

Programs for Numerical Methods with the HP 48 G/G+/GX Calculator

By

Gilberto E. Urroz, Ph.D., P.E.

Distributed by

 *infoClearinghouse.com*

©2000 Gilberto E. Urroz
All Rights Reserved

PROGRAMS FOR NUMERICAL METHODS USING THE HP 48 G/G+/GX	2
Numerical methods with matrices	3
LUFAC: LU factorization using Crout method	3
TRIDG: Thomas algorithm for the solution of linear equations with a tri-diagonal matrix	4
ITRM: Iterative methods for solution of systems of linear equations	5
EIGEN: Calculation of eigenvalues	6
POWER(1): calculating the largest (in absolute value) eigenvalue by the power method	6
POWER(2): calculating the smallest (in absolute value) eigenvalue by the inverse power method	7
Shifting eigenvalues procedure	8
Solution of non-linear equations	8
Interval Halving Method	9
Graphics solution	9
Numerical solution	10
Fixed point iteration	11
Graphics solution	11
Numerical solution	12
Newton-Raphson method	12
Graphics solution	12
<u>Numerical solution</u>	13
A second example	13
The Secant Method	15
Graphics solution	15
Numerical solution	15
Solving other problems	15
Polynomial Approximation and Interpolation	16
Polynomial evaluation and deflation	16
Direct-fit Polynomial	17
Lagrange polynomials	17
Tables of differences	18
Newton Forward- and Backward-Difference Polynomials	19
Least-Square Polynomial Fitting	21
Multivariate Polynomial Approximation	23
Numerical Integration in the HP48G/GX calculator	25
Numerical integration using Newton forward-difference polynomials	25
Trapezoidal rule	25
Simpson's 1/3 Rule	26
Simpson's 3/8 rule	27
Newton-Cotes Formulas	27
Romberg Integration	28
Gaussian Quadrature	29

Programs for numerical methods using the HP 48 G/G+/GX

This guidebook contains information describing a number of sub-directories where numerical solutions have been implemented for matrix and linear algebra operations, eigenvalue estimation, solution of non-linear equations, polynomial approximation and interpolation, and numerical integration. A zip file containing these programs can be downloaded at:

<http://www.engineering.usu.edu/cee/faculty/gurro/MyBooks/NumPrograms.zip>

You will need a cable to transfer the programs from the computer to your calculator. Information on HP's connection cable can be found at:

<http://www.hp.com/calculators/accessories/connect.html>

Please report any errors in this document to: gurro@cc.usu.edu

Note: You will find many references to “the textbook” in this guide book. The textbook used to develop the present guidebook was Hoffman, J.D. , “Numerical Methods for Engineers and Scientists,” Mc-Graw Hill, Inc. , New York. 1992. However, the subjects referred to should be available in any numerical methods textbook.

Numerical methods with matrices

Within the sub-directory HOME\NUMM\MATX you will find the following variables:

[→ABS][MSIM][LUFAC][TRIDG][ITRM][EIGEN]

The operation of [→ABS] and of [MSIM] is not discussed in this document. In this guidebook we discuss the operation of the other subdirectories.

LUFAC: LU factorization using Crout method

Press [LUFAC] to get into this sub-directory. You will find the following variables:

[A][→LUF][P][Ap][U][L]

A is used to store a square matrix, e.g.,

$$\mathbf{A} = \begin{bmatrix} 2 & 3 & 5 \\ 3 & 1 & -2 \\ 1 & 3 & 4 \end{bmatrix}$$

In the calculator, use the following keystrokes:

[[2 3 5][3 1 -2][1 3 4]] [ENTER]

To enter matrix in level 1

[←][A]

To store matrix in variable A

[A]

To check that storage took place

Press [→LUF] to perform the LU factorization through Crout method with pivoting. You will get a message box with the message "Ready", indicating that the factorization was performed. Press [OK].

The result is the matrices **P**, **Ap**, **U** and **L**, such that

$$\mathbf{P} \cdot \mathbf{A} = \mathbf{Ap} = \mathbf{L} \cdot \mathbf{U}$$

P is a permutation matrix indicating any row exchange (pivoting) used to improve the LU decomposition. **A** is the original matrix. **Ap** is the permuted matrix, which can be calculated by multiplying **P** with **A**. **L** and **U** are the lower and upper triangular matrices resulting from the Crout method application on **Ap**.

For example, for the matrix A used above, press [P] to get:

$$\mathbf{P} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

This can be interpreted as indicating that, after pivoting, the original row 1 ended in row 3, row 2 into row 1, and row 3 into row 2. To verify this, press [Ap], to get

$$\mathbf{Ap} = \begin{bmatrix} 3 & 1 & -2 \\ 1 & 3 & 4 \\ 2 & 3 & 5 \end{bmatrix}$$

Compare the location of each row in \mathbf{Ap} with those of \mathbf{A} to see how the matrix \mathbf{P} can be used to interpret the pivoting.

The corresponding triangular matrices \mathbf{L} and \mathbf{U} are obtained by pressing [\mathbf{L}] and [\mathbf{U}], respectively. For the present case they are:

$$\mathbf{L} = \begin{bmatrix} 3 & 0 & 0 \\ 1 & 2.67 & 0 \\ 2 & 2.33 & 2.25 \end{bmatrix}, \quad \mathbf{U} = \begin{bmatrix} 1 & 0.33 & -0.67 \\ 0 & 1 & 1.75 \\ 0 & 0 & 1 \end{bmatrix}$$

When using matrices \mathbf{L} and \mathbf{U} for solving for the linear system, $\mathbf{A}\cdot\mathbf{x}=\mathbf{b}$, we first re-write the linear system as $\mathbf{P}\cdot\mathbf{A}\cdot\mathbf{x} = \mathbf{P}\cdot\mathbf{b}$. If we call $\mathbf{P}\cdot\mathbf{b} = \mathbf{b}^*$, then, the linear system becomes $\mathbf{Ap}\cdot\mathbf{x} = \mathbf{b}^*$. Since, $\mathbf{Ap} = \mathbf{L}\cdot\mathbf{U}$, we re-write the linear system as $\mathbf{L}\cdot\mathbf{U}\cdot\mathbf{x} = \mathbf{b}^*$, from which we can write $\mathbf{U}\cdot\mathbf{x} = \mathbf{b}'$, $\mathbf{L}\mathbf{b}' = \mathbf{b}^*$. Therefore, we would first solve for \mathbf{b}' from $\mathbf{L}\cdot\mathbf{b}' = \mathbf{b}^*$, and then solve for \mathbf{x} by using $\mathbf{U}\cdot\mathbf{x} = \mathbf{b}'$.

For example, for the present case, if $\mathbf{b} = [1 \ 3 \ 5]^T$, we will calculate $\mathbf{b}^* = \mathbf{P}\cdot\mathbf{b} = [3 \ 5 \ 1]^T$. Then, solve for \mathbf{b}' from $\mathbf{L}\cdot\mathbf{b}' = \mathbf{b}^*$, i.e., if $\mathbf{b}' = [b_1' \ b_2' \ b_3']^T$, we need to solve the set of equations:

$$\begin{aligned} 3\cdot b_1' &= 3 \\ b_1'+2.67\cdot b_2' &= 5 \\ b_1'+2.33\cdot b_2'+2.25\cdot b_3' &= 1 \end{aligned}$$

The solution is $b_1'=1.00$, $b_2' = 1.50$, and $b_3'=-2.00$, i.e., $\mathbf{b}' = [1 \ 1.5 \ -2]^T$.

Next, we solve for \mathbf{x} from $\mathbf{U}\cdot\mathbf{x} = \mathbf{b}'$, i.e., if $\mathbf{x} = [x_1 \ x_2 \ x_3]^T$, we need to solve the set of equations:

$$\begin{aligned} x_1+0.33\cdot x_2-0.67\cdot x_3 &= 1 \\ x_2+1.75\cdot x_3 &= 5 \\ x_3 &= 3 \end{aligned}$$

The solution is $x_1= -2.00$, $x_2 = 5.00$, and $x_3= -2.00$, i.e., $\mathbf{x} = [-2 \ 5 \ -2]^T$. To check, press [\mathbf{A}], enter the vector \mathbf{x} , [-2 5 -2][ENTER], and multiply, [×]. You should recover the vector $\mathbf{b} = [1 \ 3 \ 5]^T$.

TRIDG: Thomas algorithm for the solution of linear equations with a tri-diagonal matrix

The sub-directory TRIDG is located at HOME\NUMM\MATX\TRIDG. Within that subdirectory you find the following variables:

[A] [B] [→SOL] [X] [→AB] [→X]

To operate using this sub-directory you need to enter the tri-diagonal matrix ($n \times n$) as a matrix **A** of three rows and n columns. The columns correspond to the diagonal below the main diagonal, the main diagonal itself, and the diagonal above the main diagonal, respectively. This means that the elements a_{11} and a_{n3} will be zero.

