Министерство образования и науки Российской Федерации

Новосибирский государственный технический университет

Кафедра экономической информатики

Курсовая работа

по дисциплине «Численные методы»

на тему: «Метод квадратных корней для симметричной матрицы при решении СЛАУ»

Новосибирск, 2010

Содержание

Введение

1. Математическая постановка задачи

2. Описание программного обеспечения

3. Описание тестовых задач

4. Анализ результатов. Выводы

Заключение

Список использованной литературы

**Введение**

# В данной работе мы будем исследовать метод квадратных корней для симметричной матрицы при решении систем линейных алгебраических уравнений (СЛАУ).

В жизни, очень часто приходится описывать состояние различных объектов, в том числе и экономических с помощью математических моделей. После того, как объект описан такой моделью, очень часто необходимо найти его состояние равновесия.

Именно тогда, чтобы найти это состояние, приходится решать систему алгебраических уравнений. В нашем случае система состоит из n линейных уравнений с n неизвестными, и ее можно описать так:



Также данную систему можно записать и в матричном виде:

Тогда мы будем иметь матрицу коэффициентов А:

,



столбец свободных членов уравнений f:

,



и столбец неизвестных х:

.



Чтобы данная СЛАУ имела единственное решение, нужно, чтобы определитель матрицы коэффициентов А не был равен нулю (det(A))≠0.

Данную систему можно решить многими методами. Например, методом Гаусса. Решение этой системы методом Гаусса потребует выполнить

действий,



где n – число неизвестных в уравнении. А это довольно таки трудоемко, особенно при больших порядках числа n.

Еще одним точным методом для решения данных СЛАУ является рассматриваемый в данной работе метод квадратных корней для симметричной матрицы А.

Изучать данный метод мы будем следующим образом. Сначала рассмотрим математическую постановку задачи для метода квадратных корней при решении СЛАУ. В данном разделе будет полностью описана математическая модель метода. Затем рассматривается разработанная реализация данного метода в среде MatLab 7.0. После того, как метод будет реализован, можно провести анализ точности этого метода. Анализ будет основываться на исследовании влияния мерности матрицы А, ее обусловленности, разреженности на точность полученного решения. По результатам исследования будет приведен график зависимости точности полученного решения от мерности матрицы А.

метод решение корень симметричная матрица

**1. Математическая постановка задачи**

Метод квадратных корней используется для решения линейной системы вида Ах=f (1.1), в которой матрица А является симметричной, т.е. аij=aji , где (i, j = 1, 2, …, n).

Данный метод является более экономным и удобным по сравнению с решением систем общего вида. Решение системы осуществляется в два этапа.

*Прямой ход*. Представим матрицу А в виде произведения двух взаимно транспонированных треугольных матриц:

А = Т′ Т, (1.2)

где , а .



Перемножая матрицы T′ и T и приравнивая матрице A, получим следующие формулы для определения tij:

(1.3)



После того, как матрица Т найдена, систему (1.1) заменяем двумя эквивалентными ей системами с треугольными матрицами

T′y = b, Tx = y. (1.4)

*Обратный ход*. Записываем в развернутом виде системы (1.4):

(1.5)



(1.6)



И из этих систем (1.5) и (1.6) последовательно находим

(1.7)



При вычислениях применяется обычный контроль с помощью сумм, причем при составлении суммы учитываются все коэффициенты соответствующей строки.

Заметим, что при действительных aij могут получиться чисто мнимые tij. Метод применим и в этом случае.

Метод квадратных корней дает большой выигрыш во времени по сравнению с другими методами (например, методом Гаусса), так как, во-первых, существенно уменьшает число умножений и делений (почти в два раза для больших n), во-вторых, позволяет накапливать сумму произведений без записи промежуточных результатов.

Всего метод квадратных корней требует



операций умножения и деления (примерно в два раза меньше, чем метод Гаусса), а также n операций извлечения корня.

# **2. Описание программного обеспечения**

Метод квадратных корней был реализован через функцию function [e,x]=mkk(a,f) , с входными переменными а и f и выходными e и х, где

а – матрица коэффициентов А,

f – столбец свободных членов,

х – столбец найденных решений,

е – столбец ошибок.

Столбец ошибок вычисляется, как Е=А\*х-f.

