Министерство образования и науки Российской Федерации

НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Кафедра прикладной математики

Лабораторная работа № 4

по дисциплине

«Уравнения математической физики»

Факультет прикладной математики и информатики

Группа ПМ-01

Студент Курочкин А.В.

Преподаватель Задорожный А.Г.

Персова М.Г.

Вариант 3

Новосибирск

2013

1. Цель работы

Изучить особенности реализации итерационных методов BCG, BCGStab, GMRES для СЛАУ с несимметричными разреженными матрицами. Исследовать влияние предобусловливания на сходимость этих методов.

1. Задание

Решить СЛАУ методом GMRES с LU-предобусловливанием.

1. Анализ

Выбирается начальное приближение:



Далее выполняются итерации метода, на каждой итерации вычисляются элементы вспомогательных матриц V и H размера nm и (m+1)m, где n – размерность решаемой СЛАУ, а m – глубина метода.

Пусть vi – i-й столбец матрицы V.



Далее для μ=1,…,m выполняется:



Если , то процесс построения H заканчивается, и m=, иначе



Далее ищется новое приближение преобразованной задачи.

, размера m+1.



1. Текст программ

GMRES.h

#ifndef GMRES\_HMFE\_H\_

#define GMRES\_HMFE\_H\_

#include <math.h>

#include <iostream>

#include <string.h>

using namespace std;

class GMRES

{

private:

int n;

int \*ig, \*jg;

double \*gu, \*gl, \*di;

double \*rp;

double \*Uu, \*Ll, \*Ld;

void precond();

double dot\_prod(double \*a, double \*b);

void mull\_A(double \*f, double \*&x);

void SolutionSLAE\_L(double \*f, double \*&x);

void SolutionSLAE\_U(double \*f, double \*&x);

void mult\_V\_y(int mr,double \*res);

void insert\_col(int j,double \*x);

void calc\_h\_w(int i,int j);

void calc\_HtH(int mr);

void gauss(double \*M,double \*vector,int mr);

double \*h,\*h2,\*v,\*w,\*y;

int m;

public:

void init(int\* s\_ig, int\* s\_jg, double\* s\_gu, double\* s\_gl, double\* s\_di, int s\_n);

void setF(double\* s\_rp);

void SolutionSLAE(double \*&solution, int &its);

};

#endif

GMRES.cpp

#include "GMRES.h"

void GMRES::init(int \*s\_ig, int \*s\_jg, double \*s\_gu, double \*s\_gl, double \*s\_di, int s\_n)

{

ig = s\_ig;

jg = s\_jg;

gu = s\_gu;

gl = s\_gl;

di = s\_di;

n = s\_n;

precond();

}

void GMRES::setF(double \*s\_rp) rp = s\_rp;

void GMRES::precond()

{

double sum\_l, sum\_u, sum\_d;

int copy\_end = ig[n];

Ll = new double [copy\_end];

Uu = new double [copy\_end];

Ld = new double [n];

for(int i = 0; i < copy\_end; i++)

{

Ll[i] = gl[i];

Uu[i] = gu[i];

}

for(int i = 0; i < n; i++) Ld[i] = di[i];

for(int k = 1, k1 = 0; k <= n; k++, k1++)

{

sum\_d = 0;

int i\_s = ig[k1], i\_e = ig[k];

for(int m = i\_s; m < i\_e; m++)

{

sum\_l = 0;

sum\_u = 0;

int j\_s = ig[jg[m]], j\_e = ig[jg[m] + 1];

for(int i = i\_s; i < m; i++)

{

for(int j = j\_s ; j < j\_e; j++)

{

if(jg[i] == jg[j])

{

sum\_l += Ll[i] \* Uu[j];

sum\_u += Ll[j] \* Uu[i];

j\_s++;

}

}

}

Ll[m] = Ll[m] - sum\_l;

Uu[m] = (Uu[m] - sum\_u) / Ld[jg[m]];

sum\_d += Ll[m] \* Uu[m];

}

Ld[k1] = Ld[k1] - sum\_d;

}

}

double GMRES::dot\_prod(double \*a, double \*b)

{

double dp = 0;

for(int i = 0; i < n; i++)

dp += a[i] \* b[i];

return dp;

}

void GMRES::mull\_A(double \*f, double \*&x)

{

for(int i = 0; i < n; i++)

{

double v\_el = f[i];

x[i] = di[i] \* v\_el;

for(int k = ig[i], k1 = ig[i+1]; k < k1; k++)

{

int j = jg[k];

x[i] += gl[k] \* f[j];

x[j] += gu[k] \* v\_el;

}

}

}

void GMRES::SolutionSLAE\_L(double \*f, double \*&x)

{

for(int k = 1, k1 = 0; k <= n; k++, k1++)

{

double sum = 0;

for(int i = ig[k1]; i < ig[k]; i++)

sum += Ll[i] \* x[jg[i]];

x[k1] = (f[k1] - sum) / Ld[k1];

