Hemant Heer

Naïve Gaussian Elimination with Partial Pivoting- Graphical User Interface

User's Manual

12/5/2013

The layout of the GUI is simple in that the user only needs to enter the coefficients of the linear equations and the solutions to the linear equations (these values are not the solutions to the variables because those will be determined via the algorithm for Gaussian Elimination with Partial Pivoting). The input type recommended for the input boxes in both the coefficient panel input and the right hand side vector is the numeric type and attempting to enter strings into the input boxes will not make the 'Solve System of Linear Equations' button available to the user. If such an error is made, the user can immediately change the input to match the recommended format and the solution button will be made available again.

The variables being used for the system of linear is not necessary for the user to change but can be changed if the user desires clarity when treating the system with an augmented matrix. The format of the input boxes in the panel named 'Variables' is the string type.

The solutions panel will automatically display the values of x, y, z (in the case where the user has changed the variable names, the first column of text boxes will show the correct variable names) if a unique solution exists.

If the system of linear equations does not appear to have a solution, then there are 2 visual changes the user will be alerted to. The first of which is warning window that displays the type of error involved and the second involves creating the row reduced echelon form of the augmented matrix into the input boxes.

Furthermore, if the system of linear equations appears to have infinite solutions by visual inspection a similar set of information will be conveyed in the same manner. There will be window that appears containing information that relays to the user that there are infinite solutions and the input boxes that are part of an augmented matrix will show the row reduced echelon form of the matrix. Even the panel 'Solutions' will show what would have been the solutions

If the user wants to exit the GUI, the 'x' button on the top right corner of every window in the Windows Operating System will be used to shut down the program.

<u>Layout of the GUI and its creation</u>

The GUI's basic structure involves 4 panels which hold different parts of the augmented matrix in the interface. These panels are: 'Coefficients', 'Variables', 'Right Hand Side Vector', and 'Solutions' and they function as holders for the input boxes in which the user will enter linear equation parameters. In total there are 15 edit boxes that were used when the interface was being created using GUIDE and 6 static boxes which the program uses to pull values from the edit boxes and display the solutions to the linear equations. The final object is a button that gives the user a visual object to press and calculate the solutions to the linear equations. The other function of the button is to disable itself if the input to the edit boxes in both the 'Solutions' and 'Right Hand Side Vector' panels is not a number.

Since the code for each of the edit boxes is the same and the only other non-redundant code belongs to the push_button callback function. Example code from the edit_1 callback function will be displayed and a line-line discussion of the code will be given here.

```
if isnan(input1)
    % isnan checks if the input is 'not a number' and if true the
    % push button is disabled.
    set(handles.push_button,'Enable','off')
else
    % Enable the Push button
    set(handles.push_button,'Enable','on')
end
```

Since the edit box will contain a string, as Matlab understands it, I convert the value into a string and then check if the input is 'not a number' and if it is then the push button will be disabled until the user enters a correct type of input. By tagging the push button with a string that I have chosen, I can access the button with its own tag and handle from another callback function.

The next major part of the GUI is the push button callback function. I've removed redundant just to focus on the parts of the push button that do all the heavy lifting. After the user has entered the necessary parameters into the panels to program calls the edit box's handle from the tag while in the push_button callback function and converts the string entered into an integer. This is the same process when pulling values from the boxes in which the user enters the right hand side vector.

The next set of lines focuses on creating two matrices that will become the augmented matrix. All the coefficients will be written into the matrix A and the right hand side vector values will be put into vector b. Since the project only required a set of 3 linear equations there are only 3 unknowns in the problem, so n is designated as the number of unknowns.

The rank of the coefficient matrix is stored in r and the rref function converts the augmented matrix (A and b together are passed into the function) and also captures the rank of the augmented matrix in jb. These ranks will be used in detecting what type of system the set of linear equations is.

If the rank of coefficient matrix is equal to the rank of the augmented matrix, then the program checks if the rank is equal to the number of unknowns and if that evaluates as true than there are unique solutions to the set of linear equations that user entered into the system. All the program does is

identify the static text boxes in the solutions panel and uses the handle for the tag to show the solutions to all the variables in the static text boxes. In addition, the program will also capture the string entered in the edit boxes within the variable panel via the get function and the handles parameter(referencing the tag of the edit boxes) and sets the static box in the solutions to contain that variable name. There is no error checking when the user adds their own string into the edit box because the variable names are irrelevant and only serve to give the user a little control over how they want to see the system of linear equations.

If the rank of the coefficient matrix is not equal to the number of unknowns, then the GUI will update the edit boxes in the coefficient panel and the right hand vector panel with the row reduced echelon form of the augmented matrix after Gauss Jordan Elimination was used. This is accomplished by using the handles and the tag parameter to automatically show the user information on the screen. In addition to updating the edit boxes in both panels, a warning dialog box is generated to inform the user that the set of linear equations has infinite solutions (which is why the row reduced echelon form is shown on the screen).

The final case of 'No Solutions' is accomplished by checking if the rank of the coefficient matrix is equal to the rank of the augmented matrix. If these values are not equal to each other, then a warning dialog box appears on screen and informs the user that this system does not have any solutions. The row reduced echelon form is displayed in the edit boxes of both the 'Solutions' panel and the 'Right hand side vector panel' by using the set function and accessing the handles via the tags of all the edit boxes. The fourth column of the output of the rref function(this is the row reduced echelon form of the augmented matrix) contains the solutions to the variables in the system and is displayed by accessing all the values and placing them into static text boxes in the solutions panel using the set function, the handle, and the tag.

```
function push_button_Callback(hObject, eventdata, handles)
% hObject    handle to push_button (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%Get all user input from GUI using handles to the edit boxes
coeff1 = str2double(get(handles.edit1,'String'));

right_hand13 = str2double(get(handles.edit13,'String'));
%Solve linear equations using Naive Gauss Elimination with Partial Pivoting
%Function


A = [coeff1,coeff2,coeff3;coeff4,coeff5,coeff6;coeff7,coeff8,coeff9];
b = [right_hand13;right_hand14;right_hand15];
```

```
n = 3;
r = rank(A);
[answers jb] = rref([A,b])


if r==length(jb)
    if r==n
        %Show solutions in static text boxes in solutions panel
        set(handles.Fin_x,'String',num2str(answers(1,4)));
        set(handles.Fin_y,'String',num2str(answers(2,4)));
        set(handles.Fin_z,'String',num2str(answers(3,4)));

        %Also get and set variable names that user inputs--Doesn't really matter
        name1 = get(handles.var_set1,'String');

        set(handles.var_name1,'String',name1);
    else
        warndlg('Infinite Number of Solutions,system not solvable.Try another set of Linear
Equations. Input boxes will now show row reduced echelon form of augmented
matrix.','Warning!')



        %Show row reduced echelon form in input boxes for user
        set(handles.edit1,'String',answers(1,1));

        set(handles.edit4,'String',answers(2,1));

        set(handles.edit7,'String',answers(3,1));

        set(handles.edit13,'String',answers(1,4));

    end

else
    warndlg('No Solutions: Parallel set of Linear Equations. Try another set of Linear
Equations','Warning')

    %Show row reduced echelon form in input boxes for user
    set(handles.edit1,'String',answers(1,1));

    set(handles.edit13,'String',answers(1,4));
end
```

References

http://www.mathworks.com/help/matlab/creating_guis/gui-with-multiple-axes-guide.html