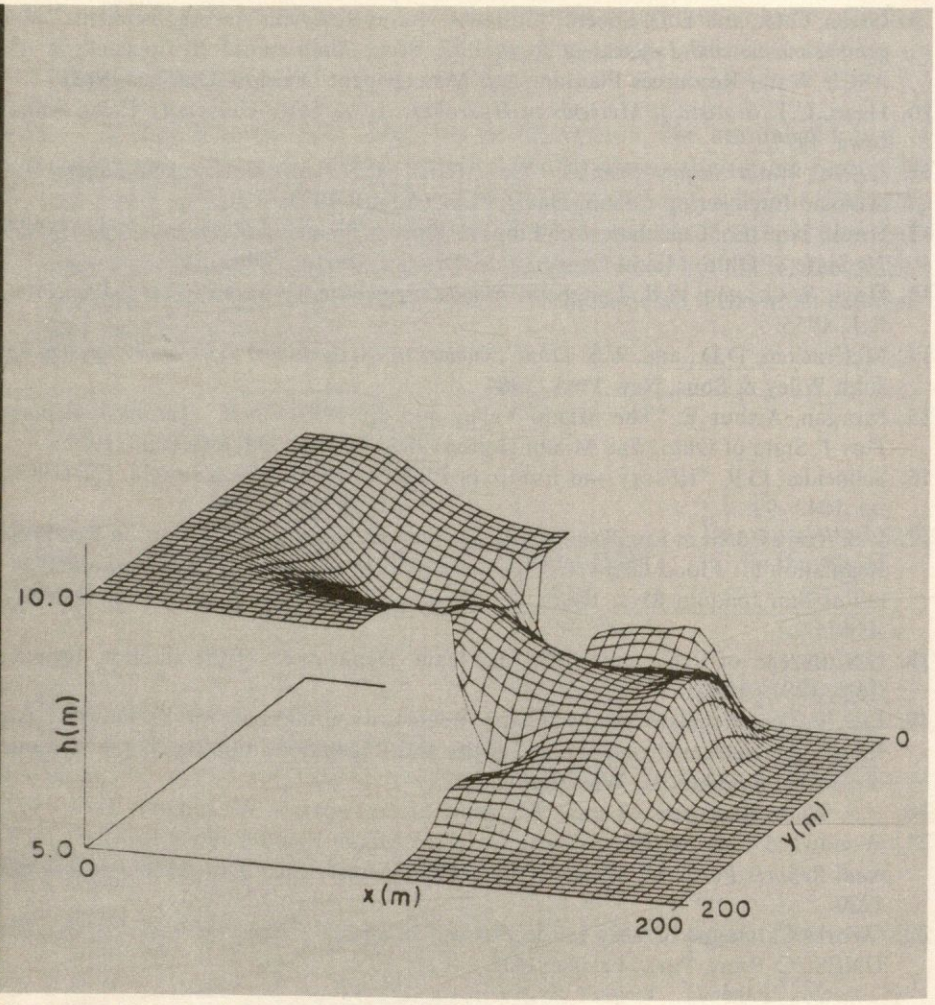
CLEVELAND

10



Plot of computed depths for two-dimensional dam break flow (Courtesy of Robert J. Fennema)

Mathematical Modeling of Hydraulic Systems

We discussed traditional methods of analysis in Chapters 1 through 9 and introduced physical models in Chapter 7. In this chapter, we will present a description of mathematical models and illustrate their use as a design tool. Most of the analysis procedures discussed in this and the following chapters are suitable for digital computer solutions.

We first outline the steps involved in the development of a mathematical model. Then we present a number of numerical analysis procedures. For illustration, we include the application of these procedures for the solution of several typical hydraulic problems.

10-1 Mathematical Models

A *mathematical model* is a representation of the behavior of a particular system in the form of mathematical equations. For specified parameters, the system response may be determined from this model. Though usually used as a design tool, these models may also be used for real-time control or operation of the system. For example, the mathematical model of a municipal pipe network may be used for its design, for investigating the effects of various modifications, and for optimal operation of pumps, reservoirs, and so forth. If certain components of a system fail, necessary remedial measures and operational strategies may be decided fairly quickly thus making the "let us do this to see what happens" approach unnecessary.

Usually, the following steps are involved in the development of a mathematical model:

1. Derivation of governing equations
2. Selection of solution procedures
3. Development and verification of computer codes

We briefly describe each of these steps in the following paragraphs.

Derivation of Governing Equations

The "physics" of the phenomenon has to be first understood to derive the governing equations. In this derivation assumptions are made so that the resulting equations are simple but adequately describe the phenomenon. If equations are too complex, they may not be easily amenable to a solution. For example, if the transverse and vertical components of the flow velocity are small, they may be neglected, thereby resulting in a one-dimensional model.

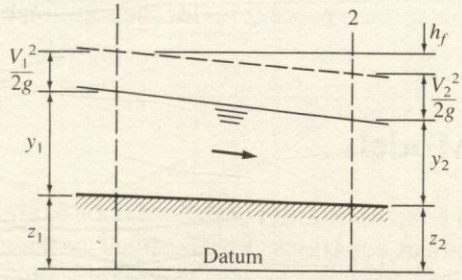
Governing equations are usually derived by applying the well-known laws of mechanics, such as conservation of mass, momentum, and energy. For simplification, some of the variables may be eliminated by combining different equations. By using reference variables, the governing equations may be nondimensionalized.

Table 10-1 Typical Governing Equations in Hydraulic Engineering

Algebraic Equations

Single equation — energy equation between two channel sections:

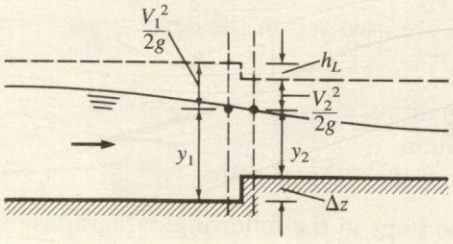
$$z_1 + y_1 + \frac{V_1^2}{2g} = z_2 + y_2 + \frac{V_2^2}{2g} + h_f$$



Two equations — energy and continuity equations at a channel junction:

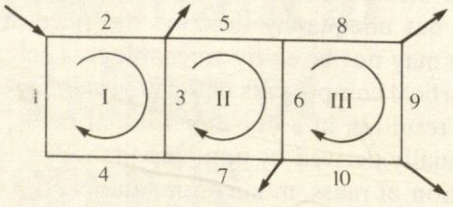
$$y_1 + \frac{V_1^2}{2g} = \Delta z + y_2 + \frac{V_2^2}{2g} + h_e$$

$$A_1 V_1 = A_2 V_2$$



System of equations — Kirchhoff laws for a pipe network:

$$\sum_{i=1}^{n} h_{f_i}^j = 0$$



where $h_{f_i}^j = c_i Q_i |Q_i|$ = head loss in the ith pipe of jth loop. Similar equations may be written for the other loops.

Ordinary Differential Equations

Single equation — Equation for the variation of flow depth in gradually varied flow:

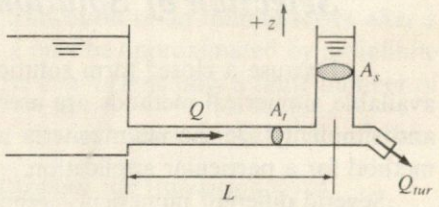
$$\frac{dy}{dx} = \frac{S_0 - S_f}{1 - (\mathrm{Fr})^2}$$

where Fr = Froude number, S_0 = slope of channel bottom, and S_f = slope of the energy grade line.

Two equations — water-level oscillations in a surge tank:

$$\frac{dz}{dt} = \frac{Q - Q_{\mathrm{tur}}}{A_s}$$

$$\frac{dQ}{dt} = \frac{gL}{A_t}(-z - cQ|Q|)$$



where Q_{tur} = turbine flow, and c = coefficient of head losses in the tunnel.

Partial Differential Equations

Hyperbolic — unsteady flow in pipes:

$$\frac{\partial H}{\partial t} + \frac{a^2}{gA}\frac{\partial Q}{\partial x} = 0$$

$$\frac{\partial Q}{\partial t} + gA\frac{\partial H}{\partial x} + \frac{fQ|Q|}{2DA} = 0$$

where H = piezometric head, Q = discharge, a = wave speed, A = pipe cross-sectional area, D = pipe diameter, and f = Darcy-Weisbach friction factor.

Parabolic — diffusion of a tracer:

$$\frac{\partial c}{\partial t} = D\frac{\partial^2 c}{\partial x^2}$$

where c = concentration of the tracer, and D = molecular diffusion coefficient.

Elliptic — stream function for irrotational flow:

$$\frac{\partial^2 \psi}{\partial x^2} + \frac{\partial^2 \psi}{\partial y^2} = 0$$

where ψ = stream function.

The governing equations may be algebraic, ordinary differential equations, or partial differential equations, depending on the phenomenon under consideration. Physical phenomena representing the rate of change of one or more variables with respect to other variables are described by differential equations. If the dependent variables are functions of only one independent variable, then the resulting equations are ordinary differential equations. However, if the dependent variables are functions of more than one independent variable, the resulting equations are partial differential equations.

