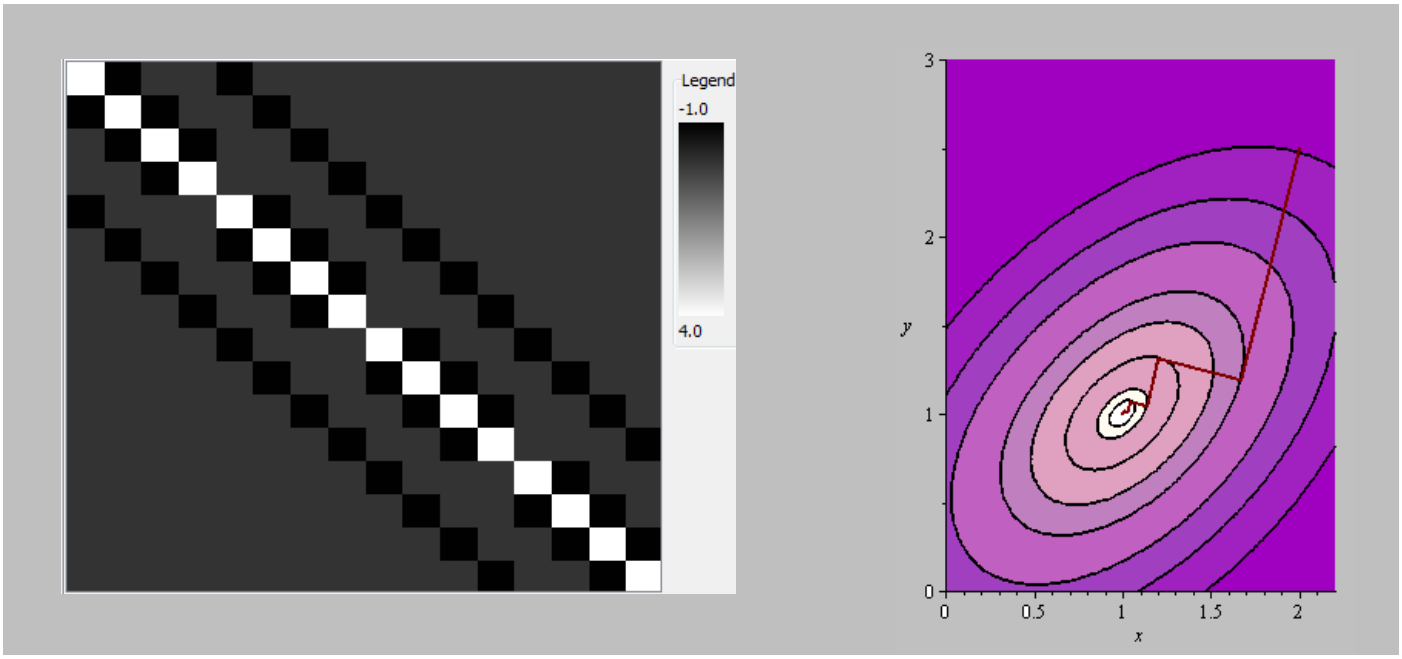
Large Scale Linear Systems of Equations



Mihály Klincsik

Large Scale Linear Systems of Equations

Pécs

2019

The Large Scale Linear Systems of Equations course material was developed under the project EFOP 3.4.3-16-2016-00005 "Innovative university in a modern city: open-minded, value-driven and inclusive approach in a 21st century higher education model".

EFOP 3.4.3-16-2016-00005 "Innovative university in a modern city: open-minded, value-driven and inclusive approach in a 21st century higher education model".

Mihály Klincsik

Large Scale Linear Systems of Equations

Pécs

2019

A Large Scale Linear Systems of Equations tananyag az EFOP-3.4.3-16-2016-00005 azonosító számú,
„Korszerű egyetem a modern városban: Értékközpontúság, nyitottság és befogadó szemlélet egy 21. századi felsőoktatási modellben” című projekt keretében valósul meg.

1. Matrices, graphs. Condition number.

LARGE-SCALE LINEAR SYSTEMS OF EQUATIONS
Computer Science Engineering Msc
Theoretical and Practical lesson

1. Matrices, vectors in Maple. Visualizations. Graph representations.

Give symbolically $n = 4$ dimensional vector and $n \times n$ matrix using Maple!

```
> restart : with(LinearAlgebra) : with(plots) :
> n := 4
> V, M := Vector(n, symbol = v), Matrix(n, symbol = m)
```

$$n := 4$$

$$V, M := \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix}, \begin{bmatrix} m_{1,1} & m_{1,2} & m_{1,3} & m_{1,4} \\ m_{2,1} & m_{2,2} & m_{2,3} & m_{2,4} \\ m_{3,1} & m_{3,2} & m_{3,3} & m_{3,4} \\ m_{4,1} & m_{4,2} & m_{4,3} & m_{4,4} \end{bmatrix}$$

Produce the **multiplication** between matrix and vector! (Dot product)

```
> M.V
```

$$\begin{bmatrix} m_{1,1} v_1 + m_{1,2} v_2 + m_{1,3} v_3 + m_{1,4} v_4 \\ m_{2,1} v_1 + m_{2,2} v_2 + m_{2,3} v_3 + m_{2,4} v_4 \\ m_{3,1} v_1 + m_{3,2} v_2 + m_{3,3} v_3 + m_{3,4} v_4 \\ m_{4,1} v_1 + m_{4,2} v_2 + m_{4,3} v_3 + m_{4,4} v_4 \end{bmatrix}$$

Let us take sub-matrices: a row vector, a column vector, a sub-matrix!

```
> M[1];# first row of matrix M
```

$$\begin{bmatrix} m_{1,1} & m_{1,2} & m_{1,3} & m_{1,4} \end{bmatrix}$$

```
> M[1..n, 2];# second column of matrix M    M[.., 2]; # same as above
```

$$\begin{bmatrix} m_{1,2} \\ m_{2,2} \\ m_{3,2} \\ m_{4,2} \end{bmatrix}$$

$$\begin{bmatrix} m_{1,2} \\ m_{2,2} \\ m_{3,2} \\ m_{4,2} \end{bmatrix}$$

```
> M[2..3, 3..4];# an inside block of matrix M
```

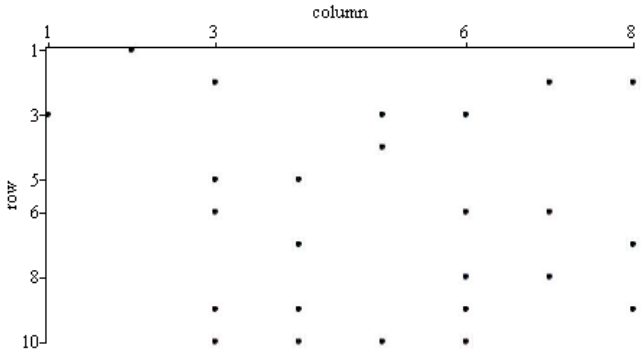
$$\begin{bmatrix} m_{2,3} & m_{2,4} \\ m_{3,3} & m_{3,4} \end{bmatrix}$$

Generating random **spare matrix** and visualization

```
> C := RandomMatrix(8, 10, density = 0.25)
```

$$C := \begin{bmatrix} 0 & 0 & 27 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 92 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -46 & 0 & 0 & -61 & 22 & 0 & 0 & 25 & -93 \\ 0 & 0 & 0 & 0 & -78 & 0 & 82 & 0 & -2 & -2 \\ 0 & 0 & -19 & 99 & 0 & 0 & 0 & 0 & 0 & -4 \\ 0 & 0 & -50 & 0 & 0 & 90 & 0 & 24 & -50 & 69 \\ 0 & -81 & 0 & 0 & 0 & 88 & 0 & 20 & 0 & 0 \\ 0 & 91 & 0 & 0 & 0 & 0 & -67 & 0 & -18 & 0 \end{bmatrix}$$

```
> sparsematrixplot(C, 'matrixview', symbol = solidcircle);
```



A graph G = (V,E) consists of a set V of nodes and a set of edges E ⊂ V ×V . Edges are denoted as pairs (v, v') ∈ V ×V with two distinct elements.
The sparsity pattern of a general matrix A can be represented by a graph G = (V, E) with nodes V and edges E, its so-called adjacency graph.
Here, the set V of nodes is simply the set of unknowns $\{x_i$, i = 1 . . . n} (or the corresponding indices i), and two nodes $x_i \neq x_j$ are connected by an edge

$(x_i , x_j) \in$ E iff $A_{i,j} \neq 0$, i.e. if equation i involves the unknown x_j .

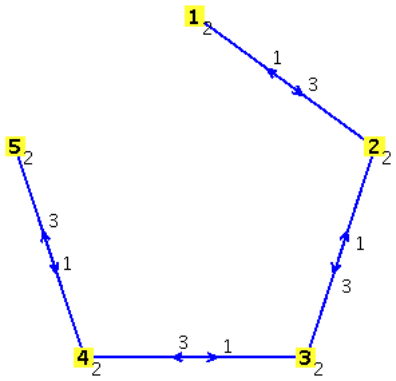
Generating graph on the basis of adjacency matrix

```
> with(GraphTheory) :
> A1 := BandMatrix([1, 2, 3], 1, 5);#band matrix
```

$$A1 := \begin{bmatrix} 2 & 3 & 0 & 0 & 0 \\ 1 & 2 & 3 & 0 & 0 \\ 0 & 1 & 2 & 3 & 0 \\ 0 & 0 & 1 & 2 & 3 \\ 0 & 0 & 0 & 1 & 2 \end{bmatrix}$$

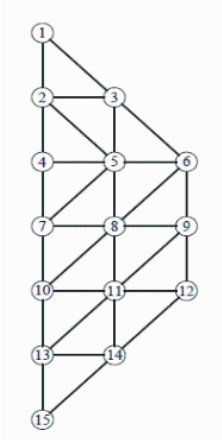
```
> G1 := Graph(directed, A1);
> DrawGraph(G1)
```

G1 := Graph 1: a directed weighted graph with 5 vertices and 13 arc(s)



1.1. Example.

Giving the adjacency matrix of the following mesh and drawing the graph with styles "spring" and "planar" !



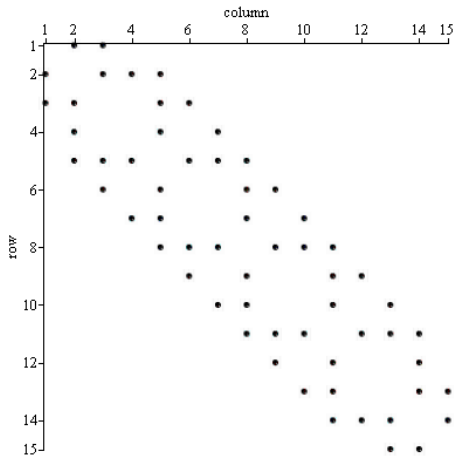
Solution

> $A2 := Matrix(15, \{(1,2)=1, (1,3)=1, (2,3)=1, (2,4)=1, (2,5)=1, (3,5)=1, (3,6)=1, (4,5)=1, (4,7)=1, (5,6)=1, (5,7)=1, (5,8)=1, (6,8)=1, (6,9)=1, (7,8)=1, (7,10)=1, (8,9)=1, (8,10)=1, (8,11)=1, (9,11)=1, (9,12)=1, (10,11)=1, (10,13)=1, (11,12)=1, (11,13)=1, (11,14)=1, (12,14)=1, (13,14)=1, (13,15)=1, (14,15)=1\}, fill=0, shape=symmetric)$

$$A2 := \begin{bmatrix} 15 \text{ x } 15 \text{ Matrix} \\ \text{Data Type: anything} \\ \text{Storage: triangular}_{upper} \\ \text{Order: Fortran\_order} \end{bmatrix}$$

Since the size of the matrix is greater than 10x10, therefore only a space holder shows the sizes and type of the matrix. Double click on space holder you get a matrix browser and can see in picture format.

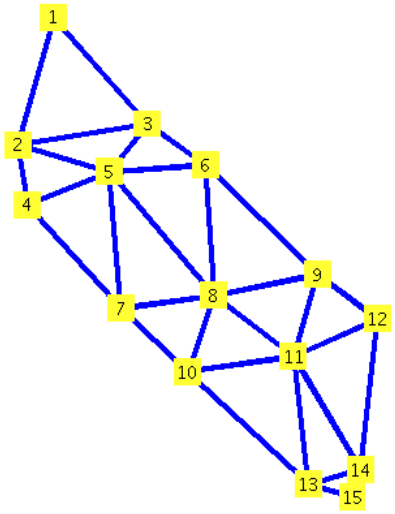
> $sparsematrixplot(A2, 'matrixview', symbol=solidcircle);$



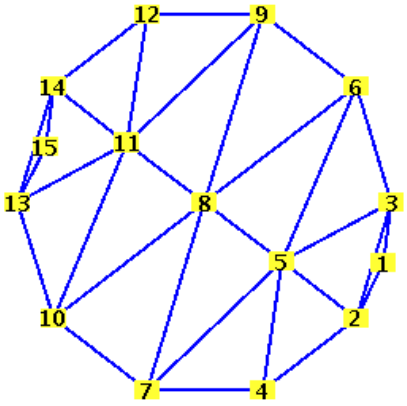
> $G2 := Graph(A2)$

> $DrawGraph(G2, style = spring)$

G2 := Graph 2: an undirected unweighted graph with 15 vertices and 30 edge(s)

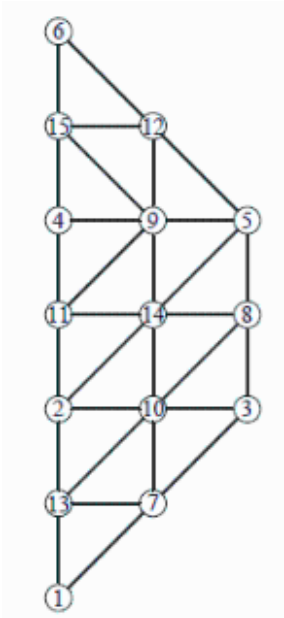


> $DrawGraph(G2, style = planar)$



1.1. Exercise.

Give the adjacency matrix of the following mesh! Plot the matrix in sparse format! Draw the graph again in spring and planar styles!



1.2. Exercise. Giving a directed graph G which adjacency matrix A is given below

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Draw the graph in spring style.

2. Vector norms and matrix norms in Maple

Consider

$$A \cdot x = b$$

systems of linear equations with $x = x_0$ approximation solution and $x = x^*$ **exact solution**.

Measure the error with $e = x^* - x_0$ difference vector, if we know x^* exact solution. The problem is to determine the value of x^*, so we measure the accuracy by using the

$$r = b - A \cdot x$$

residual vector $||r||$ norm or $\frac{||r||}{||b||}$ relative to b vector norm. When these are "near" zeros, then the computation algorithm stops.

2.1. DEFINITION. Vector norms

Let X a linear space. The mapping $||.||: x \to [0, \infty)$ is called **norm**, iff fulfilled the following criterions

(i) $||x|| \geq 0$ and $||x|| = 0$ if **and** only if $x = 0$

(ii) $||\alpha \cdot x|| = |\alpha| \cdot ||x||$, for all $\alpha \in \mathbb{R}$

(iii) $||x + y|| \leq ||x|| + ||y||$ (triangular inequality)

The vector norms in $X=\mathbb{R}^n$ are imbedded into Maple *LinearAlgebra* package.

```
> restart :
> with(LinearAlgebra) : with(plots) :
> V := Vector(4, symbol = v)
```

$$V := \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix}$$

2.2. EXAMPLE. ℓ_1 ,ℓ_∞ and ℓ_2 vector norms

ℓ_1 – norm: $\|V\|_1 = \sum_{i=1}^{n} |v_i|$

```
> `||V||`[1] = VectorNorm(V, 1);
```

$$||V||_1 = |v_1| + |v_2| + |v_3| + |v_4|$$

So the norm $\|V\|_1$ is the sum of the absolute values of the coordinates.

ℓ_∞ – norm: $\|V\|_\infty = \max_{1 \le i \le n} |v_i|$

```
> `||V||`[infinity] = VectorNorm(V, infinity);
```

$$||V||_\infty = \max(|v_1|, |v_2|, |v_3|, |v_4|)$$

So the norm $\|V\|_\infty$ is the maximum value of the absolute values of the coordinates.

ℓ_2 – norm or Euclidean-norm: $\|V\|_2 = \sqrt{\sum_{i=1}^{n} |v_i|^2} = \sqrt{V.V}$

```
> `||V||`[2] = VectorNorm(V, 2);
> `||V||`[Euclidean] = VectorNorm(V, Euclidean);
```

$$||V||_2 = \sqrt{|v_1|^2 + |v_2|^2 + |v_3|^2 + |v_4|^2}$$

$$||V||_{Euclidean} = \sqrt{|v_1|^2 + |v_2|^2 + |v_3|^2 + |v_4|^2}$$

So the norm $\|V\|_2$ is the square root of the sum of the coordinate's squares.

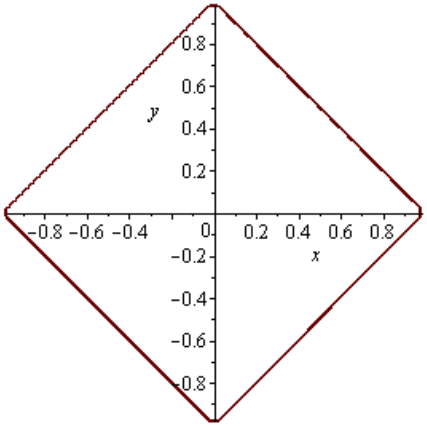
2.3. EXAMPLE. Give $\|x\| = 1$ points on the plane $\mathbb{R}^2$ in different norms!

