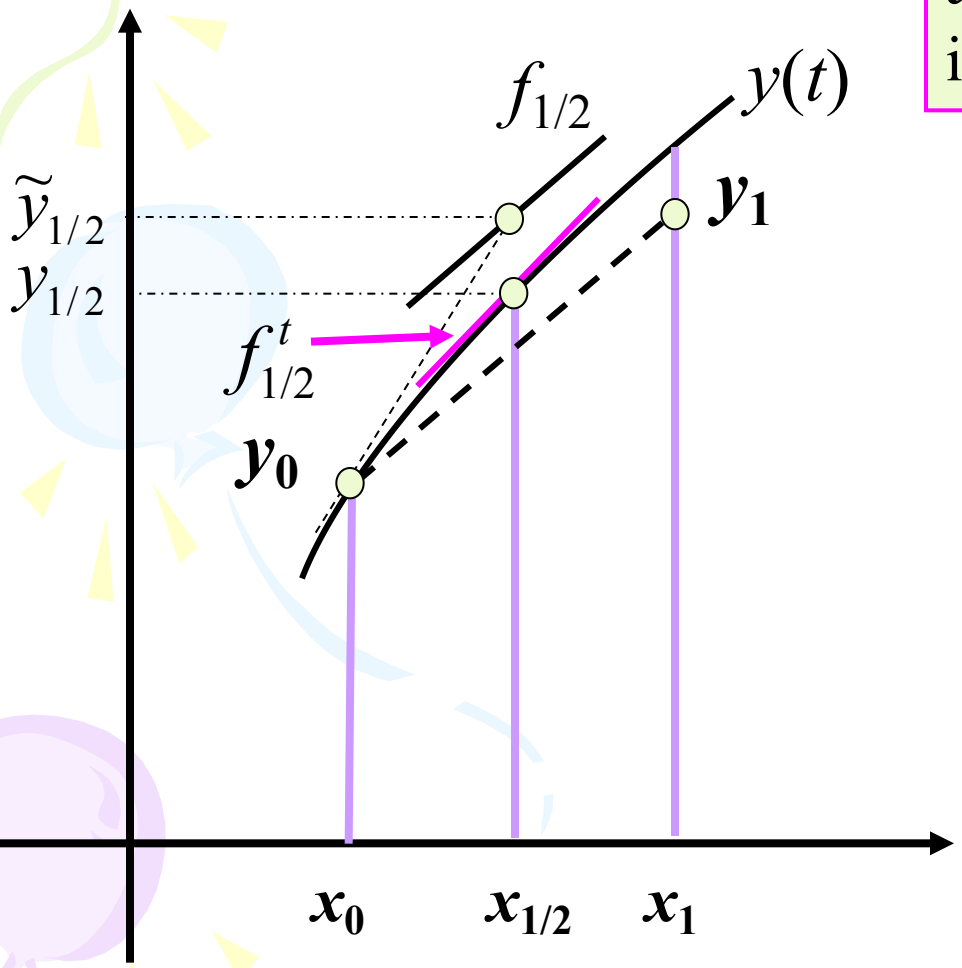
Then set $y_1 = y_0 + hf(x_{1/2},\tilde{y}_{1/2}) \equiv y_0 + hf_{1/2}$ *i.e.* integrate full step

Repeat extrapolation on half step :

Define $\tilde{y}_{3/2} = y_1 + \frac{h}{2} f(x_1, y_1) \equiv y_1 + \frac{h}{2} f_1$

Repeat integration : $y_2 = y_1 + hf(x_{3/2},\tilde{y}_{3/2}) \equiv y_1 + hf_{3/2}$ *etc...*

Modified Euler Method graphically



In this method the slope is estimated at the mid-point and the $y(x)$ is integrated forward using that slope.

Notice that at the predicted $\tilde{y}_{1/2}$ the slope $f_{1/2}$ is not necessarily equal to the true derivative ($f^t_{1/2}$) at the true y midpoint.
This is simply because $\tilde{y}_{1/2}$ does not have to be equal to $y_{1/2}$.
Even if we did know $f^t_{1/2}$ we would still not predict forward with perfect accuracy.

Can show that errors in this method are

$$e_i \propto O(h^3) \quad \& \quad E_n \propto O(h^2)$$

Improved Euler Method

- Similar in concept to the trapezoid rule in quadrature, by utilizing an average of the **start and end** point derivatives we can better approximate the change in y over the interval

$$\tilde{y}_1 = y_0 + hf(x_0, y_0), \quad \tilde{f}_1 = f(x_1, \tilde{y}_1) \quad \text{Prediction step}$$

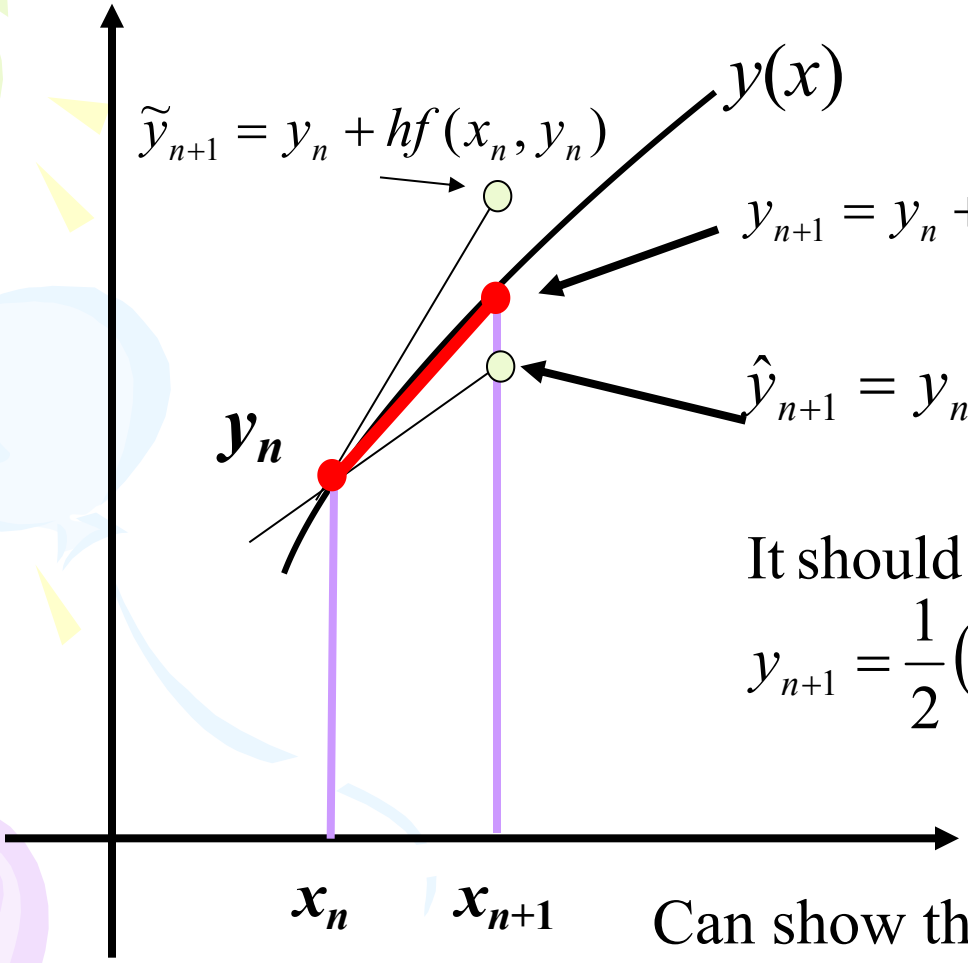
$$y_1 = y_0 + \frac{h}{2}\left(f_0 + \tilde{f}_1\right), \quad f_1 = f(x_1, y_1) \quad \text{Correction step}$$

$$\tilde{y}_2 = y_1 + hf_1, \quad \tilde{f}_2 = f(x_2, \tilde{y}_2) \quad \text{Prediction step}$$

$$y_2 = y_1 + \frac{h}{2}\left(f_1 + \tilde{f}_2\right), \quad f_2 = f(x_2, y_2) \quad \text{Correction step}$$

- This is the simplest example of so called "*Predictor-Corrector*" methods, (which includes Runge-Kutta methods)

