Ordinary differential equations

- Mathematical modeling involves posing models and then solving these models numerically or analytically
- ODEs represent a powerful method of modeling
  - Especially if things depend on rates of change
- Rate of change of distance is velocity

$$v=dx/dt \qquad x(t) = \int_0^t \frac{dx}{d\tau} d\tau$$

- Knowing the velocity as a function of τ we can integrate using numerical quadrature
- Rate of change of velocity is acceleration

$$a=dv/dt = d^2x/dt^2$$

- Given initial conditions ($v(0)=0$, $x(0)=0$) find the location $x(t)$ at time t given that the object falls with a constant acceleration of 10 m/s^2

Solution by simple integration

$$\int dv = \int 10 dt$$

$$v = 10\,t + c_1$$

$$\int dx/dt\, dt = \int 10t\, dt$$

$$x = 10t^2/2 + c_1 t + c_2$$

- Use initial conditions
- $v(0)=0$ so c_1 is zero
- $x(0)=0$ so c_2 is zero
- Final solution $x=5t^2$
- Could handle more complex functions of t under the integral

$$\frac{dx}{dt} = f(t) \qquad x(t) = \int_0^t f(\tau) d\tau$$

What if simple integration would not work?

- Example: Let the velocity be a function of *x* and *t*
- *dx/dt=f(x,t)* $x(t) = \int_0^t f(x(\tau), \tau) d\tau$
- Cannot be simply integrated
- This is the typical type of problem we need to solve in ODEs
- This is nonlinear because solution *x* depends on itself
- The linear case could be solved using numerical quadrature

Standard form: Initial Value Problems for ODEs

- We have an "ordinary" differential equation
- Standard form

$$\frac{dy}{dx} = f(x, y)$$

  - Subject to $y(a) = y_0$
- Goal provide values of *y* in an interval [*a*,*b*]
- For future assume
  - Function *f* is given as a blackbox function which we can call with specified numerical arguments
  - *y* and *f* may be vector-valued

Ordinary vs Partial

- ODE
  - Everything is function of a single independent variable
- PDE
  - There are several (more than one) independent variables
- ODEs are characterized by order and degree

Order and Degree

Differential equations are often classified with respect to **order**. The order of a differential equation is the order of the highest order derivative present in the equation.

Example 1

$$\frac{\mathrm{d}^3 y}{\mathrm{d}x^3} + 4x\left(\frac{\mathrm{d}y}{\mathrm{d}x}\right)^2 = y\frac{\mathrm{d}^2 y}{\mathrm{d}x^2} + \mathrm{e}^y \qquad \text{has order 3.}$$

The **degree** of a differential equation is the power of the highest order derivative in the equation. In previous example the degree is 1.

Example 2

$$\left(\frac{\mathrm{d}^2 y}{\mathrm{d}x^2}\right)^3 + \frac{\mathrm{d}y}{\mathrm{d}x} = \sin x \qquad \text{is of order 2 and degree 3.}$$

Big picture: ODE for a function *y(x)*

- knowledge of a relation of its derivatives, the ODE, which is given valid in a specified range of values of *x*
  - *y* is assumed differentiable and continuous (at least so far that the equation makes sense)
- Tool: Taylor series to express *f(x)* over polynomial basis
  - Either solve for values of *f(x)* at specified locations
  - Or, determine coefficients to polynomial approximation
- Will also need values of the function itself, and/or of its derivatives at specified locations (side conditions)
  - **Initial value problem.** Values of function and the derivatives needed are specified at the beginning of the interval.
  - **Boundary value problem.** Values are specified at the ends of the interval

Standard form

• **standard form**,

$$y' = f(t; y)$$

$$y(t_0) = y_0$$

•where the function *y* has *m* components,

•y' means the derivative with respect to *t*, and

•y_0 is a given vector of initial conditions (numbers).

