Numerical Marine Hydrodynamics

- Numerical Differentiation
  - Newton Interpolation
  - Finite Differences
- Ordinary Differential Equations
  - Initial Value Problems
    - Euler's Method
    - Taylor Series Methods
      - Error analysis
    - Predictor-Corrector Methods
    - Runge-Kutta Methods
    - Stiff Differential Equations
    - Multistep Methods
  - Systems of differential equations
  - Boundary Value Problems
    - Shooting method
    - Direct Finite Difference methods

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

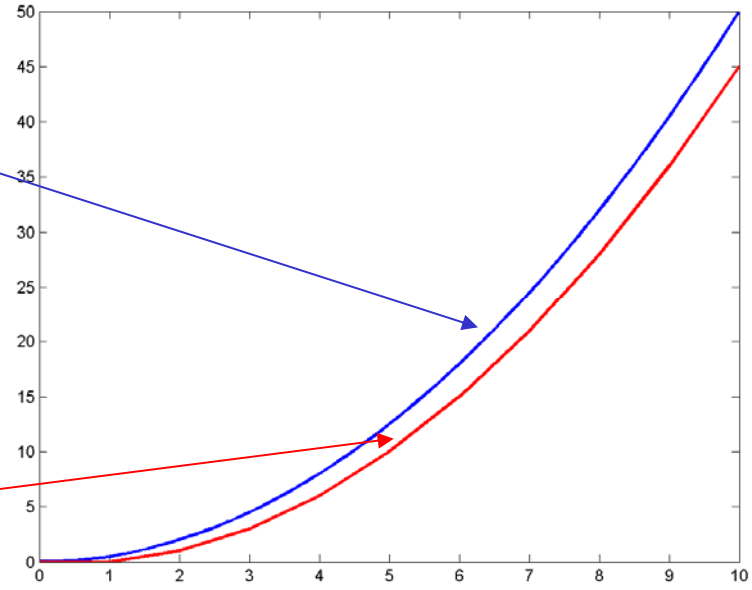
$$x_n = nh$$

Finite Difference (forward)

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

Recurrence

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



euler.m

Initial Value Problems
Predictor-Corrector methods

Initial Slope Estimate

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

Predictor

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

Endpoint Derivative Estimate

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

Average Derivative Estimate

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

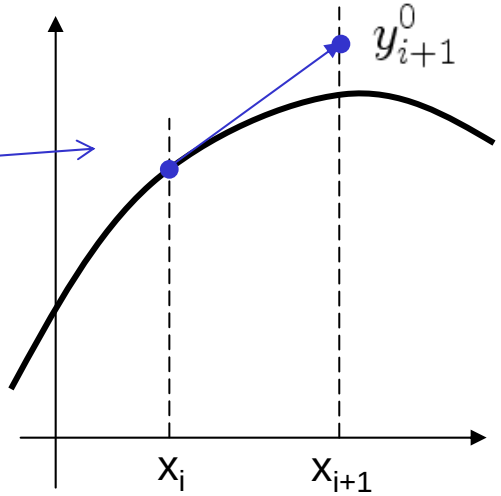
Corrector

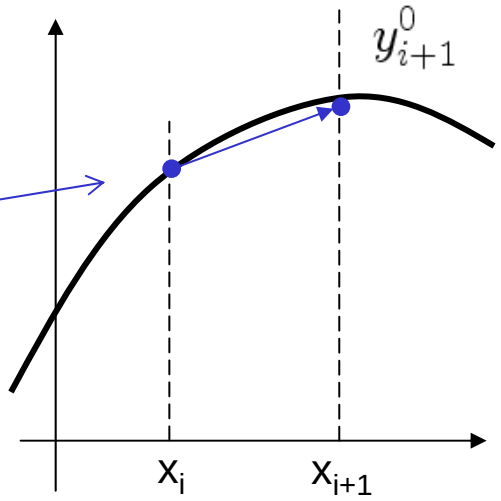
$$y_{i+1} = y_i + \frac{f(x_i, y_i) + f(x_{i+1}, y_{i+1}^0)}{2}h$$

