Gaussian Elimination Method with Backward Substitution Using Matlab

Huda Alsaud

King Saud University

Huda Alsaud Gaussian Elimination Method with Backward Substitution Using

- 4 同 2 4 日 2 4 日 2

The main purpose of these slides is to demonstrate how to write a function m-file that will solve an arbitrary system (Ax = b) of N linear equations in N unknowns $x_i, i = 1 : N$ using the Gaussian Elimination algorithm as covered in class. The MATLAB program of the Gaussian Elimination algorithm can be done in various ways. However, since these slides were prepared for students how didn't learn MATLAB before, we will present some MATLAB statements which will be used in the program, but we limit the selection to the material which is needed later and for more details we refer to the references [1] and [2].

イロト イポト イヨト イヨト

The program will be as follows

- A function m-file, with inputs $A = (a_{ij})_{N \times N}$ and $b = (b_j)_{N \times 1}$ and output the solution vector, $x = (x_i)_{N \times 1}$.
- Step 1 For j = 1 : (N 1) do steps 2-3
- Step 2 If $a_{jj} = 0$ then find the smallest integer $j and <math>a_{pj} \neq 0$ and then $E_j \leftrightarrow E_p$. If no integer p can be found then output (The system has no unique solution).
- Step 3 For i = j + 1 : N

$$m = a_{ij}/a_{jj}$$
 preform $E_i - mE_j \to E_i$.

イロト イポト イヨト イヨト 二日

- Step 4 If $a_{NN} = 0$ then output (The system has no unique solution).
- Step 5 Set $x_N = b_N/a_{NN}$.

• Step 6 For
$$i = N - 1 : 1$$

$$x_i = (b_i - \sum_{j=i+1}^N a_{ij} x_j) / a_{ii}.$$

• Step 7 output x_1, \cdots, x_N .

< ロ > < 同 > < 回 > < 回 > < □ > <

We will discuss the following

- (1) Vectors and Matrices.
- (2) For Statement.
- (3) If Statement.
- (4) Functions that return more than one value.
- (5) Create a M-file to calculate Gaussian Elimination Method with Backward Substitution.
- (6) Homework.

イロト イポト イヨト イヨト

For Statement If Statement Functions that Return More than One Value Create a M-file to calculate Gaussian Elimination Method

Vectors and Matrices

Create Row Vector.

There are several ways to create row vector variables. The most direct way is to put the values in square brackets, separated by either space or commas.

Example

$$>> v = [1 4 - 2 0]$$

or

$$>> v = [1, 4, -2, 0]$$

The elements in a vector are numbered sequentially each element number is called the index.

For Statement If Statement Functions that Return More than One Value Create a M-file to calculate Gaussian Elimination Method

Example 1 2 3 4 v=[1 4 -2 0]

The *n*th element in the vector is v(n).



Create Matrix Variables

The values within a row are separated by either space or commas, and the different rows are separated by semicolons.

For Statement If Statement Functions that Return More than One Value Create a M-file to calculate Gaussian Elimination Method

Example 1 2 3 4 v=[1 4 -2 0]

The *n*th element in the vector is v(n).

Create Matrix Variables

The values within a row are separated by either space or commas, and the different rows are separated by semicolons.

<ロ> <部> <き> <き> <き> <き> <き</p>

For Statement If Statement Functions that Return More than One Value Create a M-file to calculate Gaussian Elimination Method

Example

>>
$$A = [1 \ 4 \ -8; 3 \ 0 \ -5]$$

A=
 $1 \ 4$
 $3 \ 0$

To refer to matrix elements the row and then the column subscripts are given in parentheses.



 $-8 \\ -5$

For Statement If Statement Functions that Return More than One Value Create a M-file to calculate Gaussian Elimination Method

Example

>>
$$A = [1 \ 4 \ -8; 3 \ 0 \ -5]$$

A=
 $1 \ 4 \ -8$
 $3 \ 0 \ -5$

To refer to matrix elements the row and then the column subscripts are given in parentheses.



