$$k_i = f\left(t_n + c_i h, y_n + h\sum_{j=1}^{s} a_{ij} k_j\right)$$

Each method listed on this page is def
the method in a table as follows:

$$\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \dots & a_{1s} \\ c_2 & a_{21} & a_{22} & \dots & a_{2s} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_s & a_{s1} & a_{s2} & \dots & a_{ss} \\ \hline & b_1 & b_2 & \dots & b_s \end{array}$$

Butcher Tablea for the method from lab 5 also called RK2 or IE2.

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1 & 1 & 0 \\ \hline & 1/2 & 1/2 \end{array}$$

also called..,

Heun's method [edit]

Heun's method is a second-order m
improved Euler's method, or modifie

$$k_1 = f\left(t_n + c_1 h, y_n + h\sum_{j=1}^{s} a_{ij} k_j\right)$$

0 0

$$K_1 = f(t_n, y_n)$$

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1 & 1 & 0 \\ \hline & 1/2 & 1/2 \end{array}$$

$$\begin{array}{c|cc} c_1 & a_{11} & a_{12} \\ c_2 & a_{21} & a_{22} \\ \hline & b_1 & b_2 \end{array}$$

$$k_2 = f\left(t_n + c_2 h, y_n + h\sum_{j=1}^{s} a_{2j} k_j\right)$$

1

$a_{21} = 1$ $a_{22} = 0$

diagonal is zero and everything above it zero then the method is explicit.

Means don't have to invert the f function...

$$k_2 = f(t_n + h, y_n + hk_1)$$

Therefore...

① $k_1 = f(t_n, y_n)$

② $k_2 = f(t_n + h, y_n + hk_1)$

$$y_{n+1} = y_n + h\sum_{i=1}^{s} b_i k_i$$

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

```
function rk2(tn,yn)
    k1=f(tn,yn)
    k2=f(tn+h,yn+h*k1)
    return yn+0.5*h*(k1+k2)
end
```

Explicit midpoint method.

same as before

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1/2 & 1/2 & 0 \\ \hline & 0 & 1 \end{array}$$

$k_1 = f(t_n, y_n)$

$k_2 = f(t_n + \frac{1}{2}h, y_n + h\frac{1}{2}k_1)$

$$k_i = f\left(t_n + c_i h, y_n + h\sum_{j=1}^{s} a_{ij} k_j\right)$$

$a_{21} = \frac{1}{2}$ $a_{22} = 0$

$k_1 = f(t_n, y_n)$

$k_2 = f(t_n + \frac{1}{2}h, y_n + h\frac{1}{2}k_1)$

$y_{n+1} = y_n + hk_2$

Explicit midpoint method

Kutta's third-order method

0	0	0	0
1/2	1/2	0	0
1	−1	2	0
	1/6	2/3	1/6

$k_1 = f(t_n, y_n)$

$k_2 = f(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_1)$

$k_3 = f(t_n + h, y_n - hk_1 + 2hk_2)$

note $-1+2 = 1$

generally $c_i = \sum_{j=1}^{s} a_{ij}$

$y_{n+1} = y_n + h(\frac{1}{6}k_1 + \frac{2}{3}k_2 + \frac{1}{6}k_3)$

Heun's third-order method

0	0	0	0
1/3	1/3	0	0
2/3	0	2/3	0
	1/4	0	3/4

always adds to 1..

$k_1 = f(t_n, y_n)$

$k_2 = f(t_n + \frac{h}{3}, y_n + \frac{h}{3}k_1)$

$k_3 = f(t_n + \frac{2}{3}h, y_n + \frac{2}{3}hk_2)$

$y_{n+1} = y_n + h(\frac{1}{4}k_1 + \frac{3}{4}k_3)$

Can reuse memory for k_1 when computing k_2 and reuse k_2 to compute k_3.

```
# Order 3 methods
function RK3(tn,yn)
    k1=f(tn,yn)
    k2=f(tn+0.5*h,yn+0.5*h*k1)
    k3=f(tn+h,yn-h*k1+2*h*k2)
    return yn+h*(k1/6+2*k2/3+k3/6)
end
function Heun3(tn,yn)
    k1=f(tn,yn)
    k2=f(tn+h/3,yn+h*k1/3)
    k3=f(tn+(2.0/3.0)*h,yn+(2.0/3.0)*h*k2)
    return yn+h*(0.25*k1+0.75*k3)
end
```

Now let's test the methods...