Table 10-1 presents typical examples of different types of equations.

Selection of Solution Procedures

Because a closed-form solution of nonlinear equations is not usually available, numerical methods are used for their solution. Accuracy, efficiency, and simplicity are the main criteria used during the selection of a numerical method for a particular application.

Several different numerical methods are available for the solution of governing equations depending on their type. A list of these methods follows (1, 4):

Algebraic equations
- Linear equations: Gauss elimination, Gauss-Seidel methods
- Nonlinear equations: Newton-Raphson method, method of successive approximations, bisection method

Ordinary differential equations
- Single-step methods: Euler, Improved Euler, Runge-Kutta
- Multistep methods: Predictor-corrector, Adam-Bashforth-Moulton

Partial differential equations
- Method of characteristics, Finite-difference methods, Finite-element method, Boundary-integral method, Spectral method, Pseudo-Spectral method

Development and Verification of Computer Codes

A computer code may be divided into three parts: reading the input data, specifying the solution algorithm, and printing the computed results and preparing the necessary plots. The input data give the system parameters and may specify the layout for the program output.

Once a computer program has been developed and its computed results appear reasonable, it is necessary to verify their validity by comparing them with some reliable data. These data may be the exact solution for idealized and simplified cases, or they may be obtained from laboratory and field experiments. The experimental accuracy of the field or laboratory results should be consid-

ered while doing these comparisons. Moreover, the code should be verified over the complete range of system parameters.

10-2 Errors in Numerical Analysis*

To confidently use the computed results, it is usually necessary to have an estimate of the errors. Two types of errors are introduced into the numerical computations: truncation errors and roundoff errors. Superficially, these errors may appear small, but in repetitive calculations, they may grow and mask the actual results.

A *truncation error* is caused by the truncation of an infinite series after a finite number of terms. For example, $\sin x$ may be approximated by an infinite series. For actual calculations, however, we can include only a finite number of terms; for example, $\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!}$, in which terms higher than x^5 have been neglected. These neglected terms introduce the truncation error.

A *roundoff error* is introduced into the computations because irrational numbers are represented by a finite number of digits in the computer. For example, 4/3 will be represented as 1.333333 in machine computations involving seven significant digits.

10-3 Interpolations

It often is necessary to interpolate results from tabulated data. For example, the discharge through a spillway may be determined at a number of water levels by using hydraulic models. The flows at intermediate water levels may then be interpolated from the measured data. Depending on the curve selected to describe the relationship between the dependent and independent variables, these interpolations may be linear or higher order.

In this section we will discuss *parabolic interpolation*. Let us assume we have values of y at equal intervals of x, and we want to interpolate the value of y at an intermediate value of x such that $x_i < x < x_{i+1}$ (Fig. 10-1, page 546). Let us have new coordinate axes, x', y', with the origin at point B, having coordinates (x_i, y_i), and pass a parabola through points A, B, and C, having the coordinates (x_{i-1}, y_{i-1}), (x_i, y_i), and (x_{i+1}, y_{i+1}), respectively. The equation of this parabola in terms of the new coordinates, x' and y', will be

$$y' = ax' + bx'^2 \tag{10-1}$$

* For more detailed information about numerical analysis, see Chapra (1) and McCracken (4).

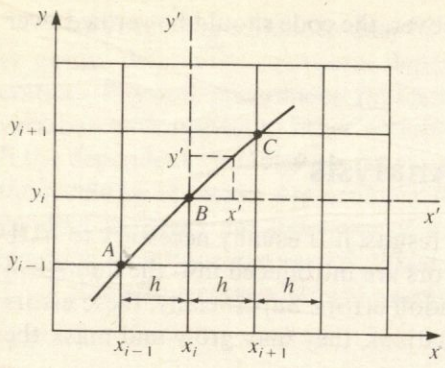


Figure 10-1 Parabolic interpolation

By substituting into this equation the coordinates of points A and C, (x_{i-1}, y_{i-1}) and (x_{i+1}, y_{i+1}), denoting the interval of x at which values of y are specified as h ($h = x_{i+1} - x_i = x_i - x_{i-1}$), and noting that $y' = y - y_i$, we obtain

$$y_{i-1} - y_i = -ah + bh^2 \tag{10-2}$$

$$y_{i+1} - y_i = ah + bh^2 \tag{10-3}$$

Simultaneous solution of these equations for a and b yields

$$a = \frac{1}{2}\frac{y_{i+1} - y_{i-1}}{h}$$

$$b = \frac{1}{2}\frac{y_{i+1} - 2y_i + y_{i-1}}{h^2} \tag{10-4}$$

By substituting into Eq. (10-1), these expressions for a and b and $r = (x - x_i)/h$, and writing x' and y' in terms of x and y, we obtain

$$y = y_i + \frac{1}{2}r[y_{i+1} - y_{i-1} + r(y_{i+1} + y_{i-1} - 2y_i)] \tag{10-5}$$

This equation may be used to interpolate values of y from the given coordinates of points A, B, and C. *Note*: This equation is valid only for equally spaced grid points.

Figure 10-2 lists a subroutine for parabolic interpolation using this expression.

EXAMPLE 10-1 The following table lists the flows over a spillway for different reservoir levels. Determine the flows at water-level elevations of 102 m, 114 m, 123 m, and 138 m.

Spillway Rating Curve	
Reservoir level (m)	Discharge (m^3/s)
100	0.0
105	33.5
110	96.6
115	178.6
120	277.0
125	393.7
130	519.3
135	656.2
140	804.5

SOLUTION The interval at which spillway flows are stored for different reservoir levels is 5 m. We will write a short program to read the following input data: reservoir elevation, corresponding flows, and the reservoir levels at which flows have to be interpolated. We will use subscripted arrays for these variables and a subroutine for parabolic interpolation. Input to the subroutine is through its argument list. For each reservoir level, the flows are interpolated in the subroutine from the stored data, then they are printed in the main program. Figure 10-2 lists the computer program, its input data, and program output. ■

Figure 10-2 Program for parabolic interpolation

Program Listing

```
C     PROGRAM FOR PARABOLIC INTERPOLATION
C
C     **** NOTATION  ****
C
C     DX = INTERVAL FOR STORING INDEPENDENT VARIABLE
C     N  = NUMBER OF DATA POINTS TO BE STORED
C     NI = NUMBER OF INTERMEDIATE VALUES OF X AT WHICH Y IS TO
C          BE INTERPOLATED
C     X  = ARRAY FOR INDEPENDENT VARIABLE
C     XI = ARRAY OF INTERMEDIATE VALUES OF X AT WHICH Y IS TO BE
C          INTERPOLATED
C     Y  = ARRAY OF DEPENDENT VARIABLE
C
      DIMENSION X(50),Y(50),XI(50)
      READ (5,*) N,DX,(X(I), I=1,N)
      READ (5,*) (Y(I), I=1,N)
      WRITE (6,20)
20    FORMAT(5X, 'X',8X,'Y')
      DO 30 I=1,N
      WRITE (6,25) X(I),Y(I)
25    FORMAT(F8.2,2X,F8.2)
30    CONTINUE
      READ(5,*) NI, (XI(I),I=1,NI)
      WRITE (6,40)
40    FORMAT(/5X,'INTERPOLATED VALUES'/)
      WRITE(6,42)
42    FORMAT(5X,'XI',10X,'YI')
      DO 50 I = 1,NI
      XINT=XI(I)
      CALL INT(XINT,X,Y,DX,YI)
      WRITE(6,45) XI(I),YI
45    FORMAT(2F10.2)
```

(continued)

Figure 10-2 (continued)

```
50    CONTINUE
      STOP
      END
      SUBROUTINE INT(XI,X,Y,DX,YI)
      DIMENSION X(50),Y(50)
      J=1+(XI-X(1))/DX
      R=(XI-X(J))/DX
      IF (J.EQ.1) R=R-1.
      IF (J.LT.2) J=2
      YI=Y(J)+0.5*R*(Y(J+1)-Y(J-1)+R*(Y(J+1)+Y(J-1)-2.*Y(J)))
      RETURN
      END
```

Input Data

```
9,5.,100.,105.,110.,115.,120.,125.,130.,135.,140.
0.0,33.5,96.6,178.6,277.,393.7,519.3,656.2,804.5
4,102.,114.,123.,138.
```

Program Output

```
     X          Y
100.00          .00
105.00        33.50
110.00        96.60
115.00       178.60
120.00       277.00
125.00       393.70
130.00       519.30
135.00       656.20
140.00       804.50

   INTERPOLATED VALUES

  XI           YI
102.00         9.85
114.00       160.69
123.00       344.82
138.00       743.81
```

10-4 Nonlinear Algebraic Equations

Two commonly used methods for determining the roots of nonlinear algebraic equations are the bisection method and the Newton-Raphson method. Details of these methods are given in the following paragraphs.

