Math/CS 467/667 Programming Assignment 1 Solution Key

1. The Lorenz system is a three dimensional ordinary differential equation of the form

$$\frac{dy}{dt} = f(y)$$

which a given initial condition $y(0) = a$ where $y(t)$ is a vector in $\mathbf{R}^3$ and

$$f(y) = \begin{bmatrix} -10y_1 + 10y_2 \\ 28y_1 - y_2 - y_1y_3 \\ y_1y_2 - (8/3)y_3 \end{bmatrix}.$$

Let Y^n be an approximation of $y(1)$ obtained using a step size of $h = 1/n$. Define the error

$$E_n = \|Y^n - y(1)\| = \left\{ \sum_{i=1}^{3} \left(Y_i^n - y_i(1)\right)^2 \right\}^{1/2}.$$

Show that if $E_n \le Kh^k$ then

$$\|Y^n - Y^{2n}\| \le K \left\{ 1 + \frac{1}{2^k} \right\} h^k.$$

Since $h = 1/n$ then

$$E_{2n} \le K\Big(\frac{1}{2n}\Big)^k = K\frac{h^k}{2^k}.$$

It follows that

$$\|Y^n - Y^{2n}\| \le \|Y^n - y(1)\| + \|y(1) - Y^{2n}\| \le Kh^k + K\frac{h^k}{2^k} = K \left\{ 1 + \frac{1}{2^k} \right\} h^k.$$

2. Write a program to approximate solutions of the Lorenz system using Euler's forward difference method and the initial condition

$$a = \begin{bmatrix} 2 \\ 3 \\ 15 \end{bmatrix}.$$

Compute Y^n for $n = 64, 128, 256, 512, \ldots, 65536$.

The C code is given by

```
1 #  prog2b.jl -- integration of a chaotic non-linear system
2
3 using Printf, LinearAlgebra
4
5 function f(y)
6     r=[10*(y[2]-y[1]),
```

```
        (28.0-y[3])*y[1]-y[2],
        y[1]*y[2]-(8/3)*y[3]]
    return r
end

function euler(t,y,h)
    return y+h*f(y)
end

function solve(t0,y0,h,n)
    yn=copy(y0)
    for j=1:n
        tn=t0+(j-1)*h
        yn=euler(tn,yn,h)
    end
    return yn
end

function main()
    nres=11
    t0=0.0; T=1.0
    y0=[2.0,3.0,15.0]
    yold=zeros(3)
    @printf("#%4s %19s %19s %19s\n","n","y1","y2","y3")
    N=Vector{Int}(undef,nres)
    Y=Vector{Vector{Float64}}(undef,nres)
    n=64
    for i=1:nres
        h=T/n
        yn=solve(t0,y0,h,n)
        N[i]=n
        Y[i]=copy(yn)
        @printf("%5d %19.14f %19.14f %19.14f\n",
            n,yn[1],yn[2],yn[3])
        n*=2
    end

    @printf("\n\n#%4s %18s %20s %20s\n",
        "n","|Yn-Y2n|","log(h)","log(|Yn-Y2n|)")
    for i=2:nres
        e=norm(Y[i-1]-Y[i])
        @printf("%5d %18.15f %20.13e %20.13e\n",
            N[i],e,-log(N[i]),log(e))
    end
end
```

```
52
53 main()
```