Improved Euler method graphically



$$y_{n+1} = y_n + \frac{f(x_n, y_n) + f(x_{n+1}, \tilde{y}_{n+1})}{2} h$$

$$\hat{y}_{n+1} = y_n + f(x_{n+1}, \tilde{y}_{n+1}) h$$

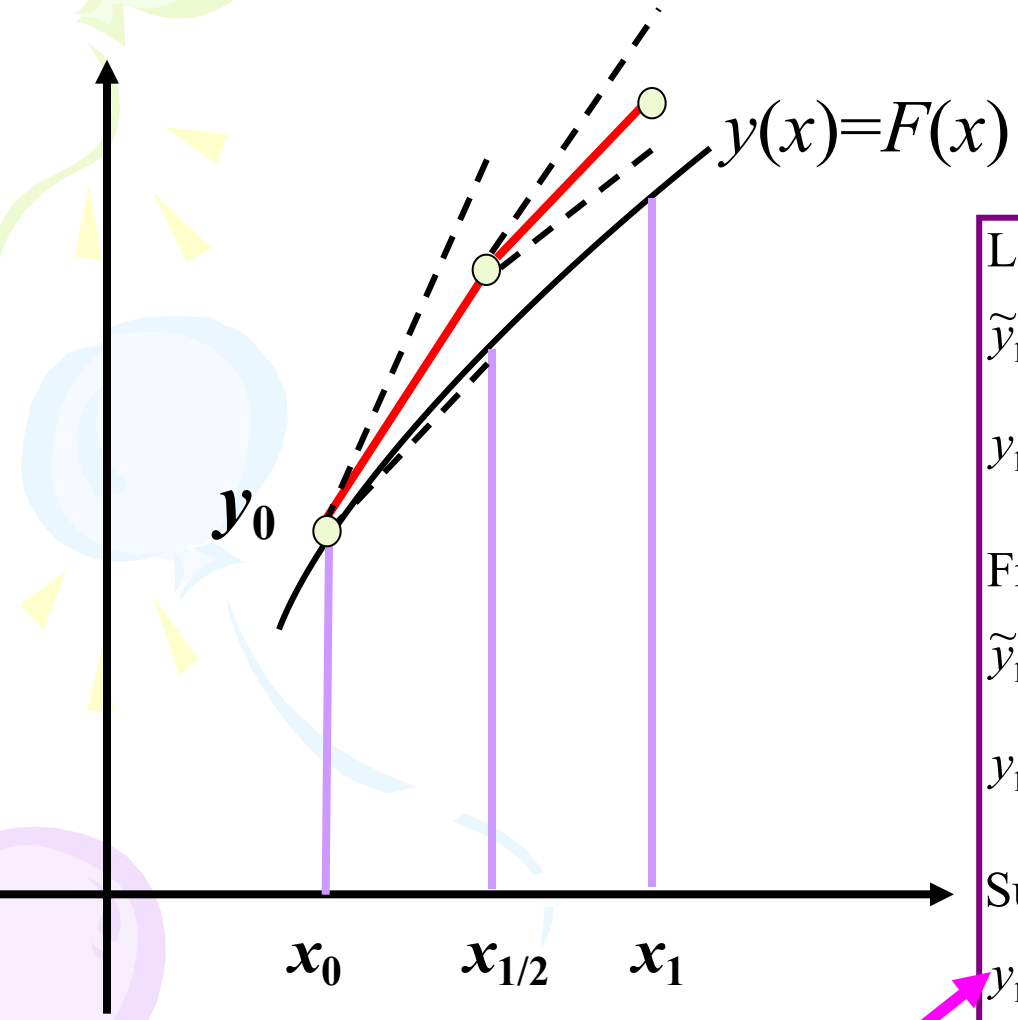
It should be clear that

$$y_{n+1} = \frac{1}{2}(\tilde{y}_{n+1} + \hat{y}_{n+1})$$

Can show that errors in this method are also

$$e_i \propto O(h^3) \quad \mathrm{E_n} \propto O(h^2)$$

Higher order scheme preliminaries



- As a first motivation consider repeating Improved Euler method but with two substeps

Let $x_{1/2} = x_0 + h/2$ then define first substep:

$$\tilde{y}_{1/2} = y_0 + \frac{h}{2} f(x_0, y_0)$$

$$y_{1/2} = y_0 + \frac{h}{2}\frac{1}{2}\left(f(x_0, y_0) + f(x_{1/2}, \tilde{y}_{1/2})\right) \quad \text{(A)}$$

From $x_{1/2}$ we do the second substep:

$$\tilde{y}_1 = y_{1/2} + \frac{h}{2} f(x_{1/2}, y_{1/2})$$

$$y_1 = y_{1/2} + \frac{h}{2}\frac{1}{2}\left(f(x_{1/2}, y_{1/2}) + f(x_1, \tilde{y}_1)\right) \quad \text{(B)}$$

Substitute for $y_{1/2}$ in (B) using (A) to get

$$y_{1,h/2} = y_0 + \frac{h}{4}\left(f(x_0, y_0) + f(x_{1/2}, \tilde{y}_{1/2}) + f(x_{1/2}, y_{1/2}) + f(x_1, \tilde{y}_1)\right) \quad \text{(C)}$$

Indicates y_1 was obtained with two $h/2$ steps

General points about higher order methods

- Higher order accurate schemes will use more substep evaluations
- We need to ensure these are chosen optimally so as to produce the most accuracy for the least number of function evaluations
- Let's go back to the Improved Euler method on a series of steps and apply Richardson-Extrapolation approach

4th order Runge-Kutta

- For a single Improved Euler method we get

$$y_{1,h} = y_0 + \frac{f(x_0, y_0) + f(x_1, \tilde{y}_1)}{2} h \qquad \text{(D)}$$

- A *global* evaluation with n steps will produce an error E_n, of order $O(h^2)$

- If we halve the size of h then the global error produced by (D) with the smaller step size must be ¼ that of the previous evaluation

- That must correspond to the error produced by using the formula $y_{1,h/2}$ (*i.e.* C) over the n steps

Subtraction step – defining

- Since we know the global error term produced by using $y_{1,h/2}$
  $\mathrm{Err}(y_{1,h/2})=\mathrm{K}(h/2)^2=\mathrm{K}h^2/4$ (E)

- Global error term for $y_{1,h}$ $\mathrm{Err}(y_{1,h})=\mathrm{K}h^2$ (F)

- Subtract (F) from 4×(E) & define new y_1:

$$3y_1 \equiv 4y_{1,h/2} - y_{1,h}$$

$$\Rightarrow y_1 = \frac{4y_{1,h/2} - y_{1,h}}{3}$$ which we can write out

$$\Rightarrow y_1 = y_0 + \frac{h}{6}\left(f_0 + 2\tilde{f}_{1/2} + 2f_{1/2} + \tilde{f}_1\right)$$

This formula will actually be globally 4th order accurate!

Fourth Order Runge Kutta

The following is sometimes called the classical fourth-order RK method

$$y_{i+1} = y_i + \left[\frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)\right]h$$

where

$$k_1 = f(x_i, y_i)$$

$$k_2 = f\left(x_i + \frac{1}{2}h, y_i + \frac{1}{2}k_1 h\right)$$

$$k_3 = f\left(x_i + \frac{1}{2}h, y_i + \frac{1}{2}hk_2\right)$$

$$k_4 = f(x_i + h, y_i + hk_3)$$

The most popular $O(h^5)$ truncation error, round-off error?

Fourth Order Runge Kutta

Note that for ODE that are a function of x alone that this is also the equivalent of Simpson's 1/3 Rule

$$y_{i+1} = y_i + \left[\frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)\right]h$$

where

$$k_1 = f(x_i, y_i)$$

$$k_2 = f\left(x_i + \frac{1}{2}h, y_i + \frac{1}{2}k_1 h\right)$$

$$k_3 = f\left(x_i + \frac{1}{2}h, y_i + \frac{1}{2}hk_2\right)$$

$$k_4 = f(x_i + h, y_i + hk_3)$$

Runge-Kutta Methods

- **Runge-Kutta** (RK) methods achieve the accuracy of a Taylor series approach without requiring the calculation of a higher derivative

- Many variations exist but all can be cast in the generalized form:

$$y_{i+1} = y_i + \phi(x_i, y_i, h)h$$

ϕ is called the incremental function

ϕ, Incremental Function

can be interpreted as a representative slope over the interval

$$\phi = a_1 k_1 + a_2 k_2 + \cdots + a_n k_n$$

where the a's are constant and the k's are:

$$k_1 = f(x_i, y_i)$$

$$k_2 = f(x_i + p_1 h, y_i + q_{11} k_1 h)$$

$$k_3 = f(x_i + p_2 h, y_i + q_{21} k_1 h + q_{22} k_2 h)$$

$$\vdots$$

$$k_n = f(x_i + p_n h, y_i + q_{n-1,1} k_1 h + q_{n-1,2} k_2 h + \cdots + q_{n-1,n-1} k_{n-1} h)$$

p's and q's are constant

$$\phi = a_1 k_1 + a_2 k_2 + \cdots + a_n k_n$$

where the a's are constant and the k's are:

$$k_1 = f(x_i, y_i)$$

$$k_2 = f(x_i + p_1 h, y_i + q_{11} k_1 h)$$

$$k_3 = f(x_i + p_2 h, y_i + q_{21} k_1 h + q_{22} k_2 h)$$

$$\vdots$$

$$k_n = f(x_i + p_n h, y_i + q_{n-1,1} k_1 h + q_{n-1,2} k_2 h + \cdots + q_{n-1,n-1} k_{n-1} h)$$

NOTE:

k's are recurrence relationships,
that is k_1 appears in the equation for k_2
which appears in the equation for k_3,.....
This recurrence makes RK methods efficient for computer calculations

Runge-Kutta Methods

Various types of RK methods can be devised by employing different number of terms in the increment function as specified by n.

First order RK method with $n=1$ is **Euler's method**.

