# 2.29 Numerical Fluid Mechanics
# Fall 2011 – Lecture 20

## REVIEW Lecture 19: Finite Volume Methods

– Review: Basic elements of a FV scheme and steps to step-up a FV scheme

– One Dimensional examples

- Generic equation: $\frac{d\left(\Delta x\,\bar{\Phi}_j\right)}{dt} + f_{j+1/2} - f_{j-1/2} = \int_{x_{j-1/2}}^{x_{j+1/2}} s_\phi(x,t)\,dx$
- Linear Convection (Sommerfeld eqn): convective fluxes
  - 2nd order in space, then 4th order in space, links to CDS
- Unsteady Diffusion equation: diffusive fluxes
  - Two approaches for 2nd order in space, links to CDS

– Two approaches for the approximation of surface integrals (and volume integrals)

– Interpolations and differentiations (express symbolic values at surfaces as a function of nodal variables)

- Upwind interpolation (UDS): $\phi_e = \begin{cases} \phi_P \text{ if } \left(\vec{v}.\vec{n}\right)_e > 0 \\ \phi_E \text{ if } \left(\vec{v}.\vec{n}\right)_e < 0 \end{cases}$ (first-order and diffusive)
- Linear Interpolation (CDS): $\phi_e = \phi_E \lambda_e + \phi_P (1-\lambda_e)$ where $\lambda_e = \frac{x_e - x_P}{x_E - x_P}$ (2nd order, can be oscillatory)

  $\left.\frac{\partial \phi}{\partial x}\right|_e \approx \frac{\phi_E - \phi_P}{x_E - x_P}$
- Quadratic Upwind interpolation (QUICK) $\begin{cases} \phi_e = \phi_U + g_1(\phi_D - \phi_U) + g_2(\phi_U - \phi_{UU}) \\ \phi_e = \frac{6}{8}\phi_U + \frac{3}{8}\phi_D - \frac{1}{8}\phi_{UU} - \frac{3\Delta x^3}{48}\left.\frac{\partial^3 \phi}{\partial x^3}\right|_D + R_3 \end{cases}$
- Higher order (interpolation) schemes

# TODAY (Lecture 20): Time-Marching Methods and ODEs – Initial Value Problems

- Time-Marching Methods and Ordinary Differential Equations – Initial Value Problems
  - Euler's method
  - Taylor Series Methods
    - Error analysis
  - Simple 2nd order methods
    - Heun's Predictor-Corrector and Midpoint Method
  - Runge-Kutta Methods
  - Multistep/Multipoint Methods: Adams Methods
  - Practical CFD Methods
  - Stiff Differential Equations
  - Error Analysis and Error Modifiers
  - Systems of differential equations

# References and Reading Assignments

- Chapters 25 and 26 of “Chapra and Canale, Numerical Methods for Engineers, 2010/2006.”
- Chapter 6 on “Methods for Unsteady Problems” of “J. H. Ferziger and M. Peric, Computational Methods for Fluid Dynamics. Springer, NY, 3rd edition, 2002”
- Chapter 6 on “Time-Marching Methods for ODE’s” of “H. Lomax, T. H. Pulliam, D.W. Zingg, *Fundamentals of Computational Fluid Dynamics (Scientific Computation).* Springer, 2003”
- Chapter 5.6 on “Finite-Volume Methods” of T. Cebeci, J. P. Shao, F. Kafyeke and E. Laurendeau, Computational Fluid Dynamics for Engineers. Springer, 2005.

# Methods for Unsteady Problems – Time Marching Methods
## ODEs – Initial Value Problems (IVPs)

- Major difference with spatial dimensions: Time advances in a single direction
  - FD schemes: discrete values evolved in time
  - FV schemes: discrete integrals evolved in time
- After discretizing the spatial derivatives (or the integrals for finite volumes), we obtained a (coupled) system of (nonlinear) ODEs, for example:

$$\frac{d\,\bar{\mathbf{\Phi}}}{dt} = \mathbf{B}\,\bar{\mathbf{\Phi}} + (\mathbf{bc}) \quad \text{or} \quad \frac{d\,\bar{\mathbf{\Phi}}}{dt} = \mathbf{B}(\bar{\mathbf{\Phi}}, t)\,; \qquad \text{with } \bar{\mathbf{\Phi}}(t_0) = \bar{\mathbf{\Phi}}_0$$

- Hence, methods used to integrate ODEs can be directly used for the time integration of spatially discretized PDEs
  - We already utilized several time-integration schemes with FD schemes. Others are developed next.
  - For IVPs, methods can be developed with a single eqn.: $\frac{d\phi}{dt} = f(\phi, t)\,; \quad \text{with } \phi(t_0) = \phi_0$
  - Note: solving steady (elliptic) problems by iterations is similar to solving time-evolving problems. Both problems thus have analogous solution schemes.

# Ordinary Differential Equations
# Initial Value Problems

ODE: *x* often plays the role of time (following Chapra & Canale's and MATLAB's notation)

$$\frac{d\phi}{dt} = f(t,\phi)\ ; \quad \text{with} \ \ \phi(t_0) = \phi_0 \qquad \Leftrightarrow \qquad \frac{dy}{dx} = f(x,y)\ ; \quad \text{with} \ \ y(x_0) = y_0$$

$$y'(x) = f(x,y)\ , \quad x \in [a,b]$$

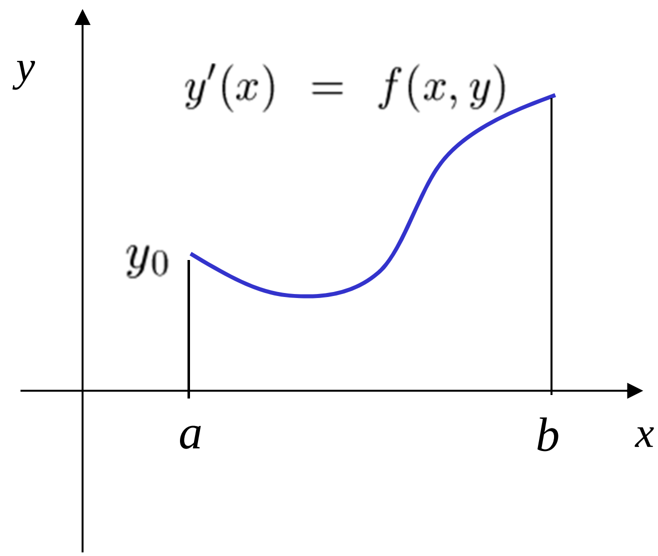
$$y(x_0) = y_0$$

i) For Linear Differential Equation:

$$f(x,y) = -p(x)y + q(x)$$

ii) For Non-Linear Differential Equation:

$f(x,y)$ non-linear in y




Linear differential equations can often be solved analytically

Non-linear equations almost always require numerical solution

# Ordinary Differential Equations
# Initial Value Problems: **Euler's Method**

Differential Equation

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

Example

$$f(x,y) = x \ (y = x^2/2 + p)$$

Discretization

$$x_n = nh$$

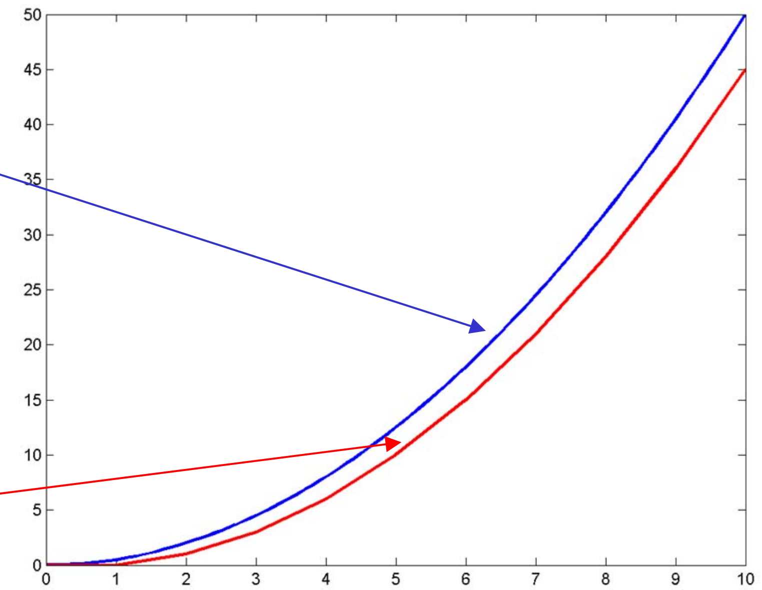
Finite Difference (forward)

$$\frac{dy}{dx}|_{x=x_n} \simeq \frac{y_{n+1} - y_n}{h}$$

Recurrence

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

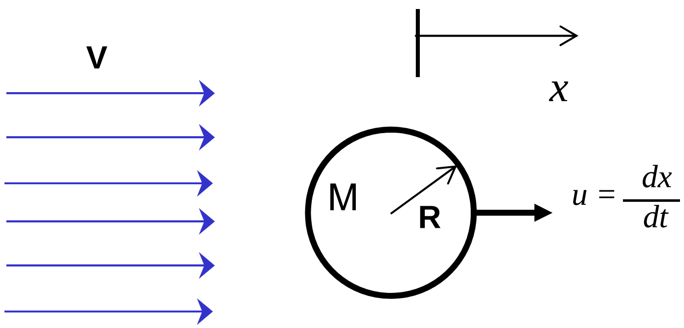
Truncation error (in time): $O(h^2)$




euler.m

(from Lecture 1)

# Sphere Motion in Fluid Flow




Equation of Motion – 2nd Order Differential Equation

$$M\frac{d^2x}{dt^2} = 1/2\rho C_d \pi R^2 \left(V - \frac{dx}{dt}\right)^2$$

Rewite to 1st Order Differential Equations

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

$$\frac{du}{dt} = \frac{\rho C_d \pi R^2}{2M}(V^2 - 2uV + u^2)$$

Taylor Series Expansion
(Here forward Euler)

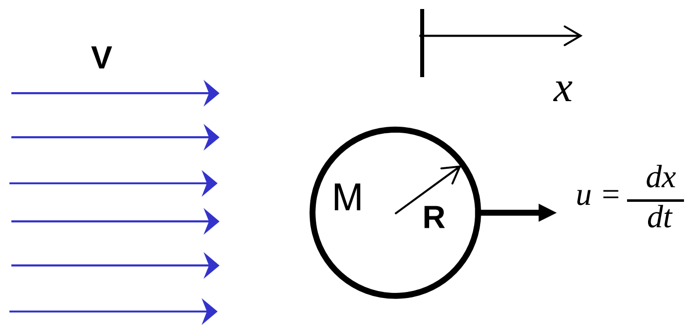
Euler' Method - Difference Equations – First Order scheme

$$u_{i+1} = u_i + \left(\frac{du}{dt}\right)_i \Delta t, \quad u(0) = 0$$

$$x_{i+1} = x_i + \quad u_i \quad \Delta t, \quad x(0) = 0$$

(from Lecture 1)

# Sphere Motion in Fluid Flow
## MATLAB Solutions




dudt.m

```
function [f] = dudt(t,u)
% u(1) = u
% u(2) = x
% f(2) = dx/dt = u
% f(1) = du/dt=rho*Cd*pi*r/(2m)*(v^2-2uv+u^2)
rho=1000;
Cd=1;
m=5;
r=0.05;
fac=rho*Cd*pi*r^2/(2*m);
v=1;

f(1)=fac*(v^2-2*u(1)+u(1)^2);
f(2)=u(1);
f=f';
```

sph_drag_2.m

$$u_{i+1} = u_i + \left(\frac{du}{dt}\right)_i \Delta t, \quad u(0) = 0$$

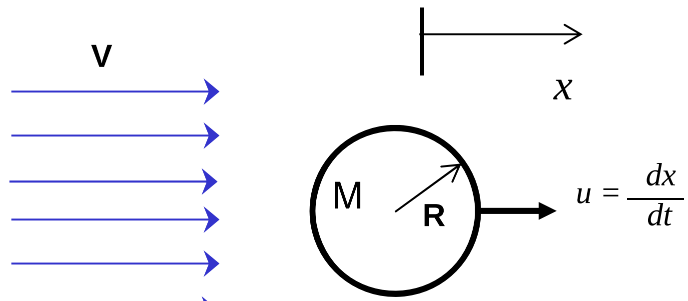
$$x_{i+1} = x_i + u_i \Delta t, \quad x(0) = 0$$

