§1.5. Computational Techniques

1. Tridiagonal Linear System

$$\begin{cases} -b_i x_{i-1} + a_i x_i - c_i x_{i+1} = d_i, \quad i=1,\cdots,N-1, \\ x_0 = 0, \quad x_N = 0. \end{cases}$$

$$\Rightarrow \begin{bmatrix} a_1 & -c_1 & & \\ -b_2 & a_2 & -c_2 & \\ & \ddots & \ddots & -c_{N-2} \\ & & -b_{N-1} & a_{N-1} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{N-2} \\ x_{N-1} \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_{N-2} \\ d_{N-1} \end{bmatrix}.$$

$i=1$, $\quad -b_1 x_0 + a_1 x_1 - c_1 x_2 = d_1 \Rightarrow x_1 = \frac{c_1}{a_1} x_2 + \frac{d_1}{a_1} \equiv V_1 + \beta_1 x_2,$

$i=2$, $\quad -b_2 x_1 + a_2 x_2 - c_2 x_3 = d_2 \Rightarrow x_2 = \frac{c_2}{a_2 - b_2\beta_1} x_3 + \frac{d_2 + b_2 V_1}{a_2 - b_2\beta_1} \equiv V_2 + \beta_2 x_3.$

$\cdots$

$$\boxed{\begin{array}{l} \begin{cases} V_k = \dfrac{d_k + b_k V_{k-1}}{a_k - b_k \beta_{k-1}}, \quad \beta_k = \dfrac{c_k}{a_k - b_k \beta_{k-1}}, \quad k=1,2,\cdots,N-1, \\ V_0 = 0, \quad \beta_0 = 0. \end{cases} \\ \begin{cases} x_m = V_m + \beta_m x_{m+1}, \quad m = N-1, N-2, \cdots, 2, 1, \\ x_N = 0. \end{cases} \end{array}}$$

called Thomas Algorithm

Example 1. C-N Scheme

$$\begin{cases} -rU_{i-1}^{n+1} + (1+2r)U_i^{n+1} - rU_{i+1}^{n+1} = rU_{i-1}^n + (1-2r)U_i^n + rU_{i+1}^n + \Delta t f_i^{n+\frac{1}{2}}, \\ U_0^{n+1} = U_N^{n+1} = 0, \quad i=1,\cdots,N-1. \end{cases}$$

We will choose $a_i = 1+2r$, $b_i = r$, $c_i = r$, $d_i = rU_{i-1}^n + (1-2r)U_i^n + rU_i^n + \Delta t f_i^{n+\frac{1}{2}}$

and Apply the Thomas Algorithm.

For the scheme $-U_{i-1} + (2+h^2)U_i - U_{i+1} = h^2 g(x_i)$, $\quad a_i = 2+h^2$, $b_i = +1$, $c_i = +1$.

$d_1 = h^2 g(x_1) + u_a$, $d_i = h^2 g(x_i)$, $\cdots$ $d_{N-1} = h^2 g(x_{N-1}) + u_b$.

2. Point Iterative methods

$$A\vec{x} = \vec{b} \Rightarrow \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1N} \\ a_{21} & a_{22} & \cdots & a_{2N} \\ \cdots & & & \\ a_{N1} & a_{N2} & \cdots & a_{NN} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \end{bmatrix} \Rightarrow \sum_{j=1}^{N} a_{ij} x_j = b_i, \quad i = 1, 2, \cdots N.$$

$(a_{ii} \neq 0)$

$$a_{i1} x_1 + a_{i2} x_2 + \cdots + a_{i,i-1} x_{i-1} + a_{ii} x_i + a_{i,i+1} x_{i+1} + \cdots + a_{iN} x_N = b_i$$

$$\Rightarrow x_i = \frac{1}{a_{ii}} \{ b_i - \sum_{\substack{j=1 \\ j \neq i}}^{N} a_{ij} x_j \}$$

$$\Rightarrow \boxed{x_i^{(I+1)} = \frac{1}{a_{ii}} \{ b_i - \sum_{\substack{j=1 \\ j \neq i}}^{N} a_{ij} x_j^{(I)} \}, \quad \text{called Jacobi Iteration}}$$