For example, try entering this matrix: [[0 4 -1][-1 4 -1][-1 4 -1][-1 4 0]][ENTER]. To store it in A, use [←][A].

Also, enter a vector **b** = [150 200 150 100][ENTER]. Store it in variable b using: [←][B].

To solve for vector **x** in the equation $\mathbf{Ax} = \mathbf{b}$ press the button [→SOL]. The result for this particular case is $\mathbf{x} = [57.89 \ 81.57 \ 68.42 \ 42.11]$.

The variables [→AB] and [→X] are used to implement the Thomas algorithm: [→AB] simplifies **A** and **b**, and [→X] performs the back substitution to solve for **x**. These two programs are called by [→SOL]. To see details of these programs use: [↵][→AB], [↵][→X], and [↵][→SOL].

ITRM: Iterative methods for solution of systems of linear equations

This subdirectory, located in HOME\NUMM\MATX\ITRM contains programs that allow you to show the average residual and the current iteration of the value of **x** using either the Jacobi method ([→JAC]) or successive overrelaxation ([→SOR]). [Note: when used with $\omega = 1.00$, the SOR method becomes the Gauss-Seidel iterative method.]

To solve the system $\mathbf{Ax} = \mathbf{b}$, you need to enter the square matrix **A** and the vector **b** in the corresponding variables A and b. For example:

[[4 -1 0 0] [-1 4 -1 0] [0 -1 4 -1] [0 0 -1 4]][ENTER] [←][A].

[150 200 150 100][ENTER] [←][B].

We need to enter an initial value for the vector **x**, say [0 0 0 0][ENTER] [←][X].

To use the *Jacobi method*, just press [→JAC]. The result is the average absolute value of the residual, $R = 150$, and the improved value of $\mathbf{x} = [37.5 \ 50 \ 37.5 \ 25]$. The average residual is still pretty large, therefore, we need to keep iterating. Press [→JAC] and watch the results until the value of R is small enough. For the present case, the values of R for the second through the twelfth iteration are: 59.375, 24.22, 9.76, 3.96, 1.60, 0.65, 0.26, 0.11, 0.043, 0.017, 0.007. The latest value is close enough to zero. The corresponding solution is [57.89 81.58 68.42 42.10]. Compare with those obtained through use of the tri-diagonal method.

To use the *overrelaxation method*, enter a value for the parameter ω , say, [1].[.][5] [←][ω]. Re-initialize the solution by using: [0 0 0 0][ENTER] [←][X]. Then, press the [→SOR] button until the residual is as close to zero as you want it. For the first twelve iterations you should get the following values of R : 211.16, 57.40, 27.57, 12.91, 7.23, 5.17, 2.89, 1.51, 0.88, 0.54, 0.23, 0.081. At this point the solution is $\mathbf{x} = [57.88 \ 81.58 \ 68.43 \ 42.12]$. A better rate of convergence is achieved by using $\omega = 1.1$, with the average residual being 0.014 after 6 iterations only.

Recall that by using $\omega = 1.0$, the SOR method becomes the Gauss-Seidel iteration method.

EIGEN: Calculation of eigenvalues

This sub-directory, located at HOME\NUMM\MATX\EIGEN contains three sub-directories, namely [CEQ2], [CEQ3], and [FADLE]. The sub-directories [CEQ2] and [CEQ3] are used to produce the coefficients of the polynomial that represents the characteristic equation $\det(\mathbf{A}-\lambda\mathbf{I}) = 0$, for the matrix \mathbf{A} having a size of 2×2 and 3×3 , respectively.

For a 2×2 matrix, the characteristic equation is given by:

$$\lambda^2 - \text{tr}(\mathbf{A})\lambda + \det(\mathbf{A}) = 0,$$

where $\text{tr}(\mathbf{A})$ is the trace of \mathbf{A} .

For a 3×3 matrix, the characteristic equation is given by:

$$\lambda^3 - \text{tr}(\mathbf{A})\lambda^2 + (\sum \det(\mathbf{M}_{ij}))\lambda - \det(\mathbf{A}) = 0,$$

where $\text{tr}(\mathbf{A})$ is the trace of \mathbf{A} , and \mathbf{M}_{ij} are the reduced 2×2 matrices, $\mathbf{M}_{ij} = [[a_{ii} \ a_{ij}][a_{ji} \ a_{jj}]$, and the summation includes three reduced matrices corresponding to $(i = 1, j=2)$, $(i = 1, j=3)$, and $(i = 2, j=3)$.

To operate [CEQ2] or [CEQ3] enter the corresponding 2×2 and 3×3 matrix into \mathbf{A} , then press [\rightarrow EVAl]. The result, in any of those sub-directories, is a vector containing the coefficients of the polynomial characteristic equation corresponding to descending order of powers of λ , and a second vector with the roots of the corresponding polynomial, i.e., with the eigenvalues.

For example, in [CEQ2], enter $[[2 \ 3][3 \ 1]]$ [\leftarrow][\mathbf{A}], then press [\rightarrow EVL]. The results are:

Coeff: [1 -3 -7], i.e., $\lambda^2 - 3\lambda - 7 = 0$, and, $\lambda = [-1.54 \ 4.54]$.

For a second example, in [CEQ3], enter $[[2 \ 3 \ 5][3 \ 1 \ -2][1 \ 3 \ 4]]$ [\leftarrow][\mathbf{A}], then press [\rightarrow EVL]. The results are: Coeff: [1 -7 6 -18], i.e., $\lambda^3 - 7\lambda^2 + 6\lambda - 18 = 0$, and, $\lambda = [(0.25, 0.65), (0.25, -0.65), (6.50, 0)]$. In this case two of the eigenvalues are complex numbers, namely, $\lambda_1 = 0.25 + 0.65i$, and $\lambda_2 = 0.25 - 0.65i$.

The sub-directory FADLE (HOME\NUMM\MATX\EIGEN\FADLE) contains an implementation of the Fadeev-Levelier method to generate the polynomial characteristic equation for a square matrix of any size. To operate, enter the matrix \mathbf{A} , for example: $[[1 \ 2 \ 3 \ 1][2 \ 1 \ 2 \ -2][3 \ 2 \ 5 \ -1][1 \ 3 \ -2 \ 6]]$ [\leftarrow][\mathbf{A}], then press [\rightarrow EVL]. The result in this case is Coeff: [1 -13 39 19 -40], i.e., $\lambda^4 - 13\lambda^3 + 39\lambda^2 + 19\lambda - 40 = 0$, and, $\lambda = [0.90 \ -1.06 \ 5.45 \ 7.71]$.

Note: once you have determined the eigenvalues of a matrix, you can find their associated eigenvectors by using the equation $(\mathbf{A}-\lambda\mathbf{I})\mathbf{x} = 0$, and solving for \mathbf{x} as outlined in page 69 of the textbook.

POWER(1): calculating the largest (in absolute value) eigenvalue by the power method

This subdirectory, located in HOME\NUMM\MATX\EIGEN\POWER, contains the program [\rightarrow NEW] that lets you iterate the procedure for the power method for eigenvalues. This method calculates improved values of the largest (in absolute value) eigenvalue of a matrix and its associated eigenvector. Within sub-directory POWER you will find the following variables:

$$[\mathbf{A}] [\text{AINV}] [\mathbf{X}] [\text{IU}] [\rightarrow \text{NEW}] [\rightarrow \text{NEI}]$$

To obtain the largest (in absolute value) eigenvector, enter the matrix \mathbf{A} ($n \times n$) in the corresponding variable A. Enter a first guess for the vector \mathbf{x} , typically a vector will all n components equal to 1.0, in variable x . The method requires that one of the components in the vector remain equal to 1.0 at each iteration. Therefore, store the index corresponding to that component in variable IU, i.e., IU can take a value of 1 to n . Press \rightarrow NEW] to obtain the next value of the eigenvalue λ and its associated eigenvector \mathbf{x} . Continue pressing \rightarrow NEW] until the eigenvalue converges to a steady value.

As an example, use:
 $\left[\begin{array}{ccc} 2 & 3 & 5 \\ 3 & 1 & -2 \\ 1 & 3 & 4 \end{array} \right]$ \leftarrow [A]
 $\left[\begin{array}{ccc} 1 & 1 & 1 \end{array} \right]$ [ENTER] \leftarrow [X]
 1 \leftarrow [IU]
 Press \rightarrow NEW] until λ converges.

For this example you should get values of $\lambda = 10.0, 6.60, 6.36, 6.49, 6.51, 6.50, 6.50, 6.50$. Thus, the largest (in absolute value) eigenvalue converges to 6.50. The corresponding eigenvector is [1.00 0.28 0.73].

