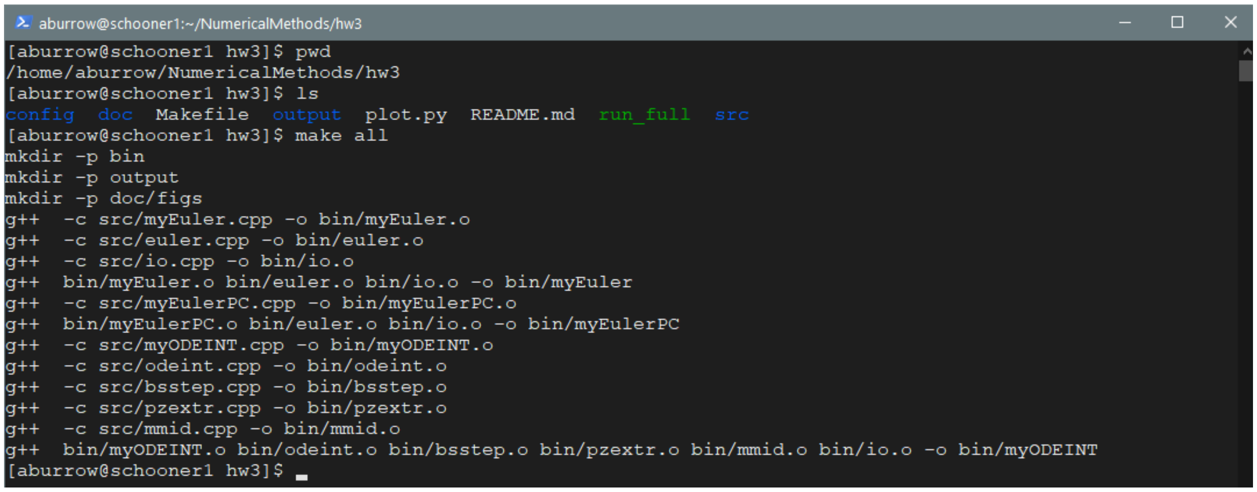
Assignment 3: Results/Description

Problem 1

For this problem I have written three programs (`myEuler`, `myEulerPC`, and `myODEINT`) to solve the following problems. These may be compiled all together with a `make all` as shown below.



```
[aburrow@schooner1 hw3]$ pwd
/home/aburrow/NumericalMethods/hw3
[aburrow@schooner1 hw3]$ ls
config  doc  Makefile  output  plot.py  README.md  run_full  src
[aburrow@schooner1 hw3]$ make all
mkdir -p bin
mkdir -p output
mkdir -p doc/figs
g++  -c src/myEuler.cpp -o bin/myEuler.o
g++  -c src/euler.cpp -o bin/euler.o
g++  -c src/io.cpp -o bin/io.o
g++  bin/myEuler.o bin/euler.o bin/io.o -o bin/myEuler
g++  -c src/myEulerPC.cpp -o bin/myEulerPC.o
g++  bin/myEulerPC.o bin/euler.o bin/io.o -o bin/myEulerPC
g++  -c src/myODEINT.cpp -o bin/myODEINT.o
g++  -c src/odeint.cpp -o bin/odeint.o
g++  -c src/bsstep.cpp -o bin/bsstep.o
g++  -c src/pzextr.cpp -o bin/pzextr.o
g++  -c src/mmid.cpp -o bin/mmid.o
g++  bin/myODEINT.o bin/odeint.o bin/bsstep.o bin/pzextr.o bin/mmid.o bin/io.o -o bin/myODEINT
[aburrow@schooner1 hw3]$
```

This generates executables `myEuler`, `myEulerPC`, and `myODEINT` in the "./bin" directory, which may be run one at a time. There is also a `plot.py` in the root which may be run to generate plots from the output of the executables. However, to run all these programs at once, I have included a `run_full` bash script that may be executed for more ease.

