Numerically solving ODEs

Ordinary differential equations (ODEs) have the form:

$$\frac{dx}{dt} = f(t)$$

- ODEs are functions of a single variable.
- The order of an ODE corresponds to the highest order derivative in the equation
- Initial value ODEs have a defined starting value
- Boundary value ODEs have defined starting and ending points
  - solved iteratively
  - at least second order ODE

Radioactive decay is described by a first order ODE:

$$\frac{dC}{dt} = -\lambda C$$

Predator(X)-Prey(Y) models may be first order ODEs:

$$\frac{dX}{dt} = BXY - A_1X$$
$$\frac{dY}{dt} = -BXY + A_2Y$$

Where BXY is the growth in predator as it eats the prey. The AX terms describe Predator death rate and Prey growth rate.

The oscillation of a spring (and other things) is characterized by a second order ODE:

$$\frac{w}{g}\frac{d^2x}{dt^2} + b\frac{dx}{dt} + kx(t) = 0$$

where the weight at the end of a spring (w) moves over distances (x) in response to gravity (g), a spring constant (k), and a friction (b).

Any ODE can be reduced to a first order ODE. For example:

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

let $z(x) = \frac{dy}{dx}$

$$\frac{dz(x)}{dx} + q(x)z(x) = r(x)$$

Because higher order ODE can be reduced, most numerical methods focus on solving first order ODEs.

The solution to the ODE characterizing radioactive decay:

$$\frac{dC}{dt} = -C\lambda$$

can be solved directly.

$$C = C_0e^{-\lambda\Delta t}$$

This pair of equations can be used to evaluate a variety of numerical solution methods.

Euler's method

- simple
- based on Taylor series truncated at the first term.

$$C(t+\Delta t) = C(t) + \frac{dC}{dt}\frac{\Delta t^1}{1!} + \frac{d^2C}{dt^2}\frac{\Delta t^2}{2!}\dots$$

Truncating this after the first order term:

$$C(t+\Delta t) = C(t) + \frac{dC}{dt}\frac{\Delta t^1}{1!}$$

Further estimating the derivative using the current concentration:

$$\frac{dC}{dt} \approx -C(t)\lambda$$

Explicit Euler's Method

$$C(t+\Delta t) = C(t) + \frac{dC}{dt}\frac{\Delta t^1}{1!}$$
$$\frac{dC}{dt} \approx -C(t)\lambda$$
$$C(t+\Delta t) = C(t) - C(t)\lambda\Delta t$$

- This is not a very good method, requires very small time step size to get accurate results.
- This method has stability problems when the step size is large.
- Why is this only accurate for a small step size?
- Will accuracy improve as step size is decreased?
- What errors are introduced in numerical methods?

Errors in Numerical Methods

- Truncation error (missing terms in Taylor Series)
- Round Off:Computer 'only' accurate to 16(?) decimal places.
- Examine the following Python script:
  - How doe the solution based on Euler's method vary with step size?
  - How does the decay constant influence the solution (large λ , large step)?

```
## Explicit Euler's Method
## solve radiocative devay numerically and compare with analytic solution
import math,pylab
def RadDecay(InitConc,Lambda,Time): ## define analytic solution
    FinalConc=InitConc*math.exp(-Lambda*Time)
    return FinalConc
def DecayDeriv(Conc,Lambda): ## define derivative
    DCDt=-Lambda*Conc
    return DCDt
InitConc=100 ##initialize
TotalTime=5000.
Lambda=math.log(2.)/5000.
steps=10
DelT=TotalTime/steps
TimeNow=0
Conc1=[]
Conc2=[]
TimeSteps=[]
TimeSteps.append(TimeNow)
Conc1.append(InitConc)
Conc2.append(InitConc)
for i in range(steps):
    TimeNow=TimeNow+DelT
    TimeSteps.append(TimeNow)
    Conc1.append(RadDecay(InitConc,Lambda,TimeNow))
    Conc2.append(Conc2[i]+DelT*DecayDeriv(Conc2[i],Lambda))
pylab.plot(TimeSteps,Conc2, markerfacecolor='red',marker='o',markersize=3.,linestyle='None')
pylab.plot(TimeSteps,Conc1,color='black')
pylab.show()
```