Repeat the problem with $IU = 2$, i.e., 1 \leftarrow [IU]. The values of λ are now 2.00, 8.00, 7.38, 6.49, 6.45, 6.50, 6.51, 6.50, 6.50. As expected, λ converges to 6.50, but the eigenvector is now [3.59 1.00 2.63]. You can, however, show that $[3.59 \ 1.00 \ 2.63] = 3.59 \cdot [1.00 \ 0.28 \ 0.73]$, i.e., the two eigenvectors are parallel, and, therefore, equivalent. [Note: vectors \mathbf{u} and \mathbf{v} are parallel if $\mathbf{u} = c \cdot \mathbf{v}$, where c is a real constant].

POWER(2): calculating the smallest (in absolute value) eigenvalue by the inverse power method

Sub-directory POWER also contains the program \rightarrow NEI] that lets you iterate the procedure for the inverse power method for eigenvalues. This method calculates improved values of the smallest (in absolute value) eigenvalue of a matrix and its associated eigenvector.

First, enter the matrix \mathbf{A} ($n \times n$) in level 1 of the display, and press [ENTER] to obtain a second copy. Store the matrix in the variable A for future reference. Press the $[1/x]$ button to obtain the inverse, \mathbf{A}^{-1} . Enter a first guess for the vector \mathbf{x} , typically a vector will all n components equal to 1.0, in variable x . The inverse power method also requires that one of the components in the vector remain equal to 1.0 at each iteration. Therefore, store the index corresponding to that component in variable IU. Press \rightarrow NEI] to obtain the next value of the eigenvalue λ and its associated eigenvector \mathbf{x} . Continue pressing \rightarrow NEI] until the eigenvalue converges to a steady value.

As an example, use:
 $\left[\begin{array}{ccc} 2 & 3 & 5 \\ 3 & 1 & -2 \\ 1 & 3 & 4 \end{array} \right]$ [ENTER][ENTER]
 \leftarrow [A]
 $[1/x]$
 \leftarrow [AINV]
 $\left[\begin{array}{ccc} 1 & 1 & 1 \end{array} \right]$ [ENTER] \leftarrow [X]
 1 \leftarrow [IU]
 Press \rightarrow NEI] until λ converges.

For this example you should get values of $\lambda = 9.00, 0.55, 2.54, -1.73, 1.31, -3.30, 0.73, -12.17, 0.22, 9.94, -0.29, 3.50, -0.92, 1.95, -1.90$, at this point you have already performed 15 iterations and there is no convergence. (I tried up to 40 iterations without convergence). The reason is that this particular matrix has only one real eigenvalue, the largest one that we found earlier, i.e., $\lambda = 6.50$. The other two eigenvalues are complex, therefore, the inverse power method will not be able to provide a solution for the smallest (in absolute value) eigenvalue, and we would have to rely on the Fadeev-Leverlier method presented earlier to obtain the other characteristic equation and the eigenvalues.

There is no reliable way to determine whether a matrix of real coefficients has complex eigenvalues. One known fact is, however, that symmetric real matrices have only real eigenvalues. Let's try using the inverse power method with a symmetric matrix:

```

[[ 2 3 5][3 1 -2][5 -2 4]] [ENTER] [ENTER]
[←][ A ]
[1/x]
[←][AINV]
[ 1 1 1 ][ENTER] [←][ X ]
1 [←][ IU ]
Press [→NEI] until  $\lambda$  converges.

```

The first ten values of λ for this case are: 3.67, 9.31, 2.20, 6.39, 2.21, 4.77, 2.58, 4.07, 2.88, 3.07. Although the eigenvalue seems to be oscillating from a small number to a larger one and so forth, it is actually converging to a number ... (the last two values). Let's keep using the [→NEI] program, with values of $\lambda = 3.55, 3.19, 3.46, 3.25, 3.41, 3.29, 3.38, 3.31, 3.36, 3.33, 3.35, 3.33, 3.34, 3.34, 3.34$. Convergence is achieved after about 25 iterations. It seems that convergence is slower for the inverse power method. The smallest (in absolute value) eigenvalue is, therefore, $\lambda = 3.34$, and its associated eigenvector is [1.00 2.16 - 1.03].

You can check, by using [1 1 1] as the initial value of x , and repeated use of the program [→NEW], that the largest (in absolute value) eigenvalue for this symmetric real matrix is $\lambda = 8.12$, and its associated eigenvector is [1.00 0.09 1.17].

Shifting eigenvalues procedure

I have not prepared a separate program for applying the method of shifting eigenvalues (page 77-79 in textbook) since it basically uses the power and inverse power methods already programmed in sub-directory POWER.

As an example, we will follow the procedure suggested in page 77 of the textbook for the matrix \mathbf{A} used above, i.e., $\mathbf{A} = [[2 3 5][3 1 -2][5 -2 4]]$. Start by storing \mathbf{A} in variable A. The steps to follow next, as indicated in page 77, are:

1. *Solve for the largest (in magnitude) eigenvalue λ_1 .* You can verify that $\lambda_1 = 8.12$ using the power method as shown above.
2. *Shift the eigenvalues of \mathbf{A} by $s = \lambda_1 (= 8.12)$.* This is accomplished by calculating a matrix $\mathbf{A}_s = \mathbf{A} - \lambda_1 \mathbf{I}$. In the calculator you can use the following keystrokes:

```
[ A ] [8] [.] [1] [2] [ENTER] [3] [MTH] [MAKE] [IDN] [×] [ - ]
```

Store this new matrix in A.

3. *Solve for the eigenvalue λ_s of the shifted matrix \mathbf{A}_s by the power method.* Using the power method procedure outlined above, we find $\lambda_s = -12.58$.
4. *Calculate the largest eigenvalue of opposite sign by $\lambda = \lambda_s + s$.* For this case, $\lambda = \lambda_s + s = -12.58 + 8.12 = -4.46$.

Solution of non-linear equations

Under subdirectory NUMM you will have a subdirectory called NLEQ (Non-Linear EQUations) containing the following subdirectories:

- TPLO: used to illustrate x-y plots in the calculator
- INTH: interval-halving solution for non-linear equations
- P3.7: refers to problem 3.7 in the textbook, in which the fixed-point iteration method is used
- NWT: Newton-Raphson method for solving non-linear equations
- SECN: secant method for solving non-linear equations
- TWOC: used to illustrate conic plots in the calculator
- NLSY: examples of systems of non-linear equations when the equations are not those of conics.

NOTE: The examples contained in TWOC and NLSY will be solved using the library SOLVESYS, developed by Sune Bredahl (e-mail: c947086@student.dtu.dk), and available in the Internet at the following URL: www.gbar.dtu.dk/~c947086/hp48.html (Mr. Bredahl is a computer scientist from Denmark who develops binary applications for the HP48G/GX calculator as a hobby).

Interval Halving Method

Use subdirectory INTH. Within that subdirectory you will find the following variables:

[A][B][C][F][\rightarrow NXC][X]

If you press [NXT] you will get the next menu:

[EQ][PPAR][ZPAR][CHSL][CHKS][]

For solving non-linear equations using the interval-halving method we need to be concerned only with variables A, B, C, F, and \rightarrow NXC. To plot the function $f(x)$ we need to use the variable EQ. X, PPAR and ZPAR are used in the plotting of the function. CHSL and CHKS are programs used in the implementation of the interval-halving method.

Press [NXT] or [VAR] to get to the main menu. Press [\rightarrow][F] to see the function that is currently defined in variable F. You will get the following program:

<< \rightarrow x 'x^3-2*x^2-2*x+1' >>

This is the function from problem 1(d).

Graphics solution

If you press [NXT][EQ], you will find that the current value of the EQ variable is the function 'x^3-2*x^2-2*x+1', i.e., the same as the algebraic expression included in F. You use the variable EQ to plot the function and give you an idea of where the roots are located. To plot the function use the procedure outlined above for plotting expressions of the type $y=f(x)$. For this particular case use the following keystroke sequence to see the plot: [\rightarrow][PLOT][ERASE][DRAW] (The plot ranges are already defined). Press [EDIT][NXT][LABEL] to see the labels. Press [NXT][NXT][PICT][FCN] and use the [ROOT] key to find the three roots of the curve. You should obtain as solutions: -1, 0.3819, and 2.6180. Now, that would be cheating since we have not use interval-halving at all. To use the method, try the following. Get out of the plot environment:

[NXT][NXT][PICT][CANCL][ENTER].

