Computer Exercise 2.2.3

The following program will use Gaussian elimination with scaled partial pivoting to solve the following $\mathbf{Ax} = \mathbf{b}$ system:

$$\begin{bmatrix} 0.4096 & 0.1234 & 0.3678 & 0.2943 \\ 0.2246 & 0.3872 & 0.4015 & 0.1129 \\ \underline{\mathbf{0.3345}} & 0.1920 & 0.3781 & 0.0643 \\ 0.1784 & 0.4002 & 0.2786 & 0.3927 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 0.4043 \\ 0.1550 \\ 0.4240 \\ 0.2557 \end{bmatrix}$$

Here, the matrix equation is mostly the same as in computer exercise 2.2.2 except element a_{31} of $\mathbf{A}$ is modified from 0.3645 to 0.3345 (the modification is indicated by the boldface and underline), so as to simulate slight mistypes in data entry in order for us to study the effects of minor perturbations to the matrix system. We will compare the solutions of this system and that of 2.2.2 by taking the ratio of the respective components where we designate the solution of this system as $\mathbf{x}_{est}$ and the solution of 2.2.2 as $\mathbf{x}_{sol}$,

```
A = [0.4096, 0.1234, 0.3678, 0.2943;
    0.2246, 0.3872, 0.4015, 0.1129;
    0.3345, 0.1920, 0.3781, 0.0643;
    0.1784, 0.4002, 0.2786, 0.3927];
b = [0.4043, 0.1550, 0.4240, 0.2557]';
[Amod2, bmod2, xest] = gespp(A2,b2);
xest
```

```
xest = 4×1
    6.7831
    3.5914
   -6.4451
   -1.5179
```

Now, we compare this solution to the solution from 2.2.2 by determining the following component wise ratio $|\mathbf{x}_{est}^{(i)}/\mathbf{x}_{sol}^{(i)}|,\ i \in \{1, 2, 3, 4\}$.

```
%acquire x from problem 2.2.2
xsol = x;
factordif = abs(xest./xsol)
```

```
factordif = 4×1
    1.9601
    2.3008
    2.1965
    3.5296
```

It turns out that each element of $\mathbf{x}_{est}$ is overall at least twice as much as $\mathbf{x}_{sol}$. We can explore why this is by considering the condition number of the original matrix from 2.2.2:

```
%original matrix
A1 = [0.4096, 0.1234, 0.3678, 0.2943;
    0.2246, 0.3872, 0.4015, 0.1129;
```

```
    0.3645, 0.1920, 0.3781, 0.0643;
    0.1784, 0.4002, 0.2786, 0.3927];

n1a = norm(A1, 2);
n1b = norm(inv(A1), 2);
condition_number = n1a*n1b
```

```
ans = 46.1393
```

We see that the condition number is mildly far away from one, but it isn't overwhelmingly large. However, considering the effects of just a single data point (matrix entry) on the final solution, we might as well consider it ill conditioned enough since we can easily imagine this much of a difference leading to catastrophic consequences (e.g. perhaps a collapsing of a bridge resulting from a slightly erroneous solution for a mathematical model of some dynamical system). We can explore this further by considering the determinant of the original matrix

```
det(A1)
```

```
ans = -0.0024
```

The determinant is fairly close to zero which makes the near singluarity aspect of the original matrix clearer, so it is no wonder, then, why just a slight modification resulted in a different final solution.

```
function [Amod, bmod, x] = gespp(A,b)
    n = length(b);
    %set index vector
    l = (1:n);
    %set scale vector
    s = zeros(length(l), 1);
    for i = 1:n
        s(i) = max(abs(A(i, :)));
    end
    %forward elimination
    for k = 1:(n-1)
        max_r = 0;
        pivot_index = l(1);
        for i = k:n
            if (abs(A(l(i), k))/s(l(i))) > max_r
                pivot_index = i;
                max_r = (abs(A(l(i), k))/s(l(i)));
            end
        end
        a = l(pivot_index);
        l(pivot_index) = l(k);
        l(k) = a;
        for i = (k+1):n
            mult = A(l(i), k)/A(l(k), k);
            for j = k:n
                A(l(i), j) = A(l(i), j) - mult*A(l(k), j);
            end
            b(l(i)) = b(l(i)) - mult*b(l(k));
        end
```

```
    end
    Amod = A;
    bmod = b;
    %back substitution
    x = zeros(n, 1);
    x(n) = b(l(n))/A(l(n), n);
    for u = (n-1):-1:1
        sum = 0;
        for v = (u+1):n
            sum = sum + (A(l(u), v)*x(v));
        end
        x(u) = (b(l(u)) - sum)/(A(l(u), u));
    end
end
```