

DM587  
Scientific Programming

## Numerical Methods LU Factorization

Marco Chiarandini

Department of Mathematics & Computer Science  
University of Southern Denmark

*[Based on slides by Lieven Vandenberghe, UCLA]*

# Outline

Operation Count  
LU Factorization  
Other Topics

1. Operation Count

2. LU Factorization

3. Other Topics

In solving large scale-linear systems, Gaussian elimination and Gauss-Jordan elimination are not suitable because of:

- computer roundoff errors
- memory usage
- speed

Computer methods are based on LU decomposition.

# Outline

Operation Count  
LU Factorization  
Other Topics

1. Operation Count

2. LU Factorization

3. Other Topics

# Complexity of matrix algorithms

- flop counts
- vector-vector operations
- matrix-vector product
- matrix-matrix product

# Floating point numbers

$$x = m \cdot \beta^e ; \quad l \leq e \leq u$$

with mantissa  $m$ , base  $\beta$ , and exponent  $e$

$$m = \pm d_0.d_1d_2 \cdots d_t , \quad 0 \leq d_i < \beta$$

	$\beta$	$t$	$l$	$u$
IEEE SP	2	23	-126	127
IEEE DP	2	52	-1022	1023
Cray	2	48	-16383	16384
HP calculator	10	12	-499	499
IBM mainframe	16	6	-64	63

# Flop counts

## floating-point operation (flop)

- one floating-point addition, subtraction, multiplication, or division
- other common definition: one multiplication followed by one addition

## flop counts of matrix algorithm

- total number of flops is typically a polynomial of the problem dimensions
- usually simplified by ignoring lower-order terms

## applications

- a simple, machine-independent measure of algorithm complexity
- not an accurate predictor of computation time on modern computers

# Vector-vector operations

- inner product of two  $n$ -vectors

$$\mathbf{x}^T \mathbf{y} = x_1 y_1 + x_2 y_2 + \dots + x_n y_n$$

$n$  multiplications and  $n - 1$  additions =  $2n$  flops ( $2n$  if  $n \gg 1$ )

- addition or subtraction of  $n$ -vectors:  $n$  flops
- scalar multiplication of  $n$ -vector :  $n$  flops



# Matrix-vector product

matrix-vector product with  $m \times n$ -matrix  $A$ :

$$y = Ax$$

$m$  elements in  $y$ ; each element requires an inner product of length  $n$ :

$$(2n - 1)m \text{ flops}$$

approximately  $2mn$  for large  $n$  special cases

- $m = n$ ,  $A$  diagonal:  $n$  flops
- $m = n$ ,  $A$  lower triangular:  $n(n + 1)$  flops
- $A$  very sparse (lots of zero coefficients):  $\#flops \ll 2mn$

# Matrix-matrix product

product of  $m \times n$ -matrix  $A$  and  $n \times p$ -matrix  $B$ :

$$C = AB$$

$mp$  elements in  $C$ ; each element requires an inner product of length  $n$ :

$$mp(2n - 1) \text{ flops}$$

approximately  $2mnp$  for large  $n$ .

## Approximate Cost for an $n \times n$ Matrix $A$ with Large $n$

Algorithm	Cost in Flops
Gauss–Jordan elimination (forward phase)	$\approx \frac{2}{3}n^3$
Gauss–Jordan elimination (backward phase)	$\approx n^2$
$LU$ -decomposition of $A$	$\approx \frac{2}{3}n^3$
Forward substitution to solve $Lx = b$	$\approx n^2$

# Outline

Operation Count  
**LU Factorization**  
Other Topics

1. Operation Count

2. LU Factorization

3. Other Topics

- factor-solve method
- LU factorization
- solving  $Ax = b$  with  $A$  nonsingular
- the inverse of a nonsingular matrix
- LU factorization algorithm
- effect of rounding error
- sparse LU factorization

# Definitions

## Definition (Triangular Matrices)

An  $n \times n$  matrix is said to be **upper triangular** if  $a_{ij} = 0$  for  $i > j$  and **lower triangular** if  $a_{ij} = 0$  for  $i < j$ . Also  $A$  is said to be **triangular** if it is either upper triangular or lower triangular.

Example:

$$\begin{bmatrix} 3 & 0 & 0 \\ 2 & 1 & 0 \\ 1 & 4 & 3 \end{bmatrix} \quad \begin{bmatrix} 3 & 5 & 1 \\ 0 & 1 & 3 \\ 0 & 0 & 7 \end{bmatrix}$$

## Definition (Diagonal Matrices)

An  $n \times n$  matrix is **diagonal** if  $a_{ij} = 0$  whenever  $i \neq j$ .