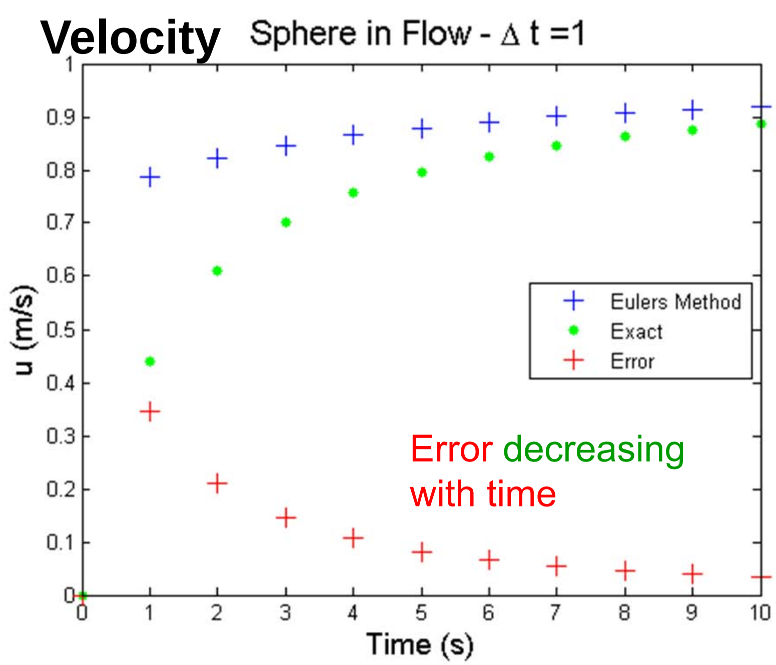
```
x=[0:0.1:10];
%step size
h=1.0;
% Euler's method, forward finite difference
t=[0:h:10];
N=length(t);
u_e=zeros(N,1);
x_e=zeros(N,1);
u_e(1)=0;
x_e(1)=0;
for n=2:N
    u_e(n)=u_e(n-1)+h*fac*(v^2-2*v*u_e(n-1)+u_e(n-1)^2);
    x_e(n)=x_e(n-1)+h*u_e(n-1);
end
% Runge Kutta
u0=[0 0]';
[tt,u]=ode45(@dudt,t,u0);

figure(1)
hold off
a=plot(t,u_e,'+b');
hold on
a=plot(tt,u(:,1),'.g');
a=plot(tt,abs(u(:,1)-u_e),'+r');
...
figure(2)
hold off
a=plot(t,x_e,'+b');
hold on
a=plot(tt,u(:,2),'.g');
a=plot(tt,abs(u(:,2)-x_e),'xr');
...
```

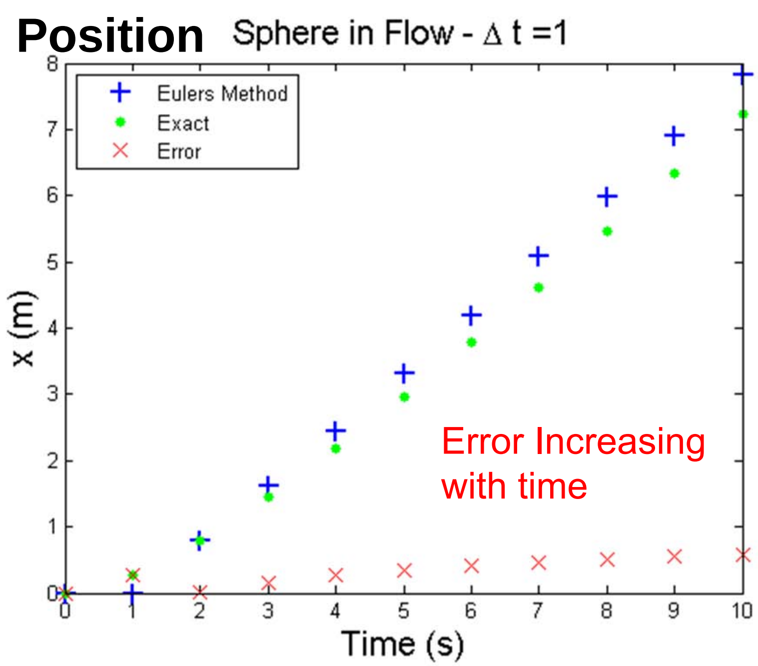
(from Lecture 1)

# Sphere Motion in Fluid Flow

## Error Propagation



# Initial Value Problems: **Taylor Series Methods**

## "Utilize the known value of the time-derivative (the RHS)"

Initial Value Problem:

$$y' = f(x,y)\ ,\quad y(x_0) = y_0$$

Taylor Series

$$y(x) = y_0 + (x - x_0)y'(x_0) + \frac{(x-x_0)^2}{2}y'' + \cdots$$

Derivatives can be evaluated using the ODE:

$$y' = f(x,y) \;\Rightarrow\; y'(x_0) = f(x_0, y_0)$$

$$y'' = \frac{df(x,y)}{dx} = f_x + f_y y' = f_x + f_y f$$

$$y''' = \frac{d^2 f(x,y)}{dx^2} = f_{xx} + f_{xy} f + f_{yx} f + f_{yy} f^2 + f_y f_x + f_y^2 f$$

$$= f_{xx} + 2 f_{xy} f + f_{yy} f^2 + f_x f_y + f_y^2 f$$

where partial derivatives are denoted by:

$$f_x = \frac{\partial}{\partial x}, \qquad f_y = \frac{\partial}{\partial y}$$

Truncate series to $k$ terms, insert the known derivatives

$$y_1 = y(x_1) = y_0 + h y'(x_0) + \frac{h^2}{2!}y''(x_0) + \cdots + \frac{h^k}{k!}y^{(k)}(x_0)$$

$$y_2 = y(x_2) = y_1 + h y'(x_1) + \frac{h^2}{2!}y''(x_1) + \cdots + \frac{h^k}{k!}y^{(k)}(x_1)$$

$$\vdots$$

$$y_n = y(x_n) = y_{n-1} + h y'(x_{n-1}) + \frac{h^2}{2!}y''(x_{n-1}) + \cdots + \frac{h^k}{k!}y^{(k)}(x_{n-1})$$

with a discretization and step size $h$,

$$h = \frac{b-a}{N}$$

$$x_n = a + nh\ ,\quad n = 0, 1, \ldots N$$

Recursion Algorithm:

$$y(x_{n+1}) = y_{n+1} = y_n + hT_k(x_n, y_n) + \frac{h^{k+1}}{(k+1)!}y^{(k+1)}(\xi)$$

where $T_k(x_n, y_n) = f(x_n, y_n) + \frac{h}{2!}f'(x_n, y_n) + \cdots \frac{h^{k-1}}{k!}f^{(k-1)}(x_n, y_n)$