2.3.1. $\|x\|_1 = 1$ is a square on its vertex with sides $\sqrt{2}$.

Let us give the implicit equation of the curve in 1-norm and draw this curve using the "implicitplot" from "plots" package!

```
> e1 := VectorNorm(Vector([x, y]), 1) = 1;
> r1 := implicitplot(e1, x = -1..1, y = -1..1, scaling = constrained, grid = [50, 50]) : r1
```

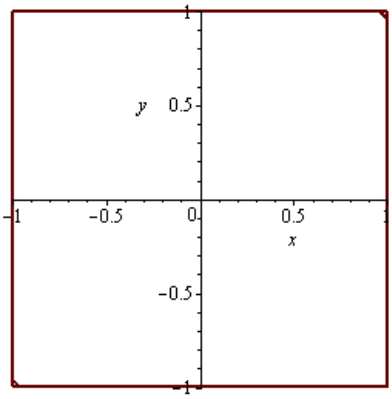
$$e1 := |x| + |y| = 1$$



2.3.2. $\|x\|_\infty = 1$ is a square with parallel sides of axis and side length a=2

```
> ev := VectorNorm(Vector([x, y]), ∞) = 1;
> rv := implicitplot(ev, x = -1..1, y = -1..1, scaling = constrained, grid = [50, 50]) : rv
```

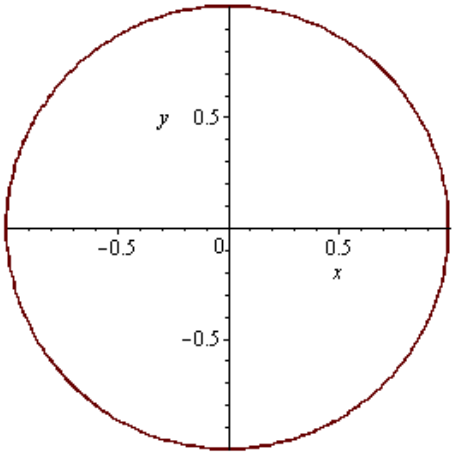
$$ev := \max(|x|, |y|) = 1$$



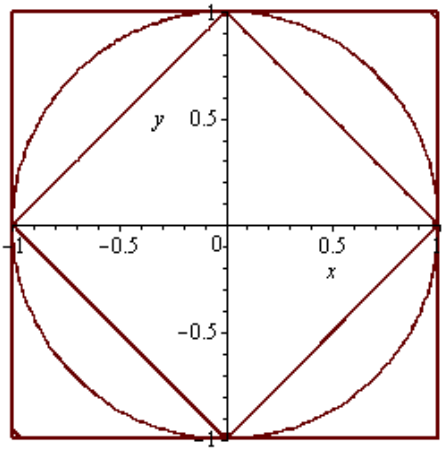
2.3.3. $\|x\|_2 = 1$ is a circle with centre of the origin and radius r=1

```
> e2 := VectorNorm(Vector([x, y]), 2) = 1
> r2 := implicitplot(e2, x = -1..1, y = -1..1, scaling = constrained, grid = [50, 50]) : r2
```

$$e2 := \sqrt{|x|^2 + |y|^2} = 1$$




```
> display(r1, r2, rv)
```



Using vector norms we can define the norm of a matrix A .

2.4. DEFINITION. Matrix norm

Let $A : \mathbb{R}^n \to \mathbb{R}^n$ is a squared matrix with sizes $n \times n$. Then the formula

$$\|A\| = \max_{x \neq 0} \frac{\|A \cdot x\|}{\|x\|} = \max_{\|x\| = 1} \|A \cdot x\|$$

gives the **norm of matrix** A , where the vector norm is one of giving above.

The norm of a matrix has the properties (i)-(iii) as in definition 2.1. and furthermore

(iv) $\|A \cdot x\| \leq \|A\| \cdot \|x\|$

(v) $\|A \cdot B\| \leq \|A\| \cdot \|B\|$, where A and B $n \times n$ matrices.

The norm of matrix A can be interpreted ‖A‖ as the greatest factor, with expanding the matrix A some unit vector x .

2.5. Example. Consider matrix $A = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$, which is mapping the plane $\mathbb{R}^2$ onto plane $\mathbb{R}^2$. Let us calculate the norm of matrix A in different matrix norms and give the unit vector to expand with maximum value!

> $A := \langle\langle 1, 0\rangle | \langle 1, 1\rangle\rangle$

$$A := \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$$

2.5.1. Let us calculate the 1-norm of matrix A!

> $MatrixNorm(A, 1)$

$$2$$

Choose a unit vector (in 1-norm), such that matrix A expand it by $2-$ times? Let us start from example 2.3.1 and get a point $\begin{bmatrix} x \\ y \end{bmatrix}$ on the lines of square and apply the mapping A .

> $transzformed := A.Vector([x, y]); image\_norm1 := VectorNorm(transzformed, 1)$

$$transzformed := \begin{bmatrix} x + y \\ y \end{bmatrix}$$

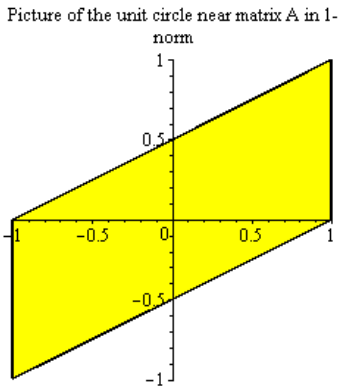
$$image\_norm1 := |x + y| + |y|$$

Since a linear transformation maps a line segments into line segments therefore we get the transformed figure by transforming the corners of the square and join with segments.

```
> paralelogramma1 := map(convert, [A.Vector([1, 0]), A.Vector([0, 1]), A.Vector([-1, 0]), A
    .Vector([0,-1])], list);
```

```
> polygonplot(paralelogramma1, color = yellow, title
    = "Picture of the unit circle near matrix A in 1-norm", scaling = constrained)
```

$$paralelogramma1 := [[1, 0], [1, 1], [-1, 0], [-1, -1]]$$



The point $\begin{bmatrix} 1 \\ 1 \end{bmatrix}$ is the farthest point of the parallelogram in 1-norm and $\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$.

2.5.2. Let us calculate the ∞-norm of matrix A!

```
> MatrixNorm(A, ∞)
```

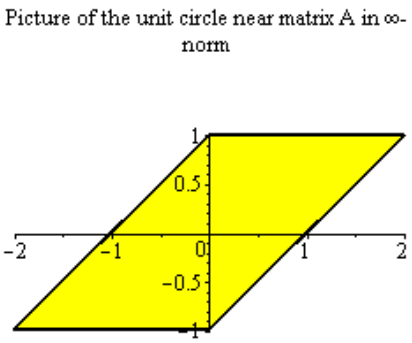
$$2$$

Choose a unit vector (in ∞ -norm), such that matrix A expand it by $2-$ times? Let us start from example 2.3.2 and get a point $\begin{bmatrix} x \\ y \end{bmatrix}$ on the lines of square and apply the mapping A

```
> paralelogramma2 := map(convert, [A.Vector([1, 1]), A.Vector([-1, 1]), A.Vector([-1,
    -1]), A.Vector([1,-1])], list)
```

```
> polygonplot(paralelogramma2, color = yellow, title
    = "Picture of the unit circle near matrix A in∞-norm", scaling = constrained)
```

$$paralelogramma2 := [[2, 1], [0, 1], [-2, -1], [0, -1]]$$



The point $\begin{bmatrix} 2 \\ 1 \end{bmatrix}$ is the farthest point of the parallelogram in ∞-norm and $\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$

2.5.3. Let us calculate the 2-norm of matrix A!

```
> MatrixNorm(A, 2); simplify(%);
```

$$\sqrt{\frac{1}{2}\sqrt{5} + \frac{3}{2}}$$

$$\frac{1}{2} + \frac{1}{2}\sqrt{5}$$

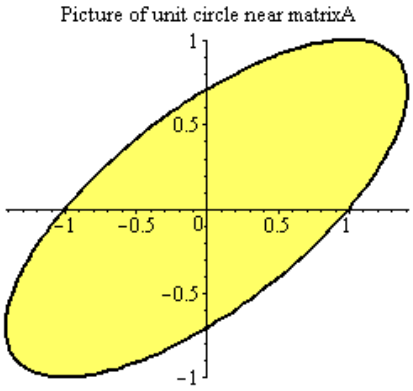
The matrix 2-norm is the well know ratio of the golden section $\Phi = \frac{1+\sqrt{5}}{2} \approx 1.618033$.

Choose a unit vector (in 2 -norm), such that matrix A expand it by $\Phi = \frac{1+\sqrt{5}}{2}$ -times? Let us start from example 2.3.3 and get a point $\begin{bmatrix} x \\ y \end{bmatrix}$ on the curve of circle and apply the mapping A

The parameterization of the unit circle $\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \cos(t) \\ \sin(t) \end{bmatrix}$ where $(0 \le t \le 2\pi)$ and apply the mapping A.

```
> picture := A.Vector([cos(t), sin(t)]); ellipse := op(convert(%, list)) :
>
  drawe1 := plot([ellipse, t = 0..2·π], color = black, thickness = 2) :
  drawe2 := plot([ellipse, t = 0..2·π], filled = true, color = yellow, title
      = "Picture of unit circle near matrixA") :
  display(drawe2, drawe1)
```

$$picture := \begin{bmatrix} \cos(t) + \sin(t) \\ \sin(t) \end{bmatrix}$$

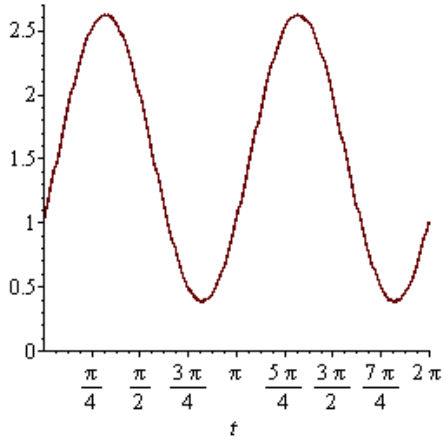


Form the 2-norm of the point of ellipse and its square power. Determine the maximum value of this function f on interval $[0, 2\cdot\pi]$!

```
> ellipsenorm := simplify(VectorNorm(picture, 2)) assuming t :: real;
  f := unapply(ellipsenorm^2, t);
>
> plot(f(t), t = 0..2·π, 0..2.7)
```

$$ellipsenorm := \sqrt{-\cos(t)^2 + 2\cos(t)\sin(t) + 2}$$

$$f := t \to -\cos(t)^2 + 2\cos(t)\sin(t) + 2$$



The maximum value of f is attained on interval $0 < t < \frac{\pi}{2}$. Determine the roots of the derivative function!

> $f'(t) = 0; \% + 2 \cdot (\sin(t)^2 - \cos(t)^2);$
> $subs([2 \cdot \sin(t) \cdot \cos(t) = \sin(2 \cdot t), \cos(t)^2 = \cos(2 \cdot t) + \sin(t)^2], \%); \frac{\%}{\cos(2 \cdot t)}$

> $T := solve(\%) + \frac{\pi}{2};$

$$2\cos(t)\sin(t) - 2\sin(t)^2 + 2\cos(t)^2 = 0$$

$$2\cos(t)\sin(t) = 2\sin(t)^2 - 2\cos(t)^2$$

$$\sin(2t) = -2\cos(2t)$$

$$\frac{\sin(2t)}{\cos(2t)} = -2$$

$$T := -\frac{1}{2}\arctan(2) + \frac{1}{2}\pi$$

We get $T = \frac{\pi}{2} - \frac{\arctan(2)}{2}$ an angle and determine the 2-norm $\sqrt{f(T)}$!

> $f(T); combine(\%); maximum = \sqrt{\%}$

$$-\sin\left(\frac{1}{2}\arctan(2)\right)^2 + 2\sin\left(\frac{1}{2}\arctan(2)\right)\cos\left(\frac{1}{2}\arctan(2)\right) + 2$$

$$\frac{1}{2}\sqrt{5} + \frac{3}{2}$$

$$maximum = \frac{1}{2} + \frac{1}{2}\sqrt{5}$$

We give the unit vector U, for which the $A.U$ vector norm attain its maximum value at $\Phi = \frac{1 + \sqrt{5}}{2}$.

> $U := Vector([\cos(T), \sin(T)]); A.U; combine(VectorNorm(\%, 2))$

$$U := \begin{bmatrix} \sin\left(\frac{1}{2}\arctan(2)\right) \\ \cos\left(\frac{1}{2}\arctan(2)\right) \end{bmatrix}$$

$$\begin{bmatrix} \sin\left(\frac{1}{2}\arctan(2)\right) + \cos\left(\frac{1}{2}\arctan(2)\right) \\ \cos\left(\frac{1}{2}\arctan(2)\right) \end{bmatrix}$$

$$\frac{1}{2} + \frac{1}{2}\sqrt{5}$$

So $\begin{bmatrix} \sin\left(\frac{1}{2}\arctan(2)\right) \\ \cos\left(\frac{1}{2}\arctan(2)\right) \end{bmatrix}$ is a unit vector in 2-norm and the image of this vector under matrix A is a vector with 2-norm is the maximal Φ value. Give this vector with square root signs!

> $z := \frac{\sqrt{5}-1}{2}$; $U = simplify\left(\frac{1}{\sqrt{1+z^2}}\right) \cdot Vector([z, 1])$; $simplify(A.rhs(\%))$;

$norma = simplify(VectorNorm(\%, 2))$; $\left(rhs(\%) = \frac{\sqrt{5}+1}{2}\right) = is\left(rhs(\%) = \frac{\sqrt{5}+1}{2}\right)$

$$z := \frac{1}{2}\sqrt{5} - \frac{1}{2}$$

$$\begin{bmatrix} \sin\left(\frac{1}{2}\arctan(2)\right) \\ \cos\left(\frac{1}{2}\arctan(2)\right) \end{bmatrix} = \begin{bmatrix} \frac{2\left(\frac{1}{2}\sqrt{5} - \frac{1}{2}\right)}{\sqrt{10-2\sqrt{5}}} \\ \frac{2}{\sqrt{10-2\sqrt{5}}} \end{bmatrix}$$

$$\begin{bmatrix} \frac{\sqrt{5}+1}{\sqrt{10-2\sqrt{5}}} \\ \frac{2}{\sqrt{10-2\sqrt{5}}} \end{bmatrix}$$

$$norma = -\frac{2\sqrt{5}}{-5+\sqrt{5}}$$

$$\left(-\frac{2\sqrt{5}}{-5+\sqrt{5}} = \frac{1}{2} + \frac{1}{2}\sqrt{5}\right) = true$$

So $x = \begin{bmatrix} \sin\left(\frac{1}{2}\arctan(2)\right) \\ \cos\left(\frac{1}{2}\arctan(2)\right) \end{bmatrix} = \begin{bmatrix} \frac{2\left(\frac{1}{2}\sqrt{5} - \frac{1}{2}\right)}{\sqrt{10-2\sqrt{5}}} \\ \frac{2}{\sqrt{10-2\sqrt{5}}} \end{bmatrix}$ is the unit vector in 2-norm, which image near matrix A and its norm $\|A.x\|_2 = \Phi = \frac{1+\sqrt{5}}{2}$.

2.6. Example. Matrix norms with elements of the matrix

2.6.1 Determine the 1-norm of a 2×2 size matrix B by symbolically!

```
> B := Matrix(2, 2, symbol = b);
> `||B||`[1] = MatrixNorm(B, 1);
```

$$B := \begin{bmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \end{bmatrix}$$

$$||B||_1 = \max\left(|b_{1,1}| + |b_{2,1}|, |b_{1,2}| + |b_{2,2}|\right)$$

So the 1-norm of $n \times m$ size $A = \left(a_{1..n,1}, a_{1..n,2}, \ldots, a_{1..n,m}\right)$ matrix is the maximum value of the 1-norm of the $a_{1..n,1}, a_{1..n,2}, \ldots, a_{1..n,m}$ column vectors

$$\|A\|_1 = \max_{1 \le j \le m}\left(|a_{1,j}| + |_{2,j}| + \ldots + |a_{n,j}|\right) = \max_{1 \le j \le m}\left(\|a_{1..n,j}\|_1\right)$$

2.6.2. Determine the ∞-norm of a 2×2 size matrix B by symbolically!

```
> `||B||`[∞] = MatrixNorm(B, infinity);
```

$$||B||_\infty = \max\left(|b_{1,1}| + |b_{1,2}|, |b_{2,1}| + |b_{2,2}|\right)$$

So the ∞-norm of a 2x2 matrix is maximum value of the 1-norm of the row vectors.
Similarly an $n \times m$ sized $A = \left(a^T_{1,1..m}, a^T_{2,1..m}, \ldots, a^T_{n,1..m}\right)$ matrix ∞-norm is the maximum value of the 1-norm of the $a^T_{1,1..m}, a^T_{2,1..m}, \ldots, a^T_{n,1..m}$ row vectors

$$\|A\|_\infty = \max_{1 \le i \le n}\left(|a_{i,1}| + |a_{i,2}| + \ldots + |a_{i,m}|\right) = \max_{1 \le i \le n}\left(\|a_{i,1..m}\|_1\right)$$

The matrix 2-norm is a little bit complicated! See it in exercise 2.8.

```
> MatrixNorm(B, 2, conjugate = false);
```

$$\left(\max\left(\left|RootOf\left(\_Z^2 + \left(-b_{1,1}^2 - b_{1,2}^2 - b_{2,1}^2 - b_{2,2}^2\right)\_Z + b_{1,1}^2 b_{2,2}^2 - 2\,b_{1,1}\,b_{1,2}\,b_{2,1}\,b_{2,2} + b_{1,2}^2 b_{2,1}^2, index = 1\right)\right|, \left|RootOf\left(\_Z^2 + \left(-b_{1,1}^2 - b_{1,2}^2 - b_{2,1}^2 - b_{2,2}^2\right)\_Z + b_{1,1}^2 b_{2,2}^2 - 2\,b_{1,1}\,b_{1,2}\,b_{2,1}\,b_{2,2} + b_{1,2}^2 b_{2,1}^2, index = 2\right)\right|\right)\right)^{1/2}$$

EXERCISES

2.7. Let us show the norm of vectors ℓ_1 ,ℓ_∞ és ℓ_2 in example 2.2. are fulfil the requirements of the norm in definition 2.1.!

