MAT 423: Homework 3 (09/12)

Ex 1. [3pts]

We use Gauss elimination on the following tableau

$$\left[\begin{array}{ccc|c} 1 & 2 & -1 & 0 \\ 0 & -1 & 1 & 0 \\ 3 & 3 & \alpha & \beta \end{array}\right] \Rightarrow \left[\begin{array}{ccc|c} 1 & 2 & -1 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & -3 & \alpha+3 & \beta \end{array}\right] \Rightarrow \left[\begin{array}{ccc|c} 1 & 2 & -1 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & \alpha & \beta \end{array}\right].$$

We deduce that if $\alpha \neq 0$ then there exists a unique solution. If $\alpha = 0$ and $\beta \neq 0$, then there is no solution. If $\alpha = \beta = 0$, then there are multiple solutions.

Extra: [+1pt] Each equation can be interpreted as a plane P in $\mathbb{R}^3$. As such, if $\alpha \neq \beta$, the three plans intersect at a single point (the solution x). Whereas if $\alpha = 0$ and $\beta \neq 0$, the three plans do not have a common intersection. Finally if $\alpha = \beta = 0$, the intersection is line giving multiple solutions.

Ex 2. [4pts]

a) [2pts] See programs at the end.

b) [2pts] We estimate the number of operations for matrices of size $n = 5 : 5 : 25$ and use linear regression to estimate the growth in log-log scale (see figure 1).

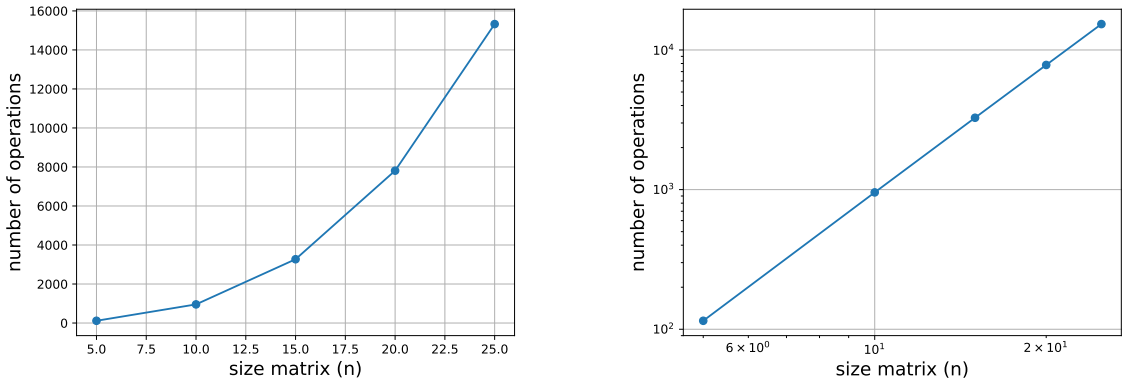

Figure 1: Number of operations performed to solve the linear system $Ax = b$ using Gauss elimination. The growth is cubic: $\mathcal{O}(n^3)$ (the estimation of the slope of the curve in log-log scale is 3.04).

Ex 3. [3pts]

a) Integrating twice the equation gives:

$$u(x) = x^2 + c_1 \cdot x + c_2.$$

Using the boundary conditions, we deduce the values of the constant c_1 and c_2: 1pt

$$u(x) = \frac{1}{2}x(1-x).$$

b) The linear system gives the following $(N-1)\ddot{O}(N-1)$ problem $A\mathbf{u} = \mathbf{b}$ with .5+.5pt

$$A = \begin{bmatrix} 2 & -1 & 0 & & \dots & 0 \\ -1 & 2 & -1 & & & \vdots \\ 0 & -1 & 2 & -1 & & \\ & & \ddots & \ddots & \ddots & \\ \vdots & & & -1 & 2 & -1 \\ 0 & \dots & & & -1 & 2 \end{bmatrix}, \quad b = \begin{pmatrix} \Delta x^2 \\ \Delta x^2 \\ \vdots \\ \Delta x^2 \\ \Delta x^2 \end{pmatrix} \quad \text{and } \Delta x = \frac{1}{N}.$$

When $N = 10$, the solution is: 1pt

$$\mathbf{u} = \begin{pmatrix} 0.045 \\ 0.080 \\ 0.105 \\ 0.12 \\ 0.125 \\ 0.12 \\ 0.105 \\ 0.080 \\ 0.045 \end{pmatrix}$$

c^*) For this particular choice of right-hand side (i.e. 1), the numerical solution is actually *exact*. Indeed, the error of the method is second order $\mathcal{O}(\Delta x^2)$ but as the exact solution $u(x)$ is also a second order polynomial, the numerical approximation is exact. This is very (very) specific case. For another choice of right-hand side, we would have found: $|u(x_i) - u_i| = \mathcal{O}(\Delta x^2)$.