Numerical solution

Press [VAR] to see the main menu. Now, we know that the first solution is -1 (because we cheated, as shown above), but suppose that we didn't. We did notice from the graph that the curve goes from negative to positive when x goes from -1 to 0, so let's use $a = -1$ and $b = 0$, by using:

[2][+/-][←][A]
[0][←][B]

Then, to obtain c , press [→NXC] (NeXt C). It shows $c:-1$. Pressing [→NXC] shows the message: A solution exists at a . Press [OK] and then [A] to see what the solution is. Surprise! It is equal to -1. Let's try something more challenging.

Select $a = 0$, $b = 1.5$. Enter:

[0][←][A]
[1][.][5][←][B]

Then press [→NXC]. You'll get $c: 0.75$. Press [→NXC] again. Keep pressing it until the value of c converges to a steady value. You should get values of $c = 0.375, 0.5625, 0.4687, 0.4218, 0.3984, 0.3867, 0.3808, 0.3837, 0.3823, 0.3815, 0.3819, 0.3821, 0.3820, 0.3820, 0.3819, 0.3819, 0.3819$. After 18 iterations the interval-halving method is providing a value of c close to 0.3819. To see how well our value of 0.3819 satisfies the condition $f(x) = 0$, enter 0.3819 in level 1 of the display and press the key [F]. We find that $f(0.3819) = 0.000204$. In other words, the error incurred in evaluating $f(x)$ with $x = 0.3819$ is of the order of $2E-4$. That is close enough to $f(x) = 0$, that we will take the second root of the equation to be $x = 0.3819$. Compare with the value found above using the ROOT command in the HP48G.

To find the third root, try using $a = 2$, $b = 3$. Within 18 iterations your solution should converge to $x = 2.6180$, which is the same found through the use of the ROOT command.

If you want to use this subdirectory to solve for the roots of a different function you need to replace the values of the variables F and EQ. As an illustration, let's use interval halving to solve for problem 1(a). First enter the algebraic expression defining the function, i.e., 'x-COS(x)' in level 1 of the display:

['][α][←][X][-][COS][α][←][X][ENTER][ENTER]

Then, store this expression into EQ:

[NXT][←][EQ]

Now, modify the expression in level 1 to read 'f(x) = x-COS(x)' by using:

[←][EDIT][▶][α][←][F][←][()][α][←][X][▶][←][=][ENTER]

Now, we will define the function as a program by using:

[←][DEF]

To check the current contents of variable F, press [↔][F]. It should show: << → x 'x - COS(x)' >>

To plot the function use a procedure similar to that shown above. We will go directly to solving the equation by using the interval limits suggested by the problem, i.e., $a = 0.5$ and $b = 1.0$. Enter those values in the corresponding variables:

[0][.][5][←][A]
 [1][←][B]

Using the program [→NXC], after about 20 iterations, the solution converges to c: 0.7390. Let's check the value of the function f(x) for x = 0.7390. Enter that value in level 1 of the display and press [F]. The result is -0.0001424. The error is then of the order of 1E-4.

Fixed point iteration

An example of this method is contained in subdirectory P3.7 (Problem 3.7). Within that subdirectory you will find the following menu:

[XA][XB][XC][EQ][EQ1][EQ2]

Pressing [NXT] shows the second menu:

[EQ3][X][PPAR]

The variables XA, XB, and XC were loaded by using [←][DEF] for the following expressions:

'xa(x)=EXP(x)-(3*x+2)'
 'xb(x)=(EXP(x)-2)/3'
 'xc(x)=LN(3*x+2)'

The resulting programs can be seen by pressing the following:

[↵][XA] To get: << → x, 'EXP(x)-(3*x+2)' >>
 [↵][XB] To get: << → x, '(EXP(x)-2)/3' >>
 [↵][XC] To get: << → x, 'LN(3*x+2)' >>

The variables EQ1, EQ2, and EQ3 contain lists of functions of x, namely,

EQ1 = {x 'EXP(x)-(3*x+2)'}, EQ2 = { x '(EXP(x)-2)/3' }, and EQ3 = {x 'LN(3*x+2)' }.

They are used to show the intersection of the curves $y = x$ and $y = f(x)$, which is the solution to each of the corresponding problems.

Graphics solution

For example, the current content of EQ is the same as EQ1. To plot the figure use: [↵][PLOT][ERASE][DRAW]. The resulting figure shows two points of intersection. To find what they are, use [FCN], then move the cursor close to either one of them, say the left one, and press [ISECT]. The result is I-SECT: (-0.4552, -0.4552). The second point of intersection is located at (2.1253, 2.1253). In other words, the solutions for the expression $x = \exp(x) - (3x+2)$ are $x = -0.4552$ and $x = 2.1253$. Press [NXT][NXT][PICT][CANCL][ENTER] to return to the normal display. The coordinates of the intersection points will be shown in the stack.

To solve case b and c using the PLOT environment, first you need to copy the contents of EQ2 or EQ3 into EQ, then follow a procedure similar to that shown above. You may need to change the plot ranges (horizontal and vertical), but the general procedure is the same as above.

Numerical solution

The solution using fixed-point iteration consists in entering an initial value for whichever case you want to solve, the pressing the corresponding function [XA], [XB], or [XC], until the function converges.

(a) For example, enter 0.0 and press [XA] repeatedly. You should get values oscillating between positive and negative without convergence. Let's try a different starting value, say -0.5. Enter that value and press [XA] repeatedly, the results again diverge. Let's use a positive value, say, 2.0, to start the solution. Again, the solution diverges. It seems that the fixed-point equation defined in XA is very unstable in the sense that it quickly diverges regardless of the initial value chosen for the solution.

(b) Let's try instead XB, with $x = 0$. Press [XB] repeatedly to get: -0.3333, -0.4278, -0.4493, -0.01388, -0.3379, -0.4289, -0.4495, -0.4503, -0.4549, -0.4551, -0.4552, -0.4552. One solution is then, $x = -0.4552$. Starting with values of 1.0 or 2.0 will get you back to the negative solution. Starting with a value of 2.5 or 3.0 will give you a divergent process. It seems that using XB will only allow you to obtain one solution.

(c) Let's try instead XC with $x = 0$ (or any other positive number). Pressing [XC] repeatedly will give you a solution at $x = 2.1253$. Starting with a negative value, say -1.0, will give you a complex solution. The expression for XC, therefore converges only to the positive solution.

Moral of the exercise: fixed-point iteration depends not only on the initial value selected for the iteration, but also in the equation chosen to solve the problem. One of the expressions used here always gave us divergence, and the other two would only show convergence towards one, but not the other, solution.

Solving other problems: simply define your function $xf = f(x)$ by entering the expression in level 1, as 'xf = f(x)' and using [↵][DEF]. Then, select an initial value, and press [XF] repeatedly until convergence is achieved or until it is clear that the solution oscillates or diverges. In the latter case, try a different expression for $xf = f(x)$.

Newton-Raphson method

Use subdirectory NWT. In that subdirectory you will find the menu:

[X][→NEW][F][FP][EQ][PPAR]

The next menu is obtained by pressing [NXT]:

[ZPAR][][][][][][]

EQ, PPAR and ZPAR are used for plotting. The function [F] is a program, generated by using [↵][DEF], while [EQ] contains the same algebraic expression as used in [F]. The function [FP] contains the derivative of [F]. This was also defined using [↵][DEF].

The current examples has $f = \ll \rightarrow x \text{ 'x^3-2*x+1' } \gg$, $fp = \ll \rightarrow x \text{ '3*x^2-4*x' } \gg$, $EQ = \text{'x^3-2*x+1'}$.

Graphics solution

The procedure is the same as that shown for the Interval-halving method. Therefore, the graphics solution will not be presented here. The solutions, obtained using the command ROOT in the PLOT environment are: -0.6180, 1.00, 1.6180.

Numerical solution

Enter an initial value for x, then press [→NEW] until the solution converges. For example, set x = -10, i.e.,

[1][0][+/-][←][X]

Pressing [→NEW] repeatedly produces values of x = -10.0, -6.4735, -4.1380, -2.6053, -1.6223, -1.0291, -0.7263, -0.6286, -0.6181, -0.6180, -0.6180, -0.6180. The solution converges to x = -0.6180. Enter that value in level 1 of the stack and press [F]. The result is $f(-0.6180) = 0.0001229$, i.e., the error is of the order of $1E-4$.

Now, enter a value of x = 10.0, and press [→NEW] repeatedly to prove that the solution converges to x = 1.6180, with an error of the order of $4E-5$.

To find the third solution you have to enter a value between 0.5 and 1.25, otherwise, the solution converges to either of the two solutions found before. For example, try x = 0.8. The solution converges to 1.0 in 4 iterations. The error is zero, i.e., this is an exact solution.

A second example

You can use the procedure programmed in [→NEW] (for NEWton-Raphson) to obtain numerical solutions to other equations as long as you enter the original function (using ' $f(x) = \text{expression containing } x$ ' and [←][DEF]) and its derivative (using ' $fp(x) = \text{expression containing } x$ ' and [←][DEF]). You can even use the calculator to obtain the derivative for you. The following example shows you how to proceed.