which generates output

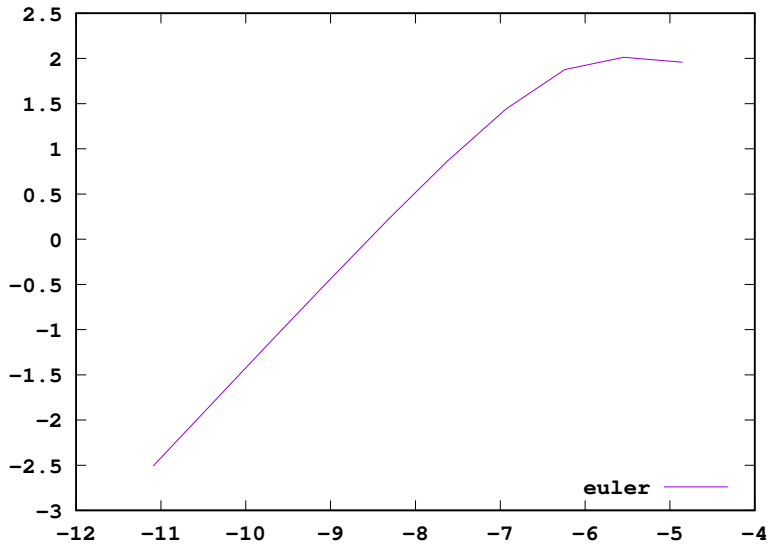
```
#   n                y1                  y2                  y3
   64  -9.69978827124415   -7.78996746984789   30.72466119973522
  128 -13.37234518640962  -12.80293797386648   34.13472724313560
  256 -13.33684881600047  -17.62699523674917   28.40582877598391
  512 -11.62086377869534  -17.96721751135157   22.10580958772410
 1024 -10.13074666717946  -16.77108440528200   18.33157199996538
 2048  -9.21363702282563  -15.72373591985231   16.41361053841384
 4096  -8.71230404053936  -15.07933273503957   15.47039862337052
 8192  -8.45114509961543  -14.72647238779146   15.00610041427197
16384  -8.31797998332034  -14.54236242834484   14.77622038563917
32768  -8.25075649800784  -14.44838864670079   14.66190400794265
65536  -8.21698526021469  -14.40092239534972   14.60490883434304


#   n          |Yn-Y2n|               log(h)        log(|Yn-Y2n|)
  128  7.088448207445854 -4.8520302639196e+00  1.9584664459884e+00
  256  7.489530433532963 -5.5451774444796e+00  2.0135061031117e+00
  512  6.538394116052697 -6.2383246250395e+00  1.8776915873131e+00
 1024  4.230372652889907 -6.9314718055995e+00  1.4422900867877e+00
 2048  2.369937787726844 -7.6246189861594e+00  8.6286370489822e-01
 4096  1.247493062267095 -8.3177661667193e+00  2.2113598731913e-01
 8192  0.638973586352901 -9.0109133472793e+00 -4.4789216136991e-01
16384  0.323224771528324 -9.7040605278392e+00 -1.1294073106920e+00
32768  0.162537081380518 -1.0397207708399e+01 -1.8168491101336e+00
65536  0.081498413070771 -1.1090354888959e+01 -2.5071717304492e+00
```

3. Graph $\log \|Y^n - Y^{2n}\|$ versus $\log h$ to verify the order of convergence for Euler's method numerically.

Plot the second table in the previous output using `gnuplot` and the script

```
1 set terminal postscript enhanced color eps font "Courier-Bold"
2 set output '2b.eps'
3 set style data lines
4 set size 0.7,0.7
5 set key spacing 1.2
6 set key bottom
7 plot "prog2b.out" index 1 using 3:4 ti "euler"
```

to obtain the graph



The graph has a linear slope for $\log h \in [-11, -8]$. Computing this slope obtains

$$\frac{0.22113598731911 + 2.5071717304524}{-8.3177661667193 + 11.090354888959} \approx 0.9840290026$$

which verifies the order of convergence is 1.

4. Compute Y^n using Runge-Kutta methods of orders 2 and 4 and verify the order of convergence by graphing $\log \|Y^n - Y^{2n}\|$ versus $\log h$.

