MSSC 6040 - Homework 6

Henri Medeiros Dos Reis

December 6, 2022

Problem 1 (5 pts)**.** Compute by hand an LDL-decomposition and a Cholesky decomposition of the following matrix.

$$A = \begin{bmatrix} 4 & 1 & 1 \\ 1 & 2 & 0 \\ 1 & 0 & 1 \end{bmatrix}.$$

(i.e., first compute the LDL-decomposition then derive the Cholesky decomposition from it). Check your answer using the MATLAB commands `[L,D] = ldl(A)` and `R = chol(A)`.

Attempt to repeat the same steps for the matrix:

$$B = \begin{bmatrix} 4 & 1 & 1 \\ 1 & 2 & -1 \\ 1 & -1 & 1 \end{bmatrix}.$$

Something will prevent you from being able to compute the Cholesky decomposition. What is it?

Solution 1.

$$A = \begin{bmatrix} 4 & 1 & 1 \\ 1 & 2 & 0 \\ 1 & 0 & 1 \end{bmatrix} \Rightarrow \begin{bmatrix} 4 & 1 & 1 \\ 0 & \frac{7}{4} & \frac{-1}{4} \\ 0 & \frac{-1}{4} & \frac{3}{4} \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & 0 & 0 \\ \frac{-1}{4} & 1 & 0 \\ \frac{-1}{4} & 0 & 1 \end{bmatrix} A$$

Now we can repeat on columns

$$A^{(1)} = \begin{bmatrix} 4 & 0 & 0 \\ 0 & \frac{7}{4} & \frac{-1}{4} \\ 0 & \frac{-1}{4} & \frac{3}{4} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ \frac{-1}{4} & 1 & 0 \\ \frac{-1}{4} & 0 & 1 \end{bmatrix} A \begin{bmatrix} 1 & \frac{-1}{4} & \frac{-1}{4} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \Rightarrow$$

$$\begin{bmatrix} 4 & 0 & 0 \\ 0 & \frac{7}{4} & 0 \\ 0 & 0 & \frac{5}{7} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & \frac{1}{7} & 1 \end{bmatrix} A^{(1)} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & \frac{1}{7} \\ 0 & 0 & 1 \end{bmatrix} \Rightarrow$$

$$L_2 L_1 A L_1^* L_2^* = D$$

$$L = L_1^{-1}L_2^{-2} = \begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{4} & 1 & 0 \\ \frac{1}{4} & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & \frac{-1}{7} & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{4} & 1 & 0 \\ \frac{1}{4} & \frac{-1}{7} & 1 \end{bmatrix}$$

$$D = \begin{bmatrix} 4 & 0 & 0 \\ 0 & \frac{7}{4} & 0 \\ 0 & 0 & \frac{5}{7} \end{bmatrix}$$

Now for the Cholesky decomposition, let $R = D^{\frac{1}{2}}L^*$, $A = R^*R$

$$R = \begin{bmatrix} \sqrt{4} & 0 & 0 \\ 0 & \sqrt{\frac{7}{4}} & 0 \\ 0 & 0 & \sqrt{\frac{5}{7}} \end{bmatrix} \begin{bmatrix} 1 & \frac{1}{4} & \frac{1}{4} \\ 0 & 1 & \frac{-1}{7} \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 2 & \frac{1}{2} & \frac{1}{2} \\ 0 & \frac{\sqrt{7}}{2} & \frac{-\sqrt{7}}{14} \\ 0 & 0 & \frac{\sqrt{35}}{7} \end{bmatrix}$$

```
A = [4 1 1; 1 2 0 ;1 0 1]
[L,D] = ldl(A)
R = chol(A)
%-------------- OUTPUT---------------------------
%L =

%    1.0000         0         0
%    0.2500    1.0000         0
%    0.2500   -0.1429    1.0000


%D =

%    4.0000         0         0
%         0    1.7500         0
%         0         0    0.7143


%R =

%    2.0000    0.5000    0.5000
%         0    1.3229   -0.1890
%         0         0    0.8452
```

