Chapter 4. Numerical Integration

1. Numerical solutions of ODEs

Classification of ODE's: IVP(initial-value problems) & BVP(boundary-value problems)

Ex.) IVP: $y''=-yx \quad B.C.s: y(0)=2,\ y'(0)=1$

BVP: $y''=-yx \quad B.C.s: y(0)=2,\ y(1)=0$

1.1. Initial-value problems for ODE's

$$y^{(m)} = f(x, y, y', y'', \ldots, y^{(m-1)})$$

- The above equation can represent either *m*th-order differential equation, a system of equations of mixed order but with total of m, or a system of m first-order equations.

- Consider the initial-value problem: $y' = f(x, y),\ \ y(x_0) = y_0\ ,\ x_0 \le x \le x_N$

(1) Explicit methods:

<u>Euler's method:</u> $y_{i+1} = y_i + hf(x_i, y_i)$

Runge-Kutta (4th-order)

- most widely used formulas for the numerical solution of ODEs'.

- Advantages: easy to program
  good stability characteristics
  desired step size
  self-starting
- Disadvantages: require more computer time than other methods of comparable accuracy
  local error estimates are somewhat difficult to obtain

- Formula of the R-K type: $\frac{dy}{dt} = f(t, y)$

$$y_{i+1} = y_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)h$$

$$k_1 = f(x_i, y_i), \;\; k_2 = f(x_i + \frac{1}{2}h, y_i + \frac{1}{2}k_1 h)$$

$$k_3 = f(x_i + \frac{1}{2}h, y_i + \frac{1}{2}k_2 h), k_4 = f(x_i + h, y_i + k_3 h)$$

$$y_{i+1} = y_i + \frac{k_1}{6} + \frac{k_2}{3} + \frac{k_3}{3} + \frac{k_4}{6}$$

$$k_1 = hf(x_i, y_i), \;\; k_2 = hf(x_i + \frac{h}{2}, y_i + \frac{k_1}{2}),$$

$$k_3 = hf(x_i + \frac{h}{2}, y_i + \frac{k_2}{2}), k_4 = hf(x_i + h, y_i + k_3)$$

Example 1: R-K method: 1st-order ODE $$\frac{dy}{dt} = \frac{4t}{y} - ty, \quad y(0) = 3 \quad (\Delta t = 0.1)$$

```
c ... case 1.
c ... runge-kutta method (1st order ODE)

      implicit double precision (a-h,o-z)
      parameter (nf=5)
      external f1

      open(unit=2,file='ode1.dat',status='unknown')

c ... initial data
      t = 0.d0
      dt= 1.d-1
      y = 3.d0
      ntime = nf/dt
      write(2,20) t,y

c ... numerical integration
      do i=1,ntime
      call runge (f1,dt,t,y)
      t=t+dt
      if(mod(i,1).eq.0.0) write(2,20) t,y
 20   format(1x,'t=',f12.6,2x,'y=',f12.6)
      enddo

      stop
      end
```

```
c ------------------------------------------------
      subroutine runge (f1,dt,t,y)
      implicit double precision (a-h,o-z)
      parameter (hf=0.5d0)
      external f1
      y1=dt*f1(t,y)
      y2=dt*f1(t+hf*dt,y+hf*y1)
      y3=dt*f1(t+hf*dt,y+hf*y2)
      y4=dt*f1(t+dt,y+y3)
      y=y+(y1+2.d0*y2+2.d0*y3+y4)/6.d0
      return
      end

        double precision function f1(t,y)
        implicit double precision (a-h,o-z)
        f1=4.d0*t/y - t*y
        return
        end
```

solution: t= 0.000000 y= 3.000000
t= 0.200000 y= 2.967145
t= 0.400000 y= 2.874147
t= 0.600000 y= 2.736491
t= 0.800000 y= 2.576134
t= 1.000000 y= 2.416485
t= 1.200000 y= 2.276983
t= 1.400000 y= 2.168942