Iterative Heun

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

Heun's Method





Initial Value Problems
Predictor-Corrector methods

Midpoint Method

Midpoint Estimate

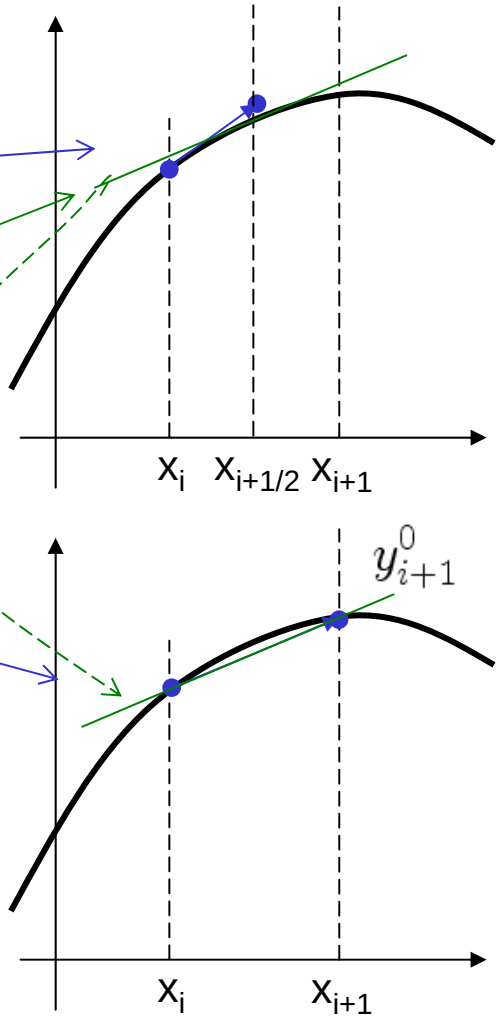
$$y_{i+1/2} = y_i + f(x_i, y_i)\frac{h}{2}$$

Midpoint Derivative Estimate

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

Midpoint Recurrence

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



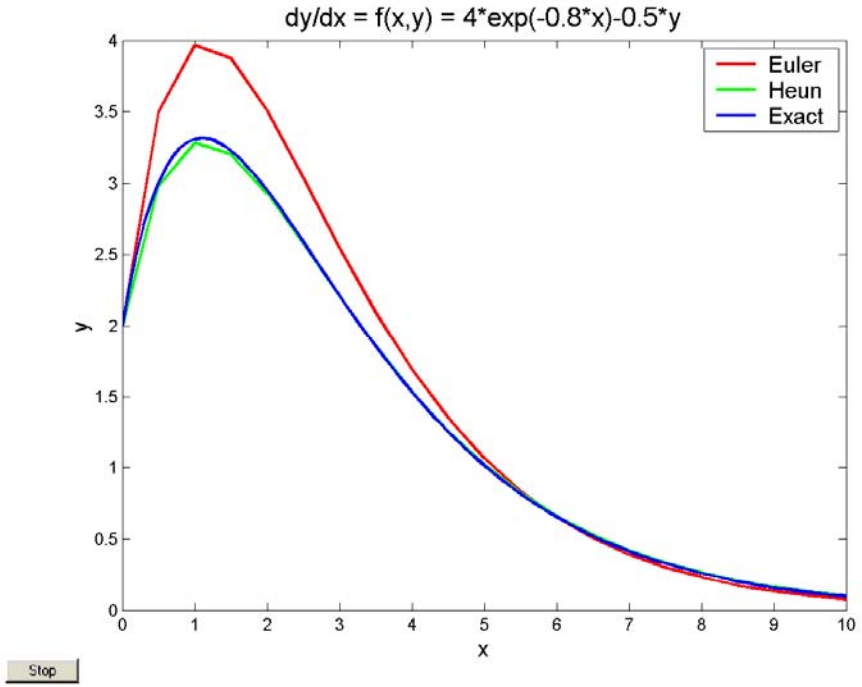
Heun's and Midpoint methods are superior to Euler's method as they use an intermediate estimate of the derivative.

