

Numerical Analysis Program Instructions

About the Program Disk

This site contains programs to accompany Numerical Analysis, Eighth Edition by Burden and Faires in C, FORTRAN, Java, Maple, Mathematica, MATLAB, and Pascal. There is a program in each of these formats for each of the methods presented in the book.

Every program is illustrated with a sample problem or example that is closely correlated to an example in the text. This permits each program to be run initially in the language of your choice to see the form of the input and output. The programs can then be modified for other problems by making minor changes. The form of the input and output are, as near as possible, the same in each of the programming systems. This was done to permit an instructor using the programs to discuss the programs generically, without regard to the particular programming system an individual student is using.

Java Programs

These programs run directly from the web site provided you have a Java plug-in that is available at no cost from numerous sources. If you follow the web page link to the Java programs you will be directed to sites where these plug-ins are available.

The Other Programs

For the other programming and Computer Algebra Systems you need to have a copy of the programming system. The programs are designed to run on a mini-

mally configured DOS platform, but have also been presented in a text mode that will permit their use on no-PC systems. There are six subdirectories, one for each of the computer languages and one for the data files. The data files have been constructed so that will work with any of the systems. Each of the subdirectories contains a compressed (zip) file that permits you to download all the programs for that system in one file. You will need to decompress (unzip) this compressed file if you download the programs in this manner.

During the execution of some of the programs you will be asked questions of the form

```
Has the function F been created in the program
immediately preceding the INPUT procedure ?
```

To run the sample problems you should enter the response

```
Y (for Yes)
```

since the functions are embedded within the programs. The functions will need to be changed, however, if the programs are modified to solve other problems.

There is a slight exception in the case of the FORTRAN programs, since FORTRAN requires that any non-numeric input be enclosed in single back quotes. The response for FORTRAN program would consequently be

```
'Y' (for Yes)
```

Some of the programs require the input of large amounts of data or generate extensive output. To enable the programs to be run quickly and efficiently, the input data can be placed in data files and the data files read by the program. When the output is likely to be extensive, the programs have been constructed so that it is convenient to place the output directly into an output file. The program will prompt you for the form of the input and output you would like to use. For example, when running the program for Neville's method, ALGO31 . EXT, using the defined data file ALG031 . DTA for the sample problem, you will first see a screen that states:

```
Choice of input method:
1. Input entry by entry from the keyboard
2. Input data from a text file
3. Generate data using a function F
Choose 1, 2, or 3 please
```

If you choose 1 you will need to enter all the data for the program from the keyboard, and any mistake in a data entry will require the program to be rerun.

Choosing 3 causes the program to ask if the function has been defined. If you answer

'Y' (for Yes)

the program will assume that the function has been defined in the program and will use that function to produce the required data. If you answer

N (for No)

the program will assume that you want to change the function before continuing and will terminate execution so that the correct function can be entered into the program. Once this has been done the program can be rerun.

Since the Maple and Mathematica programs accept functional input, choosing 3 will cause a prompt for the input of the correct function.

There are a few special commands that are required for the Maple and Mathematica programs. Once a Maple program has been loaded, you should scroll to the top of the file and hit ENTER on the text portion of the line

```
> Restart;
```

This will re-initialize Maple and move the cursor to the end the first block below the

```
> Restart;
```

line. Hitting ENTER again will compile the program and send the cursor to the line on which the program name is listed. Hitting ENTER on the program name will run the program.

The situation for Mathematica is similar. In this case you should scroll to the top of the file and position the cursor within the first cell. Hitting the INSERT key or SHIFT-ENTER will compile and run the program.

Program Descriptions

Listed below by chapter are the descriptions of the individual programs. Since they are essentially language-independent, the program calls are listed with the extension .EXT. This should be replaced by the correct extension for the language

4

you are using. These are

Format	Extension
C	.C
FORTRAN	.FOR
Maple	.MWS
Mathematica	.MA
MATLAB	.M
Pascal	.PAS

Remember that if you are using FORTRAN, non-numeric input must always be enclosed in single left quotes.

Numerical Analysis Programs

Programs for Chapter 2

BISECTION METHOD

“ALG021.EXT”

This program uses the Bisection Method to approximate a root of the equation $f(x) = 0$ lying in the interval $[a, b]$. The sample problem uses

$$f(x) = x^3 + 4x^2 - 10.$$

INPUT: $a = 1, \quad b = 2, \quad TOL = 5 \times 10^{-4}, \quad N_0 = 20$

FIXED-POINT ITERATION

“ALG022.EXT”

This program uses Fixed-point Iteration to approximate a solution of $g(x) = x$ lying in the interval $[a, b]$. The sample problem uses

$$g(x) = \sqrt{\frac{10}{4+x}}.$$

INPUT: $p_0 = 1.5, \quad TOL = 5 \times 10^{-4}, \quad N_0 = 25$

NEWTON'S METHOD

“ALG023.EXT”

This program uses Newton's Method to approximate a root of the equation $f(x) = 0$. The sample problem uses

$$f(x) = \cos x - x \quad \text{with} \quad f'(x) = -\sin x - 1.$$

INPUT: $p_0 = \frac{\pi}{4}, \quad TOL = 5 \times 10^{-4}, \quad N_0 = 15$

SECANT METHOD**“ALG024.EXT”**

This program uses the Secant Method to approximate a root of the equation $f(x) = 0$. The sample problem uses

$$f(x) = \cos x - x.$$

INPUT: $p_0 = \frac{1}{2}$, $p_1 = \frac{\pi}{4}$, $TOL = 5 \times 10^{-4}$, $N_0 = 15$

METHOD OF FALSE POSITION**“ALG025.EXT”**

This program uses the Method of False Position to approximate a root of the equation $f(x) = 0$. The sample problem uses

$$f(x) = \cos x - x.$$

INPUT: $p_0 = \frac{1}{2}$, $p_1 = \frac{\pi}{4}$, $TOL = 5 \times 10^{-4}$, $N_0 = 15$