Example 2: R-K method: 2nd-order ODE

$$\frac{d^2y}{dt^2}+2\frac{dy}{dt}+4y=0,\ \ y(0)=2,\ \ \frac{dy(0)}{dt}=0 \quad\Rightarrow\quad \frac{dz}{dt}=-2z-4y,\ \ \frac{dy}{dt}=z,\ \ y(0)=2,\ \ z(0)=0$$

```fortran
c ... case 2.
c ... runge-kutta method (2nd order ODE)
      implicit double precision (a-h,o-z)
      external f1,f2
      parameter (n=5)

      open(unit=2,file='ode2.dat',status='unknown')
c ...  initial data
      t = 0.d0
      dt= 1.d-1
      z = 0.d0
      y = 2.d0
      ni= n/dt
      write(2,20) t,y,z
c ... integration
      do 10 i=1,ni
      call runge (f1,f2,dt,y,z)
      t=t+dt
      if(mod(i,5).eq.0.0) write(2,20) t,y,z
20    format(1x,'t=',f12.6,2x,'y=',f12.6,2x,'z=',f12.6)
10    continue

      stop
      end
```

Solution: t= 0.000000 y= 2.000000 z= 0.000000
t= 1.000000 y= 0.301136 z=-1.677142
t= 2.000000 y=-0.306259 z= 0.198154
t= 3.000000 y=-0.004571 z= 0.203573
t= 4.000000 y= 0.041989 z=-0.050870
t= 5.000000 y=-0.004342 z=-0.021541

```fortran
c -----------------------------------------------
      subroutine runge (f1,f2,dt,y,z)
      parameter (hf=5d-1)
      implicit double precision (a-h,o-z)
      external f1,f2
      z1=dt*f1(y,z)
      y1=dt*f2(y,z)
      z2=dt*f1(y+hf*y1,z+hf*z1)
      y2=dt*f2(y+hf*y1,z+hf*z1)
      z3=dt*f1(y+hf*y2,z+hf*z2)
      y3=dt*f2(y+hf*y2,z+hf*z2)
      z4=dt*f1(y+y3,z+z3)
      y4=dt*f2(y+y3,z+z3)
      z=z+(z1+2.d0*z2+2.d0*z3+z4)/6.d0
      y=y+(y1+2.d0*y2+2.d0*y3+y4)/6.d0
      return
      end

      double precision function f1(y,z)
      implicit double precision (a-h,o-z)
      f1=-2.d0*z-4.d0*y
      return
      end

      double precision function f2(y,z)
      implicit double precision (a-h,o-z)
      f2=z
      return
      end
```

(2) Implicit methods

The implicit Euler method (1st-order accurate): $y_{i+1} = y_i + hf_{i+1}$ i=0,…,n-1

$$y(0) = y_0$$

The traperzoidal method (2nd-order accurate): $y_{i+1} = y_i + \frac{h}{2}(f_{i+1} + f_i)$ i=0,…,n-1

$$y(0) = y_0$$

1.2. Boundary-value problems for ODEs

(1) One space variable

$$Ly = ay'' + by' + cy = f(x, y', y'') \quad x \in \Omega$$

Typical B.C's: $y(x_0)=y_0$, $y(x_N)=y_N$ or $y(x_0)=y_0$, $y'(x_N)=0$

Ex.) y'' + ½ $\alpha(u'^2+u^2) = 1$, $0 \le x \le \pi/2$; y(0)=1, y(π/2)=0

When α=0, $y = 1 - \left(\frac{p}{4} + \frac{2}{p}\right)x + \frac{1}{2}x^2$

α≠0 (α=1): y=1-*sin*x (solution is unique)

(another case, if α=1 and B.C's, y(0)=1, y'(π/2)=0 then, y=1±sinx (two solutions))
When attempting to solve problem computationally, the question of existence and uniqueness of the theoretical (or exact) solution should always be born in mind.