Example:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 3 \end{bmatrix}$$

## Multiple right-hand sides

two equations with the same matrix but different right-hand sides

$$Ax = b, \quad A\tilde{x} = \tilde{b}$$

- factor  $A$  once ( $f$  flops)
- solve with right-hand side  $b$  ( $s$  flops)
- solve with right-hand side  $\tilde{b}$  ( $s$  flops)

cost:  $f + 2s$  instead of  $2(f + s)$  if we solve second equation from scratch

**conclusion:** if  $f \gg s$ , we can solve the two equations at the cost of one

# LU factorization

## LU factorization without pivoting

$$A = LU$$

- $L$  unit lower triangular,  $U$  upper triangular
- does not always exist (even if  $A$  is nonsingular)

## LU factorization (with row pivoting)

$$A = PLU$$

- $P$  permutation matrix,  $L$  unit lower triangular,  $U$  upper triangular
- exists if and only if  $A$  is nonsingular (see later)

**cost:**  $(2/3)n^3$  if  $A$  has order  $n$



## Solving linear equations by LU factorization

solve  $Ax = b$  with  $A$  nonsingular of order  $n$

**factor-solve method** using LU factorization

1. factor  $A$  as  $A = PLU$  ( $(2/3)n^3$  flops)
2. solve  $(PLU)x = b$  in three steps
  - permutation:  $z_1 = P^T b$  (0 flops)
  - forward substitution: solve  $Lz_2 = z_1$  ( $n^2$  flops)
  - back substitution: solve  $Ux = z_2$  ( $n^2$  flops)

**total cost:**  $(2/3)n^3 + 2n^2$  flops, or roughly  $(2/3)n^3$

this is the standard method for solving  $Ax = b$

## Multiple right-hand sides

two equations with the same matrix  $A$  (nonsingular and  $n \times n$ ):

$$Ax = b, \quad A\tilde{x} = \tilde{b}$$

- factor  $A$  once
- forward/back substitution to get  $x$
- forward/back substitution to get  $\tilde{x}$

cost:  $(2/3)n^3 + 4n^2$  or roughly  $(2/3)n^3$

**exercise:** propose an efficient method for solving

$$Ax = b, \quad A^T \tilde{x} = \tilde{b}$$

## Inverse of a nonsingular matrix

suppose  $A$  is nonsingular of order  $n$ , with LU factorization

$$A = PLU$$

- inverse from LU factorization

$$A^{-1} = (PLU)^{-1} = U^{-1}L^{-1}P^T$$

- gives interpretation of solve step: evaluate

$$x = A^{-1}b = U^{-1}L^{-1}P^Tb$$

in three steps

$$z_1 = P^Tb, \quad z_2 = L^{-1}z_1, \quad x = U^{-1}z_2$$

## Computing the inverse

solve  $AX = I$  by solving  $n$  equations

$$AX_1 = e_1, \quad AX_2 = e_2, \quad \dots, \quad AX_n = e_n$$

$X_i$  is the  $i$ th column of  $X$ ;  $e_i$  is the  $i$ th unit vector of size  $n$

- one LU factorization of  $A$ :  $2n^3/3$  flops
- $n$  solve steps:  $2n^3$  flops

**total:**  $(8/3)n^3$  flops

**conclusion:** do not solve  $Ax = b$  by multiplying  $A^{-1}$  with  $b$

## LU factorization without pivoting

partition  $A$ ,  $L$ ,  $U$  as block matrices:

$$A = \begin{bmatrix} a_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}, \quad L = \begin{bmatrix} 1 & 0 \\ L_{21} & L_{22} \end{bmatrix}, \quad U = \begin{bmatrix} u_{11} & U_{12} \\ 0 & U_{22} \end{bmatrix}$$

- $a_{11}$  and  $u_{11}$  are scalars
- $L_{22}$  unit lower-triangular,  $U_{22}$  upper triangular of order  $n - 1$

determine  $L$  and  $U$  from  $A = LU$ , *i.e.*,

$$\begin{aligned} \begin{bmatrix} a_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} &= \begin{bmatrix} 1 & 0 \\ L_{21} & L_{22} \end{bmatrix} \begin{bmatrix} u_{11} & U_{12} \\ 0 & U_{22} \end{bmatrix} \\ &= \begin{bmatrix} u_{11} & U_{12} \\ u_{11}L_{21} & L_{21}U_{12} + L_{22}U_{22} \end{bmatrix} \end{aligned}$$

