# Gaussian Elimination: Solving system of equations

Key idea: keep track of transformations performed in putting matrix in echelon form.

Given matrix A, compute matrices M and U such that MA = U

- ▶ *U* is in echelon form
- M is invertible

To solve  $A\mathbf{x} = \mathbf{b}$ :

- ightharpoonup Compute M and U so that MA = U
- ightharpoonup Compute the matrix-vector product  $M\mathbf{b}$ , and solve  $U\mathbf{x} = M\mathbf{b}$ .

#### **Claim:** This gives correct solution to $A\mathbf{x} = \mathbf{b}$

**Proof:** Suppose **v** is a solution to U**x** = M**b**, so U**v** = M**b** 

- ▶ Multiply both sides by  $M^{-1}$ :  $M^{-1}(U\mathbf{v}) = M^{-1}M\mathbf{b}$
- Use associativity:  $(M^{-1}U)\mathbf{v} = (M^{-1}M)\mathbf{b}$
- ► Cancel  $M^{-1}$  and M:  $(M^{-1}U)\mathbf{v} = \mathbb{1}\mathbf{b}$
- ▶ Use  $M^{-1}U = A$ : A**v** = 1**b** = **b**

**How** to solve  $U\mathbf{x} = M\mathbf{b}$ ?

- ▶ If *U* is triangular, can solve using *back-substitution* (triangular\_solve)
- ▶ In general, can use similar algorithm

# Gaussian Elimination: Finding basis for null space

Instead of finding basis for null space of A, find basis for  $\{\mathbf{u} : \mathbf{u} * A = \mathbf{0}\} = \text{Null } A^T$ 

Find M, U such that MA = U and U is in echelon form and M is invertible

	0	1	2	3	4			A	В	C	D			0	1	2	3
0	1	0	0	0	0		0	1	0	1	0		0	1	0	1	0
1	1	1	0	0	0	s.le	1	1	1	1	0	_	1	0	1	0	0
2	1	1	1	0	0	*	2	0	1	0	1	_	2	0	0	0	1
3	1	0	1	1	0		3	1	1	1	1		3	0	0	0	0
4	1	1	1	0	1		4	0	0	0	1		4	0	0	0	0
		_							~				$\overline{}$		~		
		٨	1						Α						U		

Last two rows of U are zero vectors

- $\triangleright$  Row 3 of *U* is (row 3 of *M*) \* *A*
- $\triangleright$  Row 4 of *U* is (row 4 of *M*) \* *A*

# Gaussian Elimination: Finding basis for null space

Find M, U such that MA = U and U is in echelon form and M is invertible

		0	1	2	3	4			A	В	C	D			0	1	2	3
	0	1	0	0	0	0		0	1	0	1	0		0	1	0	1	0
	1	1	1	0	0	0	s.le	1	1	1	1	0	_	1	0	1	0	0
	2	1	1	1	0	0	*	2	0	1	0	1	_	2	0	0	0	1
	3	1	0	1	1	0		3	1	1	1	1		3	0	0	0	0
	4	1	1	1	0	1		4	0	0	0	1		4	0	0	0	0
`	_		$\overline{}$	_				_		~				_		~		
				Л	_					Α						U		
	Las	t tv	vo r	OWS	s ot	Ua	are zei	O VE	ecto	rs								

► Row 3 of *U* is (row 3 of *M*) \* *A* 

Therefore two rows in  $\{\mathbf{u}: \mathbf{u}*A=\mathbf{0}\}$  are rows 3 and 4 of M To show that these two rows form a basis for  $\{\mathbf{u}: \mathbf{u}*A=\mathbf{0}\}...$  dim Row A=3

By Rank-Nullity Theorem, dim Row  $A + \dim \text{Null } A^T = \text{number of rows} = 5$ Shows that dim Null  $A^T = 2$ 

Since M is invertible, all its rows are linearly independent.

