

Assignment Three

MT4112

Directory locked Tuesday 6th December at 11:00pm.

All work **must** be done in the sub-directory ‘assign3’ off your home directory. **REMEMBER** follow exactly the guidelines in the question.

1 Gaussian Elimination

A typical linear system can be written down as,

$$x_1 - x_2 + 2x_3 - x_4 = -8 \quad (1)$$

$$2x_1 - 2x_2 + 3x_3 - 3x_4 = -20 \quad (2)$$

$$x_1 + x_2 + x_3 = -2 \quad (3)$$

$$x_1 - x_2 + 4x_3 + 3x_4 = 4 \quad (4)$$

This linear system of equations can be written in matrix form as $A\mathbf{x} = \mathbf{v}$ which represents the four equations and four unknowns where;

$$A = \begin{pmatrix} 1 & -1 & 2 & -1 \\ 2 & -2 & 3 & -3 \\ 1 & 1 & 1 & 0 \\ 1 & -1 & 4 & 3 \end{pmatrix}, \quad \mathbf{v} = \begin{pmatrix} -8 \\ -20 \\ -2 \\ 4 \end{pmatrix} \quad \text{and} \quad \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}$$

The variables x_i where $i \in \{1, 2, 3, 4\}$ are the elements of the unknown vector \mathbf{x} . It is common to see a linear system, such as the one above, written down in its augmented matrix form $[A|\mathbf{v}]$. That is

$$[A|\mathbf{v}] = \left[\begin{array}{cccc|c} 1 & -1 & 2 & -1 & -8 \\ 2 & -2 & 3 & -3 & -20 \\ 1 & 1 & 1 & 0 & -2 \\ 1 & -1 & 4 & 3 & 4 \end{array} \right]$$

Consider two linear systems, both with the same number of unknowns, they are said to be **equivalent** if and only if they have the same set of solutions. This idea of equivalence is important as the idea behind solving a linear system is to form an equivalent linear system that is easier to solve. This forming of an equivalent system makes use of **elementary row operations** described as follows.

As an example consider the linear system described by the above four equations, E_i ($i \in \{1, 2, 3, 4\}$).

- Equation E_i can be multiplied by any non-zero constant λ and the resulting equation can be used in place of E_i . This operation is denoted by $(\lambda E_i) \rightarrow (E_i)$
- Equation E_j ($j = \{1, 2, 3, 4\}$) can be multiplied by any non-zero constant λ and the resulting equation can be added to E_i and the result used in place of E_i . This operation is denoted by $(E_i + \lambda E_j) \rightarrow (E_i)$.
- Equation E_i and E_j can be transposed in order (interchanged). This operation is denoted by $(E_i) \leftrightarrow (E_j)$.

The above three operations can be used to reduce a linear system of equations to *upper triangular* (or *reduced*) form as follows.

First write down the linear system in its augmented form

$$A^{(1)} = \left[\begin{array}{cccc|c} 1 & -1 & 2 & -1 & -8 \\ 2 & -2 & 3 & -3 & -20 \\ 1 & 1 & 1 & 0 & -2 \\ 1 & -1 & 4 & 3 & 4 \end{array} \right]$$

Now we are going to use multiples of equation E_1 (row one) to zero the elements in column one of rows two, three and four. To do this we use the element $A_{(1,1)}^{(1)}$ as the *pivot element*. The operations required to do this are $(E_2 - 2E_1) \rightarrow (E_2)$, $(E_3 - E_1) \rightarrow (E_3)$ and $(E_4 - E_1) \rightarrow (E_4)$

$$A^{(2)} = \left[\begin{array}{cccc|c} 1 & -1 & 2 & -1 & -8 \\ 0 & 0 & -1 & -1 & -4 \\ 0 & 2 & -1 & 1 & 6 \\ 0 & 0 & 2 & 4 & 12 \end{array} \right]$$

Now we are going to use multiples of row two to zero the elements in column two of rows three and four. To do this we use the element $A_{(2,2)}^{(2)}$ as the *pivot element*. Now we have a problem. Element $A_{(2,2)}^{(2)}$ is zero so the procedure cannot continue as before. The solution here is to use the the row-interchange operation $(E_i) \leftrightarrow (E_j)$. A search is made of the rest of the elements in column two that lie below $A_{(2,2)}^{(2)}$ (the zero pivot). Starting with $A_{(3,2)}^{(2)}$ and then $A_{(4,2)}^{(2)}$. The row-interchange is made using the row in which the first non zero element in column two is found. Therefore the operation to produce a valid pivot element in row two is $(E_2) \leftrightarrow (E_3)$ as the first non zero element, in column two row number greater than two, is found at $A_{(3,2)}^{(2)}$.

$$A^{(2)'} = \left[\begin{array}{cccc|c} 1 & -1 & 2 & -1 & -8 \\ 0 & 2 & -1 & 1 & 6 \\ 0 & 0 & -1 & -1 & -4 \\ 0 & 0 & 2 & 4 & 12 \end{array} \right]$$

Since element $A_{(4,2)}^{(2)'}$ is already zero then $A^{(3)} = A^{(2)'}$ and we can go on to calculate $A^{(4)}$ by using the operation $(E_4 + 2E_3) \rightarrow (E_4)$.

$$A^{(4)} = \left[\begin{array}{cccc|c} 1 & -1 & 2 & -1 & -8 \\ 0 & 2 & -1 & 1 & 6 \\ 0 & 0 & -1 & -1 & -4 \\ 0 & 0 & 0 & 2 & 4 \end{array} \right]$$

Now our matrix is in *upper triangular* form. That is all the elements under the leading diagonal are zero.