2.8. Show that the 2- vector norm induced a matrix norm which is given by the formula $\|A\|_2 = \sqrt{\lambda_{\max}(A^T \cdot A)}$, where $\lambda_{\max}$ is denoted the greatest eigenvalue of the matrix and A^T denotes the transpose matrix of matrix A !

2.9. Prove, that the matrix 2-norm $A = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$ in example 2.5.3. is equivalent with determining the maximum value of function $f(x,y) = \sqrt{(x+y)^2 + y^2}$ with respect to $x^2 + y^2 = 1$ *condition* . Use the following Maple commands.

```
> with(Optimization) :
> Maximize(sqrt((x + y)^2 + y^2), {x^2 + y^2 = 1})
```

$$[1.6180339887532899, [x = 0.52573111137559, y = 0.85065080881404]]$$

2.10. Show that any two vector norms are equivalent, i.e exist positive constants $0 < c_1 \le c_2$ such that

$$c_1 \cdot \|x\|_1 \le \|x\|_2 \le c_2 \cdot \|x\|_2$$

fulfils.

3. Condition number of systems of linear equations and matrices.

We will write linear equations as

(3.1) $$A \cdot x = b$$

where A is a non-singular n × n matrix, b ∈ R^n is given vector, and

$$\tilde{x} = A^{-1} \cdot b \in R^n$$

the exact solution have to be found.

Throughout this point x will denote a potential solution and $\{x^k\}_{k \geq 0}$ the sequence of iterates. We will denote the i-th component of a vector x by x_i and the i-th component of x^k by x_i^k. We will rarely need to refer to individual components of vectors.

3.1. DEFINITION. Residual vector
The vector

$$r = b - A \cdot x$$

is called the **residual** vector for linear equations (3.1).

Most iterative methods terminate when the residual

$$r^k = b - A \cdot x^k$$

is sufficiently small. One termination criterion is

$$\frac{\|r^{(k)}\|}{\|b\|} < \varepsilon$$

3.2. DEFINITION. Error vector
The difference vector

$$e = x - \tilde{x}$$

is called the **error** vector for linear equations (3.1) between approximate solution x and exact solution $\tilde{x}$.

Try to estimate relative changes $\frac{\|r\|}{\|r^0\|}$ of the residual with the relative changes $\frac{\|e\|}{\|e^0\|}$ of the error!

Since

$$r = b - A \cdot x = A \cdot \tilde{x} - A \cdot x = A \cdot (\tilde{x} - x) = -A \cdot e.$$

we have

$$\|e\| = \|A^{-1} \cdot A \cdot e\| \leq \|A^{-1}\| \cdot \|A \cdot e\| = \|A^{-1}\| \cdot \|r\| .$$

On the other hand

$$\|r^0\| = \|A \cdot e^0\| \leq \|A\| \cdot \|e^0\|.$$

Hence

$$\frac{\|e\|}{\|e^0\|} \leq \frac{\|A^{-1}\| \cdot \|r\|}{\|A\|^{-1} \cdot \|r^0\|} = \|A^{-1}\| \cdot \|A\| \cdot \frac{\|r\|}{\|r^0\|}$$

The upper bound between the relative norms $\kappa(A) = \|A\| \cdot \|A^{-1}\|$ is called the condition number of

system or matrix!

3.3. DEFINITION. Condition number of matrix

Let $A : \mathbb{R}^n \to \mathbb{R}^n$ is a squared invertible matrix. Then the number

$$\kappa(A) = \|A\| \cdot \|A^{-1}\|$$

is called the **condition number** matrix A.

Calculation of the condition number is imbedded into Maple using different matrix norms. The default norm is the ∞ norm!

```
> restart :
> with(LinearAlgebra) : with(plots) :
>
```

```
A := RandomMatrix(2, 2);
κ('A', ∞) = ConditionNumber(A, ∞);
'||A||_∞ · ||A^(-1)||_∞' = MatrixNorm(A) · MatrixNorm(A^(-1));
κ('A', ∞) = evalf(rhs(%))
```

$$A := \begin{bmatrix} 44 & -31 \\ 92 & 67 \end{bmatrix}$$

$$\kappa(A, \infty) = \frac{2703}{725}$$

$$Norm(A, \infty)\ Norm\left(\frac{1}{A}, \infty\right) = \frac{2703}{725}$$

$$\kappa(A, \infty) = 3.7283$$

```
>
```

```
κ('A', 2) = ConditionNumber(A, 2);
'||A||_2 · ||A^(-1)||_2' = MatrixNorm(A, 2) · MatrixNorm(A^(-1), 2);
κ('A', 2) = evalf(rhs(%))
```

$$\kappa(A, 2) = \sqrt{\frac{317}{1345600} + \frac{3}{1345600}\sqrt{5185}}\ \sqrt{7925 + 75\sqrt{5185}}$$

$$Norm(A, 2)\ Norm\left(\frac{1}{A}, 2\right) = \sqrt{\frac{317}{1345600} + \frac{3}{1345600}\sqrt{5185}}\ \sqrt{7925 + 75\sqrt{5185}}$$

$$\kappa(A, 2) = 2.2975$$

3.4. Lemma

The condition number of a matrix A is $\kappa(A) = \|A\| \cdot \|A^{-1}\| \geq 1$ always greater than or equal to 1.

Proof

Since

$$I_n = A \cdot A^{-1}$$

where I n is the n×n identity matrix. From the matrix norm property 2.4. (v) follows

$$1 = \|I\| = \|A \cdot A^{-1}\| \leq \|A\| \cdot \|A^{-1}\| = \kappa(A). \qquad Q.e.d.$$

3.5. Lemma

The condition number of matrix A fulfils $\kappa(A) = \|A\| \cdot \|A^{-1}\| \geq \dfrac{\max\limits_i |\lambda_i|}{\min\limits_i |\lambda_i|}$ inequality, where λ_i denotes the eigenvalue of matrix A .

Proof

Since the matrix A is invertible, therefore $\lambda_i \neq 0$. Denote by v^i is the eigenvector belongs to eigenvalue λ_i of matrix A

$$A \cdot v^i = \lambda_i \cdot v^i$$

Taking the norm

$$|\lambda_i| \cdot \|v^i\| = \|A \cdot v^i\| \leq \|A\| \cdot \|v^i\|$$

$$|\lambda_i| \leq \|A\|.$$

Since this inequality is fulfilled for all index i therefore

$$\varrho(A) = \max_i |\lambda_i| \leq \|A\| \ .$$

where the positive number $\varrho(A) = \max\limits_i |\lambda_i|$ is called the **spectral radius** of matrix A .

Furthermore the eigenvalues of the inverse matrix A^{-1} are $\dfrac{1}{\lambda_i}$ and the corresponding eigenvectors are the same x_i

$$A^{-1} \cdot v^i = \frac{1}{\lambda_i} \cdot v^i \ .$$

From this equality

$$\frac{1}{|\lambda_i|} \cdot \|v^i\| = \|A^{-1} \cdot v^i\| \leq \|A^{-1}\| \cdot \|v^i\|$$

$$\frac{1}{|\lambda_i|} \leq \|A^{-1}\|.$$

Since this inequality fulfills for all index i therefore

$$\max_i \frac{1}{|\lambda_i|} = \frac{1}{\min\limits_i |\lambda_i|} \leq \|A^{-1}\|$$

Take the multiplication of the two obtained inequalities

$$\frac{\max\limits_i |\lambda_i|}{\min\limits_i |\lambda_i|} \leq \|A\| \cdot \|A^{-1}\| = \kappa(A).$$

Q.e.d.

Remark. Since the ratio $\dfrac{\max\limits_i |\lambda_i|}{\min\limits_i |\lambda_i|}$ is independent of the matrix norm, therefore it is better characterize the condition of matrix A . The matrix is called ill conditioned when this number is "great". This means that there is a "small" eigenvalue together with "large" eigenvalue.

3.6. EXAMPLE

Let us illustrate lemmas 3.4. and 3.5 by using random **symmetric** matrices!

```
> A := RandomMatrix(3, 3, shape = symmetric);
  eigenvalues = evalf(Eigenvalues(A)); abs(rhs(%));
  ratio = max(%)/min(%);
  κ = evalf(ConditionNumber(A), 2)
```

$$A := \begin{bmatrix} 29 & 99 & 69 \\ 99 & 8 & 27 \\ 69 & 27 & -4 \end{bmatrix}$$

$$eigenvalues = \begin{bmatrix} 149.9166868 - 1.\,10^{-8}\,I \\ -91.57353002 - 2.86410161610^{-8}\,I \\ -25.34315682 + 4.06410161610^{-8}\,I \end{bmatrix}$$

$$\begin{bmatrix} 149.9166868 \\ 91.57353002 \\ 25.34315682 \end{bmatrix}$$

$$ratio = 5.9155$$

$$\kappa = 10.0000$$

3.7. DEFINITION. A-norm

Let the square matrix A is symmetric and positive definite (SPD). Then the number

$$\|x\|_A = \sqrt{x^T \cdot A \cdot x}$$

is called $A-$ **norm.**

3.8. Example

The solution may change dramatically when the matrix is ill conditioned. i.e. $\kappa(A)$ is a large number!

```
> restart : with(LinearAlgebra) : system1 := [5·x − 331·y = 3.5, 6·x − 397·y = 5.2]
```

$$system1 := [5x - 331y = 3.5, 6x - 397y = 5.2]$$

```
> A1, b1 := GenerateMatrix(system1, [x, y]); solution1 = LinearSolve(A1, b1)
```

$$A1, b1 := \begin{bmatrix} 5 & -331 \\ 6 & -397 \end{bmatrix}, \begin{bmatrix} 3.5 \\ 5.2 \end{bmatrix}$$

$$solution1 = \begin{bmatrix} 331.699999999962 \\ 4.99999999999943 \end{bmatrix}$$

```
> Determinant(A1);
> evalf(Eigenvalues(A1)); %[2]/%[1]; ConditionNumber(A1, ∞)
```

$$1$$

$$\begin{bmatrix} -0.0025510 \\ -391.9974490 \end{bmatrix}$$

$$1.5366\,10^{5}$$

$$293384$$

Change only the element of matrix A on the upper left corner from 5 to 4.9 !

> $system2 := [\,4.9\cdot x - 331\cdot y = 3.5,\ 6\cdot x - 397\cdot y = 5.2]$

$$system2 := [4.9\,x - 331\,y = 3.5,\ 6\,x - 397\,y = 5.2]$$

> $A2, b2 := GenerateMatrix(system2, [x, y]);\ solution2 = LinearSolve(A2, b2)$

$$A2, b2 := \begin{bmatrix} 4.9 & -331 \\ 6 & -397 \end{bmatrix}, \begin{bmatrix} 3.5 \\ 5.2 \end{bmatrix}$$

$$solution2 = \begin{bmatrix} 8.14987714987713 \\ 0.110073710073710 \end{bmatrix}$$

> $Determinant(A2);$

> $evalf(Eigenvalues(A2));\ \frac{\%[2]}{\%[1]};\ ConditionNumber(A2, \infty)$

$$40.7000$$

$$\begin{bmatrix} -0.103827544399337 + 0.\,I \\ -391.996172455601 + 0.\,I \end{bmatrix}$$

$$3775.45452628564 - 0.\,I$$

$$7208.4521$$

The solution changes dramatically!

4. Cayley-Hamilton theorem

The Cayley-Hamilton theorem states that any square matrix $A \in \mathbb{C}^{n\times n}$ satisfies its own characteristic equation, where the characteristic polynomial

$$p(z) : z \to det(z\cdot I - A)$$

$$p(z) = (z - \lambda_1)\cdot\cdot\cdot(z - \lambda_n) = z^n + c_1\cdot z^{n-1} + \cdot\cdot\cdot + c_{n-1}\cdot z + c_n$$

satisfies

(4.1) $$p(A) = A^n + c_1\cdot A^{n-1} + \cdot\cdot\cdot + c_{n-1}\cdot A + c_n\cdot I = 0$$

A direct consequence of the Cayley-Hamilton theorem is that – provided $A \in \mathbb{R}^{n\times n}$ is non-singular – the inverse A^{-1} can be expressed as follows (after by multiplying (4.1) by A^{-1})

$$A^{-1} = -\frac{1}{c_n}\cdot A^{n-1} - \frac{c_1}{c_n}\cdot A^{n-2} - \cdot\cdot\cdot - \frac{c_{n-1}}{c_n}\cdot I$$

where $c_n = (-1)^n\cdot det(A) \neq 0$ by assumption.

This representation of A^{-1} in terms of a matrix polynomial of degree $(n-1)$ 1 (with coefficients depending on the spectrum of A) may be viewed as a motivation for the class of Krylov subspace methods which we will introduce later on.

Remark 4.1.

For diagonalizable matrices A, the Cayley-Hamilton Theorem is easy to prove: $A = S\cdot D\cdot S^{-1}$ where D is a diagonal matrix with diagonal entries $D_{i,i} = \lambda_i$. For any polynomial p , we compute

$$p(A) = S\cdot p(D)\cdot S^{-1}.$$

A calculation reveals

$$p(\mathrm{D}) = Diag\left(p\left(\lambda_1\right), p\left(\lambda_2\right),\ldots, p\left(\lambda_n\right)\right),$$

which implies the assertion of the Cayley-Hamilton Theorem because of $p\left(\lambda_i\right) = 0$ for all $i = 1, 2,\ldots, n$.

4.1. EXAMPLE

> $restart : with(LinearAlgebra) :$

> $M := Matrix([[1, 2], [3, 4]])$

$$M := \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

> $p := CharacteristicPolynomial(M, \lambda)$

$$p := \lambda^2 - 5\,\lambda - 2$$

> $M^2, -5 \cdot M, -2 \cdot IdentityMatrix(2); add(i, i = \%)$

$$\begin{bmatrix} 7 & 10 \\ 15 & 22 \end{bmatrix}, \begin{bmatrix} -5 & -10 \\ -15 & -20 \end{bmatrix}, \begin{bmatrix} -2 & 0 \\ 0 & -2 \end{bmatrix}$$

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

Since $p(M) = M^2 - 5\,M - 2 \cdot I = 0$ therefore the inverse can be get by a polynomial of matrix M

$$M^2 - 5\,M = 2 \cdot I$$

$$M - 5 \cdot I = 2 \cdot M^{-1}$$

$$M^{-1} = \frac{1}{2} \cdot M - \frac{5}{2} \cdot I$$

> $M^{-1} = \frac{1}{2} \cdot M - \frac{5}{2} \cdot IdentityMatrix(2)$

$$\begin{bmatrix} -2 & 1 \\ \frac{3}{2} & -\frac{1}{2} \end{bmatrix} = \begin{bmatrix} -2 & 1 \\ \frac{3}{2} & -\frac{1}{2} \end{bmatrix}$$

This M is not a normal matrix, but diagonalizable.

> $M^{+} .M - M.M^{+}$

$$\begin{bmatrix} 5 & 3 \\ 3 & -5 \end{bmatrix}$$

> $DD := JordanForm(M);\ S := JordanForm(M, output = 'Q'); simplify\left(S^{-1}.M.S\right)$

$$DD := \begin{bmatrix} \frac{5}{2} - \frac{1}{2}\sqrt{33} & 0 \\ 0 & \frac{5}{2} + \frac{1}{2}\sqrt{33} \end{bmatrix}$$

$$S := \begin{bmatrix} \frac{1}{2} + \frac{1}{22}\sqrt{33} & \frac{1}{66}\left(-3 + \sqrt{33}\right)\sqrt{33} \\ -\frac{1}{11}\sqrt{33} & \frac{1}{11}\sqrt{33} \end{bmatrix}$$

$$\begin{bmatrix} \frac{5}{2} - \frac{1}{2}\sqrt{33} & 0 \\ 0 & \frac{5}{2} + \frac{1}{2}\sqrt{33} \end{bmatrix}$$

> $M = simplify\left(S.DD.S^{-1}\right)$

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

2. Discretization of differential equations

LARGE-SCALE LINEAR SYSTEMS OF EQUATIONS
Computer Science Engineering Msc
Theoretical and Practical lesson

Partial Differential Equations (PDEs) constitute by far the biggest source of sparse matrix problems. The typical way to solve such equations is to discretize them, i.e., to approximate them by equations that involve a finite number of unknowns. The matrix problems that arise from this discretization are generally large and sparse, i.e., they have very few nonzero entries.

There are several different ways to discretize a Partial Differential Equation. The simplest method uses finite difference approximations for the partial differential operators. The Finite Element Method replaces the original function by a function which has some degree of smoothness over the global domain, but which is piecewise polynomial on simple cells, such as small triangles or rectangles. This method is probably the most general and well understood discretization technique available.