Test problem:

Idea start with the answer and then make an ODE that has that function as it's answer...

$y' = f(t,y) \qquad y(t_0) = y_0 \qquad$ Initial value problem.

Suppose I wanted the answer to be $y = g(t)$ for some function $g(t)$ that can specify...

Simple idea: Let $f(t,y) = g'(t)$ and $y(t_0) = g(t_0)$

then the ODE is

$$y' = g'(t) \quad \text{and} \quad y(t_0) = g(t_0)$$

too simple test problem its only integration...

plug in $y = g(t)$ and this is a solution, since solutions are unique that's the only solution.

More complicated $f(t,y)$

$$y' = \varphi'(t) + N(t, y - \varphi(t)) \qquad \text{and} \qquad y(t_0) = \varphi(t_0)$$

where $N(t,0) = 0$ for all t.

means that $N(t, y - \varphi(t)) = 0$ when $y = \varphi(t)$

Test problem

<u>Example</u> $\varphi(t) = \sin t$ $\qquad N(t,z) = z^2$

$\varphi'(t) = \cos t$ $\qquad N(t, y - \varphi(t)) = (y - \sin t)^2$

$t_0 = 0$

$$y' = \cos t + (y - \sin t)^2 \quad \text{where} \quad y(0) = \sin(0) = 0$$

Test problem

<u>Example</u> $\varphi(t) = \sin t$ $\qquad N(t,z) = \sin z$

$\varphi'(t) = \cos t$ $\qquad N(t, y - \varphi(t)) = \sin(y - \sin t$

$t_0 = 0$

$$y' = \cos t + \sin(y - \sin t) \quad \text{where} \quad y(0) = \sin(0) = 0$$

Do the numerics to see the different rates of convergence, but since that's not of the final (final is only theoretical) then I'll do the calculation outside of lecture and add that to this discussion after class...

After class...

```julia
f(t,y)=cos(t)+(y-sin(t))^2
t0=0.0; y0=0.0
T=7.0
Ns=(2).^(1:10)
hs=(T-t0)./Ns
Es=zeros(length(hs))
for j=1:10
    global h=hs[j]
    yn=y0
    for n=0:Ns[j]-1
        tn=t0+n*h
        yn=Heun3(tn,yn)
    end
    Es[j]=abs(yn-sin(T))
    println("y(T)=",sin(T)," yN=",yn," EN=",Es[j])
end
```