Suppose that $f(x) = e^x - \sin(\pi x/3)$, the procedure to load EQ, F and FP is as follows:

- Enter the algebraic expression defining f(x) in level 1 of the stack, and make two copies of it:

['][←][e^x] [α][←][X] [▶] [-] [SIN] [←][π] [×] [α][←][X] [÷] [3] [ENTER][ENTER][ENTER]

There will be three copies of the expression ' $\text{EXP}(x) - \text{SIN}(\pi * x / 3)$ ' in levels 1 through 3 of the stack.

- For graphical analysis, store one copy of the expression in EQ by using: [←][EQ]
- Modify the second copy to make it look like this: ' $f(x) = \text{EXP}(x) - \text{SIN}(\pi * x / 3)$ '. Use:

[←][EDIT][▶][α][←][F][←][()] [α][←][X][▶][←][=][ENTER]

- Load f(x) by using: [←][DEF]

Next, we need to take the derivative of x using the HP48G/GX command, but first, we need to purge the variable [X]. Otherwise, the derivative will be evaluated at the current value of x instead of producing a general expression fp(x).

- Purge [X] by using: ['][X] [←][PURG]
 - Evaluate the derivative by entering: ['] [α][←][X] [ENTER] [←][∂]
- The resulting expression is: ' $\text{EXP}(x) - \text{COS}(\pi * x / 3) * (\pi / 3)$ '
- Modify the expression to read: ' $f_p(x) = \text{EXP}(x) - \text{COS}(\pi * x / 3) * (\pi / 3)$ ' by using:

[←][EDIT][▶][α][←][F][←][P][←][()] [←][X][α][▶][←][=][ENTER]

- Load $fp(x)$ by using: [←][DEF]
- Enter a value of x , say $x = 1.0$, by using: [1][.] [←][X][STO].
- Press [VAR], if needed, to see the subdirectory variable menu.

At this point all the required variables are loaded and you can proceed as in the first example described above. However, there is a small surprise in store for you. If you press [→NEW], instead of getting a numerical value, you get an algebraic expression. Try it. That is so because, every time that you have p in an expression, the calculator tends to preserve the expression as an algebraic rather than producing a numeric result. To force the calculator to produce numeric results when evaluating [F] and [FP], we need to modify the programs stored in those variables as follows:

First, recall and modify the contents of [F] by using:

[↔][F][←][EDIT][▼][▼] (i.e., position cursor to the left of the >> symbol) [←][→NUM][ENTER]

Store the modified program into [F]:

[←][F]

Next, recall and modify the contents of [FP] by using:

[↔][FP][←][EDIT][▼][▼] (i.e., position cursor to the left of the >> symbol) [←][→NUM][ENTER]

Store the modified program into [FP]:

[←][FP]

Reload the value of 1 into x , by using:

[1][←][X].

Now, press [→NEW]. You should get $x = 0.1560$. Pressing [→NEW] repeatedly will produce a solution that converges to $x = -3.0454$. (See problem 1(b) in the Numerical Methods textbook).

NOTE: In some instances, when obtaining the symbolic derivative using the HP48G/GX ∂ command, you will need to simplify the resulting expression by using the command [COLCT] available in the menu [←][SYMBOLIC].

The Secant Method

Use subdirectory SECN. In that subdirectory you will find the menu:

```
[ X1 ][ X2 ][→NEW][→FP][ F ][ FP ]
```

The next menu is obtained by pressing [NXT]:

```
[ X ][ EQ ][PPAR][ZPAR][ ][ ]
```

X, EQ, PPAR and ZPAR are used for plotting. The function [F] is a program, generated by using [↵][DEF], while [EQ] contains the same algebraic expression as used in [F]. The program [→FP] calculates the derivative of [F] using the secant method. Current values of the derivative are stored in variable [FP]. The method requires you to provide two initial values [X1] and [X2], where x_2 is very close to x_1 . The program used to achieve convergence is [→NEW], which calls program [→FP].

The current examples has the same function as in the interval-halving example, i.e.,

```
f = << → x 'x^3-2*x^2-2*x+1'>>, EQ = 'x^3-2*x^2-2*x+1'.
```

Graphics solution

The procedure is exactly the same as that shown for the Interval-halving method. Therefore, the graphics solution will not be presented here. The solutions, obtained before, are: -1, 0.3819, and 2.6180.

Numerical solution

Enter initial values for x_1 and x_2 , then press [→NEW] until the solution converges. For example, set $x_1 = 0$, $x_2 = 0.1$, i.e.,

```
[0][↵][ X1 ][.][1][↵][ X2 ]
```

Pressing [→NEW] repeatedly produces values of $x = 0.4566, 0.3741, 0.3818, 0.3819, 0.3819, 0.3819$. The solution converges to $x = 0.3189$. Enter that value in level 1 of the stack and press [F]. The result is $f(-0.6180) = 0.00020$, i.e., the error is of the order of $2E-4$.

Now, enter a value of $x_1 = -2.0$, $x_2 = -1.9$, and press [→NEW] repeatedly to prove that the solution converges to $x = -1$, an exact solution, i.e., error = 0.0, in about 10 iterations.

To find the third solution use, for example, $x_1 = 5.0$, $x_2 = 5.1$. Convergence to $x = 2.6180$ is achieved in about 8 iterations.

Solving other problems

For the secant method you only need to define $f(x)$, by entering ' $f(x) = \text{expression containing } x$ ' in level 1 of the stack and using [↵][DEF]. The procedure for the solution is as outlined above.

Polynomial Approximation and Interpolation

Use sub-directory PA&I under sub-directory NUMM. The following sub-directories are available:

POLN: for polynomial evaluation and deflation.
DFPL: direct-fit polynomials
LGNP: Lagrange polynomials
TABLS: for generating difference tables
NF&B: Newton forward- and backward-difference polynomials
CFIT: for least-squares approximation
LSPL: for least-squares polynomial fitting

Polynomial evaluation and deflation

Use sub-directory POLN. You will find the following variables:

[INFO][A][→PX][↑DFP][N][B]

Pressing [NXT] you will get the following menu:

[QX][NSTM][REVV][V→L][L→V][]

Press [NXT] to get to main menu. Press [INFO] to get an explanation of the sub-directory's operation. The instructions indicate that you should store the vector $\mathbf{a} = [a_n \ a_{n-1} \ a_{n-2} \ \dots \ a_3 \ a_2 \ a_1 \ a_0]$, containing the coefficients of the polynomial, $P_n(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_2 x^2 + a_1 x + a_0$, into variable [A], by using [←][A]. Next, enter a value of x in level 1 of the display, and, finally, press the [→PX] key to evaluate the polynomial P_n at x, as well as the deflated polynomial, Q_{n-1} at x. The program also provides the coefficients of the deflated polynomial, $P'_n(x) = Q_{n-1}(x)$, as a vector.

As an example, store the vector [1 -15 85 -225 274 -120] into [A]. These values correspond to the coefficients of the polynomial, $P_5(x) = x^5 - 15x^4 + 85x^3 - 225x^2 + 274x - 120$. Enter the value 2.5 in the display, and press [→PX]. The results are:

```
4:                x: 2.5
3:      P5(x): -1.40625
2:      Q4(x): .5625
1: Q4-Coeff:[ 1 -12.5 53.75 -90.625 47.4375]
```

These results are interpreted as follows:

$P_5(2.5) = -1.40625$; $P'_5(2.5) = Q_4(2.5) = 0.5625$, and $Q_4(x) = x^4 - 12.5x^3 + 53.75x^2 - 90.625x + 47.4375$.

Deflating is accomplished when we know one of the roots of the polynomial. (Deflating is basically synthetic division). For example, with the same vector of coefficients as before, enter a value of x = 2 in the display, and press [→PX]. The results are:

```
4:                x: 2
3:      P5(x): 0
2:      Q4(x): -6
1: Q4-Coeff:[ 1 -13 59 -107 60]
```

These results are interpreted as follows:

$P_5(2) = 0$ (i.e., $x=2$ is a root of $P_5(x)$); $P'_5(2) = Q_4(2) = -6$, and $Q_4(x) = x^4 - 13x^3 + 59x^2 - 107x + 60$.

Pressing [\uparrow DFT] will take you to sub-directory DFPL.

Direct-fit Polynomial

Use sub-directory DFPL. You will find the following variables:

[INFO][X][F][N][\rightarrow GET][\uparrow POL]

Pressing [NXT] you will get the following menu:

[CHK][GCFF][CRMT][][]

Press [NXT] to get to main menu. Press [INFO] to get an explanation of the sub-directory's operation. For direct-fit polynomials you are required to store lists of values of x and $f(x)$, these are the lists $\{x\}$ and $\{f\}$, as well as the order of the polynomial you want to fit to the data, i.e., n . If m is the size of the list, n should be chosen so that $1 \leq n < m$. Press [\rightarrow GET] to get the coefficients of the fitted polynomial.

