

Numerical Analysis – Mth 351

Archive – Spring 2001 Files

July 15, 2001

This archive contains the midterm test, final exam, and assignments from Mth 351, Spring 2001. The original test instructions, headers and formatting have not been preserved.

Contents

1	Midterm Test	1
2	Final Exam	3
3	Assignment 1	5
4	Assignment 2	6
5	Assignment 3	7
6	Assignment 4	8
7	Assignment 4 Results	13
8	Contact Information	15

1 Midterm Test

Problem 1. (25 points if correct, 0 points if wrong). Consider the function $f(x) = x^2 - \cos(4\pi x)$. The function f is guaranteed to have a root in each of the intervals

- A.) $[0, \frac{1}{8}]$, $[\frac{1}{8}, \frac{1}{4}]$, $[\frac{3}{8}, \frac{1}{2}]$ B.) $[0, \frac{1}{8}]$, $[\frac{3}{8}, \frac{1}{2}]$, $[\frac{3}{4}, \frac{7}{8}]$
C.) $[0, \frac{1}{8}]$, $[\frac{3}{8}, \frac{1}{2}]$, $[\frac{1}{2}, \frac{5}{8}]$ D.) $[0, \frac{1}{8}]$, $[\frac{3}{8}, \frac{1}{2}]$, $[\frac{5}{4}, \frac{11}{8}]$ E.) None of the above.

← Letter corresponding to your answer to problem 1.

Problem 2. (25 points if correct, 0 points if wrong). If $g(x) = \sin(x^3 - x^2)$ the Taylor polynomial of $g(x)$ about the origin and of degree ≤ 7 is

$$p(x) = -x^2 + x^3 + \frac{1}{6}x^6 - \frac{1}{2}x^7.$$

One can show $|g^{(8)}(x)| \leq 32000$ for $0 \leq x \leq 1$. Use the given information and a Taylor remainder estimate to obtain an error estimate of the form

$$|g(x) - p(x)| \leq M x^8 \quad \text{for } 0 \leq x \leq 1.$$

One can show that the best value for M is $M = \frac{1}{2}$, but the Taylor estimate does not do that well. Indeed the Taylor estimate gives

- A.) $M = \frac{35}{63}$ B.) $M = \frac{40}{63}$
 C.) $M = \frac{45}{63}$ D.) $M = \frac{50}{63}$ E.) None of the above.

← Letter corresponding to your answer to problem 2.

Problem 3. (25 points if correct, 0 points if wrong). Given $f(2.01) = 3.54$, $f(2.03) = 3.57$, $f(2.05) = 3.71$ and $f(2.07) = 3.75$ use a central (symmetric) second order numeric differentiation approximation to estimate $f''(2.03)$.

- A.) 2.75 B.) 4.25
 C.) 5.50 D.) 275 E.) None of the above.

← Letter corresponding to your answer to problem 3.

Problem 4. (25 points). If

$$g(x) = \frac{x^2 + 1}{2x - 1}$$

then $g: [\frac{3}{2}, 2] \rightarrow [\frac{3}{2}, 2]$ is a contraction map. If $x_0 \in [\frac{3}{2}, 2]$ and $x_{n+1} = g(x_n)$ for $n \geq 0$ compute

$$\alpha = \lim_{n \rightarrow \infty} x_n.$$

Problem 5. (25 points). Consider a (lousy) computer with a floating point representation with a mantissa of only 8 bits, and suppose it employs chopping rather than rounding. Assume the mantissa m is normalized so $1 \leq m < 2$. **Part (A)** What (logical) exponent do you store when you store $\frac{22}{7}$? **Part (B)** What error do you incur in storing $\frac{22}{7}$? (You may express the error in decimal).

Problem 6. (25 points). Write a *brief technical* essay describing what topic in Mth 351 that you have enjoyed the most so far. Do not write more than two or three paragraphs. Make sure you employ reasonably correct English and incorporate one or two equations or symbolic statements.

