FINAL PROJECT PROPOSAL

SOLVING SYSTEM OF 3 EQUATIONS

INTENDED USER

Anyone who wants to solve a system of 3 equations with three unknown variables, solves 3X4 augmented matrix. The user can solve the system of equation by using gaussian Jordon method.

OVERVIEW

The project solves system of three equations by using the Gaussian method.

Methodology

1. Gaussian Elimination

* Step 1- take augmented matrix from the system of equations.
* Step 2- convert to row echelon form.
* Step 3- eliminate a variable from the equation by changing one element to zero in the coefficient or augmented matrix.
* Step 4-perform sequence of elementary row operations. Gives upper triangular matrix.
* Step 5- then back substitutes to get the right values of the variables.

PROBLEM SOLVING

Project will solve the system of equations inputted by the user after the user chooses which action. The program will finally output the values of the unknown variables.

TECHNOLOGIES USED

* Command line user interface.
* One- and two-dimensional array and vectors.
* OOP

CASE ANALYSIS

* User will be able to select either to solve or exit from the console by choosing a number and typing it in the command line.
* If the user chooses the first method the name of the method is displayed with an output statement saying, “solving is in process”.

DATA DESIGN

* The program is about data types in 1D and 2D arrays and vectors such as int, float, double etc.
* The best way to represent data is through objects, array and vector<vector<double>>.
* The data should be aggregated by using a class to define another classes entity. The data aggregated will impact the output.

UI DESIGN

When the run command for the program is given in the command line the first output on the console is as follows

Solving System of Equations!

Gaussian Elimination

ALGORITHM

* OOP- object oriented method
* Create a header and cpp file for each class.
* There will be three classes which are Start, Gaussian and main.

Start.h

Private:

Int num\_equations- the variables indicates how many equations are inputted by the user.

vector<vector<double> >donemat\_ - is the vector<vector<double>> variable for the done matrix.

gauss\_ - is the variable of Gaussian class type (object of gaussian class).

Vector<vector<double> > input\_mat – vector variable of the augmented matrix input by the user.

Public:

Start()- default constructor

Void readmatrix()- reads in the input augmented matrix by asking to plug in each element of the matrix individually.

Void solve\_equations(int method)- function does all the operations of gaussian Jordon elimination within it and prints out the main and the output.

~Start()- destructor

Start.cpp

Start()- default constructor, gives default value for num\_equations as zero. Creates object of gaussian class and puts it on the heap. Uses .clear to clear both donemat and input\_mat each time.

Void readmatrix()- reads the elements of the augmented matrix inputted. The first,second,third and fourth element are initialized as double.

1. A cout statement is printed to enter the number of equations inputted and that value is stored in num\_equations.
2. A for loop is created to loop from i=0 to lesser than num\_equation and increment each time.
3. Than multiple cout statements are inputted to input each element of the augmented matrix one by one.
4. Than the inputs are stored in mat\_row variable which is a vector from the end using push\_back.

Void solve\_equations(int method)

1. Returns nothing and has an int parameter. Prints out the main lines of the program which says solving system of equations by gaussian Jordon method.
2. Than the readmatrix() function is run to display the commands to input the numbers.
3. Than if the parameter passed is 0 it implements the solve(input\_mat\_) method from gaussian class by calling it through its object gauss\_.
4. Else if 1 is passed it does other methods if added to the program.
5. Nested for loops are made which print out upper\_tringularmat which equals to gauss\_->solve as many times it is lesser than the size of the upper\_triangular\_mat.size().
6. .size returns the size of the vector.