Ex 4.

a) See script at the end.

b) If we do not perform partial pivoting, the Gauss elimination 'crashes' as the pivot becomes zeros. Using the partial pivoting, we find the solution: $\mathbf{x} = (-9, 3, 5)^T$.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%        Gauss elimination   (MATLAB/OCTAVE)       %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function x=func_pivot_gauss(A,b)
    % Solve the solution x to: Ax=b
    %    using Gauss elimination
    %
    % init
    Tableau = [A b];
    N = length(b);
    nb_operations = 0;
    %---------------------------%
    %   A) Gauss elimination    %
    %---------------------------%
    for j=1:(N-1)
        if (Tableau(j,j) == 0 )
            % danger!
            list_pivot = Tableau(j:end,j);
            i0 = argmax(abs(list_pivot)) + (j-1);
            if (Tableau(i0,j) ~= 0)
                % exchange row i0 and j
                tp = Tableau(j,:);
                Tableau(j,:) = Tableau(i0,:);
                Tablea(i0,:)  = tp;
            else
                % singular
                printf('--------singular matrix!--------')
                return
            end
        end
        % pivot
        a_jj = Tableau(j,j);
        for i=(j+1):N
            Tableau(i,:) = Tableau(i,:) - Tableau(i,j)/a_jj*Tableau(j,:);
            nb_operations = nb_operations + 3*(N-j);
        end
    end
    %--------------------------------%
    %   B) Backward substituation    %
    %--------------------------------%
    x = zeros(N,1);
    for i=(N:-1:1)
        y = Tableau(i,N+1);          % y_i = b_i - ... a_ij x_j - ...
        for j=(i+1):N
```

```
            y = y - Tableau(i,j)*x(j);
            nb_operations = nb_operations + 2;
        end
        x(i) = y/Tableau(i,i);        % a_ii x_i = y_i
        nb_operations = nb_operations + 1;
    end
    %--- test
    %A*x-b
    printf('number of operations: ')
    printf(num2str(nb_operations))
    printf('\n')
end
```

```
#####################################################
##        Gauss elimination    (Python)             ##
#####################################################
import numpy as np

def pivot_gauss(A, b: np):
    ''' Solve the solution x to: Ax=b
    using Gauss elimination
    '''

    # init
    Tableau = np.concatenate((A,b),axis=1)
    N = len(b)
    nb_operations = 0
    #--------------------------#
    #   A) Gauss elimination    #
    #--------------------------#
    for j in range(N-1):
        if (Tableau[j,j] == 0 ):
            # danger!
            list_pivot = Tableau[j:,j]
            i0 = np.argmax(abs(list_pivot))  + j
            if (Tableau[i0,j] != 0):
                # exchange row i0 and j
                tp = Tableau[j,:]
                Tableau[j,:] = Tableau[i0,:]
                Tableau[i0,:]  = tp
            else:
                # singular
                print("--------singular matrix!--------")
                return
        # pivot
        a_jj = Tableau[j,j]
        for i in np.arange(j+1,N):
            Tableau[i,:] -= Tableau[i,j]/a_jj*Tableau[j,:]
            nb_operations += 3*(N-(j+1))
    #--------------------------------#
    #   B) Backward substituation    #
    #--------------------------------#
    x = np.zeros((N,1))
    for i in np.arange(N-1,-1,step=-1):
        y = Tableau[i,N]          # y_i = b_i - ... a_ij x_j - ...
        for j in range(i+1,N):
            y -= Tableau[i,j]*x[j]
```

```
            nb_operations += 2
        x[i] = y/Tableau[i,i]        # a_ii x_i = y_i
        nb_operations += 1
    #--- test
    print(np.matmul(A,x)-b)
    print("number of operations: ",nb_operations)

    return x
```

```
#####################################################
##        Gauss elimination    (Julia)              ##
#####################################################
function pivot_gauss(A,b)
    #= Solve the solution x to: Ax=b
    using Gauss elimination
    =#

    # init
    Tableau = [A b]
    N = length(b)
    nb_operations = 0
    #--------------------------#
    #   A) Gauss elimination    #
    #--------------------------#
    for j=1:(N-1)
        if (Tableau[j,j] == 0 )
            # danger!
            list_pivot = Tableau[j:end,j]
            i0 = argmax(abs.(list_pivot)) + (j-1)
            if (Tableau[i0,j] != 0)
                # exchange row i0 and j
                tp = Tableau[j,:]
                Tableau[j,:] .= Tableau[i0,:]
                Tableau[i0,:]  .= tp
            else
                # singular
                println("--------singular matrix!--------")
                return
            end
        end
        # pivot
        a_jj = Tableau[j,j]
        for i=(j+1):N
            Tableau[i,:] .-= Tableau[i,j]/a_jj*Tableau[j,:]
            nb_operations += 3*(N-j)
        end
    end
    #-------------------------------#
    #   B) Backward substituation    #
    #-------------------------------#
    x = zeros(N)
    for i=(N:-1:1)
        y = Tableau[i,N+1]          # y_i = b_i - ... a_ij x_j - ...
```

```
        for j=(i+1):N
            y -= Tableau[i,j]*x[j]
            nb_operations += 2
        end
        x[i] = y/Tableau[i,i]        # a_ii x_i = y_i
        nb_operations += 1
    end
    #--- test
    println(A*x-b)
    println("number of operations: ",nb_operations)

    return x
end
```