2 Final Exam

Note: Some problems may have more than one technically “correct” answer. In that case the best answer receives full credit, whereas correct, but not best, answers receive partial credit. For example, a small integer greater than 5 is probably 6 or a bit more, whereas 10^9 is also technically correct, but probably not best (depending on what selections are available). Problems with partial credit are indicated by multiple points, e.g., 20, 30.

Problem 7. (30 points if correct, 0 points if wrong). Suppose we have a binary computer with a mantissa of length 9. Suppose we chop numbers to store them and suppose we do not pack the most significant bit. Assuming we do not incur overflow, what is the roundoff error in storing $32/5$?

- A.) $\frac{1}{128}$ B.) $\frac{3}{320}$
 C.) $\frac{1}{640}$ D.) $\frac{3}{1280}$ E.) None of the foregoing.

← Letter corresponding to your answer to problem 7.

Problem 8. (30 points if correct, 0 points if wrong). The function $f(x) = 4x - \exp(x)$ has two roots, approximately 0.357402956 and 2.153292364. Suppose you take as your initial guess $x_0 = 1$ and apply Newton’s method to calculate successive approximations x_1, x_2, \dots . Find x_2 and from the list below select the number closest to x_2 .

- A.) 0.33333 B.) 0.35725
 C.) 2.11111 D.) 2.15333 E.) 2.15321

← Letter corresponding to your answer to problem 8.

Problem 9. (30, 15 points if correct, 0 points if wrong). Consider the polynomial $p(x) = x^5 - 4x^4 + 2x^3 + 12x - 3$. A quick calculation shows $p(2) = 5$ and $p(3) = 6$. Compute $p\left(\frac{5}{2}\right)$ and use your result to estimate the number of roots of $p(x)$ in the interval $(2, 3)$.

- A.) none B.) at least 1
 C.) at least 2 D.) impossible to tell E.) None of the foregoing.

← Letter corresponding to your answer to problem 9.

Problem 10. (30 points if correct, 0 points if wrong). Given the following Newton divided differences

$x_0 = 0.11$	$x_1 = 0.43$	$x_2 = 0.54$	$x_3 = 0.75$
$f[x_0] = 0.8717$	$f[x_1] = 0.7745$	$f[x_2] = 0.7027$	$f[x_3] = 0.4871$
	$f[x_0, x_1] = -0.3038$	$f[x_1, x_2] = -0.6527$	$f[x_2, x_3] = -1.027$
		$f[x_0, x_1, x_2] = -0.8114$	$f[x_1, x_2, x_3] = \dots$
			$f[x_0, x_1, x_2, x_3] = \dots$

compute the divided difference $f[x_0, x_1, x_2, x_3]$. Select the closest number from the list below.

- A.) -1.170 B.) -0.991
 C.) -0.560 D.) -0.347 E.) -0.229

← Letter corresponding to your answer to problem 10.

Problem 11. (30, 10 points if correct, 0 points if wrong). Let f be a function with Taylor polynomial at the origin, $p(x)$ of degree 5, say

$$p(x) = \frac{1}{2} + \frac{7}{16}x^3 + \frac{3}{4}x^5.$$

Suppose we know $|f^{(6)}(x)| \leq \frac{3}{8}$ for $|x| \leq 1$. We decide to use $p(1/2) = \frac{37}{64}$ to approximate $f(1/2)$. Find a good upper bound for the absolute error $|f(1/2) - p(1/2)|$.

- A.) $\frac{1}{860160}$ B.) $\frac{1}{122880}$
 C.) $\frac{1}{46080}$ D.) 3.07×10^{18} E.) None of the foregoing.

← Letter corresponding to your answer to problem 11.

