Assignment 5
Iterative methods for linear systems

Matrix Computations and Applications

The deadline for this assignment can be found at:

http://www8.cs.umu.se/kurser/5DA002/HT16/timetable.html

(Link Planning and Readings on the course homepage.)

- The submission should consist of:
  - The complete report, including
    * A front page with the following information:
      1. Your name.
      2. The course name.
      4. The assignment number.
      5. The version of the submission (in case of re-submissions).
  - An appendix with the source code.
  - To simplify feedback, the main report (optionally excluding the appendix) must have numbered sections and page numbers.

- The submitted code must be Matlab-compatible. If you choose to work in Octave, verify that your code is Matlab-compatible before you submit your assignment.

- If you write your report using \LaTeX, double-check that your references are ok and not “Figure ???” before you submit.

- Your report should be submitted as a pdf file uploaded via the https://www8.cs.umu.se/~labres/py/handin.cgi page, also available as the results link at the bottom left of the course home page.

- Furthermore, the source code should be available in a folder called edu/5da002/assN in your home folder, where N is the assignment number. You will probably have to create the folder yourself.
1 Some points to keep in mind

1. It must be possible to read your report without referring to this document.
2. Above all, it is critical that you explain your reasoning to the reader.
3. The “correct” answer is literally worthless, unless it is crystal clear how it was obtained.
4. A string of unrelated equations does not represent a piece of reasoning! The equations must be connected using the appropriate logical symbols such as $\iff$, $\Rightarrow$, and $\Leftarrow$, or the equivalent written sentences.
5. You are encouraged to review each others projects before submitting them to me. In my mind, this is not cheating, but a valuable training exercise. Moreover, it gives me more time to address any nontrivial issues in your projects. Keep in mind that this is my personal viewpoint, so check with the relevant teachers before you apply this practice to other projects and courses.

2 The problems

Remark 1 The following problem stresses the theoretical foundation for Krylov subspace methods. If $A$ is non-singular, then the solution of $Ax = b$ can be written as

$$x = p(A)b$$  \hspace{1cm} (1)

for a polynomial $p$ of degree at most $n - 1$. The degree of the polynomial depends entirely on the choice of $b$ and any number between 0 and $n - 1$ is possible.

Problem 1 Consider the matrix $A \in \mathbb{R}^{5 \times 5}$ given by

$$A = VDV^{-1}$$  \hspace{1cm} (2)

where $V \in \mathbb{R}^{5 \times 5}$ is the non-singular matrix given by

$$V = \begin{bmatrix}
1 & 1 & 1 & 1 & 1 \\
5 & 6 & 7 & 8 & 9 \\
10 & 15 & 21 & 28 & 36 \\
10 & 20 & 35 & 56 & 84 \\
5 & 15 & 35 & 70 & 126
\end{bmatrix}$$  \hspace{1cm} (3)

and $D$ is the diagonal matrix

$$D = \begin{bmatrix}
1 & 2 & 3 & 4 \\
2 & 3 & 4 & 5
\end{bmatrix}$$  \hspace{1cm} (4)

1. Let $W$ be the matrix given by

$$W = \begin{bmatrix}
126 & -56 & 21 & -6 & 1 \\
-420 & 196 & -77 & 23 & -4 \\
540 & -260 & 106 & -33 & 6 \\
-315 & 155 & -65 & 21 & -4 \\
70 & -35 & 15 & -5 & 1
\end{bmatrix}$$  \hspace{1cm} (5)

Show that $W = V^{-1}$. Notice that you have several choices to pick from:
(a) Compute $V^{-1}$ be hand. Painful and error prone.
(b) Compute $V^{-1}$ using MATLAB. Painless, but subject to a different kind of errors, see below.
(c) Verify by hand that $VW = WV = I$ and quote the appropriate theorem. Doable, but time consuming.
(d) Same as above, but using MATLAB. Fast, but utterly worthless, unless you explain why MATLAB’s calculations can be trusted.

2. Why does MATLAB not return the correct answer for $V^{-1}$ when you issue the command `inv(V)`. You will have to use `format long` to disable the cosmetic rounding and reveal what MATLAB perceives $V^{-1}$ to be.