STEFFENSEN'S METHOD**“ALG026.EXT”**

This program uses Steffensen's Method to approximate a solution of $g(x) = x$. The sample problem uses

$$g(x) = \sqrt{\frac{10}{4+x}}.$$

INPUT: $p_0 = 1.5$, $TOL = 5 \times 10^{-4}$, $N_0 = 15$

HORNER'S METHOD**“ALG027.EXT”**

This program uses Horner's Method to evaluate an arbitrary polynomial of the form

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0.$$

The sample problem considers the polynomial

$$f(x) = 2x^4 - 3x^2 + 3x - 4.$$

INPUT: $n = 4$, $a_0 = -4$, $a_1 = 3$, $a_2 = -3$, $a_3 = 0$, $a_4 = 2$,
 $x_0 = -2$

MÜLLER'S METHOD**“ALG028.EXT”**

This program uses Müller's Method to approximate a root of an arbitrary polynomial of the form

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0.$$

The sample problem uses

$$f(x) = 16x^4 - 40x^3 + 5x^2 + 20x + 6.$$

INPUT: $n = 4$, $a_0 = 6$, $a_1 = 20$, $a_2 = 5$, $a_3 = -40$, $a_4 = 16$,

$TOL = 0.00001$, $N_0 = 30$, $x_0 = \frac{1}{2}$, $x_1 = -\frac{1}{2}$, $x_2 = 0$

Programs for Chapter 3**NEVILLE ITERATED INTERPOLATION****“ALG031.EXT”**

This program uses Neville's Iterated Interpolation Method to evaluate the n^{th} degree interpolating polynomial $P(x)$ on the $n + 1$ distinct numbers x_0, \dots, x_n at the number x for a given function f . The sample problem considers the Bessel function of the first kind of order zero at $x = 1.5$.

INPUT: ALG031.DTA, $n = 4$, $x = 1.5$

**NEWTON INTERPOLATORY
DIVIDED-DIFFERENCE FORMULA****“ALG032.EXT”**

This program uses Newton's Interpolatory Divided-Difference Formula to evaluate the divided-difference coefficients of the n^{th} degree interpolatory polynomial $P(x)$ on the $n + 1$ distinct numbers x_0, \dots, x_n for a given function f . The sample problem considers the Bessel function of the first kind of order zero.

INPUT: ALG032.DTA, $n = 4$

HERMITE INTERPOLATION**“ALG033.EXT”**

This program uses Hermite’s Interpolation Method to obtain the coefficients of the Hermite interpolating polynomial $H(x)$ on the $n + 1$ distinct numbers x_0, \dots, x_n for a given function f . The sample problem considers the Bessel function of the first kind of order zero.

INPUT: ALG033.DTA, $n = 2$

NATURAL CUBIC SPLINE INTERPOLATION**“ALG034.EXT”**

This program uses the Natural Cubic Spline Method to construct the free cubic spline interpolant S for a function f . The sample problem considers $f(x) = e^{2x}$ on the interval $[0, 1]$.

INPUT: (Select input option 2.) ALG034.DTA, $n = 4$

CLAMPED CUBIC SPLINE INTERPOLATION**“ALG035.EXT”**

This program uses the Clamped Cubic Spline Method to construct the clamped cubic spline interpolant s for the function f . The sample problem considers $f(x) = e^{2x}$ on the interval $[0, 1]$.

INPUT: (Select input option 2.) ALG035.DTA, $n = 4$,
 $FPO = 2$, $FPN = 2e^2$

BÉZIER CURVE ALGORITHM 3.6**“ALG036.EXT”**

This program uses the Bézier Curve method to construct parametric curves to approximate given data. The sample program considers

$$\begin{aligned} (x_0, y_0) &= (0, 0); & (x_0^+, y_0^+) &= (1/4, 1/4) \\ (x_1, y_1) &= (1, 1); & (x_1^-, y_1^-) &= (1/2, 1/2); & (x_1^+, y_1^+) &= (-1/2, -1/2) \\ (x_2, y_2) &= (2, 2); & (x_2^-, y_2^-) &= (-1, -1) \end{aligned}$$

INPUT: ALG036.DTA, $n = 2$

Programs for Chapter 4

COMPOSITE SIMPSON'S RULE

“ALG041.EXT”

This program uses Composite Simpson's Rule to approximate

$$\int_a^b f(x) dx.$$

The sample problem uses

$$f(x) = \sin x, \quad \text{on } [0, \pi].$$

INPUT: $a = 0, \quad b = \pi, \quad n = 10$

ROMBERG INTEGRATION

“ALG042.EXT”

This program uses the Romberg Method to approximate

$$\int_a^b f(x) dx.$$

The sample problem uses

$$f(x) = \sin x, \quad \text{on } [0, \pi].$$

INPUT: $a = 0, \quad b = \pi, \quad n = 6$

ADAPTIVE QUADRATURE

“ALG043.EXT”

This program uses the Adaptive Quadrature Method to approximate

$$\int_a^b f(x) dx$$

within a given tolerance $TOL > 0$. The sample problem uses

$$f(x) = \frac{100}{x^2} \sin \frac{10}{x}, \quad \text{on } [1, 3].$$

INPUT: $a = 1, \quad b = 3, \quad TOL = 0.0001, \quad N = 20$

**COMPOSITE SIMPSON'S RULE
FOR DOUBLE INTEGRALS**

“ALG044.EXT”

This program uses the Composite Simpson's Rule for Double Integrals to approximate

$$\int_a^b \int_{c(x)}^{d(x)} f(x, y) dy dx.$$

The sample problem uses

$$f(x, y) = e^{\frac{y}{x}}$$

with

$$c(x) = x^3, \quad d(x) = x^2, \quad a = 0.1 \quad \text{and} \quad b = 0.5.$$

INPUT: $a = 0.1, \quad b = 0.5, \quad m = 10, \quad n = 10$

**GAUSSIAN QUADRATURE
FOR DOUBLE INTEGRALS**

“ALG045.EXT”

