Final Report

Paul Poulsen

CSCE 340: Numerical Analysis I

Dr. Ziguo Zhong

December 11, 2013

Introduction

This report summarizes my application of the computational methods taught in this course toward solving the four projects. The tables of computed values can be found at the end of the report.

I chose to implement the various numerical methods using the C Programming Language. Each source file is a standalone program that implements the given numerical method according to the project specification. A makefile is included with each project, and can be run using:

```
$ make all
$ make clean
```

The output of each executable is a series of comma-separated values that can be redirected to a csv file (with the exception of the Monte Carlo method, which just outputs a single value). From there, tables were imported into this report and styled.

Project 1

First, I rearranged Kepler's equation to:

$$f(x, y) = y - 0.9 \times \sin y - x = 0$$

This allows us to use the standard root-finding algorithms: bisection method, Newton's method, and secant method. Since x does not change value during an application of the bisection method, we can view Kepler's equation as:

$$f(y) = y - 0.9 \times \sin y - c = 0$$

where c is some constant that we will set for each value of x. For Newton's method, then, we use:

$$f'(y) = 1 - 0.9 \times \cos y$$

For each method, I applied the root-finding algorithm 30 times for equally spaced steps between $x = 0$ and $x = \pi$. The tolerance value (TOL) used for each method was 1×10^{-9} in an effort to provide accuracy around 8 decimal digits. For the bisection method, this tolerance will guarantee accuracy for 8 digits. However, for Newton's and the secant method, this only means the algorithm will cut off when the difference between one estimate and the next is below this tolerance.

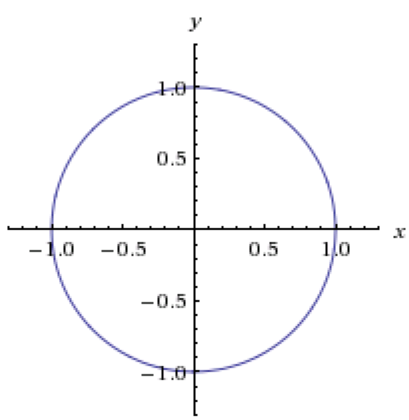
For the bisection method, the starting bounds were $a = 1 - TOL$ and $b = \pi + TOL$. I chose these values because all values of y would fall between 0 and π. For Newton's method, I chose a starting value $p_0 = {}^{\pi}/_{2}$ for all x, as this was in the middle of the two extremes. And for the secant method, I would change the starting values to $p_0 = x; p_1 = x - TOL$. These starting values helped avoid error at the upper and lower bounds while keeping execution time down. All methods ran until the tolerance value was reached (i.e. there was no cutoff at a certain step count).

The results for each method can be found at the end of the report.

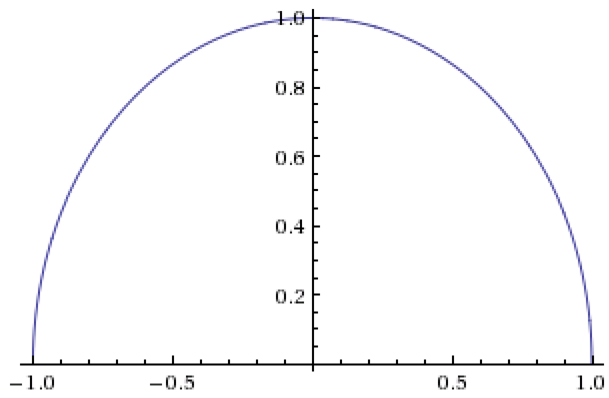
Project 2

The equation for a unit circle is:

$$y^2 + x^2 = 1$$

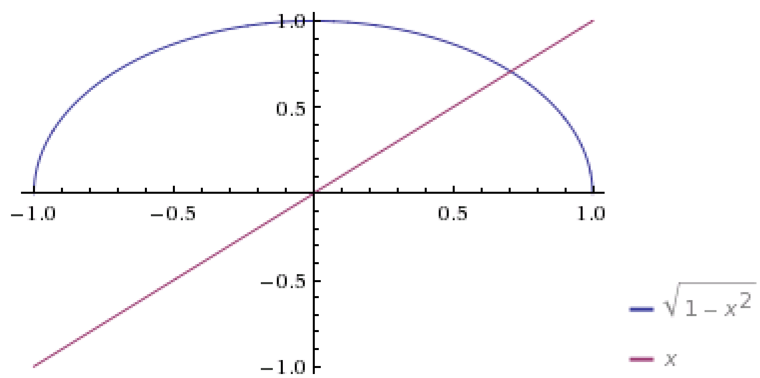


If we rearrange this into a function $y = f(x)$ we get a semicircle:



$$y = \sqrt{1 - x^2}$$

Then, if we plot $y = x$ on top of the circle, we see that it cuts the first quadrant into two eighths of circles.