Problem 12. (30, 20 points if correct, 0 points if wrong). Let $q(x) = -x^5 + \frac{7}{2}x^4 - \frac{77}{16}x^3 + \frac{13}{4}x^2 - \frac{63}{64}x + \frac{9}{64}$. One can show $0 \leq q(x) \leq \frac{9}{64}$ for each x with $0 \leq x \leq 1$ and $|q'(x)| \leq \frac{63}{64}$ for each x with $0 \leq x \leq 1$. Thus we conclude:

- A.) $p(x)$ has a unique fixed point α in $[0, 1]$ and $\alpha > 0.15$
 B.) $p(x)$ has a unique fixed point α in $[0, 1]$ and $\alpha < 0.15$
 C.) $p(x)$ has a unique fixed point α in $[0, 1]$
 D.) $p(x)$ has no fixed points
 E.) None of the foregoing.

← Letter corresponding to your answer to problem 12.

Problem 13. (30 points if correct, 0 points if wrong). Find the interpolation polynomial $p(x)$ of degree ≤ 3 for the function $f(x) = (1 + x^2)^{-1}$ with nodes at $x = -1, 1, 2, 3$. Then compute $p(0)$.

- A.) 1.0 B.) 0.9
 C.) 0.8 D.) 0.7 E.) None of the foregoing.

← Letter corresponding to your answer to problem 13.

Problem 14. (30 points if correct, 0 points if wrong). Denote by T_n the compound trapezoidal rule with n subintervals (for some function f on some interval $[a, b]$). If n is even we can apply Richardson extrapolation to obtain Simpson's rule $S_n = \frac{4}{3}T_n - \frac{1}{3}T_{n/2}$. If n is divisible by 4 we can extrapolate once again to obtain Bode's rule $B_n = \frac{16}{15}S_n - \frac{1}{15}S_{n/2}$. Given the information in the following table compute B_8 .

$T_1 = 8.389056$	$T_2 = 6.912810$	$T_4 = 6.521610$	$T_8 = 6.422298$
...	$S_2 = 6.420728$	$S_4 = 6.391210$	$S_8 =$
...	...	$B_4 =$	$B_8 =$

- A.) 6.389194 B.) 6.389060
 C.) 6.391210 D.) 6.389242 E.) None of the foregoing.

← Letter corresponding to your answer to problem 14.

Problem 15. (30 points if correct, 0 points if wrong). Let f be a smooth function. Compute

$$\lim_{h \rightarrow 0} \frac{2f(a) - 3f(a+h) + f(a+3h)}{3h^2}.$$

Hint: Use L'Hospital's rule or Taylor polynomials.

- A.) 0 B.) $f(a)$
 C.) $f'(a)$ D.) $f''(a)$ E.) None of the foregoing.

← Letter corresponding to your answer to problem 15.

Problem 16. (30 points). Write one or two brief paragraphs describing the topic in Mth 351 that you enjoyed most, or found most useful. You may wax poetic, but be sure to include some technical information. Humor is great if tasteful and relevant.

3 Assignment 1

Let f be a smooth function defined say in the interval $[a, b]$ where $a < b$. Let $0 < h < b - a$. Then there is a unique polynomial $p_h(x)$ of degree ≤ 3 ,

$$p_h(x) = A_h + B_h(x - a) + C_h(x - a)^2 + D_h(x - a)^3,$$

such that

$$p_h(a) = f(a), \quad p'_h(a) = f'(a), \quad p_h(a+h) = f(a+h), \quad p'_h(a+h) = f'(a+h).$$

Clearly

$$A_h = f(a) \quad \text{and} \quad B_h = f'(a).$$

Part (A). Compute the coefficients C_h and D_h . As a check on your work note

$$C_h = \frac{3f(a+h) - 3f(a) - f'(a+h)h - 2f'(a)h}{h^2}.$$

Part (B). Once you have all the coefficients use L'Hôpital's rule to compute

$$\lim_{h \rightarrow 0} p_h(x).$$

Part (C). (Philosophical question) Explain why your answer to Part (B) seems reasonable.

Part (D). Explain why

$$f''(a) \approx 2 \frac{3f(a+h) - 3f(a) - f'(a+h)h - 2f'(a)h}{h^2}$$

if h is small.