イロト イポト イヨト イヨト

э

To refer to entire row use a colon for the column subscript,

Example
>> A(1,:)
ans=
1 4 -8

This refer to the entire second column,

>> A(:,2)		
ans= 4		

イロト イポト イヨト イヨト

To refer to entire row use a colon for the column subscript,

Example
>> A(1,:)
ans=
1 4 -8

This refer to the entire second column,

Exampl	le
>> A(:,2)
ans=	4
	0

イロン イボン イヨン イヨン

Dimensions

The *length* and *size* functions in MATLAB are used to find

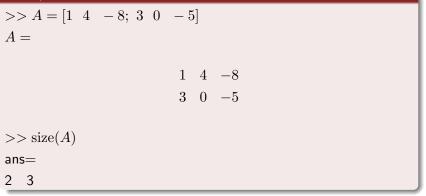
dimensions of vectors and matrices.

Example >> v = [1 4 - 2 0] v= 1 4 -2 0 >> length(v) ans= 4

イロト イポト イヨト イヨト 二日

For Statement If Statement Functions that Return More than One Value Create a M-file to calculate Gaussian Elimination Method

Example



For Statement

The for statement, or the for loop, is used when it is necessary to repeat statements in a script or a function, and when it is known ahead of time how many times the statements will be repeated.

The general form of the For loop is	
For i=range	
action	
end	

Example: Calculate the sum $\sum_{i=1}^{n} i$

```
s=0
```

```
tor i=1:n
```

```
s=s+i
```

end

For Statement

The for statement, or the for loop, is used when it is necessary to repeat statements in a script or a function, and when it is known ahead of time how many times the statements will be repeated.

The general form of the For loop is	
For i=range	
action	
end	

Example: Calculate the sum $\sum_{i=1}^n i$
s=0
for i=1:n
s=s+i
end

Huda Alsaud Gaussian Elimination Method with Backward Substitution Using

Breaking from a Loop.

Sometimes you may want MATLAB to jump out of a for loop, for example if a certain condition is met. Inside the loop, you can use the command *break* to tell MATLAB to stop running the loop and skip to the next line after the end of the loop. For example, to compute $\sum_{N=1}^{100} \frac{1}{N^4}$ and stop only when the terms become so small compared with the machine precision that the numerical sum stops changing we use the command *break*.

(日) (同) (王) (日) (日)

Example: Compute $\sum_{N=1}^{100} \frac{1}{N^4}$	
s=0	
for N=1:100	
a=s	
s=s+N^ (-4)	
if s=a	
break	
end	
end	

To stop executing of M-file, without running any further commands, use the command *return*.

- 4 同 2 4 日 2 4 日 2

If Statement

The if statement choose whether a statement, or group of statements, is executed or not.

The general form o	the if statement follows	
If condition		
action		
end		
$ if \ k < 10 $		
k = k * 4		
end		

▲ 同 ▶ ▲ 目 ▶ ▲ 目 ▶

If Statement

The if statement choose whether a statement, or group of statements, is executed or not.

The general form of the if statement follows
If condition
action
end

Example	
if $k < 10$	
k = k * 4	
end	J
	(日)(四)(金)(金)(金)

Huda Alsaud Gaussian Elimination Method with Backward Substitution Using

When the if statement is executed, the condition is evaluated. If the value of the condition is true the action will be executed, if not the action will not be executed.

To choose between two statements use if-else statement.

The general form is	
if condition	
action 1	
else	
action 2	
end	

(4 同) 4 ヨ) 4 ヨ)

To choose from among more than two actions use elseif.

The general form is
if condition 1
action 1
elseif condition 2
action 2
else
action 3
end

< ロ > < 同 > < 回 > < 回 > :

Example

if k > 1

fprintf('k is greater than 1')

elseif k < 1

```
fprintf('k is less than 1')
```

else

```
fprintf('k is equal to 1')
```

end

イロト イポト イヨト イヨト

Functions that return more than one value

Function that return more than one value must have more than one output argument in the function header in square brackets.

The general form looks like the following

function [output argument]=functionname(input argument)
statement here

end

Example

```
\begin{aligned} function[area, circum] &= areacirc(rad) \\ area &= pi * rad^2; \\ circum &= 2 * pi * rad; \\ end \end{aligned}
```

Functions that return more than one value

Function that return more than one value must have more than one output argument in the function header in square brackets.