$i = 1, \cdots, N, \quad I = 0, 1, 2, \cdots$

until $\max_{1 \le i \le N} | x_i^{(I+1)} - x_i^{(I)} | \le \varepsilon$ or $\sqrt{\frac{1}{N} \sum_{i=1}^{N} | x_i^{(I+1)} - x_i^{(I)} |^2} \le \varepsilon$

$$\Rightarrow \boxed{x_i^{(I+1)} = \frac{1}{a_{ii}} \{ b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(I+1)} - \sum_{j=i+1}^{N} a_{ij} x_j^{(I)} \}. \quad i = 1, \cdots N, \quad I = 0, 1, 2, \cdots \quad \text{called Gauss - Seidel Iteration}}$$

The GS Iteration converges twice as fast as the Jacobi Iteration

$$\Rightarrow \begin{cases} \bar{x}_i^{(I+1)} = \frac{1}{a_{ii}} \{ b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(I+1)} - \sum_{j=i+1}^{N} a_{ij} x_j^{(I)} \}, \\ x_i^{(I+1)} = \omega \bar{x}_i^{(I+1)} + (1-\omega) x_i^{(I)}, \quad 0 < \omega < 2. \end{cases}$$

when $1 < \omega < 2$. $\omega_{opt} = \frac{2}{1 + \sqrt{1 - \mu_{max}^2}}$. Successive Over-Relaxation (SOR) Iteration

The SOR Iteration converges squarely as fast as the GS Iteration

However, determining ω_{opt} will be difficult sometimes.

3. Conjugate Gradient Method (CGM)

$$A\vec{x}=\vec{b},\ (A^T=A)\quad A=\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1N} \\ a_{21} & a_{22} & \cdots & a_{2N} \\ & \cdots\cdots & & \\ a_{N1} & a_{N2} & \cdots & a_{NN} \end{bmatrix},\quad a_{ij}=a_{ji}.$$

$$(\vec{x}^TA\vec{x}>0)$$

$\Longleftrightarrow$ $E(\vec{y})=\frac{1}{2}(\vec{y},A\vec{y})-(\vec{y},\vec{b})$ has a unique minimum at $\vec{y}=\vec{x}$.

Here, $\langle\vec{y},\vec{b}\rangle=\vec{y}^T\vec{b}=\sum_{i=1}^{N}y_ib_i$. Euclidean Inner Product.

Idea: Given $\vec{x}^0 \Rightarrow \vec{r}^0=\vec{b}-A\vec{x}^0$, creat $\vec{x}^1=\vec{x}^0+\alpha_0\vec{r}^0$

$$E(\vec{x}^1)=E(\vec{x}^0+\alpha\vec{r}^0)$$
$$=E(\vec{x}^0)+\alpha_0(\vec{r}^0,A\vec{x}^0)-\alpha_0(\vec{r}^0,\vec{b})+\frac{\alpha_0^2}{2}(\vec{r}^0,A\vec{r}^0)$$
$$=E(\vec{x}^0)-\alpha_0(\vec{r}^0,\vec{r}^0)+\frac{\alpha_0^2}{2}(\vec{r}^0,A\vec{r}^0).$$

Take $\frac{\partial E(\vec{x}^1)}{\partial\alpha_0}=0$ to find the minimum point $\Rightarrow \alpha_0=\frac{(\vec{r}^0,\vec{r}^0)}{(\vec{r}^0,A\vec{r}^0)}$.

$$\boxed{\Rightarrow \begin{cases}\vec{x}^{k+1}=\vec{x}^k+\alpha_k\vec{r}^k,\\ \vec{r}^{k+1}=\vec{b}-A\vec{x}^{k+1}=\vec{r}^k-\alpha_kA\vec{r}^k.\end{cases}\quad \text{where } \alpha_k=\frac{(\vec{r}^k,\vec{r}^k)}{(\vec{r}^k,A\vec{r}^k)},\ k=0,1,2,\cdots}$$

Called Steepest Descent Method.

However, the iteration converges slowly. To improve the iteration, we introduce

$$\vec{x}^{k+1}=\vec{x}^k+\alpha_k\{\vec{r}^k+\tau_k(\vec{x}^k-\vec{x}^{k-1})]$$
$$=\vec{x}^k+\alpha_k\vec{p}^k,$$
$$\vec{p}^{k+1}=\vec{r}^{k+1}+\tau_{k+1}(\vec{x}^{k+1}-\vec{x}^k)=\vec{r}^{k+1}+\tau_{k+1}\alpha_k\vec{p}^k\equiv\vec{r}^{k+1}+\beta_k\vec{p}^k,$$
$$\vec{r}^{k+1}=\vec{b}-A\vec{x}^{k+1}=\vec{r}^k-\alpha_kA\vec{p}^k=\vec{r}^k-\alpha_k\vec{q}^k,$$

$$\vec{q}^{k+1} = A\vec{p}^{k+1} = A\vec{r}^{k+1} + \beta_k \vec{q}^k,$$

where (α_k, β_k) conjugate parameters given $\alpha_k = \frac{|\vec{r}^k|^2}{(\vec{q}^k, \vec{p}^k)}$, $\beta_k = \frac{|\vec{r}^{k+1}|^2}{|\vec{r}^k|^2}$.

