The University of Texas
at Austin

CS383C Numerical Analysis

Final Exam

Edited by LaTeX

Department of Computer Science

STUDENT

Jimmy Lin

xl5224

COURSE COORDINATOR

Robert A. van de Geijn

UNIQUE NUMBER

53180

RELEASE DATE

Dec. 2 2014

DUE DATE

Dec. 10 2014

TIME SPENT

10 hours

December 10, 2014

Exercises

Part I
Cholesky Factorization

1 SPD

1.1 Show A_{00} is SPD

Proof. Since A is SPD, then

$$A^T = A \tag{1}$$

$$\forall x,\ x^T A x \geq 0 \tag{2}$$

From (1), we have

$$A^T = \begin{pmatrix} A_{00} & a_{10} \\ a_{10}^T & \alpha_{11} \end{pmatrix}^T = \begin{pmatrix} A_{00}^T & a_{10} \\ a_{10}^T & \alpha_{11} \end{pmatrix} = A \tag{3}$$

Then

$$A_{00}^T = A_{00} \tag{symmetry of A}$$

Also, we denote arbitrary $x = \begin{pmatrix} x_0 \\ \chi_1 \end{pmatrix}$ and then from (2), we have

$$x^T A x = \begin{pmatrix} x_0 \\ \chi_1 \end{pmatrix}^T \begin{pmatrix} A_{00} & a_{10} \\ a_{10}^T & \alpha_{11} \end{pmatrix} \begin{pmatrix} x_0 \\ \chi_1 \end{pmatrix} = x_0^T A_{00} x_0 + 2\chi_1 a_{10}^T x_0 + \chi_1^2 \alpha_{11} > 0,\ \forall x_0, \chi_1 \tag{4}$$

Let $\chi_1 = 0$, then we have

$$x_0^T A_{00} x_0 > 0,\ \forall x_0 \tag{positive definiteness}$$

In terms of (symmetry of A) and (positive definiteness), then it is proved that A_{00} is SPD. □

1.2 $l_{10}^T = a_{10}^T L_{00}^{-T}$ is well defined

Since L_{00} is non-singular, then it is easy to derive that L_{00}^T is also non-singluar. Then L_{00}^{-T} exists. Hence,

$$l_{10}^T = a_{10}^T L_{00}^{-T} \tag{5}$$

is well-defined.

1.3 $\alpha_{11} - l_{10}^T l_{10} > 0$

Let partition arbitrary $x = \begin{pmatrix} x_0 \\ \chi_1 \end{pmatrix}$, then since A is SPD, we have

$$\begin{aligned} x^T A x &= \begin{pmatrix} x_0 \\ \chi_1 \end{pmatrix}^T \begin{pmatrix} A_{00} & a_{10} \\ a_{10}^T & \alpha_{11} \end{pmatrix} \begin{pmatrix} x_0 \\ \chi_1 \end{pmatrix} \\ &= x_0^T A_{00} x_0 + 2\chi_1 x_0^T a_{10} + \chi_1 \alpha_{11} > 0 \end{aligned} \tag{6}$$

Now we instantiate $\begin{pmatrix} -A_{00}^{-1} a_{10} \\ 1 \end{pmatrix}$, then from (6), we have

$$\begin{aligned} & x'^T A x' > 0 \\ \Leftrightarrow\quad & a_{10}^T A_{00}^{-T} A_{00} A_{00}^{-1} a_{10} - 2 a_{10}^T A_{00}^{-T} a_{10} + \alpha_{11} > 0 \\ \Leftrightarrow\quad & a_{10}^T A_{00}^{-T} a_{10} - 2 a_{10}^T A_{00}^{-T} a_{10} + \alpha_{11} > 0 \\ \Leftrightarrow\quad & \alpha_{11} - a_{10}^T A_{00}^{-T} a_{10} > 0 \\ \Leftrightarrow\quad & \alpha_{11} - a_{10}^T A_{00}^{-1} a_{10} > 0 \\ \Leftrightarrow\quad & \alpha_{11} - a_{10}^T L_{00}^{-T} L_{00}^{-1} a_{10} > 0 \\ \Leftrightarrow\quad & \alpha_{11} - l_{10}^T l_{10} > 0 \end{aligned} \tag{7}$$

1.4 Show equality

$$L \cdot L^T = \begin{pmatrix} L_{00} & l_{10} \\ l_{10}^T & \lambda_{11} \end{pmatrix} \begin{pmatrix} L_{00} & l_{10} \\ l_{10}^T & \lambda_{11} \end{pmatrix}^T = \begin{pmatrix} L_{00}L_{00}^T & L_{00}l_{10} \\ l_{10}^T L_{00}^T & l_{10}^T l_{10} + \lambda_{11}^2 \end{pmatrix} \tag{8}$$

Obviously, $L \cdot L^T = A$ if only if

$$\begin{aligned} A_{00} &= L_{00}L_{00}^T \\ a_{10} &= L_{00}l_{10} \\ \alpha_{11} &= L_{10}^T l_{10} + \lambda_{11}^2 \end{aligned} \tag{9}$$

2

2.1 Another proof of Cholesky Factorization Theorem

Proof by induction.

