Extra Credit Problems from Chapter 2.6.

2. Use Euler's method to obtain a four-decimal approximation of the value $y(0.2)$ where

$$y' = x + y^2, \qquad y(0) = 0.$$

Carry out the recursion $y_{n+1} = y_n + hf(x_n, y_n)$ by hand, first using $h = 0.1$ and then using $h = 0.05$.

For $h = 0.1$ we calculate $y_2 \approx y(1.2)$ as

$$\begin{aligned}
y_0 &= 0 \\
y_1 &= y_0 + hf(x_0, y_0) = 0 + (0.1)f(0,0) = 0 + (0.1)(0 + 0^2) = 0 \\
y_2 &= y_1 + hf(x_1, y_1) = 0 + (0.1)(0.1 + 0^2) = 0.01.
\end{aligned}$$

For $h = 0.5$ we calculate $y_4 \approx y(1.2)$ as

$$\begin{aligned}
y_0 &= 0 \\
y_1 &= y_0 + hf(x_0, y_0) = 0 + (0.05)f(0,0) = 5 + (0.05)(0 + 0^2) = 0 \\
y_2 &= y_1 + hf(x_1, y_1) = 0 + (0.05)(0.05 + 0^2) = 0.0025 \\
y_3 &= y_2 + hf(x_2, y_2) = 0.0025 + (0.05)(0.1 + 0.0025^2) \approx 0.0075 \\
y_4 &= y_3 + hf(x_3, y_3) = 0.0075 + (0.05)(0.15 + 0.0075^2) \approx 0.0150.
\end{aligned}$$

4. Use Euler's method to obtain a four-decimal approximation of the value $y(1.5)$ where

$$y' = 2xy, \qquad y(1) = 1.$$

Find an explicitly solution for each initial-value problem and then construct tables showing x_n, y_n, actual value, absolute error and the percent relative error.

To explicitly solve the ordinary differential equation, note that it is linear and multiply by the integrating factor

$$\mu = \exp\left(\int -2x dx\right) = e^{-x^2}$$

to obtain

$$\left(ye^{-x^2}\right)' = 0.$$

Consequently

$$y(x)e^{-x^2} = c \qquad \text{or} \qquad y(x) = ce^{x^2}.$$

After solving for c to satisfy $y(1) = 1$, the exact solution may be written as

$$y(x) = e^{-1}e^{x^2} = \exp(x^2 - 1).$$

The C program

```
#include <stdio.h>
#include <math.h>

double f(double x,double y){
    return 2*x*y;
}
double exact(double x){
    return exp(x*x-1);
}
int main(){
    int i;
    double x0=1,xn=1.5,y0=1;
    for(int n=5;n<=10;n+=5){
        double h=(xn-x0)/n;
        double y=y0;
        printf("Euler's Method Table for h=%g\n",h);
        printf("%12s %12s %12s %12s %12s\n","xn","yn","actual",
            "abs-error","%-rel-error");
        for(int i=0;;i++){
            double x=x0+i*h;
            double v=exact(x);
            printf("%12.5g %12.5g %12.5g %12.5g %12.5g\n",
                x,y,v,v-y,100*(v-y)/v);
            if(i>=n) break;
```

```
25              y=y+h*f(x,y);
26          }
27          if(n<10) printf("\n");
28      }
29      return 0;
30  }
```

produces the tables

```
Euler's Method Table for h=0.1
          xn           yn       actual    abs-error  %-rel-error
           1            1            1            0            0
         1.1          1.2       1.2337     0.033678       2.7299
         1.2        1.464       1.5527     0.088707       5.7131
         1.3       1.8154       1.9937      0.17836       8.9459
         1.4       2.2874       2.6117      0.32434       12.419
         1.5       2.9278       3.4903      0.56253       16.117

Euler's Method Table for h=0.05
          xn           yn       actual    abs-error  %-rel-error
           1            1            1            0            0
        1.05          1.1       1.1079    0.0079373       0.7164
         1.1       1.2155       1.2337     0.018178       1.4735
        1.15       1.3492       1.3806      0.03137       2.2722
         1.2       1.5044       1.5527     0.048344       3.1135
        1.25       1.6849       1.7551     0.070167        3.998
         1.3       1.8955       1.9937     0.098217       4.9264
        1.35       2.1419       2.2762      0.13427       5.8989
         1.4       2.4311       2.6117      0.18063        6.916
        1.45       2.7714       3.0117      0.24026       7.9778
         1.5       3.1733       3.4903      0.31707       9.0841
```