$$\begin{bmatrix} & A & \\ & & \end{bmatrix} = \begin{bmatrix} & U_1 & \\ & & \end{bmatrix}$$

$$\begin{bmatrix} & M_1 & \\ & & \end{bmatrix} \begin{bmatrix} & A & \\ & & \end{bmatrix} = \begin{bmatrix} & U_2 & \\ & & \end{bmatrix}$$

$$\begin{bmatrix} & M_2 & \\ & & \end{bmatrix} \begin{bmatrix} & M_1 & \\ & & \end{bmatrix} \begin{bmatrix} & A & \\ & & \end{bmatrix} = \begin{bmatrix} & U_3 & \\ & & \end{bmatrix}$$

$$\begin{bmatrix} & M_3 & \\ & & \end{bmatrix} \begin{bmatrix} & M_2 & \\ & & \end{bmatrix} \begin{bmatrix} & M_1 & \\ & & \end{bmatrix} \begin{bmatrix} & A & \\ & & \end{bmatrix} = \begin{bmatrix} & U_4 & \\ & & \end{bmatrix}$$

$$\begin{bmatrix} 0 & 2 & 4 & 2 & 8 \\ 2 & 1 & 0 & 5 & 4 \\ 4 & 1 & 2 & 4 & 2 \\ 5 & 0 & 0 & 2 & 8 \end{bmatrix} = \begin{bmatrix} 0 & 2 & 4 & 2 & 8 \\ 2 & 1 & 0 & 5 & 4 \\ 4 & 1 & 2 & 4 & 2 \\ 5 & 0 & 0 & 2 & 8 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & -2 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 2 & 4 & 2 & 8 \\ 2 & 1 & 0 & 5 & 4 \\ 4 & 1 & 2 & 4 & 2 \\ 5 & 0 & 0 & 2 & 8 \end{bmatrix} = \begin{bmatrix} 0 & 2 & 4 & 2 & 8 \\ 2 & 1 & 0 & 5 & 4 \\ 0 & -1 & 2 & -6 & -6 \\ 5 & 0 & 0 & 2 & 8 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -2 & 1 & 0 \\ 0 & -2 & 5 & 0 & 0 & 2 & 8 \end{bmatrix} = \begin{bmatrix} 0 & 2 & 4 & 2 & 8 \\ 2 & 1 & 0 & 5 & 4 \\ 4 & 1 & 2 & 4 & 2 \\ 5 & 0 & 0 & 2 & 8 \end{bmatrix} = \begin{bmatrix} 0 & 2 & 4 & 2 & 8 \\ 2 & 1 & 0 & 5 & 4 \\ 0 & -1 & 2 & -6 & -6 \\ 0 & -2.5 & 0 & -10.5 & -2 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 2 & 4 & 2 & 8 \\ 2 & 1 & 0 & 5 & 4 \\ 4 & 1 & 2 & 4 & 2 \\ 5 & 0 & 0 & 2 & 8 \end{bmatrix} = \begin{bmatrix} 0 & 2 & 4 & 2 & 8 \\ 2 & 1 & 0 & 5 & 4 \\ 4 & 1 & 2 & 4 & 2 \\ 5 & 0 & 0 & 2 & 8 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -2 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 2 & 4 & 2 & 8 \\ 2 & 1 & 0 & 5 & 4 \\ 4 & 1 & 2 & 4 & 2 \\ 5 & 0 & 0 & 2 & 8 \end{bmatrix} = \begin{bmatrix} 0 & 2 & 4 & 2 & 8 \\ 2 & 1 & 0 & 5 & 4 \\ 0 & -1 & 2 & -6 & -6 \\ 5 & 0 & 0 & 2 & 8 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -2 & 1 & 0 \\ 0 & -2 & 5 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 2 & 4 & 2 & 8 \\ 2 & 1 & 0 & 5 & 4 \\ 4 & 1 & 2 & 4 & 2 \\ 5 & 0 & 0 & 2 & 8 \end{bmatrix} = \begin{bmatrix} 0 & 2 & 4 & 2 & 8 \\ 2 & 1 & 0 & 5 & 4 \\ 0 & -1 & 2 & -6 & -6 \\ 0 & -2.5 & 0 & -10.5 & -2 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 2 & 4 & 2 & 8 \\ 2 & 1 & 0 & 5 & 4 \\ 