This program uses Gaussian Quadrature to approximate

$$\int_a^b \int_{c(x)}^{d(x)} f(x, y) dy dx.$$

The sample problem uses $f(x, y) = e^{y/x}$ with

$$c(x) = x^3, \quad d(x) = x^2, \quad a = 0.1 \quad \text{and} \quad b = 0.5.$$

INPUT: $a = 0.1, \quad b = 0.5, \quad m = 5, \quad n = 5$

**GAUSSIAN QUADRATURE
FOR TRIPLE INTEGRALS**

“ALG046.EXT”

This program uses the Gaussian Quadrature to approximate

$$\int_a^b \int_{c(x)}^{d(x)} \int_{\alpha(x,y)}^{\beta(x,y)} f(x, y, z) dz dy dx.$$

The sample problem uses

$$f(x, y, z) = \sqrt{x^2 + y^2}$$

with

$$\alpha(x, y) = \sqrt{x^2 + y^2}, \quad \beta(x, y) = 2,$$

$$c(x) = 0.0, \quad d(x) = \sqrt{4 - x^2}, \quad a = 0, \quad \text{and} \quad b = 2.$$

INPUT: $a = 0, \quad b = 2, \quad m = 5, \quad n = 5, \quad p = 5$

Programs for Chapter 5

EULER METHOD

“ALG051.EXT”

This program uses the Euler Method to approximate the solution of an initial value problem of the form

$$y' = f(t, y), \quad y(a) = \alpha, \quad a \leq t \leq b.$$

The sample problem uses

$$f(t, y) = y - t^2 + 1, \quad y(0) = 0.5, \quad 0 \leq t \leq 2.$$

INPUT: $a = 0, \quad b = 2, \quad \alpha = 0.5, \quad N = 10$

RUNGE-KUTTA METHOD OF ORDER FOUR

“ALG052.EXT”

This program uses the Runge-Kutta Method of order four to approximate the solution of the initial value problem of the form

$$y' = f(t, y), \quad y(a) = \alpha, \quad a \leq t \leq b.$$

The sample problem uses

$$f(t, y) = y - t^2 + 1, \quad y(0) = 0.5, \quad 0 \leq t \leq 2.$$

INPUT: $a = 0, \quad b = 2, \quad \alpha = 0.5, \quad N = 10$

RUNGE-KUTTA-FEHLBERG METHOD**“ALG053.EXT”**

This program uses the Runge-Kutta-Fehlberg Method to approximate the solution of the initial value problem of the form

$$y' = f(t, y), \quad y(a) = \alpha, \quad a \leq t \leq b$$

to within a given tolerance. The sample problem uses

$$f(t, y) = y - t^2 + 1, \quad y(0) = 0.5, \quad 0 \leq t \leq 2.$$

INPUT: $a = 0, \quad b = 2, \quad \alpha = 0.5, \quad TOL = 0.00001, \quad HMIN = 0.01,$
 $HMAX = 0.25$

**ADAMS FOURTH-ORDER
PREDICTOR-CORRECTOR METHOD****“ALG054.EXT”**

This program uses the Adams Fourth-Order Predictor-Corrector Method to approximate the solution of an initial value problem of the form

$$y' = f(t, y), \quad y(a) = \alpha, \quad a \leq t \leq b.$$

The sample problem uses

$$f(t, y) = y - t^2 + 1, \quad y(0) = 0.5, \quad 0 \leq t \leq 2.$$

INPUT: $a = 0, \quad b = 2, \quad \alpha = 0.5, \quad N = 10$

**ADAMS VARIABLE-STEP SIZE
PREDICTOR-CORRECTOR METHOD****“ALG055.EXT”**

This program uses the Adams Variable Step Size Predictor-Corrector Method to approximate the solution of an initial value problem of the form

$$y' = f(t, y), \quad y(a) = \alpha, \quad a \leq t \leq b$$

to within a given tolerance. The sample problem uses

$$f(t, y) = y - t^2 + 1, \quad y(0) = 0.5, \quad 0 \leq t \leq 2.$$

INPUT: $a = 0, \quad b = 2, \quad \alpha = 0.5, \quad TOL = 0.00001, \quad HMIN = 0.01,$
 $HMAX = 0.25$

EXTRAPOLATION METHOD**“ALG056.EXT”**

This program uses the Extrapolation Method to approximate the solution of an initial value problem of the form

$$y' = f(t, y), \quad y(a) = \alpha, \quad a \leq t \leq b$$

to within a given tolerance. The sample problem uses

$$f(t, y) = y - t^2 + 1, \quad y(0) = 0.5, \quad 0 \leq t \leq 2.$$

INPUT: $a = 0, \quad b = 2, \quad \alpha = 0.5, \quad TOL = 0.00001, \quad HMIN = 0.01,$
 $HMAX = 0.25$

RUNGE-KUTTA METHOD FOR SYSTEMS OF DIFFERENTIAL EQUATIONS**“ALG057.EXT”**

This program uses the Runge-Kutta for Systems of Differential Equations Method to approximate a the solution of the m th-order system of first-order initial value problems. The sample problem considers the second order system

$$\begin{aligned} f_1(u_1, u_2) &= -4u_1 + 3u_2 + 6, & u_1(0) &= 0, \\ f_2(u_1, u_2) &= -2.4u_1 + 1.6u_2 + 3.6, & u_2(0) &= 0. \end{aligned}$$

INPUT: $a = 0, \quad b = 0.5, \quad \alpha_1 = 0, \quad \alpha_2 = 0, \quad N = 5$

