

# Computational Physics I - Lecture 3, part 1

Adam Foster, Dorothea Golze, Patrick Rinke  
Levi Keller, Yashasvi Ranawat, Ygor Morais Jaques

Aalto University  
School of Science  
Department of Applied Physics

# Solutions of linear and non-linear equations

**Linear equation:**  $ax + b = c$

- sets of linear equations are very common in physics
- they can be solved with matrix algebra
- matrix algebra is one of the most important applications in computational physics

# Solutions of linear and non-linear equations

**Linear equation:**  $ax + b = c$

- sets of linear equations are very common in physics
- they can be solved with matrix algebra
- matrix algebra is one of the most important applications in computational physics

**Non linear equation:**  $x = f(x)$

- non-linear equations are even more common than linear
- they are much harder to solve than linear equations
- numeric approaches for non-linear eqns are very important

# Solutions of linear and non-linear equations

**Linear equation:**  $ax + b = c$  **today**

- sets of linear equations are very common in physics
- they can be solved with matrix algebra
- matrix algebra is one of the most important applications in computational physics

**Non linear equation:**  $x = f(x)$  **next week**

- non-linear equations are even more common than linear
- they are much harder to solve than linear equations
- numeric approaches for non-linear eqns are very important

# Simultaneous linear equations

**set of linear equations:**

$$\begin{aligned}2w + x + 4y + z &= -4 \\3w + 4x - y - z &= 3 \\w - 4x + y + 5z &= 9 \\2w - 2x + y + 3z &= 7\end{aligned}$$

- techniques for solving simultaneous sets of equations are well understood and straightforward,
- but humans are slow and prone to error in such calculations
- computers are perfectly suited for this, in particular for large systems with many variables

# Simultaneous linear equations

- cast the set of equations into matrix form:

$$\begin{pmatrix} 2 & 1 & 4 & 1 \\ 3 & 4 & -1 & -1 \\ 1 & -4 & 1 & 5 \\ 2 & -2 & 1 & 3 \end{pmatrix} \begin{pmatrix} w \\ x \\ y \\ z \end{pmatrix} = \begin{pmatrix} -4 \\ 3 \\ 9 \\ 7 \end{pmatrix} \quad \text{or in short} \quad \mathbf{Ax} = \mathbf{v}$$

# Simultaneous linear equations

- cast the set of equations into matrix form:

$$\begin{pmatrix} 2 & 1 & 4 & 1 \\ 3 & 4 & -1 & -1 \\ 1 & -4 & 1 & 5 \\ 2 & -2 & 1 & 3 \end{pmatrix} \begin{pmatrix} w \\ x \\ y \\ z \end{pmatrix} = \begin{pmatrix} -4 \\ 3 \\ 9 \\ 7 \end{pmatrix} \quad \text{or in short} \quad \mathbf{Ax} = \mathbf{v}$$

- algebraically, inversion seems easiest solution:

$$\mathbf{x} = \mathbf{A}^{-1}\mathbf{v}$$

# Simultaneous linear equations

- cast the set of equations into matrix form:

$$\begin{pmatrix} 2 & 1 & 4 & 1 \\ 3 & 4 & -1 & -1 \\ 1 & -4 & 1 & 5 \\ 2 & -2 & 1 & 3 \end{pmatrix} \begin{pmatrix} w \\ x \\ y \\ z \end{pmatrix} = \begin{pmatrix} -4 \\ 3 \\ 9 \\ 7 \end{pmatrix} \quad \text{or in short} \quad \mathbf{Ax} = \mathbf{v}$$

- algebraically, inversion seems easiest solution:

$$\mathbf{x} = \mathbf{A}^{-1}\mathbf{v}$$

- But numerically matrix inversion is not the best solution!
- There are more efficient ways.



# Gaussian elimination and back substitution

- suppose we could transform the equations into this form:

$$\begin{pmatrix} 1 & a_{01} & a_{02} & a_{03} \\ 0 & 1 & a_{12} & a_{13} \\ 0 & 0 & 1 & a_{23} \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} w \\ x \\ y \\ z \end{pmatrix} = \begin{pmatrix} v_0 \\ v_1 \\ v_2 \\ v_3 \end{pmatrix}$$



# Gaussian elimination and back substitution

- suppose we could transform the equations into this form:

$$\begin{pmatrix} 1 & a_{01} & a_{02} & a_{03} \\ 0 & 1 & a_{12} & a_{13} \\ 0 & 0 & 1 & a_{23} \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} w \\ x \\ y \\ z \end{pmatrix} = \begin{pmatrix} v_0 \\ v_1 \\ v_2 \\ v_3 \end{pmatrix}$$

- then the solution is simple:

$$z = v_3$$

$$y = v_2 - a_{23}z$$

$$x = v_1 - a_{12}y - a_{13}z$$

$$w = v_0 - a_{01}x - a_{02}y - a_{03}z$$

# Gaussian elimination and back substitution

- suppose we could transform the equations into this form:

$$\begin{pmatrix} 1 & a_{01} & a_{02} & a_{03} \\ 0 & 1 & a_{12} & a_{13} \\ 0 & 0 & 1 & a_{23} \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} w \\ x \\ y \\ z \end{pmatrix} = \begin{pmatrix} v_0 \\ v_1 \\ v_2 \\ v_3 \end{pmatrix}$$

- then the solution is simple:

$$z = v_3$$

$$y = v_2 - a_{23}z$$

$$x = v_1 - a_{12}y - a_{13}z$$

$$w = v_0 - a_{01}x - a_{02}y - a_{03}z$$

**back substitution**



# Gaussian elimination and back substitution

## Gaussian elimination

- To arrive at the upper tridiagonal form we apply two rules consecutively:
  1. If we multiply any row of  $A$  and the corresponding row of  $v$  by a constant, the solution does not change.
  2. If we add to or subtract from any row of  $A$  a multiple of any other row, and we do the same for  $v$ , then the solution does not change.

# Gaussian elimination and back substitution

Rule 1:

$$\begin{pmatrix} 2 & 1 & 4 & 1 \\ 3 & 4 & -1 & -1 \\ 1 & -4 & 1 & 5 \\ 2 & -2 & 1 & 3 \end{pmatrix} \begin{pmatrix} w \\ x \\ y \\ z \end{pmatrix} = \begin{pmatrix} -4 \\ 3 \\ 9 \\ 7 \end{pmatrix}$$

Diagram illustrating Rule 1: The first row of the coefficient matrix is divided by 2. Orange arrows point from the text "divide by 2" to the elements 2, 1, 4, 1, and the constant -4.

# Gaussian elimination and back substitution

Rule 1:

**divide by 2**

$$\begin{pmatrix} 2 & 1 & 4 & 1 \\ 3 & 4 & -1 & -1 \\ 1 & -4 & 1 & 5 \\ 2 & -2 & 1 & 3 \end{pmatrix} \begin{pmatrix} w \\ x \\ y \\ z \end{pmatrix} = \begin{pmatrix} -4 \\ 3 \\ 9 \\ 7 \end{pmatrix}$$

**now equal to 1**

$$\begin{pmatrix} 1 & 0.5 & 2 & 0.5 \\ 3 & 4 & -1 & -1 \\ 1 & -4 & 1 & 5 \\ 2 & -2 & 1 & 3 \end{pmatrix} \begin{pmatrix} w \\ x \\ y \\ z \end{pmatrix} = \begin{pmatrix} -2 \\ 3 \\ 9 \\ 7 \end{pmatrix}$$

# Gaussian elimination and back substitution

## Rule 2:

subtract 3 times first row from 2nd

$$\begin{pmatrix} 1 & 0.5 & 2 & 0.5 \\ 3 & 4 & -1 & -1 \\ 1 & -4 & 1 & 5 \\ 2 & -2 & 1 & 3 \end{pmatrix} \begin{pmatrix} w \\ x \\ y \\ z \end{pmatrix} = \begin{pmatrix} -2 \\ 3 \\ 9 \\ 7 \end{pmatrix}$$

# Gaussian elimination and back substitution

**Rule 2:** **now equal to 0**

$$\begin{pmatrix} 1 & 0.5 & 2 & 0.5 \\ 0 & 2.5 & -7 & -2.5 \\ 1 & -4 & 1 & 5 \\ 2 & -2 & 1 & 3 \end{pmatrix} \begin{pmatrix} w \\ x \\ y \\ z \end{pmatrix} = \begin{pmatrix} -2 \\ 9 \\ 9 \\ 7 \end{pmatrix}$$

**subtract 3 times first row from 2nd**

$$\begin{pmatrix} 1 & 0.5 & 2 & 0.5 \\ 3 & 4 & -1 & -1 \\ 1 & -4 & 1 & 5 \\ 2 & -2 & 1 & 3 \end{pmatrix} \begin{pmatrix} w \\ x \\ y \\ z \end{pmatrix} = \begin{pmatrix} -2 \\ 3 \\ 9 \\ 7 \end{pmatrix}$$



# Gaussian elimination and back substitution

$$\begin{pmatrix} 1 & 0.5 & 2 & 0.5 \\ 0 & 2.5 & -7 & -2.5 \\ 1 & -4 & 1 & 5 \\ 2 & -2 & 1 & 3 \end{pmatrix} \begin{pmatrix} w \\ x \\ y \\ z \end{pmatrix} = \begin{pmatrix} -2 \\ 9 \\ 9 \\ 7 \end{pmatrix}$$

- By applying Rule 1 and Rule 2 successively, we can set all diagonal elements to 1 and the lower triangle to 0.

