CSE521 (Spring 2018)
Programming Assignment I
3/1/2018

The following is a description of fouriermotzkin.py, a program that takes a system of inequalities and outputs: lower and upper bounds of each variable and a loop nest of the solution [if it exists]

The program takes in n (dimension of matrix), A (coefficient matrix) and b (R.H.S. of inequalities) as input with appropriate print statements guiding user on how to feed values to the program. After forming the matrices from the input, the program calls upon the function 'fouriermotzkin(A,b,num_var)' to reduce the system down to one variable. It achieves this by:

1. Arranging all the inequalities in a sorted order
2. Diving each inequality by the absolute value of its corresponding first element (i.e. value in the first column)
3. Create a system of inequalities without the first variable by adding the positive and negative inequalities to create [# positive x # negative + # inequalities with zero in the first column] new inequalities.
4. The new system of inequalities does not contain the first variable. This new system again calls the 'fouriermotzkin(A,b,num_var)' function

The process is repeated recursively until all but one variable is eliminated which is then returned to represent [upper/lower] bound on the variable (depending on its coefficient in the original system).

There are two approaches that we can use from this point to find the bounds on other variables:

1. Calling the 'fouriermotzkin(A,b,num_var)' function with the original system of inequalities to reduce to eliminate all but 2 variables. After the function returns, feed the bound which was found in the earlier step to find the bound on a new variable, OR
2. Rotate the coefficient Matrix 'A' to make the second variable as the third variable and feed it to 'fouriermotzkin(A,b,num_var)' function to find a bound automatically.

In this program we have chose to do the latter, though we do have code available for the former method. After the bounds are found on all the

variables, the entire system is multiplied by (-1) to convert the original system in Ax >= b form instead of Ax <= b form. The entire process is repeated once again to ensure we have upper and lower bounds on each variable.

After we have found the bounds, we try to find the integer solution space (if it exists). If any of the bounds on any of the variables stretch to +∞ or -∞, the system is said to have infinite solutions, and an appropriate message is displayed on the console. If not, then a solution matrix is created and all the possible solutions are enumerated in this solution matrix which is displayed at the end.

How to use the code:

To run the program, you need to have python installed on your system. After doing so, run the following command on the console:

- `python fouriermotzkin.py`

Enter the values as prompted on the screen and press enter.

For user convenience, I have included some examples with hard coded values for matrices A and b. To run these examples, type:

- `python example#.py` [where # represents a number from 1- 5]

The examples handles a number of edge cases such as all positive coefficients, parallel lines, multiple solutions etc.

References:

1. https://www.youtube.com/watch?v=4Gfqcox9xaw, Youtube video showing how Fourier Motzkin Elimination works
2. https://promathmedia.files.wordpress.com/2013/10/alexander_schrijver_theory_of_linear_and_integerbookfi-org.pdf, Theory of Linear and Integer Programming by Alexander Schrijver
3. http://www.cs.cmu.edu/~odonnell/toolkit13/lecture13-anonymous.pdf, CMU Lecture on Fourier Motzkin Elimination
4. https://en.wikipedia.org/wiki/Fourier%E2%80%93Motzkin_elimination, Wikipedia entry describing on Fourier Motzkin Elimination