1. Finite Difference Methods

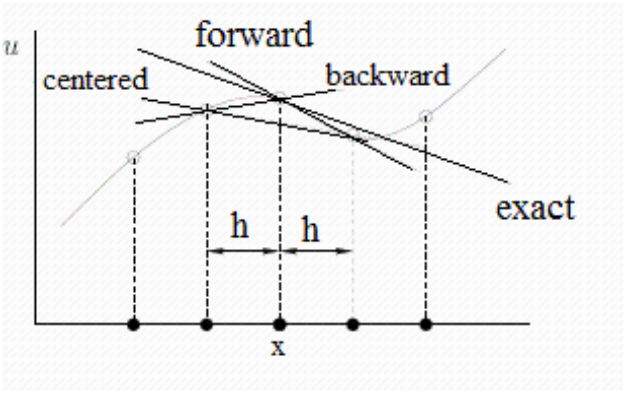
This section gives an overview of finite difference discretization techniques.
The simplest way to approximate the first derivative of a function u at the point x is via the formula

1.1.Forward difference $\mathrm{D}_+ u(x) = \dfrac{u(x+h) - u(x)}{h}$

1.2. Backward difference $\mathrm{D}_- u(x) = \dfrac{u(x) - u(x-h)}{h}$

1.3. Centred difference approximation

$$\mathrm{D}_0 u(x) = \frac{u(x+h) - u(x-h)}{2h} = \frac{1}{2}\left(\mathrm{D}_+ u(x) + \mathrm{D}_- u(x)\right)$$



Taylor-polynomial together with error term at point x and $(x+h)$

$$\mathrm{u}\ (x+h) = u(x) + h\cdot\frac{\mathrm{d}u}{\mathrm{d}x} + \frac{h^2}{2}\cdot\frac{\mathrm{d}^2u}{\mathrm{d}x^2} + \frac{h^3}{6}\cdot\frac{\mathrm{d}^3u}{\mathrm{d}x^3} + \frac{h^4}{24}\cdot\frac{\mathrm{d}^4u}{\mathrm{d}x^4}(\xi_+) \tag{1.4}$$

From this follows the **forward** approximation of the derivative with error term

$$\frac{\mathrm{d}u(x)}{\mathrm{d}x} = \frac{u(x+h) - u(x)}{h} - \frac{h}{2}\frac{\mathrm{d}^2u}{\mathrm{d}x^2} + \mathrm{O}(h^2).$$

In similar way we obtain the third order Taylor-polynomial at point x and $(x-h)$

(1.5) $$u\ (x-h) = u(x) - h\cdot\frac{\mathrm{d}u}{\mathrm{d}x} + \frac{h^2}{2}\cdot\frac{\mathrm{d}^2u}{\mathrm{d}x^2} - \frac{h^3}{6}\cdot\frac{\mathrm{d}^3u}{\mathrm{d}x^3} + \frac{h^4}{24}\cdot\frac{\mathrm{d}^4u}{\mathrm{d}x^4}(\xi_-)$$

From this follows the **backward** approximation of the derivative with error term

$$\frac{\mathrm{d}u(x)}{\mathrm{d}x} = \frac{u(x) - u(x-h)}{h} - \frac{h}{2}\frac{\mathrm{d}^2u}{\mathrm{d}x^2} + \mathrm{O}(h^2).$$

Take the arithmetic mean of the forward and backward differences

$$u'(x) = \frac{u(x+h) - u(x-h)}{2\cdot h} - \frac{u^{(3)}(\xi)}{3!}\cdot h^2$$

We have got the c**entred difference approximation** of derivative with error term

Adding equations (1.4) and (1.5) we get the centred difference approximation of the second derivative with error term

(1.6) $$\frac{\mathrm{d}^2u(x)}{\mathrm{d}x^2} = \frac{u(x+h) - 2\,u(x) + u(x-h)}{h^2} - \frac{h^2}{12}\cdot\frac{\mathrm{d}^4u}{\mathrm{d}x^4}(\xi).$$

The dependence of this derivative on the values of u at the points involved in the approximation is often represented by a "stencil" or "molecule,"

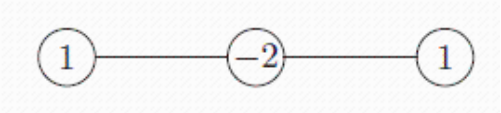


Figure 1. The three-point stencil for the centred difference approximation to the second order derivative.

2. Finite Differences for 1-D Problems: theory and numerical experiments

Consider the one-dimensional equation

(2.1) $$-u''(x) = f(x), x \in [0, 1]$$

with homogeneous boundaries

$$u(0) = u(1) = 0$$

The interval [0,1] can be discretized uniformly by taking the n + 2 points

(2.2) $$x_i = i\cdot h,\ i = 0, 1,..., (n+1),$$

where $h = \frac{1}{n+1}$ step size.

Because of the Dirichlet-boundary conditions are known, i.e. the values $u(x_0) = 0$ and $u(x_{n+1}) = 0$. At every other point, an approximation u_i is sought for the exact solution $u(x_i)$.

If the centred difference approximation (1.6) is used, then by the equation (2.1) expressed at the point $x = x_i$, the unknowns u_i, u_{i-1}, u_{i+1} satisfy the relation in which $f_i = f(x_i)$

$$-u_{i-1} + 2\cdot u_i - u_{i+1} = h^2\cdot f_i \qquad (i = 1, 2, 3,..., n).$$

Thus, we have a linear system obtained is of the form

$$A\cdot x = b$$

where the matrix

$$A = \begin{bmatrix} 2 & -1 & & & & \\ -1 & 2 & -1 & & 0 & \\ & -1 & 2 & -1 & & \\ & & -1 & 2 & -1 & \\ 0 & & & -1 & 2 & -1 \\ & & & & -1 & 2 \end{bmatrix}$$

a special **tridiagonal** or band matrix. Non-zero elements are on the main diagonal and parallel over and above it. The right hand side vector

$$b = h^2 \cdot \begin{bmatrix} f_1 \\ f_2 \\ . \\ . \\ f_{n-1} \\ f_n \end{bmatrix}$$

Construct matrix A using Maple LinearAlgebra package $A = BandMatrix([-1, 2, -1], 1, n)$ procedure!

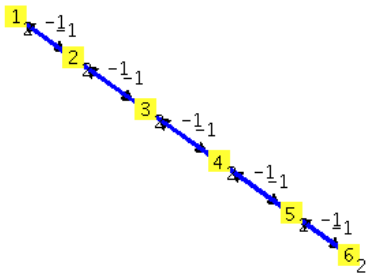
```
> restart : with(LinearAlgebra) :
> A := LinearAlgebra:-BandMatrix([-1, 2, -1], 1, 6);
```

$$A := \begin{bmatrix} 2 & -1 & 0 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & 0 & -1 & 2 \end{bmatrix}$$

Draw the directed graph G such that their adjacency matrix is A!

```
> with(GraphTheory) :
> G := Graph(A, directed);
> DrawGraph(G, style = spring)
```

G := Graph 1: a directed weighted graph with 6 vertices and 16 arc(s)



2.1. EXAMPLE. Approximation of exact solution with given function $f(x)$

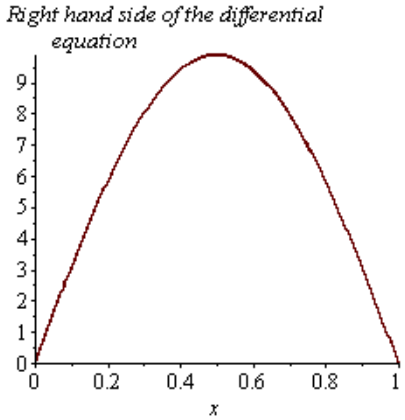
Take the function $f(x) = \pi^2 \cdot \sin(\pi \cdot x)$ on the right hand side in differential equation (2.1) and in

growing n.

```
> restart : with(LinearAlgebra) : with(plots) :
> f := x→π^2·sin(π·x) :f'(x) = f(x);
> a, b := 0, 1;
> plot(f(x), x = a ..b, title = `Right hand side of the differential equation`  )
```

$$f(x) = \pi^2 \sin(\pi x)$$

$$a, b := 0, 1$$



Give **analytic solution** of the problem using Maple *dsolve* procedure

```
> de := -d^2/dx^2 u(x) = f(x);
> pe := u(a) = 0, u(b) = 0
```

$$de := -\left(\frac{d^2}{dx^2} u(x)\right) = \pi^2 \sin(\pi x)$$

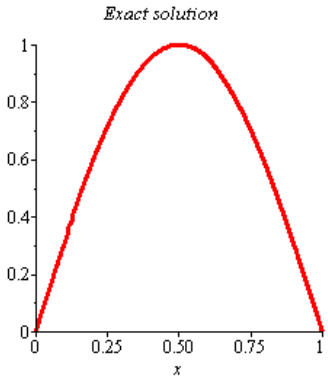
$$pe := u(0) = 0, u(1) = 0$$

```
> sol := dsolve({de, pe}); odetest(sol, {de, pe})
```

$$sol := u(x) = \sin(\pi x)$$

$$\{0\}$$

```
> exact := plot(rhs(sol), x = a ..b, color = red, thickness = 3, xtickmarks = 5, scaling
      = constrained, title = `Exact solution`) : exact
```



Divide the $[0, 1]$ interval 4 equal subintervals with $n = 5$. Now we get 3 linear equations with 3 unknown of variables

> $n := 3; h := \frac{(b-a)}{n+1}; X := Vector([seq(i \cdot h, i = 1..n)])$

$$n := 3$$

$$h := \frac{1}{4}$$

$$X := \begin{bmatrix} \frac{1}{4} \\ \frac{1}{2} \\ \frac{3}{4} \end{bmatrix}$$

> $i := 'i' : d2 := \frac{d^2}{dx^2} u(x) = \frac{u_{i-1} - 2 \cdot u_i + u_{i+1}}{h^2}$

$$d2 := \frac{d^2}{dx^2} u(x) = 16\, u_{i-1} - 32\, u_i + 16\, u_{i+1}$$

> $i := 'i' : discrete := eval\left(h^2 \cdot de, \left[d2, x = x_i\right]\right)$

$$discrete := -u_{i-1} + 2\, u_i - u_{i+1} = \frac{1}{16} \pi^2 \sin\left(\pi\, x_i\right)$$

Substitute value $x_i = i \cdot h$ at the right hand side!

> $sys := NULL : \textbf{for } i \textbf{ from } 1 \textbf{ to } n \textbf{ do} \quad sys := sys, eval\left(discrete, x_i = X_i\right) \textbf{ end do}:$

> $sys := [\, sys \,]$

$$sys := \left[-u_0 + 2\, u_1 - u_2 = \frac{1}{32} \pi^2 \sqrt{2}, -u_1 + 2\, u_2 - u_3 = \frac{1}{16} \pi^2, -u_2 + 2\, u_3 - u_4 = \frac{1}{32} \pi^2 \sqrt{2} \right]$$

From the boundary values $u_0 = u_{n+1} = 0$. Therefore *variables* $u_1, u_2, \ldots, u_n$ remain only unknown values!

> $sys := eval\left(sys, \left[u_0 = 0, u_{n+1} = 0\right]\right)$

$$sys := \left[2\, u_1 - u_2 = \frac{1}{32} \pi^2 \sqrt{2}, -u_1 + 2\, u_2 - u_3 = \frac{1}{16} \pi^2, -u_2 + 2\, u_3 = \frac{1}{32} \pi^2 \sqrt{2} \right]$$

So we have n linear equations with n unknown variables. Generate matrix A and right hand side using *LinearAlgebra* package *GenerateMatrix* procedure and solve with *LinearSolve* procedure!

> $var := \left[\, seq\left(u_i, i = 1..n\right) \,\right];$

> $(A, B) := evalf\left(GenerateMatrix\left(sys, var\right)\right);$

$$var := \left[u_1, u_2, u_3\right]$$

$$A, B := \begin{bmatrix} 2.0000 & -1.0000 & 0.0000 \\ -1.0000 & 2.0000 & -1.0000 \\ 0.0000 & -1.0000 & 2.0000 \end{bmatrix}, \begin{bmatrix} 0.4362 \\ 0.6170 \\ 0.4362 \end{bmatrix}$$

Calculate the elapsed time to solve the linear equations!

> $start := time(\,) : U := LinearSolve\left(A, B\right); printf\left("Ellapsed\ time : \%f\backslash n", time(\,) - start\right);$

$$U := \begin{bmatrix} 0.7447 \\ 1.0532 \\ 0.7447 \end{bmatrix}$$

```
Ellapsed time : 0.016000
```

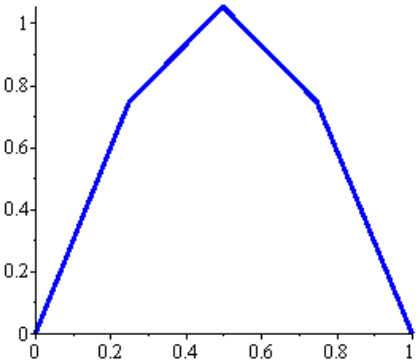
Put the zero boundary values back into the solution vector on first and the last positions!

> $X1, U1 := Vector([\mathrm{a}, convert(X, list), \mathrm{b}]), Vector([rhs(pe[1]), convert(U, list), rhs(pe[2])])$

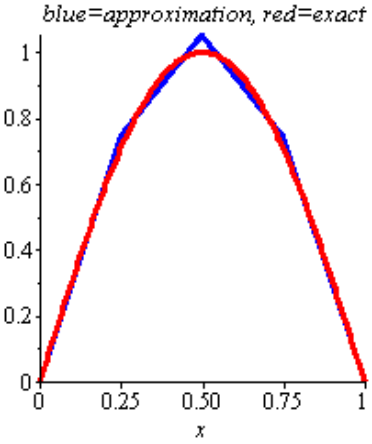
$$X1, U1 := \begin{bmatrix} 0 \\ \frac{1}{4} \\ \frac{1}{2} \\ \frac{3}{4} \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 0.7447 \\ 1.0532 \\ 0.7447 \\ 0 \end{bmatrix}$$

Plot the approximation of the solution

> $approx := plot(X1, U1, color = blue, thickness = 3) : approx$



> $plots[display]([approx, exact], title = `blue=approximation, red=exact`)$



2.2. EXERCISE

Taking the number n up to $n = 100, 200, 300, 1000$ values! Watching the required time and the

accuracy of the approximation of the system of linear equations to solve!

2.3. EXERCISE

Change the right hand side of the differential equation (2.1) to function $f(x)=\pi^2\cdot\sin(k\cdot\pi\cdot x)$ with values $k=2,3,4$! Run the program from beginning and watch the approximation accuracy!

2.4. EXERCISE

Show that the eigenvalues of band matrix $A=BandMatrix([-1,2,-1],1,n)$ obtained in exercise 2.1. are

$$\lambda_j=2\cdot\left(1-\cos\left(\frac{\pi}{n+1}\cdot j\right)\right)$$

positive. Therefore symmetric matrix *A is positive definite* . Furthermore the eigenvector which belongs to eigenvalue λ_i is

$$v_j^T=\left(\sin\left(\frac{1\cdot j\cdot\pi}{n+1}\right),\sin\left(\frac{2\cdot j\cdot\pi}{n+1}\right),\sin\left(\frac{3\cdot j\cdot\pi}{n+1}\right),\ldots,\sin\left(\frac{n\cdot j\cdot\pi}{n+1}\right)\right)$$

for all $j=1,2,3,\ldots,n$.

2.5. EXERCISE (homework)

Solve the boundary problem

$$-u''(x)=f(x),u(0)=u(1)=0$$

with function $f(x)$

```
> f := x→piecewise(0.25 ≤ x and x ≤ 0.75, 1, 0):
> f := x→x·(1 − x)
```

$$f:=x\to x\,(1-x)$$

```
> f := x→x^2·(1 − x)
```

$$f:=x\to x^2\,(1-x)$$

3. Finite Differences for 2-D Problems: theory

Similarly as in the previous case, consider this simple problem for function $u(x,y)$ with two variables

(3.1) $$-\Delta u(x,y)=-\left(\frac{\partial^2}{\partial x^2}u(x,y)+\frac{\partial^2}{\partial y^2}u(x,y)\right)=f(x,y)$$

Poisson-equation within unit square $T=\{(x,y)\mid 0<x<1,0<y<1\}=[0,1]^2$ and on its boundary $(x,y)\in\Gamma=\partial T$

$$u(x,y)=0$$

Dirichlet-type boundary conditions**.**

Both intervals can be discretized uniformly by taking $(n+1)x(n+1)$ pieces of small squares. We get $(n+2)^2$ number points in the square and on boundary. Looking for $u_{i,j}=u(x_i,y_j)$ approximate solution in grid points

$$x_i=i\cdot h,y_j=j\cdot h,\quad\left(i,j=0,1,2,\ldots,n,h=\frac{1}{n+1}\right).$$

Approximate second order partial derivatives

(3.2) $$\frac{\partial^2 u(x,y)}{\partial x^2}\approx\frac{u(x+h,y)-2\cdot u(x,y)+u(x-h,y)}{h^2}$$

(3.3) $$\frac{\partial^2 u(x,y)}{\partial y^2} \approx \frac{u(x,y+h) - 2\cdot u(x,y) + u(x,y-h)}{h^2}$$

with centred differences.