For example, store $\{3.4\ 3.5\ 3.6\}$ in [X], and $\{0.294118\ 0.285714\ 0.277778\}$ in [F]. Next, store a value of 5 into [N], and press [\rightarrow GET]. As a result, you will get a message indicating that you should "Use $1 \leq n < 3$ ". Store a value of 2 into [N], and press [\rightarrow GET]. The result is now the vector:

[2.34E-2 -.2455 .858314],

or the polynomial: $P_2(x) = 0.0234x^2 - 0.2455x + 0.85831$.

To evaluate the fitted polynomial for a given value of x , you can press the key [\uparrow POL], which will take you into sub-directory POLN (see above). Next, you can store the vector currently in the display into [A], enter the value of x , and press [\rightarrow PX] to get the evaluated polynomial. For this case, for example, you can check that $P_2(3.55) = 0.28168$. Also, check that $P_2(3.4) = 0.29411799998$, $P_2(3.5) = 0.285713999998$, and $P_2(3.6) = 0.0.2777779999998$.

Lagrange polynomials

Use sub-directory LGNP. You will find the following variables:

[INFO][XL][FL][N][\rightarrow PX][GETC]

Pressing [NXT] you will get the following menu:

[GETF][PRDT][CHK][][]

Press [NXT] to get to main menu. Press [INFO] to get an explanation of the sub-directory's operation. The instructions require that you store lists with values of $\{x\}$ and $\{f\}$ into variables [X] and [F], respectively. You also need to enter the degree of the polynomial, n , into variable [N]. If m is the size of the lists $\{x\}$ and $\{f\}$, n should be chosen so that $1 \leq n < m$. To evaluate the polynomial, enter the value of x in level 1 of the stack and press [\rightarrow PX].

If the lists used for the polynomial evaluation are $xL = \{x_1 x_2 \dots x_m\}$, and $fL = \{f_1 f_2 \dots f_m\}$, the polynomial evaluated can be written as:

$$P_n(x) = C_1(x)f_1 + C_2(x)f_2 + \dots + C_{n+1}f_{n+1},$$

where

$$C_j(x) = N_j(x)/D_j(x), \quad (j = 1, 2, \dots, n+1),$$

and

$$N_j(x) = \prod_{\substack{k=1,2,\dots,n+1 \\ k \neq j}} (x - x_k) \qquad D_j(x) = \prod_{\substack{k=1,2,\dots,n+1 \\ k \neq j}} (x_j - x_k)$$

For example, store the lists $xL = \{ 3.4 \quad 3.5 \quad 3.55 \quad 3.65 \}$ and $fL = \{ 0.294118 \quad 0.285714 \quad 0.28169 \quad 0.273973 \}$. Store $n = 5$, enter a value of 3.55 in level 1, and press [\rightarrow PX]. You will get, as a result, a message requesting you to "Use $1 \leq n < 4$ ". Enter a value of $n = 3$, enter 2.5 in level 1, and press [\rightarrow PX] again. The results are now: $x: 2.5$, $P_3(x): 0.397414$, interpreted as: $P_3(2.5) = 0.387414$.

Tables of differences

Use sub-directory TABLS. You will find the following variables:

$$[INF0][FL][N][\rightarrow TBL][\rightarrow \Delta F][\rightarrow \nabla F]$$

Pressing [NXT] you will get the following menu:

$$[\Delta F][\nabla F][F][M][\rightarrow ZLIS][CRMT]$$

Press [NXT] to get to main menu. Press [INF0] to get an explanation of the sub-directory's operation. The instructions indicate that you should store the list $\{fL\}$, and the level of differences in the table, n , into variables $[fL]$ and $[N]$, respectively. The next step is to generate a table of differences by using [\rightarrow TBL]. Once this table has been generated (it is stored in variable $[F]$ and shown in level 1 of the stack), you can use either [\rightarrow ΔF] to generate the vector $\Delta F = [f_0 \Delta f_0 \Delta^2 f_0 \dots \Delta^n f_0]$ (for forward-difference polynomials), or the vector $\nabla F = [f_0 \nabla f_0 \nabla^2 f_0 \dots \nabla^n f_0]$ (for backward-difference polynomials).

For example, enter the list $fL = \{0.333333 \quad 0.322581 \quad 0.,312500 \quad 0.303030 \quad 0.294118 \quad 0.285714 \quad 0.277778 \quad 0.270270 \quad 0.263158 \quad 0.256410 \quad 0.250000\}$ into the variable $[FL]$. Enter $n = 5$ into $[N]$. Then, press [\rightarrow TBL], to get the following table:

	1	2	3	4	5	6
1	0.333333	-0.010752	0.000671	-0.000060	0.000007	-0.000004
2	0.322581	-0.010081	0.000611	-0.000053	0.000003	0.000007
3	0.312500	-0.009470	0.000558	-0.000050	0.000010	-0.000010
4	0.303030	-0.008912	0.000508	-0.000040	0.000000	0.000008
5	0.294118	-0.008404	0.000468	-0.000040	0.000008	-0.000008
6	0.285714	-0.007936	0.000428	-0.000032	0.000000	0.000006
7	0.277778	-0.007508	0.000396	-0.000032	0.000006	0
8	0.270270	-0.007112	0.000364	-0.000026	0	0
9	0.263158	-0.006748	0.000338	0	0	0
10	0.256410	-0.006410	0	0	0	0
11	0.250000	0	0	0	0	0

This table contains the same information as that in Table 4.5, page 131, of the textbook for $f(x)$ and the differences, except for the alignment of the table entries (Table 4.5 is reproduced below). The top row of the table above is recognized as the values $\Delta F = [f_0 \Delta f_0 \Delta^2 f_0 \dots \Delta^n f_0]$, while the values at the bottom of each column, and above the bold-faced zeros, represent $\nabla F = [f_0 \nabla f_0 \nabla^2 f_0 \dots \nabla^n f_0]$. To get those vectors, try pressing [$\rightarrow\Delta F$] and [$\rightarrow\nabla F$]. The results are:

$$\Delta F = [0.333333 \ -0.010752 \ 0.000671 \ -0.000060 \ 0.000007 \ -0.000004]$$

$$\nabla F = [0.250000 \ -0.006410 \ 0.000338 \ -0.000026 \ 0.000006 \ 0.000006]$$

Newton Forward- and Backward-Difference Polynomials

Use sub-directory NF&BP. You will find the following variables:

[INF0][S][FL][N][\rightarrow GFP][\rightarrow GPB]

Pressing [NXT] you will get the following menu:

[CSI][][][][][]

Press [NXT] to get to main menu. Press [INF0] to get an explanation of the sub-directory's operation. The instructions indicate that you should store a value of s into [S], a list of values {fL} into [FL], and the order of the polynomial to be calculated, n , into [N]. Pressing [\rightarrow GFP] will calculate the Newton forward-difference polynomial according to equation (4.67) in page 134 of the textbook. Pressing [\rightarrow GPB] will calculate the Newton backward-difference polynomial according to equation (4.80) in page 137 of the textbook.

The formulas are based on tables of values of x and $f(x)$ such that the values of x are equally spaced by an amount h , i.e., $xL = \{x_a \ x_a+h \ x_a+2h \ \dots \ x_a+(m-1)h\}$. If x is the value at which we want to evaluate the polynomial, then,

$$s = (x-x_0)/h$$

where x_0 represents the first value in the list xL if using the forward-difference polynomial, or the last element of the list xL if using the backward-difference polynomial. (We do not actually use the xL list in the calculations, however).

For example, using the data of table 4.5, page 131 (reproduced below), of the textbook, we could select $x_0 = 3.40$, and use a Newton forward-difference polynomial of order $n = 3$ evaluated at $x = 3.44$. From the table, it follows that $h = 0.1$, therefore, $s = (x-x_0)/h = (3.44-3.40)/0.10=0.4$. For a polynomial of order n we need the to have at least 4 elements. For $x_0 = 3.40$, and a Newton forward-difference polynomial, therefore,

Least-Square Polynomial Fitting

Suppose that n data points (x_i, y_i) can be fitted to a polynomial relationship of the form

$$y = a_0 + a_1x + a_2x^2 + \dots + a_nx^n,$$

Where n is the order of the polynomial. The parameters $a_0, a_1, a_2 \dots a_n$, are the solution to a set of linear equations given, in matrix form, by $\mathbf{M} \cdot \mathbf{b} = \mathbf{c}$, where the vectors \mathbf{b} and \mathbf{c} , and the matrix \mathbf{M} , are defined as follows:

$$\mathbf{b} = [a_0 \ a_1 \ a_2 \ \dots \ a_n]^T; \quad \mathbf{c} = [\Sigma Y \ \Sigma xY \ \Sigma x^2Y \ \dots \ \Sigma x^nY]^T$$

$$\mathbf{M} = \begin{bmatrix} N & \sum x & \dots & \sum x^p \\ \sum x & \sum x^2 & \dots & \\ \vdots & \vdots & \ddots & \vdots \\ \sum x^n & \sum x^{n+1} & \dots & \sum x^{2n} \end{bmatrix}$$

Subdirectory LSPL in your HP48G calculator contains a program [\rightarrow SOL] that calculates the parameters in the vector \mathbf{b} when given the values of x and Y .

