1. **[Properties of LU factorization, 1+2pt]** We study basic properties of the LU-factorization.

   (a) Give an example of an invertible 3×3 matrix that does not have any zero entries, for which the LU decomposition without pivoting fails.

(a) Example:

Let $A = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 1 & 3 & 4 \end{bmatrix}$ We claim A is a Matrix desired.

Suppose

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 1 & 3 & 4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{bmatrix} \begin{bmatrix} U_{11} & U_{12} & U_{13} \\ 0 & U_{22} & U_{23} \\ 0 & 0 & U_{33} \end{bmatrix}$$

Process the Computation.

$U_{11} = 1$

$U_{11} \cdot l_{21} = 2 \Rightarrow l_{21} = 2$

$U_{11} \cdot l_{31} = 1 \Rightarrow l_{31} = 1$

$U_{12} = 2$

$l_{21} \cdot U_{12} + U_{22} = 4 \Rightarrow U_{22} = 0$ which cause a problem

$U_{13} = 3$

$U_{13} \cdot l_{21} + U_{23} = 4$

$\Rightarrow U_{23} = -2$

$U_{12} \cdot l_{31} + U_{22} \cdot l_{32} = 3$

$2 + 0 = 3$ which is not possible

Therefore. A is not LU decomposible.

(b) Show that the LU factorization of an invertible matrix $A \in \mathbb{R}^{n\times n}$ is unique. That is, if

$$A = LU = L_1U_1$$

with upper triangular matrices U, U_1 and unit lower triangular matrices L, L_1, then necessarily $L = L_1$ and $U = U_1$. You can use the results we discussed in class about products of lower/upper triangular matrices, and their inverses.

proof.

If $LU = L_1U_1$, As A is invertible, which induce L, U, L_1, U_1 invertible.

(as otherwise $\det(A) = \det(L_1)\det(U) = \det(L)\det(U)$ would be 0)

then $L_1^{-1}L = U_1 \cdot U^{-1}$

We claim $L_1^{-1}L = U_1U^{-1} = I$

As: 1° L_1^{-1} is a unit lower matrix as it is an inverse of L_1, which is a lower unit matrix.

2° $L_1^{-1} \cdot L$ is a unit lower matrix as it is a product of 2 lower unit matrix.

Similarly. U_1U^{-1} is upper triangle as U^{-1} is upper as U is, and then the property of product makes U_1U^{-1} upper triangle.

However, as $L_1^{-1} \cdot L = U_1U^{-1}$, where LHS is lower $\triangle$ and RHS is Upper $\triangle$

$L_1^{-1} \cdot L = U_1 \cdot U^{-1}$ is diagonal. Further more, as $L_1^{-1} \cdot L$ is unit.

we have $L_1^{-1} \cdot L = U_1 \cdot U^{-1} = I$

As $L_1^{-1} \cdot L = I = U_1 \cdot U^{-1}$

We have $L_1 = L$ and $U_1 = U$, where Uniqueness follows.

2. **[LU for transposed matrix, 2+2pt]** Let $n \geq 2$. Consider a matrix $A \in \mathbb{R}^{n \times n}$ for which every leading principal submatrix of order less than n is non-singular.

   (a) Show that A can be factored in the form $A = LDU$, where $L \in \mathbb{R}^{n \times n}$ is unit lower triangular, $D \in \mathbb{R}^{n \times n}$ is diagonal and $U \in \mathbb{R}^{n \times n}$ is *unit* upper triangular.

proof.

Given all leading principal Sub matrices of order $<n$ is invertible

We have $A = LU'$

Here U' is not necessary unit, however, by the nature of factorization. $U'_{jj} \neq 0$ unless $j = n$

It suffices to show that U' admits a decomposition S.t. $U' = DU$ where D diagonal. U unit Upper.

We Construct it in this way

Consider U^T, which is a Lower Triangle

Apply LU decomposition on U^T, as all leading Sub matrix of A is invertible. only the last entry of U^T can be 0. $\Rightarrow$ LU decomposible.

$$U^T = L_2 U_2$$

$$\Rightarrow U = (L_2 U_2)^T = U_2^T \cdot L_2^T$$

Notice that L_2^T here is unit upper triangle.

Here we prove that U_2^T is diagonal.

As L_2^T is unit, it is invertible.