Initial Value Problems
Predictor Corrector Methods

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);
```



Initial Value Problems
Taylor Series Methods

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

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

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

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

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

Partial Derivatives

$$f_x = \frac{\partial}{\partial x}$$

$$f_y = \frac{\partial}{\partial y}$$

Truncate series to *k* terms

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

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

$$\vdots$$

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

Choose Step Size *h*

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

Discretization

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

Recursion Algorithm

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

with

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

Local 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
Runge-Kutta Methods

Taylor Series Recursion

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

Runge-Kutta Recursion

$$y_{n+1} = y_n + ak_1 + bk_2$$
$$k_1 = hf(x_n, y_n)$$
$$k_2 = hf(x_n + \alpha h, y_n + \beta k_1)$$

Match $\boldsymbol{a,b},\alpha,\beta$ to match Taylor series as much as possible

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

Substitute 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**

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

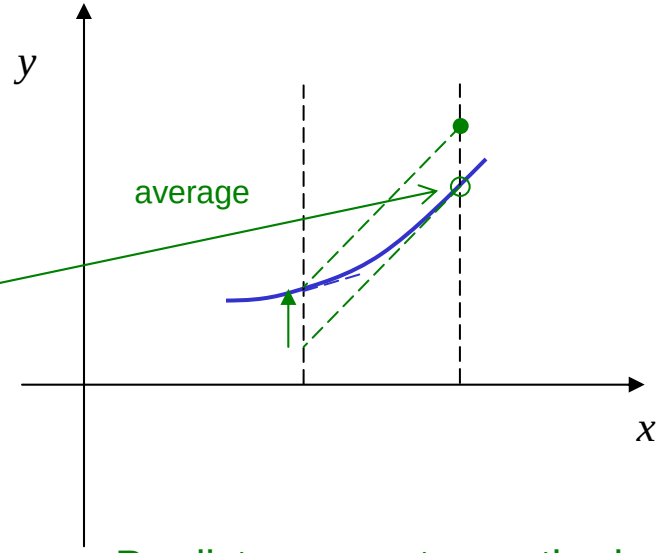
Initial Value Problems
Runge-Kutta Methods

Initial Value Problem

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

2nd Order Runge-Kutta

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



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

Initial Value Problems
Runge-Kutta Methods

Euler's Method

$$x_n = nh$$

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

Recurrence

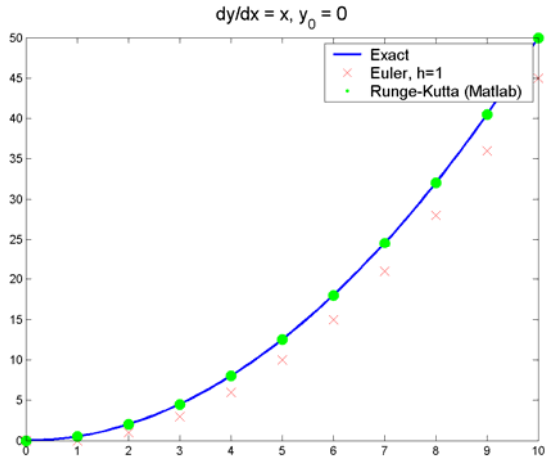
$$y_{n+1} = y_n + hf(nh, y_n)$$

4th Order Runge-Kutta

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

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 automatically ensures convergence

Matlab inefficient for large problems –> Convergence Analysis

Ordinary Differential Equations
Stiff Systems

Stiff IVP

$$\frac{dy}{dt} = -1000y + 3000 - 2000e^{-t}$$

Fast decay ↓ ↓ Slow decay

$$y = 3 - 0.998e^{-1000t} - 2.002e^{-t}$$

Fast term IVP

$$\frac{dy}{dt} = -ay$$

$$y(0) = y_0$$

Solution

$$y = y_0 e^{-at}$$

Explicit Euler

$$y_{i+1} = y_i + \left.\frac{dy}{dt}\right|_i h$$

$$= y_i - a y_i h$$

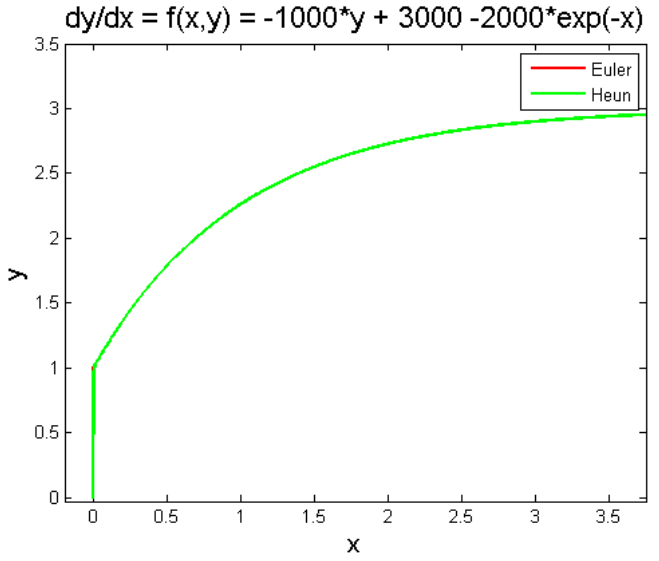
$$= y_i(1 - ah)$$

Stability Requirement

$$h < 2/a$$

stiff.m