# Linear equations - Exercise 1

**Solve:**

$$\begin{pmatrix} 2 & 1 & 4 & 1 \\ 3 & 4 & -1 & -1 \\ 1 & -4 & 1 & 5 \\ 2 & -2 & 1 & 3 \end{pmatrix} \begin{pmatrix} w \\ x \\ y \\ z \end{pmatrix} = \begin{pmatrix} -4 \\ 3 \\ 9 \\ 7 \end{pmatrix}$$

1. Complete the Gaussian elimination part of the program.
2. Add a print statement that prints the matrix at every step to check that the program is eliminating correctly.
3. Check your final solution for the vector  $\mathbf{x} = (w, x, y, z)$

## Talking points:

1. What do you observe?
2. Is your solution correct?

# Linear equations - Example 1

**Show model solution.**

# Linear equations - Example 1

**Show model solution.**

**Key concept: Gaussian elimination**

With *Gaussian elimination* and *back substitution* we can solve a set of linear equations efficiently.

# Gaussian elimination - Pivoting

## Pivoting in Gaussian elimination

- suppose the set is slightly different:

$$\begin{pmatrix} 0 & 1 & 4 & 1 \\ 3 & 4 & -1 & -1 \\ 1 & -4 & 1 & 5 \\ 2 & -2 & 1 & 3 \end{pmatrix} \begin{pmatrix} w \\ x \\ y \\ z \end{pmatrix} = \begin{pmatrix} -4 \\ 3 \\ 9 \\ 7 \end{pmatrix}$$

# Gaussian elimination - Pivoting

## Pivoting in Gaussian elimination

- suppose the set is slightly different:

**division by 0**


$$\begin{pmatrix} 0 & 1 & 4 & 1 \\ 3 & 4 & -1 & -1 \\ 1 & -4 & 1 & 5 \\ 2 & -2 & 1 & 3 \end{pmatrix} \begin{pmatrix} w \\ x \\ y \\ z \end{pmatrix} = \begin{pmatrix} -4 \\ 3 \\ 9 \\ 7 \end{pmatrix}$$

# Gaussian elimination - Pivoting

## Pivoting in Gaussian elimination

- suppose the set is slightly different:

**division by 0**


$$\begin{pmatrix} 0 & 1 & 4 & 1 \\ 3 & 4 & -1 & -1 \\ 1 & -4 & 1 & 5 \\ 2 & -2 & 1 & 3 \end{pmatrix} \begin{pmatrix} w \\ x \\ y \\ z \end{pmatrix} = \begin{pmatrix} -4 \\ 3 \\ 9 \\ 7 \end{pmatrix}$$

- The solution is to swap this row with another one to make that the first row. Then Gaussian elimination and back substitution can be applied again. Care has to be taken, however, to not introduce problems elsewhere.

# Linear equations - LU decomposition

**Sets of equations:**  $Ax = v_1, Ax = v_2, \dots, Ax = v_n$

**Question:**

**If we want to apply  $A$  to different vectors  $v$ , Gauss Elimination is wasteful, because it has to be carried out over and over again. Is there a better way?**



# Linear equations - LU decomposition

- We wish to transform a general matrix  $\mathbf{A}$

$$\mathbf{A} = \begin{pmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{pmatrix}$$

so that it can be applied to any vector  $\mathbf{v}$ .

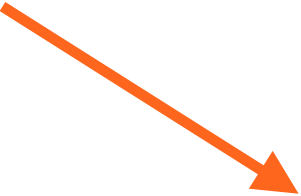
- We are looking for matrix operations that transform  $\mathbf{A}$ .

# Linear equations - LU decomposition

- The operations that turn the first row into its correct form can be encapsulated in the following matrix multiplication

$$\frac{1}{a_{00}} \begin{pmatrix} 1 & 0 & 0 & 0 \\ -a_{10} & a_{00} & 0 & 0 \\ -a_{20} & 0 & a_{00} & 0 \\ -a_{30} & 0 & 0 & a_{00} \end{pmatrix} \begin{pmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{pmatrix} = \begin{pmatrix} 1 & b_{01} & b_{02} & b_{03} \\ 0 & b_{11} & b_{12} & b_{13} \\ 0 & b_{21} & b_{22} & b_{23} \\ 0 & b_{31} & b_{32} & b_{33} \end{pmatrix}$$

lower triangular  
matrix


$$\mathbf{L}_0 \mathbf{A} = \mathbf{B}$$

# Linear equations - LU decomposition

- The operations that turn the first row into its correct form can be encapsulated in the following matrix multiplication

$$\frac{1}{a_{00}} \begin{pmatrix} 1 & 0 & 0 & 0 \\ -a_{10} & a_{00} & 0 & 0 \\ -a_{20} & 0 & a_{00} & 0 \\ -a_{30} & 0 & 0 & a_{00} \end{pmatrix} \begin{pmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{pmatrix} = \begin{pmatrix} 1 & b_{01} & b_{02} & b_{03} \\ 0 & b_{11} & b_{12} & b_{13} \\ 0 & b_{21} & b_{22} & b_{23} \\ 0 & b_{31} & b_{32} & b_{33} \end{pmatrix}$$

**lower triangular matrix**

$$\mathbf{L}_0 \mathbf{A} = \mathbf{B}$$

**now we have to continue with B**

# Linear equations - LU decomposition

- Operating on **B** with a new matrix

$$\frac{1}{b_{11}} \begin{pmatrix} b_{11} & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -b_{21} & b_{11} & 0 \\ 0 & -b_{31} & 0 & b_{11} \end{pmatrix} \begin{pmatrix} 1 & b_{01} & b_{02} & b_{03} \\ 0 & b_{11} & b_{12} & b_{13} \\ 0 & b_{21} & b_{22} & b_{23} \\ 0 & b_{31} & b_{32} & b_{33} \end{pmatrix} = \begin{pmatrix} 1 & c_{01} & c_{02} & c_{03} \\ 0 & 1 & c_{12} & c_{13} \\ 0 & 0 & c_{22} & c_{23} \\ 0 & 0 & c_{32} & c_{33} \end{pmatrix}$$

lower triangular  
matrix


$$\mathbf{L}_1 \mathbf{B} = \mathbf{L}_1 \mathbf{L}_0 \mathbf{A} = \mathbf{C}$$

# Linear equations - LU decomposition

- Operating on **B** with a new matrix

$$\frac{1}{b_{11}} \begin{pmatrix} b_{11} & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -b_{21} & b_{11} & 0 \\ 0 & -b_{31} & 0 & b_{11} \end{pmatrix} \begin{pmatrix} 1 & b_{01} & b_{02} & b_{03} \\ 0 & b_{11} & b_{12} & b_{13} \\ 0 & b_{21} & b_{22} & b_{23} \\ 0 & b_{31} & b_{32} & b_{33} \end{pmatrix} = \begin{pmatrix} 1 & c_{01} & c_{02} & c_{03} \\ 0 & 1 & c_{12} & c_{13} \\ 0 & 0 & c_{22} & c_{23} \\ 0 & 0 & c_{32} & c_{33} \end{pmatrix}$$

lower triangular  
matrix

$$\mathbf{L}_1 \mathbf{B} = \mathbf{L}_1 \mathbf{L}_0 \mathbf{A} = \mathbf{C}$$

we need two  
more operations  
for **C**

# Linear equations - LU decomposition

- The final two operations are:

$$\mathbf{L}_2 = \frac{1}{c_{22}} \begin{pmatrix} c_{22} & 0 & 0 & 0 \\ 0 & c_{22} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -c_{32} & c_{22} \end{pmatrix} \quad \text{and} \quad \mathbf{L}_3 = \frac{1}{d_{33}} \begin{pmatrix} d_{33} & 0 & 0 & 0 \\ 0 & d_{33} & 0 & 0 \\ 0 & 0 & d_{33} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- Putting it all together we have:

$$\mathbf{L}_3 \mathbf{L}_2 \mathbf{L}_1 \mathbf{L}_0 \mathbf{A} = \mathbf{L}_3 \mathbf{L}_2 \mathbf{L}_1 \mathbf{L}_0 \mathbf{v} = \mathbf{U} \mathbf{v}$$