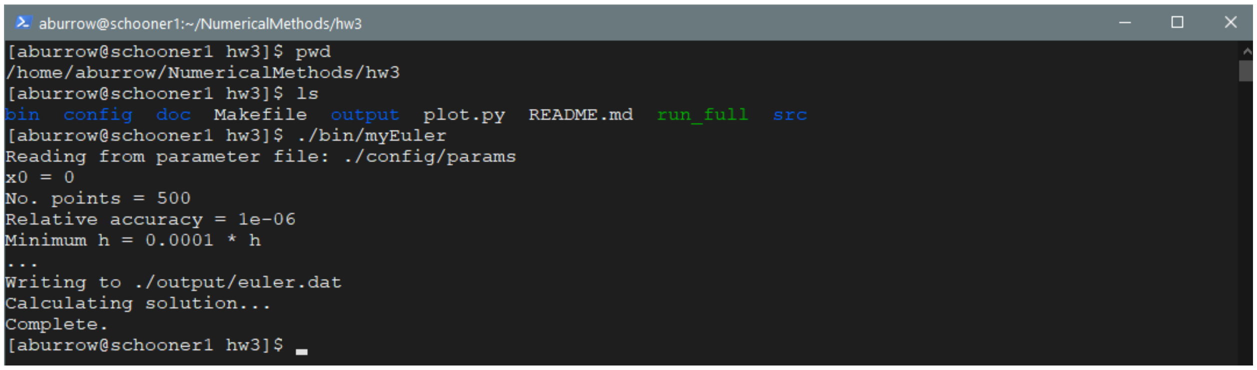
For all of these problems, I am solving the equation

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

for $y(x)$ where $f(x, y) = -\cos x$. We know already that the analytic solution is $y(x) = -\sin x$, and so I use the boundary condition $y(0) = 0$, and I solve the equations on the interval $[0, 20\pi]$ (which is 10 cycles of the periodic function).

(a)

Below I run the program that demonstrates solving our differential equation using the Euler method:



```
[aburrow@schooner1 hw3]$ pwd
/home/aburrow/NumericalMethods/hw3
[aburrow@schooner1 hw3]$ ls
bin  config  doc  Makefile  output  plot.py  README.md  run_full  src
[aburrow@schooner1 hw3]$ ./bin/myEuler
Reading from parameter file: ./config/params
x0 = 0
No. points = 500
Relative accuracy = 1e-06
Minimum h = 0.0001 * h
...
Writing to ./output/euler.dat
Calculating solution...
Complete.
[aburrow@schooner1 hw3]$
```

I solve for 500 points of the solution, and those are plotted in Figure 1, along with the residual from the analytic solution. We see that the general trend of the analytic solution is represented by the calculated solution. Looking at the residuals, they are on the order of 10^{-1}, and they are actually asymmetric, so that there is only negative residual for this solution. Because of this, we see that the solution seems shifted down and to the right, so that the amplitudes are well described by the calculated solution.