- **Error is proportional to O(h)**

Second Order RK Methods

$$y_{i+1} = y_i + (a_1 k_1 + a_2 k_2)h$$

where

$$k_1 = f(x_i, y_i)$$

$$k_2 = f(x_i + p_1 h, y_i + q_{11} k_1 h)$$

$\phi = a_1 k_1 + a_2 k_2 + \cdots + a_n k_n$

where the a's are constant and the k's are:

$$k_1 = f(x_i, y_i)$$

$$k_2 = f(x_i + p_1 h, y_i + q_{11} k_1 h)$$

$$k_3 = f(x_i + p_2 h, y_i + q_{21} k_1 h + q_{22} k_2 h)$$

$$\vdots$$

$$k_n = f(x_i + p_n h, y_i + q_{n-1,1} k_1 h + q_{n-1,2} k_2 h + \cdots + q_{n-1,n-1} k_{n-1} h)$$

Second Order RK Methods

- We have to determine values for the constants a_1, a_2, p_1 and q_{11}
- To do this consider the Taylor series in terms of y_{i+1} and $f(x_i,y_i)$

$$y_{i+1} = y_i + (a_1 k_1 + a_2 k_2)h$$

$$y_{i+1} = y_i + f(x_i, y_i)h + f'(x_i, y_i)\frac{h^2}{2}$$

Second Order RK Methods

Now, $f'(x_i, y_i)$ must be determined by the chain rule for differentiation

$$f'(x_i, y_i) = \frac{\partial f}{\partial x} + \frac{\partial f}{\partial y}\frac{dy}{dx}$$

substituting

$$y_{i+1} = y_i + f(x_i, y_i)h + \left(\frac{\partial f}{\partial x} + \frac{\partial f}{\partial y}\frac{dy}{dx}\right)\frac{h^2}{2}$$

The basic strategy underlying Runge-Kutta methods is to use algebraic manipulations to solve for values of a_1, a_2, p_1 and q_{11}

Values of a_1, a_2, p_1, and q_{11} are evaluated by setting the second order equation to Taylor series expansion to the second order term.

Second Order RK Methods

$$y_{i+1} = y_i + (a_1 k_1 + a_2 k_2) h$$

$$y_{i+1} = y_i + f(x_i, y_i) h + \left(\frac{\partial f}{\partial x} + \frac{\partial f}{\partial y} \frac{dy}{dx} \right) \frac{h^2}{2}$$

By setting these two equations equal to each other and recalling:

$$k_1 = f(x_i, y_i)$$

$$k_2 = f(x_i + p_1 h, y_i + q_{11} k_1 h)$$

we derive three equations to evaluate the four unknown constants

Second Order RK Methods

$$\left.\begin{aligned} a_1 + a_2 &= 1 \\ a_2 p_1 &= \frac{1}{2} \\ a_2 q_{11} &= \frac{1}{2} \end{aligned}\right\}$$

Because we have three equations with four unknowns,
we must assume a value of one of the unknowns.

Suppose we specify a value for a_2.

What would the equations be?

Second Order RK Methods

$$a_1 = 1 - a_2$$

$$p_1 = q_{11} = \frac{1}{2a_2}$$

- Because we can choose an infinite number of values for a_2 there are an infinite number of second order RK methods.

- Every solution would yield exactly the same result if the solution to the ODE were quadratic, linear or a constant.

- Lets review three of the most commonly used and preferred versions.

Second Order RK Methods

$$y_{i+1} = y_i + (a_1 k_1 + a_2 k_2)h$$

where

$$k_1 = f(x_i, y_i)$$

$$k_2 = f(x_i + p_1 h, y_i + q_{11} k_1 h)$$

$$a_1 + a_2 = 1$$

$$a_2 p_1 = \frac{1}{2}$$

$$a_2 q_{11} = \frac{1}{2}$$

Consider the following:

Case 1: $a_2 = 1/2$

Case 2: $a_2 = 1$

These two methods have been previously studied.

What are they?

Second Order RK Methods

$$a_1 = 1 - a_2 = 1 - 1/2 = 1/2$$

$$a_2 p_1 = \frac{1}{2}$$

$$a_2 q_{11} = \frac{1}{2}$$

$$p_1 = q_{11} = \frac{1}{2a_2} = 1$$

$$y_{i+1} = y_i + \left(\frac{1}{2}k_1 + \frac{1}{2}k_2\right)h$$

where

$$k_1 = f(x_i, y_i)$$

$$k_2 = f(x_i + h, y_i + k_1 h)$$

Case 1: $a_2 = 1/2$

This is Heun's Method with a single corrector.

Note that k_1 is the slope at the beginning of the interval and k_2 is the slope at the end of the interval.

$$y_{i+1} = y_i + (a_1 k_1 + a_2 k_2)h$$

where

$$k_1 = f(x_i, y_i)$$

$$k_2 = f(x_i + p_1 h, y_i + q_{11} k_1 h)$$

Second Order RK Methods

$$a_1 = 1 - a_2 = 1 - 1 = 0$$

$$a_2 p_1 = \frac{1}{2}$$

$$a_2 q_{11} = \frac{1}{2}$$

$$p_1 = q_{11} = \frac{1}{2a_2} = \frac{1}{2}$$

$$y_{i+1} = y_i + k_2 h$$

where

$$k_1 = f(x_i, y_i)$$

$$k_2 = f\left(x_i + \frac{1}{2}h, y_i + \frac{1}{2}k_1 h\right)$$

Case 2: $a_2 = 1$

This is the Improved Polygon Method.

$$y_{i+1} = y_i + (a_1 k_1 + a_2 k_2)h$$

where

$$k_1 = f(x_i, y_i)$$

$$k_2 = f(x_i + p_1 h, y_i + q_{11} k_1 h)$$

Ralston's Method

Ralston (1962) and Ralston and Rabinowitiz (1978) determined that choosing $a_2 = 2/3$ provides a minimum bound on the truncation error for the second order RK algorithms.

This results in $a_1 = 1/3$ and $p_1 = q_{11} = 3/4$

$$y_{i+1} = y_i + \left(\frac{1}{3}k_1 + \frac{2}{3}k_2\right)h$$

where

$$k_1 = f(x_i, y_i)$$

$$k_2 = f\left(x_i + \frac{3}{4}h, y_i + \frac{3}{4}k_1 h\right)$$

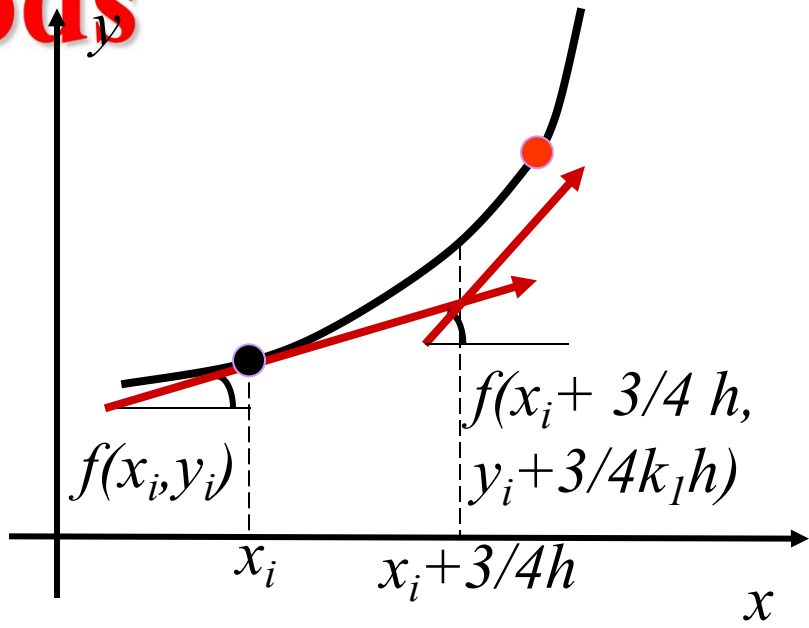
Runge-Kutta Methods

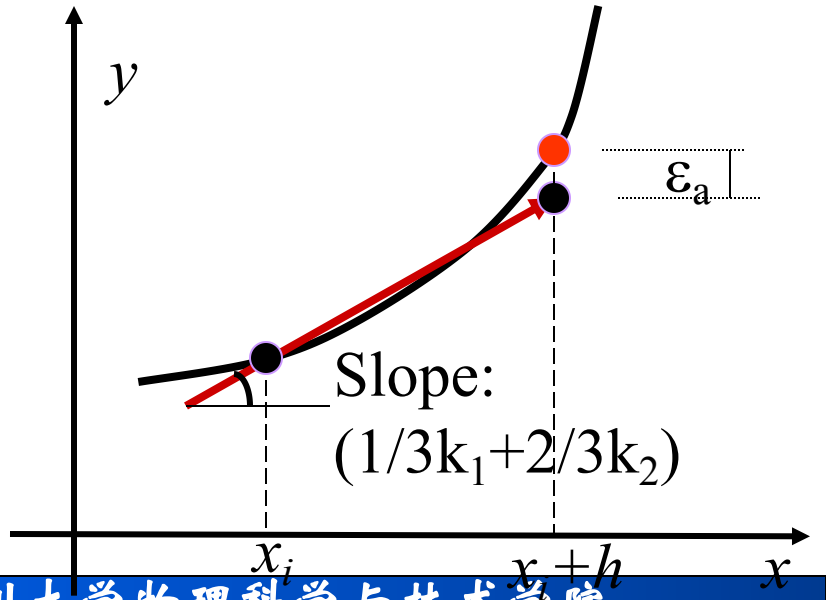
- **Ralston's Method:**

$$y_{i+1} = y_i + (\frac{1}{3}k_1 + \frac{2}{3}k_2)\cdot h$$

$$k_1 = f(x_i, y_i)$$

$$k_2 = f(x_i + \frac{3}{4}h, y_i + \frac{3}{4}k_1 h)$$





Third Order Runge-Kutta Methods

- Derivation is similar to the one for the second-order
- Results in six equations and eight unknowns.
- One common version results in the following

