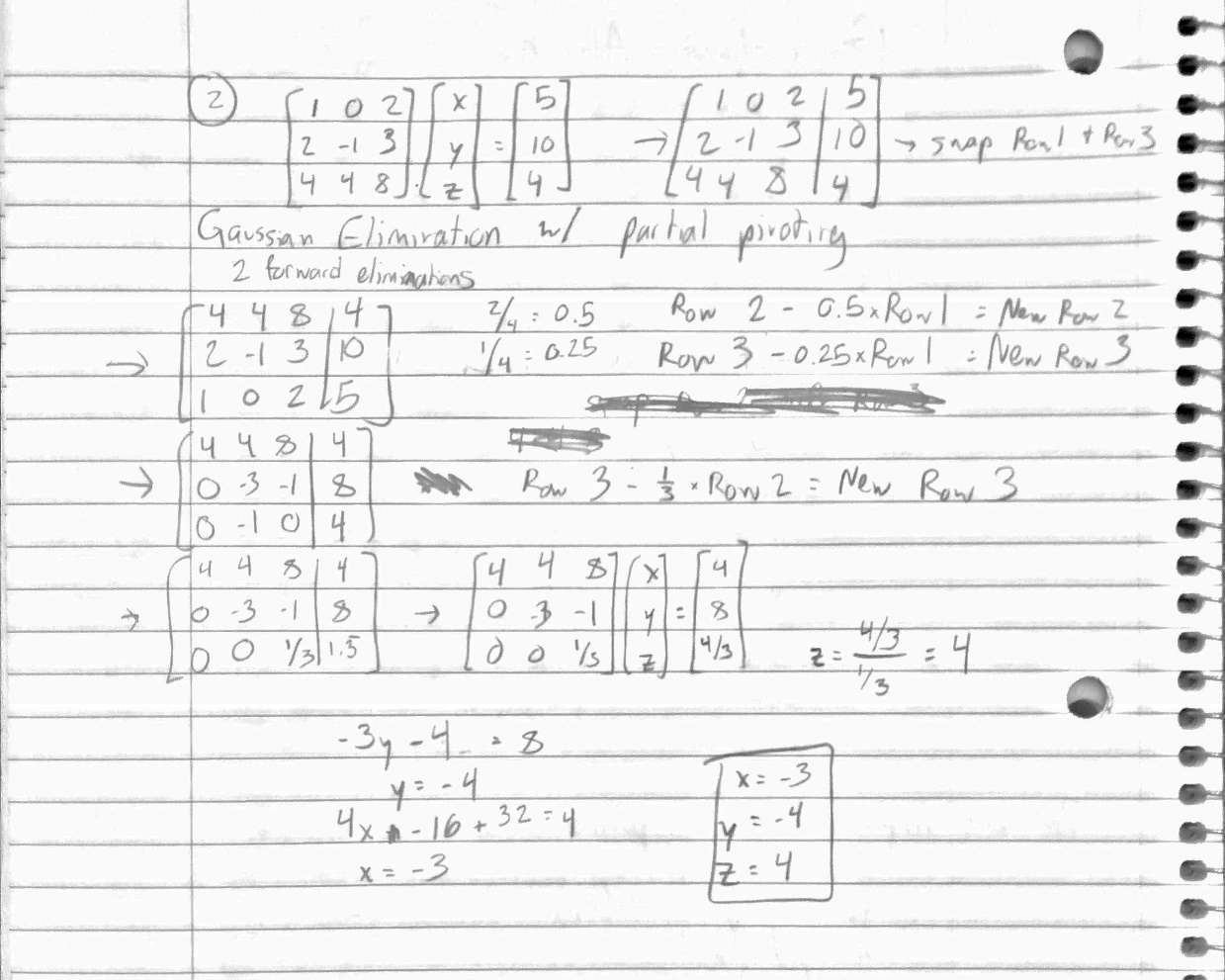
Matt McDade

Applied Numerical Methods HW 3

**Problem 1:**

|  |  |
| --- | --- |
| CODE  a = [5 -2 0 1;  -1 6 2 -9;  0 0 3 -4;  1 1 1 1];  b = [7; 12; 3; 0];  x = a\b | OUTPUT  x =  1.8734  0.4675  -0.9091  -1.4318 |