(2) Two space variables

$$Lu = a\frac{\partial^2 u}{\partial x^2} + b\frac{\partial^2 u}{\partial x \partial y} + c\frac{\partial^2 u}{\partial y^2} = f\left(x, y, u, \frac{\partial u}{\partial x}, \frac{\partial u}{\partial y}\right) \quad (x, y) \in \Omega$$

- Elliptic in Ω if b^2-4ac<0, parabolic if b^2-4ac=0 , and hyperbolic if b^2-4ac>0
- Elliptic equations possess no such directions (=characteristic curves) at a point, whereas parabolic and hyperbolic equations possess one and two.
- Elliptic equations cannot be solved using step-by-step integration along a characteristic curve)
- Moreover, well-posed elliptic problems have their B.C.'s specified on a closed boundary, whereas parabolic and hyperbolic problems do not.

(3) Typical boundary conditions

- Dirichlet problem: $u = a(x, y)$ (x,y) $\in \Gamma$; α is prescribed function on Γ
- Neumann problem: $\frac{\partial u}{\partial n} = b(x, y)$ (x,y) $\in \Gamma$; β is prescribed function on Γ
- Robin problem (mixed B.C.): $a(x, y)u + b(x, y)\frac{\partial u}{\partial n} = g(x, y)$ (x,y) $\in \Gamma$; α, β>0 on Γ

- Finite difference table (even spacing mesh size)

	Backward									Forward									
	Div	L4	L3	L2	L1	*I*	R1	R2	R3	Err	Div	L3	L2	L1	I	R1	R2	R3	R4	Err
F'	h				-1	***1***				h	h				**-1**	1				h
	2h			1	-4	***3***				h^2	2h				**-3**	4	-1			h^2
F''	h^2			1	-2	***1***				h	h^2				**1**	-2	1			h
	h^2		-1	4	-5	***2***				h^2	h^2				**2**	-5	4	-1		h^2
F'''	h^3		-1	3	-3	***1***				h	h^3				**-1**	3	-3	1		h
	$2h^3$	3	-14	24	-18	***5***				h^2	$2h^3$				**-5**	18	-24	14	-3	h^2

	Central								Offset (one R1 and several L 's)								
	Div	L3	L2	L1	I	R1	R2	R3	Err	Div	L3	L2	L1	I	R1	R2	R3	Err
F'	2h			-1	***0***	1			h^2									
	12h		1	-8	***0***	8	-1		h^4									
F''										$2h^2$		1	-1	***-1***	1			h
	h^2			1	**-2**	1			h^2	$9h^2$	2	0	-3	***-2***	3			h
	$12h^2$		-1	16	***-30***	16	-1		h^4	$3h^2$	1	-1	0	***-1***	1			h
										$7h^2$	-2	8	-5	***-6***	5			h^2
F'''	$2h^3$		-1	2	***0***	-2	1		h^2	h^3		-1	3	***-3***	1			h
	$8h^3$	1	-8	13	***0***	-13	8	-1	h^4	$2h^3$	1	-6	12	***-10***	3			h^2

(4) Numerical methods

Runge-Kutta (shooting method, explicit)

Example 3: 2^{nd}-order ODE (two-point boundary value problem):

$$\frac{d^2y}{dt^2}+\frac{1}{4}y=8 \qquad y(0)=0,\ y(10)=Y_n=0 \text{ (shooting method).}$$

- Assume $\frac{dy}{dt}(0)=U$
- Two solutions of the initial value problem are carried out using two estimates, U_o and U_{oo}, yielding Y_o, Y_{oo}.
- New estimate (linear interpolation using two estimates):

$$U_1=U_o+\left[Y_n-Y_0\right]\left[\frac{U_o-U_{oo}}{Y_o-Y_{oo}}\right]$$

- In the linear case, desired solution is obtained within only three iterations.