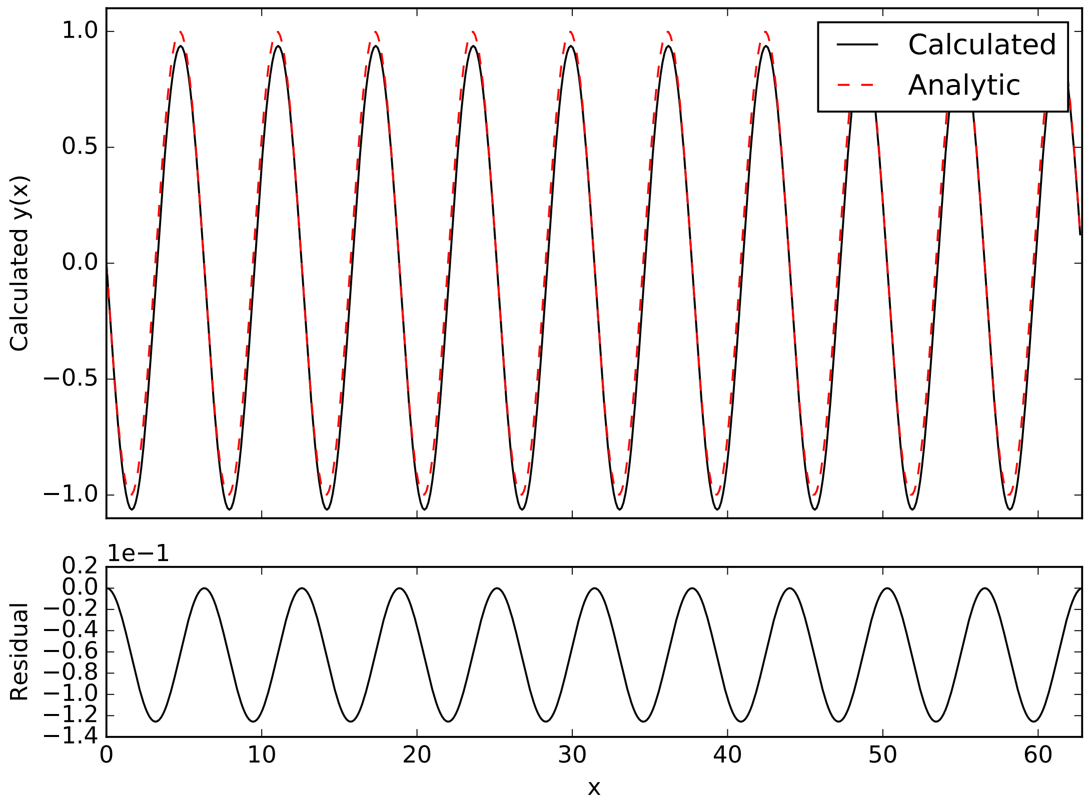
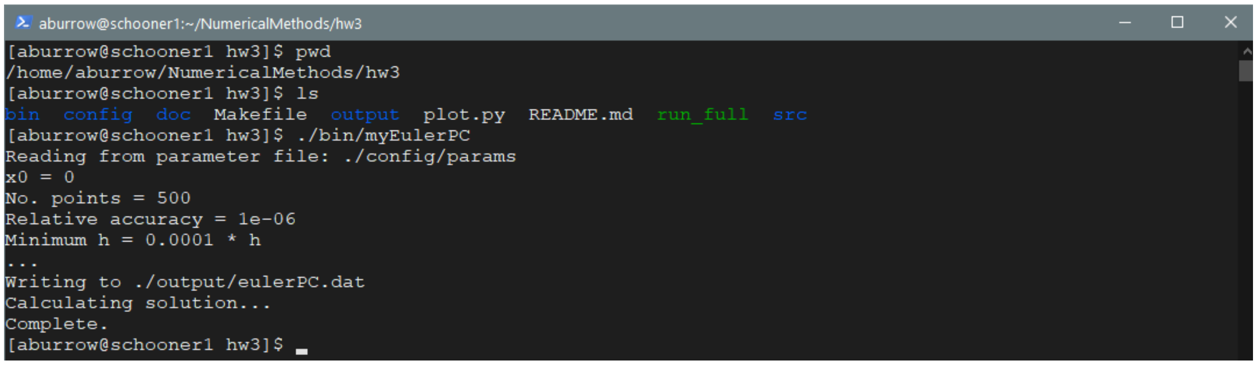


Figure 1: Solution using the Euler method.

(b)

Below I run the program that demonstrates solving our differential equation using the Euler predictor-corrector method:



```
[aburrow@schooner1 hw3]$ pwd
/home/aburrow/NumericalMethods/hw3
[aburrow@schooner1 hw3]$ ls
bin  config  doc  Makefile  output  plot.py  README.md  run_full  src
[aburrow@schooner1 hw3]$ ./bin/myEulerPC
Reading from parameter file: ./config/params
x0 = 0
No. points = 500
Relative accuracy = 1e-06
Minimum h = 0.0001 * h
...
Writing to ./output/eulerPC.dat
Calculating solution...
Complete.
[aburrow@schooner1 hw3]$
```

Again the solution is found at 500 points, which is shown in Figure 2. It is clearly a much better fit from the top panel, however the residuals explain this by showing error on order of 10^{-3}. The residuals are also symmetric and for this case they are greatest at the extrema of the solution.