$\Rightarrow U_2^T = U \cdot (L_2^T)^{-1}$ which is a product of 2 upper matrix

$\Rightarrow U_2^T$ is upper

but it is also lower as U_2 is upper.

$\Rightarrow U_2^T$ is diagonal.

Therefore, we have. $A = L' \cdot U_2^T \cdot L_2^T$

where. $L \quad D \quad U$

(b) If the factorization $A = LU$ is known, where L is unit lower triangular and U is upper triangular, show how to find the LU-factors of the transpose A^T. Note that our requirement for an LU-factorization is that L is *unit* lower triangular, and U is upper triangular.

proof.

As $A = LU$ is known.

Then we may use A to get

$$A = L D U'$$

$$\Rightarrow A^T = U'^T D L^T$$

$$\Rightarrow A^T = U'^T (D L^T)$$

here U'^T is lower unit triangle

and DL^T is upper triangle.

3. **[Cholesky factorization, 3+2pt]**

(a) A matrix A is called symmetric if $A = A^T$. If we compute $A = LDU$ for a symmetric matrix A, and find that each diagonal entry of $D > 0$, argue that A can be factored as $A = RR^T$ and find R in terms of L, D, U. Note that the factorization $A = RR^T$ can only be done for some types of symmetric matrices (specifically symmetric-positive-definite (spd) matrices), and is referred to as a Cholesky factorization.

proof.

First, we claim A is invertible. Indeed. as $A = LDU$

$\det(A) = \det(L)\det(D)\det(U)$

As L and U are both unit triangular matrix. $\det(U) > 0$ $\det(L) > 0$.

Also. as D has diagonal entry > 0. $\det(D) > 0$

$\Rightarrow \det(A) \neq 0$ i.e. A invertible.

As $A = LDU$,

$$A^T = U^T D^T L^T = U^T D L^T = LDU = A$$

Here we can write $A = L(DU)$

and $A = U^T(DL^T)$

Notice. L and U^T are lower unit Triangle.

DU and DL^T are Upper Triangle.

Given the conclusion of Question 1. the LU decomposition of A invertible is unique.

We have: $\begin{cases} L = U^T \\ DL^T = DU \end{cases} \Rightarrow L = U^T$

(Note that as D invertible. the second equation deduce $L = U^T$ as well)

Therefore.

$$A = L D L^T$$

As the entry of D is positive.

We can construct $E = \text{diag}(\sqrt{D_{ii}})$ s.t $E^2 = D$

$$A = LE \cdot E L^T$$

Therefore. $A = (LE)\cdot(LE)^T$

where $E = \text{diag}(\sqrt{D_{ii}})$

(b) The algorithm to find the Cholesky factorization of an $n \times n$ matrix A is as follows:
for $j = 1, \ldots, n$

i. $r_{jj} = \sqrt{a_{jj} - \sum_{k=1}^{j-1} r_{jk}^2}$

ii. for $i > j$: $r_{ij} = \frac{1}{r_{jj}} \left(a_{ij} - \sum_{k=1}^{j-1} r_{ik} r_{jk} \right)$

Use this to compute the Cholesky factorization of

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

To avoid computing the square root of a negative number (which happens for matrices that are not spd), check at each step that the quantity under the square root in the computation of r_{jj} is positive. If it is not, display an error message[1] and stop the code. Please hand in your commented code.

See Code page below

4. **[Backward substituition implementation, 5pt]** Write a code for backward substitution to solve systems of the form $U\boldsymbol{x} = \boldsymbol{b}$, i.e., write a function `x = backward(A,b)`, which expects as inputs an upper triangular matrix $U \in \mathbb{R}^{n\times n}$, and a right hand side vector $\boldsymbol{b} \in \mathbb{R}^n$, which returns the solution vector $\boldsymbol{x} \in \mathbb{R}^n$. The function should find the size n from the vector $\boldsymbol{b}$ and also check if the matrix and the vector sizes are compatible before it starts to solve the system. Please hand in your commented code. Apply your program for the computation of for $\boldsymbol{x} \in \mathbb{R}^4$, with

$$U = \begin{bmatrix} 1 & 2 & 6 & -1 \\ 0 & 3 & 1 & 0 \\ 0 & 0 & 4 & -1 \\ 0 & 0 & 0 & 2 \end{bmatrix}, \quad \boldsymbol{b} = \begin{bmatrix} -1 \\ -3 \\ -2 \\ 4 \end{bmatrix}.$$