Program:

```
c ... case 3.
c ... runge-kutta method (2nd order ODE, shooting method)
      implicit double precision (a-h,o-z)
      parameter (nmax=2000,n=10,tol=1.d-7)
      double precision y(0:nmax),z(0:nmax)
      external fy,fz

      open(unit=2,file='ode3.dat',status='unknown')

c ... input data
      iter = 1
      dt = 0.1d0
      nx = n/dt
      z0asm = 10.d0
      y(0) = 0.d0
      ynx = 0.d0
40    z(0) = z0asm
      t = 0.d0

c ... integration
      do i=1,nx
      call runge (fy,fz,dt,y(i-1),z(i-1),y(i),z(i))
      t = t + dt
      enddo

c ... shooting method
      print *,iter,z(0),t,y(nx)
      if (dabs(y(nx)).lt.tol) go to 50
      if (iter.gt.50) stop 'Maximum iterations'
      z0tmp=z0asm
      if (iter.eq.1) z0asm=z0asm*1.01d0
      if (iter.ge.2) z0asm=z0asm
     & + (ynx-y(nx))*(z0asm-z0old)/(y(nx)-ynxold)
      z0old=z0tmp
      ynxold=y(nx)
      iter=iter+1
      go to 40
```

```
50    t = 0.d0
      do i=0,nx
      if(mod(i,5).eq.0) write(2,30) t,y(i)
      t = t + dt
      enddo
30    format(1x,'  t=',f12.6,'  y=',f12.6)

      stop
      end

c ------------------------------------------------
        subroutine runge (fy,fz,dt,y,z,ya,za)
        implicit double precision (a-h,o-z)
        parameter (hf=0.5d0)
        external fy,fz

        z1=dt*fz(y)
        y1=dt*fy(z)
        z2=dt*fz(y+hf*y1)
        y2=dt*fy(z+hf*z1)
        z3=dt*fz(y+hf*y2)
        y3=dt*fy(z+hf*z2)
        z4=dt*fz(y+y3)
        y4=dt*fy(z+z3)
        za=z+(z1+2.d0*z2+2.d0*z3+z4)/6.d0
        ya=y+(y1+2.d0*y2+2.d0*y3+y4)/6.d0
        return
        end

        double precision function fz(y)
        implicit double precision (a-h,o-z)
        fz = 8.d0-y/4.d0
        return
        end

        double precision function fy(z)
        implicit double precision (a-h,o-z)
        fy = z
        return
        end
```

Matrix method (Tridiagonal form)

$$\frac{d^2y}{dt^2} + Ay = B, \quad y(0) = 0, y(L) = 0 \quad (0 \le t \le L)$$

$$\begin{bmatrix} a & 1 & & & & & \\ 1 & a & 1 & & & & \\ & 1 & a & 1 & & & \\ & & - & - & - & & \\ & & & - & - & - & \\ & & & & 1 & a & 1 \\ & & & & & 1 & a \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ - \\ - \\ y_{n-1} \\ y_n \end{bmatrix} = \begin{bmatrix} B(\Delta x)^2 \\ B(\Delta x)^2 \\ B(\Delta x)^2 \\ - \\ - \\ B(\Delta x)^2 \\ B(\Delta x)^2 \end{bmatrix}$$

where $\alpha=-2+A(\Delta x)^2$

- Advantage: its simplicity.
- Drawback: when encountered in dealing with nonlinear differential equations.

  Iterative technique is required

Example 4: $$\frac{d^2y}{dt^2}+\frac{1}{4}y=8,\ y(0)=y(10)=0$$

```
c ... case 4.
c ... tridiagonal matrix type
      implicit double precision (a-h,o-z)
      parameter (nmax=2000,n=10)
      double precision y(nmax)

c ... input data
      dt = 0.1d0
      nx = n/dt
      print *,nx

c ... integration
      call tri_mat(y,nx-1,dt)

      open (unit=2,file='ode4.dat',status='unknown')
      t = 0.d0
      do i=0,nx
      if(mod(i,5).eq.0) write(2,10) t,y(i)
      t = t + dt
      enddo
10    format(1x,' t=',f12.6,1x,' y=',f12.6)

      stop
      end
c ---------------------------------------------
      subroutine tri_mat(xs,n,dt)
      implicit double precision (a-h,o-z)
      double precision xs(n)
      double precision xa(n),xb(n),xc(n),xd(n)

      xa(1) = 0.d0
      do i=2,n
      xa(i) = 1.d0
      enddo

      do i=1,n
      xb(i) = dt*dt/4.d0 - 2.d0
      enddo

      do i=1,n-1
      xc(i) = 1.d0
      enddo
      xc(n) = 0.d0

      do i=1,n
      xd(i) = 8.d0*dt*dt
      enddo

      call gelm(xs,n,xa,xb,xc,xd)

      return
      end
c ---------------------------------------------
      subroutine gelm(xs,n,xa,xb,xc,xd)
      implicit double precision (a-h,o-z)
      double precision xa(n),xb(n),xc(n),xd(n)
      double precision xs(n)

      do i=1,n-1
      xb(i+1) = xb(i+1) - xc(i)*xa(i+1)/xb(i)
      xd(i+1) = xd(i+1) - xd(i)*xa(i+1)/xb(i)
      enddo
      xs(n) = xd(n)/xb(n)
      do i=n-1,1,-1
      xs(i) = (xd(i)-xc(i)*xs(i+1))/xb(i)
      enddo
      return
      end
```