}

}

void GMRES::SolutionSLAE\_U(double \*f, double \*&x)

{

double\* f1 = new double [n];

for(int i = 0; i < n; i++)

f1[i] = f[i];

for(int k = n, k1 = n - 1; k > 0; k--, k1--)

{

x[k1] = f1[k1] / Ld[k1];

double v\_el = x[k1];

for(int i = ig[k1]; i < ig[k]; i++)

f1[jg[i]] -= Uu[i] \* v\_el;

}

delete[] f1;

}

void GMRES::SolutionSLAE(double \*&solution, int &its)

{

int maxiter = 1000;

double eps = 1E-14;

m = 3;

double rp\_norm = sqrt(dot\_prod(rp, rp));

solution = new double [n];

for(int i = 0; i < n; i++)

solution[i] = 0.0;

double \*tmp = new double[n];

double \*hlp = new double[n];

w = new double[n];

y = new double[m];

v = new double[m\*n];

h = new double[m\*(m+1)];

h2 = new double[m\*m];

int j,newm = m,flag = 0,k = 0;

double norma\_r,h\_last;

long int i;

mull\_A(solution,tmp);

for(i = 0; i < n; i++) w[i] = rp[i] - tmp[i];

SolutionSLAE\_L(w, hlp);

norma\_r = sqrt(dot\_prod(hlp,hlp));

while(norma\_r / rp\_norm > eps && k<maxiter)

{

k++;

flag = 0;

for(i = 0; i < n; i++) hlp[i] /= norma\_r;

insert\_col(0,hlp);

for(j = 0; j < newm && !flag; j++)

{

SolutionSLAE\_U(hlp,tmp);

mull\_A(tmp,hlp);

SolutionSLAE\_L(hlp, w);

for(i = 0; i <= j; i++) calc\_h\_w(i,j);

h[(j + 1) \* m + j]=sqrt(dot\_prod(w,w));

h\_last=h[(j + 1)\* m + j];

if(fabs(h\_last) > eps \* eps && j != newm - 1)

{

for(i = 0; i < n; i++) hlp[i] = w[i] / h\_last;

insert\_col(j+1,hlp);

}

if(fabs(h\_last) < eps \* eps)

{

newm = j + 1;

flag = 1;

}

}

if(flag)

{

y[0] = norma\_r;

for(i = 1; i < newm; i++) y[i] = 0;

gauss(h,y,newm);

}

else

{

for(i=0; i<newm; i++) y[i] = h[i] \* norma\_r;

calc\_HtH(newm);

gauss(h2,y,newm);

}

mult\_V\_y(newm,tmp);

SolutionSLAE\_U(tmp,hlp);

for(i = 0; i < n; i++) solution[i] += hlp[i];

mull\_A(solution,tmp);

for(i = 0; i < n; i++) w[i] = rp[i] - tmp[i];

SolutionSLAE\_L(w, hlp);

norma\_r=sqrt(dot\_prod(hlp,hlp));

}

its = k;

delete[] hlp;

delete[] tmp;

delete[] w;

delete[] y;

delete[] v;

delete[] h;

delete[] h2;

}

void GMRES::mult\_V\_y(int mr,double \*res)

{

long int k;

int p;

for(k = 0; k < n; k++)

{

res[k] = 0;

for(p = 0; p < mr; p++)

res[k] += v[k \* m + p] \* y[p];

}

}

void GMRES::insert\_col(int j,double \*x)

{

for(long int i = 0; i < n; i++) v[i \* m + j] = x[i];

}

void GMRES::calc\_h\_w(int i,int j)

{

int ind=i \* m + j;

long int k;

h[ind] = 0;

for(k = 0; k < n; k++) h[ind] += w[k] \* v[k \* m + i];

for(k = 0; k < n; k++) w[k] -= h[ind] \* v[k \* m + i];

}

void GMRES::calc\_HtH(int mr)

{

int i,j,k,idx;

memset(h2,0.,m\*m\*sizeof(double));

for(i = 0; i < mr; i++)

for(j = 0; j < mr; j++)

{

idx=i \* m + j;

for(k = 0; k <=m r; k++)

h2[idx] += h[k \* m + i] \* h[k \* m + j];

}

}

void GMRES::gauss(double \*M, double \*vector, int mr)

{

for(int i = 0; i < mr; i++)

{

int ind\_di = i \* (m + 1);

int ind\_maxstr = i;

double \*lead, temp, max;

max = fabs(M[ind\_di]);

for(int j = 1; j < mr - i; j++)

if(fabs(M[ind\_di + j \* m]) > max)

{

max = fabs(M[ind\_di + j \* m]);

ind\_maxstr = i + j;

}

for(int j = i; j < mr; j++)

{

temp = M[i \* m + j];

M[i \* m + j] = M[ind\_maxstr \* m + j];

M[ind\_maxstr \* m + j] = temp;

}

temp = vector[i];

vector[i] = vector[ind\_maxstr];

vector[ind\_maxstr] = temp;

lead = &M[ind\_di];

for(int j = 1; j < mr - i; j++)

