Assignment 3

Brett Bonine
ASTR 5900: Numerical Methods

March 4, 2021

Problem 1

Numericall solve the ordinary differential equation

$$\frac{dy}{dx} = -cos(x)$$

a) Using the Euler Method

Answer: The Euler Method is the most straightforward of the three we'll look at. Assuming a stepsize h, we can estimate the value for y_{n+1} according to

$$y_{n+1} = y_n + hy'_n \tag{1}$$

We see that this method is analogous to the Newton Method we used for finding roots via the slope of the tangent line at a point. We choose h such that there are 100 equally spaced steps on the interval $[0, 20\pi]$.The results of this procedure are shown below in Figure 1.

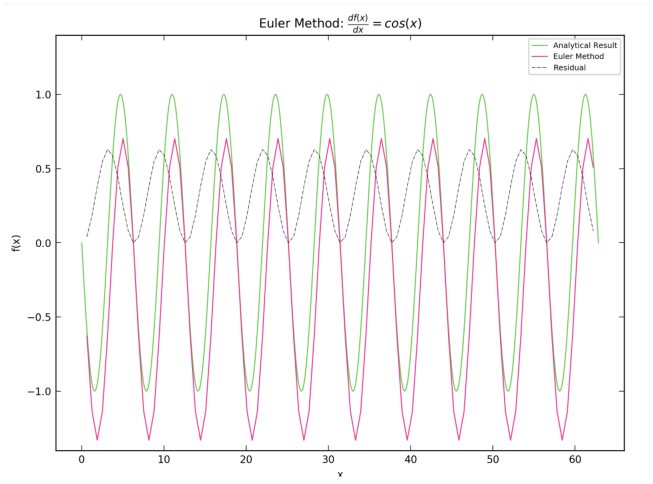


Figure 1: The results for the basic Euler method for 100 steps. Although easy to implement, there is a very clear offset in both phase and amplitude.

b) Using the Euler-Predictor Method Method

Answer: In the ordinary Euler Method, we evaluated the slope of the tangent line *at a point*. To make this method more accurate for nonlinear functions, we can instead evaluate the arithmetic mean of the slope between two steps. In this Modified-Euler formalism, we estimate the next value for y_n according to

$$y_{n+1} = y_n + h\frac{y'_n + y'_{n+1}}{2} \tag{2}$$

The result for this procedure for 100 steps is shown below in Figure 2.

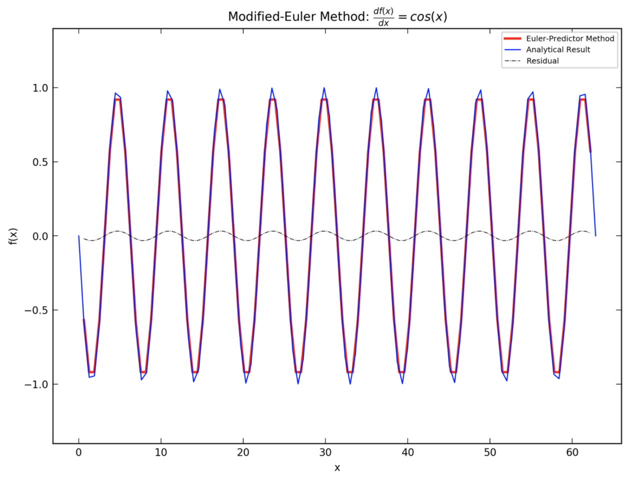


Figure 2: Same as Fig 1, but with the slight correction to the algorithm taking into account the mean of the slope between two steps. A significantly smaller residual is already apparent.

c) Using ODEINT

Answer: For this problem, we'll use the C++ distribution of ODEINT as stored on schooner. The results of this (also shown for 100 steps are shown below in Figure 3.