Local Truncation Error: $E = \frac{h^{k+1}f^{(k)}(\xi, y(\xi))}{(k+1)!} = \frac{h^{k+1}y^{(k+1)}(\xi)}{(k+1)!}\ ,\quad x_n < \xi < x_n + h$

# Initial Value Problems: Taylor Series Methods

## Summary of General Taylor Series Method

$$x_n = a + nh \ , \quad n = 0, 1, \ldots N$$

$$y(x_{n+1}) = y_{n+1} = y_n + hT_k(x_n, y_n) + \frac{h^{k+1}}{(k+1)!} y^{(k+1)}(\xi)$$

where:

$$T_k(x_n, y_n) = f(x_n, y_n) + \frac{h}{2!} f'(x_n, y_n) + \cdots \frac{h^{k-1}}{k!} f^{(k-1)}(x_n, y_n)$$

$$E = \frac{h^{k+1} f^{(k)}(\xi, y(\xi))}{(k+1)!} = \frac{h^{k+1} y^{(k+1)}(\xi)}{(k+1)!} \ , \quad x_n < \xi x_n + h$$

**Example: Euler's method** $\quad k = 1$

$$y_{n+1} = y_n + hf(x_n, y_n)$$
$$E = \frac{h^2}{2!} y''(\xi)$$

Note: expensive to compute higher-order derivatives of $f(x,y)$, especially for spatially discretized PDEs => other schemes needed

## Numerical Example – Euler's Method

$$y' = y \ , \quad y(0) = 1 \ , \quad y = e^x$$

$$y(0.01) \simeq y1 = y_0 + hf(x_0, y_0) = 1 + 0.01 \cdot 1 = 1.01$$
$$y(0.02) \simeq y2 = y_1 + hf(x_1, y_1) = 1.01 + 0.01 \cdot 1.01 = 1.021$$
$$y(0.03) \simeq y3 = y_2 + hf(x_2, y_2) = 1.021 + 0.01 \cdot 1.021 = 1.03121$$

$$y(0.03) = 1.0305$$

=> Global Error Analysis, i.e.:

As truncation errors are added at each time step and propagated in time, what is the final total/global truncation error obtained?

# Initial Value Problems: Taylor Series Methods

## Euler's global/total truncation error bound, obtained recursively

$$y' = f(x,y)\ , \quad y(x_0) = y_0$$

Estimate (Euler): $y_{n+1} = y_n + hf(x_n, y_n)\ , \quad n = 0, 1, \cdots$

$$x_n = x_0 + nh$$

Error at step n: $e_n = y(x_n) - y_n$

Exact: $y(x_{n+1}) = y(x_n) + hy'(x_n) + \frac{h^2}{2}y''(\xi_n)\ , \quad x_n < \xi_n < x_{n+1}$

$$e_\ell = \frac{h^2}{2}y''(\xi^n)$$

$$e_{n+1} = y(x_{n+1}) - y_{n+1} = y(x_n) + hy'(x_n) + \frac{h^2}{2}y''(\xi_n) - y_n - hf(x_n, y_n)$$

$$e_{n+1} = (y(x_n) - y_n) + h\left[f(x_n, y(x_n)) - f(x_n, y_n)\right] + \frac{h^2}{2}y''(\xi_n)$$

Since up to $O(e_n^2)$ :

$$f(x_n, y(x_n)) - f(x_n, y_n) = \frac{\partial f(x_n, y_n)}{\partial y}(y(x_n) - y_n) = f_y(x_n, y_n)e_n$$

$$\Rightarrow \quad e_{n+1} = e_n + hf_y(x_n, y_n)e_n + \frac{h^2}{2}y''(\xi_n)$$

$$|e_{n+1}| \leq |e_n| + h|f_y(x_n, y_n)e_n| + \frac{h^2}{2}|y''(\xi_n)|$$

Assume derivatives are bounded: $|f_y(x_n, y_n)| \leq L\ , \quad |y''(\xi_n)| \leq Y$

$$|e_{n+1}| \leq (1 + hL)|e_n| + \frac{h^2}{2}Y$$

$$\eta_{n+1} = (1 + hL)\eta_n + \frac{h^2}{2}Y\ , \quad \eta_0 = 0$$

$$\eta_n = \frac{hY}{2L}[(1 + hL)^n - 1]$$

=> Global Error Bound for Euler's scheme:

$$\begin{aligned} |e_n| \leq \eta_n &= \frac{hY}{2L}[(1 + hl)^n - 1] \\ &\leq \frac{hY}{2L}[(e^{hL})^n - 1] \\ &= \frac{hY}{2L}[e^{hLn} - 1] \\ &\Rightarrow \end{aligned}$$

$$|e_n| \leq \frac{hY}{2L}[(e^{(x_n - x_0)L} - 1]$$

O(1) in h!

= Euler's global or total error bound

# Initial Value Problems: Taylor Series Methods

## Example of Euler's global/total error bound

Example: $y' = y\ ,\quad y(0) = 1\ ,\quad x \in [0,1]$

Exact solution: $y = e^x$

Derivative Bounds: $f_y = 1 \Rightarrow L = 1$

$$y''(x) = e^x \Rightarrow Y = e$$

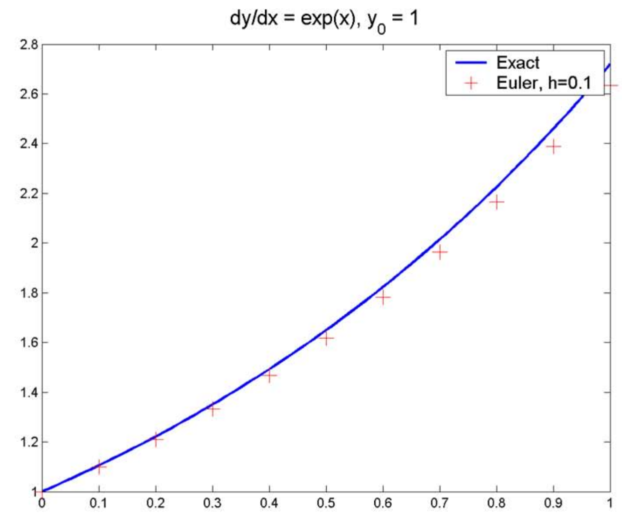
$x - x_0 = n\,h = 1 \Rightarrow \quad |e_n| \leq \dfrac{he}{2}(e-1)$