The C code is given by

```
#  prog2d.jl -- integration of a chaotic non-linear system

using Printf, LinearAlgebra

function f(y)
    r=[10*(y[2]-y[1]),
       (28.0-y[3])*y[1]-y[2],
       y[1]*y[2]-(8/3)*y[3]]
    return r
end

function euler(t,y,h)
    return y+h*f(y)
end

function rk2(t,y,h)
    k1=f(y)
    k2=f(y+h*2/3*k1)
    return y+h*(1/4*k1+3/4*k2)
```

```
20 end
21
22 function rk3(t,y,h)
23     k1=f(y)
24     k2=f(y+h*2/3*k1)
25     k3=f(y+h*2/3*k2)
26     return y+h*(1/4*k1+3/8*k2+3/8*k3)
27 end
28
29 function rk4(t,y,h)
30     k1=f(y)
31     k2=f(y+h*1/2*k1)
32     k3=f(y+h*1/2*k2)
33     k4=f(y+h*k3)
34     return y+h*(1/6*k1+1/3*k2+1/3*k3+1/6*k4)
35 end
36
37 function solve(step,t0,y0,h,n)
38     yn=copy(y0)
39     for j=1:n
40         tn=t0+(j-1)*h
41         yn=step(tn,yn,h)
42     end
43     return yn
44 end
45
46 function main()
47     nres=11
48     t0=0.0; T=1.0
49     y0=[2.0,3.0,15.0]
50     for k=1:3
51         step=[rk2,rk3,rk4][k]
52         name=["RK2","RK3","RK4"][k]
53         @printf("#%s method computing y(%g)\n",name,T)
54         @printf("#%4s %19s %19s %19s\n","n","y1","y2","y3")
55         N=Vector{Int}(undef,nres)
56         Y=Vector{Vector{Float64}}(undef,nres)
57         n=64
58         for i=1:nres
59             h=T/n
60             yn=solve(step,t0,y0,h,n)
61             N[i]=n
62             Y[i]=copy(yn)
63             @printf("%5d %19.14f %19.14f %19.14f\n",
64                 n,yn[1],yn[2],yn[3])
```

```
            n*=2
        end
        @printf("\n\n#%4s %18s %20s %20s\n",
            "n","|Yn-Y2n|","log(h)","log(|Yn-Y2n|)")
        for i=2:nres
            e=norm(Y[i-1]-Y[i])
            @printf("%5d %18.15f %20.13e %20.13e\n",
                N[i],e,-log(N[i]),log(e))
        end
        if k<3
            @printf("\n\n")
        end
    end
end

main()
```

which generates output

```
#RK2 method computing y(1)
#   n                 y1                   y2                   y3
   64   -8.72875463905269  -15.13208695329234   15.43075772360616
  128   -8.26393985593300  -14.47405313305299   14.66662724329503
  256   -8.19623728104049  -14.37346148741356   14.56556398236884
  512   -8.18550468849651  -14.35696971618285   14.55091208631216
 1024   -8.18359646978421  -14.35393877226764   14.54856006469860
 2048   -8.18321626911562  -14.35331771642115   14.54813548006970
 4096   -8.18313332534885  -14.35317954274479   14.54804974656670
 8192   -8.18311410273340  -14.35314713587289   14.54803086418686
16384   -8.18310948624878  -14.35313930123911   14.54802646243931
32768   -8.18310835577508  -14.35313737596859   14.54802540185868
65536   -8.18310807611255  -14.35313689882448   14.54802514169531


#   n           |Yn-Y2n|               log(h)        log(|Yn-Y2n|)
  128  1.110385825794761 -4.8520302639196e+00  1.0470754573464e-01
  256  0.157848346613323 -5.5451774444796e+00 -1.8461205384497e+00
  512  0.024532531852560 -6.2383246250395e+00 -3.7077552115023e+00
 1024  0.004284848345251 -6.9314718055995e+00 -5.4526701195656e+00
 2048  0.000842932393461 -7.6246189861594e+00 -7.0786238007335e+00
 4096  0.000182542780806 -8.3177661667193e+00 -8.6085259970367e+00
 8192  0.000042145682565 -9.0109133472793e+00 -1.0074378309001e+01
16384  0.000010102910383 -9.7040605278392e+00 -1.1502687018938e+01
32768  0.000002471733932 -1.0397207708399e+01 -1.2910590656837e+01
65536  0.000000611197687 -1.1090354888959e+01 -1.4307845383653e+01


#RK3 method computing y(1)
#   n                 y1                   y2                   y3
   64   -8.03534813005494  -14.14360184797095   14.30026158194662
  128   -8.16432904920729  -14.32667620459421   14.51628572158876
  256   -8.18074901868859  -14.34981494717387   14.54403677669892
```