Part (E). Let $f(x) = \sin(x)$, $a = 0$ and $h = \pi$ above and let $p(x) = p_\pi(x)$. Let $q(x)$ be the Taylor-Maclaurin polynomial of degree ≤ 3 for $\sin(x)$. Graph the errors $f(x) - p(x)$ and $f(x) - q(x)$ on the interval $[0, \pi]$. Comment on your result. As a check on your work note that $p(x) = x - \frac{1}{\pi}x^2$.

Part (F). Let $f(x) = \cos(x)$, $a = 0$ and $h = \pi$ above and let $p(x) = p_\pi(x)$. Let $q(x)$ be the Taylor-Maclaurin polynomial of degree ≤ 3 for $\cos(x)$. Graph the errors $f(x) - p(x)$ and $f(x) - q(x)$ on the interval $[0, \pi]$. Comment on your result.

If you have difficulty producing the graphs in Part (E) and in Part (F) make a table of values of $f(x) - p(x)$ and $f(x) - q(x)$ on the interval $[0, \pi]$ instead, and comment on what you see.

4 Assignment 2

Recall the popular order 2 derivative estimates

$$\begin{aligned} x'(a) &= \frac{x(a+h) - x(a-h)}{2h} + \mathcal{O}(h^2) \\ x''(a) &= \frac{x(a+h) - 2x(a) + x(a-h)}{h^2} + \mathcal{O}(h^2) \end{aligned}$$

Consider a body with position $x(t)$ moving through a viscous fluid in accord with the equation

$$x'' + kx' = 0$$

where k is a constant. The following times and positions are recorded by a diligent experimenter.

x(t)	3.099	3.476	3.664	3.758
t	0.300	0.500	0.700	0.900

Use the first three data points and the last three data points together with the derivative approximations above to obtain two estimates of $k = -\frac{x''}{x'}$.

Comment on the agreement, or the lack of it, between the two estimates.

5 Assignment 3

Your assignment is to compute the unit round for various floating point data type for various compilers. I have provided some sample c code below but you may prefer to write your own, or to try a different language.

GNU gcc on i386 Linux yields an executable which produces the output

```
Unit round - float: 1.192093e-07      (float.h claims 1.192093e-07)
Unit round - double: 2.220446e-16     (float.h claims 2.220446e-16)
Unit round - long double: 1.084202e-19 (float.h claims 1.084202e-19)
```

but your result will not be so tidy with other compilers on other systems.

Turn in a copy of your code and a copy of your results. For each result indicate the compiler used and the type of machine you ran it on. There are many different computers available to you on campus, so try at least two different types.

```
/*
** epsilon.c
** Mth 351 Spring 2001 - Bent Petersen
**
** Microsoft C/C++ cl 32-bit 12.00.8168      cl /Od epsilon.c
** GNU C/C++ gcc egcs-2.91.66 (egcs 1.1.2)  gcc -o epsilon -i epsilon.c
**
** This code looks pretty silly but computing the unit
** round (epsilon) is a bit tricky. The obvious code may
** not work, or may work but yield incorrect results. It
** is necessary to force stores of intermediate results to
** prevent the compiler from doing the calculations at the
** full precision of the FPU.
*/

#include <stdio.h>
#include <float.h>

int main( void ) {

/* gcc wants main() to return an int - void main(void) produces */
/* a warning message. MS cl doesn't care. */

float x1=1.0, y1, z1=2.0;
double x2=1.0, y2, z2=2.0;
long double y3=2.0;
```

```

while ( z1 > 1.0 ) {
    y1 = x1;
    x1 /= 2.0;
    z1 = 1.0 + x1;
}
printf("\nUnit round - float: %e\t(float.h claims %e)",
       y1, FLT_EPSILON );

while ( z2 > 1.0 ) {
    y2 = x2;
    x2 /= 2.0;
    z2 = 1.0 + x2;
}
printf("\nUnit round - double: %e\t(float.h claims %e)",
       y2, DBL_EPSILON );

while ( y3 + 1.0 > (long double)1.0 )
    y3 /= 2.0 ;
printf("\nUnit round - long double: %e\t(float.h claims %e)\n",
       2.0*(double)y3, (double)LDBL_EPSILON );

/* In float.h for gcc LDBL_EPSILON is declared as a long double */
/* and therefore the cast to a double is required for printf(). */
/* Likewise y3 must be cast to a double or printf() will print */
/* nonsense. */

return 0;
}

