Министерство образования и науки Российской Федерации

Новосибирский государственный технический университет

Кафедра прикладной математики

Численные методы

Лабораторная работа №2

Факультет ПМИ

Группа ПМ-01

Студент Жигалов П.С.

Преподаватель Задорожный А.Г.

Персова М.Г.

Вариант 8

Новосибирск

2012

**1. Цель работы**

Разработать программы решения СЛАУ методами Якоби, Гаусса-Зейделя, блочной релаксации с хранением матрицы в диагональном формате. Исследовать сходимость методов для различных тестовых матриц и её зависимость от параметра релаксации. Изучить возможность оценки порядка числа обусловленности матрицы путем вычислительного эксперимента.

**2. Анализ**

Пусть дана система линейных алгебраических уравнений:.Выбирается начальное приближение 

***Метод Якоби***. Каждое следующее приближение в методе Якоби с параметром релаксации рассчитывается по формуле:

 где  – номер текущей итерации.

***Метод Гаусса-Зейделя.*** Каждое последующее приближение в методе Гаусса-Зейделя с параметром релаксации рассчитывается по формуле:

.

Условия выхода из итерационного процесса для рассмотренных методов:

• выход по относительной невязке:

• защита от зацикливания: ,  – максимальное количество итераций.

*Метод блочной релаксации.* Исходная матрица  разбивается на квадратные блоки равной размерности. Вектор правой части и вектор неизвестных разбиваются на блок-векторы соответствующей размерности.

Решим СЛАУ:

  

Решение полученных систем рекомендуется выполнять с использованием факторизации матрицы , причём факторизацию следует выполнять 1 раз перед первой итерацией.

**3. Текст программы**

program main

implicit none

common/matrix/n,m,eps,maxiter

integer n,m,vect,res,maxiter,ia(9)

real\*8 mem(40960),eps

open(10,file='../input.txt',status='old')

call read\_(mem,ia,vect,res)

close(10)

call calc\_jacobi(mem(1),ia,mem(vect+1),mem(res+1),1.0d00)

!call calc\_gauss\_seidel(mem(1),ia,mem(vect+1),mem(res+1),1.0d00)

!call calc\_block\_relaxation(mem(1),ia,mem(vect+1),mem(res+1),1.0d0)

open(20,file='../output.txt',status='unknown')

call write\_(mem(res+1))

close(20)

end

subroutine read\_(mem,ia,vect,res)

implicit none

common/matrix/n,m,eps,maxiter

integer n,m,i,vect,res,maxiter,ia(9)

real\*8 mem(\*),eps

read(10,\*) n,m,eps,maxiter

ia(1)=0

read(10,\*) (mem(ia(1)+i), i=1,n)

ia(2)=ia(1)+n

read(10,\*) (mem(ia(2)+i), i=1,n-1)

ia(3)=ia(2)+n-1

read(10,\*) (mem(ia(3)+i), i=1,n-m-2)

ia(4)=ia(3)+n-m-2

read(10,\*) (mem(ia(4)+i), i=1,n-m-3)

ia(5)=ia(4)+n-m-3

read(10,\*) (mem(ia(5)+i), i=1,n-m-4)

ia(6)=ia(5)+n-m-4

read(10,\*) (mem(ia(6)+i), i=1,n-1)

ia(7)=ia(6)+n-1

read(10,\*) (mem(ia(7)+i), i=1,n-m-2)

ia(8)=ia(7)+n-m-2

read(10,\*) (mem(ia(8)+i), i=1,n-m-3)

ia(9)=ia(8)+n-m-3

read(10,\*) (mem(ia(9)+i), i=1,n-m-4)

vect=ia(9)+n-m-4

read(10,\*) (mem(vect+i), i=1,n)

res=vect+n

read(10,\*) (mem(res+i), i=1,n)

end

subroutine write\_(mem)

implicit none

common/matrix/n,m,eps,maxiter

integer n,m,i,maxiter

real\*8 mem(n),eps

write(20,\*) (mem(i), i=1,n)

do i=1,n

print\*, mem(i)

end do

end

subroutine calc\_jacobi(a,ia,f,x,omega)

implicit none

common/matrix/n,m,eps,maxiter

integer n,m,maxiter,iter,i,ia(9)

real\*8 a(\*),f(n),x(n),y(n),eps,omega,residual

iter=0

do while(residual(a,ia,x,f).ge.eps.and.iter.lt.maxiter)

do i=1,n

y(i)=x(i)

end do

call next\_iter(a,ia,f,y,x,y,omega)

iter=iter+1

if(mod(iter,10).eq.0) print\*,iter,residual(a,ia,x,f)

end do

if(iter.ge.maxiter)then

print\*,'Solution can`t found! Breaking...'