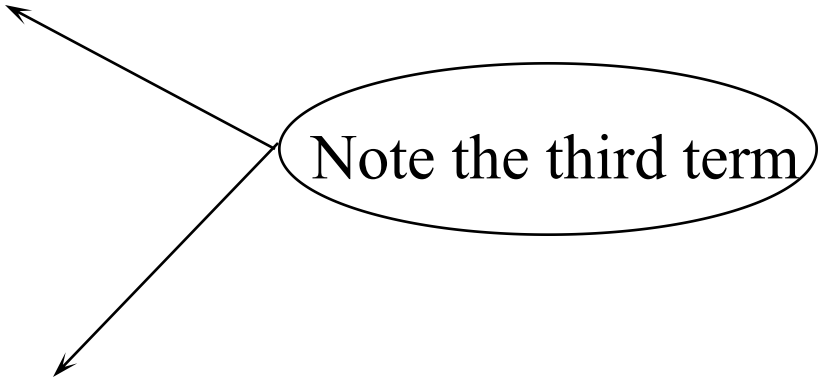
$$y_{i+1} = y_i + \left[\frac{1}{6}(k_1 + 4k_2 + k_3)\right]h$$

where

$$k_1 = f(x_i, y_i)$$

$$k_2 = f\left(x_i + \frac{1}{2}h, y_i + \frac{1}{2}k_1 h\right)$$

$$k_3 = f(x_i + h, y_i - hk_1 + 2hk_2)$$



NOTE: if the derivative is a function of x only, this reduces to Simpson's 1/3 Rule

Fourth Order Runge Kutta

Note that for ODE that are a function of x alone that this is also the equivalent of Simpson's 1/3 Rule

$$y_{i+1} = y_i + \left[\frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)\right]h$$

where

$$k_1 = f(x_i, y_i)$$

$$k_2 = f\left(x_i + \frac{1}{2}h, y_i + \frac{1}{2}k_1 h\right)$$

$$k_3 = f\left(x_i + \frac{1}{2}h, y_i + \frac{1}{2}hk_2\right)$$

$$k_4 = f(x_i + h, y_i + hk_3)$$

Higher Order RK Methods

- When more accurate results are required, Bucher's (1964) fifth order RK method is recommended
- There is a similarity to Boole's Rule
- The gain in accuracy is offset by added computational effort and complexity

Sixth Order RK Methods or higher?

Systems of Equations

- Many practical problems in engineering and science require the solution of a system of simultaneous differential equations

$$\frac{dy_1}{dx} = f_1(x, y_1, y_2, \cdots, y_n)$$

$$\frac{dy_2}{dx} = f_2(x, y_1, y_2, \cdots, y_n)$$

$$\vdots$$

$$\frac{dy_n}{dx} = f_n(x, y_1, y_2, \cdots, y_n)$$

- Solution requires *n* initial conditions
- All the methods for single equations can be used
- The procedure involves applying the one-step technique for every equation at each step before proceeding to the next step

$$\frac{dy_1}{dx} = f_1(x, y_1, y_2, \cdots, y_n)$$
$$\frac{dy_2}{dx} = f_2(x, y_1, y_2, \cdots, y_n)$$
$$\vdots$$
$$\frac{dy_n}{dx} = f_n(x, y_1, y_2, \cdots, y_n)$$

Ordinary Differential Equations

A second order equation includes a second derivative.

- Linear 2nd order ODE

$$\frac{d^2y}{dx^2} + p\frac{dy}{dx} + Qy = f(x)$$

- Non-Linear 2nd order ODE

$$\frac{d^2y}{dx^2} + p(x,y)\frac{dy}{dx} + Q(x,y)\cdot y = f(x)$$

- Higher order equations can be reduced to a system of first order equations, by redefining a variable.

Second order ODEs

- Consider the general second order ODE

$$y''(x) = g(x, y, y')$$

- We now require two initial values be provided, namely the y_0 value and the derivative $y'(x_0)$
  - These are called the Cauchy conditions

- If we let $z=y'$, then $z'=y''$ and we have

$$\left.\begin{aligned} y' &= z \quad ; & y(x_0) &= y_0 \\ z' &= y'' = g(x, y(x), z(x)) \; ; & z(x_0) &= z_0 \end{aligned}\right\} \quad (1)$$

Second order ODEs cont

$$\text{Let } \vec{Y} \equiv \begin{bmatrix} y \\ z \end{bmatrix} \text{ so } \vec{Y}_0 = \begin{bmatrix} y_0 \\ z_0 \end{bmatrix}, \text{ and } \vec{F} \equiv \begin{bmatrix} z \\ g \end{bmatrix}$$

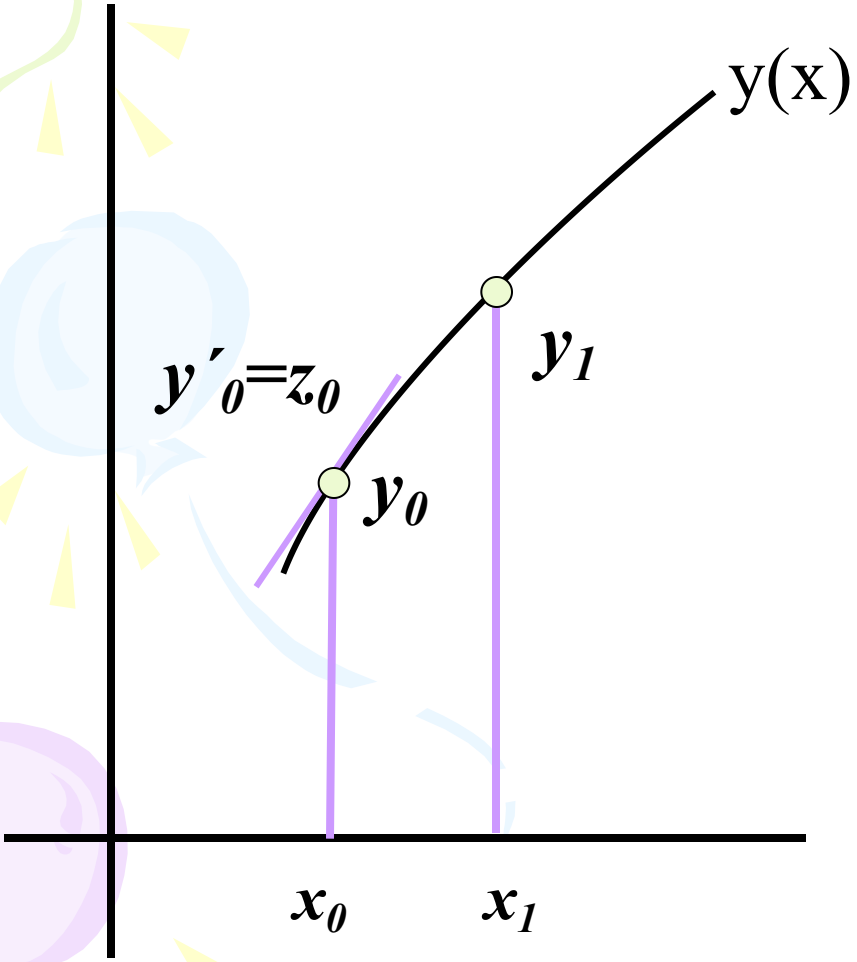
then we can write the system as

$$\begin{bmatrix} y' \\ z' \end{bmatrix} = \vec{F} \Rightarrow \vec{Y}' = \vec{F} \text{ with } \vec{Y}_0 = \begin{bmatrix} y_0 \\ z_0 \end{bmatrix}$$

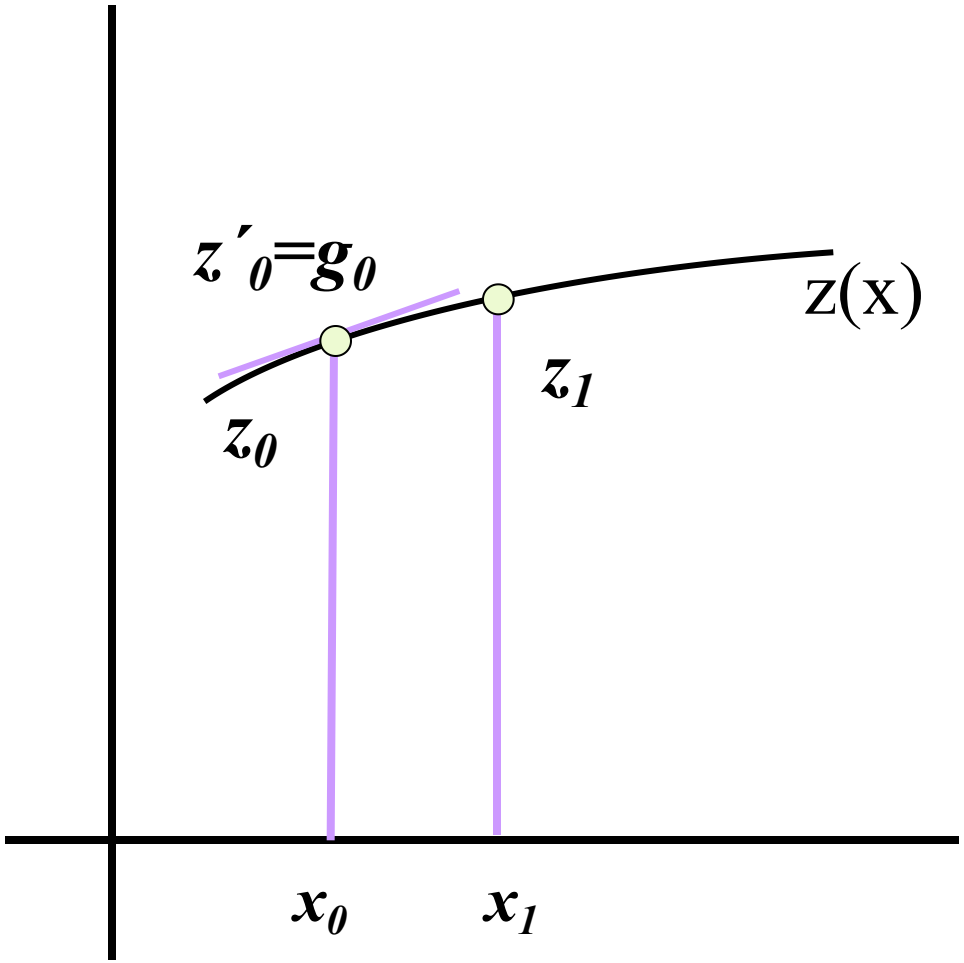
- We have thus turned a second order ODE into a first order ODE for a vector
- Can apply the R-K solver to the system but you now have two components to integrate
- At each step must update all x, y and z values