4 & 1 & 2 & 4 & 2 \\ 5 & 0 & 0 & 2 & 8 \end{bmatrix} = \begin{bmatrix} 0 & 2 & 4 & 2 & 8 \\ 2 & 1 & 0 & 5 & 4 \\ 4 & 1 & 2 & 4 & 2 \\ 5 & 0 & 0 & 2 & 8 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & -2 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 2 & 4 & 2 & 8 \\ 2 & 1 & 0 & 5 & 4 \\ 4 & 1 & 2 & 4 & 2 \\ 5 & 0 & 0 & 2 & 8 \end{bmatrix} = \begin{bmatrix} 0 & 2 & 4 & 2 & 8 \\ 2 & 1 & 0 & 5 & 4 \\ 0 & -1 & 2 & -6 & -6 \\ 5 & 0 & 0 & 2 & 8 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & -2 & 1 & 0 & 0 \\ 0 & -2 & 5 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 2 & 4 & 2 & 8 \\ 2 & 1 & 0 & 5 & 4 \\ 4 & 1 & 2 & 4 & 2 \\ 5 & 0 & 0 & 2 & 8 \end{bmatrix} = \begin{bmatrix} 0 & 2 & 4 & 2 & 8 \\ 2 & 1 & 0 & 5 & 4 \\ 0 & -1 & 2 & -6 & -6 \\ 0 & -2.5 & 0 & -10.5 & -2 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0.5 & 0 & 1 & 0 & 0 \\ 0.5 & 0 & 1 & 0 & 0 \\ 0 & 0 & -2 & 5 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 2 & 4 & 2 & 8 \\ 2 & 1 & 0 & 5 & 4 \\ 4 & 1 & 2 & 4 & 2 \\ 5 & 0 & 0 & 2 & 8 \end{bmatrix} = \begin{bmatrix} 0 & 2 & 4 & 2 & 8 \\ 2 & 1 & 0 & 5 & 4 \\ 0 & -1.5 & -2 & -6 & -6 \\ 0 & -2.5 & 0 & -10.5 & -2 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 2 & 4 & 2 & 8 \\ 2 & 1 & 0 & 5 & 4 \\ 4 & 1 & 2 & 4 & 2 \\ 5 & 0 & 0 & 2 & 8 \end{bmatrix} = \begin{bmatrix} 0 & 2 & 4 & 2 & 8 \\ 2 & 1 & 0 & 5 & 4 \\ 4 & 1 & 2 & 4 & 2 \\ 5 & 0 & 0 & 2 & 8 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & -2 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & -2 & 1 & 0 & 0 \\ 0 & -2 & 5 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 2 & 4 & 2 & 8 \\ 2 & 1 & 0 & 5 & 4 \\ 4 & 1 & 2 & 4 & 2 \\ 5 & 0 & 0 & 2 & 8 \end{bmatrix} = \begin{bmatrix} 0 & 2 & 4 & 2 & 8 \\ 2 & 1 & 0 & 5 & 4 \\ 0 & -1 & 2 & -6 & -6 \\ 5 & 0 & 0 & 2 & 8 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & -2.5 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 2 & 4 & 2 & 8 \\ 2 & 1 & 0 & 5 & 4 \\ 4 & 1 & 2 & 4 & 2 \\ 5 & 0 & 0 & 2 & 8 \end{bmatrix} = \begin{bmatrix} 0 & 2 & 4 & 2 & 8 \\ 2 & 1 & 0 & 5 & 4 \\ 0 & -1 & 2 & -6 & -6 \\ 0 & -2.5 & 0 & -10.5 & -2 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ .5 & -2 & 1 & 0 \\ 0 & -2.5 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 2 & 4 & 2 & 8 \\ 2 & 1 & 0 & 5 & 4 \\ 4 & 1 & 2 & 4 & 2 \\ 5 & 0 & 0 & 2 & 8 \end{bmatrix} = \begin{bmatrix} 0 & 2 & 4 & 2 & 8 \\ 2 & 1 & 0 & 5 & 4 \\ 0 & 0 & 4 & -5 & -2 \\ 0 & -2.5 & 0 & -10.5 & -2 \end{bmatrix}$$