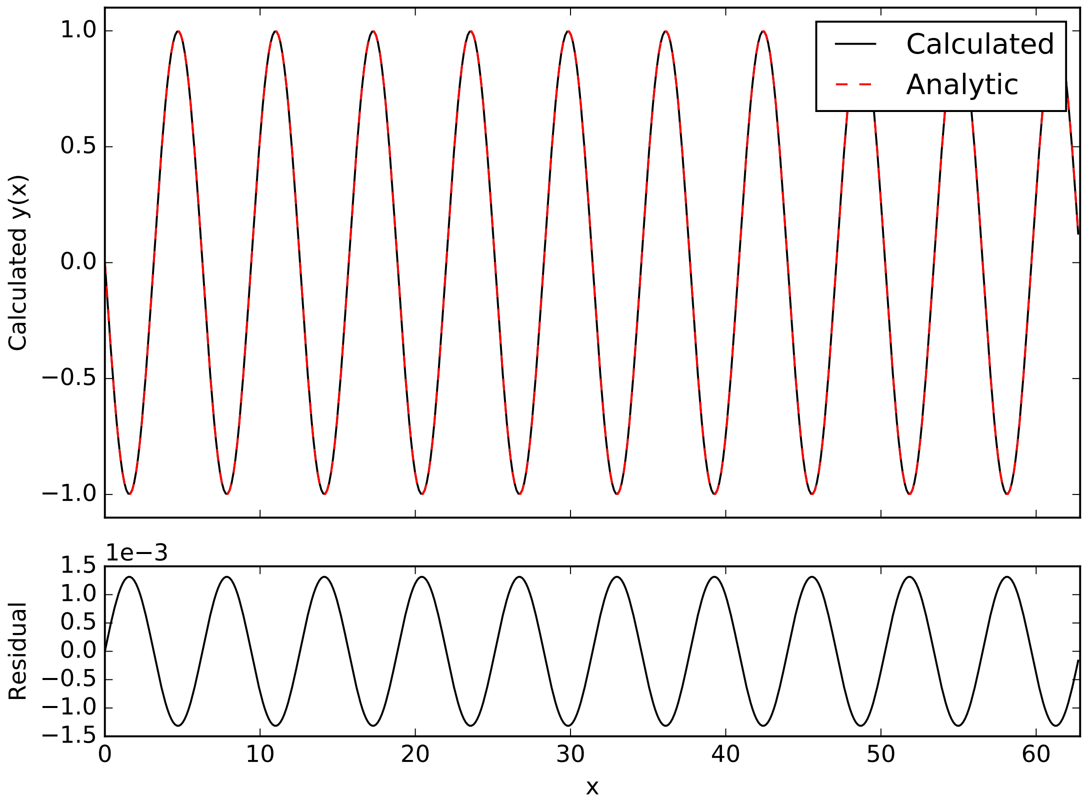
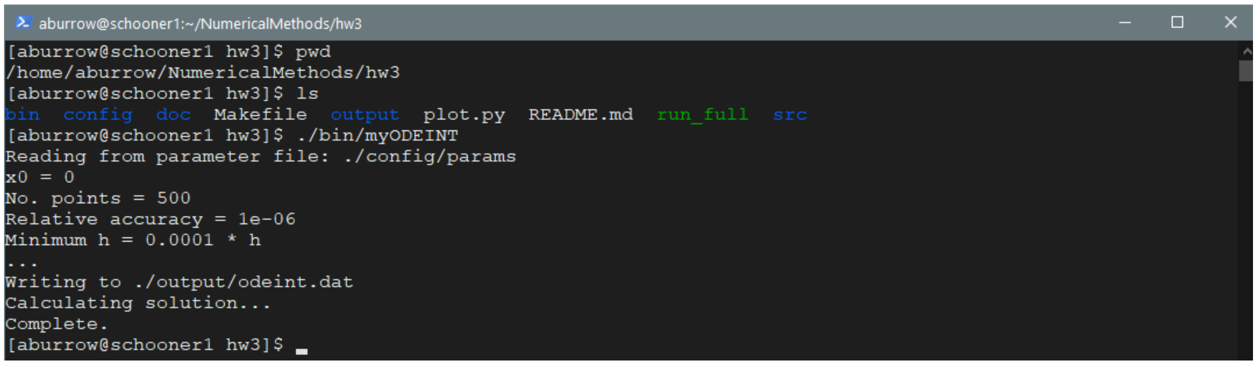


Figure 2: Solution using the Euler predictor-corrector method.