To perform a polynomial regression using the programs in this sub-directory use this procedure:

1. Store the values of x and y lists, and the value of n . [\rightarrow DAT] to set up the data matrix necessary to solve for the vector of coefficients $\mathbf{b} = [a_0, a_1, a_2, \dots, a_n]$.
2. Press [\rightarrow SOL] to solve for \mathbf{b} . The display will show the values of the vector of coefficients $\mathbf{b} [a_0, a_1, a_2, \dots, a_n]$, the value of n , and the correlation coefficient r . (Note: the time required for the calculator to perform the regression increases with the value of n , i.e., a regression with $n = 5$ will take much longer than a regression with $n = 2$).
3. Press [\rightarrow PLT] to plot residual errors, e vs. y .
4. Enter a value of x and press [\rightarrow Y] to get $y = a_0 + a_1x + a_2x^2 + \dots + a_nx^n$.
5. Press [NXT] for the next menu.
6. Press [N] to see the number of data points, N .
7. Press [B] to see the vector \mathbf{b} .
8. Press [R] to see the correlation coefficient for the multiple linear regression, r .
9. Press [YBAR] to see the mean value of Y , \bar{Y} .
10. Press [YH] to see the vector \mathbf{y} . This vector contains the regression values corresponding to the original values of the independent variables, i.e., $y_i = a_0 + a_1x_i + a_2x_i^2 + \dots + a_nx_i^n$, for $i = 1, 2, \dots, N$.
11. Press [SSE] to see the error sum of squares, $SSE = \Sigma(Y_i - y_i)$.
12. Press [SST] to see the total sum of squares, $SST = \Sigma(Y_i - \bar{y})$.
13. Press [SE] to see the standard error of estimate, s_e . This value is an estimator of the standard deviation of the distribution of the independent variable x .

Press [NXT][NN] to see the number of points, $N = 9$.

Press [B] to get the vector $\mathbf{b} = [11.93 \ -0.71 \ -0.50 \ 0.13 \ -0.01]$. This means that our regression equation can be written as:

$$y = 11.93 - 0.71x - 0.50x^2 + 0.013x^3 - 0.01x^4.$$

Press [YBAR] to see $\bar{Y} = 8.944$.

Press [YH] to see the list $\mathbf{y} = \{11.93 \ 10.84 \ 9.43 \ 8.21 \ 7.48 \ 7.36 \ 7.78 \ 8.48 \ 8.99 \}$.

Press [SSE] to see the error sum of squares, $SSE = \Sigma(Y_i - y_i) = 1.20$.

Press [NXT][SST] to see the total sum of squares, $SST = \Sigma(y_i - \bar{y}) = 20.72$

Press [SE] to see the standard error of estimate, $s_e = 0.41$.

Press [M] to see the matrix \mathbf{M} .

Press [C] to see the vector \mathbf{c} .

Multivariate Polynomial Approximation

Two examples are shown following: the linear bivariate polynomial: $z = a + bx + cy$, and the quadratic bivariate polynomial, $z = a + bx + cy + dx^2 + ey^2 + fxy$. In both cases the solution for the coefficients (a, b, c, etc.) requires solving a matrix equation of the form $\mathbf{A}\boldsymbol{\xi} = \mathbf{b}$. Methods for the solution of the matrix equation were discussed elsewhere. The main problem will be to build the matrix \mathbf{A} and the vector \mathbf{b} . This could be easily approached, however, by using lists to store the values of x and y, and operating with those lists.

For example, using :

x	1150	1150	1150	1200	1200	1200	1250	1250	1250
y	800	1000	1200	800	1000	1200	800	1000	1200
z	1380.4	1500.2	1614.5	1377.7	1499	1613.6	1375.2	1497.1	1612.6

Where x represents P, y represents T, and z represents C_p .

Create a sub-directory to store this problem, and within that directory store the variables:

```
x = { 1150 1150 1150 1200 1200 1200 1250 1250 1250 }
y = { 800 1000 1200 800 1000 1200 800 1000 1200 }
z = { 1380.4 1500.2 1614.5 1377.7 1499 1613.6 1375.2 1497.1 1612.6 }
```

Then, use operations with lists to calculate the coefficients of the system of linear equations that result from the least-square method for this case:

For example, to calculate N, press:

[X][PRG][LIST][ELEM][SIZE]. The result is $N = 9$. Press [VAR] to recover variables.

To calculate, for example, Σx_i , use:

[X][MTH][LIST][ΣLIST]. The result is $\Sigma x_i = 10800$.

To calculate, $\Sigma x_i y_i$, use:

[VAR][X][Y][×][MTH][LIST][ΣLIST]. The result is $\sum x_i y_i = 10800000$.

To calculate, for example, $\sum x_i^4$, use:

[VAR][X][4][y^x][MTH][LIST][ΣLIST].]. The result is $\sum x_i^4 = 1.87920375E13$.

As a final example, to calculate $\sum x_i^3 y_i$, use:

[VAR][X][3][y^x][Y][×][MTH][LIST][ΣLIST]. The result is $\sum x_i^3 y_i = 1.5606E13$.

Many of the 36 coefficients required in equations (4.132) are repeated, therefore, the number of coefficients to be calculated is 21. The resulting matrix, as shown in page 153, is symmetric.

Data for exercises

Data for application of the methods discussed in this guidebook is available in sub-directory DATA.

Numerical Integration in the HP48G/GX calculator

Use sub-directory NMIN under sub-directory NUMM. The following sub-directories are available:

NFDP: for numerical integration using Newton forward-difference polynomials.
NECO: for numerical integration using Newton-Cotes formulas
GAUS: for numerical integration using Gaussian Quadrature

Numerical integration using Newton forward-difference polynomials

Use sub-directory NFDP. You will find the following variables:

[INFO][H][F][→TRP][→S13][→S38]

Pressing [NXT] you will get the following menu:

[GETF][CS13][CH13][CS38][CH38][]

Press [NXT] to get to main menu. Press [INFO] to get an explanation of the sub-directory's operation. The instructions indicate that you should store the constant x-increment, h, and a list of values of f(x), $\{f\} = \{f_1, f_2, f_3, \dots, f_n\}$ corresponding to $\{x_0, x_1 = x_0+h, x_2 = x_0+2h, \dots, x_n = x_0+nh\}$. The numerical integrals can be calculated using [→TRP] for the trapezoidal rule, [→S13] for the Simpson's 1/3 rule, or [→S38] for the Simpson's 3/8 rule. If you want to calculate the values of f given a function f(x) you can use [GETF] to get the list {f}. The following examples illustrate the use of the different programs.

Trapezoidal rule

Calculate the integral of the function $f(x) = 1/x$, $1 < x < 2$, using the trapezoidal rule. Within sub-directory NFDP press [VAR][NXT][GETF] to get into directory GETF with the purpose of generating the list of values of f(x) to be used in the integration. The variables to be entered are $x_0 = 1$, $x_N = 2$, $h = 0.1$ (arbitrarily selected). Use the following keystrokes:

[1][←][X0] [2][←][XN] [.] [1][←][H]

Next, we need to define the function $f(x) = 1/x$ by using the following keystrokes:

['] [α][←][F] [←][()] [α][←][X] [▶] [←][=] [1][÷] [α][←][X] [ENTER] [←][DEF]

Pressing [↔][F] will show the program that defines the function as: $\ll \rightarrow x \text{ '1/x' } \gg$. Press [DEL] to clear the stack.