~Start()

1. Deletes gauss\_ from the heap.

Gaussian.h

Public:

std::vector<std::vector< double > > solve(std::vector<std::vector< double> >)- returns a vector and has a vector as parameter , calls all the function in the gaussian class and solves the matrix.

std::vector<std::vector< double > > reduce\_first\_col(std::vector<std::vector< double > >)- returns a vector and has a vector parameter, this function takes in a 3 by 4 matrix and returns the same matrix but with the first column zeroed out.

std::vector<std::vector< double > > reduce\_second\_col(std::vector<std::vector< double > >)-returns a vector and has a vector parameter, takes in the vector and second column is zeroed out.

void print\_mat(std::vector<std::vector< double> >)- returns void and has a vector parameter and prints the matrix modified.

std::vector<double> back\_substitution(std::vector<std::vector< double > >)- returns a vector<double> and has a parameter of vector<vector<double>>.

Gaussian.cpp

std::vector<std::vector< double > > Gaussian::solve(std::vector<std::vector< double> > mat)

1. Initializes first\_row\_zeroed\_mat and second\_row\_zeroed\_mat as a vector<vector<double>>.
2. Equates both the variables initialized to the function reduce\_first\_col(mat) and reduce\_second\_col(first\_row\_zeroed\_mat).
3. Then use the print\_mat function with first\_row\_zeroed and second row zeroed\_mat as parameter and print it out.
4. Also calls the back\_substitution method to run.
5. The function returns the second\_row\_zeroed\_mat value.

std::vector<std::vector< double > > Gaussian::reduce\_first\_col(std::vector<std::vector< double > > mat)

1.Gets amount by which the second and third rows should be subtracted .

2. intializes variables row2\_multiplier and row3\_multiplier to double and equates it to mat the function parameter. By using .at which helps return reference to the element at position n in the vector.

3.row2\_multiplier is equated to mat at(1) and (0) position and then multiplied by -1 and row3\_multiplier is equated to mat.at(2).at(0) and multiplied by 1.

4. then modify mat to updated version without zeroes and first calculate the multipliers.

5. multipliers take second row from first column and negate it and do similar for row 3.

6. mat.at(1) and at 0 ,1,2,3 is added with row multiplier multiplied with mat at(0) and at(0),(1),(2),(3).

7. it returns the mat then.

std::vector<std::vector< double > > Gaussian::reduce\_second\_col(std::vector<std::vector< double > > mat)

1. Reduces the second column. Returns a vector<vector<double>> and has a parameter vector<vector<double>> mat.
2. Add row3\_multiplier\*row2 to row3.
3. Row3\_multipler is equated to mat.at(2).at(1)/mat.at(1).at(1) multiplied by -1.
4. Similar to reduce\_first\_col we equate mat.at(2).at(0),(1),(2),(3) to row3\_multiplier multiplied by mat.at(1).at(0),(1),(2),(3).
5. Return mat.

void Gaussian::print\_mat(std::vector<std::vector< double> > mat)

1. Prints out the matrix. There is a nested for loop the first loop controls the row and inner loop controls the column.
2. Then after each loop it prints the mat.at(i).at(j) and then prints a newline.

std::vector<double> back\_substitution(std::vector<std::vector< double > >)

1. Performs backsubstitution an input matrix. The input matrix is assumed to be in upper triangular form. Returns a vector of the values for x,y, and z. The first element is the

value for x, the second is the value for y, and the third is the value for z The input matrix is also assumed to be 3 by 4.

1. Declare a variable xyz\_values as a vector<double>.
2. Then three variables x,y and z are declared as double. Declaring three variables makes the process explicit.
3. Equate z to mat the parameter at(2).at(3) divided by mat.at(2).at(2).
4. Equate y to mat at(1).at(3)- (z\*mat.at(0).at(2))divided by mat.at(1).at(1).
5. Equate x to mat.at(0)at.(3)- (z\*mat.at(0).at(2))-(y\*mat.at(0).at(1))/mat.at(0).at(0)
6. Then print out each of the variable with cout statement.
7. Then store each variable in xyz\_values and use .push\_back to store from back.

Main.cpp

1. Makes an object of start class called start and puts it on the heap.
2. Using start calls the solve\_equations(0) function from start class with parameter int method=0.

Solver.h (abstract class)

1. It is a abstract class with the solve function which can be inherited by any method class and be defined.