```

6 Assignment 4

This assignment illustrates the approximation of a function by means of Lagrange interpolation polynomials. You will be asked to compute the Lagrange interpolation polynomial for several different choices of nodes. The observed error will depend quite a bit on the choice of interpolation points (nodes). Be sure that your comments address this point.

Let $f(x) = e^{2x}$.

For each of the following sets of nodes

- **Case (A)** $x_k = -1 + \frac{k}{4}$, $k = 0, \dots, 8$
- **Case (B)** $x_k = -\cos\left(\frac{2k+1}{18}\pi\right)$, $k = 0, \dots, 8$ (ČEBYŠEV)
- **Case (C)** $x_k = -1 + \frac{k}{8}$, $k = 0, \dots, 8$

do the following:

1. Compute the Newton divided differences

$$f[x_0], f[x_0, x_1], \dots, f[x_0, x_1, \dots, x_8]$$

2. Next compute the Lagrange interpolation polynomial by using the Newton formula

$$\begin{aligned} p_8(x) = & f[x_0] + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1) + \\ & \dots + f[x_0, x_1, \dots, x_8](x - x_0)(x - x_1) \dots (x - x_7). \end{aligned}$$

at each of the 25 points

$$u_k = -1 + \frac{k}{12}, \quad k = 0, 1, \dots, 24$$

Compute also the values of the function f at each of these points and the actual error in $p_8(u_k)$.

3. Use the error estimate

$$\text{error at } u_k = \frac{1}{9!} f^{(9)}(\xi)(u_k - x_0)(u_k - x_1) \dots (u_k - x_8)$$

and an upper bound for $|f^{(9)}(\xi)|$ on the interval $[-1, 1]$ to obtain a theoretical upper bound for the error for each u_k .

4. Write comments on the quality of each approximation, the relation between the observed error and the theoretical error, and whatever else comes to mind. You might also consider graphing the errors. Be creative!

You may use any programming language you like, or a spreadsheet, or if you are a true glutton for punishment, a calculator. If you use Maple, you will have the easiest time of it. See either of

http://www.peak.org/~petersen/maple/maple_notes.html#351s2001interp

http://www.peak.org/~petersen/maple/maple_notes.html#351divdiff

for an indication of how easy it is to do interpolation in Maple.

I have appended some c code which, while not very c idiomatic, does work. You may use it as is, or adapt it to your situation - as you wish.

You can download `interpol.c` from my web page.

```
/*
 * interpol.c - Lagrange interpolation polynomial example
 *             Mth 351 Spring 2001, Bent E. Petersen
 *             ANSI C
 *
 * This is lousy code since so much of it is hardwired and
 * there are numerous global variables. It's probably a hanging
 * offense to write such stuff, but it works. Just don't emulate it.
 *
 * MS Visual C/C++ compiler : cl interpol.c
 * GNU gcc compiler         : gcc -i interpol.c -o interpol -lm
 *
 * It might be useful to capture the output in a file to view
 * later, e.g.,
 *
 * interpol > interpol.txt
 */
```