- **Base Case**: $n = 1$. Obviously, $A = L_{00}L_{00}^T$ holds. Say, $A = \alpha_{11}$. In this case, we have $L_{00} = \sqrt{\alpha_{11}}$.
- **Inductive Cases**: Assume the result is true for SPD matrix $A \in \mathbb{R}^{(n-1)\times(n-1)}$. We will show that it holds for $A \in \mathbb{R}^{n\times n}$. Partition A and L as indicated on the instruction. Let

$$l_{10}^T = a_{10}^T \cdot L_{00}^{-T} \tag{10}$$

$$\lambda_{11} = \sqrt{\alpha_{11} - l_{10}^T l_{10}} \tag{11}$$

  Then L is the desired Cholesky factor of L, that is, $A = LL^T$.
- By the principle of mathematical induction, the theorem holds.

2.2 Bordered Cholesky Algorithm

```
% Copyright 2014 The University of Texas at Austin
%
% For licensing information see
%                http://www.cs.utexas.edu/users/flame/license.html
%
% Programmed by: Jimmy Xin Lin
%                jimmylin@utexas.edu

function [ A_out ] = BCA_unb( A )

  [ ATL, ATR, ...
    ABL, ABR ] = FLA_Part_2x2( A, ...
                               0, 0, 'FLA_TL' );

  while ( size( ATL, 1 ) < size( A, 1 ) )

    [ A00,  a01,     A02,  ...
      a10t, alpha11, a12t, ...
      A20,  a21,     A22 ] = FLA_Repart_2x2_to_3x3( ATL, ATR, ...
                                                    ABL, ABR, ...
                                                    1, 1, 'FLA_BR' );

    %------------------------------------------------------------%

    a10t = a10t * inv(tril(A00))';
    alpha11 = sqrt(alpha11 - a10t * a10t');

    %------------------------------------------------------------%

    [ ATL, ATR, ...
      ABL, ABR ] = FLA_Cont_with_3x3_to_2x2( A00,  a01,     A02,  ...
                                             a10t, alpha11, a12t, ...
                                             A20,  a21,     A22, ...
                                             'FLA_TL' );

  end

  A_out = [ ATL, ATR
            ABL, ABR ];

return
```

3 Cost of Bordered Algorithm

At the iteration i,

- $a10t$ update: i^2 (multiplication)
- $\alpha11$ udpate: i (subtraction) + i (dot product)

$$\text{total cost } = \sum_{i=1}^{n}(i^2+2i) = \frac{1}{3}n^3 + n^2 \approxeq \frac{1}{3}n^3 \tag{12}$$

Part II
Method of Relatively Robust Representations

1 LDL^T Factorization for indefinite matrices

```
% Copyright 2014 The University of Texas at Austin
%
% For licensing information see
%                http://www.cs.utexas.edu/users/flame/license.html
%
% Programmed by: Jimmy Xin Lin
%                jimmylin@utexas.edu

function [ A_out ] = LDL_unb( A )

  [ ATL, ATR, ...
    ABL, ABR ] = FLA_Part_2x2( A, ...
                               0, 0, 'FLA_TL' );
  while ( size( ATL, 1 ) < size( A, 1 ) )

    [ A00,  a01,     A02,  ...
      a10t, alpha11, a12t, ...
      A20,  a21,     A22 ] = FLA_Repart_2x2_to_3x3( ATL, ATR, ...
                                                    ABL, ABR, ...
                                                    1, 1, 'FLA_BR' );

    %------------------------------------------------------------%

    l21 = a21 / alpha11;
    A22 = A22 - l21 * a21';
    a21 =  l21;

    %------------------------------------------------------------%

    [ ATL, ATR, ...
      ABL, ABR ] = FLA_Cont_with_3x3_to_2x2( A00,  a01,     A02,  ...
                                             a10t, alpha11, a12t, ...
                                             A20,  a21,     A22, ...
                                             'FLA_TL' );
  end

  A_out = [ ATL, ATR
            ABL, ABR ];

return
```