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

specify initial condition

approximate solution on $[t_0, T]$,

Time steps

size of time steps

The error $|y(T) - y_N|$

need to be global so the time stepping methods know the time step

compute N time steps

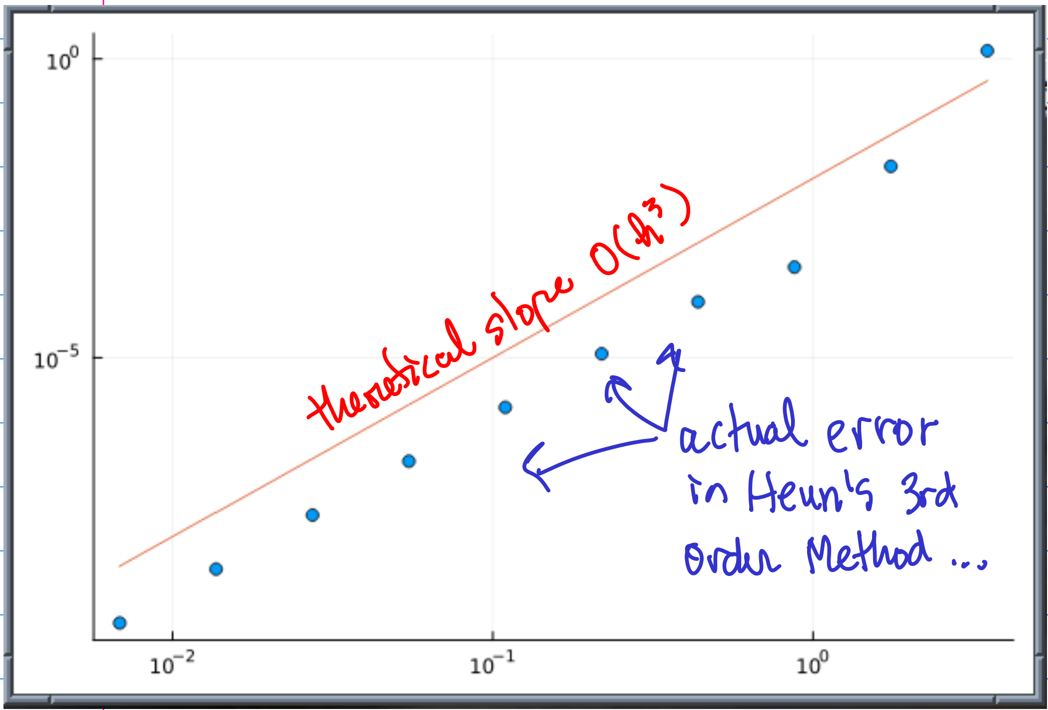
exact

approx

error

```julia
julia> scatter(hs,Es,xscale=:log10,yscale=:log10,legend=false)
```

```julia
julia> plot!(h->1e-2*h^3,hs)
```



Slopes look pretty much the same ... So it works!

Same computation for RK3

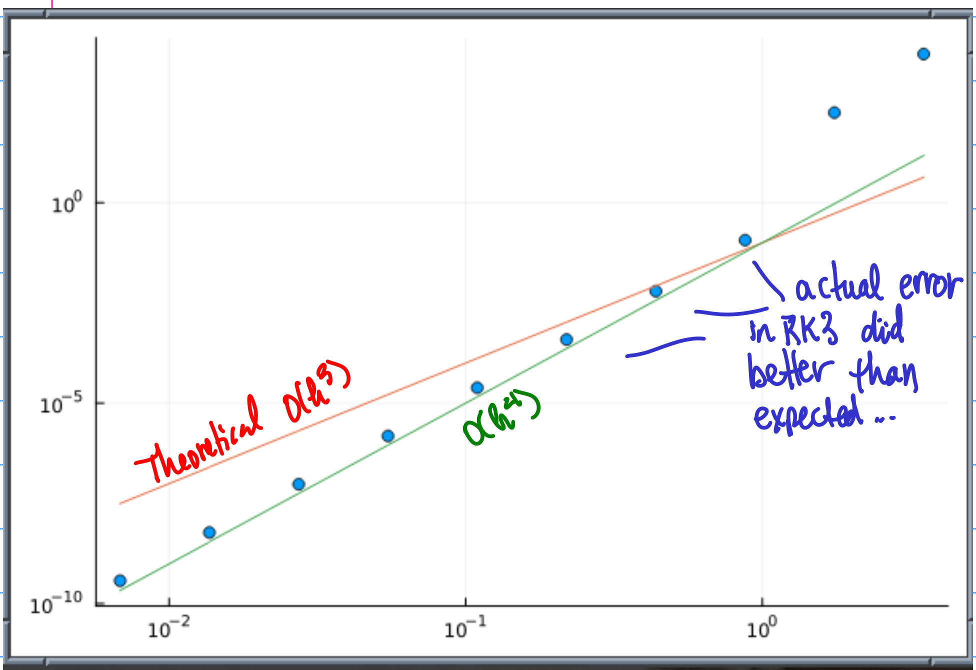
```
f(t,y)=cos(t)+(y-sin(t))^2
t0=0.0; y0=0.0
T=7.0
Ns=(2).^(1:10)
hs=(T-t0)./Ns
Es=zeros(length(hs))
for j=1:10
    global h=hs[j]
    yn=y0
    for n=0:Ns[j]-1
        tn=t0+n*h
        yn=RK3(tn,yn)
    end
    Es[j]=abs(yn-sin(T))
    println("y(T)=",sin(T)," yN=",yn," EN=",Es[j])
end
```