TRAPEZOIDAL METHOD WITH NEWTON ITERATION**“ALG058.EXT”**

This program uses the Trapezoidal Method with Newton Iteration to approximate the solution to the initial value problem

$$y' = f(t, y), \quad y(a) = \alpha, \quad a \leq t \leq b.$$

The sample problem uses

$$\begin{aligned} f(t, y) &= 5e^{5t}(y - t)^2 + 1, & y(0) &= -1, & 0 \leq t \leq 1, \\ f_y(t, y) &= 10e^{5t}(y - t). \end{aligned}$$

INPUT: $a = 0, \quad b = 1, \quad \alpha = -1, \quad N = 5, \quad TOL = 0.000001,$
 $M = 10$

Programs for Chapter 6

GAUSSIAN ELIMINATION WITH BACKWARD SUBSTITUTION

“ALG061.EXT”

This program uses the Gaussian Elimination with Backward Substitution Method to solve an $n \times n$ linear system of the form $Ax = b$. The sample problem solves the linear system

$$\begin{aligned}x_1 - x_2 + 2x_3 - x_4 &= -8 \\2x_1 - 2x_2 + 3x_3 - 3x_4 &= -20 \\x_1 + x_2 + x_3 &= -2 \\x_1 - x_2 + 4x_3 + 3x_4 &= 4.\end{aligned}$$

INPUT: ALG061.DTA, $n = 4$

GAUSSIAN ELIMINATION WITH PARTIAL PIVOTING

“ALG062.EXT”

This program uses the Gaussian Elimination with Partial Pivoting Method to solve an $n \times n$ linear system. The sample problem solves the linear system

$$\begin{aligned}x_1 - x_2 + 2x_3 - x_4 &= -8 \\2x_1 - 2x_2 + 3x_3 - 3x_4 &= -20 \\x_1 + x_2 + x_3 &= -2 \\x_1 - x_2 + 4x_3 + 3x_4 &= 4.\end{aligned}$$

INPUT: ALG062.DTA, $n = 4$

GAUSSIAN ELIMINATION WITH SCALED PARTIAL PIVOTING

“ALG063.EXT”

This program uses the Gaussian Elimination with Scaled Partial Pivoting Method to solve an $n \times n$ linear system. The sample problem solves the linear system

$$\begin{aligned}x_1 - x_2 + 2x_3 - x_4 &= -8 \\2x_1 - 2x_2 + 3x_3 - 3x_4 &= -20 \\x_1 + x_2 + x_3 &= -2 \\x_1 - x_2 + 4x_3 + 3x_4 &= 4.\end{aligned}$$

INPUT: ALG063.DTA, $n = 4$

LU FACTORIZATION**“ALG064.EXT”**

This program uses the *LU* Factorization Method to factor the $n \times n$ matrix A into the product $A = LU$ of a lower triangular matrix L and an upper triangular matrix U . The matrix factored in the sample problem is

$$A = \begin{bmatrix} 6 & 2 & 1 & -1 \\ 2 & 4 & 1 & 0 \\ 1 & 1 & 4 & -1 \\ -1 & 0 & -1 & 3 \end{bmatrix}$$

INPUT: ALG064.DTA, $n = 4$, $ISW = 1$

LDL*^t FACTORIZATION*“ALG065.EXT”**

This program uses the *LDL*^t Factorization Method to factor the positive definite $n \times n$ matrix A into the product LDL^t , where L is a lower triangular matrix and D is a diagonal matrix. The matrix factored in the sample problem is

$$A = \begin{bmatrix} 4 & -1 & 1 \\ -1 & 4.25 & 2.75 \\ 1 & 2.75 & 3.5 \end{bmatrix}$$

INPUT: ALG065.DTA, $n = 3$

CHOLESKI'S METHOD**“ALG066.EXT”**

This program uses the Choleski Method to factor the positive definite $n \times n$ matrix A into the product LL^t , where L is a lower triangular matrix. The matrix factored in the sample problem is

$$A = \begin{bmatrix} 4 & -1 & 1 \\ -1 & 4.25 & 2.75 \\ 1 & 2.75 & 3.5 \end{bmatrix}$$

INPUT: ALG066.DTA, $n = 3$

**CROUT REDUCTION FOR
TRIDIAGONAL LINEAR SYSTEMS**

“ALG067.EXT”

This program uses the Crout Reduction for Tridiagonal Linear Systems Method to solve a tridiagonal $n \times n$ linear system. The sample system is

$$\begin{aligned} 2x_1 - x_2 &= 1 \\ -x_1 + 2x_2 - x_3 &= 0 \\ -x_2 + 2x_3 - x_4 &= 0 \\ -x_3 + 2x_4 &= 1. \end{aligned}$$

INPUT: ALG067.DTA, $n = 4$

Programs for Chapter 7

JACOBI ITERATIVE METHOD

“ALG071.EXT”

This program uses the Jacobi Iterative Method to approximate the solution to the $n \times n$ linear system $A\mathbf{x} = \mathbf{b}$, given an initial approximation $\mathbf{x}_0 = (x_1^{(0)}, x_2^{(0)}, \dots, x_n^{(0)})^t$. The sample problem approximates the solution to the linear system

$$\begin{aligned} 10x_1 - x_2 + 2x_3 &= 6 \\ -x_1 + 11x_2 - x_3 + 3x_4 &= 25 \\ 2x_1 - x_2 + 10x_3 - x_4 &= -11 \\ 3x_2 - x_3 + 8x_4 &= 15. \end{aligned}$$

starting with the initial vector $\mathbf{x}_0 = (0, 0, 0, 0)^t$.

INPUT: ALG071.DTA, $n = 4$, $TOL = 0.001$, $N = 30$

GAUSS–SEIDEL ITERATIVE METHOD**“ALG072.EXT”**

This program uses the Gauss-Seidel Iterative Method to approximate the solution to the $n \times n$ linear system $A\mathbf{x} = \mathbf{b}$, given an initial approximation $\mathbf{x}_0 = (x_1^{(0)}, x_2^{(0)}, \dots, x_n^{(0)})^t$. The sample problem approximates the solution to the linear system

$$\begin{aligned} 10x_1 - x_2 + 2x_3 &= 6 \\ -x_1 + 11x_2 - x_3 + 3x_4 &= 25 \\ 2x_1 - x_2 + 10x_3 - x_4 &= -11 \\ 3x_2 - x_3 + 8x_4 &= 15 \end{aligned}$$

starting with the initial vector $\mathbf{x}_0 = (0, 0, 0, 0)^t$.