Diagrammatically

Remember y'=z



Remember z'=g(x,y,y')



Fourth-Order Runge-Kutta using vectorized notation

$$\boldsymbol{r}_{i+1} = \boldsymbol{r}_i + \left[\frac{1}{6}\left(\boldsymbol{k}_1 + 2\boldsymbol{k}_2 + 2\boldsymbol{k}_3 + \boldsymbol{k}_4\right)\right]h$$

where

$$\boldsymbol{r}_{i+1} = \boldsymbol{r}(t_i + h), \qquad \boldsymbol{r}_i = \boldsymbol{r}(t_i), \qquad \frac{d\boldsymbol{r}_i}{dt} = f\left(\boldsymbol{r}_i, t_i\right)$$

$$\boldsymbol{k}_1 = f\left(\boldsymbol{r}_i, t\right)$$

$$\boldsymbol{k}_2 = f\left(\boldsymbol{r}_i + \frac{1}{2}\boldsymbol{k}_1, t_i + \frac{1}{2}h\right)$$

$$\boldsymbol{k}_3 = f\left(\boldsymbol{r}_i + \frac{1}{2}\boldsymbol{k}_2, t_i + \frac{1}{2}h\right)$$

$$\boldsymbol{k}_4 = f\left(\boldsymbol{r}_i + \boldsymbol{k}_3, t_i + h\right)$$

Implementing Runge Kutta

Because of Python's Dynamic Typing and Vectorization, one function works for any dimension!

```
def rk4(f, r, t, h):
    """ 4th order Runge-Kutta method
            …
    """
    k1 = h*f(r,t)
    k2 = h*f(r+0.5*k1,t+0.5*h)
    k3 = h*f(r+0.5*k2,t+0.5*h)
    k4 = h*f(r+k3,t+h)
    return (k1 + 2*k2 + 2*k3 + k4)/6
```

Implementing Runge Kutta

```
def fN(N, t):
    """ 1 dimensional function"""
    tau = 2.0
   return - N/tau

N, NPoints = 1, []
for t in tPoints:
NPoints += [N]
N += rk4(fN, N, t, h)
plt.plot(tPoints, NPoints)
```

Implementing Runge Kutta

```
def f2D(r, t):
    """ 2 dimensional function"""
    g,L = 9.81, 0.1
    theta,omega = r[0],r[1]
    fTheta = omega
    fOmega = -g/L * theta
    return np.array([fTheta,fOmega], float)

r = np.array([10*np.pi/180, 0], float)
for t in tPoints:
thetaPoints += [r[0]]
omegaPoints += [r[1]]
r += rk4(f2D, r, t, h)
plt.plot(tPoints, thetaPoints)
```

The van der Pol oscillator

The van der Pol oscillator appears in electronic circuits and in laser physics

$$\frac{d^2x}{dt^2} - \mu(1-x^2)\frac{dx}{dt} + \omega^2 x = 0$$

Solve with initial conditions x = 1 and dx/dt = 0.

$$f(\boldsymbol{r}, t) = \begin{vmatrix} \frac{dx}{dt} = v \\ \frac{dv}{dt} = \mu(1-x^2)v - \omega^2 x = 0 \end{vmatrix}$$

```
def f(r, t):
    """ vectorized function for the
        van  der Pol oscillator """
    mu,omega = 3.0,1.0 # constants
    x = r[0]
    v = r[1]
    fx = v
    fv = -omega**2 * x + mu * (1 – \
        x**2) * v
    return np.array([fx, fv], float)
```

The van der Pol oscillator

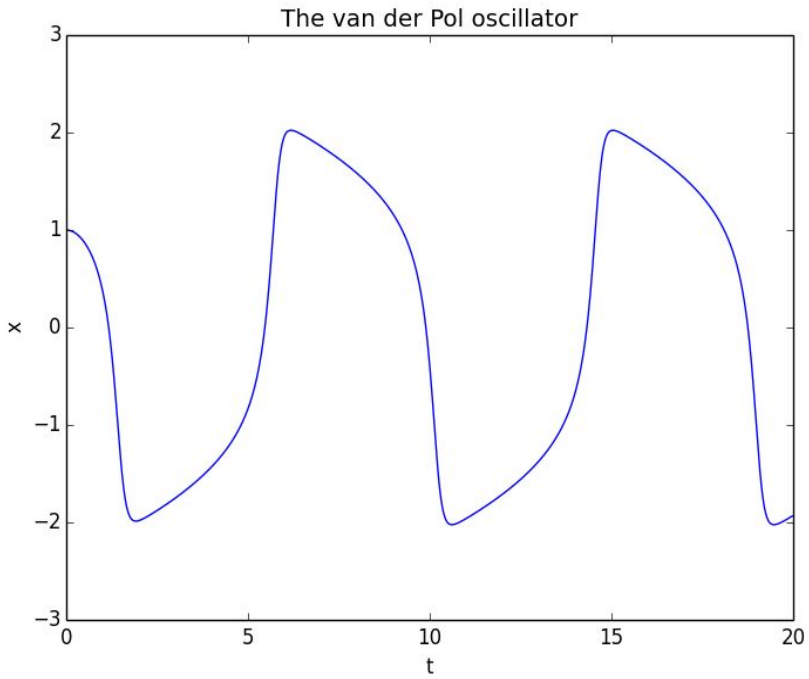
The van der Pol oscillator appears in electronic circuits and in laser physics

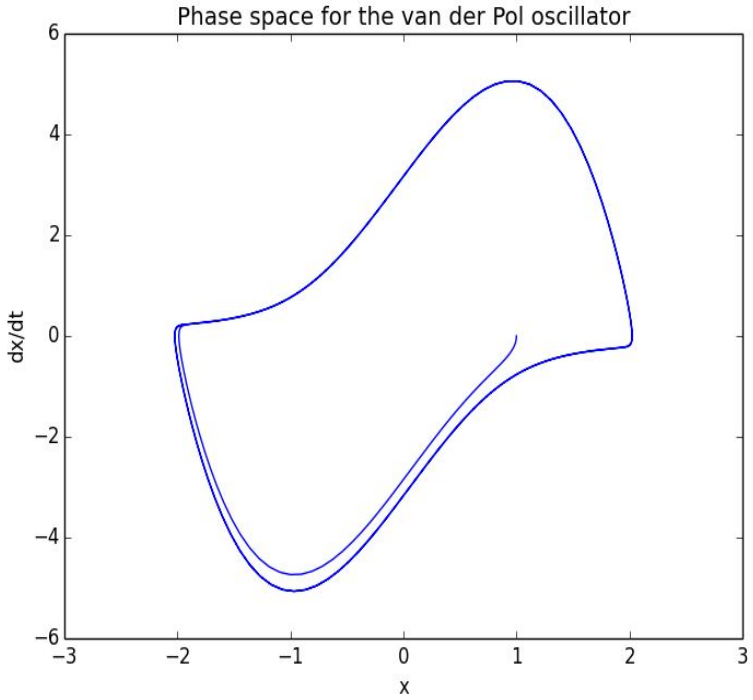
```
#
# main
# initialization
N = 1000 # Number of steps
tMin,tMax = 0.0,20.0 # time range
h = (tMax-tMin)/N # time step
# define time values
tPoints = np.arange(tMin, tMax, h)
# creating lists for plotting
xPoints,vPoints = [],[]
# set initial conditions
x0,v0 = 1.0,0.0
r = np.array([x0,v0], float)
```

```
# solve for the motion
for t in tPoints:
xPoints += [r[0]]
vPoints += [r[1]]
r += rk4(f, r, t, h)
# Now generate plots
...
```

The van der Pol oscillator

The van der Pol oscillator appears in electronic circuits and in laser physics





Comments

- There are an infinite number of ways of choosing slopes and substeps
- Mathematically all these methods are parameterized and limited by using appropriate Taylor expansions and matching coefficients
  - See W. Gear "Numerical Initial Value Problems in ODEs", 1971, pgs 27-36
- However, one must always make an arbitrary choice to derive a specific algorithm
- Many numerical analysts have considered all the possible choices and 4^{th} order Runge-Kutta is considered a very good method