# Linear equations - LU decomposition

- The final two operations are:

$$\mathbf{L}_2 = \frac{1}{c_{22}} \begin{pmatrix} c_{22} & 0 & 0 & 0 \\ 0 & c_{22} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -c_{32} & c_{22} \end{pmatrix} \quad \text{and} \quad \mathbf{L}_3 = \frac{1}{d_{33}} \begin{pmatrix} d_{33} & 0 & 0 & 0 \\ 0 & d_{33} & 0 & 0 \\ 0 & 0 & d_{33} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- Putting it all together we have:

$$\mathbf{L}_3 \mathbf{L}_2 \mathbf{L}_1 \mathbf{L}_0 \mathbf{A} = \mathbf{L}_3 \mathbf{L}_2 \mathbf{L}_1 \mathbf{L}_0 \mathbf{v} = \mathbf{U} \mathbf{v}$$

  
upper diagonal  
matrix

# Linear equations - LU decomposition

- The final two operations are:

$$\mathbf{L}_2 = \frac{1}{c_{22}} \begin{pmatrix} c_{22} & 0 & 0 & 0 \\ 0 & c_{22} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -c_{32} & c_{22} \end{pmatrix} \quad \text{and} \quad \mathbf{L}_3 = \frac{1}{d_{33}} \begin{pmatrix} d_{33} & 0 & 0 & 0 \\ 0 & d_{33} & 0 & 0 \\ 0 & 0 & d_{33} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- Putting it all together we have:

$$\mathbf{L}_3 \mathbf{L}_2 \mathbf{L}_1 \mathbf{L}_0 \mathbf{A} = \mathbf{L}_3 \mathbf{L}_2 \mathbf{L}_1 \mathbf{L}_0 \mathbf{v} = \mathbf{U} \mathbf{v}$$

now we need to find the inverse of the left-hand side to obtain an expression for  $\mathbf{A}$

upper diagonal matrix



# Linear equations - LU decomposition

- With the following inverses of  $\mathbf{L}_n$

$$\mathbf{L}_0^{-1} = \begin{pmatrix} a_{00} & 0 & 0 & 0 \\ a_{10} & 1 & 0 & 0 \\ a_{20} & 0 & 1 & 0 \\ a_{30} & 0 & 0 & 1 \end{pmatrix}, \quad \mathbf{L}_1^{-1} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & b_{11} & 0 & 0 \\ 0 & b_{21} & 1 & 0 \\ 0 & b_{31} & 0 & 1 \end{pmatrix}$$
$$\mathbf{L}_2^{-1} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & c_{22} & 0 \\ 0 & 0 & c_{32} & 1 \end{pmatrix}, \quad \mathbf{L}_3^{-1} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & d_{33} \end{pmatrix}$$

- we obtain:

$$\mathbf{L} = \mathbf{L}_0^{-1} \mathbf{L}_1^{-1} \mathbf{L}_2^{-1} \mathbf{L}_3^{-1} = \begin{pmatrix} a_{00} & 0 & 0 & 0 \\ a_{10} & b_{11} & 0 & 0 \\ a_{20} & b_{21} & c_{22} & 0 \\ a_{30} & b_{31} & c_{32} & d_{33} \end{pmatrix}$$

# Linear equations - LU decomposition

lower triangular  
matrix

easy to calculate  
from known  
elements

- we obtain:

$$\mathbf{L} = \mathbf{L}_0^{-1} \mathbf{L}_1^{-1} \mathbf{L}_2^{-1} \mathbf{L}_3^{-1} = \begin{pmatrix} a_{00} & 0 & 0 & 0 \\ a_{10} & b_{11} & 0 & 0 \\ a_{20} & b_{21} & c_{22} & 0 \\ a_{30} & b_{31} & c_{32} & d_{33} \end{pmatrix}$$

# Linear equations - LU decomposition

$$\text{LU decomposition: } \mathbf{A} = \mathbf{L}\mathbf{U}$$

lower triangular  
matrix

easy to calculate  
from known  
elements

- we obtain:

$$\mathbf{L} = \mathbf{L}_0^{-1}\mathbf{L}_1^{-1}\mathbf{L}_2^{-1}\mathbf{L}_3^{-1} = \begin{pmatrix} a_{00} & 0 & 0 & 0 \\ a_{10} & b_{11} & 0 & 0 \\ a_{20} & b_{21} & c_{22} & 0 \\ a_{30} & b_{31} & c_{32} & d_{33} \end{pmatrix}$$

# Linear equations - LU decomposition

LU decomposition of our problem:  $A\mathbf{x} = \mathbf{v}$

$$A\mathbf{x} = LU\mathbf{x} = L\mathbf{y} = \mathbf{v} \quad \text{with} \quad U\mathbf{x} = \mathbf{y}$$

# Linear equations - LU decomposition

LU decomposition of our problem:  $A\mathbf{x} = \mathbf{v}$

$$A\mathbf{x} = LU\mathbf{x} = L\mathbf{y} = \mathbf{v} \quad \text{with} \quad U\mathbf{x} = \mathbf{y}$$

1st back substitution  
gives  $\mathbf{y}$  from  $\mathbf{v}$



# Linear equations - LU decomposition


LU decomposition of our problem:  $Ax = v$

$$Ax = LUx = Ly = v \quad \text{with} \quad Ux = y$$

1st back substitution  
gives  $y$  from  $v$



2nd back substitution  
gives  $x$  from  $y$



# Linear equations - LU decomposition


LU decomposition of our problem:  $Ax = v$

$$Ax = LUx = Ly = v \quad \text{with} \quad Ux = y$$

1st back substitution  
gives  $y$  from  $v$



2nd back substitution  
gives  $x$  from  $y$



With LU decomposition and two back substitutions we can solve  $Ax = v$  for any  $v$ .



# Linear equations - LU decomposition

$$\text{LU decomp.: } \mathbf{Ax} = \mathbf{LUx} = \mathbf{Ly} = \mathbf{v} \quad \text{with} \quad \mathbf{Ux} = \mathbf{y}$$

- For the exercises, you will write your own LU decomposition.



# Linear equations - LU decomposition

**LU decomp.:**  $Ax = LUx = Ly = v$  with  $Ux = y$

- For the exercises, you will write your own LU decomposition.
- Python has a build in solver for simultaneous linear equations that uses LU decomposition and back substitution.

```
from numpy.linalg import solve  
x = solve(A,v)
```



# Linear equations - LU decomposition

**LU decomp.:**  $Ax = LUx = Ly = v$  with  $Ux = y$

- For the exercises, you will write your own LU decomposition.
- Python has a built-in solver for simultaneous linear equations that uses LU decomposition and back substitution.

```
from numpy.linalg import solve
x = solve(A,v)
```

- In the SciPy package you can find a LU decomposition function under `linalg`.