Текст функции на языке MatLab:

function [e,x]=mkk(a,f)

f=f'; %столбец f переводим в строку

n=size(a,1); % вычисляем мерность матрицы А

if (a==a')

if (det(a)~=0) % проверяем, чтобы система имела единственное решение

if (size(f',1)==n) %проверяем соответствует ли мерность матрицы А мерности вектора f

t=zeros(n); %создаем матрицу элементов T и заполняем ее нулями

t(1,1)=sqrt(a(1,1)); % 1.3

for k=2:n

t(1,k)=a(1,k)/t(1,1);

end

for j=2:n

for i=2:n

if (i==j)

c=0;

for k=1:(i-1)

c=c+t(k,i)^2;

end

t(i,i)=sqrt(a(i,i)-c);

else

if (i<j)

c=0;

for k=1:(i-1)

c=c+t(k,i)\*t(k,j);

end

t(i,j)=(a(i,j)-c)/t(i,i);

end

end

end

end

y=zeros(n,1); %1.7 создаем столбец у

y(1)=f(1)/t(1,1);

for i=2:n

c=0;

for k=1:(i-1)

c=c+t(k,i)\*y(k);

end

y(i)=(f(i)-c)/t(i,i);

end

x=zeros(n,1); %создаем столбец точных решений

e=zeros(n,1); % создаем столбец ошибок

x(n)=y(n)/t(n,n); %1.8 вычисляем вектор Х

for i=(n-1):-1:1

c=0;

for k=(i+1):n

c=c+t(i,k)\*x(k);

end

x(i)=(y(i)-c)/t(i,i);

e=a\*x-f';

end

else

error('Внимание! Ошибка! Размерность матрицы А не соответствует размерности вектора F');

end

else

error('Внимание! Ошибка! Определитель матрицы А равен 0')

end

else

f=f\*a';

a=a\*a';

if (det(a)~=0) % проверяем, чтобы система имела единственное решение

if size(f',1)==n %проверяем соответствует ли мерность матрицы А мерности вектора f

t=zeros(n); %создаем матрицу элементов T и заполняем ее нулями

t(1,1)=sqrt(a(1,1)); % 1.3

for k=2:n

t(1,k)=a(1,k)/t(1,1);

end

for j=2:n

for i=2:n

if (i==j)

c=0;

for k=1:(i-1)

c=c+t(k,i)^2;

end

t(i,i)=sqrt(a(i,i)-c);

else

if (i<j)

c=0;

for k=1:(i-1)

c=c+t(k,i)\*t(k,j);

end

t(i,j)=(a(i,j)-c)/t(i,i);

end

end

end

end

y=zeros(n,1);

y(1)=f(1)/t(1,1);

for i=2:n

c=0;

for k=1:(i-1)

c=c+t(k,i)\*y(k);

end

y(i)=(f(i)-c)/t(i,i);

end

x=zeros(n,1);

x(n)=y(n)/t(n,n);

for i=(n-1):-1:1

c=0;

for k=(i+1):n

c= c+t (i,k)\*x (k);

end

x (i) = (y(i)-c) /t (i,i);

end

else

error ('Внимание! Ошибка! Размерность вектора F не соответствует размерности матрицы А');

end

else

error ('Внимание! Ошибка! Определитель матрицы А равен 0');

end

end

# 

# **3. Описание тестовых задач**

После того, как функция была разработана, для ее отладки была составлена программа, где задавались матрица А, вектор f и откуда вызывалась написанная функция.

Программа имеет вид:

a=[1 0 0; 0 1 0; 0 0 1];

f=[7;8;9];

[e,x]=mkk(a,f)

Решение для данной программы выдано такое:

e =

0

0

0

x =

7

8

9

Как видим, решение правильное.

Начнем исследование метода квадратных корней. Для начала исследуем влияние мерности матрицы А на точность решения.

Для этого будем последовательно решать СЛАУ, каждый раз увеличивая мерность А. Для этого составим такую программу, которая

а) решит четыре СЛАУ с разными мерностями матрицы А,

б) посчитает четыре точности полученного решения по формуле E1=max |Ei|,

в) посчитает четыре точности полученного решения по формуле

,



в которых i – количество решенных уравнений

г) построит два графика зависимости точностей полученного решения от мерности матрицы А.