The general form looks like the following

function [output argument]=functionname(input argument)
statement here

end

Example

function[area, circum] = areacirc(rad) $area = pi * rad^2;$ circum = 2 * pi * rad;end

Example

$$\begin{aligned} &function[r] = f(s) \\ &r(1) = sqrt(s(1)^2 + s(2)^2); \\ &r(2) = s(2)/s(1) \\ &\text{end} \end{aligned}$$

The above function takes as argument a vector of 2 components and returns a vector of 2 components.

Notice that MATLAB requires a double equals sign == to test for equality, a single equals is reserved for the assignment of value to variables.

イロト イポト イヨト イヨト 二日

Create a M-file to calculate Gaussian Elimination Method

14 - broak 15 - broak 16 - fprintf('The system has no unique solution') 17 - end 18 - end 19 - end 10 - end 10 - for i=nti;)/A(j,j); 10 - b(j)=b(j); 10 - b(j)=b(j); 11 - b(j)=b(j); 12 - b(j)=b(j); 12 - end 13 - b(j)=b(j); 14 - b(j)=b(j); 15 - end 16 - end 17 - for i=nti;('The system has no solution') 17 - return 18 - end 19 - end 19 - end 19 - end 19 - end 10 - b(j)=b(j); 10 - b(j)=b(j); 10 - b(j)=b(j)=b(j)=b(j)=b(j)=b(j)=b(j)=b(j)=	э 🦷	e 👙 🤊 🐢 🖆 📲 % 📓 🖆 🗂	- #4 🖛 👄	fry 돈 - f) 🖈 💷	- 🖷 主	**
2 - N=length(b); 3 - Quassian elimination 4 - Quassian elimination 5 - Quassian elimination 5 - Quassian elimination 5 - Quassian elimination 5 - Quassian elimination 6 - Quassian elimination 7 - If A(1, 1)===================================	+ <u>-</u> ($-$ 1.0 + \div 1.1 × $\%$ $\%$	0				
<pre>3</pre>		<pre>function [x]=Guss(A,b)</pre>					
$ \begin{array}{c} & \Rightarrow & for \ j-1; (N-1) \\ & & for \ j-1; (N-1) \\ & & for \ k-j+1; N \\ & & for \ k-j+1; N \\ & & & for \ k-j+1; N \\ & & & & for \ k-j+1; \\ & & & & for \ k-j+1; \\ & & & & & for \ k-j+1; \\ & & & & & & for \ k-j+1; \\ & & & & & & & for \ k-j+1; \\ & & & & & & & & & \\ & & & & & & & & $							
<pre>5 -</pre>							
<pre>6 -</pre>			or not				
<pre>8 = (1)⁻A(k, i); 9 - A(k, i)=A(j, i); 11 - A(j, i), i, i,</pre>	6 -	for k=j+1:N					
$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$							
$\begin{array}{c c c c c c c c c c c c c c c c c c c $							
11 - $t = b(k)$,							
$ \begin{array}{c c} & b(k)=b(j), \\ b(j)=k; \\ b$							
$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$							
<pre>14 - broak 5 - broak 6 - fprintf('The system has no unique solution') 6 - end 19 - end 10 - end 10 - end 10 - for i=jt1:N 11 - b(i)=b(j); 12 - f(i)=b(j)=m*b(j); 12 - b(i)=m*b(j); 12 - if(i)=b(j)=m*b(j); 12 - end 13 - end 14 - end 15 - end 16 - end 17 - f(i)=b(j)=m*b(j); 18 - end 19 - end 10 - end 10 - end 10 - end 10 - end 11 - end 11 - end 12 - end 13 - end 14 - end 15 - end 15 - end 16 - end 17 - f(i)=b(j)=b(j)=b(j); 17 - f(i)=b(j)=b(j)=b(j); 18 - end 19 - end 10 - end 10 - for j=(i+1):N 10 - end 10 - end 10 - end 10 - for j=(i+1):N 10 - for j=(i+1):N 10 - end 11 - end 11 - end 11 - end 12 - for j=(i)=b(j)=b(j)=b(j)=b(j)=b(j)=b(j)=b(j)=b(j</pre>	13 -						