# Linear equations - Exercise 2

**Solve:**

$$\begin{pmatrix} 2 & 1 & 4 & 1 \\ 3 & 4 & -1 & -1 \\ 1 & -4 & 1 & 5 \\ 2 & -2 & 1 & 3 \end{pmatrix} \begin{pmatrix} w \\ x \\ y \\ z \end{pmatrix} = \begin{pmatrix} -4 \\ 3 \\ 9 \\ 7 \end{pmatrix}$$

- For this problem, the LU decomposition is:

$$\mathbf{L} = \begin{pmatrix} 2 & 0 & 0 & 0 \\ 3 & 2.5 & 0 & 0 \\ 1 & -4.5 & -13.6 & 0 \\ 2 & -3 & -11.4 & -1 \end{pmatrix} \quad \mathbf{U} = \begin{pmatrix} 1 & 0.5 & 2 & 0.5 \\ 0 & 1 & -2.8 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

# Linear equations - Exercise 2

1. Verify that  $L*U$  gives the matrix  $A$ . You can use the numpy routine `matmul`.
2. Perform the double back substitution  $Ly=v$  and  $Ux=y$  with the numpy.linalg routine `solve`.
3. Verify your result.
4. Apply the LU decomposition to the new vectors  $v_1=(1,0,0,0)$ ,  $v_2=(0,1,0,0)$ ,  $v_3=(0,0,1,0)$ ,  $v_4=(0,0,0,1)$ .
5. Check your result with `solve`.

## Talking points:

1. What do you observe?
2. What happens when you apply the LU decomposition to the vectors  $v_1$  to  $v_4$ ?

# Linear equations - LU decomposition

## Key concept: LU decomposition

The *LU decomposition* is one of several factorisations of a square matrix  $A$ . It factors  $A$  into a lower and an upper triangular matrix. The LU decomposition is the first step in an efficient solution of linear sets of equations.



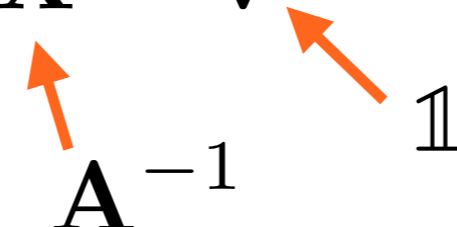
# Linear equations - Matrix inversion

**Matrix inverse:**  $\mathbf{A}\mathbf{A}^{-1} = \mathbf{1}$

# Linear equations - Matrix inversion

**Matrix inverse:**  $AA^{-1} = \mathbb{1}$

- LU decomposition is a common way to invert a matrix.

$$AX = LUX = V$$


The diagram shows the equation  $AX = LUX = V$ . Two orange arrows point from the text below to the equation. One arrow points from  $A^{-1}$  to the  $X$  term in  $AX$ . The other arrow points from  $\mathbb{1}$  to the  $V$  term in  $LUX = V$ .

**When  $V$  is the identity,  $X$  is the inverse of  $A$ .**

# Linear equations - Matrix inversion

**Matrix inverse:**  $\mathbf{A}\mathbf{A}^{-1} = \mathbf{1}$

- LU decomposition is a common way to invert a matrix.

$$\mathbf{A}\mathbf{X} = \mathbf{L}\mathbf{U}\mathbf{X} = \mathbf{1}$$

- With back substitution we can repeatedly solve for the columns of  $\mathbf{X}$  and so gradually build up the inverse of  $\mathbf{A}$ .



# Linear equations - Matrix inversion

**Matrix inverse:**  $\mathbf{A}\mathbf{A}^{-1} = \mathbf{1}$

- LU decomposition is a common way to invert a matrix.

$$\mathbf{A}\mathbf{X} = \mathbf{L}\mathbf{U}\mathbf{X} = \mathbf{1}$$

- With back substitution we can repeatedly solve for the columns of  $\mathbf{X}$  and so gradually build up the inverse of  $\mathbf{A}$ .
- This is a good example for different right hand sides  $\mathbf{v}$ .

# Linear equations - Matrix inversion

**Matrix inverse:**  $\mathbf{A}\mathbf{A}^{-1} = \mathbf{1}$

- LU decomposition is a common way to invert a matrix.

$$\mathbf{A}\mathbf{X} = \mathbf{L}\mathbf{U}\mathbf{X} = \mathbf{1}$$

- numpy has a build in function that inverts a matrix.

```
from numpy.linalg import inv
X = inv(A)
```

# Linear equations - Matrix inversion

## Key concept: matrix inversion

The *LU decomposition* provides one way to invert a square matrix numerically.

# Tridiagonal and banded matrices

**Tridiagonal matrix:** 
$$A = \begin{pmatrix} a_{00} & a_{01} & & & \\ a_{10} & a_{11} & a_{12} & & \\ & a_{21} & a_{22} & a_{23} & \\ & & a_{32} & a_{33} & a_{34} \\ & & & a_{43} & a_{44} \end{pmatrix}$$

- Gaussian elimination is especially efficient as we do not need to go through all the rows of the matrix, but only the row immediately below the current one.

# Tridiagonal and banded matrices

- Gaussian elimination is more efficient than LU decomposition. The result is:

$$\begin{pmatrix} 1 & b_{01} & 0 & 0 \\ 0 & 1 & b_{12} & 0 \\ 0 & 0 & 1 & b_{23} \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{pmatrix}$$

- Back substitution is also simple:

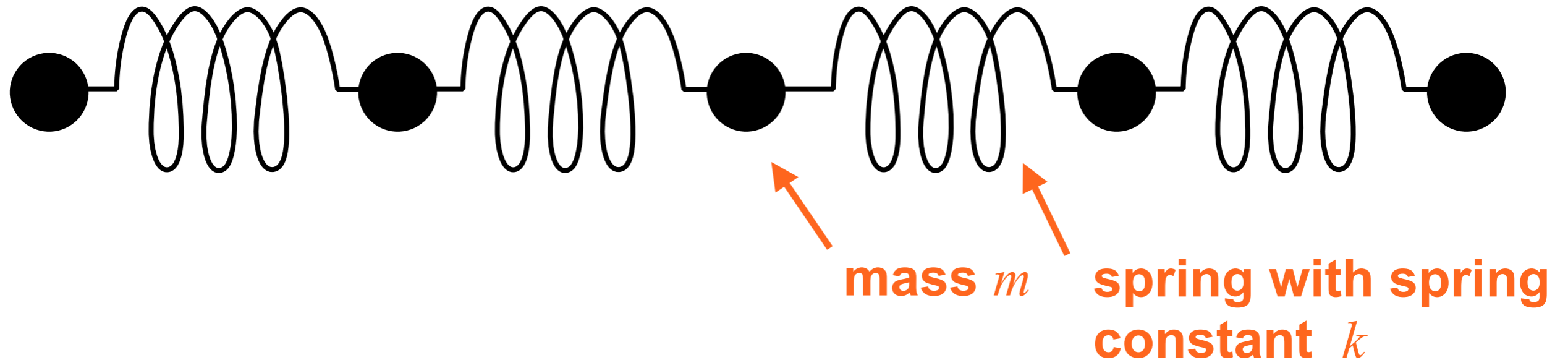
$$x_3 = y_3$$

$$x_2 = y_2 - b_{23}x_3$$

$$x_1 = y_1 - b_{12}x_2$$

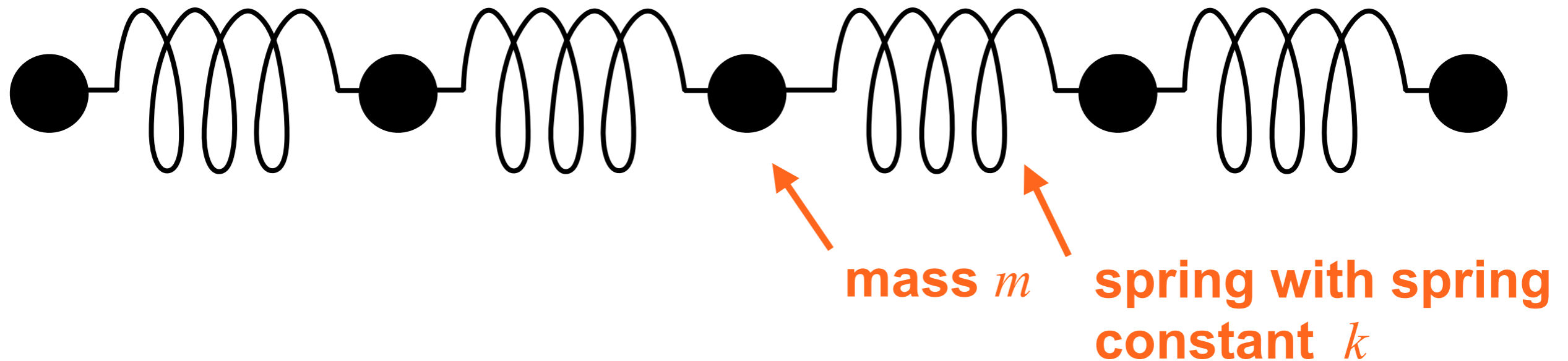
$$x_0 = y_0 - b_{01}x_1$$

# Example 1 - Vibration in a 1D system



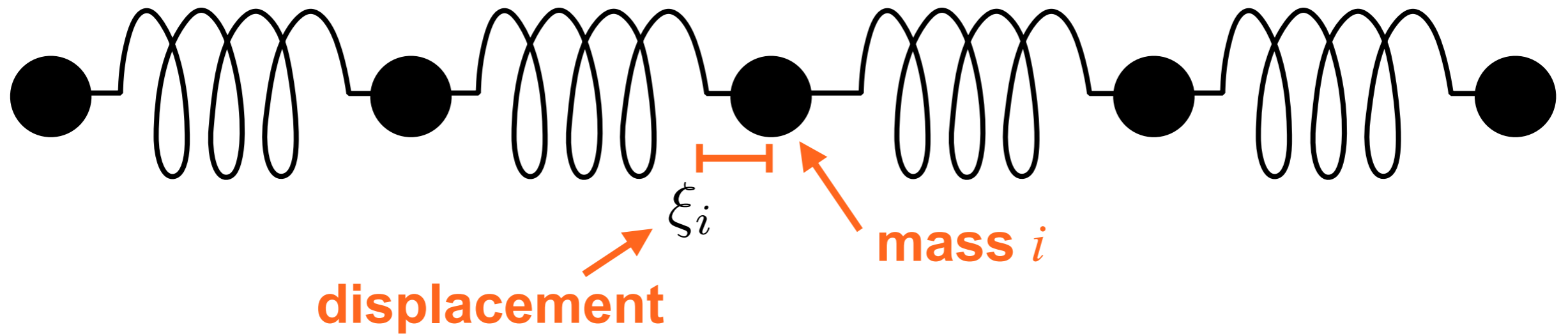
- $N$  masses  $m$  in a row joint by identical springs
- We ignore gravity and perturb the system with a force.

# Example 1 - Vibration in a 1D system



- $N$  masses  $m$  in a row joint by identical springs
- We ignore gravity and perturb the system with a force.
- The masses will start to vibrate relative to each other, which gives a good model for atoms in a solid.

# Example 1 - Vibration in a 1D system



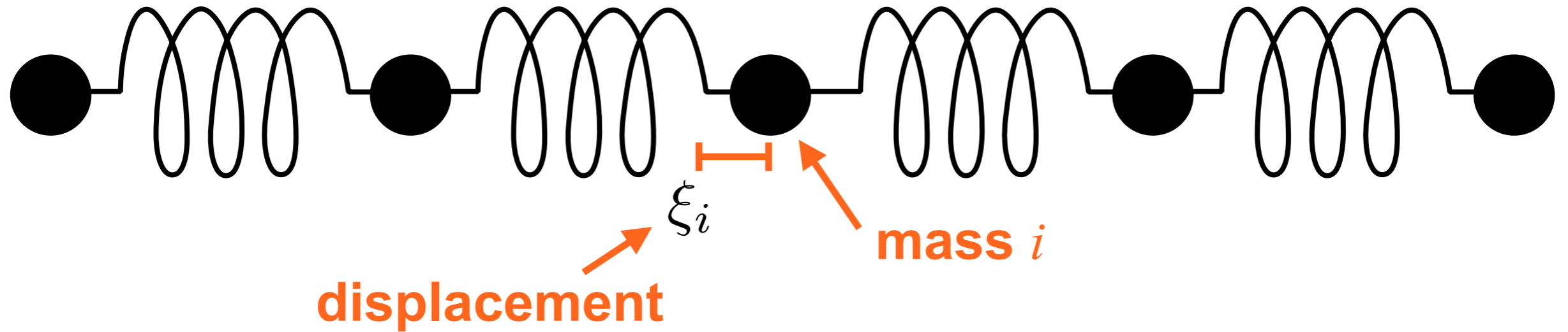
- The equations of motion for the masses are given by Newton's second law:

$$m \frac{d^2 \xi_i}{dt^2} = k(\xi_{i+1} - \xi_i) + k(\xi_{i-1} - \xi_i) + F_i$$

external force



# Example 1 - Vibration in a 1D system

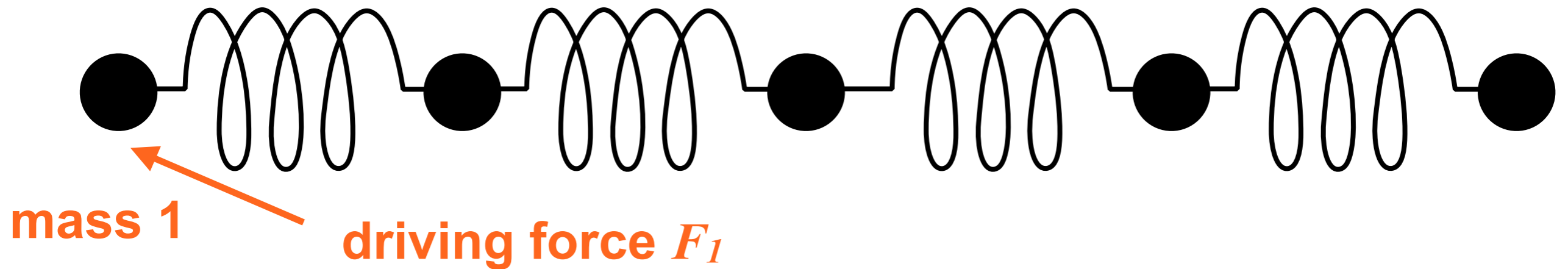


- The chain is finite so that mass 1 and  $N$  satisfy the following condition:

$$m \frac{d^2 \xi_1}{dt^2} = k(\xi_2 - \xi_1) + F_1$$

$$m \frac{d^2 \xi_N}{dt^2} = k(\xi_{N-1} - \xi_N) + F_N$$

# Example 1 - Vibration in a 1D system

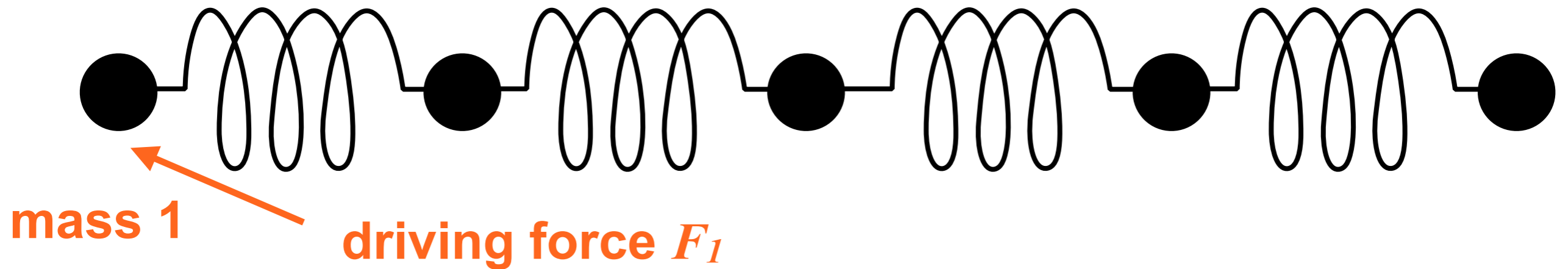


- We apply a harmonic (i.e. sinusoidal) driving force where  $C$  is a complex constant:

$$F_1 = C e^{i\omega t}$$

- This could for instance be an electromagnetic wave.