Bisection Method

Assume that we have to determine the roots of the equation

$$F(x) = 0 \tag{10-6}$$

The x-coordinates of points where the graph of $F(x)$ versus x intersects the x-axis are the roots of Eq. (10-6). We will outline a step-by-step procedure for determining one of the roots; similarly, other roots may be determined one by one.

1. Estimate two values of x, namely, x_n and x_p, so that $F(x_n)$ is negative and $F(x_p)$ is positive (see Fig. 10-3).

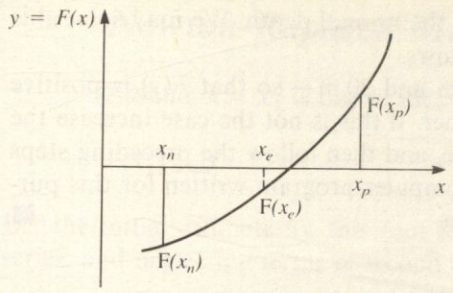


Figure 10-3 Bisection method

2. Compute $x_e = \frac{1}{2}(x_n + x_p)$.
3. Determine $F(x_e)$. If $|F(x_e)| < \epsilon$, where ϵ is specified tolerance, then x_e is the root of Eq. (10-6); otherwise, set $x_n = x_e$ if $F(x_e) < 0$, and set $x_p = x_e$ if $F(x_e) > 0$.
4. Repeat steps 2 and 3 until a solution is obtained.

Convergence to the solution in this method is usually assured although it is usually slower than that in the Newton-Raphson method, which we discuss in the next section.

In all iterative procedures, it is always a good idea to introduce a counter to avoid an unlimited number of iterations due to errors in logic, computation, programming, or other reasons. Iterations may be stopped if the number exceeds a specified number, say 50.

EXAMPLE 10-2 Use the bisection method to determine the normal depth in a trapezoidal channel carrying a flow of 110 m^3/s. The channel bottom slope is 0.0001, Manning's $n = 0.013$, channel bottom width is 20 m, and the side slopes of the channel are 2 horizontal to 1 vertical.

SOLUTION To determine the normal depth, we have to solve Manning's equation. In this case, we use the SI version:

$$Q = \frac{1}{n} AR^{2/3} S_0^{1/2} \tag{10-7}$$

where Q is the rate of discharge, n is Manning's constant, A is the flow area, R is the hydraulic radius, and S_0 is the slope of the channel bottom. Since n, Q, and S_0 are specified, and both A and R are functions of flow depth y, we can write Eq. (10-7) as

$$F(y) = Q - \frac{1}{n} AR^{2/3} S_0^{1/2} = 0 \tag{10-8}$$

The value of y that satisfies Eq. (10-8) is the normal depth. We may solve this equation by the bisection method as follows.

Let us select two values of y—0.2 m and 20 m—so that $F(y)$ is positive for one of them and negative for the other. If this is not the case increase the higher value, or decrease the lower value, and then follow the preceding steps to solve Eq. (10-8). Figure 10-4 lists a computer program written for this purpose, its input data, and program output. ■

Figure 10-4 Program for computing normal depth using the bisection method

Program Listing

```
C     COMPUTATION OF NORMAL DEPTH IN A TRAPEZOIDAL CHANNEL BY
C       USING BISECTION METHOD
C
C     **********************NOTATION ************************
C
C     AN = MANNING'S N;
C     BO = CHANNEL-BOTTOM WIDTH;
C     Q = DISCHARGE;
C     S = SLOPE OF CHANNEL SIDES, S:HORIZONTAL TO 1 VERTICAL;
C     SO = CHANNEL BOTTOM SLOPE;
C     Y = NORMAL DEPTH;
C     YP, YN = DEPTH ESTIMATES SUCH THAT YP<Y<YN
C
      AR(Y)=(BO+S*Y)*Y
      P(Y)=BO+2.*Y*SQRT(1.+S*S)
      READ(5,*) AN,Q,SO,S,BO,YP,YN
      WRITE(6,10)AN,Q,SO,S,BO
10    FORMAT(5X,'N ='F5.3,3X,'Q =',F8.3,' M3/S',3X,'SO =',F6.4,
     1 3X,'S =',F4.2,3X,'BO =',F6.2,' M')
      WRITE(6,15) YP,YN
15    FORMAT(/5X,'INITIAL ESTIMATED FLOW DEPTHS:',2X, 'YP =',
     1   F4.2,' M',3X,'YN =',F6.2,' M')
      K = 0
20    Y=0.5*(YP+YN)
      K=K+1
      IF (K.GT.50) GO TO 60
      F=Q-(AR(Y)**1.667*SQRT(SO))/(AN*P(Y)**0.667)
      IF (F.LT.0.0) YN=Y
      IF (F.GT.0.0) YP=Y
      IF (ABS(YP-YN).LE.0.001) GO TO 40
      GO TO 20
40    Y=0.5*(YP+YN)
      WRITE(6,50) Y
50    FORMAT(/5X, 'NORMAL DEPTH =',F6.3,' M')
      GO TO 80
60    WRITE(6,70)
70    FORMAT(10X,'ITERATIONS FAILED')
80    STOP
      END
```

Input Data

```
0.013,110.,0.0001,2.,20.,0.2,20.
```

Program Output

```
N = .013   Q = 110.000 M3/S   SO = .0001   S =2.00   BO = 20.00 M

INITIAL ESTIMATED FLOW DEPTHS:  YP = .20 M   YN = 20.00 M

NORMAL DEPTH = 3.069 M
```

Newton-Raphson Method

Assume $x = x_1$ is one of the roots of Eq. (10-6), that is,

$$F(x_1) = 0 \tag{10-9}$$

Let the initial estimate for this root be x_0. Expanding Eq. (10-9) in a Taylor series, and neglecting terms of second and higher order, we obtain

$$F(x_1) = F(x_0) + (x_1 - x_0)F'(x_0) = 0 \tag{10-10}$$

where $F'(x_0)$ denotes dF/dx evaluated at $x = x_0$. However, since the initial estimate, x_0, may not be very close to the root of Eq. (10-6), an iterative procedure is used to refine the solution. Let us call x_1 determined from Eq. (10-10) as x_n. Then, from Eq. (10-10), it follows that

$$x_n = x_0 - \frac{F(x_0)}{F'(x_0)} \tag{10-11}$$

If $|(x_n - x_0)| < \epsilon$, where ϵ is the specified tolerance, then x_n is the root of Eq. (10-6). Otherwise, set $x_0 = x_n$, and repeat the above procedure until a solution is obtained.

Figure 10-5 shows a geometrical representation of the convergence of the iterative process.

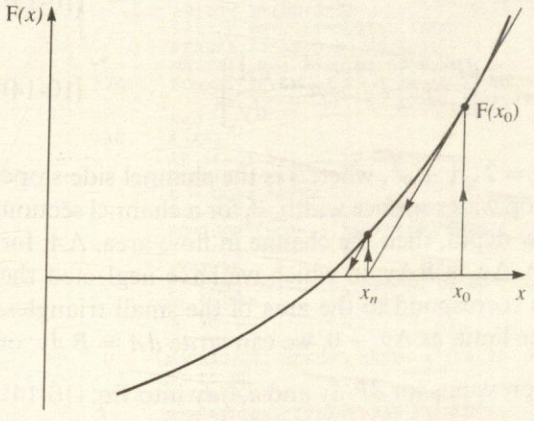


Figure 10-5 Newton-Raphson method

The disadvantages of the Newton-Raphson method are

1. It is necessary to determine the first derivative of the function F, and an expression for this may not be easily available. The value of the derivative

in such cases may be obtained by using numerical methods, such as secant method.
2. The procedure fails if $dF/dx = 0$.
3. The iterations may diverge if $F''(x)$ becomes too large.
4. It is difficult to apply if a table of values is given instead of a mathematical expression.

The advantages of the method are

1. It can be easily extended to a system of two or more equations.
2. It converges fast, thereby reducing the required number of iterations.

EXAMPLE 10-3 Solve Example 10-2 using the Newton-Raphson method.