16 fprintf('The system has no unique solution') 17 end 18 end 19 for i=j+1:N 11 for i=j+1:N 12 ma(i,j)/(j,j); 14 ma(i,j)/(j,j); 15 end 16 if A(n,i)=elio=ma(j,i); 17 end 18 end 19 for i=j+1:N 11 h=b(1)=mA(j,i); 12 end 13 edse 14 else 15 for j=(i+1):N 16 h=h+A(i,j)*x(j); 17 h=h+A(i,j)*x(j); 18 x(i)=(b(i)-h)/A(i,i); 19 end	14 -						
<pre>17 - end a - end 12 - for i=j+1:N m=A(1,j)(A(j,j); 13 - b(1)=b(1)=m*b(j); a - b(1)=b(1)=m*b(j); a - cond a - cond</pre>	15 -						
<pre>ind in the second second</pre>	16 -		system has no	unique sol	ution')		
<pre>19 - end 10 - for i=jt1:N 11 - for i=jt1:N 12 - A(i,i)=A(i,i); 13 - b(i)=b(i)=m*b(j); 14 - b(i)=b(i)=m*b(j); 15 - end 16 - end 17 - for i=fa(i,N,N)==0%check if a(n,n)=0 or not 18 - end 19 - else 10 - else 11 - b=0; 11 - b=0; 12 - i for i=(i+1):N 13 - i for i=(i+1):N 14 - b=0; 15 - i for i=(i+1):N 15 - i for i=(i+1):N 16 - end 17 - end 18 - end 19 - i for i=(i+1):N 19 - i for i=(i+1):N 10 - i for i=(i+1):N 1</pre>							
<pre>20 - end 1 - 0 for i=j+1:N m(i,i)=jA(i,i)=m*(j,i); 21 - b(i)=b(i)=m*b(j); 22 - end 23 - end 24 - b(i)=b(i)=m*b(j); 25 - end 27 - eff A(N,N)==0%check if a(n,n)=0 or not 28 - fprintf('The system has no solution') 29 - eff A(N,N)=0%check if a(n,n)=0 or not 28 - forintf('The system has no solution') 29 - eff A(N,N)=0%check if a(n,n)=0 or not 29 - eff A(N,N)=0%check if a(n,n)=0 or not 20 - eff A(N,N)=0%check if a(N,N)=</pre>							
<pre>21 - 0 for i-j+1:N m=A(i,j)/A(j,j); 32 - A(i,i)-A(i,i)-m*A(j,i); 33 - A(i,i)-A(i,i)-m*A(j,i); 34 - A(i,i)-A(i,i)-m*A(j,i); 35 - 0 end 36 - end 37 - if A(N,N)=0%check if a(n,n)=0 or not 38 - volume 39 - volume 39 - volume 30 - else 30 - else 30 - for j=(i+1):N 30 - b = b+A(i,j)*X(j); 30 - b = b+A(i,j)*X(j); 31 - x(i)=(b(i)-h)/A(i,i); 32 - end 33 - end 34 - end 35 - 0 - b = b+A(i,j)*X(j); 36 - b = b+A(i,j)*X(j); 37 - column 38 - a - b = b+A(i,j)*X(j); 39 - column 39 - end 31 - end 31 - b - b - b - b - b - b - b - b - b -</pre>	20 -						
$\begin{array}{c ccccccccccccccccccccccccccccccccccc$	21 -						
<pre>24</pre>	22 -						
<pre>22 - end 22 - end 27 - if A(N,N)==0✓ if a(n,n)=0 or not 27 - if printf('The system has no solution') 20 - else 21 - else 22 - end 23 - end 24 - else 25 - end 26 - else 27 - if a(N,N)=0 28 - else 29 - end 20 - end</pre>	23 -						
<pre>22 - end 27 - if A(N,N)==0%check if a(n,n)=0 or not 28 - iprint('The system has no solution') 29 - else 30 - else 31 %back substitution 32 - x(i)=b(1)A(N,N); 33 - infor j=(i+1):N 35 - infor j=(i+1):N 36 - infor j=(i+1):N 37 - infor j=(i+1):N 38 - infor j=(i+1):N 39 - end 41 - end Cuss Ln 34 Col 9</pre>							