3. Show that $A$ satisfies
   \[
   A = \begin{bmatrix}
   -4 & 1 & 0 & 0 & 0 \\
   0 & -3 & 2 & 0 & 0 \\
   0 & 0 & -2 & 3 & 0 \\
   0 & 0 & 0 & -1 & 4 \\
   630 & -350 & 175 & -75 & 25
   \end{bmatrix}
   \]  
   Since MATLAB cannot compute $V^{-1}$ correctly, you will either have to do the boring calculation by hand or you will have to be clever in order to avoid rounding errors when using MATLAB.

4. Now let $b \in \mathbb{R}^5$ be given by
   \[b = V(e_2 - e_1).\]  
   Show that the Krylov matrix $K_5$ given by
   \[
   K_5 = \begin{bmatrix}
   b & Ab & A^2b & A^3b & A^4b
   \end{bmatrix}
   \]  
   has rank 2. Again you have several choices to pick from:
   (a) Apply the Arnoldi algorithm to the pair $(A, b)$. Prove that you selected the right error tolerance.
   (b) Apply the MATLAB command `rref` to compute the reduced row echelon form of $K_5$ to reveal the rank. Fast, but worthless, unless you can prove that MATLAB returned the correct result.
   (c) Apply the MATLAB command `svd` to compute the singular values. Fast, but worthless, unless you can prove that MATLAB returned the correct number of nonzero singular values.
   (d) Compute the row echelon form of $K_5$ by hand. Slow, but doable.

5. Find a polynomial $p$ of degree 2 with integer coefficients such that $p(A)b = 0$.

6. Find a real polynomial $q$ of degree 1 such that $x = A^{-1}b = q(A)b$

7. Find a new right hand side $b_1 \in \mathbb{R}^5$ such that $K_5(A, b_1)$ has dimension 3 and
   \[
   x_1 = A^{-1}b_1 = \omega(A)b_1,
   \]  
   where $\omega$ is a real polynomial of degree 2. Do not search blindly for $b_1$ but take a look at how $b$ was constructed.
Remark 2 Computers make excellent servants because they do exactly as instructed. Unfortunately, very few programmers know what they are asking and many, many programs are infested with bugs which are exceedingly hard to detect and diagnose. It is critical that you compare the output and intermediate calculations of any new program with your expectations before you rely on the program in the future.

Problem 2 Consider the MATLAB implementation of Arnoldi’s method given in Figure 1. It is written by a mathematician who knows something, but not enough, about rounding errors.

1. Before you even start to run this implementation, study the code and explain what you would like to add before the documentation could be considered satisfying!

Remark 3 If you have survived 5DV005 with me, then you know what I am talking about. Otherwise, take a close look at the function intmatrix.m which you will also find in the class repository. The difference between the two codes is rather dramatic.

2. Apply the code to the matrix $A$ and the right hand side $b$ given in Problem 1. Consider the output and compare it with your expectations. The program did not terminate at the correct value of $j$. What is the correct value of $j$?

An $A$ invariant subspace $W \subset \mathbb{R}^n$ is a subspace such that $AW \subseteq W$, i.e. $A$ maps all elements of $W$ into $W$ itself. The Arnoldi algorithm generates matrices $H = [h_{ij}]$ and

$$V(:,1:j) = [v_1 \ v_2 \ \ldots \ v_j] \quad (10)$$

such that

$$(A + \Delta A_j)V(:,1:j) = V(:,1:j)H(1:j,1:j), \quad (11)$$

where $\Delta A_j$ given by

$$\Delta A_j = -h_{j+1,j}v_{j+1}e_j^T V(:,1:j)^T, \quad e_j = (0,0,\ldots,0,1)^T \in \mathbb{R}^j \quad (12)$$

is a square matrix of the same dimension as $A$ and

$$\|\Delta A_j\|_2 = h_{j+1,j} \geq 0 \quad (13)$$

The columns of $V(:,1:j)$ span a subspace $V_j$ which is not necessarily $A$ invariant, but equation (11) expresses the fact $V_j$ is invariant for the perturbed matrix $A + \Delta A_j$ and equation (13) shows that the size of $\Delta A$ is easy to compute.

3. Return to output from the faulty implementation of Arnoldi’s method. Compute the perturbations $\Delta A_i$ and compare $\frac{\|\Delta A_i\|_2}{\|A\|_2}$ with the unit round off error.