8. Use a numerical solver and Euler's method to obtain a four-decimal approximation of the value $y(0.5)$ where

$$y' = xy + \sqrt{y}, \qquad y(0) = 1.$$

The C program

```c
#include <stdio.h>
#include <math.h>

double f(double x,double y){
    return x*y+sqrt(y);
}
int main(){
    int i;
    double x0=0,xn=0.5,y0=1;
    printf("%12s %12s %12s\n","h","xn","yn");
    for(int n=5;n<=10;n+=5){
        double h=(xn-x0)/n;
        double y=y0;
        for(int i=0;i<n;i++){
            double x=x0+i*h;
            y=y+h*f(x,y);
        }
        printf("%12.5g %12.5g %12.5g\n",h,xn,y);
    }
    return 0;
}
```

produces the output

```
           h           xn           yn
         0.1          0.5       1.6902
        0.05          0.5       1.7219
```

which means that $y(0.5) \approx 1.6902$ when $h = 0.1$ and $y(0.5) \approx 1.7219$ when $h = 0.05$.

12. Use a numerical solver to obtain solution curves for the initial-value problem

$$y' = y(10 - 2y), \qquad y(0) = 1.$$

First use Euler's method and then the RK4 method. Use $h = 0.25$ in each case. Superimpose both solution curves on the same coordinate axes. Repeat, for $h = 0.1$ and then for $h = 0.05$.

The C program

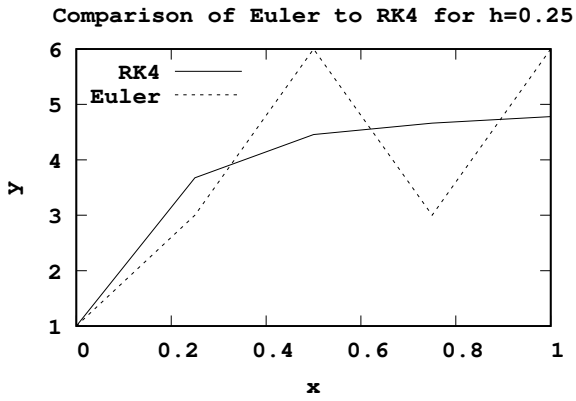
```
#include <stdio.h>
#include <math.h>

double f(double x,double y){
    return y*(10-2*y);
}
double euler(double h,double x,double y){
    return y+h*f(x,y);
}
double rk4(double h,double x,double y){
    double k1=h*f(x,y);
    double k2=h*f(x+h/2,y+k1/2);
    double k3=h*f(x+h/2,y+k2/2);
    double k4=h*f(x+h,y+k3);
    return y+(k1+2*(k2+k3)+k4)/6;
}
int main(){
    int i;
    double x0=0,xn=1,y0=1;
    for(int j=0;j<3;j++){
        int n=(int []){4,10,20}[j];
        double h=(xn-x0)/n;
        double yeuler=y0,yrk4=y0;
        printf("# h=%g\n#x yeuler yrk4\n",h);
        for(int i=0;;i++){
            double x=x0+i*h;
            printf("%g %g %g\n",x,yeuler,yrk4);
            if(i>=n) break;
            yeuler=euler(h,x,yeuler);
            yrk4=rk4(h,x,yrk4);
        }
        printf("\n\n");
    }
    return 0;
}
```

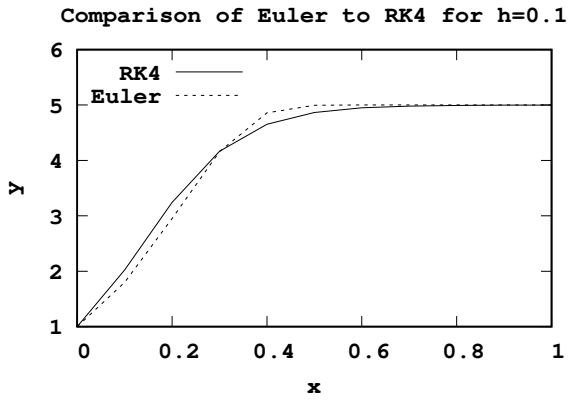
approximates the solution using both Euler's method and the RK4 method for the desired choices of h. The output is in the form of a three tables of numbers which we then plot. The plotting script for the case $h = 0.25$ looks like