```
func='-1000*y + 3000 -2000*exp(-x)';
f=inline(func,'x','y');
y0=0;
%step size
h=0.001;
% 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)
```



Ordinary Differential Equations
Stiff Systems
Implicit Schemes

Implicit Euler

$$\begin{aligned} y_{i+1} &= y_i + \left.\frac{dy}{dt}\right|_{i+1} h \\ &= y_i - a y_{i+1} h \\ &\Rightarrow \\ y_{i+1} &= \frac{y_i}{1+ah} \end{aligned}$$

stiff_imp.m

```
% Implicit Euler method
h=0.05;
xt=[0:h:10];
N=length(xt);
yt=zeros(N,1);
yt(1)=y0;
for n=2:N
    yt(n)=(yt(n-1)+ 3000*h - 2000*h*exp(-x(n)))/(1+1000*h);
end
hold on
a=plot(xt,yt,'g');
set(a,'Linewidth',2)
```

Stiff IVP

$$\frac{dy}{dt} = -1000y + 3000 - 2000e^{-t}$$

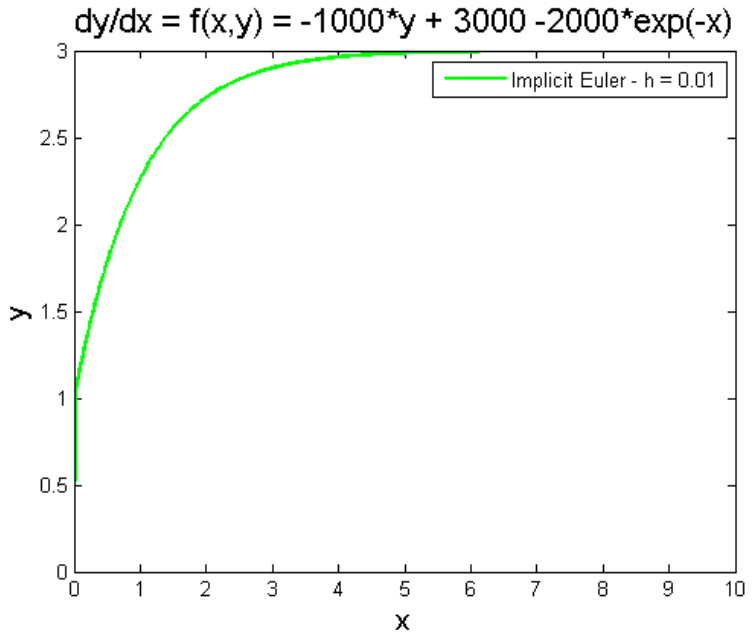
Explicit Euler

$$y_{i+1} = y_i + (-1000y_i + 3000 - 2000e^{-t_i})h$$

$$y_{i+1} = y_i + (-1000y_{i+1} + 3000 - 2000e^{-t_{i+1}})h$$

Implicit Euler

$$y_{i+1} = \frac{y_i + 3000h - 2000e^{-t_{i+1}}}{1+1000h}$$



Ordinary Differential Equations
Multistep Schemes

Heun's Predictor (Euler)

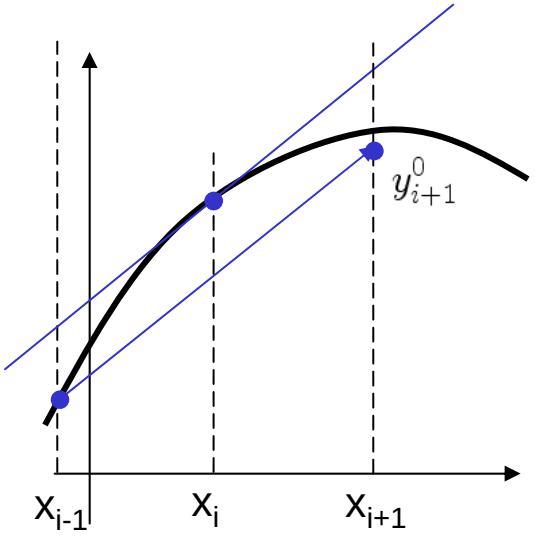
$$y_{i+1}^0 = y_i + f(x_i, y_i)h + O(h^2)$$

Heun's Corrector

$$y_{i+1} = y_i + \frac{f(x_i, y_i) + f(x_{i+1}, y_{i+1}^0)}{2}h + O(h^3)$$

Central Finite Difference

$$\frac{dy}{dx} = \frac{y_{i+1} - y_{i-1}}{2h} + O(h^2)$$



Heun's Non-self-starting Predictor

$$y_{i+1}^0 = y_{i-1} + f(x_i, y_i)2h + O(h^3)$$

Heun's Corrector

$$y_{i+1} = y_i + \frac{f(x_i, y_i) + f(x_{i+1}, y_{i+1}^0)}{2}h + O(h^3)$$

Initial Value Problems