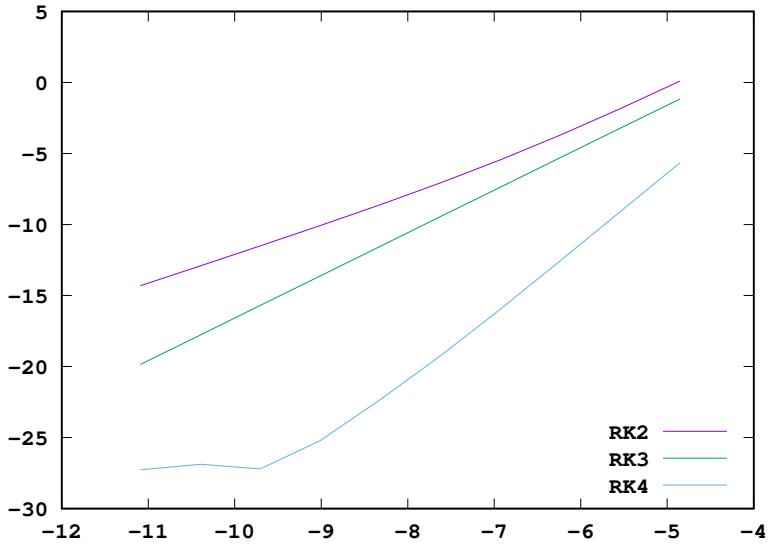
```
  512   -8.18281269779995  -14.35272092626652   14.54752597179119
 1024   -8.18307105759800  -14.35308473992388   14.54796265793439
 2048   -8.18310336712924  -14.35313023946297   14.54801725604786
 4096   -8.18310740638749  -14.35313592788291   14.54802408094948
 8192   -8.18310791131894  -14.35313663898117   14.54802493405083
16384   -8.18310797443702  -14.35313672787150   14.54802504068808
32768   -8.18310798232706  -14.35313673898323   14.54802505401808
65536   -8.18310798331338  -14.35313674037229   14.54802505568436


#   n           |Yn-Y2n|               log(h)        log(|Yn-Y2n|)
  128  0.311157076838266 -4.8520302639196e+00 -1.1674574240884e+00
  256  0.039688006572034 -5.5451774444796e+00 -3.2267062383909e+00
  512  0.004987781907165 -6.2383246250395e+00 -5.3007639756236e+00
 1024  0.000624343615485 -6.9314718055995e+00 -7.3788096753143e+00
 2048  0.000078070915586 -7.6246189861594e+00 -9.4578929701382e+00
 4096  0.000009759764890 -8.3177661667193e+00 -1.1537242246973e+01
 8192  0.000001219999360 -9.0109133472793e+00 -1.3616660223556e+01
16384  0.000000152502076 -9.7040605278392e+00 -1.5696087624999e+01
32768  0.000000019063376 -1.0397207708399e+01 -1.7775496850149e+01
65536  0.000000002383025 -1.1090354888959e+01 -1.9854895059226e+01


#RK4 method computing y(1)
#   n                  y1                  y2                  y3
   64   -8.18157152498330  -14.35089478427027   14.54563019863711
  128   -8.18305851834925  -14.35306205062053   14.54795406227568
  256   -8.18310636897497  -14.35313414968720   14.54802311118907
  512   -8.18310792882860  -14.35313664375124   14.54802501240169
 1024   -8.18310798148616  -14.35313673655432   14.54802505563920
 2048   -8.18310798337582  -14.35313674038241   14.54802505597974
 4096   -8.18310798345050  -14.35313674056053   14.54802505592790
 8192   -8.18310798345370  -14.35313674056963   14.54802505592218
16384   -8.18310798345441  -14.35313674057091   14.54802505592262
32768   -8.18310798345349  -14.35313674056959   14.54802505592123
65536   -8.18310798345413  -14.35313674057054   14.54802505592208


#   n           |Yn-Y2n|               log(h)        log(|Yn-Y2n|)
  128  0.003508352165046 -4.8520302639196e+00 -5.6526088203872e+00
  256  0.000110705511325 -5.5451774444796e+00 -9.1086369333556e+00
  512  0.000003502585931 -6.2383246250395e+00 -1.2562009024874e+01
 1024  0.000000115129116 -6.9314718055995e+00 -1.5977211590526e+01
 2048  0.000000004282642 -7.6246189861594e+00 -1.9268695655496e+01
 4096  0.000000000199979 -8.3177661667193e+00 -2.2332807151546e+01
 8192  0.000000000011211 -9.0109133472793e+00 -2.5214167095872e+01
16384  0.000000000001527 -9.7040605278392e+00 -2.7207716371580e+01
32768  0.000000000002124 -1.0397207708399e+01 -2.6877519705898e+01
65536  0.000000000001423 -1.1090354888959e+01 -2.7278468588599e+01
```