•Writing this component-by-component yields

$$y'_{(1)} = f_1(t, y_{(1)} \ldots y_{(m)})$$

$$\ldots$$

$$y'_{(m)} = f_m(t, y_{(1)} \ldots y_{(m)})$$

with $y_{(1)}(t_0), \ldots, y_{(m)}(t_0)$ given initial conditions

Writing a 2nd order system in standard form

$$u'' = g(t, u, u')$$

$$u(0) = u_0$$

$$u'(0) = v_0$$

- where u_0 and v_0 are given.
- Let $y_1 = u$ and $y_2 = u'$. Then, in standard form:

$$y_2' = g(t, y_1, y_2)$$

$$y_1' = y_2$$

$$y_1(0) = u_0 \quad y_2(0) = v_0$$

A modeling exercise: predator prey problems

- Eco-system (island) that contains rabbits and foxes
- Island has plenty of food for rabbits
- Rabbits reproduce like crazy and would fill-up the island
- Foxes eat rabbits
- Let $r(t)$ represent the number of rabbits and $f(t)$ the number of foxes.
- Model the number of rabbits and foxes on the island and decide if it will reach an equilibrium

Rabbit and fox population

- Rabbit population will grow at a certain rate
  - *a* is the natural growth rate of rabbits in the absence of predation,
- Rabbits will die as they are eaten by foxes. Let the rabbit die if it encounters a fox.
  - *b* is the death rate per encounter of rabbits due to predation,
- Fox population dies off if they cannot eat rabbits
  - *c* is the natural death rate of foxes in the absence of food (rabbits),
- Foxes reproduce if they have food
  - *e* is the efficiency of turning predated rabbits into foxes.
- Initial conditions $R(0)=r_0$ *and* $F(0)=f_0$
- Volterra equations

$$dR/dt = aR - bRF$$
$$dF/dt = ebRF - cF$$

Standard form

- Volterra's model

$$dR/dt = 2R - \alpha RF$$
$$dF/dt = \alpha RF - F$$

- Another example (Kepler)

$$\ddot{u}(t) = -u(t)/r(t)^3$$
$$\ddot{v}(t) = -v(t)/r(t)^3$$

where

$$r(t) = \sqrt{u(t)^2 + v(t)^2}$$

The vector $y(t)$ has four components,

$$y(t) = \begin{bmatrix} u(t) \\ v(t) \\ \dot{u}(t) \\ \dot{v}(t) \end{bmatrix}$$

The differential equation is

$$\dot{y}(t) = \begin{bmatrix} \dot{u}(t) \\ \dot{v}(t) \\ -u(t)/r(t)^3 \\ -v(t)/r(t)^3 \end{bmatrix}$$

```
function ydot = twobody(t,y)
r = sqrt(y(1)^2 + y(2)^2);
ydot = [y(3); y(4); -y(1)/r^3; -y(2)/r^3];
```

Taylor Series

- Let *f(x)* be a real valued function with *n* derivatives in $a \le x \le b$. Then

$$f(x) = f(a) + f'(a)(x-a) + \frac{1}{2!}f''(a)(x-a)^2 + \cdots + \frac{1}{(n-1)!}f^{(n-1)}(a)(x-a)^{n-1} + R_n$$

$$R_n \le \frac{|x-a|^n}{n!} \sup_{y \in [a,b)} \left|f^{(n-1)}(y)\right|$$

This provides us the values of *f* at any point *x* in terms of its values and that of its derivatives at *a*.