\*(lead + j) /= (\*lead);

vector[i] /= (\*lead);

\*lead = 1;

for(int j = i + 1; j < mr; j++)

{

for(int k = i + 1; k < mr; k++)

M[j \* m + k] = M[j \* m + k] - M[i \* m + k] \* M[j \* m + i];

vector[j] = vector[j] - M[j \* m + i] \* vector[i];

M[j \* m + i] = 0.0;

}

}

for(int i = mr - 1; i >= 0; i--)

for(int j = mr - 1; j > i; j--)

vector[i] = vector[i] - M[i \* m + j] \* vector[j];

}

1. Тесты

Расчётная область – куб [0,1]x[0,1]x[0,1], шаг 0,2, 125 КЭ

1. us=x3+y3+z3, uc= 2x3-y3+3z3, ω=1000, λ=1000, σ=3, χ=10-11,

fs=- ω σ uc- ω2 χ us-6 λ(x+y+z), fs= ω σ us- ω2 χ uc-6 λ(2x-y+3z)

LU:

Погрешность – 1.536e-016

Время – 0.039235 с

LOS:

Погрешность – 3.235e-016

Время – 0.077221 с

Итераций – 50

GMRES:

Погрешность – 5.533e-015

Время – 0.035244

Итераций – 8

1. us=x+y+z, uc= 2x-y+3z, ω=1000, λ=1000, σ=3, χ=10-11,

fs=- ω σ uc- ω2 χ us, fs= ω σ us- ω2 χ uc

LU:

Погрешность – 1.966e-016

Время – 0.039716 с

LOS:

Погрешность – 3.747e-016

Время – 0.079993

Итераций – 54

GMRES:

Погрешность – 5.641e-014

Время – 0.036652

Итераций – 8

1. us= sin(x+y+z);, uc= sin(x-y-z), ω=1000, λ=1000, σ=3, χ=10-11,

fs=- ω σ uc- ω2 χ us+3λ sin(x+y+z), fs= ω σ us- ω2 χ uc+3λ sin(x-y-z)

LU:

Погрешность – 1.983e-004

Время – 0.039381 с

LOS:

Погрешность – 1.983e-004

Время – 0.076187

Итераций – 50

GMRES:

Погрешность – 4.184e-004

Время – 0.035484

Итераций – 8

1. Исследования на сетке с небольшим числом КЭ

Используется первый тест, расчётная область прежняя, шаг 0,1 по всем осям, число КЭ 1000

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  | LU, время, с | LOS, iter | LOS, время, с | GMRES, iter | GMRES, время, с |
| 10-4 | 5 103 | 104 | 10-11 | 2.291 | 136 | 1.237 | 16 | 0. 569 |
| 102 | 5 103 | 104 | 10-11 | 2.290 | 583 | 4.631 | 18 | 0. 711 |
| 109 | 5 103 | 104 | 10-11 | 2.288 | 6818 | 51.028 | 5000 | 46.902 |
| 102 | 102 | 104 | 10-11 | 2.289 | 3168 | 23.733 | 264 | 2.640 |
| 102 | 105 | 104 | 10-11 | 2.287 | 344 | 2.778 | 19 | 0.309 |
| 102 | 5 103 | 0 | 10-11 | 2.288 | 107 | 1.031 | 16 | 0. 257 |
| 102 | 5 103 | 106 | 10-11 | 2.294 | 6803 | 51.203 | 2603 | 24.457 |
| 102 | 5 103 | 104 | 8.81 10-12 | 2.340 | 455 | 3.682 | 19 | 0.267 |
| 102 | 5 103 | 104 | 10-10 | 2.284 | 455 | 3.620 | 19 | 0.279 |

1. Исследования на сетке с большим числом КЭ

Используется первый тест, расчётная область прежняя, шаги 0.025, 0.025 и 0.05 – 32000 КЭ

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  | LU, время, с | LOS, iter | LOS, время, с | GMRES, iter | GMRES, время, с |
| 10-4 | 5 103 | 104 | 10-11 | 303.683 | 5463 | 254.017 | 205 | 141.963 |
| 102 | 5 103 | 104 | 10-11 | 301.621 | 5530 | 250.353 | 361 | 147.069 |
| 102 | 105 | 104 | 10-11 | 305.761 | 3443 | 232.758 | 173 | 144.481 |
| 102 | 5 103 | 0 | 10-11 | 302.253 | 1723 | 218.032 | 205 | 172.089 |
| 102 | 5 103 | 104 | 8.81 10-12 | 302.340 | 4593 | 242.956 | 361 | 157.902 |
| 102 | 5 103 | 104 | 10-10 | 302.284 | 4519 | 237.210 | 361 | 295.722 |

В результате проведённых исследований мы можем характеризовать GMRES как достаточно надёжный и относительно быстрый метод решения СЛАУ, который может эффективно (относительно ЛОС) решать СЛАУ большой размерности.