See Code page below

HW_2_code

February 27, 2022

[1]:
```
import numpy as np
```

0.1 Question 3b

[2]:
```
def choleskyfact(A):
    """
    Input: A: n*n matrix, 2d array
    Output: R n*n matrix, 2d array
    """

    # Check if the input is valid

    if A.shape[0] != A.shape[1]:
        raise ValueError("Input should be square.")

    if not np.allclose(A, A.T):
        raise ValueError("Input should be symmetric.")


    # Initialize the output R matrix

    n = A.shape[0]
    R = np.zeros((n, n))

    # Implement the iteration accordingly as the given instruction

    for j in range(n):
        temp = A[j][j] - sum([ (R[j][k])**2 for k in range(0, j) ])
        if temp <= 0:
            raise ValueError("The quantity under the square root is negative")

        R[j][j] = temp**(1/2)

        for i in range(j+1, n):
            R[i][j] = (1/R[j][j])*(A[i][j] - sum([ R[i][k]*R[j][k] for k in␣
↪range(0, j)]))
```

```
    return R
```

[3]:
```
# Apply the code to the matrix given

A = np.array([
    [2, 1, 1/2, 1/4],
    [1, 4, 1, 1/2],
    [1/2, 1, 4, 1],
    [1/4, 1/2, 1, 2]
])

print("The R solved by the self-defined function is: \n", choleskyfact(A))
print("\nCompared with the solution generated by numpy method: \n", np.linalg.
 ↪cholesky(A))
```

```
The R solved by the self-defined function is:
 [[1.41421356 0.         0.         0.        ]
 [0.70710678 1.87082869 0.         0.        ]
 [0.35355339 0.40089186 1.92724822 0.        ]
 [0.1767767  0.20044593 0.44474959 1.31558703]]

Compared with the solution generated by numpy method:
 [[1.41421356 0.         0.         0.        ]
 [0.70710678 1.87082869 0.         0.        ]
 [0.35355339 0.40089186 1.92724822 0.        ]
 [0.1767767  0.20044593 0.44474959 1.31558703]]
```

0.2 Question 4

[4]:
```
def backward(U, b):
    """
    Input: U, n*n upper matrix, 2Darray, b: n*1 array
    Output: x, n*1 array
    """
    b = b.squeeze() # in case input is an column vactor

    # Check if the input is valid

    if U.shape[0] != U.shape[1]:
        raise ValueError("U should be square.")

    if not np.allclose(U, np.triu(U)):
        raise ValueError("U should be upper triangular.")

    if b.shape[0] != U.shape[0]:
```

```
        raise ValueError("Input dimension not compatible.")

    n = U.shape[0]

    # Initialize the output array
    x = np.zeros(n)

    # Implement the iteration according to the instructions given
    for i in range(1, n+1):
        x[-i] = (1 / U[-i][-i])*(b[-i] - sum([U[-i][-j]*x[-j] for j in range(1,
↪i)]))

    return x
```

[5]:
```
# Apply the program on the matrix given.

U = np.array([
    [1, 2, 6, -1],
    [0, 3, 1, 0],
    [0, 0, 4, -1],
    [0, 0, 0, 2],
])

b = np.array([-1, -3, -2, 4])

print("The result is: \n", backward(U, b))
print("\nCompare with the result from taking inverse: \n", np.linalg.inv(U)@b.T)
```

```
The result is:
 [ 3. -1.  0.  2.]

Compare with the result from taking inverse:
 [ 3. -1.  0.  2.]
```

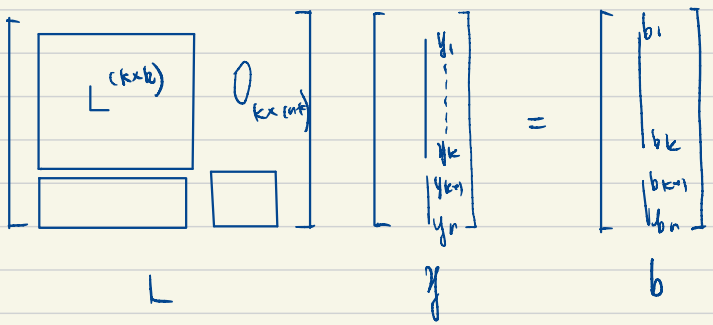
5. **[Right hand sides with many zeros, 4pt]** For a given dimension n, fix some k with $1 \leq k \leq n$. Now let $L \in \mathbb{R}^{n \times n}$ be a non-singular lower triangular matrix and let the vector $\boldsymbol{b} \in \mathbb{R}^n$ be such that $b_i = 0$ for $i = 1, 2, \ldots, k$.