Solving differential equations: Euler's method

- use Taylor series
- Euler's method

$$y(t+h) = y(t) + hy'(t) + h^2/2 \quad y''(\xi)$$

for some point ξ *in* $[t, \le t+h]$

- Note that

$$y'(t) = f(t, y(t)).$$

$\square$ *March forward*

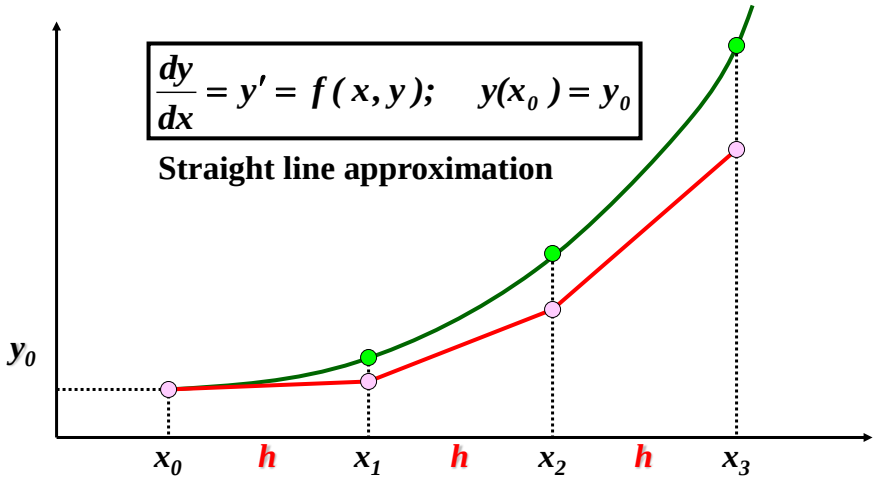
$$y(h) = y_1 = y(0) + hy'(0)$$

$$= y_0 + hf(t_0, y_0)$$

$$y_{n+1} = y_n + hf_n$$

Euler's Method:First-order Taylor

$$\frac{dy}{dx} = x + y \qquad y(0) = 1 \qquad y(x) = 2e^{x} - x - 1$$



Euler's Method Example

The initial condition is: $y(0) = 1$ $\qquad \frac{dy}{dx} = x + y$

The step size is: $\Delta h = 0.02$

Loop using initial conditions and definition of the derivative

The derivative is calculated as: $y'_{i} = x_{i} + y_{i}$

The next *y* value is calculated: $y_{i+1} = y_{i} + \Delta h\, y'_{i}$

Take the next step: $x_{i+1} = x_{i} + \Delta h$

Euler's Method Example

The results

x_n	y_n	y'_n	hy'_n	Exact Solution	Error
0	1.00000	1.00000	0.02000	1.00000	0.00000
0.02	1.02000	1.04000	0.02080	1.02040	-0.00040
0.04	1.04080	1.08080	0.02162	1.04162	-0.00082
0.06	1.06242	1.12242	0.02245	1.06367	-0.00126
0.08	1.08486	1.16486	0.02330	1.08657	-0.00171
0.1	1.10816	1.20816	0.02416	1.11034	-0.00218
0.12	1.13232	1.25232	0.02505	1.13499	-0.00267
0.14	1.15737	1.29737	0.02595	1.16055	-0.00318
0.16	1.18332	1.34332	0.02687	1.18702	-0.00370
0.18	1.21019	1.39019	0.02780	1.21443	-0.00425
0.2	1.23799	1.43799	0.02876	1.24281	-0.00482

Euler's Method

The trouble with this method is

- Lack of accuracy
- Small step size needed for accuracy

- How do we improve it?
- Euler's equation was based on first order Taylor series
  - Maybe use higher order Taylor series

Runge-Kutta Methods

The initial conditions are:

$$\frac{dy}{dx} = f(x, y) \qquad y(x_0) = y_0$$

To derive method we use the Taylor series expansion including 2nd order terms

$$y(x_{n+1}) = y(x_n) + h\frac{dy(x_n, y_n)}{dx} + \frac{h^2}{2!}\frac{d^2y(x_n, y_n)}{dx^2}$$

Runge-Kutta Methods

Expand the derivatives:

$$\frac{d^2y}{dx^2} = \frac{d}{dx}[f(x, y)] = f_x + f_y\frac{dy}{dx} = f_x + f_y f$$