Error Analysis

Initial Value Problem

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

Integrate

$$\int_{y_i}^{y_{i+1}} dy = \int_{x_i}^{x_{i+1}} f(x,y)dx$$

$$y_{i+1} = y_i + \int_{x_i}^{x_{i+1}} f(x,y)dx$$

Trapezoidal Rule

$$\int_{x_i}^{x_{i+1}} f(x,y)dx = \frac{f(x_i,y_i) + f(x_{i+1},y_{i+1})}{2}h + O(h^3)$$

Heun's Corrector

$$y_{i+1} = y_i + \frac{f(x_i,y_i) + f(x_{i+1},y_{i+1})}{2}h + O(h^3)$$

Trapezoidal Rule Error

$$E_c = -\frac{1}{12}h^3 f''(\xi_c)$$

Heun's non-Self-starter Predictor

$$y_{i+1} = y_{i-1} + \int_{x_{i-1}}^{x_{i+1}} f(x,y)dx$$

Mid-point Integration

$$\int_{x_{i-1}}^{x_{i+1}} f(x,y)dx = 2hf(x_i,y_i)$$

Heun's non Self-Starter Predictor

$$y_{i+1} = y_{i-1} + 2hf(x_i,y_i)$$

Mid-point Integration Error

$$E_p = \frac{1}{3}h^3 f''(\xi_p)$$

Initial Value Problems
Error Modifiers

Predictor

$$y(x_{i+1}) = y^0_{i+1} + \frac{1}{3}h^3 y^{(3)}(\xi_p)$$

Same Order

Corrector

$$y(x_{i+1}) = y^m_{i+1} - \frac{1}{12}h^3 y^{(3)}(\xi_c)$$

Subtract

$$0 = y^m_{i+1} - y^0_{i+1} - \frac{5}{12}h^3 y^{(3)}(\xi)$$

$$\frac{y^m_{i+1} - y^0_{i+1}}{5} = -\frac{1}{12}h^3 y^{(3)}(\xi)$$

Corrector Error

$$E_c = \frac{y^m_{i+1} - y^0_{i+1}}{5}$$

$$h^3 y^{(3)}(\xi) = -\frac{12}{5}(y^m_i - y^0_i)$$

Predictor Error

$$E_p = \frac{4}{5}(y^m_i - y^0_i)$$

Predictor Modifier

$$y^0_{i+1} \leftarrow y^0_{i+1} - \frac{4}{5}(y^m_i - y^0_i)$$

Replace by

Corrector Modifier

$$y^m_{i+1} \leftarrow y^m_{i+1} - \frac{y^m_{i+1} - y^0_{i+1}}{5}$$