**recursive algorithm:**

- determine first row of  $U$  and first column of  $L$

$$u_{11} = a_{11}, \quad U_{12} = A_{12}, \quad L_{21} = (1/a_{11})A_{21}$$

- factor the  $(n-1) \times (n-1)$ -matrix  $A_{22} - L_{21}U_{12}$  as

$$A_{22} - L_{21}U_{12} = L_{22}U_{22}$$

this is an LU factorization (without pivoting) of order  $n-1$

**cost:**  $(2/3)n^3$  flops (no proof)

## Example

LU factorization (without pivoting) of

$$A = \begin{bmatrix} 8 & 2 & 9 \\ 4 & 9 & 4 \\ 6 & 7 & 9 \end{bmatrix}$$

write as  $A = LU$  with  $L$  unit lower triangular,  $U$  upper triangular

$$A = \begin{bmatrix} 8 & 2 & 9 \\ 4 & 9 & 4 \\ 6 & 7 & 9 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix}$$

- first row of  $U$ , first column of  $L$ :

$$\begin{bmatrix} 8 & 2 & 9 \\ 4 & 9 & 4 \\ 6 & 7 & 9 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 1/2 & 1 & 0 \\ 3/4 & l_{32} & 1 \end{bmatrix} \begin{bmatrix} 8 & 2 & 9 \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix}$$

- second row of  $U$ , second column of  $L$ :

$$\begin{bmatrix} 9 & 4 \\ 7 & 9 \end{bmatrix} - \begin{bmatrix} 1/2 \\ 3/4 \end{bmatrix} \begin{bmatrix} 2 & 9 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ l_{32} & 1 \end{bmatrix} \begin{bmatrix} u_{22} & u_{23} \\ 0 & u_{33} \end{bmatrix}$$

$$\begin{bmatrix} 8 & -1/2 \\ 11/2 & 9/4 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 11/16 & 1 \end{bmatrix} \begin{bmatrix} 8 & -1/2 \\ 0 & u_{33} \end{bmatrix}$$

- third row of  $U$ :  $u_{33} = 9/4 + 11/32 = 83/32$

conclusion:

$$A = \begin{bmatrix} 8 & 2 & 9 \\ 4 & 9 & 4 \\ 6 & 7 & 9 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 1/2 & 1 & 0 \\ 3/4 & 11/16 & 1 \end{bmatrix} \begin{bmatrix} 8 & 2 & 9 \\ 0 & 8 & -1/2 \\ 0 & 0 & 83/32 \end{bmatrix}$$



**Not every nonsingular  $A$  can be factored as  $A = LU$**

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 2 \\ 0 & 1 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix}$$

- first row of  $U$ , first column of  $L$ :

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 2 \\ 0 & 1 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & l_{32} & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix}$$

- second row of  $U$ , second column of  $L$ :

$$\begin{bmatrix} 0 & 2 \\ 1 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ l_{32} & 1 \end{bmatrix} \begin{bmatrix} u_{22} & u_{23} \\ 0 & u_{33} \end{bmatrix}$$

$$u_{22} = 0, u_{23} = 2, l_{32} \cdot 0 = 1 ?$$

## LU factorization (with row pivoting)

if  $A$  is  $n \times n$  and nonsingular, then it can be factored as

$$A = PLU$$

$P$  is a permutation matrix,  $L$  is unit lower triangular,  $U$  is upper triangular

- not unique; there may be several possible choices for  $P$ ,  $L$ ,  $U$
- interpretation: permute the rows of  $A$  and factor  $P^T A$  as  $P^T A = LU$
- also known as *Gaussian elimination with partial pivoting* (GEPP)
- cost:  $(2/3)n^3$  flops

we will skip the details of calculating  $P$ ,  $L$ ,  $U$

## Example

$$\begin{bmatrix} 0 & 5 & 5 \\ 2 & 9 & 0 \\ 6 & 8 & 8 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 1/3 & 1 & 0 \\ 0 & 15/19 & 1 \end{bmatrix} \begin{bmatrix} 6 & 8 & 8 \\ 0 & 19/3 & -8/3 \\ 0 & 0 & 135/19 \end{bmatrix}$$

the factorization is not unique; the same matrix can be factored as

$$\begin{bmatrix} 0 & 5 & 5 \\ 2 & 9 & 0 \\ 6 & 8 & 8 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 3 & -19/5 & 1 \end{bmatrix} \begin{bmatrix} 2 & 9 & 0 \\ 0 & 5 & 5 \\ 0 & 0 & 27 \end{bmatrix}$$