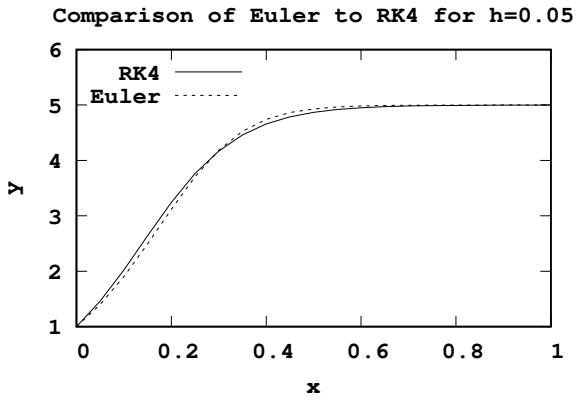
```
1  set terminal postscript enhanced eps font "Courier-Bold"
2  set output 'q12p0.eps'
3  set size 0.5,0.5
4  set key left
5  set ylabel "y"
6  set xlabel "x"
7  set title "Comparison of Euler to RK4 for h=0.25"
8  set style data lines
9  plot [:] [1:6] \
10     "q12.out" index 0 using 1:3 title "RK4", \
11     "q12.out" index 0 using 1:2 title "Euler"
```

and the resulting plot is



The plots for $h = 0.1$ and $h = 0.05$ were made with similar scripts with changes to the title on line 7 and to the index on lines 10 and 11.





Note that the difference between the curves for Euler's method and the RK4 method visibly decrease for smaller values of h.

13. Use a numerical solver and Euler's method to approximate $y(1.0)$, where $y(x)$ is the solution to

$$y' = 2xy^2, \qquad y(0) = 1.$$

First use $h = 0.1$ and then use $h = 0.05$. Repeat, using the RK4 method. Discuss what might cause the approximations to $y(1.0)$ to differ so greatly.

The C program

```
#include <stdio.h>
#include <math.h>

double f(double x,double y){
    return 2*x*y*y;
}
double euler(double h,double x,double y){
    return y+h*f(x,y);
}
double rk4(double h,double x,double y){
    double k1=h*f(x,y);
    double k2=h*f(x+h/2,y+k1/2);
    double k3=h*f(x+h/2,y+k2/2);
    double k4=h*f(x+h,y+k3);
    return y+(k1+2*(k2+k3)+k4)/6;
}
int main(){
    int i;
    double x0=0,xn=1,y0=1;
    printf("%12s %12s %12s %12s\n","h","x","yeuler","yrk4");
    for(int j=0;j<2;j++){
        int n=(int []){10,20}[j];
        double h=(xn-x0)/n;
        double yeuler=y0,yrk4=y0;
        for(int i=0;i<n;i++){
            double x=x0+i*h;
            yeuler=euler(h,x,yeuler);
            yrk4=rk4(h,x,yrk4);
        }
        printf("%12.5g %12.5g %12.4f %12.4f\n",h,xn,yeuler,yrk4);
    }
    return 0;
}
```

produces the output

h	x	yeuler	yrk4
0.1	1	3.8191	42.9931
0.05	1	5.9363	84.0132

We note that the different approximations range from 3.8191 to 84.0132. The reason these approximations differ so greatly can be seen by examining the exact solution.

The ordinary differential equation is separable. Integrating yields

$$\int \frac{1}{y^2} dy = \int 2x dx, \qquad \frac{-1}{y} = x^2 + c \qquad \text{or} \qquad y = \frac{1}{c - x^2}.$$

Since $y(0) = 1$ solving for the constant yields that $c = 1$. Therefore the exact solution is

$$y(x) = \frac{1}{1 - x^2}.$$

Since the exact solution $y(x) \to \infty$ as $x \nearrow 1$, then the value of $y(1.0)$ does not actually exist. Consequently, it is not surprising that numerical approximations for something that does not exist differ so greatly.