Conjugate Gradient Method

step 1. Given $\vec{x}^0 \Rightarrow \vec{r}^0 = \vec{b} - A\vec{x}^0$; $\vec{p}^0 = \vec{r}^0$; $\vec{q}^0 = A\vec{r}^0$.

step 2. $\vec{x}^{k+1} = \vec{x}^k + \alpha_k \vec{p}^k$, $\vec{r}^{k+1} = \vec{r}^k - \alpha_k \vec{q}^k$, $\alpha_k = \frac{|\vec{r}^k|^2}{(\vec{p}^k, \vec{q}^k)}$.

step 3. $\beta_k = \frac{|\vec{r}^{k+1}|^2}{|\vec{r}^k|^2}$, $\vec{p}^{k+1} = \vec{r}^{k+1} + \beta_k \vec{p}^k$, $\vec{q}^{k+1} = A\vec{r}^{k+1} + \beta_k \vec{q}^k$.

Repeat (2) and (3) until $|\vec{r}^k|^2 = \langle \vec{r}^k, \vec{r}^k \rangle = \sum_{i=1}^{N} (r_i^k)^2$ is very small.

The iteration converges squarely as fast as the G-S iteration, which is the same speed as SOR with w_{opt}.

Example 2. 5-point scheme for $u_{xx} + u_{yy} = -f(x,y)$:

$$U_{i+1,j} + U_{i-1,j} + U_{i,j+1} + U_{i,j-1} - 4U_{i,j} = -h^2 f_{i,j}, \quad i,j = 1, 2, \cdots N.$$

Jacobi Iteration: $U_{i,j}^{(l+1)} = \frac{1}{4}\left[h^2 f_{i,j} + U_{i+1,j}^{(l)} + U_{i-1,j}^{(l)} + U_{i,j+1}^{(l)} + U_{i,j-1}^{(l)}\right]$, $i,j = 1, \cdots, N$.

Gauss-Seidel Iteration: $U_{i,j}^{(l+1)} = \frac{1}{4}\left[h^2 f_{i,j} + U_{i-1,j}^{(l+1)} + U_{i,j-1}^{(l+1)} + U_{i+1,j}^{(l)} + U_{i,j+1}^{(l)}\right]$, $i,j = 1, \cdots, N$.

CGM: (1) Guess $U_{i,j}^0 \Rightarrow$
$$\begin{cases} r_{ij}^0 = h^2 f_{ij} + U_{i+1,j}^0 + U_{i-1,j}^0 + U_{i,j+1}^0 + U_{i,j-1}^0 - 4U_{i,j}^0, \\ p_{i,j}^0 = r_{i,j}^0, \\ q_{i,j}^0 = 4r_{i,j}^0 - r_{i+1,j}^0 - r_{i-1,j}^0 - r_{i,j+1}^0 - r_{i,j-1}^0. \end{cases}$$

$$\Rightarrow |\vec{r}^0|^2 = \sum_{i,j} (r_{i,j}^0)^2, \quad (\vec{p}^0, \vec{q}^0) = \sum_{i,j} p_{i,j}^0 q_{i,j}^0 \Rightarrow \alpha_0 = |\vec{r}^0|^2 / (\vec{p}^0, \vec{q}^0).$$

(2) $U_{i,j}^{k+1} = U_{i,j}^k + \alpha_k p_{i,j}^k$, $r_{i,j}^{k+1} = r_{i,j}^k - \alpha_k q_{i,j}^k$, $\Rightarrow |\vec{r}^{k+1}|^2 = \sum_{i,j} (r_{ij}^{k+1})^2$, $\beta_k = |\vec{r}^{k+1}|^2 / |\vec{r}^k|^2$.

(3) $p_{i,j}^{k+1} = r_{ij}^{k+1} + \beta_k p_{i,j}^k$, $q_{i,j}^{k+1} = 4r_{i,j}^{k+1} - r_{i+1,j}^{k+1} - r_{i,j+1}^{k+1} - r_{i-1,j}^{k+1} - r_{i,j-1}^{k+1} + \beta_k q_{i,j}^k$.