2 LDL^T Factorization for tridiagonal matrices

Codes:

```
function [ A_out ] = LDL_TRI_unb( A )

    % here we leave out the partion and repartion code in the notes

    %------------------------------------------------------------%

    alpha11 = alpha11 % = delta11 (no-op)
    l21 = alpha21 / alpha11; % scalar
    alpha22 = alpha22 - l21 * alpha21;
    alpha21 =  l21;

    %------------------------------------------------------------%

    % here we leave out the continue code in the notes

return
```

Costs:

- Divide: $1 \cdot n = n$ (l21 update)
- Multiply: $1 \cdot n = n$ (alpha22 update)
- Add/Subtract: $1 \cdot n = n$ (alpha21 update)

In terms of above analysis, the approximate cost is $\mathcal{O}(n)$.

Analytics: The way I come up with this algorithm is to instantiate the LDL^T factorization in last question to the case of tridiagonal matrices. That is, treat the $alpha21$ and $l21$ as vectors with only one non-zero entry.

3 UDU^T Factorization for indefinite matrices

```
% Copyright 2014 The University of Texas at Austin
%
% For licensing information see
%                http://www.cs.utexas.edu/users/flame/license.html
%
% Programmed by: Jimmy Xin Lin
%                jimmylin@utexas.edu

function [ A_out ] = UDU_unb( A )

  [ ATL, ATR, ...
    ABL, ABR ] = FLA_Part_2x2( A, ...
                               0, 0, 'FLA_BR' );

  while ( size( ABR, 1 ) < size( A, 1 ) )

    [ A00,  a01,     A02,  ...
      a10t, alpha11, a12t, ...
      A20,  a21,     A22 ] = FLA_Repart_2x2_to_3x3( ATL, ATR, ...
                                                    ABL, ABR, ...
                                                    1, 1, 'FLA_TL' );

    %------------------------------------------------------------%

    % alpha11 = alpha11 = delta11 (no-operation)
    u01 = a01 / alpha11;
    A00 = A00 - u01 * a01';
    a01 = u01;

    %------------------------------------------------------------%

    [ ATL, ATR, ...
      ABL, ABR ] = FLA_Cont_with_3x3_to_2x2( A00,  a01,     A02,  ...
                                             a10t, alpha11, a12t, ...
                                             A20,  a21,     A22, ...
                                             'FLA_BR' );

  end

  A_out = [ ATL, ATR
            ABL, ABR ];

return
```

Let $A = \begin{pmatrix} A_{00} & a_{01} \\ a_{01}^T & \alpha_{11} \end{pmatrix}$, $U = \begin{pmatrix} U_{00} & u_{01} \\ 0 & 1 \end{pmatrix}$ and $D = \begin{pmatrix} D_{00} & 0 \\ 0 & \delta_1 \end{pmatrix}$.

$$\begin{pmatrix} A_{00} & a_{01} \\ a_{01}^T & \alpha_{11} \end{pmatrix} = \begin{pmatrix} U_{00} & u_{01} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} D_{00} & 0 \\ 0 & \delta_1 \end{pmatrix} \begin{pmatrix} U_{00} & u_{01} \\ 0 & 1 \end{pmatrix}^T \tag{13}$$

$$= \begin{pmatrix} U_{00}D_{00} & \delta_1 u_{01} \\ 0 & \delta_1 \end{pmatrix} \begin{pmatrix} U_{00} & u_{01} \\ 0 & 1 \end{pmatrix}^T \tag{14}$$

$$= \begin{pmatrix} U_{00}D_{00}U_{00}^T + \delta_1 u_{01}u_{01}^T & \delta_1 u_{01} \\ \delta_1 u_{01}^T & \delta_1 \end{pmatrix} \tag{15}$$

Which yields the update rule in the above algorithm.

4 UDU^T Factorization for tridiagonal matrices

```
function [ A_out ] = UDU_TRI_unb( A )

    % here we leave out the partion and repartion code in the notes

    %-------------------------------------------------------------%

    alpha33 = alpha33 % = delta11 (no-op)
    u23 = alpha23 / alpha33; % scalar
    alpha22 = alpha22 - u23 * alpha23;
    alpha23 = u23;

    %-------------------------------------------------------------%

    % here we leave out the continue code in the notes

return
```

Note that we derive the algorithm by simply setting $l01$ and $a01$ to be a vector with single non-zero elements and then using the scalar for the update rule.