The Taylor series expansion becomes

$$y_{n+1} = y_n + hf + h^2\left[\frac{1}{2}\left(f_x + f_y f\right)\right]$$

Have expressed second derivative in terms of 1st derivatives of *f*

Runge-Kutta Methods

•Look for a formula of the type

$$y_{n+1} = y_n + ak_1 + bk_2$$

$$k_1 = hf\left(x_n, y_n\right)$$

$$k_2 = hf\left(x_n + \alpha\Delta h, y_n + \beta k_1\right)$$

• Goal is to find some intermediate points that allow us to approximate the second derivative

•Specification of α, β, a, and b provides the formula

Runge-Kutta Methods

Look at formula we want for Runge-Kutta

$$y_{n+1} = y_n + ahf + bhf\left(x_n + \alpha h, y_n + \beta hf\right)$$

Perform a multivariate Taylor series expansion of the function

$$f\left(x_n + \alpha h, y_n + \beta hf\right) = f + \alpha hf_x + \beta hf\ f_y$$

Expand, and group terms

$$y_{n+1} = y_n + ahf + bh\left(f + \alpha hf_x + \beta hf\ f_y\right)$$

$$= y_n + \left[a + b\right]hf + b\alpha h^2 f_x + b\beta h^2 f\ f_y$$

Runge-Kutta Methods

Compare with the Taylor series

$$y_{n+1} = y_n + [a+b]hf + b\alpha h^2 f + b\beta h^2 f\ f_y$$

$$y_{n+1} = y_n + hf + h^2\left[\frac{1}{2}\left(f_x + f_y f\right)\right]$$

$$[a+b] = 1$$

$$\alpha b = \frac{1}{2}$$

$$\beta b = \frac{1}{2}$$

4 Unknowns

3 Equations

Runge-Kutta Methods

The Taylor series coefficients (3 equations/4 unknowns)

$$[a+b] = 1,\ \alpha b = \frac{1}{2},\ \ \beta b = \frac{1}{2}$$

If you select "*a*" as

$$a = \frac{2}{3},\ b = \frac{1}{3},\ \ \alpha = \frac{3}{2},\ \ \beta = \frac{3}{2}$$

If you select "*a*" as

$$a = \frac{1}{2}\ \ b = \frac{1}{2},\ \ \alpha = \beta = 1$$

Runge-Kutta Method (2nd Order) Example

Consider $\frac{dy}{dx} = -y^2$ Exact Solution $y = \frac{1}{1+x}$

The initial condition is: $y(0) = 1$

The step size is: $\Delta h = 0.1$

Use the coefficients $a = \frac{1}{2}\ b = \frac{1}{2},\ \alpha = \beta = 1$

Runge-Kutta Method (2nd Order) Example

$$\Delta h = 0.1$$

$$a = \frac{1}{2}\ b = \frac{1}{2},\ \alpha = \beta = 1$$

$$k_1 = hf\left(x_i, y_i\right)$$

$$k_2 = hf\left(x_i + h, y_i + k_1\right)$$

$$y_{i+1} = y_i + \frac{1}{2}\left[k_1 + k_2\right]$$

Runge-Kutta Method (2nd Order) Example

- The values are similar to that of the Modified Euler
  - also a second order method