   (a) Let the vector $\boldsymbol{y} \in \mathbb{R}^n$ be the solution of $L\boldsymbol{y} = \boldsymbol{b}$. Show, by partitioning L into blocks, that $y_j = 0$ for $j = 1, 2, \ldots, k$.

   (b) Use this to give an alternative proof of Theorem 2.1 (iv), i.e., that the inverse of a non-singular lower triangular matrix is itself lower triangular.

(a) $Ly = b$

Partition L and correspondingly y and b into the following way.



Here we can see

$$\begin{bmatrix} b_1 \\ \vdots \\ b_k \end{bmatrix} = L^{k\times k} \begin{bmatrix} y_1 \\ \vdots \\ y_k \end{bmatrix} + 0_{k\times(n-k)} \cdot \begin{bmatrix} y_k \\ y_n \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} b_1 \\ \vdots \\ b_k \end{bmatrix} = \begin{bmatrix} | \\ 0 \\ | \end{bmatrix} = L^{k\times k} \begin{bmatrix} y_1 \\ \vdots \\ y_k \end{bmatrix}$$

$\Rightarrow$ $L^{k\times k}$ is invertible

as L is invertible.

(as non singularity of L guarantees nonzero diagonal)

$\Rightarrow$ $L^{k\times k}$ has trivial kernel.

As $b_{i<k} = 0 \Rightarrow y_{i<0} \quad \forall i \in \{1, \ldots k\}$

(b)

Let $I_{n\times n} = \begin{bmatrix} | & & | \\ e_1 & \cdots & e_n \\ | & & | \end{bmatrix}$ where $e_i = \begin{bmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 1 \end{bmatrix}$ ← ith entry

Let L be an invertible lower triangle matrix.

$$L \cdot L^{-1} = I$$

$$L \begin{bmatrix} | & & | \\ l_1 & \cdots & l_n \\ | & & | \end{bmatrix} = \begin{bmatrix} | & & | \\ e_1 & \cdots & e_n \\ | & & | \end{bmatrix}$$

here we can see $L\, l_i = e_i$.

as e_i has 0 entry from 1 to i-1.

its solution, by (a) has 0 entry from 1 to i-1

As $L^{-1} = \begin{bmatrix} | & & | \\ l_1 & \cdots & l_n \\ | & & | \end{bmatrix}$

and for each l_i, it has 0 entry from 1 to i-1

it follows L^{-1} is lower triangular as well.

6. **[Inverse matrix computation, 1+1+2pt]** Let us use the LU-decomposition to compute the inverse of a matrix[2].

   (a) Describe the algorithm (we discussed it in class) that uses the LU-decomposition of an $n \times n$ matrix A for computing A^{-1} by solving n systems of equations (one for each unit vector).

   (b) Give the floating point operation count of this algorithm.

   (c) Improve the algorithm by taking advantage of the structure (i.e., the zero entries—see previous problem) of the right-hand side. What is the new algorithm's floating point operation count?

(a) 1° Solve L-U decomposition of A.

2° Solve $A y_i = \begin{bmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix}$ ← ith entry for $i \in \{1, \cdots, n\}$

in the following way: 1° Solve $L b_i = \begin{bmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix}$ ← ith entry

2° Solve $U y_i = b_i$

3° Put y_i together into a matrix. namely: $A^{-1} = [y_1 \ \cdots \ y_n]$

(b) FLOP: Solving LU-Decompose.: $\frac{2}{3}n^3$

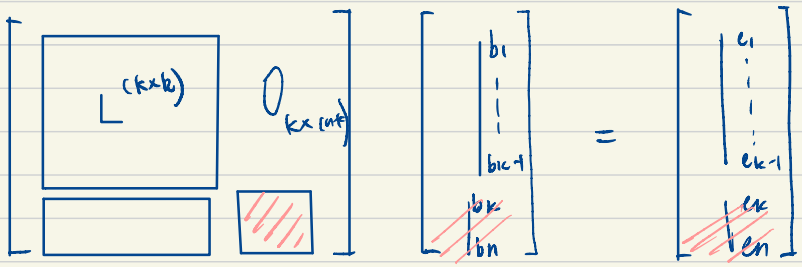
Solving y_i n times: $2 \times n^2 \times n = 2n^3$ } $\frac{2}{3}n^3 + 2n^3$