SOLUTION To use the Newton-Raphson method, we need the derivative of function F. This may be obtained by differentiating Eq. (10-8) with respect to y as follows:

$$\frac{dF}{dy} = \frac{d}{dy}\left[Q - \frac{1}{n} AR^{2/3} S_0^{1/2}\right] \tag{10-12}$$

Since we are determining the normal depth for specified values of Q, n, and S_0, these are constants for differentiation purposes. In addition, $R = A/P$. Hence we may write this equation as

$$\frac{dF}{dy} = -\frac{S_0^{1/2}}{n}\frac{d}{dy}(A^{5/3}P^{-2/3}) \tag{10-13}$$

$$= -\frac{S_0^{1/2}}{n}\left[-\frac{2}{3}A^{5/3}P^{-5/3}\frac{dP}{dy} + \frac{5}{3}P^{-2/3}A^{2/3}\frac{dA}{dy}\right] \tag{10-14}$$

For a trapezoidal channel, $dP/dy = 2\sqrt{1 + s^2}$, where s is the channel side slope (s horizontal to 1 vertical). If the top water surface width, B, for a channel section is continuous with change in flow depth, then the change in flow area, ΔA, for a small change in the flow depth, Δy, is $B\,\Delta y$, in which we have neglected the higher-order terms. (These terms correspond to the area of the small triangles, shown shaded in Fig. 10-6.) In the limit, as $\Delta y \to 0$, we can write $dA = B\,dy$, or $\dfrac{dA}{dy} = B$. By substituting these expressions for dP/dy and dA/dy into Eq. (10-14), and by simplifying the resulting equation, we obtain

$$\frac{dF}{dy} = \frac{S_0^{1/2}}{3n}[4R^{5/3}\sqrt{1 + s^2} - 5BR^{2/3}] \tag{10-15}$$

A computer program was developed using this relationship for $F'(y)$. The initial estimate for the normal depth was specified as 1 m, and the tolerance

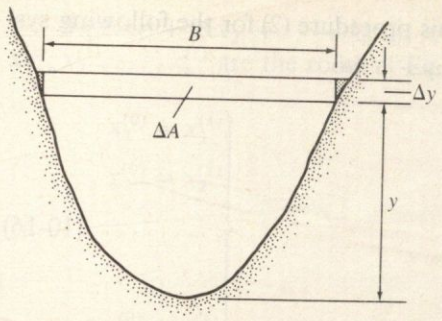


Figure 10-6 Definition sketch

for the convergence of the iterations as 0.001 m. Figure 10-7 lists the computer program, its input data, and program output. ■

Figure 10-7 Program for computing normal depth using Newton-Raphson method

Program Listing

```
C      COMPUTATION OF NORMAL DEPTH IN A TRAPEZOIDAL CHANNEL
C       USING NEWTON-RAPHSON METHOD
C
C      ******************** NOTATION ******************
C
C      AN= MANNING'S N;
C      BO = CHANNEL-BOTTOM WIDTH;
C      Q = DISCHARGE;
C      S = SLOPE OF CHANNEL SIDES, S: HORIZONTAL TO 1 VERTICAL;
C      SO = CHANNEL-BOTTOM SLOPE;
C      Y = NORMAL DEPTH;
C      YI = INITIAL ESTIMATE FOR NORMAL DEPTH.
C
       AR(Y)= Y*(BO+S*Y)
       P(Y) = BO+2.*Y*SQRT(1.+S*S)
       READ(5,*) AN,Q,BO,SO,S,YI
       WRITE(6,20) AN,Q,BO,SO,S,YI
20     FORMAT(5X,'N =',F5.3,3X,'Q =',F7.3,' M3/S',3X,'BO =',
     1    F5.2,' M',3X,'SO =',F6.4,3X,'S =',F4.2,3X,'YI =',F5.2)
       K=0
30     K=K+1
       IF (K.GT.50) GO TO 80
       R=AR(YI)/P(YI)
       B = BO+2.*S*YI
       F=Q-(AR(YI)*R**0.667*SQRT(SO))/AN
       FD=(SQRT(SO)/(3.*AN))*(4.*SQRT(1.+S*S)*R**1.667
     1    - 5.*B*R**0.667)
       DY=F/FD
       IF (ABS(DY).LE.0.001) GO TO 60
       YI=YI-DY
       GO TO 30
60     WRITE(6,70) YI
70     FORMAT(5X,'NORMAL DEPTH =',F6.2,' M')
       GO TO 90
80     WRITE(6,85)
85     FORMAT(10X,'ITERATIONS FAILED')
90     STOP
       END
```

Input Data

```
0.013,110.,20.,0.0001,2.,1.
```

Program Output

```
     N = .013   Q =110.000 M3/S   BO =20.00 M   SO = .0001
S =2.00   YI = 1.00
     NORMAL DEPTH =  3.07 M
```

Let us discuss how we may extend this procedure (2) for the following system of n equations in n unknowns, $x_1, x_2, \ldots, x_n$:

$$\left.\begin{aligned} &F_1(x_1, x_2, \ldots, x_n) = 0 \\ &F_2(x_1, x_2, \ldots, x_n) = 0 \\ &\cdots\cdots\cdots\cdots\cdots\cdots \\ &\cdots\cdots\cdots\cdots\cdots\cdots \\ &F_n(x_1, x_2, \ldots, x_n) = 0 \end{aligned}\right\} \tag{10-16}$$

Let the initial estimate for the roots be $x_1^{(0)}, x_2^{(0)}, \ldots, x_n^{(0)}$, where the superscript indicates the number of the iteration—0 for initial estimate, 1 for values obtained after one iteration, and so on. Similar to the single equation, Eq. (10-6), we may expand and rearrange Eq. (10-16) as

$$\begin{bmatrix} \frac{\partial F_1}{\partial x_1} & \frac{\partial F_1}{\partial x_2} & \cdots & \frac{\partial F_1}{\partial x_j} & \cdots & \frac{\partial F_1}{\partial x_n} \\ \frac{\partial F_2}{\partial x_1} & \frac{\partial F_2}{\partial x_2} & \cdots & \frac{\partial F_2}{\partial x_j} & \cdots & \frac{\partial F_2}{\partial x_n} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \frac{\partial F_i}{\partial x_1} & \frac{\partial F_i}{\partial x_2} & \cdots & \frac{\partial F_i}{\partial x_j} & \cdots & \frac{\partial F_i}{\partial x_n} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \frac{\partial F_n}{\partial x_1} & \frac{\partial F_n}{\partial x_2} & \cdots & \frac{\partial F_n}{\partial x_j} & \cdots & \frac{\partial F_n}{\partial x_n} \end{bmatrix}^{(0)} \begin{Bmatrix} \Delta x_1 \\ \Delta x_2 \\ \vdots \\ \Delta x_i \\ \vdots \\ \Delta x_n \end{Bmatrix} = - \begin{Bmatrix} F_1 \\ F_2 \\ \vdots \\ F_i \\ \vdots \\ F_n \end{Bmatrix}^{(0)} \tag{10-17}$$

where the functions and their partial derivatives are evaluated at $x_1^{(0)}, x_2^{(0)}, \ldots, x_n^{(0)}$. Then after solving for the Δx_i's, a better estimate for the solution is

$$\left.\begin{aligned} x_1^{(1)} &= x_1^{(0)} + \Delta x_1 \\ x_2^{(1)} &= x_2^{(0)} + \Delta x_2 \\ &\cdots\cdots\cdots\cdots \\ &\cdots\cdots\cdots\cdots \\ x_n^{(1)} &= x_n^{(0)} + \Delta x_n \end{aligned}\right\} \tag{10-18}$$

If $(|\Delta x_1| + |\Delta x_2| + \cdots + |\Delta x_n|) < \epsilon$, where ϵ is the specified tolerance, then $x_1^{(1)}, x_2^{(1)}, \ldots, x_n^{(1)}$ are the roots of Eq. (10-16); otherwise, set

$$\left.\begin{aligned} x_1^{(0)} &= x_1^{(1)} \\ x_2^{(0)} &= x_2^{(1)} \\ &\cdots\cdots\cdots \\ &\cdots\cdots\cdots \\ x_n^{(0)} &= x_n^{(1)} \end{aligned}\right\} \tag{10-19}$$

Repeat the above procedure until a solution is obtained.

10-5 Quadrature

We want to numerically determine the value of the integral

$$I = \int_a^b f(x)\,dx \tag{10-20}$$

Evaluating this integral is equivalent to determining the area, I, under the curve $y = f(x)$ and the x axis between $x = a$ and $x = b$ (Fig. 10-8). To evaluate this integral numerically, we divide the interval between $x = a$ and $x = b$ into a number of subintervals, approximately compute the area for each subinterval, and then sum these subareas to determine the total area. Let us divide the interval $x = a$ to $x = b$ into N equal subintervals of length, h:

$$h = \frac{b - a}{N} \tag{10-21}$$

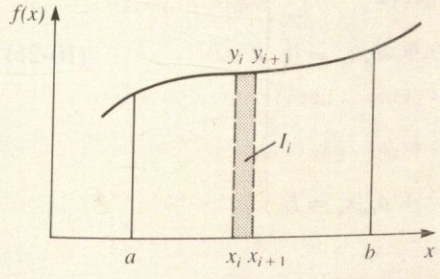


Figure 10-8 Numerical evaluation of integrals

If we consider each subarea as a trapezoid, the area of the trapezoid between $x = x_i$ and $x = x_{i+1}$ is

$$I_i = 0.5h(y_i + y_{i+1}) \tag{10-22}$$

Adding these N subareas, we obtain

$$I = I_h = \sum I_i = \frac{h}{2}(y_1 + 2y_2 + \cdots + 2y_N + y_{N+1}) \tag{10-23}$$

This is referred to as the *trapezoidal rule*, in which the truncation error, $e_t < (h^2/12)(b - a)$, and the roundoff error, e_r, is proportional to $1/h$. Theoretically, $I_h \to I$ as $h \to 0$. In reality, however, this is not the case, since the roundoff error becomes dominant as h becomes smaller and smaller.