Summing the second order partial derivatives we get numerical approximation of **Laplace-operator**

$$\Delta u(x_i, y_j) \approx \frac{u_{i-1,j} + u_{i,j-1} + u_{i+1,j} + u_{i,j+1} - 4\,u_{i,j}}{h^2}$$

at the inside grid points (x_i, y_j) $(i,j = 1, 2,..., n)$. At the grid point (i,j) and the its neighbours points are shown in picture below

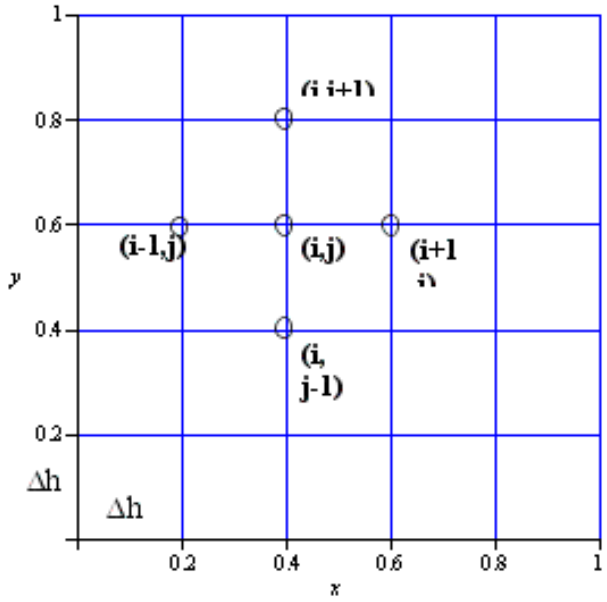


Figure 3.1. Using the points for calculation of Laplace-operator

Substitute (3.2) and (3.3) into Poisson-equation (3.1) at interior points $(x,y) = (x_i, y_j)$ for all $i, j = 1, 2,..., n$

$$4\,u_{i,j} - u_{i-1,j} - u_{i,j-1} - u_{i+1,j} - u_{i,j+1} = h^2 \cdot f_{i,j}$$

where $f_{i,j} = f(x_i, y_j)$.

Now the stencil of coefficients

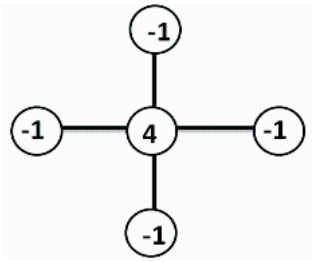


Figure 3.2. Stencil of the Laplace-operator

Using Dirichlet-type boundary conditions the values $u_{i,j}$ at boundary are zeros

$$u_{i,0} = u_{0,j} = u_{n+1,j} = u_{i,n+1} = 0,\ (i,j = 0, 1, 2,..., n+1).$$

Choose $n = 4$ for visualization, i.e. divide the $x-$ and $y-$ interval 5 equal pieces of the unit

square $[0, 1] \times [0, 1]$!
Creating the linear equations with two embedded *seq* procedure for indeces i and j ! Note that the order of the evaluation points from the bottom upwards and from left to right as follows:

$$(x_1, y_1), (x_2, y_1), (x_3, y_1), (x_4, y_1), (x_1, y_2), \ldots, (x_1, y_4), (x_2, y_4), (x_3, y_4), (x_4, y_4).$$

The internal order of the points shown below 1 to 16!

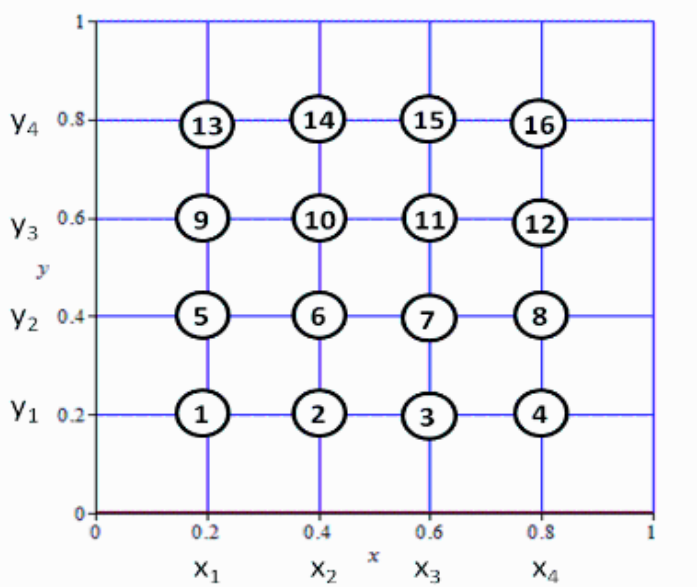


Figure 3.3. Order of evaluation process and the name of variables

```
> restart :
> i :='i':j :='j': e := 4 u[i,j]-u[i-1,j]- u[i,j-1]-u[i+1,j]-u[i,j+1] = h^2·f[i,j]
> n := 4;
> equations := [seq(seq(e, i = 1..n), j = 1..n)]; `Number of equations` = n·n
```

$$e := 4 u_{i,j} - u_{i-1,j} - u_{i,j-1} - u_{i+1,j} - u_{i,j+1} = h^2 f_{i,j}$$

$$n := 4$$

$$equations := \left[4 u_{1,1} - u_{0,1} - u_{1,0} - u_{2,1} - u_{1,2} = h^2 f_{1,1}, 4 u_{2,1} - u_{1,1} - u_{2,0} - u_{3,1} - u_{2,2} = h^2 f_{2,1}, 4 u_{3,1} - u_{2,1} - u_{3,0} - u_{4,1} - u_{3,2} = h^2 f_{3,1}, 4 u_{4,1} - u_{3,1} - u_{4,0} - u_{5,1} - u_{4,2} = h^2 f_{4,1}, 4 u_{1,2} - u_{0,2} - u_{1,1} - u_{2,2} - u_{1,3} = h^2 f_{1,2}, 4 u_{2,2} - u_{1,2} - u_{2,1} - u_{3,2} - u_{2,3} = h^2 f_{2,2}, 4 u_{3,2} - u_{2,2} - u_{3,1} - u_{4,2} - u_{3,3} = h^2 f_{3,2}, 4 u_{4,2} - u_{3,2} - u_{4,1} - u_{5,2} - u_{4,3} = h^2 f_{4,2}, 4 u_{1,3} - u_{0,3} - u_{1,2} - u_{2,3} - u_{1,4} = h^2 f_{1,3}, 4 u_{2,3} - u_{1,3} - u_{2,2} - u_{3,3} - u_{2,4} = h^2 f_{2,3}, 4 u_{3,3} - u_{2,3} - u_{3,2} - u_{4,3} - u_{3,4} = h^2 f_{3,3}, 4 u_{4,3} - u_{3,3} - u_{4,2} - u_{5,3} - u_{4,4} = h^2 f_{4,3}, 4 u_{1,4} - u_{0,4} - u_{1,3} - u_{2,4} - u_{1,5} = h^2 f_{1,4}, 4 u_{2,4} - u_{1,4} - u_{2,3} - u_{3,4} - u_{2,5} = h^2 f_{2,4}, 4 u_{3,4} - u_{2,4} - u_{3,3} - u_{4,4} - u_{3,5} = h^2 f_{3,4}, 4 u_{4,4} - u_{3,4} - u_{4,3} - u_{5,4} - u_{4,5} = h^2 f_{4,4}\right]$$

$$Number\ of\ equations = 16$$

The variables on edge are zeros because of the Dirichlet boundary!

> $boundary := \left[seq\left(u_{0,j} = 0, j = 1..(n+1)\right), seq\left(u_{i,0} = 0, i = 1..(n+1)\right), seq\left(u_{i,n+1} = 0, i = 1..(n+1)\right), seq\left(u_{n+1,j} = 0, j = 1..(n+1)\right)\right]$

$$boundary := \left[u_{0,1} = 0, u_{0,2} = 0, u_{0,3} = 0, u_{0,4} = 0, u_{0,5} = 0, u_{1,0} = 0, u_{2,0} = 0, u_{3,0} = 0, u_{4,0} = 0, u_{5,0} = 0, u_{1,5} = 0, u_{2,5} = 0, u_{3,5} = 0, u_{4,5} = 0, u_{5,5} = 0, u_{5,1} = 0, u_{5,2} = 0, u_{5,3} = 0, u_{5,4} = 0, u_{5,5} = 0\right]$$

These values are substituted into the equations. So we have also n^2 number of unknown $u_{i,j}$ variables!

> $substituted := eval(equations, boundary);$

$$substituted := \left[4u_{1,1} - u_{2,1} - u_{1,2} = h^2 f_{1,1}, 4u_{2,1} - u_{1,1} - u_{3,1} - u_{2,2} = h^2 f_{2,1}, 4u_{3,1} - u_{2,1} - u_{4,1} - u_{3,2} = h^2 f_{3,1}, 4u_{4,1} - u_{3,1} - u_{4,2} = h^2 f_{4,1}, 4u_{1,2} - u_{1,1} - u_{2,2} - u_{1,3} = h^2 f_{1,2}, 4u_{2,2} - u_{1,2} - u_{2,1} - u_{3,2} - u_{2,3} = h^2 f_{2,2}, 4u_{3,2} - u_{2,2} - u_{3,1} - u_{4,2} - u_{3,3} = h^2 f_{3,2}, 4u_{4,2} - u_{3,2} - u_{4,1} - u_{4,3} = h^2 f_{4,2}, 4u_{1,3} - u_{1,2} - u_{2,3} - u_{1,4} = h^2 f_{1,3}, 4u_{2,3} - u_{1,3} - u_{2,2} - u_{3,3} - u_{2,4} = h^2 f_{2,3}, 4u_{3,3} - u_{2,3} - u_{3,2} - u_{4,3} - u_{3,4} = h^2 f_{3,3}, 4u_{4,3} - u_{3,3} - u_{4,2} - u_{4,4} = h^2 f_{4,3}, 4u_{1,4} - u_{1,3} - u_{2,4} = h^2 f_{1,4}, 4u_{2,4} - u_{1,4} - u_{2,3} - u_{3,4} = h^2 f_{2,4}, 4u_{3,4} - u_{2,4} - u_{3,3} - u_{4,4} = h^2 f_{3,4}, 4u_{4,4} - u_{3,4} - u_{4,3} = h^2 f_{4,4}\right]$$

The matrix $x = \left[u_{i,j}\right]_{i=1..n}^{j=1..n}$ of unknown variables have to give in a vector, so we get $A \cdot x = b$ linear equations with variables in vector x. The list is sorted by the finite difference should be the same order of evaluation!

> $variables := \left[seq\left(seq\left(u_{i,j}, i = 1..n\right), j = 1..n\right)\right];$
> `Number of variables`= $numelems(variables)$

$$variables := \left[u_{1,1}, u_{2,1}, u_{3,1}, u_{4,1}, u_{1,2}, u_{2,2}, u_{3,2}, u_{4,2}, u_{1,3}, u_{2,3}, u_{3,3}, u_{4,3}, u_{1,4}, u_{2,4}, u_{3,4}, u_{4,4}\right]$$

$$Number\ of\ variables = 16$$

> $A, b := LinearAlgebra:-GenerateMatrix(substituted, variables)$

$$A, b := \left[\begin{array}{c} 16\ x\ 16\ \ Matrix \\ Data\ Type:\ anything \\ Storage:\ rectangular \\ Order:\ Fortran\_order \end{array}\right], \left[\begin{array}{c} 1\ ..\ 16\ \ Vector_{column} \\ Data\ Type:\ anything \\ Storage:\ rectangular \\ Order:\ Fortran\_order \end{array}\right]$$

The matrix A size is $n^2 \times n^2$ and the density map

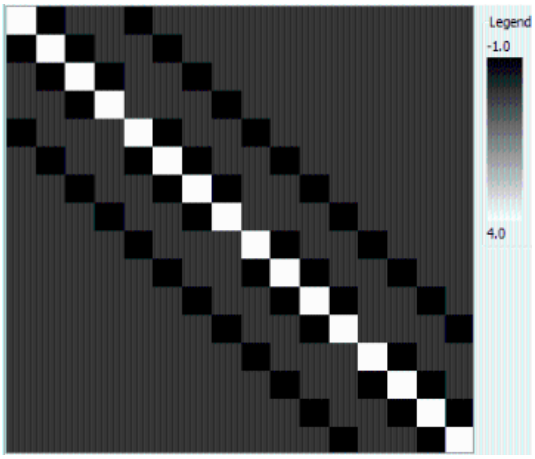


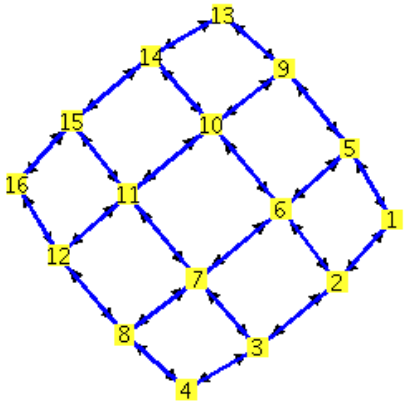
Figure 3.4. Density map of matrix A

Choose the "Image" button!

Let us view directed graph, which adjacency matrix is A !

```
> with(GraphTheory):
> G := Graph(A, directed);
> DrawGraph(G, style = spring)
```

G := Graph 1: a directed weighted graph with 16 vertices and 64 arc(s)



>

3.1. EXERCISE

Show that if we change the equation or system of equations or list $u_{i,j}$ unknown sequence (one only), the matrix shape (in the place of non-zero elements) is changed.

3.2. EXERCISE

Show that the matrix of the 2D Laplace operator can be built from the following blocks

$$A = \begin{bmatrix} B & -I & 0 & 0 \\ -I & B & -I & 0 \\ 0 & -I & B & -I \\ 0 & 0 & -I & B \end{bmatrix},$$

where each blocks represent a matrix with size n × n, I is the unit matrix and

$$B = \begin{bmatrix} 4 & -1 & 0 & 0 \\ -1 & 4 & -1 & 0 \\ 0 & -1 & 4 & -1 \\ 0 & 0 & -1 & 4 \end{bmatrix}$$

tridiagonal matrix, which can be given by $BandMatrix([-1, 4, -1], 1, n, storage = rectangular)$ command.

So the matrix A is block tridiagonal shape.

3.3. EXERCISES

Show that the eigenvalues and eigenvectors of band matrix

$$B = BandMatrix([-1, 4, -1], 1, n, storage = rectangular)$$

which is given in exercise 3.2.

$$\lambda_j(B) = 4 - 2\cdot\cos(j\cdot\pi\cdot h), j = 1, 2, \ldots, n;\ h = \frac{1}{n+1}$$

and using the notation $\Theta_j = \frac{j\cdot\pi}{n+1}$ és $h = \frac{1}{n+1}$

$$q_j = \sqrt{h}\cdot Vector\left(\left[\sin(\Theta_j), \sin(2\cdot\Theta_j), \ldots, \sin(n\cdot\Theta_j)\right]\right).$$

3.4. EXERCISE

The matrix A of 2D Laplace operator have n^2 number of eigenvalues of the following

$$\mu_{i,j}(A) = 4\cdot\left(\sin^2\left(\frac{i\cdot\pi\cdot h}{2}\right) + \sin^2\left(\frac{j\cdot\pi\cdot h}{2}\right)\right).\quad\left(i, j \in \{1, 2, \ldots, n\}, h = \frac{1}{n+1}\right)$$

and the adequate eigenvector $V_{i,j}$ are

$$V_{i,j} = Vector\left(\left[\sin(i\cdot i'\cdot\pi\cdot h)\cdot\sin(j\cdot j'\cdot\pi\cdot h), i' = 1..n, j' = 1..n\right]\right).\quad\left(h = \frac{1}{n+1}\right)$$

So the matrix A is symmetric positive definite (SPD) matrix.

4. Finite Differences for 2-D Problems: numerical experiments

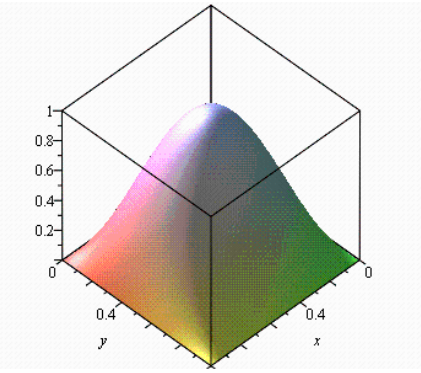
Find the function $f(x, y)$ for Poisson-equation (3.1) in section 3 when the solution is given $u(x, y) = \sin(\pi\cdot x)\cdot\sin(\pi\cdot y)$!

```
> restart
> U := unapply(sin(π·x)·sin(π·y), x, y)
```

$$U := (x, y) \rightarrow \sin(\pi x)\ \sin(\pi y)$$

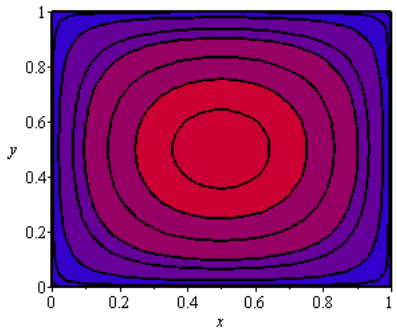
```
> f := -(diff(U(x, y), x, x) + diff(U(x, y), y, y))
> plot3d(sin(Pi·x)·sin(Pi·y), x = 0..1, y = 0..1)
```

$$f := 2\sin(\pi x)\ \pi^2\ \sin(\pi y)$$



Solution function $u(x,y)$ can be visualizing also with contourplot!

> $draw\_exact := plots[contourplot](U(x,y), x = 0..1, y = 0..1, filledregions = true, contours = [0, 0.01, 0.07, 0.2, 0.3, 0.5, 0.7, 0.9, 1], linestyle = 1, coloring = ["blue", "red"]):$
> $draw\_exact$



Make sure that the Dirichlet-type boundary conditions are met, that is, the solution is zero at the border

$$u(x.t)\Big|_{(x,y) \in \Gamma} = 0 .$$

> $U(0,y), U(1,y), U(x,0), U(x,1)$

$$0, 0, 0, 0$$

The coordinates of the grid points put into the vectors X and Y !

> $n := 3; h := \frac{1}{n+1}; X := [seq(i \cdot h, i = 0..(n+1))]; Y := X;$

$$n := 3$$

$$h := \frac{1}{4}$$

$$X := \left[0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 1\right]$$

$$Y := \left[0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 1\right]$$

We give only one stencil!

