# Lab Assessment for Module 2

**MA-423 :** Matrix Computations 14 October, 2020 S. Bora

**Marks: 28**

**Important instructions:**

- **Half the score in this assessment will count towards the final lab score of Module 2.**
- **Put all your work in a single folder and submit a link to download it on MS Teams within the declared deadline.**
- **Submitted folder should contain** *only* **the programs written for the purpose of the assessment and other supporting programs that are required for those programs to run. If there are any unnecessary programs in the folder, then upto 2 marks may be deducted from the score of every group member.**
- **All data for questions 3 and 4(b) must be presented in `format long e`.**
- **All workspace files for question 3** *must* **be prepared and submitted as per instructions.**
- **All data and observations for each question must be neatly recorded in separate pdf files with appropriate explanations and conclusions wherever necessary inside your submitted folder. Scans of pieces of paper will not be accepted!**
- **Any kind of cheating or copying from internet will result in heavy penalties for every group member.**

1. Given $kappa (\geq 1) \in \mathbb{R}$, a random $n \times n$ positive definite matrix $A$ with $\kappa(A) = kappa$, may be formed by generating a random orthogonal matrix $U$ and a diagonal matrix $D$ with $D(i,i) = kappa^{(i-1)/(n-1)}, i = 1 : n$, and setting $A = U * D * U'$. The following command will generate $U$:

   `>> X = randn(n, n); [U, R] = qr(X);`

   Use the above information to write a function program `A = condmat(n, kappa)` that generates a random $n \times n$ positive definite matrix $A$ with given condition number *kappa*.

   **(2 marks)**

2. Write a function program `x = cramer(A, b)` to find the solution of a system of equations $Ax = b$ by Cramer's rule. **(3 marks)**

3. Generate a random positive definite matrix $A$ of size $n = 100$ with condition number $kappa = 10^5$ with the help of the function program `condmat`. Set up a system $Ax = b$ whose exact solution is known by using the technique described in the Lab sheet 3. Find the relative forward error, the ratio of the norm of the residual by the norm of $b$ (all in the $\infty$-norm) and the time taken for the computed solution obtained by using the Cholesky factor of $A$. (`A\b`

will get this solution). Save the workspace in a file named $\texttt{Q3} - \texttt{100.mat}$. (It should contain the matrix $A$ and the exact solution $x$ that you have used).

Repeat the process for the computed solution obtained by Cramer's rule. Record your answers in `format long e` in the following tabular form:

$\kappa(A) = 10^5$

| Method | forward error | $\frac{\text{norm of residual}}{\text{norm of b}}$ | time |
|---|---|---|---|
| Cholesky | | | |
| Cramer's rule | | | |

Clear the previous workspace and repeat the entire process for another positive definite $85 \times 85$ system $Ax = b$ with $\kappa(A) = 10^7$ generated by using the `condmat` program. Save the new workspace as $\texttt{Q3} - \texttt{85.mat}$. Record your answers once again in `format long e` using the above tabular form in the same document file that you have created earlier. Add the answers of the following questions to your file. **(6)**

[**Marking Scheme:** 1/2 mark for each correct answer.]

(a) Does the computed solution satisfy the *Rule-of-thumb* of ill conditioning as stated in Lab sheet 3 in each case? State the rule before providing your answer. **(2)**
[**Marking Scheme:** 1/2 mark for each correct answer.]

(b) What can you say about the backward stability of the methods used for the given problems from the above data? Justify your answer in each case. **(3)**
[**Marking Scheme:** 1/2 mark for each correct answer and 1 mark for the justification (as the concept involved is common to all the answers).]

(c) Which method is the more expensive and why? **(1)**

4. Given a solution $x_c$ of an $n \times n$ system of equations $Ax = b$, let $r = b - Ax_c$ be the residual vector. Let xhat be the solution of $Ax = r$. Then $x_{\text{new}} := x_c + \text{xhat}$ is theoretically the exact solution of $Ax = b$. But there will be errors in the calculation and practically $x_{\text{new}}$ will not be the exact solution. However, it is expected to be a more accurate solution than $x_c$. The process can be repeated by setting $x_c = x_{\text{new}}$ and performing the above steps to get a fresh $x_{\text{new}}$. The iterations can be terminated when the maximum number of repetitions allowed (this should be much smaller than $n$) is exceeded or $\frac{\|\text{xhat}\|}{\|x_c\|}$ becomes less than a predefined tolerance. The is called **iterative refinement**.

   (a) Let $L, U, p$ be the outputs of $[\texttt{L}, \texttt{U}, \texttt{p}] = \texttt{gepp}(\texttt{A})$, $x_c$ the output of $\texttt{x}_\texttt{c} = \texttt{geppsolve}(\texttt{A}, \texttt{b})$, $m(\ll n)$ be a positive integer and `tol` a small positive number so that the iterations of the refinement process stop if either the relative refinements $\frac{\|\texttt{xhat}\|_\infty}{\|\texttt{x}_\texttt{c}\|_\infty} < \texttt{tol}$ or the number of iterations exceed $m$.
   Write an *efficient* function program $\texttt{x} = \texttt{iteref}(\texttt{A}, \texttt{b}, \texttt{x}_\texttt{c}, \texttt{L}, \texttt{U}, \texttt{p}, \texttt{m}, \texttt{tol})$ that iteratively refines the computed solution $\texttt{x}_\texttt{c}$ of $Ax = b$ to $\texttt{x}$ via the process described above in $O(n^2)$ flops. **(5)**

   (b) Set $\texttt{A} = \texttt{Wilkinson}(64)$ and set up a system $Ax = b$ for which the exact solution $x$ is known via the trick shown in Lab sheet 3. Compute a solution $\texttt{x}_\texttt{c} = \texttt{geppsolve}(\texttt{A}, \texttt{b})$ and its refinement $\texttt{x}$ via the above algorithm with $m = 2$. Find the norms of the residual and the forward errors associated with $\texttt{x}_\texttt{c}$ and $\texttt{x}$. **(2)**

5. Run each of the following programs and give full explanations of the values of $x$ and $k$ that you get in each case. **(2+2 = 4)**

(i)

```
x = 1; k = 0;
while 1 + x > 1
x = x/2; k = k+1;
end
```

(ii)

```
x = 1; k = 0;
while x + x > x
x = x/2; k = k+1;
end
```