Implicit methods

The previous program calculates a new value based on a current value of a function. This is an *explicit* method, using old values to calculate new values. An alternative approach is to set the right side of the equation for Euler's Method in terms of new values.

$$\frac{dC}{dt} \approx -C(t+\Delta t)\lambda$$
$$C(t+\Delta t) = C(t) - C(t+\Delta t)\lambda\Delta t$$

- derivative now evaluated at the end point
- $\frac{dC(t)}{dt}$ to $\frac{dC(t+\Delta t)}{dt}$
- In simple decay equation rearranged so $C(t+\Delta t)$ calculated directly
- in more complicated situations the equation is solved iteratively.

```
import math
def euler(DelX,Y0,dYdX,Konst):
    Y1=Y0+DelX*dYdX(Y0,Konst)
    return Y1
def eulerI(DelX,Y0,dYdX,Konst):
    Y11=Y0+10.
    Y12=Y0
    while abs(Y12-Y11)>1.e-4:
        Y11=Y12
        Y12=Y0+DelX*dYdX(Y11,Konst)
    return Y12
def dYdX(X,Konst):
    Y=-Konst*X
    return Y

DelT=1000.; T=0; Y=100
YList1=[Y]; YList2=[Y]; TList=[T]
Konst=1.e-4
for i in xrange(10):
    T=T+DelT
    TList.append(T)
    Y=euler(DelT,YList1[-1],dYdX,Konst)
    YList1.append(Y)
    Y=eulerI(DelT,YList2[-1],dYdX,Konst)
    YList2.append(Y)
print "final answer", YList1[-1],YList2[-1],math.exp(-T*Konst)
```

- Euler's method approximates the derivative over an interval at the starting (or ending) point.
- If the derivative changes over this interval, the approximation won't be very good.
- To improve Euler's method,
  - the derivative can be calculated at several points and averaged
  - the derivative can be calculated at a 'more representative' point
  - include higher order derivatives in calculation.

Huen's Methods

Huen's Method estimates the derivative at the end of the interval of interest, and uses the average of the derivative at the starting and ending point to make a final estimate.

$$\frac{dC(t)}{dt} = -\lambda C(t)$$
$$C^*(t+\Delta t) = C(t) + \Delta t\frac{dC}{dt}$$
$$\frac{dC^*(t+\Delta t)}{dt} = -\lambda C^*(t+\Delta t)$$
$$C(t+\Delta t) = C(t) + \frac{\Delta t}{2}\left(\frac{dC}{dt} + \frac{dC^*(t+\Delta t)}{dt}\right)$$

Midpoint Methods

An alternative to using an average, is to use the derivative estimated at the middle of the interval to take the full step.

$$\frac{dC(t)}{dt} = -\lambda C(t)$$
$$C(t + \frac{\Delta t}{2}) = C(t) + \frac{\Delta t}{2}\frac{dC}{dt}$$
$$\frac{dC(t + \frac{\Delta t}{2})}{dt} = -\lambda C(t + \frac{\Delta t}{2})$$
$$C(t + \Delta t) = C(t) + \Delta t \frac{dC(t + \frac{\Delta t}{2})}{dt}$$

Runge-Kutta

- Midpoint and Huen's method examples of second order Runge-Kutta.
- Higher order Runge-Kutta are popular methods for solving ODE's.
- Runge-Kutta methods are developed by comparing truncated Taylor Series to weighted average expressions.