5 Twisted Factorization: ϕ_1

For LDL^T factorization, we have

$$\begin{aligned}
A &= LDL^T \\
&= \begin{pmatrix} L_{00} & 0 & 0 \\ \lambda_{10}e_L^T & 1 & 0 \\ 0 & \lambda_{21}e_F & L_{22} \end{pmatrix} \begin{pmatrix} D_{00} & 0 & 0 \\ 0 & \delta_1 & 0 \\ 0 & 0 & D_{22} \end{pmatrix} \begin{pmatrix} L_{00} & 0 & 0 \\ \lambda_{10}e_L^T & 1 & 0 \\ 0 & \lambda_{21}e_F & L_{22} \end{pmatrix}^T \\
&= \begin{pmatrix} L_{00}D_{00}L_{00}^T & \lambda_{10}L_{00}D_{00}e_L & 0 \\ \lambda_{10}e_L^TD_{00}L_{00}^T & \lambda_{10}^2e_L^TD_{00}e_L + \delta_1 & \lambda_{21}\delta_1e_F^T \\ 0 & \lambda_{21}\delta_1e_F & \lambda_{21}^2\delta_1e_Fe_F^T + L_{22}D_{22}L_{22}^T \end{pmatrix} && (16) \\
&= \begin{pmatrix} A_{00} & \alpha_{10}e_L & 0 \\ \alpha_{10}e_L^T & \alpha_{11} & \alpha_{21}e_F^T \\ 0 & \alpha_{21}e_F & A_{22} \end{pmatrix} && (17)
\end{aligned}$$

And by matching, we have

$$\begin{aligned}
A_{00} &= L_{00}D_{00}L_{00}^T \\
\alpha_{10}e_L &= \lambda_{10}L_{00}D_{00}e_L \\
\alpha_{11} &= \lambda_{10}^2e_L^TD_{00}e_L + \delta_1 && (18) \\
\alpha_{21} &= \lambda_{21}\delta_1 \\
A_{22} &= \lambda_{21}^2\delta_1e_Fe_F^T + L_{22}D_{22}L_{22}^T
\end{aligned}$$

Similarly, for UEU^T factorization, we have

$$\begin{aligned}
A &= UEU^T \\
&= \begin{pmatrix} U_{00} & v_{01}e_L & 0 \\ 0 & 1 & v_{21}e_F^T \\ 0 & 0 & U_{22} \end{pmatrix} \begin{pmatrix} E_{00} & 0 & 0 \\ 0 & \epsilon_1 & 0 \\ 0 & 0 & E_{22} \end{pmatrix} \begin{pmatrix} U_{00} & v_{01}e_L & 0 \\ 0 & 1 & v_{21}e_F^T \\ 0 & 0 & U_{22} \end{pmatrix}^T \\
&= \begin{pmatrix} U_{00}E_{00}U_{00}^T + v_{01}\epsilon_1e_Le_L^T & v_{01}\epsilon_1e_L & 0 \\ v_{01}\epsilon_1e_L^T & v_{21}^2e_F^TE_{22}e_F + \epsilon_1 & v_{21}e_F^TE_{22}U_{22}^T \\ 0 & v_{21}U_{22}E_{22}e_F & U_{22}E_{22}U_{22}^T \end{pmatrix} && (19) \\
&= \begin{pmatrix} A_{00} & \alpha_{10}e_L & 0 \\ \alpha_{10}e_L^T & \alpha_{11} & \alpha_{21}e_F^T \\ 0 & \alpha_{21}e_F & A_{22} \end{pmatrix} && (20)
\end{aligned}$$

And by matching, we have

$$\begin{aligned}
A_{00} &= U_{00}E_{00}U_{00}^T + v_{01}\epsilon_1e_Le_L^T \\
\alpha_{10} &= v_{01}\epsilon_1 \\
\alpha_{11} &= v_{21}^2e_F^TE_{22}e_F + \epsilon_1 && (21) \\
\alpha_{21}e_F^T &= v_{21}e_F^TE_{22}U_{22}^T \\
A_{22} &= U_{22}E_{22}U_{22}^T
\end{aligned}$$