If N is an even number, we may use *Simpson's rule* to evaluate the integral, I_h:

$$I_h = \frac{h}{3}(y_1 + 4y_2 + 2y_3 + 4y_4 + \cdots + 2y_{N-1} + 4y_N + y_{N+1}) \tag{10-24}$$

In this rule, the truncation error is proportional to h^4, and the roundoff error is proportional to $1/h$.

10-6 Linear Algebraic Equations

The Gauss elimination technique may be used to solve a system of independent linear equations (4). Assume that we have n equations in n unknowns $x_1, x_2, \ldots, x_n$ and that we have rearranged the equations, if necessary, so that $a_{11} \neq 0$.

$$\left.\begin{aligned}
&a_{11}x_1 + a_{12}x_2 + \cdots + a_{1j}x_j + \cdots + a_{1n}x_n = b_1 \\
&a_{21}x_1 + a_{22}x_2 + \cdots + a_{2j}x_j + \cdots + a_{2n}x_n = b_2 \\
&\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\
&a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{ij}x_j + \cdots + a_{in}x_n = b_i \\
&\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\
&\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\
&a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nj}x_j + \cdots + a_{nn}x_n = b_n
\end{aligned}\right\} \tag{10-25}$$

Let us define $n - 1$ multipliers

$$m_i = \frac{a_{i1}}{a_{11}} \qquad i = 2, 3, \ldots, n \tag{10-26}$$

Subtract m_i times the first equation from the ith equation, and let

$$\begin{aligned} a'_{ij} &= a_{ij} - m_i a_{1j} \qquad i = 2, 3, \ldots, n \quad \text{and} \quad j = 1, 2, \ldots, n \\ b'_i &= b_i - m_i b_1 \end{aligned} \tag{10-27}$$

Then, the resulting equations can be written as

$$\left.\begin{array}{l} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1j}x_j + \cdots + a_{1n}x_n = b_1 \\ \quad 0 \quad + a'_{22}x_2 + \cdots + a'_{2j}x_j + \cdots + a'_{2n}x_n = b'_2 \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ \quad 0 \quad + a'_{n2}x_2 + \cdots + a'_{nj}x_j + \cdots + a'_{nn}x_n = b'_n \end{array}\right\} \tag{10-28}$$

If we repeat this procedure, then after the kth time, we have eliminated x_k by defining multipliers

$$m_i^{(k-1)} = \frac{a_{ik}^{(k-1)}}{a_{kk}^{(k-1)}} \qquad i = k+1, \ldots, n \tag{10-29}$$

where $a_{kk}^{(k-1)} \neq 0$. And

$$\begin{aligned} a_{ij}^{(k)} &= a_{ij}^{(k-1)} - m_i^{(k-1)} a_{kj}^{(k-1)} \\ b_i^{(k)} &= b_i^{(k-1)} - m_i^{(k-1)} b_k^{(k-1)} \\ & i = k+1, \ldots, n \qquad j = k, \ldots, n; \quad \text{and} \quad k = 1, 2, \ldots, n-1 \end{aligned} \tag{10-30}$$

When $k = n - 1$, we have eliminated all unknowns x_1 to x_{n-1} from the last equation, and the resulting equations are

$$\left.\begin{array}{r} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1j}x_j + \cdots + a_{1n}x_n = b_1 \\ a'_{22}x_2 + \cdots + a'_{2j}x_j + \cdots + a'_{2n}x_n = b'_2 \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ a_{jj}^{(j-1)}x_j + \cdots + a_{jn}^{(j-1)}x_n = b_j^{(j-1)} \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ a_{nn}^{(n-1)}x_n = b_n^{(n-1)} \end{array}\right\} \tag{10-31}$$

Now, we can determine the value of x_n from the last equation. Then substituting this computed value of x_n into next to the last equation, we can determine x_{n-1} and proceed similarly to determine the remaining unknowns:

$$\left.\begin{aligned} x_n &= \frac{b_n^{(n-1)}}{a_{nn}^{(n-1)}} \\ x_{n-1} &= \frac{b_{n-1}^{(n-2)} - a_{n-1,n}^{(n-2)} x_n}{a_{n-1,n-1}^{(n-2)}} \\ &\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ x_j &= \frac{b_j^{(j-1)} - a_{jn}^{(j-1)} x_n - \cdots - a_{j,j+1}^{(j-1)} x_{j+1}}{a_{jj}^{(j-1)}} \end{aligned}\right\} \quad j = n-2, \ldots, 1 \tag{10-32}$$

10-7 Ordinary Differential Equations

Let us consider a first-order differential equation

$$\frac{dy}{dx} = f(x, y) \tag{10-33}$$

which represents the rate of change of y with respect to x and is equal to function $f(x, y)$. A closed-form solution, that is, a direct relationship between x and y, may not be available if $f(x, y)$ is nonlinear. Therefore, numerical methods are used. These methods may be divided into two categories — single-step and multistep — with each type having several different methods. For simplicity and ease of understanding, we will discuss the use of these methods for the numerical integration of single, first-order equations. Higher-order equations may be reduced to a system of first-order equations; then these methods may be applied similarly.

Euler Method

Assume that we know the value of y at $x = x_i$ and we want to determine y at $x = x_{i+1}$. The known value may be the initial condition or may have been computed during the previous computations. From Eq. (10-33), it follows that we can determine the rate of change of y at $x = x_i$:

$$y_i' = f(x_i, y_i) \tag{10-34}$$

where the subscript i refers to the quantities at distance x_i, and a prime, $'$, indicates the derivative with respect to x. Assuming that this rate of change remains constant in the interval $x = x_i$ to $x = x_{i+1}$, we can write

$$y_{i+1} = y_i + y_i'(x_{i+1} - x_i)$$

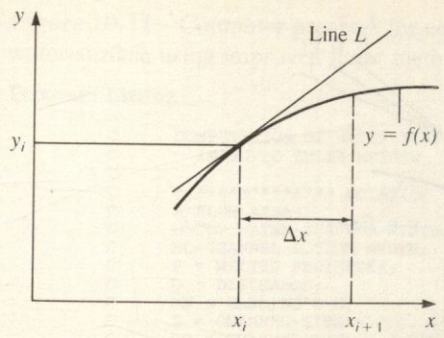


Figure 10-9 Euler method

Substitution of Eq. (10-34) into this equation yields

$$y_{i+1} = y_i + f(x_i, y_i)\,\Delta x \tag{10-35}$$

where $\Delta x = (x_{i+1} - x_i)$

Figure 10-9 shows a geometrical representation of this method. Equation (10-35) is the equation of a straight line (line L in Fig. 10-9) passing through the point (x_i, y_i), and y_i' is the slope of the tangent to the curve at (x_i, y_i). If the rate of change of y remained constant in the interval x_i to x_{i+1}, then Eq. (10-35) will give exact results. However, the rate is not usually constant; thus, a small error is introduced at each step. By expanding y_{i+1} into a Taylor's series and comparing it with Eq. (10-35), we can prove that we are including terms up to only the first power of x. Therefore, Euler's method is referred to as first-order accurate. This method although simple to program, may become unstable for large values of Δx since truncation or roundoff errors are amplified as the value of x increases.

In the Euler method, we used the slope of the tangent at only one point. By using the slope at more than one point, we can improve the accuracy of the results, as we discuss in the following paragraphs.

Improved Euler Method

In this method, the average of the slope of the tangent to the solution curve at points (x_i, y_i) and $(x_{i+1}, y_i + y_i'\Delta x)$ is used (Fig. 10-10, page 560):

$$y_{i+1} = y_i + 0.5[f(x_i, y_i) + f(x_{i+1}, y_i + y_i'\Delta x)]\,\Delta x \tag{10-36}$$

EXAMPLE 10-4 A trapezoidal channel having a bottom width of 20 m, side slopes of 2 horizontal to 1 vertical, and a bottom slope of 0.0001 carries a flow of 110 m^3/s. A bridge is planned on this channel that will raise the flow depth at

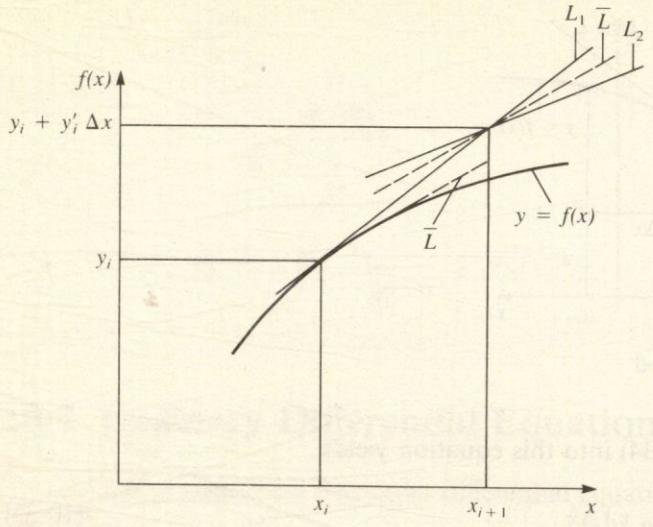


Figure 10-10 Improved Euler method

the bridge to 5 m. Compute the backwater profile in the channel. Assume the Manning n is 0.013.