Текст программы:

e1=0;

e2=0;

a=[1 0.42;.42 1]

f=[0.3;0.5]

[e,x]=mkk(a,f)

e1=max(abs(e))

e2=sqrt(sum(power(e,2)))

a=[1 0.42 .54;.42 1 .32; .54 .32 1;]

f=[0.3;0.5;.7]

[e,x]=mkk(a,f)

e1=[e1 max(abs(e))]

e2=[e2 sqrt(sum(power(e,2)))]

a=[1 0.42 .54 .66;.42 1 .32 .44; .54 .32 1 .22; .66 .44 .22 1]

f=[0.3;0.5;.7;.9]

[e,x]=mkk(a,f)

e1=[e1 max(abs(e))]

e2=[e2 sqrt(sum(power(e,2)))]

a=[1 0.42 .54 .66 .53;.42 1 .32 .44 .45; .54 .32 1 .22 .41; .66 .44 .22 1 .25; .53 .45 .41 .25 1;]

f=[0.3;0.5;.7;.9;.6]

[e,x]=mkk(a,f)

e1=[e1 max(abs(e))]

e2=[e2 sqrt(sum(power(e,2)))]

mernost=[2 3 4 5];

plot(mernost,e1);

pause;

plot(mernost,e2);

pause

Результат работы программы:

>> head5

a =

1.0000 0.4200

0.4200 1.0000

f =

0.3000

0.5000

e =

0

0

x =

0.1093

0.4541

e1 =

0

e2 =

0

a =

1.0000 0.4200 0.5400

0.4200 1.0000 0.3200

0.5400 0.3200 1.0000

f =

0.3000

0.5000

0.7000

e =

1.0e-016 \*

0.5551

0

0

x =

-0.2405

0.3737

0.7103

e1 =

1.0e-016 \*

0 0.5551

e2 =

1.0e-016 \*

0 0.5551

a =

1.0000 0.4200 0.5400 0.6600

0.4200 1.0000 0.3200 0.4400

0.5400 0.3200 1.0000 0.2200

0.6600 0.4400 0.2200 1.0000

f =

0.3000

0.5000

0.7000

0.9000

e =

1.0e-015 \*

-0.0555

0

-0.2220

0

x =

-1.2578

0.0435

1.0392

1.4824

e1 =

1.0e-015 \*

0 0.0555 0.2220

e2 =

1.0e-015 \*

0 0.0555 0.2289

a =

1.0000 0.4200 0.5400 0.6600 0.5300

0.4200 1.0000 0.3200 0.4400 0.4500

0.5400 0.3200 1.0000 0.2200 0.4100

0.6600 0.4400 0.2200 1.0000 0.2500

0.5300 0.4500 0.4100 0.2500 1.0000

f =

0.3000

0.5000

0.7000

0.9000

0.6000

e =

1.0e-015 \*

0.0555

0.2220

-0.1110

-0.3331

0

x =

-1.6362

-0.1885

0.9761

1.6642

0.7358

e1 =

1.0e-015 \*

0 0.0555 0.2220 0.3331

e2 =

1.0e-015 \*

0 0.0555 0.2289 0.4191

Построенные графики для оценки точности решения:

Для E1=max |Ei|,



Для



Как видим из решения, выданного программой, а также из графиков, ошибка растет с увеличением мерности матрицы А, а точность решения, как следствие уменьшается.

Теперь исследуем влияние разреженности матрицы А на точность решения. Для этого немного модифицируем программу, использованную для исследования влияния мерности матрицы А на точность решения: изменим в ней СЛАУ для решения. На каждом шаге будем увеличивать количество нулевых элементов в матрице.