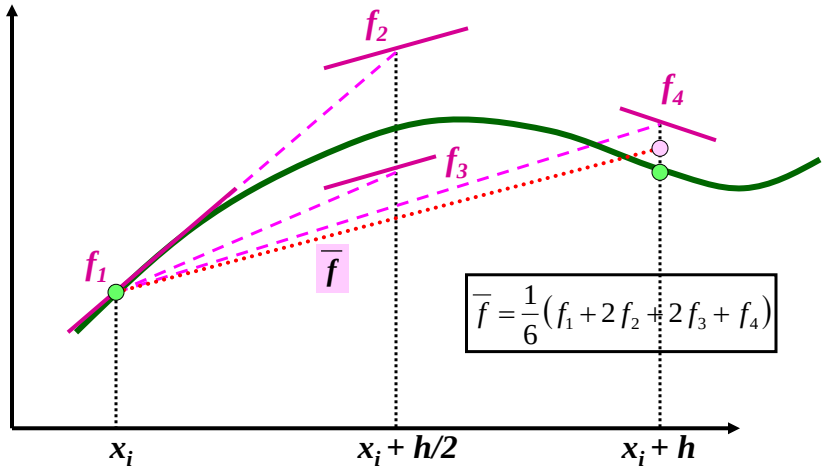
			k_1	Estimate	Solution	k_2	Exact	Error
x_n	y_n	y'_n	hy'_n	y^*_{n+1}	$y^{*'}_{n+1}$	$h(y^{*'}_{n+1})$		
0	1.00000	-1.00000	-0.10000	0.90000	-0.81000	-0.08100	1.000000	0.000000
0.1	0.90950	-0.82719	-0.08272	0.82678	-0.68357	-0.06836	0.909091	-0.000409
0.2	0.83396	-0.69549	-0.06955	0.76441	-0.58433	-0.05843	0.833333	-0.000629
0.3	0.76997	-0.59286	-0.05929	0.71069	-0.50507	-0.05051	0.769231	-0.000740
0.4	0.71507	-0.51133	-0.05113	0.66394	-0.44082	-0.04408	0.714286	-0.000789
0.5	0.66747	-0.44551	-0.04455	0.62292	-0.38802	-0.03880	0.666667	-0.000801
0.6	0.62579	-0.39161	-0.03916	0.58663	-0.34413	-0.03441	0.625000	-0.000790
0.7	0.58900	-0.34692	-0.03469	0.55431	-0.30726	-0.03073	0.588235	-0.000768
0.8	0.55629	-0.30946	-0.03095	0.52535	-0.27599	-0.02760	0.555556	-0.000738
0.9	0.52702	-0.27775	-0.02778	0.49925	-0.24925	-0.02492	0.526316	-0.000705
1	0.50067	-0.25067	-0.02507	0.47560	-0.22620	-0.02262	0.500000	-0.000671

Runge-Kutta Methods

- Fourth order Runge-Kutta method

$$y_{n+1}=y_n+1/6(k_1+2k_2+2k_3+k_4)$$
$$k_1=hf(x,y)$$
$$k_2=h(f(x+h/2,\ y+1/2\ k_1)$$
$$k_3=h(f(x+h/2,\ y+1/2\ k_2)$$
$$k_4=h(f(x+h,y+k_3))$$

4^{th}-order Runge-Kutta Method



Volterra example

- Write a function in standard form

```
function f = rabfox(t,y)
% Computes y' for the Volterra model.
% y(1) is the number of rabbits at time t.
% y(2) is the number of foxes at time t.
global alpha % interaction constant
t % a print statement, just so we can see how
  fast
% the progress is, and what stepsize is being
  used
f(1,1) = 2*y(1) - alpha*y(1)*y(2);
f(2,1) = -y(2) + alpha*y(1)*y(2);
```

Study its solution for various values of encounter

```
% Run the rabbit-fox model for various values of
% the encounter parameter alpha, plotting each
% solution.
global alpha
for i=2:-1:0,
alpha = 10^(-i)
[t,y] = ode45('rabfox',[0:.1:2], [20,10]);
plot(t,y(:,1),'r',t,y(:,2),'b');
legend('rabbits','foxes')
title(sprintf('alpha = %f',alpha));
pause
end
```

Stability

- It turns out that explicit methods are not very stable
- This means that the solution may oscillate if we use large time steps
- So, if we wish to integrate over a large interval, and we need to take many small steps to achieve accuracy, many function evaluations are needed.
- Implicit methods are usually more stable