Since $y = x$ intersects the circle in the first quadrant at $\frac{\pi}{4}$ radians, we know $x = \cos\frac{\pi}{4} = \frac{1}{\sqrt{2}}$ at the intersection. Thus:

$$Area\ of\ Circle = \pi = 8 \times Area\ of\ \frac{1}{8} Circle = 8 \int_0^{1/\sqrt{2}} \left(\sqrt{1 - x^2} - x\right) dx$$

Using this equation, I applied the Romberg algorithm as shown in class for a 10×10 grid (as can be seen in the results. The first column represents the value of the integral approximated with trapezoids. The first value is then a single trapezoid, while the tenth value is 2^9 trapezoids. Then, moving to the other columns, we smooth the trapezoids with Richardson's extrapolations.

Project 3

To create the Initial Value Problem (IVP) for this project, I first utilized the first fundamental theorem of calculus, which (roughly) says:

$$If$$

$$F(x) = \int_a^x f(t)dt$$

$$Then$$

$$F'(x) = f(x)$$

Applied to this problem, I was able to convert the integral equation to:

$$f'(\varphi) = \sqrt{1 - \frac{1}{4}\sin^2\varphi}$$

And, from the original equation, we easily obtain our initial value:

$$f(0) = 0$$

For the Runge-Kutta method, this is especially easy since the resulting equation is only a function of one variable, as opposed to $f(x, t)$. Thus, we only have to perform the adjustments to t (or, in this case, φ), and K_2 ends up being the same as K_3. The resulting table of values can be found at the end of the report.

Project 4

If the error is $\frac{1}{\sqrt{m}}$ where m is the number of sample points, and we want accuracy up to 3 decimal places, then our error can be no greater that 0.0005. Thus:

$$0.0005 = \frac{1}{\sqrt{m}} \to m = 4{,}000{,}000$$

For each point, I generate random variables x and y such that $0 \leq x \leq 1$ and $0 \leq y \leq 1$. Then, if $x^2 + y^2 < 1$, the point was considered inside the circle and I incremented n. After $4{,}000{,}000$ points, π was estimated as $\frac{n}{4{,}000{,}000} \times 4$. The resulting values are provided at the end of the report.

Results

Project 1

Bisection Method

n	x	y
0	0.00000000	0.00000000
1	0.10833078	0.66047019
2	0.21666156	0.94734080
3	0.32499234	1.14441313
4	0.43332312	1.30069184
5	0.54165391	1.43314022
6	0.64998469	1.54978605
7	0.75831547	1.65511781
8	0.86664625	1.75192341
9	0.97497703	1.84206547
10	1.08330781	1.92685728
11	1.19163859	2.00726408
12	1.29996937	2.08401931
13	1.40830016	2.15769572
14	1.51663094	2.22875105
15	1.62496172	2.29755847
16	1.73329250	2.36442758
17	1.84162328	2.42961936
18	1.94995406	2.49335692
19	2.05828484	2.55583365
20	2.16661562	2.61721925
21	2.27494640	2.67766448
22	2.38327719	2.73730485
23	2.49160797	2.79626363
24	2.59993875	2.85465421
25	2.70826953	2.91258213
26	2.81660031	2.97014679
27	2.92493109	3.02744290
28	3.03326187	3.08456181
29	3.14159265	3.14159265

Newton's Method

n	X	y
0	0.00000000	0.00000000
1	0.10833078	1.03263704
2	0.21666156	1.21956369
3	0.32499234	1.19760481
4	0.43332312	1.34233821
5	0.54165391	1.46035784
6	0.64998469	1.56586045
7	0.75831547	1.66390312
8	0.86664625	1.75640548
9	0.97497703	1.92327468
10	1.08330781	1.98310242
11	1.19163859	2.04517841
12	1.29996937	2.10889436
13	1.40830016	2.17356622
14	1.51663094	2.46134208
15	1.62496172	2.48188804
16	1.73329250	2.50855518
17	1.84162328	2.54053698
18	1.94995406	2.57711165
19	2.05828484	2.61763977
20	2.16661562	2.66155666
21	2.27494640	2.70836306
22	2.38327719	2.75761554
23	2.49160797	2.80891758
24	2.59993875	2.86191139
25	2.70826953	2.91627067
26	2.81660031	2.97169409
27	2.92493109	3.02789952
28	3.03326187	3.08461875
29	3.14159265	3.14159265