INPUT: ALG072.DTA, $n = 4$, $TOL = 0.001$, $N = 30$

**SUCCESSIVE–OVER–RELAXATION
(SOR) METHOD****“ALG073.EXT”**

This program uses the Successive-Over-Relaxation Method to approximate the solution to the $n \times n$ linear system $A\mathbf{x} = \mathbf{b}$, given a parameter ω and an initial approximation $\mathbf{x}_0 = (x_1^{(0)}, x_2^{(0)}, \dots, x_n^{(0)})^t$. The sample problem approximates the solution to the linear system

$$\begin{aligned} 4x_1 + 3x_2 &= 24 \\ 3x_1 + 4x_2 - x_3 &= 30 \\ -x_2 + 4x_3 &= -24 \end{aligned}$$

starting with the initial vector $\mathbf{x}_0 = (1, 1, 1)^t$.

INPUT: ALG073.DTA, $n = 3$, $TOL = 0.001$, $N = 30$, $\omega = 1.25$

ITERATIVE REFINEMENT**“ALG074.EXT”**

This program uses the Iterative Refinement Method to approximate a solution to the linear system $A\mathbf{x} = \mathbf{b}$ when A is suspected to be ill-conditioned. The sample problem considers the linear system

$$\begin{aligned} 3.333x_1 + 15920x_2 - 10.333x_3 &= 15913 \\ 2.222x_1 + 16.710x_2 + 9.6120x_3 &= 28.544 \\ 1.5611x_1 + 5.1791x_2 + 1.6852x_3 &= 8.4254. \end{aligned}$$

INPUT: ALG074.DTA, $n = 3$, $N = 25$, $D = 5$, $TOL = 0.00001$

**PRECONDITIONED CONJUGATE
GRADIENT METHOD****“ALG075.EXT”**

This program uses the Preconditioned Conjugate Gradient Method to approximate the solution to the $n \times n$ linear system $A\mathbf{x} = \mathbf{b}$, given a preconditioning matrix C^{-1} and an initial approximation $\mathbf{x}_0 = (x_1^{(0)}, x_2^{(0)}, \dots, x_n^{(0)})^t$. The sample problem approximates the solution to the linear system

$$\begin{aligned} 4x_1 + 3x_2 &= 24 \\ 3x_1 + 4x_2 - x_3 &= 30 \\ -x_2 + 4x_3 &= -24 \end{aligned}$$

starting with the initial vector $\mathbf{x}_0 = (0, 0, 0)^t$.

INPUT: ALG075.DTA, $n = 3$, $TOL = 0.001$, $N = 3$

Programs for Chapter 8

PADÉ APPROXIMATION

“ALG081.EXT”

This program uses Padé Approximation to compute the rational approximation

$$r(x) = \frac{p_0 + p_1x + \cdots + p_nx^n}{q_0 + q_1x + \cdots + q_mx^m}$$

to a function $f(x)$ given its Maclaurin series

$$a_0 + a_1x + a_2x^2 + \cdots .$$

The sample problem uses $f(x) = e^{-x}$, where

$$a_0 = 1, \quad a_1 = -1, \quad a_2 = \frac{1}{2}, \quad a_3 = -\frac{1}{6}, \quad a_4 = \frac{1}{24}, \quad a_5 = -\frac{1}{120}.$$

INPUT: ALG081.DTA, $m = 2$, $n = 3$

CHEBYSHEV RATIONAL APPROXIMATION

“ALG082.EXT”

This program uses Chebyshev Rational Approximation to compute the rational approximation

$$r_T(x) = \frac{p_0T_0(x) + p_1T_1(x) + \cdots + p_nT_n(x)}{q_0T_0(x) + q_1T_1(x) + \cdots + q_mT_m(x)}$$

to a function $f(x)$ given its Chebyshev expansion

$$a_0T_0(x) + a_1T_1(x) + a_2T_2(x) + \cdots .$$

The sample problem uses $f(x) = e^{-x}$, where

$$\begin{aligned} a_0 &= 1.266066, & a_1 &= -1.130318, & a_2 &= 0.271495, \\ a_3 &= -0.044337, & a_4 &= 0.005474, & a_5 &= -0.000543. \end{aligned}$$

INPUT: ALG082.DTA, $m = 2$, $n = 3$

FAST FOURIER TRANSFORM METHOD**“ALG083.EXT”**

This program uses the Fast Fourier Transform Method to compute the coefficients in the discrete trigonometric approximation for a given set of data. The sample problem constructs an approximation to the function

$$f(x) = x^4 - 3x^3 + 2x^2 - \tan x(x - 2)$$

on the interval $[0,2]$.

INPUT: (Select input option 3.) $m = 4$

Programs for Chapter 9**POWER METHOD****“ALG091.EXT”**

This program uses the Power Method to approximate the dominant eigenvalue and an associated eigenvector of an $n \times n$ matrix A given a nonzero vector \mathbf{x} . The sample problem considers the matrix

$$A = \begin{bmatrix} -4 & 14 & 0 \\ -5 & 13 & 0 \\ -1 & 0 & 2 \end{bmatrix}$$

with $\mathbf{x} = (1, 1, 1)^t$ as the initial approximation to the eigenvector.

INPUT: ALG091.DTA, $n = 3$, $TOL = 0.0001$, $N = 30$

SYMMETRIC POWER METHOD**“ALG092.EXT”**

This program uses the Symmetric Power Method to approximate the dominant eigenvalue and an associated eigenvector of a symmetric $n \times n$ matrix A given a nonzero vector \mathbf{x} . The sample problem considers the symmetric matrix

$$A = \begin{bmatrix} 4 & -1 & 1 \\ -1 & 3 & -2 \\ 1 & -2 & 3 \end{bmatrix}$$

with $\mathbf{x} = (1, 0, 0)^t$ as the initial approximation to the eigenvector.