only line that changed

```
julia> scatter(hs,Es,xscale=:log10,yscale=:log10,legend=false)

julia> plot!(h->1e-1*h^3,hs)

julia> plot!(h->1e-1*h^4,hs)
```



For this test problem RK3 method performed as $O(h^4)$ which is better than expected...

lucky!

Try the other test problem.

```
f(t,y)=cos(t)+sin(y-sin(t))
t0=0.0; y0=0.0
T=7.0
Ns=(2).^(1:10)
hs=(T-t0)./Ns
Es=zeros(length(hs))
for j=1:10
    global h=hs[j]
    yn=y0
    for n=0:Ns[j]-1
        tn=t0+n*h
        yn=RK3(tn,yn)
    end
    Es[j]=abs(yn-sin(T))
    println("y(T)=",sin(T)," yN=",yn," EN=",Es[j])
end
```

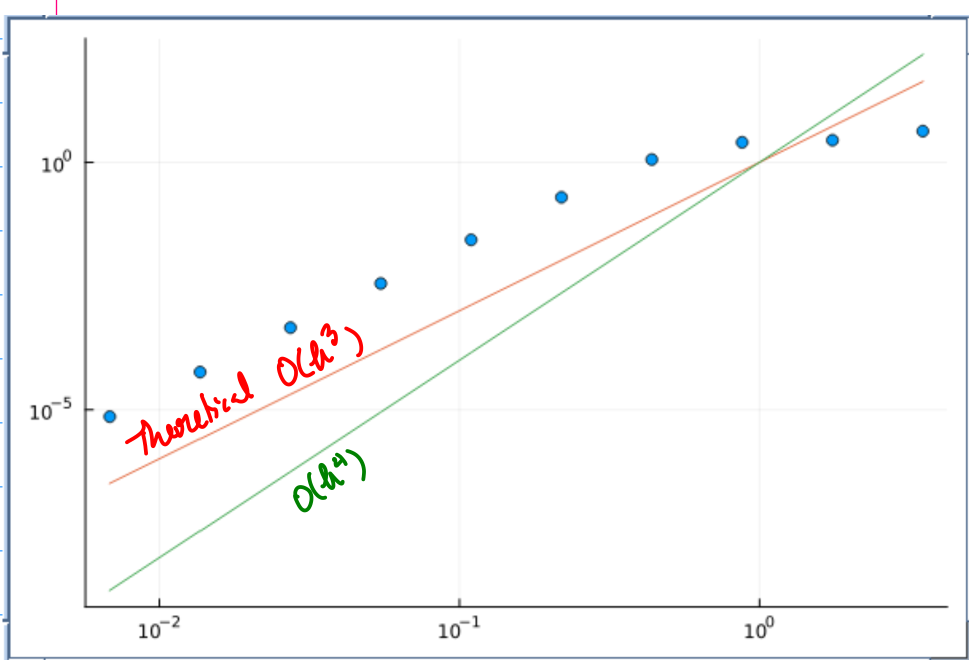
← Now solve $y'=\cos t+\sin(y-\sin(t))$

exact solution is still the same

```
julia> scatter(hs,Es,xscale=:log10,yscale=:log10,legend=false)

julia> plot!(h->h^3,hs)

julia> plot!(h->h^4,hs)
```



Not so lucky this time... The RK3 method performs as expected as $O(h^3)$.