else

print\*,'Solution found! Congratulations!'

end if

print\*,iter,residual(a,ia,x,f)

end

subroutine calc\_gauss\_seidel(a,ia,f,x,omega)

implicit none

common/matrix/n,m,eps,maxiter

integer n,m,maxiter,iter,i,ia(9)

real\*8 a(\*),f(n),x(n),y(n),eps,omega,residual

iter=0

do while(residual(a,ia,x,f).ge.eps.and.iter.lt.maxiter)

do i=1,n

y(i)=x(i)

end do

call next\_iter(a,ia,f,y,x,x,omega)

iter=iter+1

if(mod(iter,10).eq.0) print\*,iter,residual(a,ia,x,f)

end do

if(iter.ge.maxiter)then

print\*,'Solution can`t found! Breaking...'

else

print\*,'Solution found! Congratulations!'

end if

print\*,iter,residual(a,ia,x,f)

end

subroutine calc\_block\_relaxation(a,ia,f,x,omega)

implicit none

common/matrix/n,m,eps,maxiter

integer n,m,maxiter,iter,i,ia(9),blocksize,blocknum,ind1

real\*8 a(\*),f(n),x(n),y(n),eps,omega,residual\_lu

iter=0

blocksize=m+2

do while(blocksize.ge.2.and.mod(n,blocksize).ne.0)

blocksize=blocksize-1

end do

if(blocksize.ge.2)then

blocknum=n/blocksize

print\*, 'Detected:',blocksize,'- size,',blocknum,'- number.'

do i=0,blocknum-1

ind1=1+i\*blocksize

call factorize(blocksize,a(ia(1)+ind1),a(ia(2)+ind1),a(ia(6)+ind1))

end do

do while(residual\_lu(a,ia,x,f,blocksize).ge.eps.and.iter.lt.maxiter)

do i=1,n

y(i)=x(i)

end do

call next\_iter\_block(a,ia,f,x,y,omega,blocksize,blocknum)

iter=iter+1

if(mod(iter,1).eq.0) print\*,iter,residual\_lu(a,ia,x,f,blocksize)

end do

else

print\*, 'Can`t determine block size!'

end if

if(iter.ge.maxiter)then

print\*,'Solution can`t found! Breaking...'

else

print\*,'Solution found! Congratulations!'

end if

print\*,iter,residual\_lu(a,ia,x,f,blocksize)

end

subroutine next\_iter(a,ia,f,xk,xk1,y,omega)

implicit none

common/matrix/n,m,eps,maxiter

integer n,m,maxiter,i,ia(9),ind,ind1,ind2,ind6,ind7,ind8,ind9

real\*8 a(\*),eps,xk(n),xk1(n),omega,f(n),y(n)

ind1=ia(1)

do i=1,n

xk1(i)=a(ind1+i)\*xk(i)

end do

ind=1

ind2=ia(2)

do i=2,n

xk1(ind)=xk1(ind)+a(ind2+ind)\*xk(i)

ind=ind+1

end do

ind=1

ind2=ia(3)

do i=m+3,n

xk1(ind)=xk1(ind)+a(ind2+ind)\*xk(i)

ind=ind+1

end do

ind=1

ind2=ia(4)

do i=m+4,n

xk1(ind)=xk1(ind)+a(ind2+ind)\*xk(i)

ind=ind+1

end do

ind=1

ind2=ia(5)

do i=m+5,n

xk1(ind)=xk1(ind)+a(ind2+ind)\*xk(i)

ind=ind+1

end do

xk1(1)=xk(1)+omega/a(ia(1)+1)\*(f(1)-xk1(1))

ind=1

ind6=ia(6)

do i=2,m+2

xk1(i)=xk1(i)+a(ind6+ind)\*y(i-1)

xk1(i)=xk(i)+omega/a(ind1+i)\*(f(i)-xk1(i))

ind=ind+1

end do

ind7=ia(7)

ind8=ia(8)

ind9=ia(9)

xk1(m+3)=xk1(m+3)+a(ind7+1)\*y(1)+a(ind6+m+2)\*y(m+2)

xk1(m+3)=xk(m+3)+omega/a(ind1+m+3)\*(f(m+3)-xk1(m+3))

xk1(m+4)=xk1(m+4)+a(ind8+1)\*y(1)+a(ind7+2)\*y(2)+a(ind6+m+3)\*y(m+3)