For the second part of the problem, B is not a positive definite matrix, so Cholesky decomposition is not possible, but let's follow the same steps as before.

$$B = \begin{bmatrix} 4 & 1 & 1 \\ 1 & 2 & -1 \\ 1 & -1 & 1 \end{bmatrix} \Rightarrow \begin{bmatrix} 4 & 1 & 1 \\ 0 & \frac{7}{4} & \frac{-5}{4} \\ 0 & \frac{-5}{4} & \frac{3}{4} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ \frac{-1}{4} & 1 & 0 \\ \frac{-1}{4} & 0 & 1 \end{bmatrix} B$$

Now we can repeat on columns

$$B^{(1)} = \begin{bmatrix} 4 & 0 & 0 \\ 0 & \frac{7}{4} & \frac{-5}{4} \\ 0 & \frac{-5}{4} & \frac{3}{4} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ \frac{-1}{4} & 1 & 0 \\ \frac{-1}{4} & 0 & 1 \end{bmatrix} B \begin{bmatrix} 1 & \frac{-1}{4} & \frac{-1}{4} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \Rightarrow$$

$$\begin{bmatrix} 4 & 0 & 0 \\ 0 & \frac{7}{4} & 0 \\ 0 & 0 & \frac{-1}{7} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & \frac{5}{7} & 1 \end{bmatrix} A^{(1)} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & \frac{5}{7} \\ 0 & 0 & 1 \end{bmatrix} \Rightarrow$$

$$L_2 L_1 A L_1^* L_2^* = D$$

$$L = L_1^{-1} L_2^{-2} = \begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{4} & 1 & 0 \\ \frac{1}{4} & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & \frac{-5}{7} & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{4} & 1 & 0 \\ \frac{1}{4} & \frac{-5}{7} & 1 \end{bmatrix}$$

$$D = \begin{bmatrix} 4 & 0 & 0 \\ 0 & \frac{7}{4} & 0 \\ 0 & 0 & \frac{-1}{7} \end{bmatrix}$$

Now for the Cholesky decomposition, let $R = D^{\frac{1}{2}} L^*$, $B = R^* R$

Where $D^{\frac{1}{2}} = \begin{bmatrix} \sqrt{4} & 0 & 0 \\ 0 & \sqrt{\frac{7}{4}} & 0 \\ 0 & 0 & \sqrt{\frac{-1}{7}} \end{bmatrix}$, and we cannot proceed.

Problem 2 (5 pts)**.** Many applications in statistics (e.g., Monte-Carlo methods) require generating samples from a multivariate Gaussian distribution $\mathcal{N}(\mu, \Sigma)$ with a known mean $\mu \in \mathbb{R}^m$ is and covariance matrix $\Sigma \in \mathbb{R}^{m \times m}$. Suppose $R \in \mathbb{R}^{m \times m}$ is any matrix such that $\Sigma = R^* R$. Then one can show that samples of $\mathcal{N}(\mu, \Sigma)$ can be generated as $R^* z + \mu$ where $z \in \mathbb{R}^m$ is a vector whose entries are sampled from a standard normal distribution (i.e., in MATLAB, `z = randn(m,1)`).

(a) As a warm-up, consider the Gaussian distribution $\mathcal{N}(\mu, \Sigma)$ where the mean $\mu \in \mathbb{R}^d$ and covariance matrix $\Sigma \in \mathbb{R}^{3 \times 3}$ are given by:

$$\mu = \begin{bmatrix} 1 \\ -1 \\ 0.5 \end{bmatrix}, \quad \Sigma = \begin{bmatrix} 1 & 0.5 & 0.5 \\ 0.5 & 2 & 0.3 \\ 0.5 & 0.3 & 1.5 \end{bmatrix}.$$

Using a Cholesky decomposition of Σ, generate $N = 10,000$ random samples from the distribution $\mathcal{N}(\mu, \Sigma)$, and store them as the rows of a matrix $X \in \mathbb{R}^{N \times 3}$. Verify that the sample covariance matrix `C = cov(X)` computed from these samples is close to Σ.