Classical Runge-Kutta method

Equivalent to the five term Taylor expansion

$$y_{n+1} = y_n + hy'_n + \frac{h^2}{2!}y''_n + \frac{h^3}{3!}y'''_n + \frac{h^4}{4!}y^{iv}_n$$

$$f_0 = f(x_0, y_0)$$

$$\tilde{f}_{1/2} = f(x_0 + h/2, y_0 + hf_0/2)$$

$$f_{1/2} = f(x_0 + h/2, y_0 + h\tilde{f}_{1/2}/2)$$

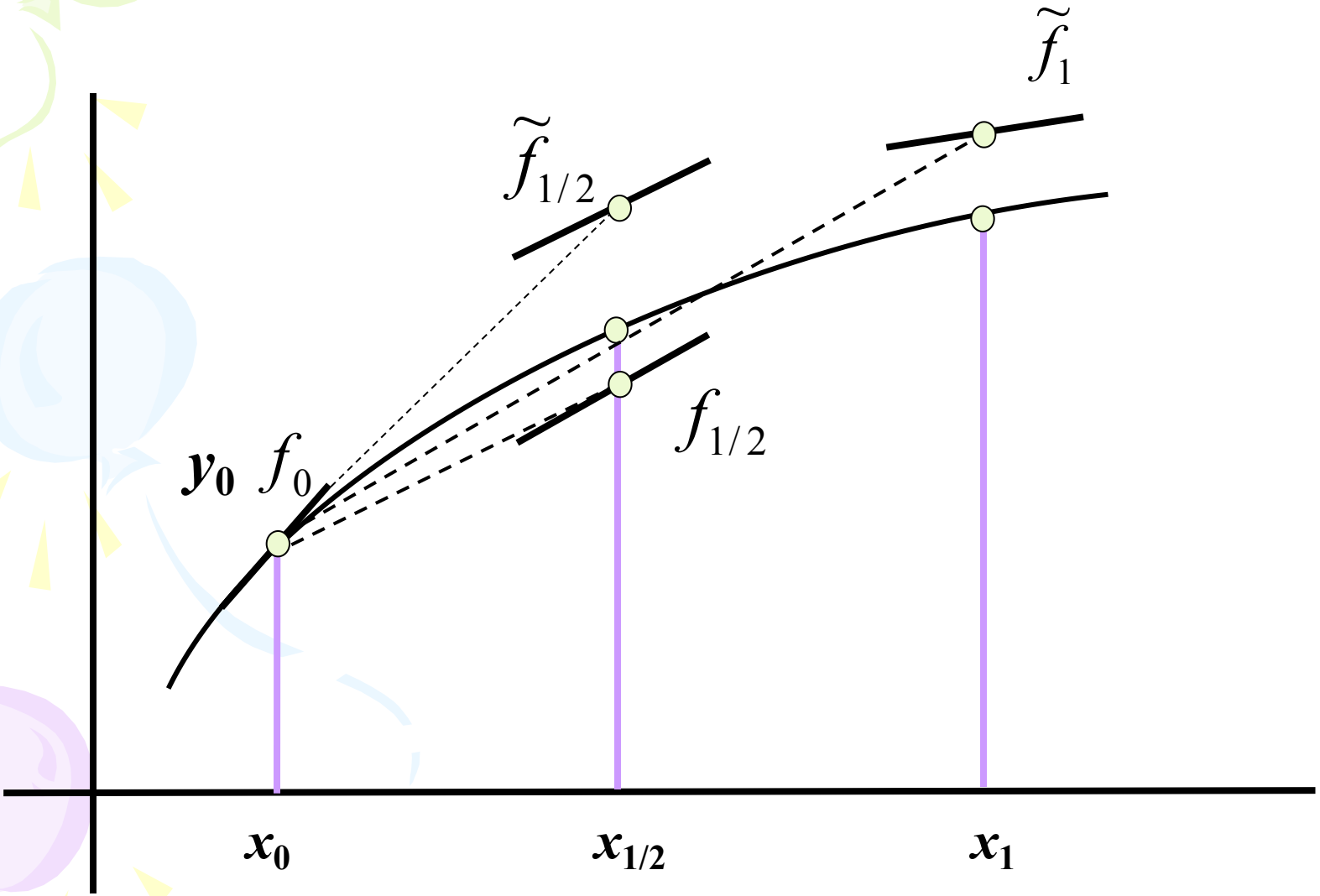
$$\tilde{f}_1 = f(x_0 + h, y_0 + hf_{1/2})$$

$$y_1 = y_0 + \frac{h}{6}(f_0 + 2\tilde{f}_{1/2} + 2f_{1/2} + \tilde{f}_1)$$

- Interpret the sum over interior points as an average slope.
- Local discretization error is proportional to h^5, so halving h leads to a 1/32 reduction in the local discretization error.
- *Very* widely used method

No proof of this – algebra is extremely lengthy!

Runge-Kutta graphically



Comparison of methods

Let's compare solutions of the following system:

$$y' = 1 - x + 4y, \quad y(0) = 1 \Rightarrow y = \frac{x}{4} - \frac{3}{16} + \frac{19}{16} e^{4x}$$

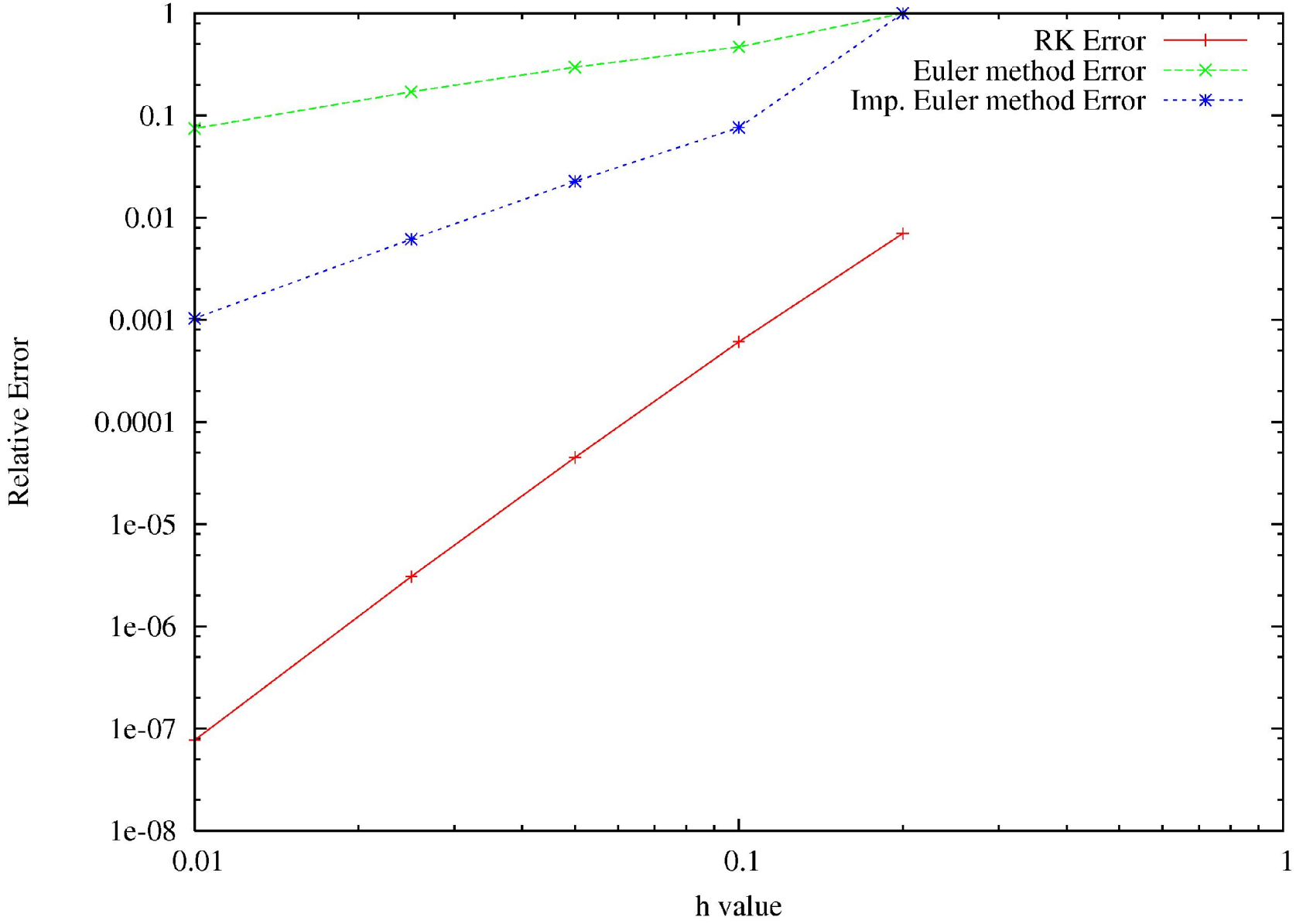
x	Euler h=0.1	Improved Euler h=0.1	RK h=0.2	RK h=0.1	Exact
0	1.0	1.0	1.0	1.0	**1.0**
0.2	1.5	1.595	2.5016	2.5050062	**2.5053299**
0.4	4.4774	5.6099494	5.7776358	5.7927853	**5.7942260**
0.6	8.9038240	12.442193	12.997178	13.047713	**13.052522**
0.8	17.537495	27.348020	28.980768	29.130609	**29.144880**
1.0	34.411490	59.938223	64.441579	64.858107	**64.897803**

Convergence with changes in *h*

Solutions now taken at x=1

h	Euler	Improved Euler	RK	Exact
0.2			64.441579 0.7% error	64.897803
0.1	34.411490	59.938223	64.858107 0.06% error	64.897803
0.05	45.588400	63.424698	64.894875	64.897803
0.025	53.807866	64.497931	64.897604	64.897803
0.01	60.037126	64.830722	64.897798 ~10^{-7} error	64.897803

RK with 10 steps better than improved Euler with 100 steps!