- ▶ Maintain *M* (initially identity) and *U* (initially *A*)
- $\triangleright$  Whatever transformations you do to U, do same transformations to M

	0	1	2	3			A	В	C	D			Α	В	C	D	
0	1	0	0	0		0	0	0	1	1		0	0	0	1	1	ColumnA:
1	0	1	0	0	*	1	1	0	1	1	=	1	1	0	1	1	select row 1
2	0	0	1	0		2	1	0	0	1		2	1	0	0	1	add it to rows 2,3
3	0	0	0	1		3	1	1	1	1		3	1	1	1	1	

- ▶ Maintain *M* (initially identity) and *U* (initially *A*)
- $\blacktriangleright$  Whatever transformations you do to U, do same transformations to M

	0	1	2	3		<i>A</i>	В	C	D				A	В	C	D	
0	1	0	0	0	0	0	0	1	1			0	0	0	1	1	ColumnA:
	-		_		1	1	0	1	1	$\checkmark$	=	1	1	0	1	1	select row 1
2	0	0	1	0	2	1	0	0	1			2	1	0	0	1	add it to rows 2,3
3	0	0	0	1	3	1	1	1	1			3	1	1	1	1	

- ▶ Maintain *M* (initially identity) and *U* (initially *A*)
- ightharpoonup Whatever transformations you do to U, do same transformations to M

	0	1	2	3			A	В	C	D				A	В	C	D	
0	1	0	0	0		0	0	0	1	1			0	0	0	1	1	ColumnA:
1	0	1	0	0	*	1	1	0	1	1	$\checkmark$	=	1	1	0	1	1	select row 1
2	0	0	1	0		2	1	0	0	1			2	1	0	0	1	add it to rows 2,3
3	0	0	0	1		3	1	1	1	1			3	1	1	1	1	
0 1 2 3	0 1 0 0 0	1 0 1 1	0 0 1 0	3 0 0 0 1	*	0 1 2 3	0 1 1 1	0 0 0 0	C 1 1 0 1	D 1 1 1 1 1	<ul><li>✓</li></ul>	=	0 1 2 3	0 1 0 0	0 0 0 1	C 1 1 1 0	D 1 1 0 0	ColumnB: select row 3 add it to no rows ColumnC: select row 0

- ▶ Maintain *M* (initially identity) and *U* (initially *A*)
- $\blacktriangleright$  Whatever transformations you do to U, do same transformations to M

	0	1	2	3			A	В	С	D				A	В	С	D	
0	1	0	0	0		0	0	0	1	1			0	0	0	1	1	ColumnA:
1	0	1	0	0	*	1	1	0	1	1	$\checkmark$	=	1	1	0	1	1	select row 1
2	0	0	1	0		2	1	0	0	1			2	1	0	0	1	add it to rows 2,3
3	0	0	0	1		3	1	1	1	1			3	1	1	1	1	
0	0 1 0	1 0 1	2 0 0	3 0 0	*	0	0 1	<i>B</i> 0 0	<i>C</i> 1 1	<i>D</i> 1 1	· ✓ ✓	=	0 1	0 1	<i>B</i> 0 0	<i>C</i> 1 1	D 1 1	ColumnB: select row 3 add it to no rows ColumnC:
2	0	1	1	0		2	1	0	0	1			2	0	0	1	0	
3	0	1	0	1		3	1	1	1	1	$\checkmark$		3	0	1	0	0	select row 0 add it to row 2