> $e := 4u_{i,j} - u_{i-1,j} - u_{i,j-1} - u_{i+1,j} - u_{i,j+1} = h^2 \cdot eval(f, [x = x_i, y = y_j])$

$$e := 4u_{i,j} - u_{i-1,j} - u_{i,j-1} - u_{i+1,j} - u_{i,j+1} = \frac{1}{8}\sin(\pi x_i)\,\pi^2 \sin(\pi y_j)$$

The boundary conditions are zeros!

> $Dirichlet := \{seq(u_{i,0} = 0, i = 0..(n+1)), seq(u_{0,j} = 0, j = 0..(n+1)), seq(u_{i,n+1} = 0, i = 0..(n+1)), seq(u_{n+1,j} = 0, j = 1..n+1)\}$

$$Dirichlet := \{u_{0,0} = 0, u_{0,1} = 0, u_{0,2} = 0, u_{0,3} = 0, u_{0,4} = 0, u_{1,0} = 0, u_{1,4} = 0, u_{2,0} = 0, u_{2,4} = 0, u_{3,0} = 0, u_{3,4} = 0, u_{4,0} = 0, u_{4,1} = 0, u_{4,2} = 0, u_{4,3} = 0, u_{4,4} = 0\}$$

System of linear equations created by varying indices i and so the value of function f.

> $equations := [seq(seq(eval(e, [x_i = X_{i+1}, y_j = Y_{j+1}]), j = 1..n), i = 1..n)]$;
$`Number of equations` = nops(equations)$

$$equations := \left[4u_{1,1} - u_{0,1} - u_{1,0} - u_{2,1} - u_{1,2} = \frac{1}{16}\pi^2, 4u_{1,2} - u_{0,2} - u_{1,1} - u_{2,2} - u_{1,3} = \frac{1}{16}\sqrt{2}\,\pi^2, 4u_{1,3} - u_{0,3} - u_{1,2} - u_{2,3} - u_{1,4} = \frac{1}{16}\pi^2, 4u_{2,1} - u_{1,1} - u_{2,0} - u_{3,1} - u_{2,2} = \frac{1}{16}\sqrt{2}\,\pi^2, 4u_{2,2} - u_{1,2} - u_{2,1} - u_{3,2} - u_{2,3} = \frac{1}{8}\pi^2, 4u_{2,3} - u_{1,3} - u_{2,2} - u_{3,3} - u_{2,4} = \frac{1}{16}\sqrt{2}\,\pi^2, 4u_{3,1} - u_{2,1} - u_{3,0} - u_{4,1} - u_{3,2} = \frac{1}{16}\pi^2, 4u_{3,2} - u_{2,2} - u_{3,1} - u_{4,2} - u_{3,3} = \frac{1}{16}\sqrt{2}\,\pi^2, 4u_{3,3} - u_{2,3} - u_{3,2} - u_{4,3} - u_{3,4} = \frac{1}{16}\pi^2\right]$$

$$Number\ of\ equations = 9$$

Evaluate the dummy variables on the edges!

> sys := $evalf(eval(equations, Dirichlet))$

$$sys := [4.0000u_{1,1} - 1.0000u_{2,1} - 1.0000u_{1,2} = 0.6170, 4.0000u_{1,2} - 1.0000u_{1,1} - 1.0000u_{2,2} - 1.0000u_{1,3} = 0.8725, 4.0000u_{1,3} - 1.0000u_{1,2} - 1.0000u_{2,3} = 0.6170, 4.0000u_{2,1} - 1.0000u_{1,1} - 1.0000u_{3,1} - 1.0000u_{2,2} = 0.8725, 4.0000u_{2,2} - 1.0000u_{1,2} - 1.0000u_{2,1} - 1.0000u_{3,2} - 1.0000u_{2,3} = 1.2340, 4.0000u_{2,3} - 1.0000u_{1,3} - 1.0000u_{2,2} - 1.0000u_{3,3} = 0.8725, 4.0000u_{3,1} - 1.0000u_{2,1} - 1.0000u_{3,2} = 0.6170, 4.0000u_{3,2} - 1.0000u_{2,2} - 1.0000u_{3,1} - 1.0000u_{3,3} = 0.8725, 4.0000u_{3,3} - 1.0000u_{2,3} - 1.0000u_{3,2} = 0.6170]$$

The matrix-shaped array of the remaining unknown variables should be listed in a vector! Take it into lexicographical order (see fig.3.3)!

> $variables := [seq(seq(u_{i,j}, j = 1..n), i = 1..n)]$; $`Number of variables` = nops(variables)$

$$variables := [u_{1,1}, u_{1,2}, u_{1,3}, u_{2,1}, u_{2,2}, u_{2,3}, u_{3,1}, u_{3,2}, u_{3,3}]$$

$$Number\ of\ variables = 9$$

> $LinearAlgebra:-GenerateMatrix(sys, variables)$

$$\begin{bmatrix} 4.0000 & -1.0000 & 0 & -1.0000 & 0 & 0 & 0 & 0 & 0 \\ -1.0000 & 4.0000 & -1.0000 & 0 & -1.0000 & 0 & 0 & 0 & 0 \\ 0 & -1.0000 & 4.0000 & 0 & 0 & -1.0000 & 0 & 0 & 0 \\ -1.0000 & 0 & 0 & 4.0000 & -1.0000 & 0 & -1.0000 & 0 & 0 \\ 0 & -1.0000 & 0 & -1.0000 & 4.0000 & -1.0000 & 0 & -1.0000 & 0 \\ 0 & 0 & -1.0000 & 0 & -1.0000 & 4.0000 & 0 & 0 & -1.0000 \\ 0 & 0 & 0 & -1.0000 & 0 & 0 & 4.0000 & -1.0000 & 0 \\ 0 & 0 & 0 & 0 & -1.0000 & 0 & -1.0000 & 4.0000 & -1.0000 \\ 0 & 0 & 0 & 0 & 0 & -1.0000 & 0 & -1.0000 & 4.0000 \end{bmatrix}, \begin{bmatrix} 0.6170 \\ 0.8725 \\ 0.6170 \\ 0.8725 \\ 1.2340 \\ 0.8725 \\ 0.6170 \\ 0.8725 \\ 0.6170 \end{bmatrix}$$

> $start\_time := time() : approximate := convert(op(solve(sys, variables)), set); stop\_time := time() : ellapsed\_time = stop\_time - start\_time$

$$approximate := \{u_{1,1} = 0.5266, u_{1,2} = 0.7448, u_{1,3} = 0.5266, u_{2,1} = 0.7448, u_{2,2} = 1.0530, u_{2,3} = 0.7448, u_{3,1} = 0.5266, u_{3,2} = 0.7448, u_{3,3} = 0.5266\}$$

$$ellapsed\_time = 0.0160$$

Put the zero values to the approximate solution at the boundaries!

> $total := approximate \textbf{ union } Dirichlet$

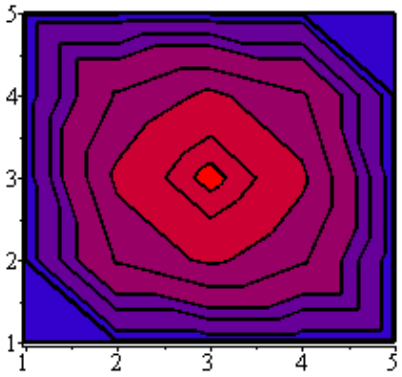
$$total := \{u_{0,0} = 0, u_{0,1} = 0, u_{0,2} = 0, u_{0,3} = 0, u_{0,4} = 0, u_{1,0} = 0, u_{1,1} = 0.5266, u_{1,2} = 0.7448, u_{1,3} = 0.5266, u_{1,4} = 0, u_{2,0} = 0, u_{2,1} = 0.7448, u_{2,2} = 1.0530, u_{2,3} = 0.7448, u_{2,4} = 0, u_{3,0} = 0, u_{3,1} = 0.5266, u_{3,2} = 0.7448, u_{3,3} = 0.5266, u_{3,4} = 0, u_{4,0} = 0, u_{4,1} = 0, u_{4,2} = 0, u_{4,3} = 0, u_{4,4} = 0\}$$

> $M := Matrix([seq([seq(eval(u_{i,j}, total), i = 0..n + 1)], j = 0..n + 1)])$

$$M := \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0.5266 & 0.7448 & 0.5266 & 0 \\ 0 & 0.7448 & 1.0530 & 0.7448 & 0 \\ 0 & 0.5266 & 0.7448 & 0.5266 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

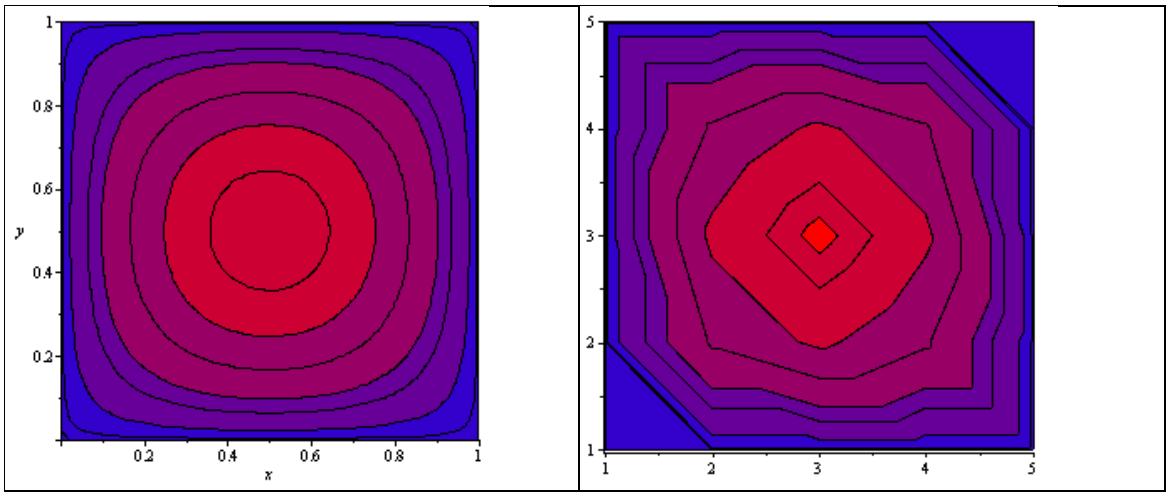
The “contourplot” command drawing the approximate solution

> $with(plots) : draw\_approximate := listcontplot(M, filledregions = true, contours = [0, 0.01, 0.07, 0.2, 0.3, 0.5, 0.7, 0.9, 1], coloring = ["blue", "red"], numpoints = 600) : draw\_approximate$



The combined figure of the exact solution and the approximate solution.

```
> figures := Matrix(1, 2, [[draw_exact, draw_approximate]]):
> display(figures, scaling = constrained)
```



4.1. EXERCISE

4.1.(a) Increasing the number n = 5, 6,... and notice how, increases the computational time required for solving a system of linear equations!

4.1.(b) Searching function $f(x, y)$ for the solution functions

$u(x, y) = \sin(k \cdot \pi \cdot x) \cdot \sin(m \cdot \pi \cdot x)$ $(m, k = 2, 3, ...)$ and finds numerical approximation

4.2. EXERCISE

Show that the following procedure create 2D Poisson-matrix with size $n^2 \times n^2$.

```
> MakeBlockTridiagonal := proc(n :: posint)
    local B, E, A, i:
    B := LinearAlgebra:-BandMatrix([-1, 4, -1], 1, n): E := Matrix(n, n, 1, shape
      = diagonal):

> A := Matrix(n^2, n^2):
> A[1..n, 1..2·n] := Matrix([B, -E]):
```

```
> i := 1 :
  while i ≤ n − 2 do
    A[i·n + 1 ..(i + 1)· n, (i − 1)·n + 1 ..(i + 2)· n] := Matrix([−E, B, −E]) :
    i := i + 1 :
  end do:
> A[(n − 1)· n + 1 ..n· n, (n − 2)· n + 1 ..n· n] := Matrix([−E, B]) :   return A : end proc:
> M := MakeBlockTridiagonal(3)
```

$$M := \begin{bmatrix} 4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 4 & 0 & 0 & -1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 4 & -1 & 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & -1 & 4 & -1 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 & -1 & 4 & 0 & 0 & -1 \\ 0 & 0 & 0 & -1 & 0 & 0 & 4 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & -1 & 4 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 4 \end{bmatrix}$$

3. Direct solutions of linear system of equations using compact $L \cdot U$ factorization of a square matrix A

LARGE-SCALE LINEAR SYSTEMS OF EQUATIONS
Computer Science Engineering Msc
Theoretical and Practical lesson

Remarkable variants of LU factorization (L means *lower triangular* and U means *upper triangular* matrix) are the **Crout - factorization** and **Doolittle - factorization**, and are known also as compact forms of the **Gauss - elimination** method.
These approaches require less intermediate results than the standard Gauss elimination method to generate the factorization of A.
Computing $A = L \cdot U$ factorization is formally equivalent to solving the following nonlinear system of n^2 equations

$$a_{i,j} = \sum_{r=1}^{\min(i,j)} l_{i,r} \cdot u_{r,j} \ , \ (i, j = 1, 2, \ldots, n)$$

the unknowns being the $n^2 + n$ coefficients of the triangular matrices L and U. If we arbitrarily set n coefficients to 1, for example the diagonal entries of L or U, we end up with the **Doolittle** and **Crout** methods, respectively, which provide an efficient way to solve system.

1. Formulate *Doolittle scheme*

Define matrix L as l**ower triangle matrix** with diagonal elements $l_{k,k} = 1$ using Maple symbolic capabilities!

```
> restart
```

Size of matrices is denoted by N

```
> N := 4;
> l := 'l': L := Matrix(N, N, shape = triangular[lower, unit], symbol = l) : L
```

$$N := 4$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ l_{2,1} & 1 & 0 & 0 \\ l_{3,1} & l_{3,2} & 1 & 0 \\ l_{4,1} & l_{4,2} & l_{4,3} & 1 \end{bmatrix}$$

Define matrix U as **upper triangle matrix**!

```
> u := 'u': U := Matrix(N, N, shape = triangular[upper], symbol = u) : U
```

$$\begin{bmatrix} u_{1,1} & u_{1,2} & u_{1,3} & u_{1,4} \\ 0 & u_{2,2} & u_{2,3} & u_{2,4} \\ 0 & 0 & u_{3,3} & u_{3,4} \\ 0 & 0 & 0 & u_{4,4} \end{bmatrix}$$

Computing the $L \cdot U$ multiplication!

```
> Matrix(N, N, symbol = a) = L.U
```

$$\begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} \\ a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} \end{bmatrix} = \Big[\Big[u_{1,1}, u_{1,2}, u_{1,3}, u_{1,4}\Big],$$
$$\Big[l_{2,1}\,u_{1,1}, l_{2,1}\,u_{1,2} + u_{2,2}, l_{2,1}\,u_{1,3} + u_{2,3}, l_{2,1}\,u_{1,4} + u_{2,4}\Big],$$
$$\Big[l_{3,1}\,u_{1,1}, l_{3,1}\,u_{1,2} + l_{3,2}\,u_{2,2}, l_{3,1}\,u_{1,3} + l_{3,2}\,u_{2,3} + u_{3,3}, l_{3,1}\,u_{1,4} + l_{3,2}\,u_{2,4} + u_{3,4}$$
$$\Big],$$
$$\Big[l_{4,1}\,u_{1,1}, l_{4,1}\,u_{1,2} + l_{4,2}\,u_{2,2}, l_{4,1}\,u_{1,3} + l_{4,2}\,u_{2,3} + l_{4,3}\,u_{3,3}, l_{4,1}\,u_{1,4} + l_{4,2}\,u_{2,4}$$
$$+ l_{4,3}\,u_{3,4} + u_{4,4}\Big]\Big]$$

From this follows the equalities

$$a_{kj} = \sum_{r=1}^{k-1} l_{kr} u_{rj} + \boxed{u_{kj}}, \qquad j = k, \ldots, n$$

for the **upper triangular part** of A.
Furthermore the equalties

$$a_{ik} = \sum_{r=1}^{k-1} l_{ir} u_{rk} + \boxed{l_{ik}}\, u_{kk}, \quad i = k+1, \ldots, n$$

for the **lower triangular part** of A.
These equations can be solved in a sequential way with respect to the boxed variables $u_{k,j}$ and $l_{i,k}$.
For the **Doolittle compact method** we thus obtain first the $k-$ th row of U and then the $k-$ th column of L, as follows:
for k = 1, . . . , n

$$u_{kj} = a_{kj} - \sum_{r=1}^{k-1} l_{kr} u_{rj} \qquad j = k, \ldots, n$$
$$l_{ik} = \frac{1}{u_{kk}} \left(a_{ik} - \sum_{r=1}^{k-1} l_{ir} u_{rk} \right) \quad i = k+1, \ldots, n.$$

Implement the LU factorization using Doolittle-method writing Maple **procedure** that perform so that the returning matrix U is an upper triangular matrix and L is a lower triangular matrix with diagonal *elements* 1 and $A = L \cdot U$.

```
> with(LinearAlgebra):
lu_Doolittle := proc(A)
  local B, n, i, j, k, r, L, U:
    # initialize B=A, L=diag(1,n) and U=0:
    n := Dimensions(A)[1]:
    B := Matrix(n, n): L := Matrix(n, n): U := Matrix(n, n):
    for i from 1 to n do L[i, i] := 1 end do:
    for i from 1 to n do
      for j from 1 to n do
        B[i, j] := A[i, j]
      end do:
    end do:
    # Doolitle - method of factorization
    for k from 1 to n do
      for j from k to n do
       U[k, j] := B[k, j]:
        for r from 1 to k - 1 do U[k, j] := U[k, j] - L[k, r]·U[r, j] end;
      end:
    for i from k + 1 to n do
        L[i, k] := B[i, k]:
        for r from 1 to k - 1 do L[i, k] := L[i, k] - L[i, r]·U[r, k] end;
        L[i, k] := L[i, k] / U[k, k]
      end;
    end;
  return (L, U):
end proc:
```

Try this procedure with positive definite $tridiag([-1, 2, -1], n)$ model matrix!

```
> A := BandMatrix([-1, 2, -1], 1, N)
```

$$A := \begin{bmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{bmatrix}$$

```
> L, U := lu_Doolittle(A)
```

$$L, U := \begin{bmatrix} 1 & 0 & 0 & 0 \\ -\frac{1}{2} & 1 & 0 & 0 \\ 0 & -\frac{2}{3} & 1 & 0 \\ 0 & 0 & -\frac{3}{4} & 1 \end{bmatrix}, \begin{bmatrix} 2 & -1 & 0 & 0 \\ 0 & \frac{3}{2} & -1 & 0 \\ 0 & 0 & \frac{4}{3} & -1 \\ 0 & 0 & 0 & \frac{5}{4} \end{bmatrix}$$

```
> L.U
```

$$\begin{bmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{bmatrix}$$

Use the *LUDecomposition* procedure which is imbedded into the *LinearAlgebra* package.