xk1(m+4)=xk(m+4)+omega/a(ind1+m+4)\*(f(m+4)-xk1(m+4))

ind=1

do i=m+5,n

xk1(i)=xk1(i)+a(ind9+ind)\*y(ind)+a(ind8+ind+1)\*y(ind+1)+a(ind7+ind+2)\*y(ind+2)+a(ind6+ind+m+3)\*y(ind+m+3)

xk1(i)=xk(i)+omega/a(ind1+i)\*(f(i)-xk1(i))

ind=ind+1

end do

end

function residual(a,ia,x,f)

implicit none

common/matrix/n,m,eps,maxiter

integer n,m,maxiter,ia(9),i,ind,ind2,ind6,ind7,ind8,ind9

real\*8 eps,a(\*),x(n),f(n),y(n),residual,norm\_2

ind2=ia(1)

do i=1,n

y(i)=a(ind2+i)\*x(i)

end do

ind=1

ind2=ia(2)

do i=2,n

y(ind)=y(ind)+a(ind2+ind)\*x(i)

ind=ind+1

end do

ind=1

ind2=ia(3)

do i=m+3,n

y(ind)=y(ind)+a(ind2+ind)\*x(i)

ind=ind+1

end do

ind=1

ind2=ia(4)

do i=m+4,n

y(ind)=y(ind)+a(ind2+ind)\*x(i)

ind=ind+1

end do

ind=1

ind2=ia(5)

do i=m+5,n

y(ind)=y(ind)+a(ind2+ind)\*x(i)

ind=ind+1

end do

ind=1

ind6=ia(6)

do i=2,m+2

y(i)=y(i)+a(ind6+ind)\*x(i-1)

ind=ind+1

end do

ind7=ia(7)

ind8=ia(8)

ind9=ia(9)

y(m+3)=y(m+3)+a(ind7+1)\*x(1)+a(ind6+m+2)\*x(m+2)

y(m+4)=y(m+4)+a(ind8+1)\*x(1)+a(ind7+2)\*x(2)+a(ind6+m+3)\*x(m+3)

ind=1

do i=m+5,n

y(i)=y(i)+a(ind9+ind)\*x(ind)+a(ind8+ind+1)\*x(ind+1)+a(ind7+ind+2)\*x(ind+2)+a(ind6+ind+m+3)\*x(ind+m+3)

ind=ind+1

end do

do i=1,n

y(i)=f(i)-y(i)

end do

residual=sqrt(norm\_2(y,n)/norm\_2(f,n))

end

subroutine factorize(blocksize,di\_central,di\_up,di\_down)

implicit none

integer blocksize,i

real\*8 di\_central(blocksize),di\_up(blocksize-1),di\_down(blocksize-1)

do i=1,blocksize-1

di\_up(i)=di\_up(i)/di\_central(i)

di\_central(i+1)=di\_central(i+1)-di\_down(i)\*di\_up(i)

end do

end

subroutine solve\_block\_slae(blocksize,di\_central,di\_up,di\_down,y,x)

implicit none

integer blocksize,i

real\*8 di\_central(blocksize),di\_up(blocksize-1),di\_down(blocksize-1),y(blocksize),x(blocksize)

x(1)=x(1)/di\_central(1)

do i=2,blocksize

x(i)=(x(i)-di\_down(i-1)\*x(i-1))/di\_central(i)

end do

y(blocksize)=x(blocksize)

do i=blocksize-1,1,-1

y(i)=x(i)-di\_up(i)\*y(i+1)

end do

end

subroutine next\_iter\_block(a,ia,f,xk1,xk,omega,blocksize,blocknum)

implicit none

common/matrix/n,m,eps,maxiter

integer ia(9),blocksize,blocknum,i,k,maxiter,n,m,ind,ind1,ind2,ind3,ind4,ind5

real\*8 a(\*),f(n),xk1(n),xk(n),y(n),omega,eps,tmp

do k=1,blocknum

ind1=(k-1)\*blocksize+1

xk1(ind1)=a(ia(1)+ind1)\*xk(ind1)+a(ia(1)+ind1)\*a(ia(2)+ind1)\*xk(ind1+1)

do i=1,blocksize-2

ind2=ind1+i-1

tmp=a(ia(6)+ind2)\*xk(ind2)

tmp=tmp+(a(ia(6)+ind2)\*a(ia(2)+ind2)+a(ia(1)+ind1+i))\*xk(ind1+i)