Plotting the convergence using the gnuplot script

```
set terminal postscript enhanced color eps font "Courier-Bold"
set style data lines
set size 0.7,0.7
```

```
set key spacing 1.2
set key bottom
set output "2dRK2.eps"
plot "prog2d.out" index 1 using 3:4 ti "RK2",\
     "prog2d.out" index 3 using 3:4 ti "RK3",\
     "prog2d.out" index 5 using 3:4 ti "RK4"
```

obtains the combined graph for the RK2, RK3 and RK4 methods



RK2 appears linear for $\log h \in [-11, -8]$ with slope

$$\frac{-8.6085259970367 + 14.307845383653}{-8.3177661667193 + 11.090354888959} \approx 2.0555949538784763$$

This verfies the RK2 method is 2nd order.

The graph for the RK3 method appears linear for $\log h \in [-11, -5]$ with slope

$$\frac{-1.1674574240884 + 19.854895059226}{-4.8520302639196 + 11.090354888959} \approx 2.9955859559039175$$

This verified the RK3 method is 3rd order.

Finally, the graph for the RK4 method appears linear for $\log h \in [-8, -5]$ with slope

$$\frac{-5.6526088203872 + 22.332807151546}{-4.8520302639196 + 8.3177661667193} \approx 4.812887882681469$$

This verifies the RK4 method is 4th order. It's interesting numerically that the convergence seems even better than 4th order.

5. Approximate $y(10)$ to three decimal places. Is it possible to achieve this accuracy using Euler's method? Can you find $y(100)$?