HW. Ex. 1.5.1, Ex 1.5.2 P. 36

```
c 1D Heat Conduction Problem solved using C-N scheme
c         Ut = k Uxx
c         U(x,0) = sin (pi*x)
c         U(0,t) = U(1,t) = 0
c The exact solution: U(x,t)= exp(-pi*pi*t)*sin(pi*x)

        dimension u1(0:5000),u2(0:5000),aa(0:5000),bb(0:500),cc(0:5000)
        dimension ex(0:5000),vv(0:5000),be(0:5000),ar(0:5000),dd(0:5000)
        double precision u1,u2,aa,bb,cc,dd,be,ar,vv,ex,dx,dt,r,err,pi

c parameters
        dt=0.001
        dx=0.01
        r=dt/(2.0*dx*dx)
        pi=3.14159265358979323846
        nx=100
        nt=0

c initial condition
        do i=0,nx
        u1(i)=sin(pi*i*dx)
        enddo
        u1(0)=0.0
        u1(nx)=0.0

c set up tridiagonal system
        do i=1,nx-1
        aa(i)=1.0+2.0*r
        bb(i)=r
        cc(i)=bb(i)
        enddo
        u2(0)=0.0
        u2(nx)=0.0
 1      do i=1,nx-1
        dd(i)=r*u1(i-1)+(1.0-2.0*r)*u1(i)+r*u1(i+1)
        enddo

c Thomas algorithm
        be(0)=0.0
        ar(0)=0.0
        do k=1,nx-1
        be(k)=cc(k)/(aa(k)-bb(k)*be(k-1))
        ar(k)=(dd(k)+bb(k)*ar(k-1))/(aa(k)-bb(k)*be(k-1))
        enddo

        vv(nx)=0.0
        do j=1,nx-1
        jj=nx-j
        vv(jj)=be(jj)*vv(jj+1)+ar(jj)
        enddo

        do i=1,nx-1
        u2(i)=vv(i)
        enddo

c exact solution and error
        do i=0,nx
        ex(i)=exp(-pi*pi*nt*dt)*sin(pi*i*dx)
        enddo
        err=0.0
        do i=1,nx-1
        err=err+(u2(i)-ex(i))*(u2(i)-ex(i))
        enddo
        err=sqrt(dx*err)
        print *, nt, err
        if(nt.eq.1000)goto 2
        nt=nt+1
```

```
      do i=0,nx
      u1(i)=u2(i)
      enddo
      goto 1

c output
 2    open(unit=6,file='solution.data')
      do i=0,nx
      write(6,3) nt*dt,u2(i),ex(i)
      enddo
 3    format(f12.8,1x,f12.10,1x,f12.10)
      end
```

```
c 2D Laplace Problem solved using Iterative Method
c          Uxx + Uyy = 0
c          U(x,0) = 0, U(x,1) = 1
c          U(0,y) = U(1,y) = 0
c The exact solution: U(x,y) = ...

        dimension uold(0:1000,0:1000),unew(0:1000,0:1000)
        dimension error(0:1000,0:1000)
        dimension rold(0:1000,0:1000),rnew(0:1000,0:1000)
        dimension pold(0:1000,0:1000),pnew(0:1000,0:1000)
        dimension qold(0:1000,0:1000),qnew(0:1000,0:1000)
        double precision uold,unew,error,errvalue,errmax,tol
        double precision rold,rnew,pold,pnew,qold,qnew,ar,be,c1,c2

c parameters
        dx=0.01
        dy=0.01
        nx=100
        ny=nx
        tol=0.000001
        it=0

c boundary condition
        do i=0,nx
        uold(i,0)=0.0
        unew(i,0)=0.0
        rold(i,0)=0.0
        rnew(i,0)=0.0
        pold(i,0)=0.0
        pnew(i,0)=0.0
        qold(i,0)=0.0
        qnew(i,0)=0.0
        enddo
        do i=0,nx
        uold(i,ny)=1.0
        unew(i,ny)=1.0
        rold(i,ny)=0.0
        rnew(i,ny)=0.0
        pold(i,ny)=0.0
        pnew(i,ny)=0.0
        qold(i,ny)=0.0
        qnew(i,ny)=0.0
        enddo
        do j=0,ny
        uold(0,j)=0.0
        unew(0,j)=0.0
        rold(0,j)=0.0
        rnew(0,j)=0.0
        pold(0,j)=0.0
        pnew(0,j)=0.0
        qold(0,j)=0.0
        qnew(0,j)=0.0
        enddo
        do j=0,ny
        uold(nx,j)=0.0
        unew(nx,j)=0.0
        rold(nx,j)=0.0
        rnew(nx,j)=0.0
        pold(nx,j)=0.0
        pnew(nx,j)=0.0
        qold(nx,j)=0.0
        qnew(nx,j)=0.0
        enddo

c****************************************
c Jacobi iteration
c 1      do i=1,nx-1
```

```
c        do j=1,ny-1
c         unew(i,j)=0.25*(uold(i-1,j)+uold(i+1,j)+uold(i,j-1)+uold(i,j+1))
c        enddo
c        enddo
c****************************************

c****************************************
c Gauss-Seidel iteration
c  1      do i=1,nx-1
c        do j=1,ny-1
c         unew(i,j)=0.25*(unew(i-1,j)+uold(i+1,j)+unew(i,j-1)+uold(i,j+1))
c        enddo
c        enddo
c****************************************

c****************************************
c Conjugate Gradient Method
c Step 1
         do i=1,nx-1
         do j=1,ny-1
         uold(i,j)=0.0
         enddo
         enddo

         c1=0.0
         do i=1,nx-1
         do j=1,ny-1
         rold(i,j)=uold(i-1,j)+uold(i+1,j)+uold(i,j-1)+uold(i,j+1)
     &             -4.0*uold(i,j)
         pold(i,j)=rold(i,j)
         qold(i,j)=-(rold(i-1,j)+rold(i+1,j)+rold(i,j-1)+rold(i,j+1))
     &             +4.0*rold(i,j)
         c1=c1+rold(i,j)*rold(i,j)
         enddo
         enddo

c Step 2
  1      c2=0.0
         do i=1,nx-1
         do j=1,ny-1
         c2=c2+pold(i,j)*qold(i,j)
         enddo
         enddo
         ar=c1/c2

         c2=0.0
         do i=1,nx-1
         do j=1,ny-1
         unew(i,j)=uold(i,j)+ar*pold(i,j)
         rnew(i,j)=rold(i,j)-ar*qold(i,j)
         c2=c2+rnew(i,j)*rnew(i,j)
         enddo
         enddo
         be=c2/c1
c Step 3
         do i=1,nx-1
         do j=1,ny-1
         pnew(i,j)=rnew(i,j)+be*pold(i,j)
         qnew(i,j)=be*qold(i,j)-(rnew(i-1,j)+rnew(i+1,j)+rnew(i,j-1)
     &             +rnew(i,j+1))+4.0*rnew(i,j)
         enddo
         enddo
c************************************

c Check convergence
         do i=1,nx-1
         do j=1,ny-1
```