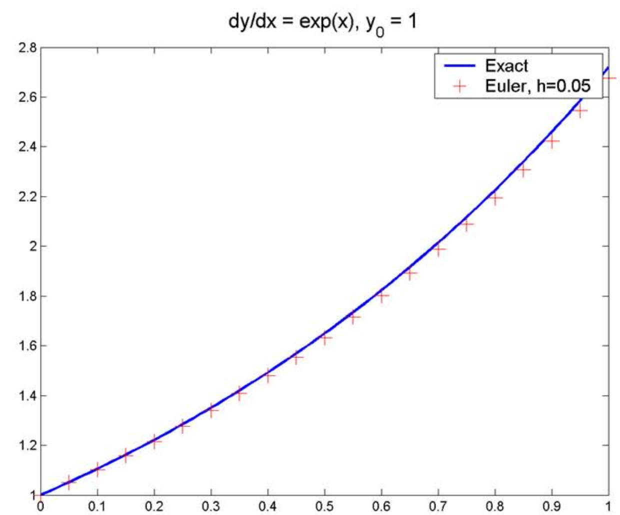
$$h = 0.1 \Rightarrow |e_n| \leq 0.24$$

Euler's Method:

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

$$\begin{aligned} y_{11} &= 2.5937 \\ y(x_{11}) &= 2.71828 \\ e_{11} &= 0.1246 < 0.24 \end{aligned}$$



# Improving Euler's Method

- For one-step (two-time levels) methods, the global error result for Euler can be generalized to any method of $n^{th}$ order:
  - If the truncation error is of $O(h^n)$, the global error is of $O(h^{n-1})$
- Euler's method assumes that the (initial) derivative applies to the whole time interval => $1^{st}$ order global error
- Two simple methods modify Euler's method by estimating the derivatives within the time-interval
  - Heun's method
  - Midpoint rule
- The intermediate estimates of the derivative lead to $2^{nd}$ order global errors
- Heun's and Midpoint methods belong to the general class of Runge-Kutta methods
  - introduced now since they are also linked to classic PDE integration schemes

# Initial Value Problems: **Heun's method**
# (which is also a "one-step" Predictor-Corrector scheme)

Initial Slope Estimate (Euler)

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

**Predictor: Euler**

$$y_{i+1}^0 = y_i + f(x_i, y_i)h$$

which allows to estimate the Endpoint Derivative/slope:

$$y_{i+1}' = f(x_{i+1}, y_{i+1}^0)$$

and so an Average Derivative Estimate:

$$\overline{y}' = \frac{y_i' + y_{i+1}'}{2} = \frac{f(x_i, y_i) + f(x_{i+1}, y_{i+1}^0)}{2}$$

**Corrector**

$$y_{i+1} = y_i + \frac{f(x_i, y_i) + f(x_{i+1}, y_{i+1}^0)}{2}h$$

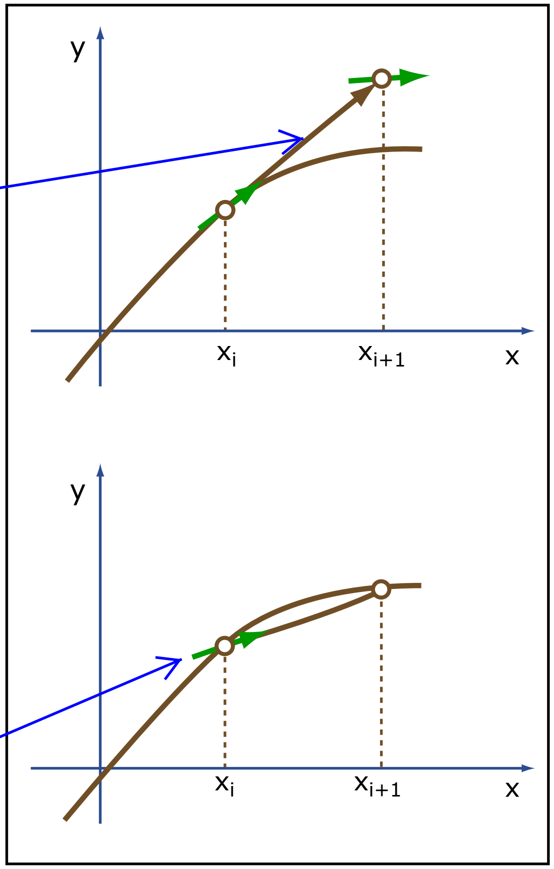



Image by MIT OpenCourseWare.

Heun can be set implicit, one can iterate => **Iterative Heun:**

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

Notes:

- Heun becomes Trapezoid rule if fully implicit scheme is used
- Heun's global error is of 2nd order: $O(h^2)$
- Convergence of iterative Heun not guaranteed + can be expensive with PDEs

# Initial Value Problems: **Midpoint method**

First: uses Euler to obtain a Midpoint Estimate:

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

Then: uses this value to obtain a Midpoint Derivative Estimate:

$$y'_{i+1/2} = f(x_{i+1/2}, y_{i+1/2})$$

Assuming that this slope is representative of the whole interval => Midpoint Method recurrence:

$$y_{i+1} = y_i + f(x_{i+1/2}, y_{i+1/2})h$$

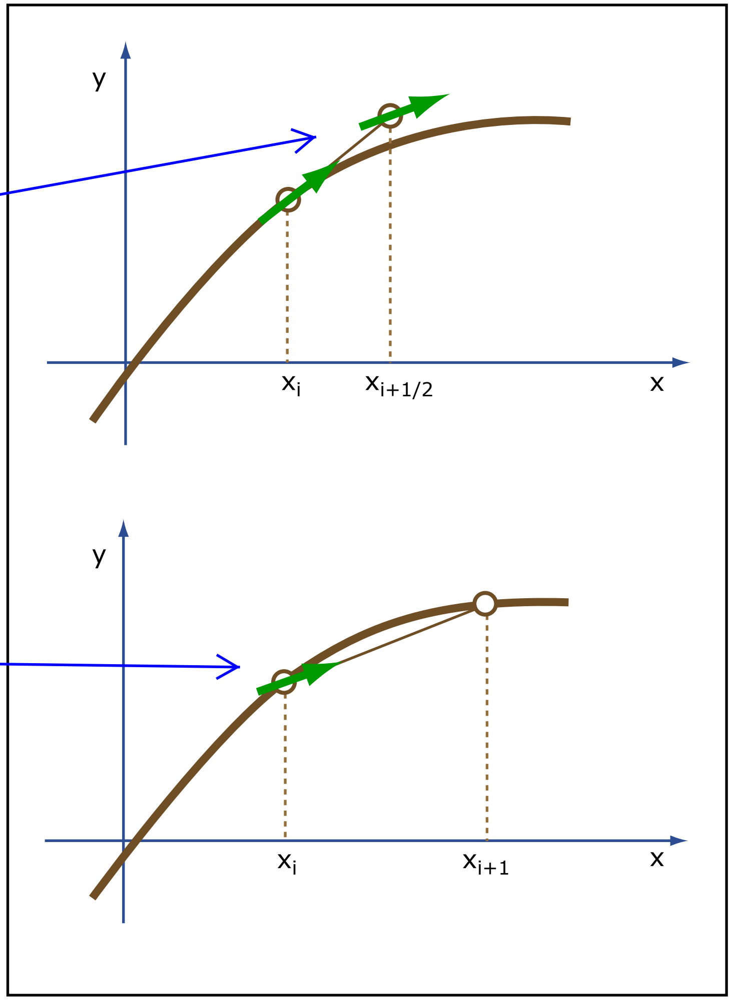



Image by MIT OpenCourseWare.

Comments:

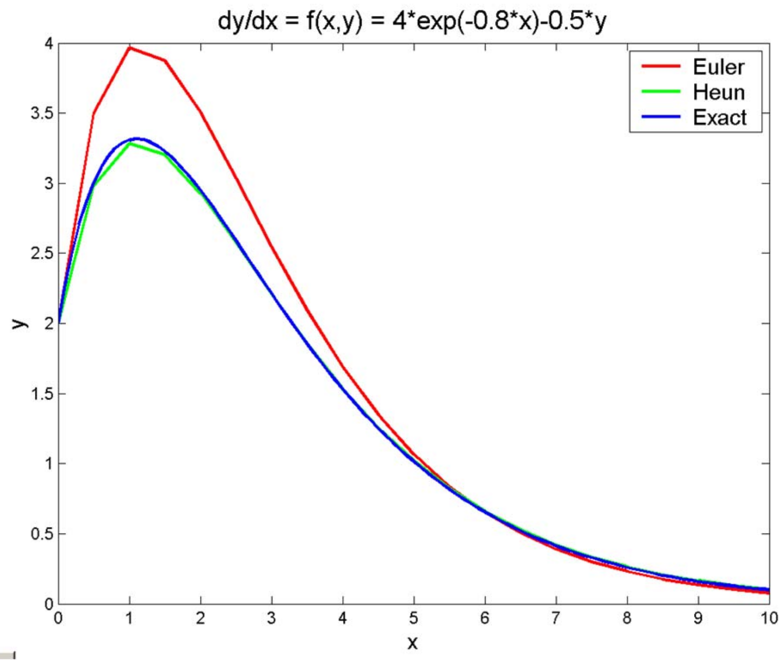
- Midpoint superior to Euler since it uses a centered FD for the first derivative
- Midpoint's global error is of 2nd order: $O(h^2)$

# Initial Value Problems: Heun's method examples

pcm.m

```
func='4*exp(-0.8*x)-0.5*y';
f=inline(func,'x','y');
y0=2;
%step size
h=0.5;
% Euler's method, forward finite difference
xt=[0:h:10];
N=length(xt);
yt=zeros(N,1);
yt(1)=y0;
for n=2:N
    yt(n)=yt(n-1)+h*f(xt(n-1),yt(n-1));
end
hold off
a=plot(xt,yt,'r');
set(a,'Linewidth',2)
% Heun's method
xt=[0:h:10];
N=length(xt);
yt=zeros(N,1);
yt(1)=y0;
for n=2:N
    yt_0=yt(n-1)+h*f(xt(n-1),yt(n-1));
    yt(n)=yt(n-1)+h*(f(xt(n-1),yt(n-1))+f(xt(n),yt_0))/2;
end
hold on
a=plot(xt,yt,'g');
set(a,'Linewidth',2)
% Exact (ode45 Runge Kutta)
x=[0:0.1:10];
hold on
[xrk,yrk]=ode45(f,x,y0);
a=plot(xrk,yrk,'b');
set(a,'Linewidth',2)
```

```
a=title(['dy/dx = f(x,y) = ' func]);
set(a,'Fontsize',16);
a=xlabel('x');
set(a,'Fontsize',14);
a=ylabel('y');
set(a,'Fontsize',14);
a=legend('Euler','Heun','Exact');
set(a,'Fontsize',14);
```




Another example:

$$y' = -2\,x^3 + 12\,x^2 - 20\,x + 8.5$$

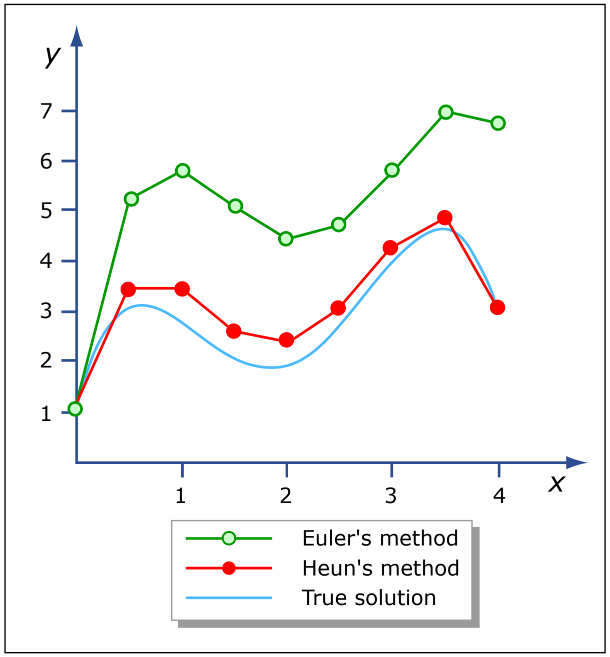



Image by MIT OpenCourseWare.