The C program

```
#  prog2e.jl -- integration of a chaotic non-linear system

using Printf, LinearAlgebra

function f(y)
    r=[10*(y[2]-y[1]),
       (28.0-y[3])*y[1]-y[2],
       y[1]*y[2]-(8/3)*y[3]]
    return r
end

function euler(t,y,h)
    return y+h*f(y)
end

function rk2(t,y,h)
    k1=f(y)
    k2=f(y+h*2/3*k1)
    return y+h*(1/4*k1+3/4*k2)
end

function rk3(t,y,h)
    k1=f(y)
    k2=f(y+h*2/3*k1)
    k3=f(y+h*2/3*k2)
    return y+h*(1/4*k1+3/8*k2+3/8*k3)
end

function rk4(t,y,h)
    k1=f(y)
    k2=f(y+h*1/2*k1)
    k3=f(y+h*1/2*k2)
    k4=f(y+h*k3)
    return y+h*(1/6*k1+1/3*k2+1/3*k3+1/6*k4)
end

function solve(step,t0,y0,h,n)
    yn=copy(y0)
    for j=1:n
        tn=t0+(j-1)*h
        yn=step(tn,yn,h)
    end
    return yn
```

```
44 end
45
46 function main()
47     nres=15
48     t0=0.0
49     y0=[2.0,3.0,15.0]
50     for l=1:2
51         T=[10.0,100.0][l]
52         for k=1:2
53             step=[euler,rk4][k]
54             name=["euler","RK4"][k]
55             @printf("#%s method computing y(%g)\n",name,T)
56             @printf("#%7s %19s %19s %19s\n","n","y1","y2","y3")
57             N=Vector{Int}(undef,nres)
58             Y=Vector{Vector{Float64}}(undef,nres)
59             n=2048
60             for i=1:nres
61                 h=T/n
62                 yn=solve(step,t0,y0,h,n)
63                 N[i]=n
64                 Y[i]=copy(yn)
65                 @printf("%8d %19.14f %19.14f %19.14f\n",
66                     n,yn[1],yn[2],yn[3])
67                 n*=2
68             end
69             @printf("\n\n#%7s %18s %20s %20s\n",
70                 "n","|Yn-Y2n|","log(h)","log(|Yn-Y2n|)")
71             for i=2:nres
72                 e=norm(Y[i-1]-Y[i])
73                 @printf("%8d %18.15f %20.13e %20.13e\n",
74                     N[i],e,-log(N[i]),log(e))
75             end
76             if k<2
77                 @printf("\n\n")
78             end
79         end
80         if l<2
81             @printf("\n\n")
82         end
83     end
84 end
85
86 main()
```