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

```
#define PI          (double)3.141592653589793238462643
#define N           8
#define NUMPOINTS  25
#define EXAMPLES   3
```

```
static char * example[] = { "(A)", "(B)", "(C)" };
```

```
static double  nodex[EXAMPLES][N+1];
static double  diffs[EXAMPLES][N+1];
```

```
static double  u[NUMPOINTS];
static double  f[NUMPOINTS];
static double  Lagr[EXAMPLES][NUMPOINTS];
static double  err[EXAMPLES][NUMPOINTS];
```

```
void fillarrays( void );
void calcdiffs( void );
void calcpolys( void );
void esterror( void );
```

```
/*
 * -----
 * compare Lagrange interpolation polynomial values with actual
 * values of exp(2x) for EXAMPLES different choice of nodes
 * -----
 */
```

```

*/
int main( void )
{
    int      c,k;
    double   actualerr, worsterr;
    fillarrays();
    calcdiffs();
    calcpolys();
    esterror();
    printf("\nMth 351 interpol - Lagrange interpolation polynomial example");
    for ( c=0; c<EXAMPLES; c++ )
        {
            worsterr = (double)0;
            printf("\n\nCASE %s:\n", example[c]);
            printf("\n Point      Actual Val      Estimated Val "
                "\n Actual Err      Theoretical Err\n");
            for ( k=0; k<NUMPOINTS; k++ )
                {
                    actualerr = f[k] - Lagr[c][k];
                    printf("\n%+7.4f  %+14.10f  %+14.10f  %+14.10f  %+14.10f", u[k],
                        f[k], Lagr[c][k], actualerr, err[c][k] );
                    worsterr = (worsterr>fabs(actualerr))? worsterr: fabs(actualerr);
                }
            printf("\n\nWorst error in CASE %s: %14.10f", example[c], worsterr );
        }
    printf("\n");
    return(0);
}
/*
* -----
* calculate values of Lagrange interpolation polynomials
* by Newton's representation and a modified Horner method
* -----
*/
void calcpolys()
{
    int      j,k,c;
    double   temp;
    for ( c=0; c<EXAMPLES; c++ )
        for ( j=0; j<NUMPOINTS; j++ )
            {
                /* modified Horner here - we are */
                temp = diffs[c][N]; /* evaluating Lagrange poly at u[k] */
                for ( k=N-1; k>=0; k-- )
                    temp = temp * (u[j] - nodex[c][k]) + diffs[c][k];
                Lagr[c][j] = temp;
            }
}

```

```

}
/*
 * -----
 * calculate Newton divided differences
 * -----
 */
void calcdiffs()
{
  int      c,j,k;
  for ( c=0; c<EXAMPLES; c++ )
    for ( k=1; k<=N; k++ )
      for ( j=N; j>=k; j-- )
        diffs[c][j] = (diffs[c][j]-diffs[c][j-1]) / (nodex[c][j]-nodex[c][j-k]);
}
/*
 * -----
 * fill arrays with node values and function values
 * -----
 */
void fillarrays()
{
  int      j,k;
  for ( k=0; k<=N; k++ )
    {
      nodex[0][k] = -1 + (double)k/4;
      nodex[1][k] = - cos( (double)(2*k+1)*PI/18 );
      nodex[2][k] = -1 + (double)k/8;
      for ( j=0; j<EXAMPLES; j++ )
        diffs[j][k] = exp( 2 * nodex[j][k] );
    }
  for ( k=0; k<NUMPOINTS; k++ )
    {
      u[k] = -1 + (double)k/12;
      f[k] = exp( 2 * u[k] );
    }
}
/*
 * -----
 * estimate error at each evaluation point u[k].
 * note we use 512 * 7.39 as an upper bound for 9th
 * derivative of exp(2x) on [-1, 1].
 * -----
 */
void esterror()
{
  int      c,j,k;

```

```

for ( c=0; c<EXAMPLES; c++ )
  for ( k=0; k<NUMPOINTS; k++ )
  {
    err[c][k] = (double)512 * 7.39 / 362880;
    for ( j=0; j<=N; j++ )
      err[c][k] *= (u[k] - nodex[c][j]);
  }
}

```

7 Assignment 4 Results

Here is the output produced by `interpol.c`

Mth 351 interpol - Lagrange interpolation polynomial example

CASE (A):