## Effect of rounding error

$$\begin{bmatrix} 10^{-5} & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

exact solution:

$$x_1 = \frac{-1}{1 - 10^{-5}}, \quad x_2 = \frac{1}{1 - 10^{-5}}$$

let us solve the equations using LU factorization, rounding intermediate results to 4 significant decimal digits

we will do this for the two possible permutation matrices:

$$P = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad \text{or} \quad P = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

**first choice of  $P$ :**  $P = I$  (no pivoting)

$$\begin{bmatrix} 10^{-5} & 1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 10^5 & 1 \end{bmatrix} \begin{bmatrix} 10^{-5} & 1 \\ 0 & 1 - 10^5 \end{bmatrix}$$

$L, U$  rounded to 4 decimal significant digits

$$L = \begin{bmatrix} 1 & 0 \\ 10^5 & 1 \end{bmatrix}, \quad U = \begin{bmatrix} 10^{-5} & 1 \\ 0 & -10^5 \end{bmatrix}$$

forward substitution

$$\begin{bmatrix} 1 & 0 \\ 10^5 & 1 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \Longrightarrow \quad z_1 = 1, \quad z_2 = -10^5$$

back substitution

$$\begin{bmatrix} 10^{-5} & 1 \\ 0 & -10^5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ -10^5 \end{bmatrix} \quad \Longrightarrow \quad x_1 = 0, \quad x_2 = 1$$

error in  $x_1$  is 100%

**second choice of  $P$ :** interchange rows

$$\begin{bmatrix} 1 & 1 \\ 10^{-5} & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 10^{-5} & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 0 & 1 - 10^{-5} \end{bmatrix}$$

$L, U$  rounded to 4 decimal significant digits

$$L = \begin{bmatrix} 1 & 0 \\ 10^{-5} & 1 \end{bmatrix}, \quad U = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$$

forward substitution

$$\begin{bmatrix} 1 & 0 \\ 10^{-5} & 1 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad \Rightarrow \quad z_1 = 0, \quad z_2 = 1$$

backward substitution

$$\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad \Rightarrow \quad x_1 = -1, \quad x_2 = 1$$

error in  $x_1, x_2$  is about  $10^{-5}$

## conclusion:

- for some choices of  $P$ , small rounding errors in the algorithm cause very large errors in the solution
- this is called **numerical instability**: for the first choice of  $P$ , the algorithm is unstable; for the second choice of  $P$ , it is stable
- from numerical analysis: there is a simple rule for selecting a good (stable) permutation (we'll skip the details, since we skipped the details of the factorization algorithm)
- in the example, the second permutation is better because it permutes the largest element (in absolute value) of the first column of  $A$  to the 1,1-position

## Sparse linear equations

if  $A$  is sparse, it is usually factored as

$$A = P_1 L U P_2$$

$P_1$  and  $P_2$  are permutation matrices

- interpretation: permute rows and columns of  $A$  and factor  $\tilde{A} = P_1^T A P_2^T$

$$\tilde{A} = L U$$

- choice of  $P_1$  and  $P_2$  greatly affects the sparsity of  $L$  and  $U$ : many heuristic methods exist for selecting good permutations
- in practice: #flops  $\ll (2/3)n^3$ ; exact value is a complicated function of  $n$ , number of nonzero elements, sparsity pattern



## Conclusion

different levels of understanding how linear equation solvers work:

**highest level:**  $x = A \backslash b$  costs  $(2/3)n^3$ ; more efficient than  $x = \text{inv}(A)*b$

**intermediate level:** factorization step  $A = PLU$  followed by solve step

**lowest level:** details of factorization  $A = PLU$

- for most applications, level 1 is sufficient
- in some situations (*e.g.*, multiple right-hand sides) level 2 is useful
- level 3 is important only for experts who write numerical libraries

## Theorem

*If  $A$  is a square matrix that can be reduced to a row echelon form  $U$  by Gaussian elimination without row interchanges, then  $A$  can be factored as  $A = LU$ , where  $L$  is a lower triangular matrix.*

We know that the Gaussian elimination operations can be accomplished by multiplying  $A$  on the left by an appropriate sequence of elementary matrices; that is, there exist elementary matrices  $E_1, E_2, \dots, E_k$  such that

$$E_k \cdots E_2 E_1 A = U$$

where  $U$  is an upper triangular matrix. Since elementary matrices are invertible, we can solve for  $A$  as

$$A = E_1^{-1} E_2^{-1} \cdots E_k^{-1} U$$

$$A = LU$$

$$L = E_1^{-1} E_2^{-1} \cdots E_k^{-1}$$

$L$  is lower triangular because:

- multiplying a row by a nonzero constant, and adding a scalar multiple of one row to another

- If  $A$  is an invertible matrix that can be reduced to row echelon form without row interchanges, then  $A$  can be factored **uniquely** as

$$A = LDU$$

where  $L$  is a lower triangular matrix with 1's on the main diagonal,  $D$  is a diagonal matrix, and  $U$  is an upper triangular matrix with 1's on the main diagonal. This is called the **LDU-decomposition** (or **LDU-factorization**) of  $A$ .