# Example 1 - Vibration in a 1D system

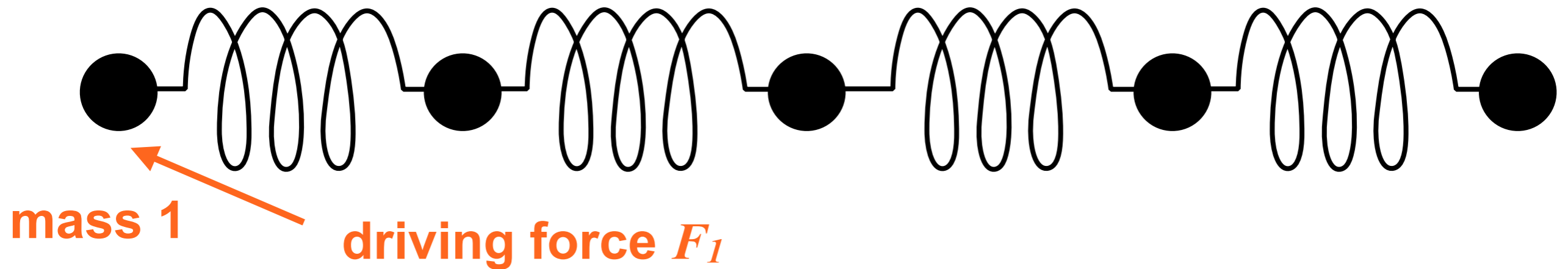


- The masses (atoms) will oscillate in with angular frequency  $\omega$ :

$$\xi_i(t) = x_i e^{i\omega t}$$

amplitude

# Example 1 - Vibration in a 1D system



- Substituting this into our set of Newton's equations gives:

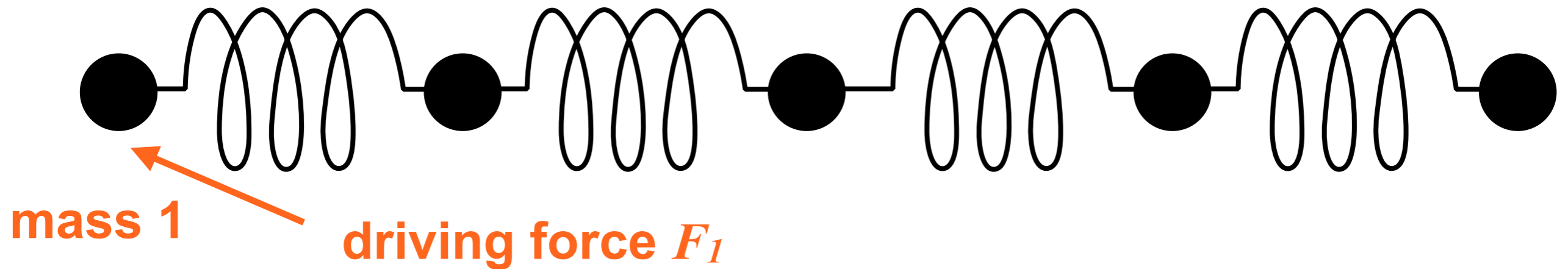
$$-m\omega^2 x_1 = k(x_2 - x_1) + C$$

$$-m\omega^2 x_i = k(x_{i+1} - x_i) + k(x_{i-1} - x_i)$$

$$-m\omega^2 x_N = k(x_{N-1} - x_N)$$

where  $i$  ranges from 2 to  $N-1$ .

# Example 1 - Vibration in a 1D system



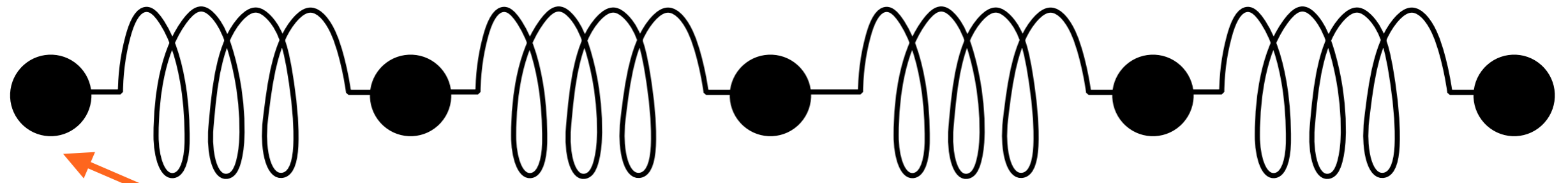
- With  $\alpha = 2k - m\omega^2$  we can rearrange to:

$$(\alpha - k)x_1 - kx_2 = C$$

$$\alpha x_i - kx_{i-1} - kx_{i+1} = 0$$

$$(\alpha - k)x_N - kx_{N-1} = 0$$

# Example 1 - Vibration in a 1D system



- But this is nothing else than a set of linear equations in tridiagonal form:

$$\begin{pmatrix} (\alpha - k) & -k & & & & \\ -k & \alpha & -k & & & \\ & -k & \alpha & -k & & \\ & & \ddots & \ddots & \ddots & \\ & & & -k & \alpha & -k \\ & & & & -k & (\alpha - k) \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{N-1} \\ x_N \end{pmatrix} = \begin{pmatrix} C \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix}$$

# Linear equations - Exercise 3

**Solve** the connected linear spring model for 26 masses with  $C=1$ ,  $m=1$ ,  $k=2$  and  $\omega=2$ .

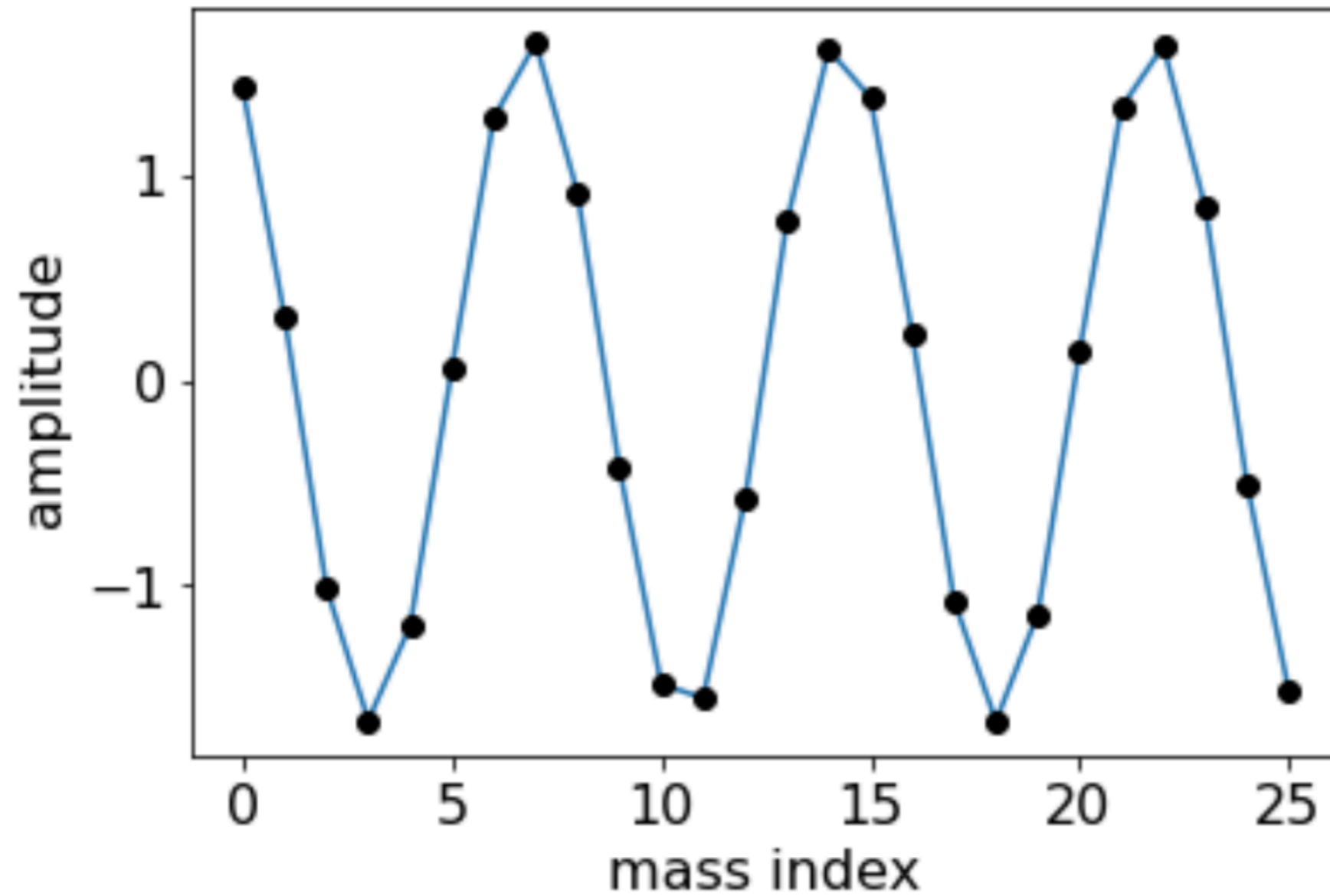
In the `in_class_exercise` notebook you find a skeleton program that calls the subroutine `banded` for the solution of a banded matrix.