(b) Open the script `montecarlo.mat`. The script loads a matrix `X`, whose rows are vectorized 32×32 image of a similar type. Treating these images as random samples from a Gaussian distribution, we may compute their sample mean μ and sample covariance matrix Σ. These are computed for you in the `mu` and `Sigma` variables. Using a Cholesky decomposition, write a code snippet to generate a random sample from the distribution $\mathcal{N}(\mu, \Sigma)$, and visualize the samples as an image using the `imagesc` command (see the example provided at the top of the script). In your write-up, include images of three of your favorite of the generated random samples, and give them affectionate names.

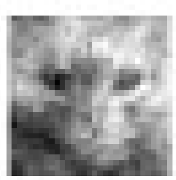
Solution 2. a-) .

```
mu = [1 ;-1; 0.5];
sigma = [1 0.5 0.5;0.5 2 0.3;0.5 0.3 1.5];
csig = chol(sigma);
n = 10000;
X = zeros([n,3]);
for i=1:n
    X(i,:) =  (csig'*randn(3,1) + mu);
end

C = cov(X);
C
%-------------- OUTPUT---------------------------
%C =

%    1.0004    0.5070    0.5143
%    0.5070    2.0104    0.2881
%    0.5143    0.2881    1.5402
```

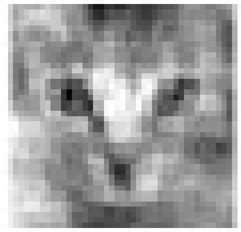
b-) .

```
figure(2)
samp_img =  (chol(Sigma)'*randn(1024,1) + mu);
samp_img = reshape(samp_img,dim);
imagesc(samp_img); axis image; axis off; colormap gray
```

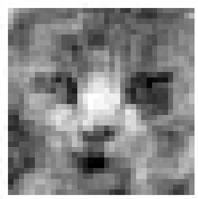
This is lp (short for little princess)



This is clumsy



This is cutie



Problem 3 (5 pts)**.** Suppose $A \in \mathbb{C}^{m\times m}$ is positive definite. Prove that $\|x\|_A := \sqrt{x^*Ax}$ is a vector norm. [Hint: Cholesky].

Solution 3.

$$\|x\|_A := \sqrt{x^*Ax}, \text{ using Cholesky decomposition, } \sqrt{x^*Ax} = \sqrt{x^*R^*Rx}$$

Since Rx is a vector, and x^*R^* is the transpose of that vector, we can use the fact that $v^*v = \|v\|_2^2$, for any vector. Then we have:

$$\sqrt{x^*R^*Rx} = \sqrt{\|Rx\|_2^2} = \|Rx\|_2$$

Problem 4 (5 pts)**.** For this problem, start with the provided MATLAB script `poisson.m`. This script considers a discrete approximation of the Poisson equation given by $Ax = b$ where A is a discrete approximation of the negative Laplacian, b is the source distribution, and x is the potential to be recovered. Here, the matrix A is defined implicitly as a function in MATLAB. To compute the matrix-vector product Ax use `A(x)` in MATLAB, rather than `A*x`.

Your task is to compare the steepest descent and conjugate gradients algorithms for solving this problem. The provided script already implements the steepest descent algorithm.

(a) Run the steepest descent (SD) algorithm using the initialization $x_0 = b$ and exit condition $\delta_{k+1} := r_{k+1}^* r_{k+1} < 0.01$ (these are the defaults in the script). How many iterations k of SD does it take to reach the exit condition? Include an image of the final iterate output by the algorithm.

(b) Modify the steepest descent code to implement the conjugate gradients (CG) algorithm. Run CG with the same initialization and the same exit condition, $\delta_{k+1} < 0.01$. How many iterates k does it take to reach the exit condition? Include a print-out/screenshot of your code in your write-up, along with an image of the final iterate output by the algorithm.