(c) LU decomposition cannot be simplified. ($\frac{2}{3}n^3$)

Since know L is lower triangle and $\begin{bmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix}$ i-th colum thus $0, \cdots, i$ entry 0.

When Solving $Lb_i = e_i$ we only need to calculate the lower part (from i+1 to n)

Namely



L bi ei

only the ▨ part.

Therefore the flop for backward sub for $L b_i = e_i$ $i \in \{1, \cdots n\}$ has only $\sum_{i=1}^{n-1} i^2$

The process of $Uy = b$ cannot be simplified

So we reduce the FLOP to: $\sum_{i=1}^{n-1} i^2 + n^3 + \frac{2}{3}n^3 = 2n^3$

7. **[Stability of the Gaussian elimination, 2+1+2+1pt]**

Consider the system

$$Ax = b, \tag{1}$$

3 of 3

where $A, L, E \in \mathbb{R}^{n\times n}$, $\boldsymbol{x}, \boldsymbol{b} \in \mathbb{R}^n$, with $A = L + E$. L is unit lower triangular, where all the subdiagonal elements are -1, i.e., $l_{i,j} = -1$ for $i < j$, $l_{i,i} = 1$ for $i = 1, \ldots, n$, and $l_{i,j} = 0$ otherwise. Additionally, E is a matrix such that $e_{i,n} = 1$ for $i = 1, \ldots, n-1$ and $e_{i,j} = 0$ otherwise. For example, when $n = 5$, we have

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 \\ -1 & 1 & 0 & 0 & 1 \\ -1 & -1 & 1 & 0 & 1 \\ -1 & -1 & -1 & 1 & 1 \\ -1 & -1 & -1 & -1 & 1 \end{pmatrix}.$$

(a) Prove that A is invertible for any n, by induction.

(a) Proof by induction.

(Base Step) $n=2$

$A = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}$ $\det(A) = 2 \neq 0$

$\Rightarrow$ A invertible.

(Inductive Step)

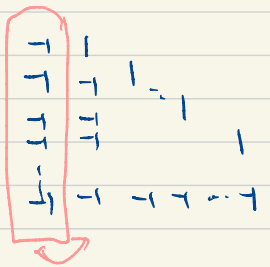
Assume for $n-1$ and prove for n

$A = \begin{bmatrix} 1 & 0 & \cdots & 0 & 1 \\ -1 & & & & \\ \vdots & & \text{(n-1)} & & \\ -1 & & & & \end{bmatrix}$

Note that to calculate the determinant of A. There're only 2 nonzero entry on the first row.

For the □, the determinant is nonzero as is assumed.

For the ▭ part. we need to permute the column to let it become the $(n-1)\times(n-1)$ A like matrix



we need to put the most left column to the right most.
So we need to switch $(n-2)$ times.
After doing so. we need to multiply (-1) to the last column.

Therefore. $\det(\square) = (-1)^{n-2}\cdot(-1)\cdot\det(\square)$ → the $(n-1)\times(n-1)$ A-like matrix.
denoted $A^{(n-1)\times(n-1)}$

$\Rightarrow \det(A) = \det(A^{(n-1)\times(n-1)}) + (-1)^{(n-1)}\cdot(-1)^{n-1}\cdot\det(A^{(n-1)\times(n-1)})$

$= 2\cdot\det(A^{(n-1)\times(n-1)})$

As is given in the assumption. $A^{(n-1)\times(n-1)}$ is invertible $\Leftrightarrow$ nonzero determinant.

$\Rightarrow \det(A)$ is nonzero $\Leftrightarrow$ invertible.

According to the (Base step) and (Inductive Step).

we have A is invertible $\forall n \in \mathbb{N}$.