HW. Ex.15.1, Ex1.5.2 on P.36

```
          error(i,j)=abs(unew(i,j)-uold(i,j))
          enddo
          enddo

          errmax=0.0
          do i=1,nx-1
          do j=1,ny-1
          if(error(i,j).gt.errmax) then
          errmax=error(i,j)
          endif
          enddo
          enddo

          if(errmax.le.tol) goto 2
          c1=c2
          do i=0,nx
          do j=0,ny
          uold(i,j)=unew(i,j)
          rold(i,j)=rnew(i,j)
          pold(i,j)=pnew(i,j)
          qold(i,j)=qnew(i,j)
          enddo
          enddo
          print *, "it=",it,"    ", "errmax=", errmax
          it=it+1
          goto 1

c output
 2        open(unit=6,file='solution.data')

          do i=0,nx
c          write(6,3) i*dx, unew(50,i)
          write(6,3) (unew(j,i),j=1,ny)
          enddo
c 3        format(f12.8,5x,f12.8)
 3        format(100(f12.8,1x))
          end
```

HW #2.

$-(U_{xx}+U_{yy}) = \pi^2 \sin\pi x \cdot \sin\pi y$, $0 \le x, y \le 1$

$u(x,y) = \sin\pi x \sin\pi y$, $\pi = 3.14159265\ldots$

1) $(nx \times ny)$ 100×100

b) $\text{Error}(i,j) = \max_{ij} |u_{ij} - U_{ij}| < 10^{-8}$

2a) F: l

It	J	GS	CGM

2) plot 3D