$$y' = y^2\cos(t) \qquad y(0) = 0.8$$

For next week find the exact solution of this ODE.

$$\frac{dy}{dt} = y^2 \cos t$$

$$\int_{y_0}^{y} \frac{dy}{y^2} = \int_{t_0}^{t} \cos t \, dt = \sin t \Big|_{t_0}^{t} = \sin t - \sin t_0$$

$$\Rightarrow \left. \frac{-1}{y} \right|_{y_0}^{y} = -\frac{1}{y} + \frac{1}{y_0}$$

Therefore

$$-\frac{1}{y} + \frac{1}{y_0} = \sin t - \sin t_0$$

$$\frac{1}{y} = \frac{1}{y_0} + \sin t_0 - \sin t$$

$$y = \frac{1}{\frac{1}{y_0} + \sin t_0 - \sin t}$$

Since $y(0) = 0.8$ then

$t_0 = 0$

$y_0 = 0.4$

$$y = \frac{1}{\frac{1}{0.8} - \sin t}$$

Write a Julia program to use trapezoid method:

```
yexact(t)=1/(1/y0+sin(t0)-sin(t))
f(t,y)=y^2*cos(t)
dfdy(t,y)=2*y*cos(t)
phi(z)=yn+h/2*(f(tn,yn)+f(tn+h,z))-z
dphi(z)=h/2*dfdy(tn+h,z)-1
g(z)=z-phi(z)/dphi(z)
```

functions following notation from last week

$$\varphi(z) = y_n + \frac{h}{2}\left(f(t_n,y_n) + f(t_{n+1}, z)\right) - z$$

Newtons method.

$$g(z) = z - \frac{\varphi(z)}{\varphi'(z)}$$

Initial conditions and other parameters:

```
y0=0.8    # initial condition
t0=0      # initial time
T=2       # final time
N=32      # number of steps
h=(T-t0)/N  # size of step
```

Numerical integrator

```
yn=y0
tn=t0
for n=1:N
    global yn,tn
    local zn
    # Newton's method
    zn=yn
    for i=1:3
        zn=g(zn)
    end
    # trapezoid step
    yn=yn+h/2*(f(tn,yn)+f(tn+h,zn))
    tn=t0+n*h
end
```

could take more steps to solve for z

The stability of the implicit method is also limited by the stability of the non-linear solver.

Backup the working program...

```
$ cp trap.jl trap-01.jl
```

Now change it to check order of convergence...

refactor the loop into a function

```
function trapsolve(y0,t0,T,N)
    global yn,tn,h
    h=(T-t0)/N  # size of step
    yn=y0
    tn=t0
    for n=1:N
        local zn
        # Newton's method
        zn=yn
        for i=1:3
            zn=g(zn)
        end
        # trapezoid step
        yn=yn+h/2*(f(tn,yn)+f(tn+h,zn))
        tn=t0+n*h
    end
    return yn
end
```

Add a new loop to check convergence

```
for j=1:7
    N=2^(j+2) # number of steps
    yapp1=trapsolve(y0,t0,T,N)
    yapp2=trapsolve(y0,t0,T,2*N)
    err1=yapp1-yexact(T)
    err2=yapp2-yexact(T)
    println(N," ",yapp2," ",yapp2-yexact(T))
    println("err1/err2=",err1/err2)
end
```

Finally check stability... since the idea of trapezoid and implicit methods in general is increased stability.

```
function trapsolve(y0,t0,T,N)
    local Ys=zeros(N)
    local Ts=zeros(N)
    global yn,tn,h
    h=(T-t0)/N  # size of step
    yn=y0
    tn=t0
    for n=1:N
        local zn
        # Newton's method
        zn=yn
        for i=1:3
            zn=g(zn)
        end
        # trapezoid step
        yn=yn+h/2*(f(tn,yn)+f(tn+h,zn))
        # euler step
#       yn=yn+h*f(tn,yn)
        tn=t0+n*h
        Ys[n]=yn
        Ts[n]=tn
    end
    return Ts,Ys
end
```

return entire trajectory

plot it

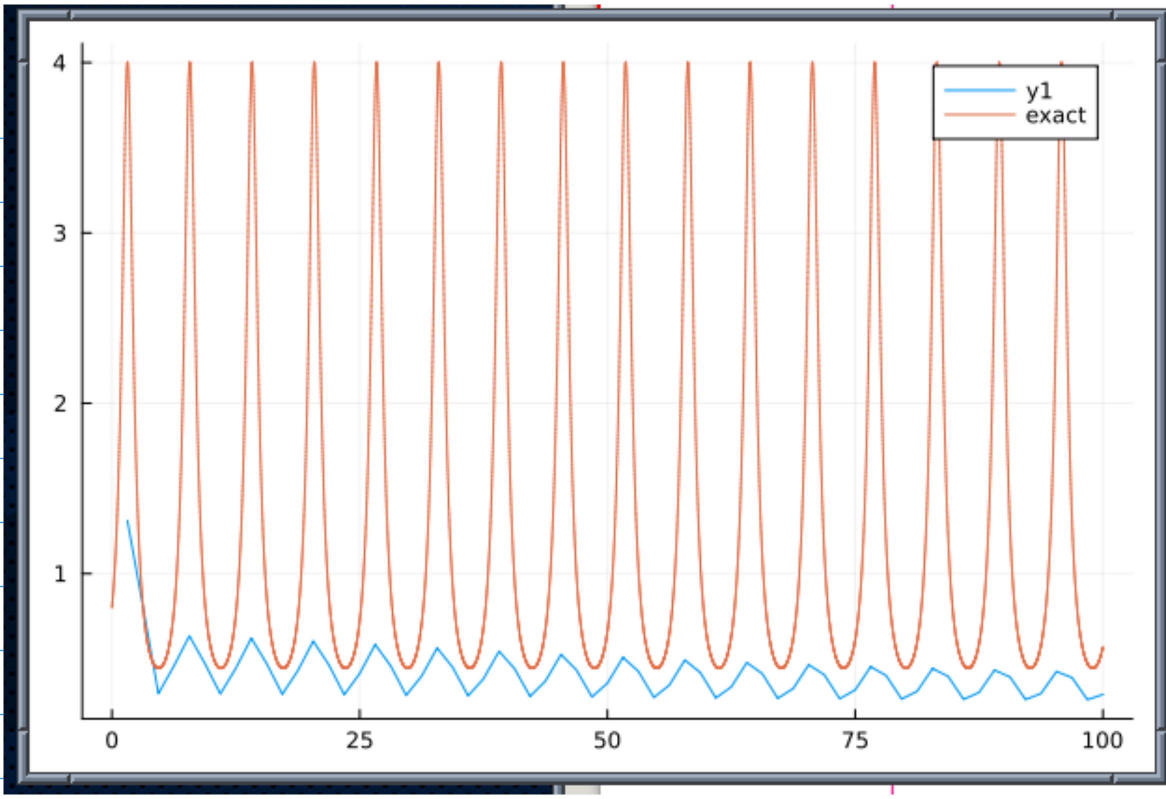
```
Ts,Ys=trapsolve(y0,t0,T,64)
plot(Ts,Ys)
```

compare to exact

```
julia> include("trap.jl")

julia> plot!(t0:0.01:T,yexact.(t0:0.01:T),label="exact")
```

The graph is



not very accurate ... but picks up the periodicity and didn't blow up to ∞.

<u>Note</u> Euler with same step size is unstable...