Текст программы:

e1=0;

e2=0;

a=[1 0.42 .54 .66 .53;.42 1 .32 .44 .45; .54 .32 1 .22 .41; .66 .44 .22 1 .25; .53 .45 .41 .25 1;]

f=[0.3;0.5;.7;.9;.6]

[e,x]=mkk(a,f)

e1=max(abs(e))

e2=sqrt(sum(power(e,2)))

a=[1 0 .54 0 .53;0 1 .32 .44 .45; .54 .32 1 .22 .41; 0 .44 .22 1 .25; .53 .45 .41 .25 1;]

f=[0.3;0.5;.7;.9;.6]

[e,x]=mkk(a,f)

e1=[e1 max(abs(e))]

e2=[e2 sqrt(sum(power(e,2)))]

a=[1 0 .54 0 .53;0 1 .32 .44 .45; .54 .32 1 .22 .41; 0 .44 .22 1 0; .53 .45 .41 0 1;]

f=[0.3;0.5;.7;.9;.6]

[e,x]=mkk(a,f)

e1=[e1 max(abs(e))]

e2=[e2 sqrt(sum(power(e,2)))]

a=[1 0 .54 0 0;0 1 0 .44 .45; .54 0 1 .22 0; 0 .44 .22 1 0; 0 .45 0 0 1;]

f=[0.3;0.5;.7;.9;.6]

[e,x]=mkk(a,f)

e1=[e1 max(abs(e))]

e2=[e2 sqrt(sum(power(e,2)))]

mernost=[2 3 4 5];

plot(mernost,e1);

pause;

plot(mernost,e2);

pause

Результат работы программы:

a =

1.0000 0.4200 0.5400 0.6600 0.5300

0.4200 1.0000 0.3200 0.4400 0.4500

0.5400 0.3200 1.0000 0.2200 0.4100

0.6600 0.4400 0.2200 1.0000 0.2500

0.5300 0.4500 0.4100 0.2500 1.0000

f =

0.3000

0.5000

0.7000

0.9000

0.6000

e =

1.0e-015 \*

0.0555

0.2220

-0.1110

-0.3331

0

x =

-1.6362

-0.1885

0.9761

1.6642

0.7358

e1 =

3.3307e-016

e2 =

4.1910e-016

a =

1.0000 0 0.5400 0 0.5300

0 1.0000 0.3200 0.4400 0.4500

0.5400 0.3200 1.0000 0.2200 0.4100

0 0.4400 0.2200 1.0000 0.2500

0.5300 0.4500 0.4100 0.2500 1.0000

f =

0.3000

0.5000

0.7000

0.9000

0.6000

e =

1.0e-015 \*

0.0555

0.1110

0.2220

0.1110

0.1110

x =

-0.1810

-0.1718

0.5355

0.7673

0.3618

e1 =

1.0e-015 \*

0.3331 0.2220

e2 =

1.0e-015 \*

0.4191 0.2989

a =

1.0000 0 0.5400 0 0.5300

0 1.0000 0.3200 0.4400 0.4500

0.5400 0.3200 1.0000 0.2200 0.4100

0 0.4400 0.2200 1.0000 0

0.5300 0.4500 0.4100 0 1.0000

f =

0.3000

0.5000

0.7000

0.9000

0.6000

e =

1.0e-015 \*

-0.0555

-0.0555

0

0.1110

0

x =

-0.4156

-0.4724

0.5213

0.9932

0.8192

e1 =

1.0e-015 \*

0.3331 0.2220 0.1110

e2 =

1.0e-015 \*

0.4191 0.2989 0.1360

a =

1.0000 0 0.5400 0 0

0 1.0000 0 0.4400 0.4500

0.5400 0 1.0000 0.2200 0

0 0.4400 0.2200 1.0000 0

0 0.4500 0 0 1.0000

f =

0.3000

0.5000

0.7000

0.9000

0.6000

e =

1.0e-015 \*

0

0

0

0

-0.1110

x =

0.0374

-0.1969

0.4863

0.8797

0.6886

e1 =

1.0e-015 \*

0.3331 0.2220 0.1110 0.1110

e2 =

1.0e-015 \*

0.4191 0.2989 0.1360 0.1110

Для E1=max |Ei|,



Для



Как видим из решения и графиков, величина ошибок уменьшается, а точность найденного решения увеличивается с увеличением количества нулевых элементов в матрице А. Это связано с тем, что увеличение числа нулевых элементов постепенно уменьшает число ненулевых элементов задействованных в вычислениях.

Теперь исследуем влияние обусловленности матрицы А на точность получаемого решения. Для этого в третий раз модифицируем нашу программу. Теперь мы будем брать обусловленные матрицы, с каждым шагом увеличивая их размерность.