SOLUTION Modern methods for computing the water surface profiles may be divided into two categories: methods based on the solution of the energy equation between different channel sections and methods based on the solution of the differential equation describing the rate of change of depth with distance. We will follow the second approach in this example.

The following differential equation describes the rate of change of flow depth with distance for gradually varied flows in a prismatic channel (3):

$$\frac{dy}{dx} = \frac{S_0 - S_f}{1 - \dfrac{BQ^2}{gA^3}} \tag{10-37}$$

where y is flow depth at distance x, S_0 is slope of the channel bottom, S_f is slope of the energy-grade line, B is top water surface width, Q is discharge, A is flow area, and g is acceleration due to gravity. Eq. (10-37) is the same as Eq. (4-32) but is in a slightly different form.

We will use the improved Euler method to numerically integrate Eq. (10-37). The initial condition is the flow depth at the bridge, $y = 5.0$ m at $x = 0$. Since the computation of the water levels proceed in the upstream direction, Δx is negative.

Figure 10-11 lists a computer program using the improved Euler method, its input data, and program output. ■

Figure 10-11 Computer program for computing water-surface using improved Euler method

Program Listing

```
C     COMPUTATION OF WATER-SURFACE PROFILE BY USING
C       IMPROVED EULER METHOD
C
C     *************** NOTATION ******************
C     A=FLOW AREA;
C     B=TOP WATER-SURFACE WIDTH;
C     BO=CHANNEL-BOTTOM WIDTH;
C     P = WETTED PERIMETER;
C     Q = DISCHARGE;
C     MN = MANNING'S N;
C     S = CHANNEL-SIDE SLOPE, S HORIZONTAL : 1 VERTICAL;
C     SO = CHANNEL-BOTTOM SLOPE;
C     X = DISTANCE ALONG CHANNEL BOTTOM, POSITIVE IN THE DOWNSTREAM
C           DIRECTION;
C     Y = FLOW DEPTH
C     YD = DEPTH AT DOWNSTREAM END.
C
      REAL MN
      DIMENSION X(100)
      AR(Y)=Y*(BO+S*Y)
      WP(Y)=BO+2.*Y*SQRT(1.+S*S)
      READ (5,*) BO,S,SO,MN,Q,YD
      READ (5,*) N,(X(I),I=1,N)
      WRITE(6,10) BO,S,SO,MN,Q,YD
10    FORMAT(2X,'B =',F5.1,' M',2X,'S =',F4.1,2X,'SO =',F6.4,
     1   2X,'N =',F5.3,2X,'Q =',F7.3,' M3/S',2X,'YD =',F5.3,' M')
      Q2=Q*Q
      QN2=(MN*Q)**2
      Y=YD
      WRITE(6,15)
15    FORMAT(6X,'X',10X,'Y')
      WRITE(6,20) X(1),Y
      DO 30 I = 2,N
      DX=X(I)-X(I-1)
      A=AR(Y)
      P=WP(Y)
      R=A/P
      SF1=QN2/(A*A*R**1.333)
      B=BO+2.*S*Y
      DY1=(SO-SF1)/(1-(B*Q2)/(9.81*A**3))
      Y2=Y+DY1*DX
      A=AR(Y2)
      P=WP(Y2)
      R=A/P
      SF2=QN2/(A*A*R**1.333)
      B=BO+2.*S*Y2
      DY2=(SO-SF2)/(1.-(B*Q2)/(9.81*A**3))
      Y= Y+0.5*(DY1+DY2)*DX
      WRITE(6,20) X(I),Y
20    FORMAT(F10.1,F10.3)
30    CONTINUE
      STOP
      END
```

Input Data

```
20.,2.,0.0001,0.013,110.,5.
10,0.0,-50.,-100.,-200.,-300.,-400.,-500.,-750.,-1000.,-1500.
```

Program Output

```
B = 20.0 M  S = 2.0  SO = .0001  N = .013  Q =110.000 M3/S  YD =5.000 M
     X          Y
     .0      5.000
   -50.0     4.996
  -100.0     4.992
  -200.0     4.983
  -300.0     4.975
  -400.0     4.966
  -500.0     4.958
  -750.0     4.937
 -1000.0     4.916
 -1500.0     4.875
```

Fourth-order Runge-Kutta Method

In this method, the slope of the curve is determined from the following equations:

$$
\begin{aligned}
k_1 &= f(x_i, y_i) \\
k_2 &= f\left(x_i + \frac{1}{2}\Delta x, y_i + \frac{1}{2}k_1 \Delta x\right) \\
k_3 &= f\left(x_i + \frac{1}{2}\Delta x, y_i + \frac{1}{2}k_2 \Delta x\right) \\
k_4 &= f(x_i + \Delta x, y_i + k_3 \Delta x)
\end{aligned}
\tag{10-38}
$$

Then,

$$
y_{i+1} = y_i + \frac{\Delta x}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{10-39}
$$

Predictor-corrector Method

In the single-step methods discussed previously, we used the known information at point x_i, and to improve accuracy, we used the value of the function at x_i, $x_i + \Delta x$, $x_i + 1/2\,\Delta x$. In the predictor-corrector method, we do not compute the function at several points but rather predict the dependent variable first by using values computed during the previous steps, "correct" this predicted value, and then "recorrect" the corrected value if necessary. This iterative procedure is continued until a solution of a desired accuracy is obtained.

Several predictor-corrector methods are available. However, to conserve space, we will discuss only one of them.

In the predictor part, let us use the Euler method to predict the value of y_{i+1}:

$$
y_{i+1}^{(0)} = y_i + f(x_i, y_i)\Delta x \tag{10-40}
$$

where the superscript indicates the number of the iteration (zero iteration is the predicted value). Then, we may correct it by using the following equation:

$$
y_{i+1}^{(1)} = y_i + \frac{1}{2}[f(x_i, y_i) + f(x_{i+1}, y_{i+1}^{(0)})]\Delta x \tag{10-41}
$$

Now, we may recorrect $y_{i+1}^{(1)}$ to obtain a better value:

$$
y_{i+1}^{(2)} = y_i + \frac{1}{2}[f(x_i, y_i) + f(x_{i+1}, y_{i+1}^{(1)})]\Delta x \tag{10-42}
$$

Thus, the jth iteration is

$$y_{i+1}^{(j)} = y_i + \frac{1}{2}[f(x_i, y_i) + f(x_{i+1}, y_{i+1}^{(j-1)})]\,\Delta x \tag{10-43}$$

We continue the iterations until $|y_{i+1}^{(j)} - y_{i+1}^{(j-1)}|$ is less than a specified tolerance. Then, we increment the value of x and repeat the above procedure.

EXAMPLE 10-5 Route the inflow hydrograph listed in Table A through a detention reservoir 420 × 420 ft having side slopes of 2 horizontal to 1 vertical. The rating curve for the outflow facilities is given in Table B. The reservoir level at time $t = 0$ is at $z = 0$.

Table A

Inflow Hydrograph	
Time (min)	Inflow (cfs)
0	0
15	5
30	16
45	39
60	104
75	322
90	555
105	722
120	626
135	481
150	318
165	162

Table B

Outflow Rating Curve	
Stage (ft)	Outflow (cfs)
0	0
1	20
2	48
3	100
4	160
5	210
6	260
7	300
8	330
9	360
10	380

SOLUTION Assuming that the reservoir water surface remains level, the following equation can be written for the variation of water level in the reservoir with time:

$$\frac{dz}{dt} = \frac{Q_{\text{in}} - Q_{\text{out}}}{A} \tag{10-44}$$

where A is horizontal surface area of the reservoir, z is elevation of the water surface in the reservoir above a specified datum, t is time, Q_{in} is inflow into the reservoir, and Q_{out} is outflow from the reservoir.