tmp=tmp+a(ia(1)+ind1+i)\*a(ia(2)+ind1+i)\*xk(ind1+i+1)

xk1(ind1+i)=tmp

end do

ind2=ind1+blocksize-2

tmp=a(ia(6)+ind2)\*xk(ind2)

tmp=tmp+(a(ia(1)+ind2+1)+a(ia(6)+ind2)\*a(ia(2)+ind2))\*xk(ind2+1)

xk1(ind1+blocksize-1)=tmp

end do

do k=1,blocknum-1

xk1(k\*blocksize)=xk1(k\*blocksize)+a(ia(2)+k\*blocksize)\*xk(k\*blocksize+1)

end do

ind=1

do i=3+m,n

xk1(ind)=xk1(ind)+a(ia(3)+ind)\*xk(i)

ind=ind+1

end do

ind=1

do i=4+m,n

xk1(ind)=xk1(ind)+a(ia(4)+ind)\*xk(i)

ind=ind+1

end do

ind=1

do i=5+m,n

xk1(ind)=xk1(ind)+a(ia(5)+ind)\*xk(i)

ind=ind+1

end do

do i=1,blocksize

xk1(i)=omega\*(f(i)-xk1(i))

end do

call solve\_block\_slae(blocksize,a(ia(1)+1),a(ia(2)+1),a(ia(6)+1),y(1),xk1(1))

do i=1,blocksize

xk1(i)=y(i)+xk(i)

end do

ind2=1

ind3=1

ind4=1

do k=2,blocknum

ind5=(k-1)\*blocksize+1

xk1(ind5)=xk1(ind5)+a(ia(6)+ind5-1)\*xk1(ind5-1)

do i=1,blocksize

if(((k-1)\*blocksize+i).gt.(m+2))then

xk1((k-1)\*blocksize+i)=xk1((k-1)\*blocksize+i)+a(ia(7)+ind4)\*xk1(ind4)

ind4=ind4+1

end if

if(((k-1)\*blocksize+i).gt.(m+3))then

xk1((k-1)\*blocksize+i)=xk1((k-1)\*blocksize+i)+a(ia(8)+ind3)\*xk1(ind3)

ind3=ind3+1

end if

if(((k-1)\*blocksize+i).gt.(m+4))then

xk1((k-1)\*blocksize+i)=xk1((k-1)\*blocksize+i)+a(ia(9)+ind2)\*xk1(ind2)

ind2=ind2+1

end if

end do

do i=ind5,k\*blocksize

xk1(i)=omega\*(f(i)-xk1(i))

end do

call solve\_block\_slae(blocksize,a(ia(1)+ind5),a(ia(2)+ind5),a(ia(6)+ind5),y(ind5),xk1(ind5))

do i=ind5,blocksize\*k

xk1(i)=y(i)+xk(i)

end do

end do

end

function residual\_lu(a,ia,x,f,blocksize)

implicit none

common/matrix/n,m,eps,maxiter

integer n,m,maxiter,ia(9),i,ind,blocksize,ind2,ind6,ind7,ind8,ind9

real\*8 eps,a(\*),x(n),f(n),y(n),residual\_lu,norm\_2,elem

ind2=ia(1)

do i=1,n

if(mod(i,blocksize).eq.1)then

y(i)=a(ind2+i)\*x(i)

else

elem=a(ia(6)+i-1)

elem=elem\*a(ia(2)+i-1)

elem=elem+a(ind2+i)

y(i)=elem\*x(i)

end if

end do

ind=1

ind2=ia(2)

do i=2,n

if(mod(ind,blocksize).eq.0)then

elem=a(ind2+ind)

else

elem=a(ia(1)+ind)

elem=elem\*a(ind2+ind)

end if

y(ind)=y(ind)+elem\*x(i)

ind=ind+1

end do

ind=1

ind2=ia(3)

do i=m+3,n

y(ind)=y(ind)+a(ind2+ind)\*x(i)

ind=ind+1

end do

ind=1

ind2=ia(4)

do i=m+4,n

y(ind)=y(ind)+a(ind2+ind)\*x(i)

ind=ind+1

end do

ind=1

ind2=ia(5)

do i=m+5,n

y(ind)=y(ind)+a(ind2+ind)\*x(i)

ind=ind+1

end do

ind=1

ind6=ia(6)

do i=2,m+2

y(i)=y(i)+a(ind6+ind)\*x(i-1)

ind=ind+1

end do

ind7=ia(7)

ind8=ia(8)

ind9=ia(9)