(b) When $n = 5$ and $\boldsymbol{b} = (1, \frac{1}{4}, \frac{1}{9}, \frac{1}{16}, \frac{1}{25})^T$, solve for $\boldsymbol{x}$ in (1) using Gaussian elimination, then backward substitution.

$$\left[\begin{array}{ccccc|c} 1 & 0 & 0 & 0 & 1 & 1 \\ -1 & 1 & 0 & 0 & 1 & 1/4 \\ -1 & -1 & 1 & 0 & 1 & 1/9 \\ -1 & -1 & -1 & 1 & 1 & 1/16 \\ -1 & -1 & -1 & -1 & 1 & 1/25 \end{array}\right] \begin{array}{l} \\ +R_1 \\ +R_1 \\ +R_1 \\ +R_1 \end{array} \quad \left[\begin{array}{ccccc|c} 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 2 & 1+1/4 \\ 0 & -1 & 1 & 0 & 2 & 1+1/9 \\ 0 & -1 & -1 & 1 & 2 & 1+1/16 \\ 0 & -1 & -1 & -1 & 2 & 1+1/25 \end{array}\right] \begin{array}{l} \\ \\ +R_2 \\ +R_2 \\ +R_2 \end{array}$$

$$\left[\begin{array}{ccccc|c} 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 2 & 1+1/4 \\ 0 & 0 & 1 & 0 & 4 & 1+1/9+1+\frac{1}{4} \\ 0 & 0 & -1 & 1 & 4 & 1+1/16+1+\frac{1}{4} \\ 0 & 0 & -1 & -1 & 4 & 1+1/25+1+\frac{1}{4} \end{array}\right] \begin{array}{l} \\ \\ \\ +R_3 \\ +R_3 \end{array}$$

$$\left[\begin{array}{ccccc|c} 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 2 & 1+1/4 \\ 0 & 0 & 1 & 0 & 4 & 1+1/9+1+\frac{1}{4} \\ 0 & 0 & 0 & 1 & 8 & 1+1/16+1+\frac{1}{4}+1+1/9+1+1/4 \\ 0 & 0 & -1 & -1 & 8 & 1+1/25+1+\frac{1}{4}+1+1/9+1+1/4 \end{array}\right] +R_4$$

$$\left[\begin{array}{ccccc|c} 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 2 & 1+1/4 \\ 0 & 0 & 1 & 0 & 4 & 1+1/9+1+\frac{1}{4} \\ 0 & 0 & 0 & 1 & 8 & 1+1/16+1+\frac{1}{4}+1+1/9+1+1/4 \\ 0 & 0 & 0 & 0 & 16 & 1+1/25+1+\frac{1}{4}+1+1/9+1+1/4+1+1/16+1+\frac{1}{4}+1+1/9+1+1/4 \end{array}\right]$$

Do back prop.

$b_5 = \frac{1}{16} \times \frac{33509}{3600} = \frac{33509}{57600}$

$b_4 = \frac{673}{144} - 8b_5 = \frac{9}{800}$

$b_3 = \frac{85}{36} - 4b_5 = \frac{431}{14400}$

$b_2 = \frac{5}{4} - 2b_5 = \frac{2431}{28800}$

$b_1 = 1 - b_5 = \frac{24031}{57600}$

(c) Now consider A, $\boldsymbol{x}$, $\boldsymbol{b}$, where n is not specified. Put (1) into the form $U\boldsymbol{x} = \boldsymbol{c}$, where U is upper triangular using Gaussian elimination (you do not have write what $\boldsymbol{c}$ is since $\boldsymbol{b}$ is not given). What is $\max_{i,j} |u_{i,j}|$?

According to the Gauss Elimination Process above.

for i th Step. we need to add i^{th} row to k^{th} row

$k \in \{i+1, \dots n\}$

Therefore. for the Last column

$$u_{in} = 1 + 1 + 2 + 4 + \cdots 2^{i-2}$$
$$= 2^{i-1}$$

As other entry is either 0 or 1

We have $\max_{i,j} [u_{i,j}] = u_{n,n} = 2^{n-1}$

(d) For large n, e.g., $n = 2000$, what problems can you envision if you try to solve (1) using Gaussian elimination on a computer? Explain.

1° Since the largest number of U is 2^{n-1} for n×n matrix.

this number is really hard to store in Computer as it is too large. Even if it can be stored it will waste a great memory as other number are small.

2° The Condition Number of the eliminated matrix tends to be big. The matrix is highly unstable.