Figure 10-12 lists a computer program using the predictor-corrector method to solve Eq. (10-44). A routing interval of 15 min was used. Linear interpolations are used to determine the outflow at intermediate levels from the stored data. Since the variation of the reservoir water level is very gradual, no iterations are used to refine the solution. ■

Figure 10-12 Program for reservoir routing using predictor-corrector method

Program Listing

```
C     RESERVOIR ROUTING USING PREDICTOR-CORRECTOR METHOD
C
C     ************** NOTATION *****************
C
C     AR(I) = SURFACE AREA OF RESERVOIR AT LEVEL ELAR(I);
C     DT = ROUTING INTERVAL;
C     NAR = NUMBER OF POINTS ON THE STAGE VS. AREA CURVE;
C     NQO = NUMBER OF POINTS ON THE STAGE VS. OUTFLOW CURVE;
C     NT = NUMBER OF POINTS ON THE INFLOW VS TIME CURVE;
C     QIN(I) = INFLOW AT TIME(I);
C     QO(I) = OUTFLOW AT WATER LEVEL ELQO(I);
C     QO1 = OUTFLOW AT TIME T = 0;
C     Z = RESERVOIR LEVEL ABOVE DATUM;
C     Z1 = RESERVOIR LEVEL AT TIME T = 0;
C     TSTOP = TIME UPTO WHICH ROUTING IS TO BE COMPUTED.
C
      DIMENSION AR(50),ELAR(50),ELQO(50),QO(50),TIME(50),QIN(50)
C
C     INITIAL CONDITIONS
C
      READ(5,*) Z1,QO1,DT,TSTOP
      WRITE(6,20) Z1,QO1,DT,TSTOP
20    FORMAT(//5X,'INITIAL RESERVOIR WATER LEVEL =',F7.2,' M'/
     1 5X,'INITIAL OUTFLOW =',F5.1,' M3/S'/
     2 5X,'ROUTING INTERVAL =',F6.1,' S'/
     3 5X,'TIME UPTO WHICH ROUTING IS TO BE DONE =',F8.1,' S'/)
C
C     STAGE VS. RESERVOIR-SURFACE AREA CURVE
C
      READ (5,*) NAR,(ELAR(I),AR(I),I=1,NAR)
      WRITE(6,30)
30    FORMAT(15X,'STAGE',5X,'RESERVOIR SURFACE AREA'/
     1  16X,'(M)',13X,'(SQ. M)')
      WRITE(6,40) (ELAR(I),AR(I),I=1,NAR)
40    FORMAT(9X,F10.1,10X,F10.1)
C
C     STAGE-OUTFLOW CURVE
C
      READ (5,*) NQO, (ELQO(I),QO(I),I=1,NQO)
      WRITE(6,60)
60    FORMAT(/15X,'STAGE',9X,'  OUTFLOW'/
     1  16X,'(M)',12X,'(M3/S)')
      WRITE(6,70) (ELQO(I),QO(I),I=1,NQO)
70    FORMAT(9X,F10.1,8X,F10.1)
C
C     INFLOW HYDROGRAPH
C
      READ(5,*) NT, (TIME(I),QIN(I), I=1,NT)
      WRITE(6,90)
90    FORMAT(/14X,'TIME',12X,'INFLOW'/
     1  14X,'(S)',13X,'(M3/S)')
      WRITE(6,100) (TIME(I),QIN(I),I=1,NT)
100   FORMAT(8X,F10.1,8X,F10.2)
      T = 0.
      WRITE(6,125)
125   FORMAT(//13X,'TIME',11X,'INFLOW',4X,'RESERVOIR LEVEL',2X,
     1   'OUTFLOW'/13X,'(S)',12X,'(M3/S)',9X,'(M)',11X,'M3/S)')
      WRITE(6,240) T,QIN(1),Z1,QO1
C
C      PREDICTOR PART
C
135   T = T+DT
      DO 140 I=1,NT
      IF (T.LT.TIME(I)) GO TO 150
140   CONTINUE
150   I1=I-1
      QIP=QIN(I1)+(T-TIME(I1))/(TIME(I)-TIME(I1))*(QIN(I)-QIN(I1))
      DO 160 I=1,NQO
      IF (Z1.LT.ELQO(I)) GO TO 170
160   CONTINUE
170   I1=I-1
      QOP=QO(I1)+(Z1-ELQO(I1))/(ELQO(I)-ELQO(I1))*(QO(I)-QO(I1))
```

```
      DO 180 I=1,NAR
      IF (Z1.LT.ELAR(I)) GO TO 190
180   CONTINUE
190   I1=I-1
      ARP=AR(I1)+(Z1-ELAR(I1))/(ELAR(I)-ELAR(I1))*(AR(I)-AR(I1))
      DZP=(QIP-QOP)/ARP
      ZP=Z1+DZP*DT
C
C     CORRECTOR PART
C
      DO 200 I=1,NQO
      IF (ZP.LT.ELQO(I)) GO TO 210
200   CONTINUE
210   I1=I-1
      QOC=QO(I1)+(ZP-ELQO(I1))/(ELQO(I)-ELQO(I1))*(QO(I)-QO(I1))
      DO 220 I=1,NAR
      IF (ZP.LT.ELAR(I)) GO TO 230
220   CONTINUE
230   I1=I-1
      ARC=AR(I1)+(ZP-ELAR(I1))/(ELAR(I)-ELAR(I1))*(AR(I)-AR(I1))
      DZC=(QIP-QOC)/ARC
      Z2=Z1+0.5*DT*(DZC+DZP)
      DO 232 I=1,NQO
      IF (Z2.LT.ELQO(I)) GO TO 235
232   CONTINUE
235   I1=I-1
      Q2=QO(I1)+(Z2-ELQO(I1))/(ELQO(I)-ELQO(I1))*(QO(I)-QO(I1))
      WRITE(6,240) T,QIP,Z2,Q2
240   FORMAT(9X,F10.2,5X,F10.2,2X,F10.2,5X,F10.2)
      IF (T.GT.TSTOP) GO TO 250
      Z1=Z2
      GO TO 135
250   STOP
      END
```

Input Data

```
0.,0.,900.,10800.
11,0.,176400.,1.,179800.,2.,183200.,3.,186600.,4.,190100.,5.,193600.
6.,197100.,7.,200700.,8.,204300.,9.,207900.,10.,211600.
11,0.,0.,1.,20.,2.,48.,3.,100.,4.,160.,5.,210.,6.,260.,7.,300.,8.,330.,
9.,360.,10.,380.
12,0.,0.,900.,5.,1800.,16.,2700.,39.,3600.,104.,4500.,322.,5400.,555.,
6300.,722.,7200.,626.,8100.,481.,9000.,318.,9900.,162.
```

Program Output

```
INITIAL RESERVOIR WATER LEVEL =     .00 M
INITIAL OUTFLOW =    .0 M3/S
ROUTING INTERVAL = 900.0 S
TIME UPTO WHICH ROUTING IS TO BE DONE = 10800.0 S

          STAGE     RESERVOIR SURFACE AREA
           (M)             (SQ. M)
            .0            176400.0
           1.0            179800.0
           2.0            183200.0
           3.0            186600.0
           4.0            190100.0
           5.0            193600.0
           6.0            197100.0
           7.0            200700.0
           8.0            204300.0
           9.0            207900.0
          10.0            211600.0

          STAGE           OUTFLOW
           (M)            (M3/S)
            .0               .0
           1.0             20.0
           2.0             48.0
```

(continued)

Figure 10-12 (continued)

Program Output

```
                3.0          100.0
                4.0          160.0
                5.0          210.0
                6.0          260.0
                7.0          300.0
                8.0          330.0
                9.0          360.0
               10.0          380.0

             TIME          INFLOW
             (S)           (M3/S)
               .0             .00
            900.0            5.00
           1800.0           16.00
           2700.0           39.00
           3600.0          104.00
           4500.0          322.00
           5400.0          555.00
           6300.0          722.00
           7200.0          626.00
           8100.0          481.00
           9000.0          318.00
           9900.0          162.00

  TIME          INFLOW     RESERVOIR LEVEL   OUTFLOW
  (S)           (M3/S)          (M)           M3/S)
    .00            .00          .00            .00
 900.00           5.00          .02            .48
1800.00          16.00          .10           1.98
2700.00          39.00          .28           5.56
3600.00         104.00          .75          15.00
4500.00         322.00         2.15          56.01
5400.00         555.00         4.23         171.55
6300.00         722.00         6.49         279.73
7200.00         626.00         7.92         327.72
8100.00         481.00         8.55         346.56
9000.00         318.00         8.44         343.06
9900.00         162.00         7.69         320.73
10800.00        176.73         7.09         302.77
11700.00        191.45         6.63         285.39
```

10-8 Finite-Difference Approximations

To numerically integrate the ordinary or partial differential equations, we replace the derivative terms by finite-difference approximations and then solve the resulting algebraic equations (1, 4). To facilitate understanding, let us consider the Taylor series expansion of function $f(x)$ about some known point x_0:

$$f(x_0 + \Delta x) = f(x_0) + \Delta x f'(x_0) + \frac{(\Delta x)^2}{2!} f''(x_0) + \frac{(\Delta x)^3}{3!} f'''(x_0) + O[(\Delta x)^4] \tag{10-45}$$

$$f(x_0 - \Delta x) = f(x_0) - \Delta x f'(x_0) + \frac{(\Delta x)^2}{2!} f''(x_0) - \frac{(\Delta x)^3}{3!} f'''(x_0) + O[(\Delta x)^4] \tag{10-46}$$

where $O[(\Delta x)^4]$ denotes terms containing fourth and higher power of Δx. Rearranging and dividing throughout by Δx, we can write Eqs. (10-45) and (10-46) as

$$f'(x_0) = \frac{f(x_0 + \Delta x) - f(x_0)}{\Delta x} + O(\Delta x) \tag{10-47}$$

$$f'(x_0) = \frac{f(x_0) - f(x_0 - \Delta x)}{\Delta x} + O(\Delta x) \tag{10-48}$$

If we neglect the leading error terms, $O(\Delta x)$, in these equations, we obtain the following expressions for the finite-difference approximations:

$$f'(x_0) = \frac{f(x_0 + \Delta x) - f(x_0)}{\Delta x} \tag{10-49}$$

$$f'(x_0) = \frac{f(x_0) - f(x_0 - \Delta x)}{\Delta x} \tag{10-50}$$

Eq. (10-49) is called *forward finite difference*, and Eq. (10-50) is called *backward finite difference*. Both are first-order accurate, that is, the leading error in both is of the first order. Referring to Fig. 10-13, we can say that we replace the slope of the tangent to the curve at $x = x_0$ by the slope of the chord line PB in the forward finite difference and by the slope of the chord line AP in the backward finite difference.