Solution:
```
t= 0.000000   y= 0.000000
t= 2.000000   y= 34.818673
t= 4.000000   y= 67.045377
t= 6.000000   y= 67.045377
t= 8.000000   y= 34.818673
t=10.000000   y= 0.000000
```

A. The linear case: $Lu = u'' + b(x)u' + c(x)u = f(x)$, (x,y) $\in \Omega$ ($\Omega=[x_0,x_N]$)

B.C.: $u(x_0)=u_0$, $u(x_N)=u_N$

Finite difference scheme → tridiagonal matrix form !!!

If this system has derivative B.C.'s ($u(x_0)=u_0$, $u'(x_N)=0$) , two methods can be used.

$$\frac{u_N - u_{N-1}}{h} = 0$$ (solve the tridiagonal matrix from u_1 to u_{N-1})

$$\frac{u_{N+1} - u_{N-1}}{2h} = 0$$ (solve the tridiagonal matrix from u_1 to u_{N-1})

B. The nonlinear case: $u'' = 1 - \frac{1}{2}\left[(u')^2 + u^2\right]$, u(0)=1, u(π/2)=0

Case 1) $$\frac{u_{i+1}^{r+1} - 2u_i^{r+1} + u_{i-1}^{r+1}}{h^2} = 1 - \frac{1}{2}\left[\left(\frac{u_{i+1}^{r} - u_{i-1}^{r}}{2h}\right)^2 + \left(u_i^r\right)^2\right]$$

u(0)=1,u(n)=0, initial guess: u(x)=1-cx → matrix form

Case 2) $$\frac{u_{i+1}^{r+1} - 2u_i^{r+1} + u_{i-1}^{r+1}}{h^2} = 1 - \frac{1}{2}\left[\left(\frac{u_{i+1}^{r} - u_{i-1}^{r}}{2h}\right)\left(\frac{u_{i+1}^{r+1} - u_{i-1}^{r+1}}{2h}\right) + \left(u_i^r\right)^2\right]$$ → matrix form

Case 3) Two iterative methods which do not involve the solution of matrix systems.

$$\frac{u_{i+1}^{r} - 2u_i^{r+1} + u_{i-1}^{r}}{h^2} = 1 - \frac{1}{2}\left[\left(\frac{u_{i+1}^{r} - u_{i-1}^{r}}{2h}\right)^2 + \left(u_i^r\right)^2\right]$$

$$\frac{u_{i+1}^{r} - 2u_i^{r+1} + u_{i-1}^{r+1}}{h^2} = 1 - \frac{1}{2}\left[\left(\frac{u_{i+1}^{r} - u_{i-1}^{r+1}}{2h}\right)^2 + \left(u_i^r\right)^2\right]$$

Newton's method

State variables at k+1th iteration

State variables at kth iteration

$$\underline{\underline{J}}^{k}(\underline{x}^{k+1}-\underline{x}^{k})=\underline{\underline{J}}^{k}\Delta x^{k}=-\underline{f}(\underline{x}^{k})$$

Residual vector

Jacobian matrix

$$\underline{\underline{J}}^{k}\equiv\left.\frac{\partial\underline{f}}{\partial\underline{x}}\right|^{k}\Rightarrow J_{ij}^{k}\equiv\frac{\partial f_i(\underline{x}^{k})}{\partial x_j}$$