Текст программы:

e1=0;

e2=0;

a=[500 501;501 500]

f=[15000;16000]

[e,x]=mkk(a,f)

e1=max(abs(e))

e2=sqrt(sum(power(e,2)))

a=[500 501 -503;501 500 499;-503 499 500]

f=[15000;16000;18000]

[e,x]=mkk(a,f)

e1=[e1 max(abs(e))]

e2=[e2 sqrt(sum(power(e,2)))]

a=[500 501 -503 500;501 500 499 -501;-503 499 500 502;500 -501 502 500]

f=[15000;16000;18000;16000]

[e,x]=mkk(a,f)

e1=[e1 max(abs(e))]

e2=[e2 sqrt(sum(power(e,2)))]

a=[500 501 -503 500 499;501 500 499 -501 500;-503 499 500 502 -501;500 -501 502 500 -500; 499 500 -501 -500 500]

f=[15000;16000;18000;16000;17000]

[e,x]=mkk(a,f)

e1=[e1 max(abs(e))]

e2=[e2 sqrt(sum(power(e,2)))]

mernost=[2 3 4 5];

plot(mernost,e1);

pause;

plot(mernost,e2);

pause

Результат работы программы:

>> head5

a =

500 501

501 500

f =

15000

16000

e =

1.0e-010 \*

-0.2910

0.5821

x =

515.4845

-484.5155

e1 =

5.8208e-011

e2 =

6.5078e-011

a =

500 501 -503

501 500 499

-503 499 500

f =

15000

16000

18000

e =

1.0e-010 \*

0.0182

0.0364

0.1455

x =

-2.0239

32.9970

1.0330

e1 =

1.0e-010 \*

0.5821 0.1455

e2 =

1.0e-010 \*

0.6508 0.1511

a =

500 501 -503 500

501 500 499 -501

-503 499 500 502

500 -501 502 500

f =

15000

16000

18000

16000

e =

1.0e-008 \*

0

0

-0.1120

0.0997

x =

14.5050

16.5505

17.4961

16.5125

e1 =

1.0e-008 \*

0.0058 0.0015 0.1120

e2 =

1.0e-008 \*

0.0065 0.0015 0.1500

a =

500 501 -503 500 499

501 500 499 -501 500

-503 499 500 502 -501

500 -501 502 500 -500

499 500 -501 -500 500

f =

15000

16000

18000

16000

17000

e =

1.0e-010 \*

-0.0364

0.0364

0.8367

-0.9459

0.1091

x =

33.0693

35.1332

-1.0682

-2.1077

-37.3144

e1 =

1.0e-008 \*

0.0058 0.0015 0.1120 0.0095

e2 =

1.0e-008 \*

0.0065 0.0015 0.1500 0.0127

Для E1=max |Ei|,



Для



В целом обусловленность матрицы А дает высокую точность решения, но по выбранным в данной работе системам трудно судить о влиянии мерности обусловленной матрицы А на точность решения.

# 

# **4. Анализ результатов. Выводы**

По исследованию можно сказать следующее. Точность решения СЛАУ методом квадратных корней для симметричной матрицы зависит от многих параметров, как то: мерность матрицы А, разреженность матрицы А, обусловленность матрицы А. Точность зависит от этих параметров как по отдельности, так и в комбинации. Можно также сказать, что точность решения сильно зависит от количества округлений во время решения и, как следствие собственно количества вычислений, которые необходимо произвести, чтобы решить СЛАУ методом квадратных корней. Было отмечено на этапе отладки программы, что, чем ближе корни системы к целым числам, тем меньше ошибка, тем выше точность.

**Заключение**

В данной курсовой работе был исследован метод квадратных корней для симметричной матрицы - один из методов решения систем линейных алгебраических уравнений. Этим методом можно решать системы вида A x = f, в которых матрица A – симметричная.

Также в данной работе были проанализированы разного рода параметры матрицы А: мерность, обусловленность, разряженность, и их влияние на точность полученного решения. В целом метод дает достаточно точные решения и может быть использован при поиске состояний равновесия в экономических моделях.

**Список использованной литературы**

1. Волков Е.А., Численные методы.- М.: «Наука», 1982.

2. Калиткин Н.Н. Численные методы.- М.: Наука,1978.

3. Сарычева О.М. Численные методы в экономике / О.М.Сарычева.-Новосибирск, 1995.- 67 стр.