produces the output

```
#euler method computing y(10)
#      n                  y1                   y2                   y3
    2048    9.65709179129176   14.15604838934646   21.67561874292528
    4096   -8.23404131444176   -1.05802277934966   34.10035321945219
    8192  -10.15919988786769  -14.66865360648860   22.56706935369528
   16384    3.30725155167164    4.92900261338278   16.24042782779068
   32768    1.42015101522318    2.48237653041534   16.42890125229822
   65536   -1.81201409058745   -3.07480687466168   14.62440449291724
  131072   -7.12712223149322  -10.06858978211086   20.23897803766334
  262144    2.08076031552524    3.38290104149757   13.65187426267135
  524288    1.76465022239860    3.22410591970935    9.94807375564609
 1048576    3.23601920345330    5.66688684134083   12.35270132637459
 2097152    1.80211306384783    2.90985611355445   13.64366575202682
 4194304   -4.31606734773873   -5.81063204132745   18.75531247859778
 8388608   -1.79633598811470   -2.73107487346315   14.79585372623922
16777216   -0.08630710517209    0.04004122155257   14.73181135489150
33554432    0.70376569733087    1.32488794766655   15.18468308398613


#      n           |Yn-Y2n|               log(h)        log(|Yn-Y2n|)
    4096 26.569430406868353 -8.3177661667193e+00  3.2797613220887e+00
    8192 17.943582244736664 -9.0109133472793e+00  2.8872325158912e+00
   16384 24.605687080084447 -9.7040605278392e+00  3.2029775983371e+00
   32768  3.095585543361101 -1.0397207708399e+01  1.1299770783429e+00
   65536  6.677229007866944 -1.1090354888959e+01  1.8987030823057e+00
  131072 10.425296638245548 -1.1783502069519e+01  2.3442352217880e+00
  262144 17.581741737376952 -1.2476649250079e+01  2.8668609622856e+00
  524288  3.720655812826665 -1.3169796430639e+01  1.3138999465177e+00
 1048576  3.730192898953663 -1.3862943611199e+01  1.3164599478557e+00
 2097152  3.365099463530932 -1.4556090791759e+01  1.2134575209157e+00
 4194304 11.815581977432197 -1.5249237972319e+01  2.4694191669317e+00
 8388608  5.613379738307923 -1.5942385152879e+01  1.7251529870314e+00
16777216  3.256898005446434 -1.6635532333439e+01  1.1807752103129e+00
33554432  1.574845689543540 -1.7328679513999e+01  4.5415729258475e-01


#RK4 method computing y(10)
#      n                  y1                   y2                   y3
    2048    1.39236316377876    2.42328584240174   15.96593830344759
    4096    1.39156160999622    2.42202285056433   15.96467355262893
    8192    1.39162081955006    2.42211591309572   15.96476126181184
   16384    1.39162793075231    2.42212709628902   15.96477194635119
   32768    1.39162848118190    2.42212796200363   15.96477277580231
   65536    1.39162851815823    2.42212802016204   15.96477283157386
  131072    1.39162852161446    2.42212802559885   15.96477283680252
  262144    1.39162852279966    2.42212802746356   15.96477283860326
  524288    1.39162852136086    2.42212802519969   15.96477283641306
 1048576    1.39162851887693    2.42212802129145   15.96477283263460
 2097152    1.39162852152208    2.42212802545342   15.96477283666087
 4194304    1.39162852361560    2.42212802874738   15.96477283984458
 8388608    1.39162852022954    2.42212802341976   15.96477283469658
16777216    1.39162851938607    2.42212802209265   15.96477283341023
33554432    1.39162850287708    2.42212799611690   15.96477280829021
```

```
#      n           |Yn-Y2n|                log(h)        log(|Yn-Y2n|)
    4096  0.001958885264879 -8.3177661667193e+00 -6.2353797099436e+00
    8192  0.000140923052695 -9.0109133472793e+00 -8.8672965421183e+00
   16384  0.000017023289656 -9.7040605278392e+00 -1.0980928171714e+01
   32768  0.000001319251187 -1.0397207708399e+01 -1.3538446264700e+01
   65536  0.000000088657294 -1.1090354888959e+01 -1.6238487525840e+01
  131072  0.000000008297185 -1.1783502069519e+01 -1.8607349518887e+01
  262144  0.000000002850353 -1.2476649250079e+01 -1.9675822870615e+01
  524288  0.000000003462981 -1.3169796430639e+01 -1.9481136031170e+01
 1048576  0.000000005976706 -1.3862943611199e+01 -1.8935396214852e+01
 2097152  0.000000006366295 -1.4556090791759e+01 -1.8872248093807e+01
 4194304  0.000000005036764 -1.5249237972319e+01 -1.9106502111555e+01
 8388608  0.000000008145608 -1.5942385152879e+01 -1.8625786958877e+01
16777216  0.000000002031587 -1.6635532333439e+01 -2.0014448489058e+01
33554432  0.000000039727843 -1.7328679513999e+01 -1.7041213551433e+01


#euler method computing y(100)
#      n                  y1                  y2                  y3
    2048                 NaN                 NaN                 NaN
    4096   -3.69414817222810   -5.51868887324954   18.88188787142159
    8192    9.71656536549145    6.51108705424435   32.06013788180076