For example, defining a function to solve an ODE based on two control points:

$$C(t + \Delta t) = C(t) + \Delta t(\frac{dC(t_1)}{dt}W_1 + \frac{dC(t_2)}{dt}W_2)$$

The weights(W) and position of the derivatives are chosen by comparing this function to a truncated Taylor series.

$$C(t + \Delta t) \approx C(t) + \frac{dC(t)}{dt}\frac{\Delta t}{1} + \frac{d^2C(t)}{dt^2}\frac{\Delta t^2}{2}$$

re-writing the weighted function:

$$C(t + \Delta t) = C(t) + \Delta t(\frac{dC(t)}{dt}W_1 + \frac{d}{dt}\left(C(t) + a\Delta t\frac{dC(t)}{dt})\right)W_2)$$
$$C(t + \Delta t) = C(t) + \Delta t\frac{dC(t)}{dt}W_1 + \Delta t W_2(\frac{dC(t)}{dt} + a\Delta t\frac{d^2C(t)}{dt^2})$$
$$C(t + \Delta t) = C(t) + \Delta t\frac{dC(t)}{dt}W_1 + \Delta t W_2\frac{dC(t)}{dt} + aW_2\Delta t^2\frac{d^2C(t)}{dt^2}$$
$$C(t + \Delta t) = C(t) + \Delta t(W_1 + W_2)\frac{dC(t)}{dt} + a\Delta t^2 W_2\frac{d^2C(t)}{dt^2}$$

Comparing the previous result to the truncated Taylor series:

$$W_1 + W_2 = 1$$
$$aW_2 = \frac{1}{2}$$

This presents a infinite number of possible solutions. When $W_1 = \frac{1}{2}$ then $W_2 = \frac{1}{2}$ and $a = 1$ (Huen's Method). When $W_1 = 0$ then $W_2 = 1$ and $a = \frac{1}{2}$ (Midpoint method). Note that a is the relative position along the interval Δt where the derivative is evaluated.

Forth Order Runge-Kutta

Using this methodology, higher order methods can be developed.

- calculate $\frac{dC}{dt}$ at starting point
- calculate $\frac{dC}{dt}$ at midpoint conc based on starting conc. and derivative
- calculate $\frac{dC}{dt}$ at midpoint conc based on starting conc. and derivative at second point
- calculate $\frac{dC}{dt}$ at endpoint conc based on starting conc. and derivative at third point
- calculate final conc using weighted average of derivatives

$$\frac{dC_0}{dt} = -\lambda C_0$$
$$C_1 = C_0 + 0.5 \cdot \Delta t \frac{dC_0}{dt}$$
$$\frac{dC_1}{dt} = -\lambda C_1$$
$$C_2 = C_0 + 0.5 \cdot \Delta t \frac{dC_1}{dt}$$
$$\frac{dC_2}{dt} = -\lambda C_2$$
$$C_3 = C_0 + \cdot \Delta t \frac{dC_3}{dt}$$
$$C_{final} = x_0 + \frac{\Delta t}{6}\left(\frac{dC_0}{dt} + 2 \cdot \frac{dC_1}{dt} + 2 \cdot \frac{dC_2}{dt} + \frac{dC_3}{dt}\right)$$

Adams multistep methods

Two classes of methods:

- Open or Explicit,
  - not self starting
  - Adams-Bashforth
- Closed or Implicit
  - some are self starting
  - Adams-Moulton
- Good general method for solving ODE's

Methods that are not 'self starting' require the application of another method to calculate a few starting values. These methods are based on the Taylor series. The explicit method start with a forward approximation:

$$y(t + \Delta t) = y(t) + \Delta t \cdot \frac{d(y(t))}{dt} + \frac{\Delta t^2}{2} \cdot \frac{d^2(y(t))}{dt^2} + \ldots \quad (1)$$
$$y(t + \Delta t) = y(t) + \Delta t \cdot \frac{d(y(t))}{dt} + \frac{\Delta t}{2}\left(\cdot \frac{d(y(t))}{dt} - \frac{d(y(t - \Delta t))}{dt}\right) \quad (2)$$
$$y(t + \Delta t) = y(t) + \Delta t\left(\frac{3}{2}\frac{dy(t)}{dt} - \frac{1}{2}\frac{dy(t - \Delta t)}{dt}\right) \quad (3)$$