- ► Maintain *M* (initially identity) and *U* (initially *A*)
- $\blacktriangleright$  Whatever transformations you do to U, do same transformations to M

0 1 2 3	0 1 0 0 0	1 0 1 0 0	2 0 0 1 0	3 0 0 0 1	*	0 1 2 3	0 1 1 1	0 0 0 0	1 1 0 1	D 1 1 1 1	✓	=	0 1 2 3	0 1 1 1	0 0 0 0	1 1 0 1	D 1 1 1 1	ColumnA: select row 1 add it to rows 2,3
0 1 2 3	0 1 0 0 0	1 0 1 1	0 0 1 0	3 0 0 0 1	*	0 1 2 3	0 1 1 1	0 0 0 1	1 1 0 1	D 1 1 1 1	√ √	=	0 1 2 3	0 1 0 0	0 0 0 1	1 1 1 0	D 1 1 0 0	ColumnB: select row 3 add it to no rows ColumnC: select row 0 add it to row 2
0 1 2 3	0 1 0 1 0	1 0 1 1	2 0 0 1 0	3 0 0 0 1	*	0 1 2 3	0 1 1 1	0 0 0 1	1 1 0 1	1 1 1 1	√ √ √ √	=	0 1 2 3	0 1 0 0	0 0 0 1	1 1 0 0	1 1 1 0	ColumnD: select row 2 done

#### Code for finding transformation to echelon form

- ▶ Initialize rowlist to be list of rows of A
- ▶ Initialize M\_rowlist to be list of rows of identity matrix

```
for c in sorted(col_labels, key=hash):
    rows_with_nonzero = [r for r in rows_left if rowlist[r][c] != 0]
    if rows_with_nonzero != []:
        pivot = rows_with_nonzero[0]
        rows_left.remove(pivot)
        new_M_rowlist.append(M_rowlist[pivot])
        for r in rows_with_nonzero[1:]:
            multiplier = rowlist[r][c]/rowlist[pivot][c]
            rowlist[r] -= multiplier*rowlist[pivot]
            M_rowlist[r] -= multiplier*M_rowlist[pivot]
    for r in rows_left: new_M_rowlist.append(M_rowlist[r])
```

Finally, return matrix M formed from M\_rowlist Code provided in module echelon

#### The black box starts to become less opaque

```
met project_along(p, v):
sigma = ((b*v)/(v*v)) if v*v != 0 else 0
return siama * v
                       def solve(A, b):
lef project_orthogonal(b
                             0.R = factor(A)
  for v in vlist:
       b = b - project
                             col_label_list =
                             return triangular
  aua_project_orthogon
   sigmadict = {len(vli 🛌
   for i,v in enumerate(vlist):
       sigma = (b*v)/def transformation(A,one=1, col_label_1
                        """Given a matrix A, and optionally
       b = b - siama*t),
                           compute matrix M such that M is
   return (b, sigmadi
                           U = M*A is in echelon form
ef orthogonalize(vlis
                        row_labels, col_labels = A.D
  vstarlist = 
                        m = len(row_labels)
   for v in vlist:
                        row_label_list = sorted(row_labels
     vstarlist.append
                        rowlist = [Vec(col_labels, {c:A[r.o
                     label list7
                        M_rows = transformation_rows(rowlise)
  auq_orthogonalize(
   vstorlist = [
```

The modules independence and solver both Gaussian elimination when working over GF(2):

- ▶ The procedure solve(A, b) computes a matrix M such that MA is in echelon form, and uses M to try to find a solution.
- ► The procedure rank(L) converts to echelon form and counts the nonzero rows to find the rank of I.

We saw that Gaussian elimination can be used to find a nonzero vector in the null space of a matrix.... You will use this in an algorithm for factoring integers.