Point	Actual Val	Estimated Val	Actual Err	Theoretical Err
-1.0000	+0.1353352832	+0.1353352832	+0.0000000000	+0.0000000000
-0.9167	+0.1598797461	+0.1598559236	+0.0000238225	+0.0001946873
-0.8333	+0.1888756028	+0.1888633892	+0.0000122136	+0.0000983893
-0.7500	+0.2231301601	+0.2231301601	+0.0000000000	+0.0000000000
-0.6667	+0.2635971381	+0.2636014662	-0.0000043281	-0.0000338587
-0.5833	+0.3114032239	+0.3114061253	-0.0000029014	-0.0000223612
-0.5000	+0.3678794412	+0.3678794412	+0.0000000000	+0.0000000000
-0.4167	+0.4345982085	+0.4345966234	+0.0000015852	+0.0000118505
-0.3333	+0.5134171190	+0.5134158399	+0.0000012791	+0.0000094152
-0.2500	+0.6065306597	+0.6065306597	+0.0000000000	+0.0000000000
-0.1667	+0.7165313106	+0.7165322881	-0.0000009775	-0.0000069709
-0.0833	+0.8464817249	+0.8464826473	-0.0000009225	-0.0000064730
+0.0000	+1.0000000000	+1.0000000000	+0.0000000000	+0.0000000000
+0.0833	+1.1813604129	+1.1813594595	+0.0000009534	+0.0000064730
+0.1667	+1.3956124251	+1.3956113810	+0.0000010441	+0.0000069709
+0.2500	+1.6487212707	+1.6487212707	+0.0000000000	+0.0000000000
+0.3333	+1.9477340411	+1.9477355006	-0.0000014595	-0.0000094152
+0.4167	+2.3009758909	+2.3009777604	-0.0000018695	-0.0000118505
+0.5000	+2.7182818285	+2.7182818285	+0.0000000000	+0.0000000000
+0.5833	+3.2112705432	+3.2112668868	+0.0000036564	+0.0000223612
+0.6667	+3.7936678947	+3.7936622561	+0.0000056386	+0.0000338587
+0.7500	+4.4816890703	+4.4816890703	+0.0000000000	+0.0000000000
+0.8333	+5.2944900505	+5.2945070604	-0.0000170099	-0.0000983893
+0.9167	+6.2547009519	+6.2547352601	-0.0000343082	-0.0001946873

+1.0000 +7.3890560989 +7.3890560989 +0.0000000000 +0.0000000000

Worst error in CASE (A): 0.0000343082

CASE (B):

Point	Actual Val	Estimated Val	Actual Err	Theoretical Err
-1.0000	+0.1353352832	+0.1353402653	-0.0000049821	-0.0000407297
-0.9167	+0.1598797461	+0.1598754611	+0.0000042850	+0.0000345377
-0.8333	+0.1888756028	+0.1888783206	-0.0000027177	-0.0000215930
-0.7500	+0.2231301601	+0.2231352345	-0.0000050744	-0.0000397353
-0.6667	+0.2635971381	+0.2635986190	-0.0000014809	-0.0000114266
-0.5833	+0.3114032239	+0.3113998644	+0.0000033595	+0.0000255391
-0.5000	+0.3678794412	+0.3678740020	+0.0000054392	+0.0000407297
-0.4167	+0.4345982085	+0.4345945403	+0.0000036682	+0.0000270516
-0.3333	+0.5134171190	+0.5134175844	-0.0000004654	-0.0000033791
-0.2500	+0.6065306597	+0.6065350055	-0.0000043458	-0.0000310644
-0.1667	+0.7165313106	+0.7165370881	-0.0000057775	-0.0000406469
-0.0833	+0.8464817249	+0.8464857389	-0.0000040140	-0.0000277889
+0.0000	+1.0000000000	+1.0000000000	+0.0000000000	+0.0000000000
+0.0833	+1.1813604129	+1.1813562646	+0.0000041482	+0.0000277889
+0.1667	+1.3956124251	+1.3956062548	+0.0000061703	+0.0000406469
+0.2500	+1.6487212707	+1.6487164741	+0.0000047966	+0.0000310644
+0.3333	+1.9477340411	+1.9477335102	+0.0000005308	+0.0000033791
+0.4167	+2.3009758909	+2.3009802156	-0.0000043247	-0.0000270516
+0.5000	+2.7182818285	+2.7182884564	-0.0000066280	-0.0000407297
+0.5833	+3.2112705432	+3.2112747747	-0.0000042315	-0.0000255391
+0.6667	+3.7936678947	+3.7936659666	+0.0000019281	+0.0000114266
+0.7500	+4.4816890703	+4.4816822400	+0.0000068304	+0.0000397353
+0.8333	+5.2944900505	+5.2944862683	+0.0000037822	+0.0000215930
+0.9167	+6.2547009519	+6.2547071181	-0.0000061661	-0.0000345377
+1.0000	+7.3890560989	+7.3890486851	+0.0000074138	+0.0000407297

