HOMEWORK FIVE

MATH-375

Michael Tanguay, Mathew Gervasi

Typed Explanations

1b.) Error fluctuates. Error decreases for increasing reaction rate (I may have the terminology wrong), a, then increases (See 1c. for additional information.)

1c.) Based on the table provided by the MATLAB code, different rates may cause an over estimation or underestimation. In the table, one can see the percentages fluctuate. This implies that there is an ideal rate that will minimize error. Increasing the rate increases the amount of iterations required to attain the solution leading to amounting costs. However, if some if-statement were implemented for tolerance, the iterative process may be cut short as the overarching guess of the reaction rate may reach a viable solution at less iterations. However, a high reaction rate may also overshoot significantly and induce extra iterations amounting to additional costs.

The equation used in the generat_SPD_mat_and_rhs_vec.m is:

A = -scal_fac*A + a*speye(n)   Eq [1.1]

This means that increasing the reaction rate (I may have the terminology wrong) or alpha, a, increases the second right hand term as a scaler on its diagonal elements alone, as the rest of the elements are zero. This then adds to the first term on the right-hand side to result in A. Obviously, higher values of a, increases A for its diagonal elements.

2b.) By far the conjugate gradient is the most accurate of methods for inverses. This usually implies it is costly. Moreover, because there are more costly operations (matrix*vector for the conjugate gradient as compared to scaler*scaler for the Jacobi method). This is also supported by the elapsed times and norm error displayed in the output subsection in MATLAB Code.

3a) Based on the following commands in Matlab:

```
length(find(A(1,:) == 0))
length(find(A(2,:) == 0))
length(find(A(3,:) == 0))
length(find(A(4,:) == 0))
length(find(A(5,:) == 0))
length(find(A(6,:) == 0))
length(find(A(end-1,:) == 0))
length(find(A(end,:) == 0))
```

There is an average of n-3 zeros in a matrix. Where n represents the row or column dimension. Therefore, there are on average 3 non-zeros.

3b.) See derivation on Written section.
3c.) See Written section.
3d.) Experimentally, on MATLAB (using the functions in question 1, hw5_q1), the relative norm error will plateau to a certain point not below 10^-2 for our special matrix A. This is past the 2000 iteration mark. Therefore, it is more effective to use a direct method. In fact, the relative

norm increases for increasing values of n and decrease correspondingly to the number of iterations. Especially for more complex matrices will there be a requirement of significantly more iterations.

MATLAB Code

INPUT/FUNCTIONS

1.)

```
function hw5_q1
clc, clear, close all, format compact, format
long
Alphas = [0, 1.0, 10.0, 100.0, 1000.0]';
n = 200;
tot_it = 100;

for k=1:length(Alphas)
   %Generate Linear System
   [A,b] =
generate_SPD_mat_and_rhs_vec(n,
Alphas(k));

   %Compute Solution
   x_jacobi = my_jacobi(A, b,
tot_it);%compute solution with your
my_jacobi() function

   %"True" Solution
   x_t = (A\b);
%compute norm of the error
 err_jacobi(k) = (norm(x_t -
x_jacobi)/norm(x_t));
end
NormError = err_jacobi';
Rates = Alphas;
T = table( Rates, NormError)
summary(T)
end

function[A,b] =
generate_SPD_mat_and_rhs_vec(n, a)
%Input:
%n: Positive Integer
%a: Reaction term

%Outputs:
%A: nxn matrix
%b: n vector

h = 1/(n+1);
x = (h:h:(1-h))';

my_two = -2*ones(n,1);
my_ones = ones(n-1,1);
scal_fac = (1/(h*h));
A = (diag(my_two) + diag(my_ones,1) +
diag(my_ones,-1));
A = -scal_fac*A + a*speye(n);

b = sin(2*pi*x);
b(1) = b(1) - scal_fac;
b(end) = b(end) - scal_fac;
end

function x = my_jacobi(A, b, tot_it)
tic
%Inputs:
%A: Matrix
%b: Vector
%tot_it: Number of iterations
%Output:
%:x The solution after tot_it
iterations/sweeps
x(1:length(b)) = 0; %k=1
 for k = 2:tot_it
     for i = 1:length(b)
           sum = 0;
     for j = 1:length(b)
         if j ~= i
            sum  = sum + A(i,j)*x(j);
         end
     end
     x(i) = -1/A(i,i)*(sum -b(i));
     end
 end
 toc
end
```