   16384    7.21303379020715    9.45567447671402   21.78060376748864
   32768    9.89068989868577   16.47982871037919   17.87711855890827
   65536    0.73641333852435    0.90826119191029   15.57308586397270
  131072  -13.63958416868028  -10.67913456516161   36.67054053804911
  262144  -13.44309248713940  -15.46980571993365   31.34367443023991
  524288   -0.30398914651074    2.40530773080069   24.05590766224847
 1048576    3.74447333107404    5.76833231503641   19.95034722219738
 2097152   -2.55398673146322   -4.11576183112279   14.11702221040225
 4194304    1.36768359076515   -4.37407227305951   28.50559337415262
 8388608    3.34884351507065    5.54679103302497   22.09036256287579
16777216    2.04875011866057   -2.12241726151513   26.88930667585435
33554432   -1.27600402209199   -2.27224400890223   11.70250464404255


#      n           |Yn-Y2n|                log(h)        log(|Yn-Y2n|)
    4096                 NaN -8.3177661667193e+00                 NaN
    8192 22.321044325760049 -9.0109133472793e+00  3.1055299252071e+00
   16384 10.982125798055625 -9.7040605278392e+00  2.3962690237224e+00
   32768  8.470288171437556 -1.0397207708399e+01  2.1365645306870e+00
   65536 18.209422313228082 -1.1090354888959e+01  2.9019391696354e+00
  131072 28.036398457770400 -1.1783502069519e+01  3.3335036108987e+00
  262144  7.166912977328935 -1.2476649250079e+01  1.9694750148045e+00
  524288 23.350958480049233 -1.3169796430639e+01  3.1506380316686e+00
 1048576  6.674998824981036 -1.3862943611199e+01  1.8983690282535e+00
 2097152 13.091737735791607 -1.4556090791759e+01  2.5719813240362e+00
 4194304 14.915669697836879 -1.5249237972319e+01  2.7024123179199e+00
 8388608 11.979303391501663 -1.5942385152879e+01  2.4831804433758e+00
16777216  9.139850288922862 -1.6635532333439e+01  2.2126440055642e+00
33554432 15.547198915049837 -1.7328679513999e+01  2.7438804883188e+00


#RK4 method computing y(100)
#      n                  y1                  y2                  y3
```

```
    2048    4.14364996849859    5.96632257127863   17.47361711136143
    4096   -5.63885044476829   -8.96637138369860   16.22347396293149
    8192   -9.85406991493646   -0.42383466213940   37.36098153630476
   16384    4.94436931779542    7.93965251520565   16.22940902882609
   32768   10.47689157548414   16.06653393203018   21.22456423996539
   65536    4.62193558375131   -3.26238872253328   31.95988130891941
  131072   -2.93671716965939    1.01048884755307   27.04164409132606
  262144   -0.54573315021434   -1.00369072147099   10.36096192532752
  524288    0.02109468939155    3.65673526853064   25.50904664734869
 1048576   -0.33753089034628   -1.35797353275571   19.31605101138085
 2097152   -7.07575870463571    0.31791127063193   33.28069199414505
 4194304  -13.22753202660132   -7.77373246861026   38.21211160709970
 8388608   14.43614101221650    5.34231964561189   42.41156939270027
16777216    8.22760066233164   12.56232109555705   19.27865668389723
33554432    7.69391444311727   11.75662973835408   19.08945890685662
```

```
#       n           |Yn-Y2n|               log(h)        log(|Yn-Y2n|)
    4096 17.895405024199544 -8.3177661667193e+00  2.8845439773343e+00
    8192 23.184849260568519 -9.0109133472793e+00  3.1434990162294e+00
   16384 27.119828136057610 -9.7040605278392e+00  3.3002651258861e+00
   32768 11.027537335158781 -1.0397207708399e+01  2.4003955386325e+00
   65536 22.872140110234543 -1.1090354888959e+01  3.1299195808422e+00
  131072  9.978966454704134 -1.1783502069519e+01  2.3004795233076e+00
  262144 16.971119039137552 -1.2476649250079e+01  2.8315130192723e+00
  524288 15.858919728454586 -1.3169796430639e+01  2.7637321009269e+00
 1048576  7.976785794745757 -1.3862943611199e+01  2.0765355477097e+00
 2097152 15.595624441778110 -1.4556090791759e+01  2.7469903904110e+00
 4194304 11.297695021848440 -1.5249237972319e+01  2.4245987245441e+00
 8388608 30.902185598346414 -1.5942385152879e+01  3.4308269127404e+00
16777216 25.016115697814879 -1.6635532333439e+01  3.2195202450975e+00
33554432  0.984761566313390 -1.7328679513999e+01 -1.5355731768883e-02
```

From the RK4 method we conclude to three decimal places that

$$y(10) \approx \begin{bmatrix} 1.392 \\ 2.422 \\ 15.965 \end{bmatrix}.$$

The output for the Euler method shows there is no convergence even for n as large as 33554432. Larger values of n are computationally expensive and it is likely roundoff error would accumulate to the point the computation wouldn't converge anyway.

I can not find $y(100)$. Neither the RK4 nor the Euler method converge. For any chance of approximating the value of $y(100)$ it would appear that floating point arithmetic with more significant digits and a higher-order integrator must be used.