Okeefe Niemann
4/13/2014
1281465
PHYS 115

Assignment 1

1 Question 1

1.1 Part a

- Input:

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

double f(double x)
{
        return 1 / (1+x);
}

int main()
{
        double x, a, b, h, exact, ans, trap();
        int i, n, iter, ITERMAX;

        exact = log((double) 2);
        ITERMAX= 8;
        b = 1.0;
        a = 0;

        printf ("         h         ans       (exact - ans)/h^2 \n");
        n=1;

        for (iter = 0; iter < ITERMAX; iter++) {
                h = (b - a) / n;
                ans = trap (f, a, b, n);
                n *= 2;
                printf ("%12.5f %12.5f%12.5f \n", h, ans, (ans-exact)/(h*h));
        }
}
```

- Output:

```
           h           ans    (exact - ans)/h^2
     1.00000       0.75000     0.05685
     0.50000       0.70833     0.06074
     0.25000       0.69702     0.06203
     0.12500       0.69412     0.06238
     0.06250       0.69339     0.06247
     0.03125       0.69321     0.06249
     0.01562       0.69316     0.06250
     0.00781       0.69315     0.06250
```

1.2 Part b

$$-\frac{1}{12}[f'(b) - f'(a)] = -\frac{1}{12}[-.25 + 1] = .06245 = \frac{I - T}{h^2}$$

Therefore, the constant that h approached in part (a) corresponds to the coefficient of the leading error in the trapezium rule.

2 Question 2

- Input:

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

double f(double x)
{
        return exp(-x)*sin(x);
}

int main() {
        double x, a, b, h, exact, ans, prevans, EPS, sum, simp();
        int i, n;

        a = 0;
        b = 2.0;
        EPS = 1.e-8;
        ans = 1.e50;

        printf ("       h         ans         (ans - prevans)/h^4\n");

        n = 1;

        while (fabs(ans - prevans) > EPS) {

        h = (b - a) / n;
                prevans = ans;

                ans = simp(f, a, b, n);
                n *= 2;
                printf("%12.9f %14.10f   %.5e \n", h, ans, (ans - prevans)/(h*h*h*h));
        }
}
```

- Output:

```
  h         ans         (ans - prevans)/h^4
 2.000000000   0.0820400165   -6.25000e+48
 1.000000000   0.4537665091   3.71726e-01
 0.500000000   0.4659349656   1.94695e-01
 0.250000000   0.4665884029   1.67280e-01
 0.125000000   0.4666271191   1.58582e-01
 0.062500000   0.4666295042   1.56309e-01
 0.031250000   0.4666296527   1.55735e-01
 0.015625000   0.4666296620   1.55591e-01
```

Comment: The precision can be checked by recognizing that the difference between the last and second to last term is of order e^{-8}.

3 Question 3

- Input:

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

double f(double x)
{
        return exp(-(x*x)/2);
}


double trap (double f(double), double a, double b, int n)
{
        double h, sum;
        int i, l;

        h = (b - a) / n;
        sum = 0.5 * (f(a) + f(b));
        for (l = 1; l < n; l++) sum += f(a + l * h);
                return h * sum;
}

int main()
{
        double x, a, b, h, exact, ans, prevans, EPS, array[20][20], trap();
        int i, j, k, l, n;

        b = 1.0;
        a = 0;


        n = 1;
        k = 1;
        ans = 1.e5;
        EPS = 1.e-12;

while (fabs(ans - prevans) > EPS){
        prevans = ans;
        for (j = 0; j < k; j++)
        {
                if (j == 0)
                {
                        array[k][0] = trap(f, a, b, n);
                        printf("n=%i: %.12e  ", n, array[k][0]);
                }
                else
                {
                        array[k][j] = (pow(4,j)*array[k][j-1] - array[k-1][j-1])/(pow(4, j)-1);
                        printf("%.12e  ", array[k][j]);
                }
        ans  = array[k][j];
        }
        n *= 2;
        k ++;
        printf("\n");
        }

return 0;
```

```
}
```

- output:

```
n=01: 8.0326533e-01
n=02: 8.4288112e-01  8.5608638e-01
n=04: 8.5245877e-01  8.5565132e-01  8.5562231e-01
n=08: 8.5483423e-01  8.5562605e-01  8.5562436e-01  8.5562439e-01
n=16: 8.5542693e-01  8.5562449e-01  8.5562439e-01  8.5562439e-01  8.5562439e-01
n=32: 8.5557503e-01  8.5562440e-01  8.5562439e-01  8.5562439e-01  8.5562439e-01  8.5562439e-01
```

4 Question 4

Comment: Proof of the midpoint rule is derived on attached handwritten pages.

5 Question 5

Comment: Parts a-c are derived on attached handwritten pages.

5.1 Part d

- Input:

```
\end{ver#include<stdio.h>
#include<math.h>

double f(double x)
{

        return 22 / 14 * sin(x) * sin(theta/2) * sin(theta/2) + 22 / 14 +  66 / 112 * sin(theta/2)*sin(
                            /* where a value equals "theta," place value for theta_m */

}

double trap (double f(double, double), double a, double b, int n)
{
double h, sum, omega;
int i;

h = (b - a) / n;
sum = 0.5 * (f(a) + f(b));
        for (i = 1; i < n; i++) sum += f(a + i * h);
                return h * sum;
}

int main() {
        double x, a, b, h, exact, ans, prevans, EPS, sum, trap(), omega;
        int i, j, n;

        a = 0;
        b = 11 / 7;
        EPS = 1.e-7;
        ans = 1.e50;

        printf ("       h         ans         precision\n");
```

```
        n = 1;

        while (fabs(ans - prevans) > EPS) {

                h = (b - a) / n;
                prevans = ans;

ans = trap(f, a, b, n);
n *= 2;
printf("%12.9f %.5e %.5e \n", h, ans, (ans - prevans));
}
        }
}
```

- Output:

```
Theta        h         ans     precision
 .1       0.015625000 1.00115 7.00873e-08
 .2       0.007812500 1.00636 6.99112e-08
22/28     0.001953125 1.09737 6.42515e-08
22/14     0.000976562 1.36999 5.48349e-08
66/28     0.000976562 1.67253 9.35866e-08
```

6 section 6

- Input:

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

double f(double x)    /* declares function */
{
        return  2 * sin(x*x) / (x*x);
}

double midpt(double f (double), double a, double b, double n)
{
        double h, sum;
        int i;

        h = (b - a) / n;
        sum = 0;
        for (i = 0; i < n; i++)
{
        sum += f(a + (h / 2) + i * h);    /* sums each step starting from their midpoints */
}
return h * sum;
}

int main() {
        double x, a, b, h, exact, ans, prevans, EPS, sum, midpt();
        int i, n;
```

```
        a = 0;
        b = 1.0;
        EPS = 1.e-4;
        ans = 1.e50;

        printf ("       h         ans         (ans - prevans)/h^4\n");

        n = 1;

        while (fabs(ans - prevans) > EPS) {      /* compares current and previous result repeatedly unt
desired  precision is achieved */
               h = (b - a) / n;
               prevans = ans;

               ans = midpt(f, a, b, n);
               n *= 2;
               printf("%12.9f %14.10f   %.5e \n", h, ans, (ans - prevans));
        }
}
```

- Output:

```
       h           ans          precision
 1.000000000   1.9792316740   -1.00000e+50
 0.500000000   1.9474427273   -3.17889e-02
 0.250000000   1.9382777519   -9.16498e-03
 0.125000000   1.9359384039   -2.33935e-03
 0.062500000   1.9353510013   -5.87403e-04
 0.031250000   1.9352039970   -1.47004e-04
 0.015625000   1.9351672364   -3.67606e-05
```