INPUT: ALG092.DTA, $n = 3$, $TOL = 0.0001$, $N = 25$

INVERSE POWER METHOD

“ALG093.EXT”

This program uses the Inverse Power Method to approximate an eigenvalue nearest to a given number q and an associated eigenvector of an $n \times n$ matrix A . The sample problem considers the matrix

$$A = \begin{bmatrix} -4 & 14 & 0 \\ -5 & 13 & 0 \\ -1 & 0 & 2 \end{bmatrix}$$

with $\mathbf{x} = (1, 1, 1)^t$ as the initial approximation to the eigenvector and the number q defined by

$$q = \frac{\mathbf{x}_t A \mathbf{x}}{\mathbf{x}_t \mathbf{x}}.$$

INPUT: ALG093.DTA, $n = 3$, $TOL = 0.0001$, $N = 25$

WIELANDT DEFLATION

“ALG094.EXT”

This program uses the Wielandt Deflation Method to approximate the second most dominant eigenvalue and an associated eigenvector of the $n \times n$ matrix A given a nonzero vector \mathbf{x}_0 . The sample problem considers the matrix

$$A = \begin{bmatrix} 4 & -1 & 1 \\ -1 & 3 & -2 \\ 1 & -2 & 3 \end{bmatrix}$$

which has the dominant eigenvalue $\lambda = 6$ and associated eigenvector $\mathbf{v} = (1, -1, 1)^t$. The initial approximation $\mathbf{x}_0 = (0, 1)^t$.

INPUT: ALG094.DTA, $n = 3$, $TOL = 0.0001$, $N = 30$

HOUSEHOLDER'S METHOD**“ALG095.EXT”**

This program uses the Householder Method to obtain a symmetric tridiagonal matrix that is similar to a given symmetric matrix A . The sample problem considers the matrix

$$A = \begin{bmatrix} 4 & 1 & -2 & 2 \\ 1 & 2 & 0 & 1 \\ -2 & 0 & 3 & -2 \\ 2 & 1 & -2 & -1 \end{bmatrix}$$

INPUT: ALG095.DTA, $n = 4$

QR METHOD**“ALG096.EXT”**

This program uses the QR Method to obtain the eigenvalues of a symmetric, tridiagonal $n \times n$ matrix of the form

$$A = \begin{bmatrix} a_1^{(1)} & b_2^{(1)} & 0 & \dots & \dots & \dots & \dots & 0 \\ b_2^{(1)} & a_2^{(1)} & b_3^{(1)} & \ddots & & & & \vdots \\ 0 & b_3^{(1)} & a_3^{(1)} & b_4^{(1)} & \ddots & & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & & & \ddots & b_{n-1}^{(1)} & a_{n-1}^{(1)} & b_n^{(1)} \\ 0 & \dots & \dots & \dots & \dots & 0 & b_n^{(1)} & a_n^{(1)} \end{bmatrix}$$

The sample problem considers the matrix

$$A = \begin{bmatrix} a_1^{(1)} & b_2^{(1)} & 0 \\ b_2^{(1)} & a_2^{(1)} & b_3^{(1)} \\ 0 & b_3^{(1)} & a_3^{(1)} \end{bmatrix} = \begin{bmatrix} 3 & 1 & 0 \\ 1 & 3 & 1 \\ 0 & 1 & 3 \end{bmatrix}$$

INPUT: ALG096.DTA, $n = 3$, $TOL = 0.00001$, $M = 30$

Programs for Chapter 10

NEWTON'S METHOD FOR SYSTEMS

“ALG101.EXT”

This program uses Newton's Method for Systems to approximate the solution of the nonlinear system of equations $\mathbf{F}(\mathbf{x}) = \mathbf{0}$ given an initial approximation \mathbf{x}_0 . The sample problem uses

$$\mathbf{F}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), f_3(\mathbf{x}))^t, \quad \text{where } \mathbf{x} = (x_1, x_2, x_3)^t$$

and

$$\begin{aligned} f_1(x_1, x_2, x_3) &= 3x_1 - \cos(x_2x_3) - 0.5 \\ f_2(x_1, x_2, x_3) &= x_1^2 - 81(x_2 + 0.1)^2 + \sin x_3 + 1.06 \\ f_3(x_1, x_2, x_3) &= e^{-x_1x_2} + 20x_3 + \frac{10\pi - 3}{3}. \end{aligned}$$

INPUT: $n = 3$, $TOL = 0.00001$, $N = 25$, $\mathbf{x}_0 = (0.1, 0.1, -0.1)^t$

BROYDEN'S METHOD

“ALG102.EXT”

This program uses the Broyden Method to approximate the solution of the nonlinear system of equations $\mathbf{F}(\mathbf{x}) = \mathbf{0}$ given an initial approximation \mathbf{x}_0 . The sample problem uses

$$\mathbf{F}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), f_3(\mathbf{x}))^t, \quad \text{where } \mathbf{x} = (x_1, x_2, x_3)^t$$

and

$$\begin{aligned} f_1(x_1, x_2, x_3) &= 3x_1 - \cos(x_2x_3) - 0.5 \\ f_2(x_1, x_2, x_3) &= x_1^2 - 81(x_2 + 0.1)^2 + \sin x_3 + 1.06 \\ f_3(x_1, x_2, x_3) &= e^{-x_1x_2} + 20x_3 + \frac{10\pi - 3}{3}. \end{aligned}$$