**Problem 2:**



**Problem 3:**

|  |  |
| --- | --- |
| CODE  a = [4 1 -1;  -1 5 2;  -2 1 6];  b = [13; 8; 2];  tol = 1e-10;  maxiter = 100;  [s, k] = jacobi(a, b, [0; 0; 0], tol, maxiter)  [s, k] = jacobi(a, b, [10; 20; -30], tol, maxiter)  function [x0, k] = jacobi(a, b, x0, tol, maxiter)  n = length(b);  x = zeros(n, 1);  for k=1:maxiter  for j=1:n  x(j) = (b(j) - a(j, [1:j-1, j+1:n]) \* x0([1:j-1,j+1:n])) / a(j,j);  abserr=norm(x-x0);  relerr=abserr/norm(x+eps);  if(abserr<tol && relerr<tol)  return  end  end  x0=x;  end | OUTPUT  s =  3.0667  1.7905  1.0571  k =  26  s =  3.0667  1.7905  1.0571  k =  30 |

**Problem 4:**

|  |  |
| --- | --- |
| CODE  a = [4 1 -1;  -1 5 2;  -2 1 6];  b = [13; 8; 2];  tol = 1e-10;  maxiter = 100;  [s, k] = gauss\_seidel(a, b, [0; 0; 0], tol, maxiter)  [s, k] = gauss\_seidel(a, b, [10; 20; -30], tol, maxiter)  function [x0, k] = gauss\_seidel(a, b, x0, tol, maxiter)  n = length(b);  x = zeros(n, 1);  for k=1:maxiter  for j=1:n  x(j) = (1/a(j, j))\*(b(j) - a(j, 1:n)\*x + a(j, j)\*x(j));  abserr=norm(x-x0);  relerr=abserr/norm(x+eps);  if(abserr<tol && relerr<tol)  return  end  end  x0=x;  end | OUTPUT  s =  3.0667  1.7905  1.0571  k =  17  s =  3.0667  1.7905  1.0571  k =  20 |

**Problem 5: (all code for this problem is listed below)**

a - c)

A 2% change in one element causes a 62.4% change in the overall solution. If I calculate the condition of the matrix A (after the perturbation) to see how difficult is to find a solution, I get 2480. Problems with higher condition numbers are said to be ill-conditioned, and 2480 is decently high. This s also how you would rate the severity of the problem. The higher the number, the less accurate any result you calculate will be.

d - f)

A 2% change in one element causes a .0098678% change in the overall solution. This is much smaller than the previous one. If I calculate the same condition number, I only get 2.0025, which is much lower than the previous as well. Since it is closer to 1, it means the severity if it being ill-conditioned, if any, is much lower.

g) cond(A) before perturbations = 901, and cond(C) before perturbations = 2. They both got worse, but cond(A) much more so, making it even more severely ill-conditioned.

|  |  |
| --- | --- |
| CODE  A = ones(9, 9);  C = ones(9, 9);    for j = 1:9  C(j, j) = 10;  end    for j = 1:9  A(j, j) = 1.01;  end  B = (1:9)';    x1 = A\B;  y1 = C\B;    condAPre = cond(A)  condCPre = cond(C)    A(5, 3) = A(5, 3) \* 1.02;  C(5, 3) = C(5, 3) \* 1.02;    x2 = A\B;  y2 = C\B;    % fprintf('\tx1(j)\t\tx2(j)\n')  % fprintf('\ty1(j)\t\ty2(j)\n')  for j = 1:9  diffX = abs(x1(j) - x2(j));  diffY = abs(y1(j) - y2(j));  % fprintf('abs(%0.4f - %0.4f) = %.4f\n', x1(j), x2(j), abs(x1(j) - x2(j)))  % fprintf('abs(%0.4f - %0.4f) = %.4f\n', y1(j), y2(j), abs(y1(j) - y2(j)))  end    percChangeX = norm(x1-x2)/norm(x1)  condA = cond(A)    percChangeY = norm(y1-y2)/norm(y1)  condC = cond(C) | OUTPUT  condAPre =  901.0000  condCPre =  2.0000  percChangeX =  0.6240  condA =  2.4803e+03  percChangeY =  9.8678e-05  condC =  2.0025 |