y(m+3)=y(m+3)+a(ind7+1)\*x(1)+a(ind6+m+2)\*x(m+2)

y(m+4)=y(m+4)+a(ind8+1)\*x(1)+a(ind7+2)\*x(2)+a(ind6+m+3)\*x(m+3)

ind=1

do i=m+5,n

y(i)=y(i)+a(ind9+ind)\*x(ind)+a(ind8+ind+1)\*x(ind+1)+a(ind7+ind+2)\*x(ind+2)+a(ind6+ind+m+3)\*x(ind+m+3)

ind=ind+1

end do

do i=1,n

y(i)=f(i)-y(i)

end do

residual\_lu=sqrt(norm\_2(y,n)/norm\_2(f,n))

end

function norm\_2(x,n)

implicit none

integer n,i

real\*8 x(n),norm\_2

norm\_2=0.0d00

do i=1,n

norm\_2=norm\_2+x(i)\*\*2

end do

end

**4. Набор тестов**

****

Максимальное число итераций – 1000000

Целевая относительная невязка – 1.0E-16

Параметр релаксации – 1.0

|  |  |  |  |
| --- | --- | --- | --- |
| Метод | Якоби | Гаусса-Зейделя | Блочной релаксации |
| Решение | 0.999999999999999  1.999999999999997  2.999999999999999  3.999999999999998  5.000000000000000  6.000000000000000  7.000000000000000  8.000000000000000  9.000000000000000  10.000000000000000 | 1.000000000000002  1.999999999999999  3.000000000000000  4.000000000000000  5.000000000000001  5.999999999999999  6.999999999999999  8.000000000000000  9.000000000000000  10.000000000000000 | 1.000000000000000  2.000000000000000  3.000000000000001  4.000000000000000  5.000000000000000  5.999999999999999  7.000000000000000  8.000000000000000  9.000000000000000  10.000000000000000 |
| Невязка | 0.564E-16 | 0.636E-16 | 0.208E-16 |
| Количество итераций | 44 | 17 | 14 |

**5. Исследования**

Вид исследуемых матриц:





Начальное приближение – нулевой вектор.

Максимальное число итераций – 1000000

Целевая относительная невязка – 1.0E-14

Метод Якоби для матрицы A (слева) и для матрицы B (справа)



Метод Якоби для матрицы A (слева) и для матрицы B (справа)



Метод Якоби для матрицы A (слева) и для матрицы B (справа)



Метод Якоби для матрицы A (слева) и для матрицы B (справа)



Метод Якоби для матрицы A (слева) и для матрицы B (справа)



Метод Якоби для матрицы A (слева) и для матрицы B (справа)



Метод Гаусса-Зейделя для матрицы A (слева) и для матрицы B (справа)



Метод Гаусса-Зейделя для матрицы A (слева) и для матрицы B (справа)



Метод Гаусса-Зейделя для матрицы A (слева) и для матрицы B (справа)



Метод Гаусса-Зейделя для матрицы A (слева) и для матрицы B (справа)



Метод Гаусса-Зейделя для матрицы A (слева) и для матрицы B (справа)



Метод Гаусса-Зейделя для матрицы A (слева) и для матрицы B (справа)



Метод Гаусса-Зейделя для матрицы A (слева) и для матрицы B (справа)



Метод блочной релаксации для матрицы A (слева) и для матрицы B (справа)



Метод блочной релаксации для матрицы A (слева) и для матрицы B (справа)



Метод блочной релаксации для матрицы A (слева) и для матрицы B (справа)



Метод блочной релаксации для матрицы A (слева) и для матрицы B (справа)



Метод блочной релаксации для матрицы A (слева) и для матрицы B (справа)



Метод блочной релаксации для матрицы A (слева) и для матрицы B (справа)



Метод блочной релаксации для матрицы A (слева) и для матрицы B (справа)



Метод блочной релаксации для матрицы A (слева) и для матрицы B (справа)



**6. Выводы**

Исследование по определению оптимального веса показали, что оптимальный вес различен для каждой матрицы и каждого метода, при этом иногда оптимальный вес лежит за пределами рекомендуемых параметров (например, для метода Якоби и матрицы A). Можно проследить закономерность, что при увеличении параметра релаксации уменьшается число шагов для получения верного ответа (до некоторого предела), а затем достаточно резкое ухудшение, вплоть до полной невозможности получить верный ответ.

Оценка числа обусловленности с помощью невязки и погрешности показала, что число обусловленности матрицы 80.852211, а число обусловленности матрицы 8.832233.