<pre>27 - if A(N,N)==0%check if a(n,n)=0 or not fprintf('The system has no solution') 29 - return 30 %hack so solution 31 %hack so solution 32 - x(N)=b(N)/A(N,N); 33 - for i=N-1:-1:1 34 - h=0; =(i+1):N 55 - for j=(i+1):N 56 - i=(b(i)=(b(i)-h)/A(i,i); 37 - end 38 - x(i)=(b(i)-h)/A(i,i); 39 - end 41 - end 57 - Guss Ln 34 Col 9</pre>							
<pre>Z8 - fprintf('The system has no solution') - elseSturn - elseSturn - solution') - elseSturn - solution' - solution' - solution' - end - end - end - end - Guss Ln 34 Col 9</pre>) or not				
<pre>29 - return 30 - blas 31 - blas 32 - blas 33 - blas 34 - blas 35 - blas 45 -</pre>	28 -						
<pre>51</pre>	29 -		,				
<pre>32 - x(N)→b(N)/A(N,N); 33 - for i=N-1:-1:1 34 - for i=N-1:-1:1 35 - for j=(i+1):N 37 - end h=h+A(i,j)*x(j); 37 - end 39 - end 41 - end 50 - Guss Ln 34 Col 9</pre>	30 -						
33 - for '= N-1:-1:1' 33 - h=0; 35 - for j=(i+1):N 36 - h=h*A(i,j)*x(j); 37 - end 37 - end 38 - x(i)=(b(i)-h)/A(i,i); 40 - end 41 - ond Guss Ln 34 Col 9							
$\begin{array}{cccccccccccccccccccccccccccccccccccc$							
<pre>35 - i for j=(i+1):N 36 - h=h+A(j,j)*x(j); 37 - end 38 - x(i)=(b(i)-h)/A(i,i); 39 - end 41 - end 41 - Guss Ln 34 Col 9</pre>							
36 - h=h+A(i,j)*x(j); end - end - end - end - end - end - end - Ln 34 Col 9							
37 - end end 38 - x(i)=(b(i)-h)/A(i,i); 39 - end 41 - end Guss Ln 34 Col 9	36 -						
39 - end 11 - end Guss Ln 34 Col 9	37 -						
40 41 - end Guss Ln 34 Col 9	38 -						
11 - end Guss Ln 34 Col 9	39 -	- end					
Guss Ln 34 Col 9							
	41 -	ena	-				
			Guss		Ln 34	Col 9	
					1 A 1 1 A	1 E 1	-

Huda Alsaud Gaussian E

Gaussian Elimination Method with Backward Substitution Using

Example

Solve the linear system

$$\begin{array}{rcl} x_1 - x_2 + 2x_3 - x_4 & = & -8\\ 2x_1 - 2x_2 + 3x_3 - 3x_4 & = & -2\\ x_1 + x_2 + x_3 & = & -2\\ x_1 - x_2 + 4x_3 + 3x_4 & = & 4 \end{array}$$

using the M-file Guss.m.

Solution



メロト メポト メポト メポト

Homework

Q1: Give two examples of linear systems such that one of them has no unique solution and the other has unique solution and then solve them using Guss.m file.

Q2: Write a system of linear equations where the solution of the system will be your last four digits of your university number, and then solve the system using Guss.m file.

Q3: Change the file Guss.m so that whenever one of the pivots $a_{kk}^{(k)}$ is zero print an error message and stop the program execution.



- B.R. Hunt, R.L. Lipsman, and J.M. Rosenberg. A Guide to MATLAB, for beginners and experienced users. Cambridge University Press, 2001.
- Stormy Attaway. MATLAB: A Practical Introduction to Programming and Problem Solving. Elsevier Inc, 2012.

- 4 同 2 4 日 2 4 日 2