RK Error
Euler method Error
Imp. Euler method Error
Relative Error
h value
1
0.1
0.01
0.001
0.0001
1e-05
1e-06
1e-07
1e-08
0.01
0.1
1

Adapting the step size

- What value of h is optimal?
  - If we consider functions that are smooth in some regions and rapidly changing in others then h for one region is not appropriate in another
- h ought to vary from one region to another
- Let's examine an algorithm to vary h
  - We'll use the relative error on a given interval as the controlling parameter

Algorithm to adapt h

① Let e=fractional error tolerance (say 10^{-4})

② Let D=x_{max}-x_{min} be the domain over which

$y' = f(x, y)$

is to be solved

③ Initially let h=0.1×e×D

④ Use R-K to find y_1=y(x_{min}+h) in one step h

⑤ Next use two R-K steps of size h/2 to find a new $\hat{y}_1$(x_{min}+h) (most accurate estimate yet)

(Now have two values we can apply the Richardson-Extrapolation to)

Algorithm to adapt *h cont.*

⑥ Compute $y_1^* = y^*(x_{\min} + h)$ using y_1 and $\hat{y}_1$ using Richardson-Extrapolation

$$y_1^* = \frac{16\hat{y}_1 - y_1}{15}$$

– Note since error is prop to h^4 must use 1/16 multiplier

⑦ Estimate error:

$$err = \left| \frac{y_1^* - \hat{y}_1}{\frac{1}{2}\left(y_1^* + \hat{y}_1\right)} \right|$$

⑧ If err $< 0.1e \Rightarrow h$ is too small. Increase by 1.5 for next step
If err $> e \Rightarrow h$ is too big. Halve h and repeat iteration
If $0.1\ e \le$ err $\le e \Rightarrow h$ is OK.

⑨ When y^* value is acceptable increment all x values to do next zone

⑩ Stop when $x = x_{\max}$

Notes

- The R.E. ensures that even though we appear to do 3 R-K steps per *h* zone, we actually get a much faster convergence rate than R-K with a step of *h*/3
- The adaptive grid is a necessary part of the error estimation process
- Without local adaption you always have to choose the *h* for the hardest regions in your problem
  - You may well be wasting time in regions that are easy to solve
- It doesn't make any sense to use R-K without adaptive stepping!

Issues with the adaptive step size algorithm

- Step (8)

  If err $< 0.1e \Rightarrow h$ is too small. Increase by 1.5 for next step

  If err $> e \Rightarrow h$ is too big. Halve h and repeat iteration

  If $0.1\ e \leq$ err $\leq e \Rightarrow h$ is OK.

  - **Clearly if there is problems with convergence this step will continue to keep dividing *h* forever. You need to set up a limit here, either by not allowing *h* to get smaller than some preset limit or counting the number of times you halve *h***

- Because of rounding error as you increment x it is very unlikely that you will precisely hit $x_{\max}$ with the final step
  - Therefore you should choose $h=\min(h, x_{\max}-x)$ where x is current position

Issues with the adaptive step size algorithm: more "gotcha's"

- Step (7) Estimate error

$$err = \left| \frac{y_1^* - \hat{y}_1}{\frac{1}{2}\left(y_1^* + \hat{y}_1\right)} \right|$$

  - Should add a very small ("tiny") number to the absolute value of the denominator to avoid a divide by zero (when the two estimates are very close to zero you might - in incredibly rare situations - get the two numbers adding to zero).
- Since you store values of y and x in an array it is possible that you may run out of storage (because you need too many points). You should do some check of the number of positions stored versus the maximum possible allowed in your decleration of the x & y arrays . This will avoid the possibility of a segmentation fault.

Summary

- As with numerical integration, low order methods do not exhibit high accuracy
- RK methods provide a good compromise between computational cost and stability
- Adaptive RK methods are extremely powerful and using convergence tests that use R.E. ensures optimal usage of compute cycles

Homework 13：05/17/2017

Problem 1: Let's compare solutions of the following system:

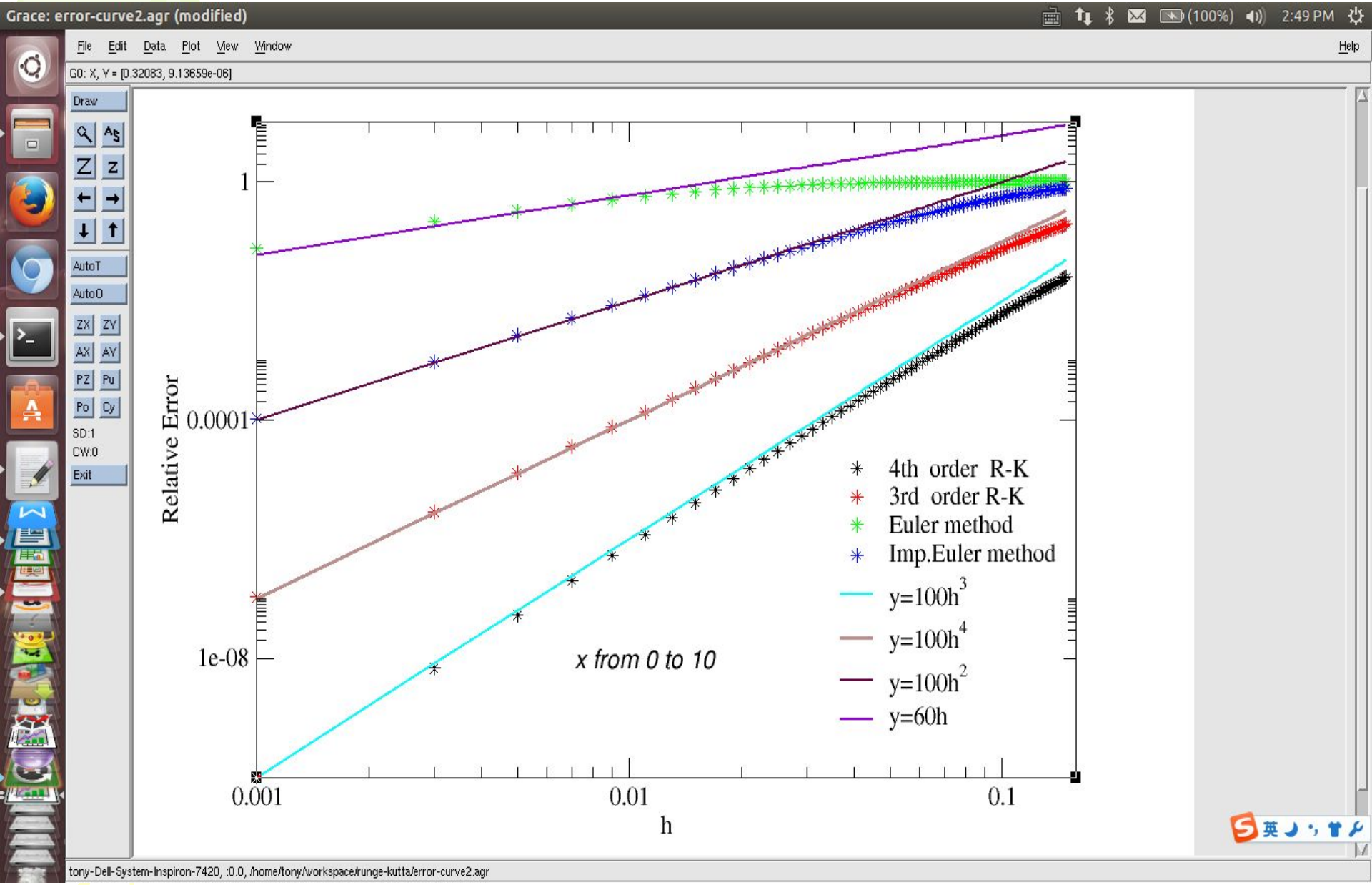
$$y' = 1 - x + 4y, y(0) = 1$$

a) Ralston's Method
b) 3th order RK
c) 4th order RK
d) 5th order RK or higher (including deducing the formula)

Hints: Choose enough number of the step *h*, and fit the relative error E_r with *h*. For example: for **Ralston's Method** , we should get

$$\mathrm{E_r} \propto O(h^2)$$

Thanks Tony (my student)



Problem 2: Calculate the motion given the nonlinear equations of motion for a pendulum:

$$\frac{d^2\theta}{dt^2} = -\frac{g}{L} + \sin(\theta)$$

a) Write a program to solve the two first-order equations obtained from the above second order equation, using the fourth-order Runge–Kutta method for a pendulum with a L=10cm arm. Use your program to calculate the angle θ of displacement for small angles for several periods of the pendulum when it is released from a standstill at $\theta = 10^\circ$ from the vertical. Make a graph of θ as a function of time and an additional graph of $d\theta/dt$ as a function of θ (i.e. the velocity of the oscillator against its position). The later plot is called a phase space plot.

b) Use your program to calculate the angle θ of displacement for several periods of the pendulum when it is released from a standstill at $\theta = 179^\circ$ from the vertical. Make a graph of θ as a function of time and a graph of $d\theta/dt$ as a function of θ.