Worst error in CASE (B): 0.0000074138

CASE (C):

Point	Actual Val	Estimated Val	Actual Err	Theoretical Err
-1.0000	+0.1353352832	+0.1353352832	+0.0000000000	+0.0000000000
-0.9167	+0.1598797461	+0.1598797371	+0.0000000090	+0.0000001922
-0.8333	+0.1888756028	+0.1888756060	-0.0000000031	-0.0000000661
-0.7500	+0.2231301601	+0.2231301601	+0.0000000000	+0.0000000000
-0.6667	+0.2635971381	+0.2635971372	+0.0000000009	+0.0000000184
-0.5833	+0.3114032239	+0.3114032246	-0.0000000007	-0.0000000136

-0.5000	+0.3678794412	+0.3678794412	+0.0000000000	+0.0000000000
-0.4167	+0.4345982085	+0.4345982078	+0.0000000007	+0.0000000136
-0.3333	+0.5134171190	+0.5134171200	-0.0000000010	-0.0000000184
-0.2500	+0.6065306597	+0.6065306597	+0.0000000000	+0.0000000000
-0.1667	+0.7165313106	+0.7165313070	+0.0000000036	+0.0000000661
-0.0833	+0.8464817249	+0.8464817355	-0.0000000106	-0.0000001922
+0.0000	+1.0000000000	+1.0000000000	+0.0000000000	+0.0000000000
+0.0833	+1.1813604129	+1.1813598472	+0.0000005657	+0.0000098865
+0.1667	+1.3956124251	+1.3956085050	+0.0000039201	+0.0000672583
+0.2500	+1.6487212707	+1.6487045294	+0.0000167413	+0.0002819061
+0.3333	+1.9477340411	+1.9476785105	+0.0000555306	+0.0009174639
+0.4167	+2.3009758909	+2.3008196648	+0.0001562261	+0.0025317945
+0.5000	+2.7182818285	+2.7178915628	+0.0003902656	+0.0062019348
+0.5833	+3.2112705432	+3.2103804653	+0.0008900778	+0.0138662155
+0.6667	+3.7936678947	+3.7917799652	+0.0018879295	+0.0288235071
+0.7500	+4.4816890703	+4.4779158559	+0.0037732145	+0.0564376068
+0.8333	+5.2944900505	+5.2873153701	+0.0071746804	+0.1051035829
+0.9167	+6.2547009519	+6.2416251568	+0.0130757951	+0.1875424242
+1.0000	+7.3890560989	+7.3660825886	+0.0229735103	+0.3225006104

Worst error in CASE (C): 0.0229735103

8 Contact Information

The contact information below is accurate as of July 15, 2001.

Copyright © 2001 by Bent E. Petersen. Permission is granted to duplicate this document for non-profit educational purposes provided that no alterations are made and provided that this copyright notice is preserved on all copies.

Bent E. Petersen		phone numbers
Department of Mathematics		office (541) 737-5163
Oregon State University		home (541) 753-1829
Corvallis, OR 97331-4605		fax (541) 737-0517

bent@alum.mit.edu
petersen@math.orst.edu
<http://ucs.orst.edu/~peterseb>
<http://www.peak.org/~petersen>
<http://web.orst.edu/~peterseb>