1. Download the file `banded.py` from MyCourses.
2. Complete the initialisation part of the program.
3. Add plot statements to plot your results in a graph.

**Talking points:**

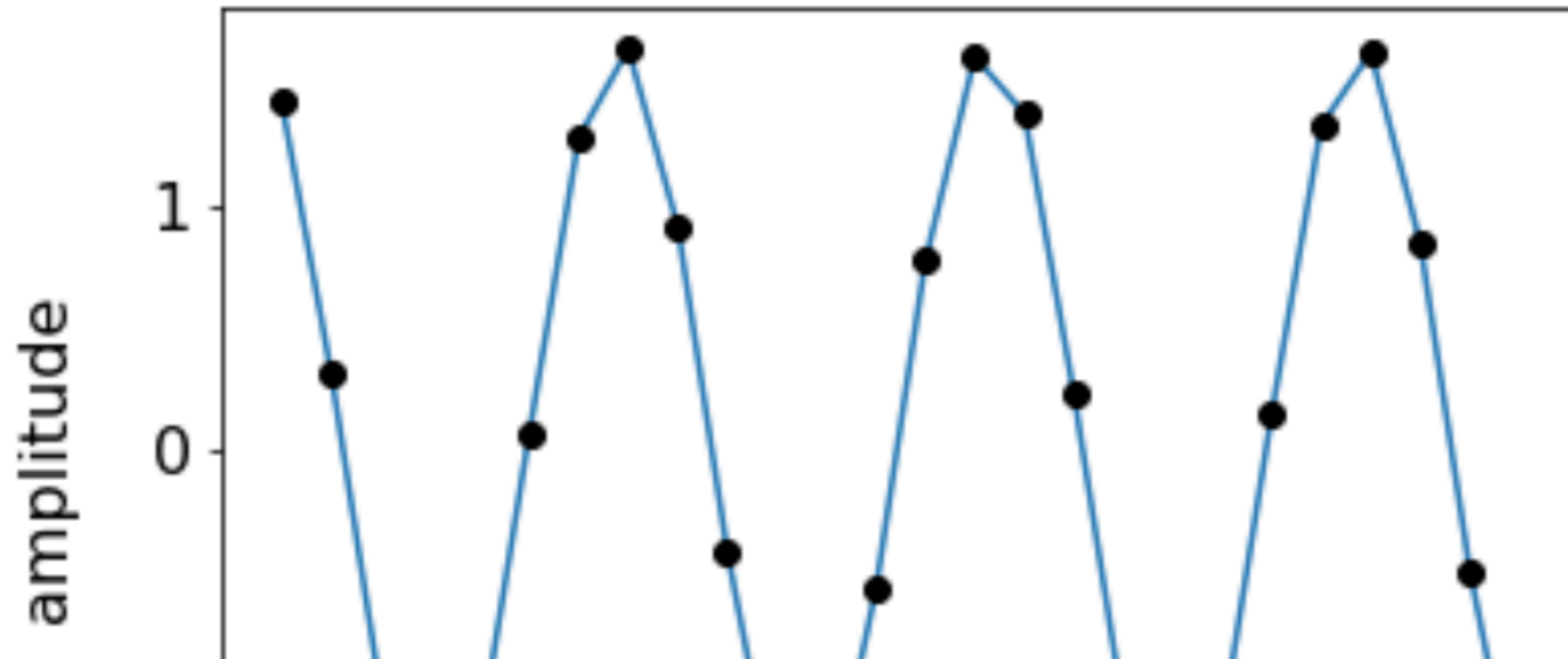
1. **What do you observe?**
2. **What can you say about the amplitudes of the vibrating masses?**

# Linear equations - Linear spring model





# Linear equations - Linear spring model



## Key concept: banded matrices

Many problems in physics result in simplified matrix equations such as banded matrices. Such problems can be solved efficiently with the techniques we just learned.

# Linear equations - Eigenvalues and eigenvectors

**Eigenvalue problem:**  $\mathbf{A}\mathbf{v} = \lambda\mathbf{v}$

- For a symmetric (or Hermitian)  $N \times N$  matrix  $\mathbf{A}$ , there are  $N$  eigenvalues  $\lambda$  and eigenvectors  $\mathbf{v}$ .

# Linear equations - Eigenvalues and eigenvectors

$$\text{Eigenvalue problem: } \mathbf{A}\mathbf{v} = \lambda\mathbf{v}$$

- For a symmetric (or Hermitian)  $N \times N$  matrix  $\mathbf{A}$ , there are  $N$  eigenvalues  $\lambda$  and eigenvectors  $\mathbf{v}$ .
- We can combine all the solutions for the  $N$  many  $\mathbf{v}_i$  into one equation:

$$\mathbf{A}\mathbf{V} = \mathbf{V}\mathbf{D}$$

orthogonal matrix

diagonal matrix containing the eigenvalues

$$\mathbf{V}^T\mathbf{V} = \mathbf{V}\mathbf{V}^T = \mathbf{1}$$

# Linear equations - Eigenvalues and eigenvectors

**Eigenvalue problem:**  $\mathbf{A}\mathbf{v} = \lambda\mathbf{v}$

- To find the eigenvalues of  $\mathbf{A}$ , we use the QR decomposition.

# Linear equations - Eigenvalues and eigenvectors

**Eigenvalue problem:**  $\mathbf{A}\mathbf{v} = \lambda\mathbf{v}$

- To find the eigenvalues of  $\mathbf{A}$ , we use the QR decomposition.
- The QR decomposition is similar to the LU decomposition.

$\mathbf{Q}$  : **orthogonal matrix**  $\mathbf{Q}^T \mathbf{Q} = \mathbf{1}$

$\mathbf{R}$  : **upper trigonal matrix**

# Linear equations - Eigenvalues and eigenvectors

**Eigenvalue problem:**  $\mathbf{A}\mathbf{v} = \lambda\mathbf{v}$

- To find the eigenvalues of  $\mathbf{A}$ , we use the QR decomposition.
- The QR decomposition is similar to the LU decomposition.

$\mathbf{Q}$  : **orthogonal matrix**  $\mathbf{Q}^T \mathbf{Q} = \mathbf{1}$

$\mathbf{R}$  : **upper trigonal matrix**

- For the graded exercises, you will write your own QR decomposition.

# Linear equations - Eigenvalues and eigenvectors

**Eigenvalue problem:**  $\mathbf{A}\mathbf{v} = \lambda\mathbf{v}$

- To find the eigenvalues of  $\mathbf{A}$ , we use the QR decomposition.
- The QR decomposition is similar to the LU decomposition.

$\mathbf{Q}$  : **orthogonal matrix**  $\mathbf{Q}^T \mathbf{Q} = \mathbf{1}$

$\mathbf{R}$  : **upper trigonal matrix**

- For the graded exercises, you will write your own QR decomposition.
- In Python, `SciPy` provides a QR decomposition.

# Linear equations - Finding eigenvalues

- Suppose we have the QR decomposition of  $A$ :

$$A = Q_1 R_1$$



# Linear equations - Finding eigenvalues

- Suppose we have the QR decomposition of  $\mathbf{A}$ :

$$\mathbf{A} = \mathbf{Q}_1 \mathbf{R}_1$$

- We multiply from the left with  $\mathbf{Q}_1^T$

$$\mathbf{Q}_1^T \mathbf{A} = \mathbf{Q}_1^T \mathbf{Q}_1 \mathbf{R}_1 = \mathbf{R}_1$$

# Linear equations - Finding eigenvalues

- Suppose we have the QR decomposition of  $\mathbf{A}$ :

$$\mathbf{A} = \mathbf{Q}_1 \mathbf{R}_1$$

- We multiply from the left with  $\mathbf{Q}_1^T$

$$\mathbf{Q}_1^T \mathbf{A} = \mathbf{Q}_1^T \mathbf{Q}_1 \mathbf{R}_1 = \mathbf{R}_1$$

- Then we define a new matrix  $\mathbf{A}_1$ :

$$\mathbf{A}_1 = \mathbf{R}_1 \mathbf{Q}_1 = \mathbf{Q}_1^T \mathbf{A} \mathbf{Q}_1$$