Hints: in a printout of your finished program together with your plots from part a) and b).

Project 2: The Character of a Short Spring

Let's examine realistic effects of a short spring system limited in its stretching length. For example, a spring made of several windings at most would stretch to the unwound length of the spring(assuming the wire itself does not stretch). One could expect some nonlinearity between the force applied to the spring and the stretch of the spring. The spring constant, instead of being constant as in a simple harmonic oscillator, could have a quadratic dependence. That is, k would be proportional to x^2 giving the nonlinear equation:

$$\frac{d^2x}{dt^2} + Ax^3 = 0$$

A realistic system will also have a frictional term proportional to the velocity. To over come losses due to friction, and to make the system more interesting at large time values, let's have the system periodically driven. The resulting equation of motion is given below:

$$\frac{d^2x}{dt^2} + b\frac{dx}{dt} + Ax^3 = B\cos(\omega_d t)$$

This equation describes what is known as Duffing's oscillator, and it is know to exhibit a wide variety of behaviors. A, B, b, and ω_d are adjustable parameters. For this study, let's set A=1 and ω_d=1; though they need not be.

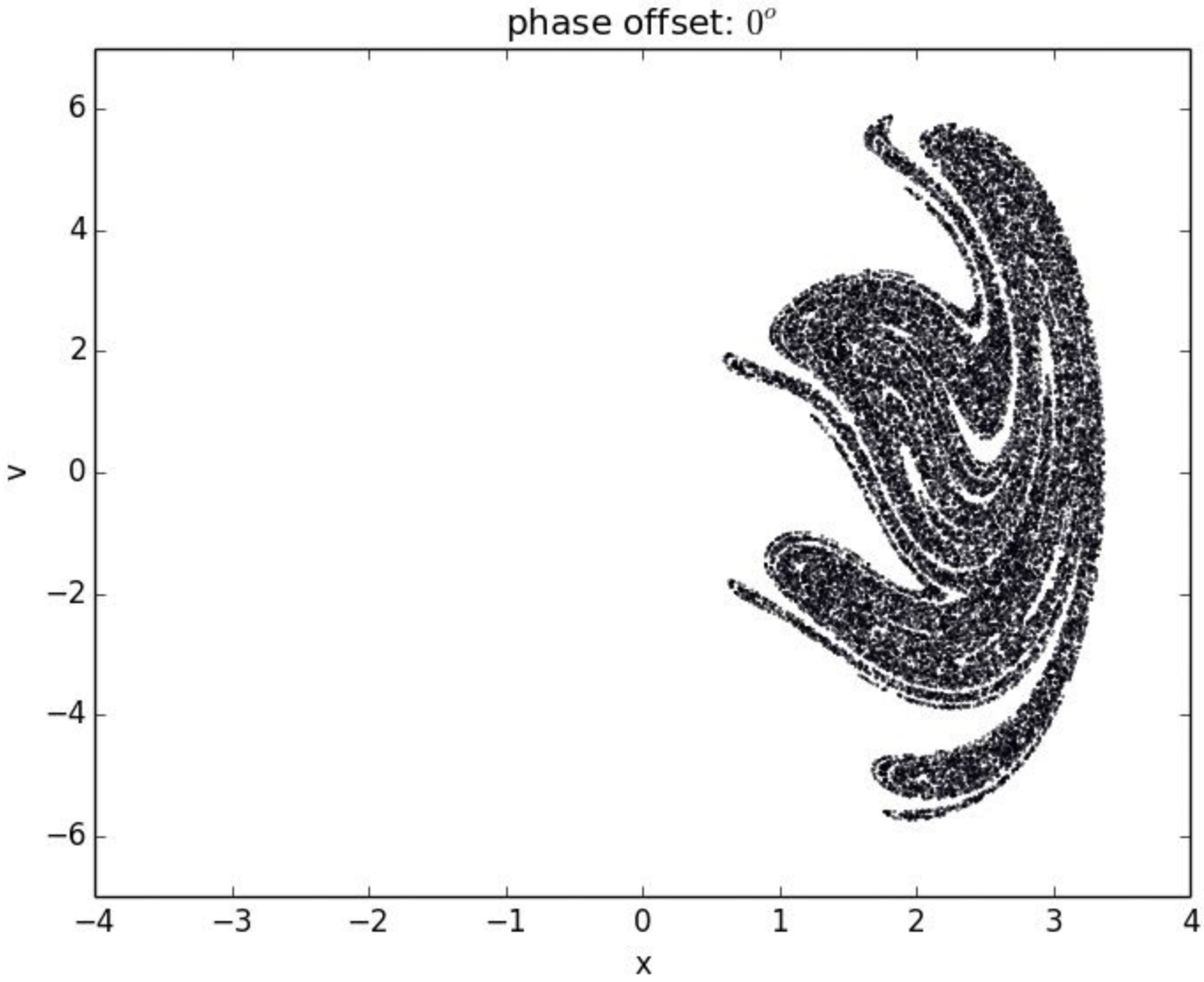
a) Write a program to solve the two first-order equations obtained from the above second order equation, using the fourth-order Runge–Kutta method. Develop your program to graph the position x versus time for 25 periods of the driving frequency ω_d and to graph the phase space (i.e. x versus dx/dt). Use as starting values for t =0, x = 3 and dx/dt=0. Test your program with parameters (B, b) = (7, 6). You should see the motion transition to a steady state harmonic motion.

b) Study the system for various choices of the following parameters (B, b) = (7, 6), (7,0.6),(10, 0.05), and (7, 0.01). For each set of parameters plot the position vs time and the phase space. After the initial transient motion, describe the motion. Is it a steady motion? Do you see any bifurcation(or splitting) of the motion? Are there any solutions which are clearly not a steady motion?

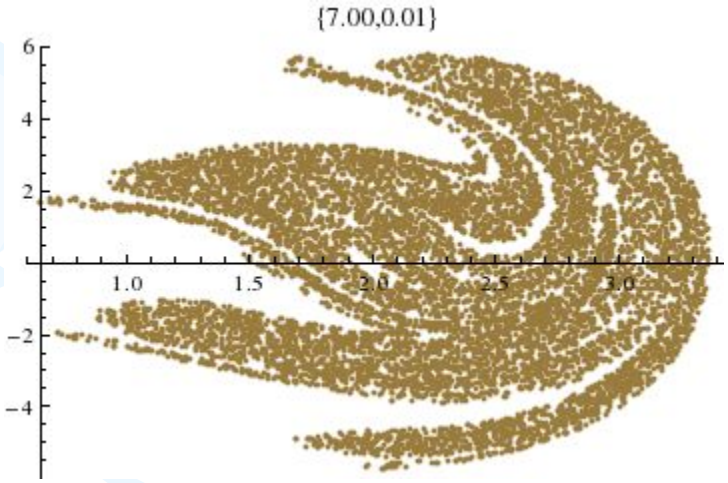
c) A measurement of the divergence is attained with the use a device known as the Poincare‘ section. To generate a Poincare' section, one asks what the phase space looks like at the same phase angle ($\omega_d t + \phi$) of the driving frequency, which for our short spring system, is the time corresponding to a multiple of 2 π (remember $\omega_d = 1$) plus an arbitrary phase. In other words, Poincare' sections are phase space plots with all points erased save those that correspond to a time value of $n2\pi$ of the forcing period.

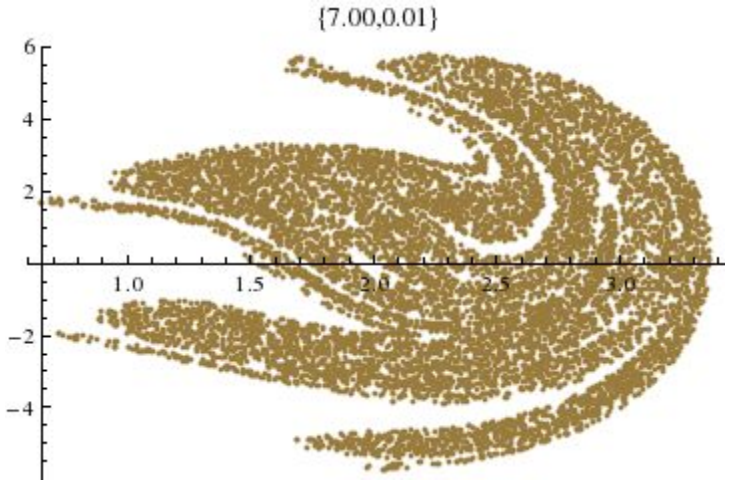
Write a new program (copy and modify the existing program) to create a Poincare' section for the parameters $(B,b) = (7, 0.01)$, at $n2\pi$. Since the Poincare' section uses only one data point per driving cycle, to receive good resolution many steps are needed(Usually one to two orders of magnitude over that required in the earlier plots). (Hint: one may want to choose a time step equivalent to one degree of the driven phase angle so that one cycle equals 360 steps .)

Chaos, chaotic behavior, is described as arising from phase space operations of stretching and folding. Because energy conservation confines the motion to a finite volume in phase space, the chaotic system cannot diverge exponentially forever. The phase space motion must at some time pass near or on prior states; this is know as stretching (exponential divergence in phase space) and folding (confinement in phase space) and it is responsible for the picturesque fractal behavior that is observed in the Poincare' section.


phase offset: 0^o
6
4
2
0
−2
−4
−6
y
−4
−3
−2
−1
0
1
2
3
4
x

刘汉琛





张智依