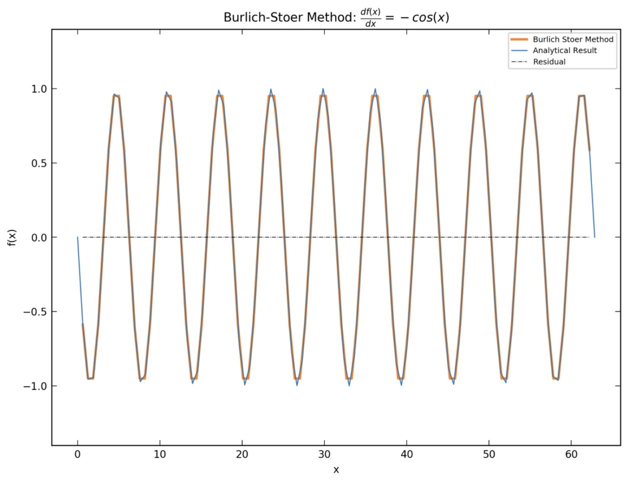


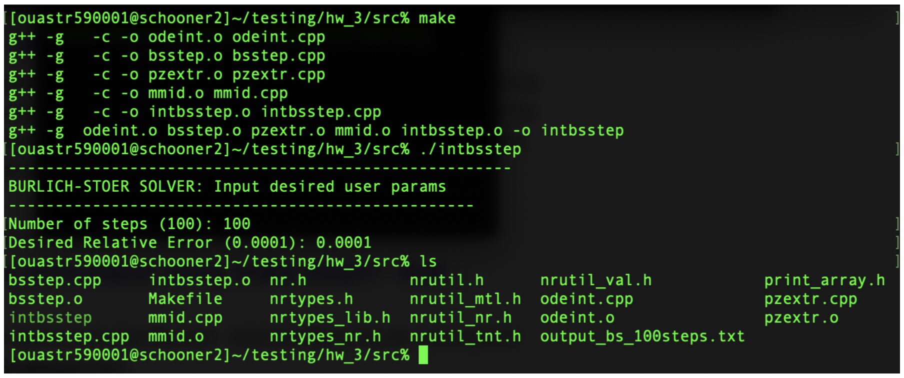
Figure 3: Same as previous plots but using the output from ODEINT. Note the extremely small residuals.

d) Discuss methods used above. Is one clealry better in accuracy? In ease of coding?

Answer: In terms of ease of coding, the most straightword to implement is the Euler method. Though this method is straight forward to implement, the numerical solution still differs considerbly in amplitude and phase compared to the analytical soluition (Fig 1.) With only a slight update to the algorithm, the Euler-Predictor-Corrector Method offers considerably improved accuracy without requiring any complicated coding (Fig 2).

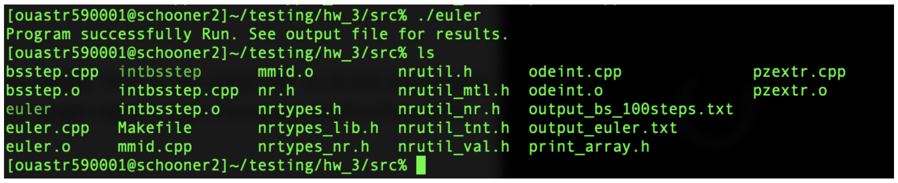
A quick quantitative assessment of the accuracy of the three algorithms is evident from the residual in each figure. We see that the Burlich-Stoer algorithm used by ODEINT is is significantly more accuarte than the two Euler methods discussed above. However, after the extreme headache required to get this algorithm up and running, it is difficult to say that this accuracy is justified by the difficulty coding it. Thus, **Of the three algorithms, the Euler-Predictor-Corrector method strikes the best balance of ease of coding and accuracy.**

APPENDIX:



```
[ouastr590001@schooner2]~/testing/hw_3/src% make
g++ -g   -c -o odeint.o odeint.cpp
g++ -g   -c -o bsstep.o bsstep.cpp
g++ -g   -c -o pzextr.o pzextr.cpp
g++ -g   -c -o mmid.o mmid.cpp
g++ -g   -c -o intbsstep.o intbsstep.cpp
g++ -g  odeint.o bsstep.o pzextr.o mmid.o intbsstep.o -o intbsstep
[ouastr590001@schooner2]~/testing/hw_3/src% ./intbsstep
------------------------------------------------------
BURLICH-STOER SOLVER: Input desired user params
--------------------------------------------------
Number of steps (100): 100
Desired Relative Error (0.0001): 0.0001
[ouastr590001@schooner2]~/testing/hw_3/src% ls
bsstep.cpp     intbsstep.o  nr.h           nrutil.h      nrutil_val.h            print_array.h
bsstep.o       Makefile     nrtypes.h      nrutil_mtl.h  odeint.cpp              pzextr.cpp
intbsstep      mmid.cpp     nrtypes_lib.h  nrutil_nr.h   odeint.o                pzextr.o
intbsstep.cpp  mmid.o       nrtypes_nr.h   nrutil_tnt.h  output_bs_100steps.txt
[ouastr590001@schooner2]~/testing/hw_3/src%
```

Figure 4: Terminal output showing that the provided makefile did, in fact, compile and produce a useable executable at somepoint.



```
[ouastr590001@schooner2]~/testing/hw_3/src% ./euler
Program successfully Run. See output file for results.
[ouastr590001@schooner2]~/testing/hw_3/src% ls
bsstep.cpp  intbsstep      mmid.o         nrutil.h      odeint.cpp              pzextr.cpp
bsstep.o    intbsstep.cpp  nr.h           nrutil_mtl.h  odeint.o                pzextr.o
euler       intbsstep.o    nrtypes.h      nrutil_nr.h   output_bs_100steps.txt
euler.cpp   Makefile       nrtypes_lib.h  nrutil_tnt.h  output_euler.txt
euler.o     mmid.cpp       nrtypes_nr.h   nrutil_val.h  print_array.h
[ouastr590001@schooner2]~/testing/hw_3/src%
```

Figure 5: Same as previous, but demonstrating that the second executable also compiled and ran.