> $p, l, u := LUDecomposition(A)$

$$p, l, u := \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 & 0 \\ -\frac{1}{2} & 1 & 0 & 0 \\ 0 & -\frac{2}{3} & 1 & 0 \\ 0 & 0 & -\frac{3}{4} & 1 \end{bmatrix}, \begin{bmatrix} 2 & -1 & 0 & 0 \\ 0 & \frac{3}{2} & -1 & 0 \\ 0 & 0 & \frac{4}{3} & -1 \\ 0 & 0 & 0 & \frac{5}{4} \end{bmatrix}$$

You can see the result is the same!

It is worth to note that the tridiagonal structure of the matrices L and U remain the **same**. However the **inverse matrix** is filled entirely with non-zero elements. This situation is called **fill-in** phenomenon.

> A^{-1}

$$\begin{bmatrix} \frac{4}{5} & \frac{3}{5} & \frac{2}{5} & \frac{1}{5} \\ \frac{3}{5} & \frac{6}{5} & \frac{4}{5} & \frac{2}{5} \\ \frac{2}{5} & \frac{4}{5} & \frac{6}{5} & \frac{3}{5} \\ \frac{1}{5} & \frac{2}{5} & \frac{3}{5} & \frac{4}{5} \end{bmatrix}$$

2. Solution of triangular systems

Assume, that we have an $LU-decomposition$ for matrix A and we want to solve the system of linear equations

$$A \cdot x = b$$
$$L \cdot U \cdot x = b$$

First step is solving the system

$$L \cdot y = b$$

for vector variable y.

Second step is solving the system

$$U \cdot x = y$$

for vector variable x.

2.1. Solution of lower triangular systems: ***forward substitution***

> $restart:$

> $with(LinearAlgebra): n := 3$

$$n := 3$$

> $l := 'l': x := 'x': b := 'b':$
$L, X, B := Matrix(n, n, shape = triangular[lower], symbol = l), Vector(n, symbol = x),$
$Vector(n, symbol = b)$

$$L, X, B := \begin{bmatrix} l_{1,1} & 0 & 0 \\ l_{2,1} & l_{2,2} & 0 \\ l_{3,1} & l_{3,2} & l_{3,3} \end{bmatrix}, \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}, \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

> $forward := L.X = B$

$$forward := \begin{bmatrix} l_{1,1}\,x_1 \\ l_{2,1}\,x_1 + l_{2,2}\,x_2 \\ l_{3,1}\,x_1 + l_{3,2}\,x_2 + l_{3,3}\,x_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

> $lhs(forward)[1] = rhs(forward)[1]; isolate(\%, x_1)$

$$l_{1,1}\,x_1 = b_1$$

$$x_1 = \frac{b_1}{l_{1,1}}$$

> $lhs(forward)[2] = rhs(forward)[2]; isolate(\%, x_2)$

$$l_{2,1}\,x_1 + l_{2,2}\,x_2 = b_2$$

$$x_2 = \frac{-l_{2,1}\,x_1 + b_2}{l_{2,2}}$$

> $lhs(forward)[3] = rhs(forward)[3]; isolate(\%, x_3)$

$$l_{3,1}\,x_1 + l_{3,2}\,x_2 + l_{3,3}\,x_3 = b_3$$

$$x_3 = \frac{-l_{3,1}\,x_1 - l_{3,2}\,x_2 + b_3}{l_{3,3}}$$

In the case of a system $Lx = b$, with L being a non-singular lower triangular matrix of order n ($n \geq 2$), the method takes the form

$$x_1 = \frac{b_1}{l_{11}},$$
$$x_i = \frac{1}{l_{ii}} \left(b_i - \sum_{j=1}^{i-1} l_{ij} x_j \right), \quad i = 2, \ldots, n.$$

Implementing the **forward substitution** algorithm writing Maple procedure!

```
forward_subs := proc(L, b)
  local n, i, j, x :
  n := Dimensions(L)[1] :
  x := Vector(n, 0) :
  # forward - substitution method
  x[1] := b[1]/L[1, 1] :
  for i from 2 to n do
    x[i] := b[i] :
    for j from 1 to (i - 1) do
      x[i] := x[i] - x[j]·L[i, j]
    end do:
    x[i] := x[i]/L[i, i] :
  end do:
  return(x) :
end proc:
```

Try the *forward_subs* procedure with matrix $A = tridiag([-1, 2, -1], n)$ *positive definite* matrix!

```
> A := BandMatrix([-1, 2, -1], 1, n);
> p, l, u := LUDecomposition(A);
> b := l.Vector(n, 1)
```

$$A := \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix}$$

$$p, l, u := \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 \\ -\frac{1}{2} & 1 & 0 \\ 0 & -\frac{2}{3} & 1 \end{bmatrix}, \begin{bmatrix} 2 & -1 & 0 \\ 0 & \frac{3}{2} & -1 \\ 0 & 0 & \frac{4}{3} \end{bmatrix}$$

$$b := \begin{bmatrix} 1 \\ \frac{1}{2} \\ \frac{1}{3} \end{bmatrix}$$

```
> forward_subs(l, b)
```

$$\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

2.2. Solution of upper triangular systems: ***backward substitution***

Similar conclusions can be drawn for a linear system $Ux = b$, where U is a non-singular upper triangular matrix of order n ($n \geq 2$). In this case the algorithm is called ***backward substitution*** and in the general case can be written as

$$x_n = \frac{b_n}{u_{nn}},$$
$$x_i = \frac{1}{u_{ii}}\left(b_i - \sum_{j=i+1}^{n} u_{ij}x_j\right), \quad i = n-1, \ldots, 1$$

> $u := 'u' : x := 'x' : b := 'b' :$
$U, X, B := Matrix(n, n, shape = triangular[upper], symbol = u), Vector(n, symbol = x),$
$Vector(n, symbol = b)$

$$U, X, B := \begin{bmatrix} u_{1,1} & u_{1,2} & u_{1,3} \\ 0 & u_{2,2} & u_{2,3} \\ 0 & 0 & u_{3,3} \end{bmatrix}, \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}, \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

> $backward := U.X = B$

$$backward := \begin{bmatrix} u_{1,1}x_1 + u_{1,2}x_2 + u_{1,3}x_3 \\ u_{2,2}x_2 + u_{2,3}x_3 \\ u_{3,3}x_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

> $lhs(backward)[3] = rhs(backward)[3]; isolate(\%, x_3)$

$$u_{3,3}x_3 = b_3$$
$$x_3 = \frac{b_3}{u_{3,3}}$$

> $lhs(backward)[2] = rhs(backward)[2]; isolate(\%, x_2)$

$$u_{2,2}x_2 + u_{2,3}x_3 = b_2$$
$$x_2 = \frac{-u_{2,3}x_3 + b_2}{u_{2,2}}$$

> $lhs(backward)[1] = rhs(backward)[1]; isolate(\%, x_1)$

$$u_{1,1}x_1 + u_{1,2}x_2 + u_{1,3}x_3 = b_1$$
$$x_1 = \frac{-u_{1,2}x_2 - u_{1,3}x_3 + b_1}{u_{1,1}}$$

Implementing the **backward substitution algorithm** writing Maple procedure!
Try the *backward_subs* procedure with matrix $A = tridiag([-1, 2, -1], n)$ *positive definite* matrix!

> $A := BandMatrix([-1, 2, -1], 1, n);$
> $p, l, u := LUDecomposition(A);$
> $b := u.Vector(n, 1)$

$$A := \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix}$$

$$p, l, u := \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 \\ -\frac{1}{2} & 1 & 0 \\ 0 & -\frac{2}{3} & 1 \end{bmatrix}, \begin{bmatrix} 2 & -1 & 0 \\ 0 & \frac{3}{2} & -1 \\ 0 & 0 & \frac{4}{3} \end{bmatrix}$$

$$b := \begin{bmatrix} 1 \\ \frac{1}{2} \\ \frac{4}{3} \end{bmatrix}$$

> *backward_subs* (u, b)

$$\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

3. The Cholesky Factorization of Symmetric and Positive Definite Matrices

As already pointed out, the factorization $L \cdot \mathrm{D} \cdot U^T$ simplifies considerably when A is symmetric because in such a case $U = L$, yielding the so-called $L \cdot \mathrm{D} \cdot L^T$ factorization. The computational cost halves, with respect to the LU factorization, to about $(n^3/3)$ flops. We shall denote by flop the single elementary floating-point operation (sum, subtraction, multiplication or division)

> *restart*

> $n := 3$; *with*(*LinearAlgebra*) : $A :=$ *BandMatrix*([−1, 2, −1], 1, n)

$$n := 3$$

$$A := \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix}$$

Show that matrix A is positive definite! Now we have to make complete squares!

> $X :=$ *Vector*(n, *symbol* = x)

$$X := \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

> *with*(*Student*[*Precalculus*]) :

> $p :=$ *expand*($X^+ .A.X$);

$$p := 2x_1^2 - 2x_1x_2 + 2x_2^2 - 2x_2x_3 + 2x_3^2$$

> *member1* := *op*(1, p) + *op*(2, p)

$$member1 := 2x_1^2 - 2x_1x_2$$

> $cs1 :=$ *CompleteSquare*(*member1*, [x_1])

$$cs1 := 2\left(x_1 - \frac{1}{2}x_2\right)^2 - \frac{1}{2}x_2^2$$

> $r1 := p - member1 + cs1$

$$r1 := \frac{3}{2}x_2^2 - 2x_2x_3 + 2x_3^2 + 2\left(x_1 - \frac{1}{2}x_2\right)^2$$

> $member2 := op(2, r1) + op(3, r1);\ cs2 := CompleteSquare(\%, x_3)$

$$member2 := -2x_2x_3 + 2x_3^2$$

$$cs2 := 2\left(x_3 - \frac{1}{2}x_2\right)^2 - \frac{1}{2}x_2^2$$

> $r2 := r1 - member2 + cs2$

$$r2 := x_2^2 + 2\left(x_1 - \frac{1}{2}x_2\right)^2 + 2\left(x_3 - \frac{1}{2}x_2\right)^2$$

> $simplify(p - r2)$

$$0$$

Similarly the eigenvalues are positive

> $Eigenvalues(A);$

$$\begin{bmatrix} 2 \\ 2-\sqrt{2} \\ 2+\sqrt{2} \end{bmatrix}$$

> $H := LUDecomposition(A, method = 'Cholesky')$

$$H := \begin{bmatrix} \sqrt{2} & 0 & 0 \\ -\frac{1}{2}\sqrt{2} & \frac{1}{2}\sqrt{6} & 0 \\ 0 & -\frac{1}{3}\sqrt{6} & \frac{2}{3}\sqrt{3} \end{bmatrix}$$

> $H.H^{+}$

$$\begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix}$$

Theorem. Let $A \in \mathbb{R}^{n \times n}$ be a symmetric and positive definite matrix. Then, there exists a unique upper triangular matrix H with positive diagonal entries such that

$$A = H^T \cdot H.$$

This factorization is called Cholesky factorization and the entries $h_{i,j}$ of H^T can be computed as follows:

$$h_{1,1} = \sqrt{a_{1,1}} \quad \text{and for } i = 2,\ldots, n$$

$$h_{ij} = \left(a_{ij} - \sum_{k=1}^{j-1} h_{ik}h_{jk}\right)/h_{jj}, \quad j = 1, \ldots, i-1,$$

$$h_{ii} = \left(a_{ii} - \sum_{k=1}^{i-1} h_{ik}^2\right)^{1/2}.$$

The algorithm can be write in the following compact form

```
for k=1:n-1
   A(k,k)=sqrt(A(k,k));   A(k+1:n,k)=A(k+1:n,k)/A(k,k);
   for j=k+1:n,   A(j:n,j)=A(j:n,j)-A(j:n,k)*A(j,k);   end
end
A(n,n)=sqrt(A(n,n));
```

Implementing the **Cholesky-factorization** algorithm writing Maple procedure!

```
factor_Cholesky := proc(A)
 local B, n, i, j, k, H :
      # initialize B=A and H=0
      n := Dimensions(A)[1] :
      B := Matrix(n, n) : H := Matrix(n, n) :
      for i from 1 to n do
        for j from 1 to n do
           B[i,j] := A[i,j]
        end do:
      end do:
      # Cholesky − factorization
      for k from 1 to (n − 1) do
           B[k, k] := sqrt(B[k, k]) :
           B[(k + 1)..n, k] := B[(k + 1)..n, k] / B[k, k] :
        for j from k + 1 to n do
          B[j..n, j] := B[j..n, j] − B[j..n, k]·B[j, k] :
        end do:
      end do:
    B[n, n] := sqrt(B[n, n]);
    for i from 1 to n do
        for j from 1 to i do
           H[i,j] := B[i,j]
        end do:
      end do:
   return (H);
 end proc:
```

Try this *Cholesky − factorization* procedure with matrix $A = tridiag([-1, 2, -1], n)$ *positive definite* matrix!

> $H := factor\_Cholesky(A); H.H^{+}$

$$H := \begin{bmatrix} \sqrt{2} & 0 & 0 \\ -\frac{1}{2}\sqrt{2} & \frac{1}{2}\sqrt{6} & 0 \\ 0 & -\frac{1}{3}\sqrt{6} & \frac{2}{3}\sqrt{3} \end{bmatrix}$$

$$\begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix}$$

4. RICHARDSON ITERATIVE METHOD FOR LINEAR EQUATIONS

LARGE-SCALE LINEAR SYSTEMS OF EQUATIONS
Computer Science Engineering Msc
Theoretical and Practical lesson

1. Simple iteration and their convergence

The most straightforward approach to an iterative solution of a linear system is to rewrite

(1.1) $$A \cdot x = b \quad (A \in \mathbb{R}^{n \times n}\ matrix, b \in \mathbb{R}^n, x \in \mathbb{R}^n)$$

as a linear fixed-point iteration. One way to do this is to write $Ax = b$ as

$$\omega \cdot A \cdot x = \omega \cdot b \qquad (\omega \in \mathbb{R})$$
$$x + \omega \cdot A \cdot x = x + \omega \cdot b$$
$$x = x - \omega \cdot A \cdot x + \omega \cdot b = (I - \omega \cdot A) \cdot x + \omega \cdot b = H \cdot x + d$$

where

$$H = (I - \omega \cdot A)\ , d = \omega \cdot b.$$

DEFINITION 1.1.
The iterative method

(1.2) $$x^{(k+1)} = H \cdot x^{(k)} + d$$

for system (1.1) is the so called (modified) **Richardson iteration**, when

$$H = (I - \omega \cdot A)\ ,\ d = \omega \cdot b$$

with appropriate number ω.

The Richardson iteration is a residual corrected iteration

$$x^{(k+1)} = (I - \omega \cdot A) \cdot x^{(k)} + \omega \cdot b = x^{(k)} + \omega \cdot (b - A \cdot x^{(k)}) = x^{(k)} + \omega \cdot r^{(k)}$$

where the vector $r = b - A \cdot x$ is called as the **residual**vector. So the iterated vector is always corrected by the residual vector multiplied by the parameter ω.

Lemma 1.2.
If M is an n × n matrix with $\|M\| < 1$ then matrix $(I - M)$ is non-singular and

(1.3) $$\left\| (I - M)^{-1} \right\| < \frac{1}{1 - \|M\|}$$

Proof. We will show that I - M is non-singular and that (1.3) holds by showing that the series

$$\sum_{j=0}^{\infty} M^j = (I - M)^{-1}$$

The partial sums

$$S_k = \sum_{j=0}^{k} M^j$$

form a Cauchy-sequence in $\mathbb{R}^{n \times n}$. To see this note that for all m > k

$$\|S_m - S_k\| \le \sum_{j=k+1}^{m} \|M^j\|.$$

Now, $\|M^j\| \le \|M\|^j$ because $\|A\|$ is a matrix norm that is induced by a vector norm. Hence

$$\|S_m - S_k\| \le \sum_{j=k+1}^{m} \|M\|^j = \|M\|^{k+1} \cdot \left(\frac{1 - \|A^{m-k}\|}{1 - \|M\|} \right) \to 0$$

as m, k $\to\infty$. Hence the sequence S_k converges, say to S. Since

$$M \cdot S_k + I = S_{k+1},$$

we must have $M \cdot S + I = S$ and hence $(I - M) \cdot S = I$. This proves that $(I - M)$ is non-singular and that $S = (I - M)^{-1}$.