Press [→GETF] to get the list of values of f. The results, for this particular case are:

```
3: x: { 1 1.1 1.2 1.3...
2: f: { 1 .909090909...
1: h: .1
```

These results show the value of h in level 1, the f list in level 2, and the corresponding x list in level 3. We need to store the values of h and f in the upper directory NFDP to calculate the integrals. To get to that directory press the key labeled [↑UP]. Next, store the values of h and f as follows:

[←][H] [←][F]

The values of x are not needed. They are only shown to verify the integration limits. You can erase the x list by pressing [DEL]. Next, press [→TRP] to calculate the numerical integral using the trapezoidal rule. The result is 0.69377140317. Check this value against the exact value $\ln(2) = 0.69314718056$. The absolute error is error is

$$\text{Absolute Error} = |\text{Integral} - \text{Exact Value}| = |0.69377140317 - 0.69314718056| = 0.00062422261.$$

The relative error is,

$$\text{Relative Error} = (\text{Absolute Error}/\text{Exact Value}) \cdot 100\% = (0.00062422261/0.69314718056) \cdot 100\% = 0.09\%$$

Simpson's 1/3 Rule

The Simpson's 1/3 rule requires that the number of data points in the list f be odd (or the number of intervals, n, between x_0 and x_N be even). The size of the current f list can be determined by using the following keystrokes:

[F] [PRG] [LIST] [ELEM] [SIZE]

The result is 11, an odd number, therefore, we can use the Simpson's 1/3 rule to calculate the integral with our current data by pressing [VAR][→S13]. The result is 0.69315023069. With Absolute Error = 0.00000305013, and Relative Error = 0.00044%.

To determine the value of h corresponding to a value of n when splitting the interval between x_0 and x_n , use

$$h = (x_n - x_0)/n.$$

Recall that n is the number of intervals for values of x between x_0 and x_n . For the Simpson's 1/3 rule, n must be even. For example, if we want to get 20 intervals (i.e., $n = 20$) between $x_0 = 1$ and $x_N = 2$ for the function $f(x) = 1/x$, we calculate h to be

$$h = (x_n - x_0)/n = (2 - 1)/20 = 0.05.$$

To obtain the f list, use the following:

[NXT][GETF] [1][←][X0] [2][←][XN] [.] [0] [5] [←][H]

The function $f(x)$ has been defined before, therefore, we go directly to calculating {f} by pressing [→GTF]. The results are now:

```
3: x: { 1 1.1 1.2 1.3...
2: f: { 1 .9523809523...
1:           h: .1
```

Press [↑UP] to get back to NFDP, and store the values of h and f:

[←][H] [←][F]

Press [DEL] to clear up the display. Then press [→S13] to calculate the numerical integral using Simpson's 1/3 rule. The result is 0.693147374667.

Note: when using [→S13] if the number of intervals is not even a warning will be shown and the calculation stopped.

Simpson's 3/8 rule

The Simpson's 1/3 rule requires that the number of intervals, n, between x0 and xN be a multiple of three. For example, we can use n = 9 in the integral calculated above by choosing h as

$$h = (x_n - x_0)/n = (2 - 1)/9 = 0.111111111111$$

Within sub-directory NFDP press [VAR][NXT][GETF] and store $h = 0.111111111111$ into [H]. Then, press [→GTF]. The results are now:

```

3: x: { 1 1.111111111111...
2: f: { 1 .900000000000...
1:      h: .111111111111

```

Press [↑UP] to get back to NFDP, and store the values of h and f:

[←][H] [←][F]

Press [DEL] to clear up the display. Then press [→S38] to calculate the numerical integral using Simpson's 3/8 rule. The result is 0.693157302259.

Note: when using [→S38] if the number of intervals is not a multiple of three a warning will be shown and the calculation stopped.

Try pressing [→S13] with the current data. You will get a message box with the warning "Simpson 1/3 Rule: Number of intervals must be even." Press [OK] to clear the message.

Press [←][UP] to get to the upper directory.

Newton-Cotes Formulas

Use sub-directory NECO. You will find the following variables:

[INF0][N][H][F][→INT][GETF]

Pressing [NXT] you will get the following menu:

[α][β][][][][]

Press [NXT] to get to main menu. Press [INF0] to get an explanation of the sub-directory's operation. The instructions indicate that you should store the order of the polynomial (also the number of intervals), n, with $1 \leq n \leq 7$, the constant x-increment, h, and a list of values of f(x), $\{f\} = \{f_1 f_2 f_3 \dots f_n\}$ corresponding to $\{x_0, x_1 = x_0+h, x_2 = x_0+2h, \dots, x_n = x_0+nh\}$. The numerical integrals are calculated using [→INT]. If you want to calculate the values of f given a function f(x) you can use [GETF] to get the list {f}. The following examples illustrate the use of this sub-directory.

Notice that n and h are related by the formula

$$h = (x_n - x_0)/n$$

For example, if we want to integrate, say $f(x) = \exp(x)$, between $x_0 = 0$ and $x_n = 1$, using $n = 5$, then

$$h = (x_n - x_0)/n = (1 - 0)/5 = 0.2$$

In sub-directory NECO, press [GETF] to get into subdirectory GETF, which will be used to generate the list {f} corresponding to this problem. Then use the following keystrokes:

[0][←][X0] [1][←][XN] [5][←][N]

Next, we need to define the function $f(x) = \exp(x)$ by using the following keystrokes:

['] [α][←][F] [←][()] [α][←][X] [▶] [←][=][←][e^x] [α][←][X] [ENTER] [←][DEF]

Pressing [↔][F] will show the program that defines the function as: << → x 'EXP(x)' >>. Press [DEL] to clear the stack.

Press [→GTF] to get the list of values of f. The results, for this particular case are:

```

4: x: { 1 .2 .4 .6 .8...
3: f: { 1 1.221402758...
2:                                     h: .1
1:                                     n: 5

```

These results show the value of n in level 1, h in level 2, the f list in level 3, and the corresponding x list in level 4. We need to store the values of n, h and f in the upper directory NFDP to calculate the integral. To get to that directory press the key labeled [↑UP]. Next, store the values of n, h and f as follows:

[←][N] [←][H] [←][F]

The values of x are not needed. They are only shown to verify the integration limits. You can erase the x list by pressing [DEL]. Next, press [→INT] to calculate the numerical integral using the Newton-Cotes formula (for n = 5, in this case). The result is 1.71828231299. Compared to the exact result, $e-1 = 1.71828182846$, the absolute error is 0.00000048453, and the relative error is 0.0000281%.

Press [←][UP] to get to the upper directory.

Romberg Integration

This algorithm basically uses the trapezoidal rule to calculate the integral for a given value of h, I(h), and for intervals corresponding to h/2, I(h/2), then extrapolates a value of the integral as

$$I(\text{extrapolated}) = I(h/2) + (I(h/2)-I(h))/3.$$

Therefore, to calculate Romberg integration use the [→TRP] in sub-directory NFDP to calculate I(h) and I(h/3). Then, use the formula shown above to calculate I(extrapolated).

For example, above we used the trapezoidal rule to calculate the integral of $f(x) = 1/x$, between $x_0 = 1$ and $x_n = 2$, with $h = 0.1$. The result was $I(h) = 0.69377140317$. We can then calculate $I(h/2)$ by using $h = 0.05$, in which case $I(h/2) = 0.69330338179$. The Romberg integration result will, therefore, be

$$I(\text{extrapolated}) = I(h/2) + (I(h/2)-I(h))/3 = 0.69330338179 + (0.69330338179-0.69377140317)/3,$$

or

$I(\text{extrapolated}) = 0.693147374663.$

The absolute error is 0.000000194103, and the relative error is 0.000028%.

Gaussian Quadrature

Use sub-directory GAUS. You will find the following variables:

[INF0][A][B][F][→GS2][→GS3]

Pressing [NXT] you will get the following menu:

[→GS4][→TC][C][M][→MC][F]

Press [NXT] to get to main menu. Press [INF0] to get an explanation of the sub-directory's operation. The instructions indicate that you should store the limits of integration $a = x_o$, and $b = x_n$, and define the function $f(x)$ using the [DEF] key. The numerical integrals are calculated using [→GS2], [→GS3], and [NXT][→GS4], for the Gaussian quadrature formulas corresponding to $n = 2, 3,$ and $4,$ respectively. If you want to calculate the values of f given a function $f(x)$ you can use [GETF] to get the list $\{f\}$. The following example illustrates the use of this sub-directory.

To calculate the integral of $f(x) = 1/x$, between $a = 1$ and $b = 2$, store the values of a and b using:

[1][←][A] [2][←][B].

Then, define $f(x) = 1/x$, by using:

['] [α][←][F] [←][()] [α][←][X] [▶] [←][=] [1][÷] [α][←][X] [ENTER] [←][DEF]

Pressing [↵][F] will show the program that defines the function as: $\ll \rightarrow x \text{ '1/x' } \gg$. Press [DEL] to clear the stack.

Press [→GS2] to get the 2nd-order Gaussian quadrature integral, the result is 0.692307692305.

Press [→GS3] to get the 3rd-order Gaussian quadrature integral, the result is 0.69312169312.

Press [NXT][→GS4] to get the 4th-order Gaussian quadrature integral, the result is 0.693146417445.

The exact result is $\ln(2) = 0.69314718056.$