INPUT: $n = 3$, $TOL = 0.00001$, $N = 25$, $\mathbf{x}_0 = (0.1, 0.1, -0.1)^t$

STEEPEST DESCENT METHOD**“ALG103.EXT”**

This program uses the Steepest Descent Method to approximate a solution to the minimum of the function

$$g(\mathbf{x}) = \sum_{i=1}^n [f_i(\mathbf{x})]^2$$

given an initial approximation \mathbf{x}_0 . This also approximates a zero of

$$\mathbf{F}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_n(\mathbf{x}))^t.$$

The sample problem uses

$$\mathbf{F}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), f_3(\mathbf{x}))^t, \quad \text{where } \mathbf{x} = (x_1, x_2, x_3)^t$$

and

$$\begin{aligned} f_1(x_1, x_2, x_3) &= 3x_1 - \cos(x_2x_3) - 0.5 \\ f_2(x_1, x_2, x_3) &= x_1^2 - 81(x_2 + 0.1)^2 + \sin x_3 + 1.06 \\ f_3(x_1, x_2, x_3) &= e^{-x_1x_2} + 20x_3 + \frac{10\pi - 3}{3}. \end{aligned}$$

INPUT: $n = 3, \quad TOL = 0.05, \quad N = 10, \quad \mathbf{x}_0 = (0, 0, 0)^t$

CONTINUATION METHOD**“ALG104.EXT”**

This program uses the Continuation Method with the Runge-Kutta method of Order Four to approximate the solution of the nonlinear system of equations $\mathbf{F}(\mathbf{x}) = \mathbf{0}$ given an initial approximation \mathbf{x}_0 . The sample problem uses

$$\mathbf{F}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), f_3(\mathbf{x}))^t, \quad \text{where } \mathbf{x} = (x_1, x_2, x_3)^t$$

and

$$\begin{aligned} f_1(x_1, x_2, x_3) &= 3x_1 - \cos(x_2x_3) - 0.5 \\ f_2(x_1, x_2, x_3) &= x_1^2 - 81(x_2 + 0.1)^2 + \sin x_3 + 1.06 \\ f_3(x_1, x_2, x_3) &= e^{-x_1x_2} + 20x_3 + \frac{10\pi - 3}{3}. \end{aligned}$$

INPUT: $n = 3, \quad TOL = 0.00001, \quad N = 1, \quad \mathbf{x}_0 = (0, 0, 0)^t$

Programs for Chapter 11

LINEAR SHOOTING METHOD

“ALG111.EXT”

This program uses the Linear Shooting Method to approximate the solution of a linear two-point boundary-value problem

$$y'' = p(x)y' + q(x)y + r(x), \quad a \leq x \leq b, \quad y(a) = \alpha, \quad y(b) = \beta.$$

The sample problem considers the boundary-value problem

$$y'' = -\frac{2}{x}y' + \frac{2}{x^2}y + \frac{\sin(\ln x)}{x^2}, \quad y(1) = 1, \quad y(2) = 2.$$

INPUT: $a = 1, \quad b = 2, \quad \alpha = 1, \quad \beta = 2, \quad N = 10$

NONLINEAR SHOOTING METHOD

“ALG112.EXT”

This program uses the Nonlinear Shooting Method to approximate the solution of a nonlinear two-point boundary-value problem

$$y'' = f(x, y, y'), \quad a \leq x \leq b, \quad y(a) = \alpha, \quad y(b) = \beta.$$

The sample problem considers the boundary-value problem

$$y'' = 4 + 0.25x^3 - 0.125yy', \quad y(1) = 17, \quad y(3) = \frac{43}{3},$$

where

$$f_y(x, y, y') = -\frac{y'}{8} \quad \text{and} \quad f_{y'}(x, y, y') = -\frac{y}{8}.$$

INPUT: $a = 1, \quad b = 3, \quad \alpha = 17, \quad \beta = \frac{43}{3}, \quad N = 20, \quad TOL = 0.0001,$
 $M = 25$

LINEAR FINITE-DIFFERENCE METHOD**“ALG113.EXT”**

This program uses the Linear Finite-Difference Method to approximate the solution of a linear two-point boundary-value problem

$$y'' = p(x)y' + q(x)y + r(x), \quad a \leq x \leq b, \quad y(a) = \alpha, \quad y(b) = \beta.$$

The sample problem considers the boundary-value problem

$$y'' = -\frac{2}{x}y' + \frac{2}{x^2}y + \frac{\sin(\ln x)}{x^2}, \quad y(1) = 1, \quad y(2) = 2.$$

INPUT: $a = 1, \quad b = 2, \quad \alpha = 1, \quad \beta = 2, \quad N = 9$

NONLINEAR FINITE-DIFFERENCE METHOD**“ALG114.EXT”**

This program uses the Nonlinear Finite-Difference Method to approximate the solution to a nonlinear two-point boundary-value problem

$$y'' = f(x, y, y'), \quad a \leq x \leq b, \quad y(a) = \alpha, \quad y(b) = \beta.$$

The sample problem considers the boundary-value problem

$$y'' = 4 + 0.25x^3 - 0.125yy', \quad y(1) = 17, \quad y(3) = \frac{43}{3},$$

where

$$f_y(x, y, y') = -\frac{y'}{8} \quad \text{and} \quad f_{y'}(x, y, y') = -\frac{y}{8}.$$

INPUT: $a = 1, \quad b = 3, \quad \alpha = 17, \quad \beta = \frac{43}{3}, \quad N = 19, \quad TOL = 0.0001, \quad M = 25$

**PIECEWISE LINEAR
RAYLEIGH-RITZ METHOD**

“ALG115.EXT”

This program uses the Piecewise Linear Rayleigh-Ritz Method to approximate the solution to a two-point boundary-value problem

$$-\frac{d}{dx} \left(p(x) \frac{dy}{dx} \right) + q(x)y = f(x), \quad 0 \leq x \leq 1, \quad y(0) = y(1) = 0.$$

The sample problem uses the differential equation

$$-y'' + \pi^2 y = 2\pi^2 \sin \pi x, \quad y(0) = 0, \quad y(1) = 0.$$

INPUT: $n = 9$, ALG115.DTA

**CUBIC SPLINE
RAYLEIGH-RITZ METHOD**

“ALG116.EXT”