- substitute equation describing derivatives, and solve for y.
- higher order methods developed by including additional terms in the truncated Taylor series

The third-order explicit formula is derived using backward finite-difference approximations for derivatives:

$$\frac{dy}{dt} = \frac{3 \cdot y_t - 4 \cdot y_{t-\Delta t} + y_{t-2\Delta t}}{2 \cdot \Delta t} \tag{4}$$

$$\frac{d^2y}{dt^2} = \frac{y_t - 2 \cdot y_{t-\Delta t} + y_{t-2\Delta t}}{\Delta t^2} \tag{5}$$

$$y_{t+\Delta t} = y_t + \Delta t \cdot \frac{dy_t}{dt} + \frac{\Delta t^2}{2} \cdot \frac{d^2y_t}{dt^2} + \frac{\Delta t^3}{6} \cdot \frac{d^3y_t}{dt^3} \dots \tag{6}$$

$$y_{t+\Delta t} = y_t + \Delta t \cdot \frac{dy_t}{dt} + \frac{\Delta t^2}{2} \cdot \frac{d}{dt}\left(\frac{dy_t}{dt}\right) + \frac{\Delta t^3}{6} \cdot \frac{d}{dt}\left(\frac{d^2y_t}{dt^2}\right) \dots \tag{7}$$

$$y_{t+\Delta t} = y_t + \Delta t \cdot \frac{dy_t}{dt} + \frac{\Delta t^2}{2} \frac{d}{dt}\left(\frac{3 \cdot y_t - 4 \cdot y_{t-\Delta t} + y_{t-2\Delta t}}{2 \cdot \Delta t}\right) + \tag{8}$$

$$\frac{\Delta t^3}{6} \frac{d}{dt}\left(\frac{y_t - 2 \cdot y_{t-\Delta t} + y_{t-2\Delta t}}{\Delta t^2}\right) \tag{9}$$

$$y_{t+\Delta t} = y_t + \Delta t \left(\frac{23}{12}\frac{dy}{dt}_t - \frac{4}{3}\frac{dy}{dt}_{t-\Delta t} + \frac{5}{12}\frac{dy}{dt}_{t-2\Delta t}\right) \tag{10}$$

The implicit method start with a backwards approximation:

$$y(t) = y(t + \Delta t) - \Delta t \cdot \frac{d(y(t + \Delta t))}{dt} + \frac{\Delta t^2}{2} \cdot \frac{d^2(y(t + \Delta t))}{dt^2} + \dots \tag{11}$$

$$y(t + \Delta t) = y(t) + \Delta t \cdot \frac{d(y(t + \Delta t))}{dt} - \frac{\Delta t}{2}\left(\cdot \frac{d(y(t + \Delta t))}{dt} - \frac{d(y(t))}{dt}\right) \tag{12}$$

$$y(t + \Delta t) = y(t) + \Delta t \left(\frac{1}{2}\frac{dy(t)}{dt} + \frac{1}{2}\frac{dy(t + \Delta t)}{dt}\right) \tag{13}$$

Substitute 'guess' for $\frac{d(y(t+\Delta t))}{dt}$, based on differential equation, and iterate to a solution.

Can improve on this by making a good 'guess' based on an explicit method (predictor-corrector)

Solving a second (or higher) order ordinary differential equations can be accomplished using the previously described methods by using substitution to recast the equation into a set of first order equations.

$$m\frac{d^2X}{dt^2} + B\frac{dX}{dt} + k \cdot X = P \tag{14}$$

$$\frac{dX}{dt} = y \tag{15}$$

$$m\frac{dy}{dt} + B \cdot y + k \cdot X = P \tag{16}$$

These equations are then solved sequentially, based on initial conditions for both X and y.