# Two-level methods for time-integration of (spatially discretized) PDEs

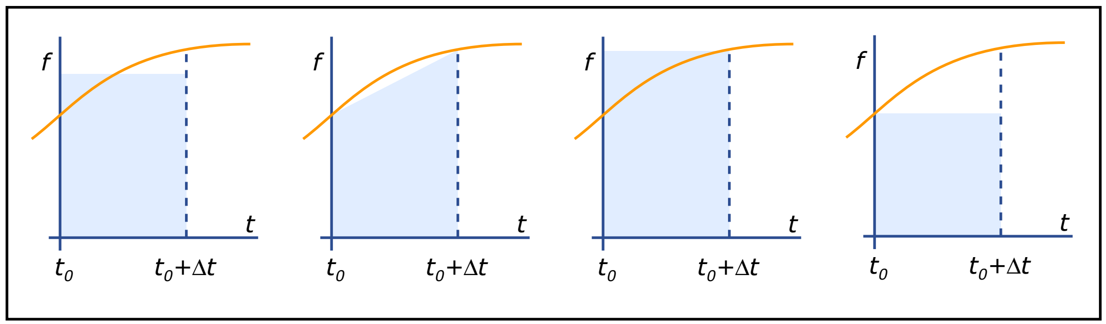
- Four simple schemes to estimate the time integral by approximate quadrature

$$\frac{d\phi}{dt} = f(t,\phi)\;; \quad \text{with } \phi(t_0) = \phi_0 \quad \Leftrightarrow \quad \int_{t_n}^{t_{n+1}} \frac{d\phi}{dt}\,dt = \boxed{\phi^{n+1} - \phi^n = \int_{t_n}^{t_{n+1}} f(t,\phi)\,dt}$$

  - Explicit or Forward Euler: $\phi^{n+1} - \phi^n = f(t_n, \phi^n)\,\Delta t$
  - Implicit or backward Euler: $\phi^{n+1} - \phi^n = f(t_{n+1}, \phi^{n+1})\,\Delta t$
  - Midpoint rule (basis for the leapfrog method): $\phi^{n+1} - \phi^n = f(t_{n+1/2}, \phi^{n+1/2})\,\Delta t$
  - Trapezoid rule (basis for Crank-Nicholson method): $\phi^{n+1} - \phi^n = \frac{1}{2}\left[\, f(t_n, \phi^n) + f(t_{n+1}, \phi^{n+1}) \,\right]\Delta t$

Reminder on global error order:

- Euler methods are of order 1
- Midpoint rule and Trapezoid rule are of order 2
- Order n = truncation error cancels if true solution is polynomial of order n




Graphs showing the approximation of the time integral of f(t) using the midpoint rule, trapezoidal rule, implicit Euler, and explicit Euler methods. Image by MIT OpenCourseWare.

- Some comments
  - All of these methods are two-level methods (involve two times and are at best 2nd order)
  - All excepted forward Euler are implicit methods
  - Trapezoid rule often yields solutions that oscillates, but implicit Euler tends to behave well

# Runge-Kutta Methods and Multistep/Multipoint Methods

$$\phi^{n+1} - \phi^n = \int_{t_n}^{t_{n+1}} f(t,\phi)\, dt$$

- To achieve higher accuracy in time, utilize information (known values of the derivative in time, i.e. the RHS) at more points in time. Two approaches:
- Runge-Kutta Methods:
  - Additional points are between $t_n$ and $t_{n+1}$, and are used strictly for computational convenience
  - Difficulty: n^th order RK requires n evaluation of the first derivative (RHS of PDE) => more expansive as n increases
  - But, for a given order, RK methods are more accurate and more stable than multipoint methods of the same order.
- Multistep/Multipoint Methods:
  - Additional points are at past time steps at which data has already been computed
  - Hence for comparable order, less expansive than RK methods
  - Difficulty to start these methods
  - Examples:
    - Adams Methods: fitting a polynomial to the derivatives at a number of past points in time
    - Lagrangian Polynomial, explicit in time (up to $t_n$): Adams-Bashforth methods
    - Lagrangian Polynomial, implicit in time (up to $t_{n+1}$): Adams-Moulton methods

# Runge-Kutta Methods

## Summary of General Taylor Series Method

$$x_n = a + nh\,,\quad n = 0, 1, \ldots N$$

$$y(x_{n+1}) = y_{n+1} = y_n + hT_k(x_n, y_n) + \frac{h^{k+1}}{(k+1)!}y^{(k+1)}(\xi)$$

where:

$$T_k(x_n, y_n) = f(x_n, y_n) + \frac{h}{2!}f'(x_n, y_n) + \cdots \frac{h^{k-1}}{k!}f^{(k-1)}(x_n, y_n)$$

$$E = \frac{h^{k+1}f^{(k)}(\xi, y(\xi))}{(k+1)!} = \frac{h^{k+1}y^{(k+1)}(\xi)}{(k+1)!}\,,\quad x_n < \xi x_n + h$$

Example: $k = 1$

Euler's method

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

$$E = \frac{h^2}{2!}y''(\xi)$$

Note: expensive to compute higher-order derivatives of *f(x,y)*, especially for spatially discretized PDEs => other schemes needed

**Aim of Runge-Kutta Methods:**

- Achieve accuracy of Taylor Series method without requiring evaluation of higher derivatives of *f(x,y)*
- Obtain higher derivatives using only the values of the RHS (first time derivative)
- Utilize points between $t_n$ and $t_{n+1}$ only

# Initial Value Problems - Time Integrations
# Derivation of 2nd order Runge-Kutta Methods

Taylor Series Recursion:

$$y(x_{n+1}) = y(x_n) + hf(x_n, y_n) + \frac{h^2}{2}(f_x + f f_y)_n + \frac{h^3}{6}(f_{xx} + 2 f f_{xy} + f_{yy} f^2 + f_x f_y + f_y^2 f)_n + O(h^4)$$

Runge-Kutta Recursion:

$$\begin{aligned} y_{n+1} &= y_n + a k_1 + b k_2 \\ k_1 &= h f(x_n, y_n) \\ k_2 &= h f(x_n + \alpha h, y_n + \beta k_1) \end{aligned}$$

Expand $k_2$ in a Taylor series:

$$\begin{aligned} \frac{k_2}{h} &= f(x_n + \alpha h, y_n + \beta k_1) \\ &= f(x_n, y_n) + \alpha h f_x + \beta k_1 f_y \\ &\quad + \frac{\alpha^2 h^2}{2} f_{xx} + \alpha h \beta k_1 f_{xy} + \frac{\beta^2}{2} {k_1}^2 f_{yy}) + O(h^4) \end{aligned}$$

Set $a, b, \alpha, \beta$ to match Taylor series as much as possible.