This program uses the Cubic Spline Rayleigh-Ritz Method to approximate the solution of a boundary-value problem

$$-\frac{d}{dx} \left(p(x) \frac{dy}{dx} \right) + q(x)y = f(x), \quad 0 \leq x \leq 1, \quad y(0) = y(1) = 0.$$

The sample problem uses the differential equation

$$-y'' + \pi^2 y = 2\pi^2 \sin \pi x, \quad y(0) = 0, \quad y(1) = 0.$$

INPUT: $n = 9$, $f'(0) = 2\pi^3$, $f'(1) = -2\pi^3$, $p'(0) = 0$, $p'(1) = 0$,
 $q'(0) = 0$, $q'(1) = 0$

Programs for Chapter 12

POISSON EQUATION FINITE-DIFFERENCE METHOD

“ALG121.EXT”

This program uses the Poisson Equation Finite-Difference Method to approximate the solution to the Poisson equation

$$\frac{\partial^2 u}{\partial x^2}(x, y) + \frac{\partial^2 u}{\partial y^2}(x, y) = f(x, y)$$

subject to boundary conditions $u(x, y) = g(x, y)$. The sample problem uses

$$f(x, y) = xe^y \quad \text{and} \quad g(x, y) = xe^y.$$

INPUT: $a = 0, \quad b = 2, \quad c = 0, \quad d = 1, \quad n = 6, \quad m = 5,$
 $TOL = 10^{-5}, \quad M = 150$

HEAT EQUATION BACKWARD-DIFFERENCE METHOD

“ALG122.EXT”

This program uses the Heat Equation Backward-Difference Method to approximate the solution to a parabolic partial-differential equation

$$\frac{\partial u}{\partial t}(x, t) = \alpha^2 \frac{\partial^2 u}{\partial x^2}(x, t), \quad 0 < x < l, \quad 0 < t$$

subject to the boundary conditions

$$u(0, t) = 0 \quad \text{and} \quad u(l, t) = 0$$

and the initial condition

$$u(x, 0) = f(x).$$

The sample problem uses

$$f(x) = \sin \pi x.$$

INPUT: $l = 1, \quad T = 0.5, \quad \alpha = 1, \quad m = 10, \quad N = 50$

CRANK-NICOLSON METHOD**“ALG123.EXT”**

This program uses the Crank-Nicolson Method to approximate the solution to a parabolic partial-differential equation

$$\frac{\partial u}{\partial t}(x, t) = \alpha^2 \frac{\partial^2 u}{\partial x^2}(x, t), \quad 0 < x < l, \quad 0 < t$$

subject to the boundary conditions

$$u(0, t) = 0 \quad \text{and} \quad u(l, t) = 0$$

and the initial condition

$$u(x, 0) = f(x).$$

The sample problem uses

$$f(x) = \sin \pi x.$$

INPUT: $l = 1, \quad T = 0.5, \quad \alpha = 1, \quad m = 10, \quad N = 50$

WAVE EQUATION**“ALG124.EXT”****FINITE-DIFFERENCE METHOD**

This program uses the Wave Equation Finite-Difference Method to approximate the solution to a wave equation

$$\frac{\partial^2 u}{\partial t^2}(x, t) - \alpha^2 \frac{\partial^2 u}{\partial x^2}(x, t) = 0, \quad 0 < x < l, \quad 0 < t$$

subject to the boundary conditions

$$u(0, t) = 0 \quad \text{and} \quad u(l, t) = 0$$

and initial conditions

$$u(x, 0) = f(x) \quad \text{and} \quad \frac{\partial u}{\partial t}(x, 0) = g(x).$$

The sample problem uses

$$f(x) = \sin \pi x \quad \text{and} \quad g(x) = 0.$$

INPUT: $l = 1, \quad T = 1, \quad \alpha = 2, \quad m = 10, \quad N = 20$

FINITE-ELEMENT METHOD**“ALG125.EXT”**

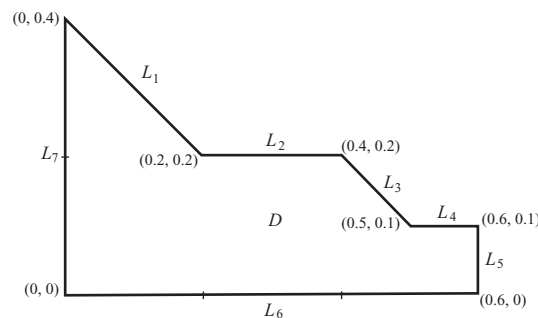
This program uses the Finite-Element Method to approximate the solution to an elliptic partial-differential equation of the form

$$\frac{\partial}{\partial x} \left(p(x, y) \frac{\partial u}{\partial x} \right) + \frac{\partial}{\partial y} \left(q(x, y) \frac{\partial u}{\partial y} \right) + r(x, y)u = f(x, y)$$

subject to Dirichlet, mixed, or Neumann boundary conditions. The sample problem considers Laplace's equation

$$\frac{\partial^2 u}{\partial x^2}(x, y) + \frac{\partial^2 u}{\partial y^2}(x, y) = 0$$

on the two-dimensional region shown in the figure below.



The boundary conditions on this region are

$$u(x, y) = 4, \quad \text{for } (x, y) \text{ on } L_6 \text{ and } L_7,$$

$$\frac{\partial u}{\partial n}(x, y) = x, \quad \text{for } (x, y) \text{ on } L_2 \text{ and } L_4,$$

$$\frac{\partial u}{\partial n}(x, y) = y, \quad \text{for } (x, y) \text{ on } L_5,$$

and

$$\frac{\partial u}{\partial n}(x, y) = \frac{x + y}{\sqrt{2}}, \quad \text{for } (x, y) \text{ on } L_1 \text{ and } L_3.$$

INPUT: ALG125.DTA