Now we consider the Twisted Factorization

$$\begin{aligned}
&\begin{pmatrix} L_{00} & 0 & 0 \\ \lambda_{10}e_L^T & 1 & v_{21}e_F^T \\ 0 & 0 & U_{22} \end{pmatrix} \begin{pmatrix} D_{00} & 0 & 0 \\ 0 & \phi_1 & 0 \\ 0 & 0 & D_{22} \end{pmatrix} \begin{pmatrix} L_{00} & 0 & 0 \\ \lambda_{10}e_L^T & 1 & v_{21}e_F^T \\ 0 & 0 & U_{22} \end{pmatrix}^T && (22) \\
&= \begin{pmatrix} L_{00}D_{00}L_{00}^T & \lambda_{10}L_{00}D_{00}e_L & 0 \\ \lambda_{10}e_L^TD_{00}L_{00}^T & \phi_1 + \lambda_{10}^2e_L^TD_{00}e_L + v_{21}^2e_F^TE_{22}e_F & v_{21}e_F^TE_{22}U_{22}^T \\ 0 & v_{21}U_{22}E_{22}e_F & U_{22}E_{22}U_{22}^T \end{pmatrix} && (23) \\
&= \begin{pmatrix} A_{00} & \alpha_{10}e_L & 0 \\ \alpha_{10}e_L^T & \alpha_{11} & \alpha_{21}e_F^T \\ 0 & \alpha_{21}e_F & A_{22} \end{pmatrix} && (24)
\end{aligned}$$

Then we have

$$\alpha_{11} = \phi_1 + \lambda_{10}^2 e_L^T D_{00} e_L + \upsilon_{21}^2 e_F^T E_{22} e_F \tag{25}$$

To satisfy (18), (21) and (25) at the same time, we need to have

$$\phi_1 = \frac{\delta_1 + \epsilon_1 - \lambda_{10}^2 e_L^T D_{00} e_L - \upsilon_{21}^2 e_F^T E_{22} e_F}{2} \tag{26}$$

Complexity:

- computation of $e_L^T D_{00} e_L$ or $e_F^T E_{22} e_F$ is $\mathcal{O}(1)$. (constant time)
- computation of the factorized matrix, it requires $\mathcal{O}(n)$ for assembling components of U and L so as to derive the resulted matrix.

6 Twisted Factorization: Eigenvector

Separate terms of the known condition as follows:

$$\underbrace{\begin{pmatrix} L_{00} & 0 & 0 \\ \lambda_{10}e_L^T & 1 & v_{21}e_F^T \\ 0 & 0 & U_{22} \end{pmatrix} \begin{pmatrix} D_{00} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & E_{22} \end{pmatrix}}_{S} \underbrace{\begin{pmatrix} L_{00} & 0 & 0 \\ \lambda_{10}e_L^T & 1 & v_{21}e_F^T \\ 0 & 0 & U_{22} \end{pmatrix}^T \begin{pmatrix} x_0 \\ \chi_1 \\ x_2 \end{pmatrix}}_{y} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \tag{27}$$

Then we derive the form of S

$$S \triangleq \begin{pmatrix} L_{00} & 0 & 0 \\ \lambda_{10}e_L^T & 1 & v_{21}e_F^T \\ 0 & 0 & U_{22} \end{pmatrix} \cdot \begin{pmatrix} D_{00} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & E_{22} \end{pmatrix} = \begin{pmatrix} L_{00}D_{00} & 0 & 0 \\ \lambda_{10}e_L^T D_{00} & 0 & v_{21}e_F^T E_{22} \\ 0 & 0 & U_{22}E_{22} \end{pmatrix} \tag{28}$$

Then relate it to y

$$S \cdot y = \begin{pmatrix} L_{00}D_{00} & 0 & 0 \\ \lambda_{10}e_L^T D_{00} & 0 & v_{21}e_F^T E_{22} \\ 0 & 0 & U_{22}E_{22} \end{pmatrix} \begin{pmatrix} y_0 \\ \psi_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \tag{29}$$

where

$$\begin{aligned} L_{00}^T x_0 + \lambda_{10}e_L\chi_1 &= y_0 \\ \chi_1 &= \psi_1 \\ v_{21}\chi_1 e_F + U_{22}^T x_2 &= y_2 \end{aligned} \tag{30}$$

Solve (29), we have

$$\begin{aligned} y_0 &= 0 \\ \psi_1 &= c \text{ (constant)} \\ y_2 &= 0 \end{aligned} \tag{31}$$

In terms of (30), for the vector x, we need to solve the following system

$$\begin{aligned} L_{00}^T x_0 + \lambda_{10}e_L\chi_1 &= 0 \\ \chi_1 &= c \\ v_{21}\chi_1 e_F + U_{22}^T x_2 &= 0 \end{aligned} \tag{32}$$

Note that this equation system has infinity number of solutions unless we set c fixed. Here, we set $c = 1$ for simplicity. Then

$$L_{00}^T x_0 = -\lambda_{10}e_L \tag{33}$$

$$U_{22}^T x_2 = -v_{21}e_F \tag{34}$$

which is actually two gaussian elimination problem. In terms of the special structure of L_{00} and U_{22}, the solution takes complexity $\mathcal{O}(n^2)$.