Substitute $k_1$ and $k_2$ in Runge Kutta

$$y_{n+1} = y_n + (a+b)hf + bh^2(\alpha f_x + \beta f f_y) + bh^3(\frac{\alpha^2}{2} f_{xx} + \alpha\beta f f_{xy} + \frac{\beta^2}{2} f^2 f_{yy}) + O(h^4)$$

Match **2nd order Taylor series**

$$\left.\begin{aligned} a + b &= 1 \\ b\alpha &= 1/2 \\ b\beta &= 1/2 \end{aligned}\right\} \Leftarrow a = b = 0.5 \,, \quad \alpha = \beta = 1$$

We have three equations and 4 unknowns =>

- There is an infinite number of Runge-Kutta methods of 2nd order
- These different 2nd order RK methods give different results if solution is not quadratic
- Usually, number of $k$'s (recursion size) gives the order of the RK method.

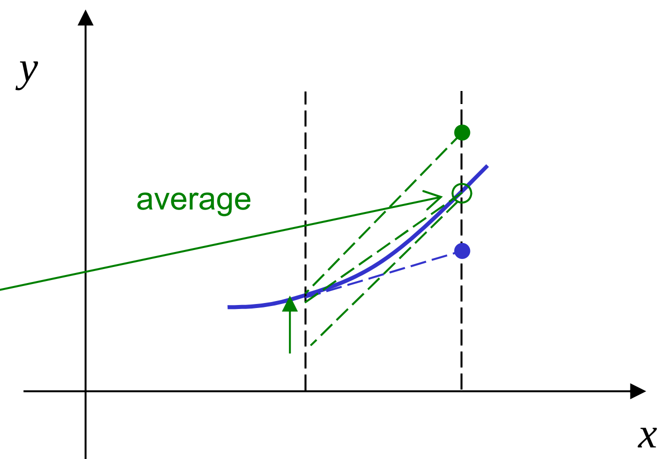
# 4th order Runge-Kutta Methods
## (Most Popular, there is an ∞ number of them, as for 2nd order)

Initial Value Problem:

$$\begin{cases} y' = f(x,y) \\ y(x_0) = y_0 \\ x_n = x_0 + nh \end{cases}$$

2nd Order Runge-Kutta (Heun's version)

$$\begin{aligned} y_{n+1} &= y_n + \tfrac{1}{2}(k_1 + k_2) \\ k_1 &= hf(x_n, y_n) \\ k_2 &= hf(x_n + h, y_n + k_1) \end{aligned}$$




Predictor-corrector method

4th Order Runge-Kutta

$$\begin{aligned} y_{n+1} &= y_n + \tfrac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \\ k_1 &= hf(x_n, y_n) \\ k_2 &= hf(x_n + \tfrac{h}{2}, y_n + \tfrac{k_1}{2}) \\ k_3 &= hf(x_n + \tfrac{h}{2}, y_n + \tfrac{k_2}{2}) \\ k_4 &= hf(x_n + h, y_n + k_3) \end{aligned}$$

Second-order RK methods

| | |
|---|---|
| b = ½, a = ½ : | Heun's method |
| b= 1, a = 0 : | Midpoint method |
| b =2/3, a = 1/3 : | Ralston's Method |

The k's are different estimates of the slope

# $4^{th}$ order Runge-Kutta Example: $\frac{dy}{dx} = x, \quad y(0) = 0$

Forward Euler's Method

$$x_n = nh$$

$$\frac{dy}{dx}|_{x=x_n} \simeq \frac{y_{n+1} - y_n}{h}$$

Forward Euler's Recurrence

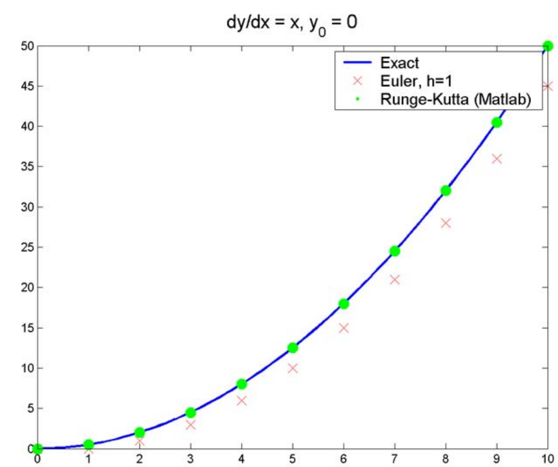
$$y_{n+1} = y_n + hf(nh, y_n)$$

$4^{th}$ Order Runge-Kutta

$$\begin{aligned} y_{n+1} &= y_n + \tfrac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \\ k_1 &= hf(x_n, y_n) \\ k_2 &= hf(x_n + \tfrac{h}{2}, y_n + \tfrac{k_1}{2}) \\ k_3 &= hf(x_n + \tfrac{h}{2}, y_n + \tfrac{k_2}{2}) \\ k_4 &= hf(x_n + h, y_n + k_3) \end{aligned}$$

rk.m

```
h=1.0;
x=[0:0.1*h:10];
y0=0;
y=0.5*x.^2+y0;
figure(1); hold off
a=plot(x,y,'b'); set(a,'Linewidth',2);
% Euler's method, forward finite difference
xt=[0:h:10]; N=length(xt);
yt=zeros(N,1); yt(1)=y0;
for n=2:N
    yt(n)=yt(n-1)+h*xt(n-1);
end
hold on; a=plot(xt,yt,'xr'); set(a,'MarkerSize',12);
% Runge Kutta
fxy='x'; f=inline(fxy,'x','y');
[xrk,yrk]=ode45(f,xt,y0);
a=plot(xrk,yrk,'.g'); set(a,'MarkerSize',30);
a=title(['dy/dx = ' fxy ', y_0 = ' num2str(y0)])
set(a,'FontSize',16);
b=legend('Exact',['Euler, h=' num2str(h)],
'Runge-Kutta (Matlab)'); set(b,'FontSize',14);
```




Matlab ode45 has its own convergence estimation

Note: Matlab inefficient for large problems, but can be used for incubation

# Multistep/Multipoint Methods

- Additional points are at time steps at which data has already been computed
- Adams Methods: fitting a (Lagrange) polynomial to the derivatives at a number of points in time
  - Explicit in time (up to $t_n$): Adams-Bashforth methods

$$\phi^{n+1} - \phi^{n} = \sum_{k=n-K}^{n} \beta_k \; f(t_k, \phi^k)\, \Delta t$$

  - Implicit in time (up to $t_{n+1}$): Adams-Moulton methods

$$\phi^{n+1} - \phi^{n} = \sum_{k=n-K}^{n+1} \beta_k \; f(t_k, \phi^k)\, \Delta t$$

  - Coefficients $\beta_k$'s can be estimated by Taylor Tables:
    - Fit Taylor series so as to cancel higher-order terms

2.29 Numerical Fluid Mechanics
Fall 2011