$\mathbf{A}_1$  is an orthogonal transformation of  $\mathbf{A}$

# Linear equations - Finding eigenvalues

- Next we will repeat the process of forming the QR decomposition of  $\mathbf{A}_1$  and forming a new matrix  $\mathbf{A}_2$  and the decomposing that into its own QR decomposition and so forth:

$$\mathbf{A}_1 = \mathbf{Q}_1^T \mathbf{A} \mathbf{Q}_1$$

$$\mathbf{A}_2 = \mathbf{Q}_2^T \mathbf{Q}_1^T \mathbf{A} \mathbf{Q}_1 \mathbf{Q}_2$$

⋮

$$\mathbf{A}_k = (\mathbf{Q}_k^T \dots \mathbf{Q}_1^T) \mathbf{A} (\mathbf{Q}_1 \dots \mathbf{Q}_k)$$

# Linear equations - Finding eigenvalues

- Next we will repeat the process of forming the QR decomposition of  $\mathbf{A}_1$  and forming a new matrix  $\mathbf{A}_2$  and the decomposing that into its own QR decomposition and so forth:

$$\mathbf{A}_1 = \mathbf{Q}_1^T \mathbf{A} \mathbf{Q}_1$$

$$\mathbf{A}_2 = \mathbf{Q}_2^T \mathbf{Q}_1^T \mathbf{A} \mathbf{Q}_1 \mathbf{Q}_2$$

⋮

$$\mathbf{A}_k = (\mathbf{Q}_k^T \dots \mathbf{Q}_1^T) \mathbf{A} (\mathbf{Q}_1 \dots \mathbf{Q}_k)$$

- It can be proven, that  $\mathbf{A}_k$  becomes eventually diagonal:

$$\mathbf{A}_k = \mathbf{D}$$

# Linear equations - Finding eigenvalues

$$\mathbf{A}_k = (\mathbf{Q}_k^T \dots \mathbf{Q}_1^T) \mathbf{A} (\mathbf{Q}_1 \dots \mathbf{Q}_k) = \mathbf{D}$$

- With

$$\mathbf{V} = \mathbf{Q}_1 \mathbf{Q}_2 \mathbf{Q}_3 \dots \mathbf{Q}_k = \prod_{i=1}^k \mathbf{Q}_i$$

this gives us:

$$\mathbf{D} = \mathbf{A}_k = \mathbf{V}^T \mathbf{A} \mathbf{V} \quad \text{or} \quad \mathbf{A} \mathbf{V} = \mathbf{V} \mathbf{D}$$

# Linear equations - Finding eigenvalues

$$\mathbf{A}_k = (\mathbf{Q}_k^T \dots \mathbf{Q}_1^T) \mathbf{A} (\mathbf{Q}_1 \dots \mathbf{Q}_k) = \mathbf{D}$$

- With

$$\mathbf{V} = \mathbf{Q}_1 \mathbf{Q}_2 \mathbf{Q}_3 \dots \mathbf{Q}_k = \prod_{i=1}^k \mathbf{Q}_i$$

this gives us:

$$\mathbf{D} = \mathbf{A}_k = \mathbf{V}^T \mathbf{A} \mathbf{V} \quad \text{or} \quad \mathbf{A} \mathbf{V} = \mathbf{V} \mathbf{D}$$

eigenvalues

eigenvectors

This is our eigenvalue equation!

# Linear equations - The QR algorithm

# Linear equations - The QR algorithm

The QR algorithm for diagonalizing a matrix (i.e. for calculating its eigenvalues and eigenvectors):



# Linear equations - The QR algorithm

The QR algorithm for diagonalizing a matrix (i.e. for calculating its eigenvalues and eigenvectors):

1. Create an  $N \times N$  matrix  $\mathbf{V}$  and set it to the identity matrix.

# Linear equations - The QR algorithm

The QR algorithm for diagonalizing a matrix (i.e. for calculating its eigenvalues and eigenvectors):

1. Create an  $N \times N$  matrix  $\mathbf{V}$  and set it to the identity matrix.
2. Choose a target accuracy  $\varepsilon$  for the off-diagonal elements of  $\mathbf{D}$ .

# Linear equations - The QR algorithm

The QR algorithm for diagonalizing a matrix (i.e. for calculating its eigenvalues and eigenvectors):

1. Create an  $N \times N$  matrix  $\mathbf{V}$  and set it to the identity matrix.
2. Choose a target accuracy  $\varepsilon$  for the off-diagonal elements of  $\mathbf{D}$ .
3. Calculate the QR decomposition  $\mathbf{A} = \mathbf{QR}$ .

# Linear equations - The QR algorithm

The QR algorithm for diagonalizing a matrix (i.e. for calculating its eigenvalues and eigenvectors):

1. Create an  $N \times N$  matrix  $\mathbf{V}$  and set it to the identity matrix.
2. Choose a target accuracy  $\varepsilon$  for the off-diagonal elements of  $\mathbf{D}$ .
3. Calculate the QR decomposition  $\mathbf{A} = \mathbf{QR}$ .
4. Update  $\mathbf{A}$  to the new value  $\mathbf{A} = \mathbf{RQ}$ .

# Linear equations - The QR algorithm

The QR algorithm for diagonalizing a matrix (i.e. for calculating its eigenvalues and eigenvectors):

1. Create an  $N \times N$  matrix  $\mathbf{V}$  and set it to the identity matrix.
2. Choose a target accuracy  $\varepsilon$  for the off-diagonal elements of  $\mathbf{D}$ .
3. Calculate the QR decomposition  $\mathbf{A} = \mathbf{QR}$ .
4. Update  $\mathbf{A}$  to the new value  $\mathbf{A} = \mathbf{RQ}$ .
5. Multiply  $\mathbf{V}$  on the right by  $\mathbf{Q}$ .

# Linear equations - The QR algorithm

The QR algorithm for diagonalizing a matrix (i.e. for calculating its eigenvalues and eigenvectors):

1. Create an  $N \times N$  matrix  $\mathbf{V}$  and set it to the identity matrix.
2. Choose a target accuracy  $\varepsilon$  for the off-diagonal elements of  $\mathbf{D}$ .
3. Calculate the QR decomposition  $\mathbf{A} = \mathbf{Q}\mathbf{R}$ .
4. Update  $\mathbf{A}$  to the new value  $\mathbf{A} = \mathbf{R}\mathbf{Q}$ .
5. Multiply  $\mathbf{V}$  on the right by  $\mathbf{Q}$ .
6. Check the off-diagonal elements of  $\mathbf{A}$ . If they are all less than  $\varepsilon$ , we are done. Otherwise go back to step 3.

# Linear equations - QR decomposition

## Key concept: QR decomposition

The *QR decomposition* is another factorisations of a square matrix  $A$ . It factors  $A$  into an orthogonal and an upper triangular matrix. Successive application of the QR decomposition diagonalises a matrix and finds its eigenvalues and eigenvectors.

# Linear equations - Eigenvalues and eigenvectors

- Numpy has build-in routines in its `linalg` module for finding eigenvalues and eigenvectors:

```
from numpy.linalg import eigh,eigvalsh  
x,V = eigh(A)  
x = eigvalsh(A)
```

eigenvalues and  
eigenvectors

returns just eigenvalues



# Linear equations - Exercise 4

$$\mathbf{A} = \begin{pmatrix} 2 & 6 & 7 \\ 1 & 0 & -1 \\ 2 & 3 & -2 \end{pmatrix}$$

- The matrix  $\mathbf{A}$  has the QR decomposition:

$$\mathbf{Q} = \frac{1}{3} \begin{pmatrix} 2 & 2 & 1 \\ 1 & -2 & 2 \\ 2 & -1 & -2 \end{pmatrix} \quad \mathbf{R} = 3 \begin{pmatrix} 1 & 2 & 1 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{pmatrix}$$

1. Verify that  $\mathbf{Q}^T \mathbf{Q} = \mathbf{1}$
2. Check that  $\mathbf{QR} = \mathbf{A}$

# Linear equations - Exercise 5

**Diagonalize:**  $A = \begin{pmatrix} 2 & 1 & 2 \\ 1 & 0 & 3 \\ 2 & 3 & -2 \end{pmatrix}$

1. Calculate the eigenvalues and eigenvectors of  $A$  using the numpy function `eigh`.
2. Verify that  $V^T A V = \mathbb{1}$

**Talking points:**

1. What do you observe?