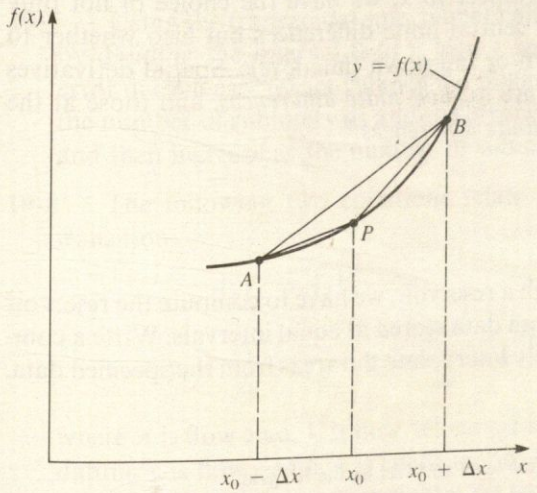


Figure 10-13 Finite-difference approximation

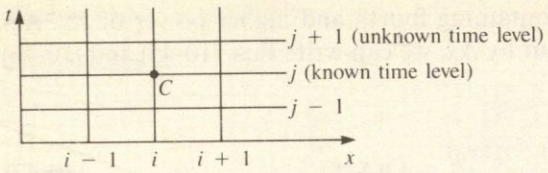


Figure 10-14 Computational grid

Subtracting Eq. (10-46) from Eq. (10-45), dividing throughout by Δx and neglecting higher-order terms, we obtain

$$f'(x_0) = \frac{f(x_0 + \Delta x) - f(x_0 - \Delta x)}{2\,\Delta x} \tag{10-51}$$

This is referred to as the *central finite difference*, and it is second-order accurate. Again, referring to Fig. 10-13, we are replacing the slope of the tangent to the curve at $x = x_0$ by the slope of the chord line AB in the central finite difference.

Let us now consider a function f that is a function of independent variables x and t. Assume that we have divided the x-t plane into a grid having spatial spacing of Δx and timewise spacing of Δt, as shown in Fig. 10-14. For brevity, we will designate the function f at point C as follows:

$$f_C = f(x_0, t_0) = f(i\,\Delta x, j\,\Delta t) = f_i^{\,j} \tag{10-52}$$

Let us say that we know the values of the dependent variables at j time level* (called the known time level) and we want to determine their values at the $j + 1$ time level (referred to as the unknown time level). To replace the spatial derivatives, that is, with respect to x, we have the choice of not only using the forward, backward, or central finite differences but also whether to use these differences at the known or unknown time levels. Spatial derivatives written at the known time level are *explicit finite differences*, and those at the unknown time level are *implicit finite differences.*†

PROBLEMS

10-1 For flood routing through a reservoir, we have to compute the reservoir area at intermediate levels from data stored at equal intervals. Write a computer program to parabolically interpolate the areas from the specified data.

* For brevity, we replace $j\,\Delta t$ time as j time level and $i\,\Delta x$ as the ith grid.

† We discuss the application of these methods for the analysis of unsteady flows in open channels in Chapter 12.

10-2 Modify the computer program in Fig. 10-12, page 564, to use parabolic interpolation instead of linear interpolation for the stage versus outflow curve. Compare the computed results for these two types of interpolations. In addition, study the effects of changing the routing interval on the computed outflows.

10-3 Derive expressions for corrections, Δx and Δy, in the Newton-Raphson method for the following nonlinear equations:

$$F(x, y) = 0$$

$$G(x, y) = 0$$

10-4 Write a computer program to determine the critical depth in a trapezoidal channel by using the bisection method. Use the data in Example 10-2, pages 549–550.

10-5 Use the Newton-Raphson method to determine the critical depth in a trapezoidal channel, and compare the number of iterations required with that in the bisection method.

10-6 To compute the water surface profile in a channel, we solve the energy equation between two consecutive channel sections. Instead of considering two sections at a time, we can write these equations for the entire channel length and then solve all equations simultaneously using the Newton-Raphson method. The flow depth for the given discharge will be known at the downstream end for subcritical flows and at the upstream end for supercritical flows. Write a computer program to compute the water surface profile in the channel of Example 10-4, pages 559–560, and compare the results with those obtained by numerically integrating the differential equation, Eq. (10-37).

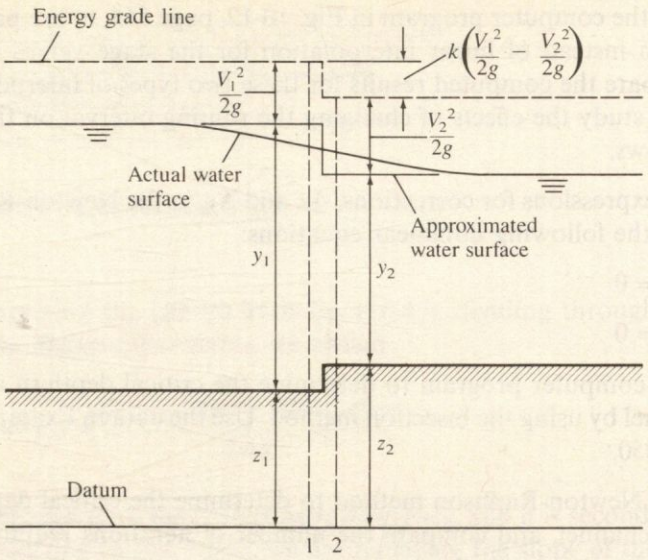
10-7 Using the trapezoidal rule, numerically evaluate the integral $\int_0^{\pi} \sin x \, dx$ by dividing the interval 0 to π into 1 to 2000 subintervals; compute the error in each case (exact value of the integral is 2); and plot a curve between the number of subintervals and the error. Why does the error first decrease and then increase as the number of subintervals increases from 1 to 2000?

10-8 The following two equations relate the flow variables at a channel transition:

$$A_1 V_1 = A_2 V_2$$

$$z_1 + y_1 + \frac{V_1^2}{2g} = z_2 + y_2 + \frac{V_2^2}{2g} + k \frac{|V_1^2 - V_2^2|}{2g}$$

where A is flow area, V is flow velocity, z is height of channel bottom above datum, y is flow depth, k is coefficient of head losses at the transition, and subscripts 1 and 2 denote variables on the upstream and downstream side of the transition.



For given values of flow depth and flow velocity upstream of the transition, write a computer program to determine the flow depth and flow velocity downstream of the transition by using the Newton-Raphson method. Use the expressions developed in Prob. 10-3.

Use this program to determine y_2 and V_2 in a 10-m wide rectangular channel having a step rise of 0.1 m in the channel bottom. Flow velocity and flow depth upstream of the transition are 3 m/s and 2 m, respectively.

10-9 Use the computer programs of Figs. 10-4, page 550, and 10-7, page 553, to solve Probs. 4-6 and 4-9, page 226.

10-10 Write a computer program to determine the alternate depth in a channel, and use this program to solve Prob. 4-20, page 227.

10-11 Solve Prob. 4-27, page 228, by using the computer programs of Probs. 10-4 and 10-5.

10-12 Solve Probs. 4-30 and 4-31, page 229, by using the computer program of Prob. 10-7.

10-13 Route the flood hydrograph of Example 10-5, page 563, through the reservoir using the Improved Euler method. Compare the results with those obtained by using the predictor-corrector method.

10-14 Investigate the effects of routing interval on the peak of outflow hydrograph by solving Example 10-5, page 563. Use the Euler method for solving the governing equation.

10-15 Write a computer program to compute the backwater profile of Prob. 4-50, page 233, using the Euler method and the fourth-order Runge-Kutta method. Compare your results with those obtained by using the methods discussed in Chapter 4.

REFERENCES

1. Chapra, S.C., and R.P. Canale. *Numerical Methods for Engineers with Personal Computer Applications.* McGraw-Hill, New York, 1985.
2. Chaudhry, M.H. *Applied Hydraulic Transients*, 2d ed. Van Nostrand Reinhold, New York, 1987.
3. Chow, V.T. *Open Channel Hydraulics.* McGraw-Hill, New York, 1958.
4. McCracken, D.D., and W.S. Dorn. *Numerical Methods and FORTRAN Programming.* John Wiley & Sons, New York, 1964.