Secant Method

n	x	y
0	0.00000000	0.00000000
1	0.10833078	1.03263712
2	0.21666156	0.75296523
3	0.32499234	0.91093687
4	0.43332312	1.08647581
5	0.54165391	1.26028020
6	0.64998469	1.42136092
7	0.75831547	1.56544503
8	0.86664625	1.69237772
9	0.97497703	1.63560714
10	1.08330781	1.76588886
11	1.19163859	1.88462249
12	1.29996937	1.99263318
13	1.40830016	2.09109283
14	1.51663094	2.46134208
15	1.62496172	2.48188804
16	1.73329250	2.50855514
17	1.84162328	2.54053692
18	1.94995406	2.57711173
19	2.05828484	2.61763975
20	2.16661562	2.66155665
21	2.27494640	2.70836306
22	2.38327719	2.75761554
23	2.49160797	2.80891759
24	2.59993875	2.86191145
25	2.70826953	2.91627070
26	2.81660031	2.97169408
27	2.92493109	3.02789951
28	3.03326187	3.08461875
29	3.14159265	3.14159265

Project 2

2.828427124746									
3.059964873438	3.137144123002								
3.120881408255	3.141186919860	3.141456439651							
3.136392314542	3.141562616637	3.141587663089	3.141589746001						
3.140291076550	3.141590663886	3.141592533702	3.141592611014	3.141592622249					
3.141267164509	3.141592527162	3.141592651380	3.141592653248	3.141592653414	3.141592653444				
3.141511275368	3.141592645654	3.141592653553	3.141592653588	3.141592653589	3.141592653589	3.141592653590			
3.141572308662	3.141592653093	3.141592653589	3.141592653590	3.141592653590	3.141592653590	3.141592653590	3.141592653590		
3.141587567335	3.141592653559	3.141592653590	3.141592653590	3.141592653590	3.141592653590	3.141592653590	3.141592653590	3.141592653590	
3.141591382025	3.141592653588	3.141592653590	3.141592653590	3.141592653590	3.141592653590	3.141592653590	3.141592653590	3.141592653590	3.141592653590

Project 3

φ	$f(\varphi)$	φ	$f(\varphi)$	φ	$f(\varphi)$	φ	$f(\varphi)$	φ	$f(\varphi)$
0	0	20	0.347059	40	0.684115	60	1.005816	80	1.313789
1	0.01745174	21	0.364217	41	0.700582	61	1.021495	81	1.328959
2	0.03490083	22	0.381348	42	0.71701	62	1.037138	82	1.344117
3	0.05234593	23	0.39845	43	0.733397	63	1.052746	83	1.359264
4	0.06978574	24	0.415524	44	0.749743	64	1.068321	84	1.374402
5	0.08721892	25	0.432569	45	0.766049	65	1.083863	85	1.389533
6	0.10464418	26	0.449582	46	0.782314	66	1.099374	86	1.404657
7	0.12206021	27	0.466564	47	0.798538	67	1.114853	87	1.419777
8	0.13946572	28	0.483513	48	0.814721	68	1.130303	88	1.434894
9	0.15685943	29	0.500429	49	0.830864	69	1.145724	89	1.450009
10	0.17424006	30	0.517311	50	0.846966	70	1.161117	90	1.465124
11	0.19160637	31	0.534158	51	0.863027	71	1.176484		
12	0.20895711	32	0.55097	52	0.879049	72	1.191825		
13	0.22629105	33	0.567745	53	0.89503	73	1.207142		
14	0.24360698	34	0.584484	54	0.910972	74	1.222436		
15	0.26090372	35	0.601185	55	0.926875	75	1.237708		
16	0.27818008	36	0.617849	56	0.942739	76	1.252959		
17	0.29543493	37	0.634474	57	0.958564	77	1.268192		
18	0.31266712	38	0.65106	58	0.974352	78	1.283407		
19	0.32987555	39	0.667607	59	0.990102	79	1.298605		

Project 4

n	$\hat{\pi}$
1	3.141770
2	3.143038
3	3.142262
4	3.142662
5	3.142370
6	3.140843
7	3.141611
8	3.140476
9	3.141270
10	3.143634