Solution 4. a-) The algorithm took 3547 iterations to converge.

iteration k=3547, delta=0.009991

b-) The algorithm took 442 iterations to converge.

```
x = b;
```

```
r = b-A(x);
p = r;
delta = r'*r;
for k=1:5000
    S = A(p);
    alpha = delta/(p'*S);
    x = x + alpha*p;
    r = r - alpha*S;
    prev_delt = delta;
    delta = r'*r;
    p = r+delta/prev_delt*p;



    %visualize current iterate x (comment out to speed up
       computation)
    if mod(k,10)==2 %every 10 iterates update the plot
        figure(1);
        imshow(imresize(unvec(x),2,'nearest'),[]);
        title(sprintf('iteration k=%d, delta=%f',k,delta),'
           fontsize',16);
    end

    if delta < 0.01
        fprintf('reached exit tol at iter %d\n',k);
        break;
    end
end
```

iteration k=442, delta=0.009461

Problem 5 (5 pts)**.** Recall that the iterates of the steepest descent algorithm for solving $Ax = b$ with A positive definite are given by

$$x_{k+1} = x_k + \alpha_k r_k,$$

where $r_k = b - Ax_k$ and $\alpha_k = \frac{r_k^* r_k}{r_k^* A r_k}$. Prove that $(x_{k+2} - x_{k+1})^*(x_{k+1} - x_k) = 0$ for all $k \geq 0$. [Hint: it suffices to show $r_{k+1}^* r_k = 0$ (Why?). Now expand r_{k+1} in terms of r_k and see what happens.]

Solution 5.

$$(x_{k+2} - x_{k+1})^*(x_{k+1} - x_k) = (x_{k+1} + \alpha_{k+1} r_{k+1} - x_{k+1})^*(x_k + \alpha_k r_k - x_k) \Rightarrow$$

$$(\alpha_{k+1} r_{k+1})^*(\alpha_k r_k) = r_{k+1}^* \alpha_{k+1}^* \alpha_k r_k$$

Since α is a scalar, we can move it up front

$$(\alpha_{k+1}\alpha_k)(r_{k+1}^* r_k) = 0 \Rightarrow r_{k+1}^* r_k = 0$$

Then, let's expand r_{k+1}^*

$$r_{k+1}^* = (b - Ax_{k+1})^* = (b - A(x_k + \alpha_k r_k))^* = b^* - (x_k + \alpha_k r_k)^* A^*$$

Then

$$r_{k+1}^* r_k = (b^* - (x_k + \alpha_k r_k)^* A^*) r_k = b^* r_k - (x_k + \alpha_k r_k)^* A^* r_k$$

$$\Rightarrow b^* r_k - \left[x_k + \frac{r_k^* r_k}{r_k^* A r_k} r_k \right]^* A^* r_k = b^* r_k - \left[x_k^* + \frac{r_k^* r_k}{r_k^* A r_k} r_k^* \right] A^* r_k$$

$$\Rightarrow b^* r_k - \left(x_k^* A^* r_k + \frac{r_k^* r_k}{r_k^* A r_k} r_k^* A^* r_k \right) = b^* r_k - (x_k^* A^* r_k + r_k^* r_k)$$

$$\Rightarrow b^* r_k - (x_k^* A^* (b - Ax_k) + (b - Ax_k)^* (b - Ax_k))$$

$$\Rightarrow b^* r_k - (x_k^* A^* b - x_k^* A^* A x_k + (b^* - x_k^* A^*)(b - Ax_k))$$

$$\Rightarrow b^* r_k - (x_k^* A^* b - x_k^* A^* A x_k + b^* b - b^* A x_k - x_k^* A^* b + x_k^* A^* A x_k)$$

$$\Rightarrow b^* r_k - (b^* b - b^* A x_k) = b^* (b - Ax_k) - (b^* b - b^* A x_k)$$

$$\Rightarrow b^* b - b^* A x_k - b^* b + b^* A x_k = 0$$

Therefore, since $r_{k+1}^* r_k = 0$, then $(x_{k+2} - x_{k+1})^* (x_{k+1} - x_k) = 0$.