4. Derive and implement a new stopping criterion which explicitly involves the 2-norm of $A$ and a user defined tolerance $\tau$. Do not repeat the mistake of our mathematician by forgetting to document the routine, the call sequence, including a minimal working example as well as your name and work email address.

5. Find a tolerance $\tau$ which allows your implementation of Arnoldi’s method to correctly detect the fact that $K_2(A,b)$ is $A$ invariant despite the fact that you are using floating point arithmetic.
function [H, V, m]=arnoldi(A,b)

% Extract dimension of the matrix
[n, k]=size(A);

% Extract the dimension of the right hand side vector
[r, s]=size(b);

% Perform dummy initialization of output variables
H=[]; V=[];

% Guard against user stupidity
if (n~=k) flag=-1; return; end;
if (n~=r) flag=-2; return; end;
if (s~=1) flag=-3; return; end;
aux=norm(b); if (aux==0) flag=-4; return; end;

% If we reach this point then A is a square matrix and b is nonzero.

% Initialize the output properly
V=zeros(n,n); H=zeros(n,n);

% Normalize the first vector
V(:,1)=b/aux;

% Create the matrices H and V
for j=1:n-1
    % Apply A to the last basis vector
    w=A*V(:,j);
    % Make w orthogonal to all vectors V(:,i)
    for i=1:j
        H(i,j)=V(:,i)'*w; w=w-H(i,j)*V(:,i);
    end
    % Save the value of the norm of w
    aux=norm(w);
    % Define the critical subdiagonal coefficient
    H(j+1,j)=aux;
    % Test for invariant subspace
    if (aux==0)
        % Invariant subspace detected
        m=j; break;
    else
        % Current subspace is not invariant, add a new vector to V
        V(:,j+1)=w/aux; m=j+1;
    end
end
% Discard any dummy rows/columns of zeros which we do not need
H=H(1:m,1:m); V=V(:,1:m);

Figure 1: An implementation of Arnoldi’s method which does not work properly in floating point arithmetic. You will find this function in the class repository.
Problem 3 The following problem deal with a linear systems of the form

\[ A_s x = (A + sI)x = b, \quad s > 0 \]  \hspace{1cm} (14)

where \( A \) is the matrix given by the file bone010.mat which you will find the class repository. It is known that the matrix \( A \) is symmetric positive definite, but we will choose \( s = 10^4 \) to circumvent some numerical problems. The dimension of this problem is \( n = 986703 \).

1. An application of the \texttt{spy} command will reveal that the matrix \( A \) can be viewed as a banded matrix. Use the \texttt{find} command to extract the sparsity pattern of the matrix and compute the bandwidth \( k \) of the matrix, i.e.

\[ k = \max\{|i - j| : \exists a_{ij} \neq 0\} \]  \hspace{1cm} (15)

2. How many GB of memory are required to store the matrix as a banded matrix using double precision numbers.

3. Explain why it is impractical to compute the Cholesky factorization of \( A_s \) if we treat it as a banded matrix.

4. Let \( x = \text{ones}(n,1) \) denote the vector of ones and let \( b = A \ast x \). Is it possible to recover \( x \) using the direct solver in \texttt{MATLAB}. When did you run out of patience?

5. Construct the simplest incomplete Cholesky factor \( L \) from \( A_s \) using the \texttt{ichol} command.
   (a) How much time did the construction require?
   (b) Verify that \( L \) has the correct number of nonzero elements.

   \textbf{Remark 4} How does the sparsity pattern of \( L \) compare to the sparsity pattern of \( A \) for this simple preconditioner?

6. Apply the preconditioned CG algorithm to the problem of solving \( Ay = b \), i.e. recovering the vector \( x \).
   (a) How much time do you need to reduce the preconditioned relative residual below the tolerance \( \texttt{tol} = 1\text{e-14} \)?
   (b) What is the corresponding value of the relative residual for the original linear system?
   (c) How many iterations where needed?
   (d) What has the largest componentwise relative error when comparing the correct solution \( x \) with the computed approximation \( \hat{y} \)?

7. Is it theoretically possible for you to get this componentwise relative error down below \( 2^{-53} \) using double precision numbers?