(c)

Finally I run the program that demonstrates solving our differential equation using the Numerical Recipes' ODEINT function with the Bulirsch-Shoer algorithm below:


aburrow@schooner1:~/NumericalMethods/hw3

```
[aburrow@schooner1 hw3]$ pwd
/home/aburrow/NumericalMethods/hw3
[aburrow@schooner1 hw3]$ ls
bin  config  doc  Makefile  output  plot.py  README.md  run_full  src
[aburrow@schooner1 hw3]$ ./bin/myODEINT
Reading from parameter file: ./config/params
x0 = 0
No. points = 500
Relative accuracy = 1e-06
Minimum h = 0.0001 * h
...
Writing to ./output/odeint.dat
Calculating solution...
Complete.
[aburrow@schooner1 hw3]$
```

To achieve the solution, I use a relative accuracy of 10^{-6} and allowed a minimum h value of 10^{-4} times the initial h. The function was applied to each of the 499 intervals (to get 500 points of the solution), and the initial h was 1/10 of the interval size. The solution is given in Figure 3. Again the top panel shows that the solution fits well with the analytic solution. The residuals, however, are extremely small ($\sim 10^{-11}$!) compared to the previous two methods. The residuals are asymmetric, and the greatest deviation is seen in the troughs of the solution.

(d)

From the results in parts (a)-(c), we can immediately conclude that the most representative solution comes from the ODEINT method, as it is orders of magintude more accurate through the entire solution interval. This is of course because there is more that goes into this method, such as a level of tolerance that is used for comparison as well as the extra functionality.

In terms of simplicity, the easiest to code are definitely the first two methods. Therefore if one wanted to do this from scratch with little hassle, the best method may be to use the Euler predictor-corrector method, as it still has the possibility of achieving a solution within an order of 10^{-3} (residuals).

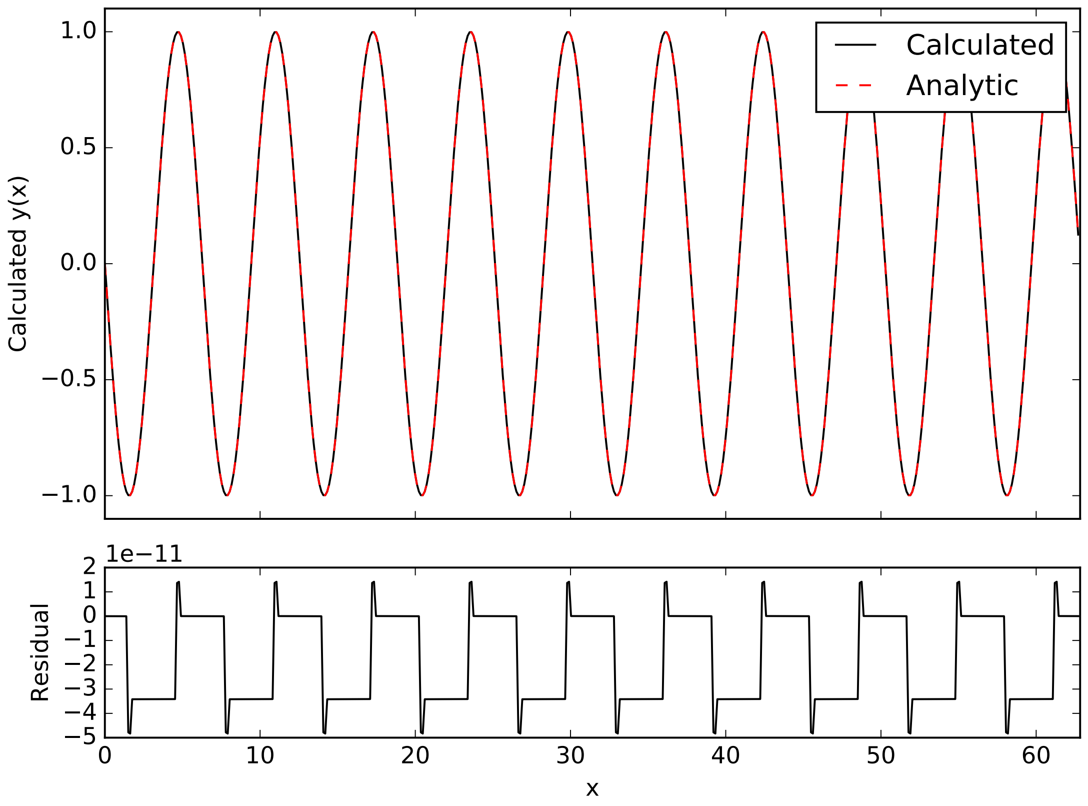


Figure 3: Solution using the Numerical Recipes ODEINT (BS) method.