For a general n by n linear system the above example illustrates how to overcome the situation where the *pivot element* $A_{(k,k)}^{(k)}$ with $k \in \{1, 2, \dots, n-1\}$ is zero. If the *pivot element* is zero then the k^{th} column from and including the $(k+1)^{th}$ row to the n^{th} row is searched for the first non zero element. If the first non zero element is found in row p then the row-interchange operation $(E_k) \leftrightarrow (E_p)$ is performed. If all the elements searched in the k^{th} column are zero then the solution to the linear system is not unique and the process terminates. Finally if the element $A_{(n,n)}^{(n)}$ is zero then the solution to the linear system is also not unique and the process terminates.

2 Backward Substitution

Given a linear system in its upper triangular augmented form we can use backward substitution to calculate the values of x_i . For example consider the linear system used above

$$A^{(4)} = \left[\begin{array}{cccc|c} 1 & -1 & 2 & -1 & -8 \\ 0 & 2 & -1 & 1 & 6 \\ 0 & 0 & -1 & -1 & -4 \\ 0 & 0 & 0 & 2 & 4 \end{array} \right]$$

$$x_4 = \frac{4}{2} = 2 \tag{5}$$

$$x_3 = \frac{-4 - (-1)x_4}{-1} = 2 \tag{6}$$

$$x_2 = \frac{6 - x_4 - (-1)x_3}{2} = 3 \tag{7}$$

$$x_1 = \frac{-8 - (-1)x_4 - 2x_3 - (-1)x_2}{1} = -7 \tag{8}$$

This process of backward substitution then gives us our solution vector \mathbf{x}

$$\mathbf{x} = \begin{pmatrix} -7 \\ 3 \\ 2 \\ 2 \end{pmatrix} \tag{9}$$

3 Question One

Write a Fortran 90 code to directly solve a general n by n linear system of equations. Your code should use Gaussian elimination to form an upper triangular linear system and then use backward substitution to calculate the solution vector.

You should create a main program called 'linear.f90' and a module file 'linear_mod.f90'. The module file should contain the following subroutines.

- 'FUNCTION getvec(m)': Inputs an arbitrary vector (1D array) of size 'm'
- 'FUNCTION getmat(m,n)': Copy this from the class project.
- 'SUBROUTINE outmat(mat)': Copy this from the class project.
- 'FUNCTION gauss_elim(mat1,vv)': This function is called from the main program and manipulates the input matrix 'mat1' into *upper triangular* form. The function returns either '.TRUE.' if the system has a unique solution and '.FALSE.' if the system is over-determined. The vector 'vv' contains the RHS of our linear system and is also changed appropriately under the row operations.
- 'FUNCTION back_sub(mat1,vv)': This function accepts as its arguments the *upper triangular* matrix 'mat1' and the altered 'vv' and uses backward substitution to calculate and 'return' the solution vector. This function is called from the main program unit in the file ('linear.f90').
- 'FUNCTION find_pivot(mat1,vv,index)': This function is called from inside the function 'gauss_elim'. It accepts as its arguments the matrix 'mat1', the RHS vector 'vv' and 'index' which is the index value identifying the location of the current pivot in the matrix. It is called when the current pivotal element is found to be zero. This function then finds a new valid pivot and does the appropriate row-interchange in the matrix 'mat1'. This function also does the corresponding element interchanges for the RHS vector 'vv'. If no appropriate pivot can be found then the function returns '.FALSE.' else it returns '.TRUE.'

Your main program in the file 'linear.f90' should make use of the necessary functions to solve the linear system

$$x_1 - x_2 + 2x_3 - x_4 = -8 \quad (10)$$

$$2x_1 - 2x_2 + 3x_3 - 3x_4 = -20 \quad (11)$$

$$x_1 + x_2 + x_3 = -2 \quad (12)$$

$$x_1 - x_2 + 4x_3 + 3x_4 = 4 \quad (13)$$

NOTE : Your code should output to the screen the *upper triangular* matrix created in your 'gauss_elim' function and also the returned result vector from the function 'back_sub'. You should redirect the output to a text file called 'linear.txt'.

HINT : While you are testing your code it may be useful to comment out the call to 'getmat' in your main program and explicitly assign the values to your matrix. For example

```
mat1(1,:)=(/1,-1,2,-1/)
mat1(2,:)=(/2,-2,3,-3/)
mat1(3,:)=(/1,1,1,0/)
mat1(4,:)=(/1,-1,4,3/)
```

NOTE : When you have finished, you should have the following files in your assignment directory that will be required for marking. There is no need to create any sub-directories inside your assignment directory.

```
Main program file : linear.f90
Module file : linear_mod.f90
Results file : linear.txt
```