“shift” the diagonal entries of  $U$  (or of  $L$ ) to a diagonal matrix  $D$  and write  $U$  as  $U = U'D$

- If desired, the diagonal matrix and the lower triangular matrix in the  $LU$ -decomposition can be multiplied to produce an  $LU$ -decomposition in which the 1's are on the main diagonal of  $U$  rather than  $L$ . (This is yet another example that LU decompositions are not unique)

Note that the columns of  $L'$  are obtained by dividing each entry in the corresponding column of  $L$  by the diagonal entry in the column. Thus, for example, we can rewrite  
 In general,  $LU$ -decompositions are not unique.

$$A = LU = \begin{bmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{bmatrix} \begin{bmatrix} 1 & u_{12} & u_{13} \\ 0 & 1 & u_{23} \\ 0 & 0 & 1 \end{bmatrix}$$

- In 1979 an important library of machine-independent linear algebra programs called LINPACK was developed at Argonne National Laboratories.
- Many of the programs in that library use the LU and other decomposition methods (SVD, Schur's decomposition, Cholesky decomposition, etc).
- Variations of the LINPACK routines in Fortran are used in many computer programs, including Scipy, MATLAB, Mathematica, and Maple.

# Outline

Operation Count  
LU Factorization  
**Other Topics**

1. Operation Count

2. LU Factorization

3. Other Topics

- A matrix  $A$  is said to be **ill conditioned** if relatively small changes in the entries of  $A$  can cause relatively large changes in the solutions of  $Ax = b$ .
- $A$  is said to be **well conditioned** if relatively small changes in the entries of  $A$  result in relatively small changes in the solutions of  $Ax = b$ .
- reaching RREF as in Gauss-Jordan requires more computation and more numerical instability hence disadvantageous.
- Gauss elimination is a **direct method**: the amount of operations can be specified in advance. **Indirect** or **Iterative methods** work by iteratively improving approximate solutions until a desired accuracy is reached. Amount of operations depend on the accuracy required. (way to go if the matrix is sparse)

# Gauss-Seidel Iterative Method

## Example

$$\begin{aligned}x_1 - 0.25x_2 - 0.25x_3 &= 50 \\ -0.25x_1 + x_2 - 0.25x_4 &= 50 \\ -0.25x_1 + x_3 - 0.25x_4 &= 25 \\ -0.25x_2 - 0.25x_3 + x_4 &= 25\end{aligned}$$

$$\begin{aligned}x_1 &= 0.25x_2 + 0.25x_3 + 50 \\ x_2 &= 0.25x_1 + 0.25x_4 + 50 \\ x_3 &= 0.25x_1 + 0.25x_4 + 25 \\ x_4 &= 0.25x_2 + 0.25x_3 + 25\end{aligned}$$

We start from an approximation, eg,  $x_1^{(0)} = 100, x_2^{(0)} = 100, x_3^{(0)} = 100, x_4^{(0)} = 100$ , and use the equations above to find a perhaps better approximation:

$$\begin{aligned}x_1^{(1)} &= 0.25x_2^{(0)} + 0.25x_3^{(0)} + 50.00 = 100.00 \\ x_2^{(1)} &= 0.25x_1^{(1)} + 0.25x_4^{(0)} + 50.00 = 100.00 \\ x_3^{(1)} &= 0.25x_1^{(1)} + 0.25x_4^{(0)} + 25.00 = 75.00 \\ x_4^{(1)} &= 0.25x_2^{(1)} + 0.25x_3^{(1)} + 25.00 = 68.75\end{aligned}$$

$$\begin{aligned}
 x_1^{(2)} &= 0.25x_2^{(1)} + 0.25x_3^{(1)} + 50.00 = 93.750 \\
 x_2^{(2)} &= 0.25x_1^{(2)} + 0.25x_4^{(1)} + 50.00 = 90.625 \\
 x_3^{(2)} &= 0.25x_1^{(2)} + 0.25x_4^{(1)} + 25.00 = 65.625 \\
 x_4^{(2)} &= 0.25x_2^{(2)} + 0.25x_3^{(2)} + 25.00 = 64.062
 \end{aligned}$$