2.)

```
function hw5_q2
clc, clear, close all, format compact, format
long
tot_its = [5, 40, 80, 160, 320, 640, 1280];
num_experiments = length(tot_its);
%Generate Linear System
n = 200;
a = 200;
[A,b] =
generate_SPD_mat_and_rhs_vec(n,a);
err_jacobi = zeros(num_experiments,1);
err_cg = zeros(num_experiments,1);
exp_num = 1;
for tot_it =tot_its
   %Compute Solutions
   %Jacobi
   x_jacobi = my_jacobi(A,b,tot_it);
   %CG
   x_cg = my_cg(A,b, tot_it);
   %"True" Solution
   x_t = A\b;
   %Errors
   err_jacobi(exp_num) = norm(x_t -
x_jacobi)/norm(x_t);
   err_cg(exp_num) = norm(x_t -
x_cg)/norm(x_t);
   exp_num = exp_num + 1;
end
Num_Iterations = [5, 40, 80, 160, 320, 640,
1280]';
Error_Jacobi  = err_jacobi;
Error_CG = err_cg;
T = table(Num_Iterations , Error_Jacobi,
Error_CG) %Sorry, this is norm error? If
%you want relative, just replace norm with
%abs function
summary(T) %I sent an email whether you
%would like norm error or relative error, but
%didn't get a response

function x = my_cg(A, b, tot_it)
format long, tic
%Inputs:
%A: Matrix
%b: Vector
%tot_it: number of iterations to take
%
%Output:
%:x The solution after tot_it iterations
x = (ones(1,length(b))*0)'; %Initial guess x0
r = b - A*x; %r0
s = r; %s0
r1 = r;
for k = 0:tot_it
   a = r1'*r1/(s'*A*s);
   x  = x + a*s;
   r2 = r1 - a*A*s;
   B2 = r2'*r2/(r1'*r1);
   s = r2 + B2*s;
   r1 = r2;
end
toc
end

function[A,b] =
generate_SPD_mat_and_rhs_vec(n, a)
%Input:
%n: Positive Integer
%a: Reaction term

%Outputs:
%A: nxn matrix
%b: n vector

h = 1/(n+1);
x = (h:h:(1-h))';

my_two = -2*ones(n,1);
my_ones = ones(n-1,1);
scal_fac = (1/(h*h));
A = (diag(my_two) + diag(my_ones,1) +
diag(my_ones,-1));
A = -scal_fac*A + a*speye(n);

b = sin(2*pi*x);
b(1) = b(1) - scal_fac;
b(end) = b(end) - scal_fac;
end
```

```
function x = my_jacobi(A, b, tot_it)
tic
%Inputs:
%A: Matrix
%b: Vector
%tot_it: Number of iterations
%Output:
%:x The solution after tot_it
iterations/sweeps
x(1:length(b)) = 0; %k=1

for k = 2:tot_it
   for i = 1:length(b)
        sum = 0;
   for j = 1:length(b)
      if j ~= i
        sum  = sum + A(i,j)*x(j);
      end
   end
   x(i) = -1/A(i,i)*(sum -b(i));
   end
end
toc
```

<u>OUTPUTS</u>

1.)

```
Elapsed time is 0.014342 seconds.
Elapsed time is 0.013413 seconds.
Elapsed time is 0.013634 seconds.
Elapsed time is 0.013310 seconds.
Elapsed time is 0.013259 seconds.
T =
  5×2 table
    Rates       NormError
    ______    ________________
      0     13.0085440112614
      1      12.954469339591
     10      12.7560431102097
    100      12.8360569351879
   1000      13.2654073493241
Variables:
    Rates: 5×1 double
        Values:
            Min          0
            Median      10
            Max       1000
    NormError: 5×1 double
        Values:
            Min       12.7560431102097
            Median     12.954469339591
            Max       13.2654073493241
```

2.)

```
Elapsed time is 0.000568 seconds.
```

```
Elapsed time is 0.002758 seconds.
Elapsed time is 0.005348 seconds.
Elapsed time is 0.002465 seconds.
Elapsed time is 0.010628 seconds.
Elapsed time is 0.002590 seconds.
Elapsed time is 0.023502 seconds.
Elapsed time is 0.004807 seconds.
Elapsed time is 0.046391 seconds.
Elapsed time is 0.010269 seconds.
Elapsed time is 0.090625 seconds.
Elapsed time is 0.018298 seconds.
Elapsed time is 0.179794 seconds.
Elapsed time is 0.033207 seconds.
T =
  7×3 table
    Num_Iterations     Error_Jacobi          Error_CG
    ______________    ________________    ____________________
          5           13.875901986608       0.728539659116126
         40           13.3319573730442      0.0760287930897383
         80           13.0196123193756      0.00515771794926383
        160           12.6386866643278      9.73457141507309e-16
        320           12.2573885589879      9.73457141507309e-16
        640           12.0203404154858      9.73457141507309e-16
       1280           11.9654766717383      9.73457141507309e-16
Variables:
    Num_Iterations: 7×1 double
        Values:
            Min              5
            Median         160
            Max           1280
    Error_Jacobi: 7×1 double
        Values:
            Min       11.9654766717383
            Median    12.6386866643278
            Max       13.875901986608
    Error_CG: 7×1 double
        Values:
            Min       9.73457141507309e-16
            Median    9.73457141507309e-16
            Max       0.728539659116126
>>
```