Noting that

$$\left\| (I - M)^{-1} \right\| \le \sum_{j=0}^{\infty} \|M\|^j = \frac{1}{1 - \|M\|}$$

proves (1.3) and completes the proof. Q.e.d.

The following corollary is a direct consequence of Lemma 1.2.

Corollary 1.3.
If $\|H\| < 1$ then the iteration (1.2) converges to

$$x = (I - H)^{-1} \cdot d$$

for all initial vector $x^{(0)}$.

A consequence of Corollary1.3. is that Richardson iteration

(1.4) $$x^{(k+1)} = (I - \omega \cdot A) \cdot x^{(k)} + \omega \cdot b$$

will converge if $\|(I - \omega \cdot A)\| < 1$.

DEFINITION 1.4..
Let $\lambda_1, \lambda_2, ..., \lambda_n$ be the eigenvalues of $A \in \mathbb{R}^{n \times n}$. Then

$$\rho(A) = \max\left(|\lambda_1|, |\lambda_2|, ..., |\lambda_n| \right)$$

is the spectral radius of A.

We don't prove the following lemma.

Lemma 1.5.
Let H is an n × n matrix. Then

$$\lim_{k \to \infty} H^k = 0 \quad \text{if and only if } \rho(A) < 1.$$

For the fundamental convergence theorem holds:

THEOREM 1.6.
For $A \in \mathbb{R}^{n \times n}$ assume that $(I - H)$ *is nonsingular* and hence that the solution x^* of $x = H \cdot x + d$ is unique. Then the iteration sequence (1.4) converge to x^* for any starting point $x^{(0)} \in \mathbb{R}^n$ if and only if $\rho(H) < 1$.

Proof.
By subtraction of $x^* = H \cdot x^* + d$ from (1.2) we obtain the error equation

$$x^{(k+1)} - x^* = H \cdot \left(x^{(k)} - x^* \right) = ... = H^{k+1} \cdot \left(x^{(0)} - x^* \right)$$

Hence we find that $\lim_{k \to \infty} \left(x^k - x^* \right) = 0$ if and only if $\lim_{k \to \infty} H^k = 0$ and the result follows from

lemma 1.5. Q.e.d.
This theorem reduces the convergence analysis of the iteration (1.4) to the algebraic problem of showing that $\rho(H) < 1$.

2. Convergence of Richardson's iteration

If $\lambda_j \in \mathbb{C}$, j = 1,..., n, are the eigenvalues of A $\in \mathbb{R}^{n \times n}$, then, for constant $\omega > 0$, the eigenvalues of the iteration matrix $H_\omega = I - \omega \cdot A$ of Richardson's
method are $\left(1 - \omega \cdot \lambda_1\right), \ldots, \left(1 - \omega \cdot \lambda_n\right)$. Hence, we obtain

$$\rho\left(H_\omega\right) = \max_{i = 1,\ldots, n} \left|1 - \omega \cdot \lambda_i\right|.$$

Evidently, we are interested in the optimal value of ω for which $\rho\left(H_\omega\right)$ is minimal. For this reason we suppose that A is symmetrizable and hence that all eigenvalues $\lambda_1, \ldots, \lambda_n$ are real. Then with the algebraically smallest and largest values

$$\lambda_{\min} = \min_{j = 1,\ldots, n} \lambda_j \ , \ \lambda_{\max} = \max_{j = 1,\ldots, n} \lambda_j$$

we evidently have

(2.1) $$\rho\left(H_\omega\right) = \max\left(\left|1 - \omega \cdot \lambda_{\min}\right|, \left|1 - \omega \cdot \lambda_{\max}\right|\right).$$

In particular, for the case of positive eigenvalues we obtain the convergence result:

Lemma 2.1.
Let all **eigenvalues** of A $\in \mathbb{R}^{n \times n}$ are real and **positive** ($\lambda_{\min} > 0$) . Then Richardson's method converges if and only if

(2.2) $$0 < \omega < \frac{2}{\lambda_{\max}}$$

where $\lambda_{\max}$ is the algebraically largest eigenvalue of A. If the iteration is converging, then $\rho\left(H_\omega\right)$ is minimal for

(2.3) $$\omega_{opt} = \frac{2}{\lambda_{\max} + \lambda_{\min}} \ , \ \rho\left(H_\omega\right) = \frac{\lambda_{\max} - \lambda_{\min}}{\lambda_{\max} + \lambda_{\min}}.$$

Proof. If (2.2) holds then

$$-1 < 1 - \omega \cdot \lambda_{\max} \le 1 - \omega \cdot \lambda_{\min} < 1$$

whence (2.1) proves $\rho\left(H_\omega\right) < 1$ and therefore the convergence.
Conversely, if there is convergence, then (2.1) implies that

$$1 > \rho\left(H_\omega\right) \ge \left|1 - \omega \cdot \lambda_{\max}\right| \ge 1 - \omega \cdot \lambda_{\max}$$

and thus $\omega \cdot \lambda_{\max} > 0$ and $\lambda_{\max} > 0$. Similarly,

$$-1 < -\rho\left(H_\omega\right) \le -\left|1 - \omega \cdot \lambda_{\max}\right| \le 1 - \omega \cdot \lambda_{\max}$$

and thus $\omega \cdot \lambda_{\max} < 2$ and, hence altogether, that (2.2) holds.
For the proof of the last part note that $t = \omega \cdot \lambda_{\max} - 1$ and $t = 1 - \omega \cdot \lambda_{\min}$ define two straight lines in the (ω, t)-plane. Clearly, their point of intersection

is specifed by the values in (2.3) and (2.1) implies that indeed $\rho\left(H_\omega\right)$ is minimal for $\omega = \omega_{opt}$.

In particular, for symmetric, positive-definite matrices the maximal eigenvalue $\lambda_{\max}$ is equal to the norm $\|A\|_2$. Thus we obtain the corollary:

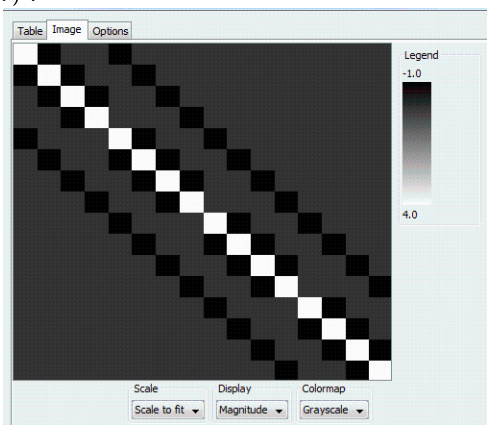
Corollary 2.2.
Let $A \in \mathbb{R}^{n \times n}$ be symmetric, positive-definite. Then Richardson's method converges if and only if

$$0 < \omega < \frac{2}{\|A\|_2}. \tag{2.4}$$

3. Richardson's iteration program for 2D Laplace-operator

Generally, standard discretization of elliptic linear partial differential operators results in symmetric, positive definite matrices. This is, for example, the case with the matrix M obtained by discretizing the 2D Laplace operator in "02_PDE_discretization.mw" Maple file. Thus **Corollary 2.2.** applies for all such problems.

```
> restart : n := 4 : with(LinearAlgebra) :
> MakeBlockTridiagonal := proc(n :: posint)
    local B, E, A, i :
    B := LinearAlgebra:-BandMatrix([-1, 4, -1], 1, n) : E := Matrix(n, n, 1, shape
      = diagonal) :
> A := Matrix(n^2, n^2) :
> A[1..n, 1..2·n] := Matrix([B, -E]) :
>
    i := 1 :
    while i ≤ n - 2 do
      A[i·n + 1..(i + 1)·n, (i - 1)·n + 1..(i + 2)·n] := Matrix([-E, B, -E]) :
      i := i + 1 :
    end do:
> A[(n - 1)·n + 1..n·n, (n - 2)·n + 1..n·n] := Matrix([-E, B]) : return A : end proc:
> A := MakeBlockTridiagonal(n) :
```



Choose the solution vector with all components equal to 1.

> $X := Vector(n^2, 1)$:

Calculate the right hand side vector from the solution X vector!

> $b := A.X$:

> $E := evalf(Eigenvalues(A)) : convert(\%, list)$

$$[0.7640, 5.2360, 2.7640, 7.2360, 3.0000, 5.0000, 1.7640, 6.2360, 3.0000, 5.0000, 1.7640, 6.2360, 4.0000, 4.0000, 4.0000, 4.0000]$$

> $\rho := \max(E);$

> $with(Student[NumericalAnalysis]) :$

> $SpectralRadius(A)$

$$\rho := 7.2360$$

$$5 + \sqrt{5}$$

> $A2 := evalf(Norm(A, 2))$

$$A2 := 7.2360$$

> $\omega max := \frac{2}{A2}$

$$\omega max := 0.2764$$

> $\omega 1 := 0.2;$

> $\omega optimal := \frac{2}{\max(E) + \min(E)}$

$$\omega 1 := 0.2000$$

$$\omega optimal := 0.2500$$

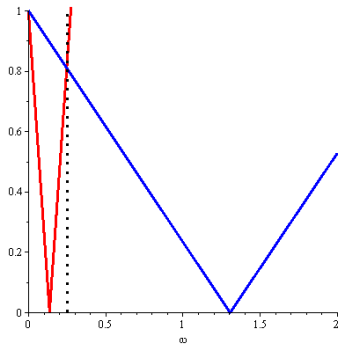
>

$\omega := '\omega'$:

$fig1 := plot([\mathrm{abs}(1 - \omega \cdot \max(E)), \mathrm{abs}(1 - \omega \cdot \min(E))], \omega = 0..2, 0..1, color = [red, blue], thickness = 3)$:

$fig2 := plot([[\omega optimal, 0], [\omega optimal, 1]], linestyle = 2, color = black, thickness = 3)$:

$plots[display](fig1, fig2)$



Example when the iteration is **converging**

> $\omega := \omega optimal; H := IdentityMatrix(n^2) - \omega \cdot A : d := \omega \cdot b$:

> $`spectral\ radius\ H` = SpectralRadius(H);$

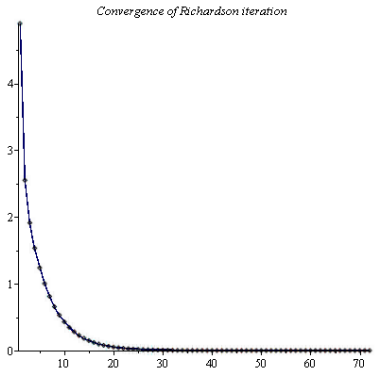
```
> x := Vector(n^2) : maxiterlimit := 100 :
  r := b - A.x : r2 := evalf(Norm(r, 2)) :
  konv := r2 : k := 0 :
  while r2 > 0.000001 and (k < maxiterlimit) do
    x := H.x + d;
    r := b - A.x;
    k := k + 1 :
    r2 := evalf(Norm(r, 2));
    konv := konv, r2;
  end do:
  residual_norms := [konv]; numelems([konv]);
  Statistics:-LineChart([konv], title = `Convergence of Richardson iteration`);
  solution_vector = convert(x, list)
```

$$\omega := 0.2500$$

$$spectral\ radius\ H = 0.8105$$

$$residual\_norms := [4.8980, 2.5500, 1.9120, 1.5340, 1.2400, 1.0030, 0.8115, 0.6565, 0.5312, 0.4297, 0.3476, 0.2813, 0.2275, 0.1841, 0.1489, 0.1205, 0.0975, 0.0789, 0.0638, 0.0516, 0.0418, 0.0338, 0.0273, 0.0221, 0.0179, 0.0145, 0.0117, 0.0095, 0.0077, 0.0062, 0.0050, 0.0041, 0.0033, 0.0027, 0.0021, 0.0017, 0.0014, 0.0011, 0.0009, 0.0007, 0.0006, 0.0005, 0.0004, 0.0003, 0.0003, 0.0002, 0.0002, 0.0001, 0.0001, 0.0001, 0.0001, 0.0001, 0.0000, 8.4440\,10^{-7}]$$

72



$$solution\_vector = [1.0000, 1.0000, 1.0000, 1.0000, 1.0000, 1.0000, 1.0000, 1.0000, 1.0000, 1.0000, 1.0000, 1.0000, 1.0000, 1.0000, 1.0000, 1.0000]$$

Example when the iteration is **diverging**

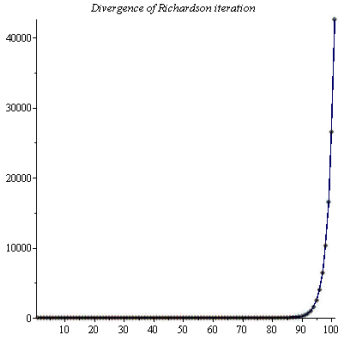
```
> ω := 0.36; H := IdentityMatrix(n^2) - ω·A : d := ω·b :
> `spectral radius H` = SpectralRadius(H);
```

```
> x := Vector(n^2) : iterlimit := 100 :
  r := b - A.x : r2 := evalf(Norm(r, 2)) :
 div := r2 : k := 0 :
 while r2 > 0.00001 and (k < iterlimit) do
   x := H.x + d;
   r := b - A.x;
   k := k + 1 :
   r2 := evalf(Norm(r, 2));
   div := div, r2;
 end do:
residual_norms := div;
Statistics:-LineChart([div], title = `Divergence of Richardson iteration`);
 solution_vector = convert(x, list)
```

$$\omega := 0.3600$$

$$\text{spectral radius } H = 1.6000$$

$$\begin{aligned} \text{residual\_norms} := {} & 4.8980, 2.3350, 1.7510, 1.3430, 1.0480, 0.8344, 0.6773, 0.5602, 0.4710, \\ & 0.4015, 0.3459, 0.3004, 0.2624, 0.2301, 0.2024, 0.1784, 0.1574, 0.1391, 0.1229, 0.1087, \\ & 0.0961, 0.0850, 0.0752, 0.0666, 0.0589, 0.0521, 0.0461, 0.0408, 0.0361, 0.0320, 0.0283, \\ & 0.0250, 0.0222, 0.0196, 0.0174, 0.0154, 0.0136, 0.0120, 0.0106, 0.0094, 0.0083, 0.0074, \\ & 0.0065, 0.0058, 0.0051, 0.0045, 0.0040, 0.0035, 0.0031, 0.0028, 0.0025, 0.0022, 0.0019, \\ & 0.0017, 0.0015, 0.0013, 0.0012, 0.0010, 0.0009, 0.0008, 0.0008, 0.0008, 0.0009, 0.0012, \\ & 0.0018, 0.0028, 0.0044, 0.0071, 0.0113, 0.0182, 0.0292, 0.0468, 0.0752, 0.1206, 0.1936, \\ & 0.3108, 0.4988, 0.8006, 1.2850, 2.0620, 3.3100, 5.3120, 8.5260, 13.6800, 21.9600, 35.2500, \\ & 56.5800, 90.8100, 145.7000, 233.9000, 375.4000, 602.6000, 967.1000, 1552.0000, 2491.0000, \\ & 3998.0000, 6417.0000, 10300.0000, 16530.0000, 26530.0000, 42580.0000 \end{aligned}$$



$$\begin{aligned} \text{solution\_vector} = [& -812.2635, 1316.8880, -1314.8880, 814.2635, 1316.8880, -2128.1514, \\ & 2130.1514, -1314.8880, -1314.8880, 2130.1514, -2128.1514, 1316.8880, 814.2635, \\ & -1314.8880, 1316.8880, -812.2635] \end{aligned}$$

5. Jacobi, Gauss-Seidel and relaxation methods

LARGE-SCALE LINEAR SYSTEMS OF EQUATIONS
Computer Science Engineering Msc
Theoretical and Practical lesson

1. Jacobi iteration

This worksheet begins by reviewing the **basic iterative methods** (BIM) for solving linear systems. Given an n×n real matrix A and a real n-vector b , the problem considered is:

Find x belonging to $\mathbb{R}^n$ such that

(1.1) $A \cdot x = b$

Equation (1.1) is a linear system, A is the coefficient matrix, b is the right-hand side vector, and x is the vector of unknowns. Most of the methods covered in this chapter involve passing from one iterate to the next by modifying one or a few components of an approximate vector solution at a time. This is natural since there are simple criteria when modifying a component in order to improve iteration. One example is to annihilate some component(s) of the residual vector $b - Ax$. The convergence of these methods is rarely guaranteed for all matrices, but a large body of theory exists for the case where the coefficient matrix arises from the finite difference discretization of Elliptic Partial Differential Equations.

We begin with the decomposition

$$A = L + \mathrm{D} + U$$

in which D is the diagonal of A, L its strict lower part, and U its strict upper part, as illustrated in Figure 1. It is always assumed that the diagonal entries of A are all nonzero.

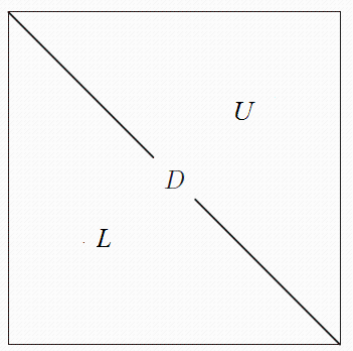


Figure 1. Initial partitioning of matrix A

Definition: Accuracy of approximation

Approximation $x^{(k+1)}$ is more appropriate as approximation $x^{(k)}$ if

$$\|b - A \cdot x^{(k+1)}\| < \|b - A \cdot x^{(k)}\|$$

Basic relation of simple iteration process which is corrected by the residual $r^{(k)} = b - Ax^{(k)}$ at step k^{th}

$$x^{(k+1)} = x^{(k)} + M^{-1}\left(b - Ax^{(k)}\right)$$

where $x^{(0)}$ is the initial guess.

The appropriate choice of matrix M is the key point of the iteration

$$Mx^{(k+1)} = Mx^{(k)} + \left(b - Ax^{(k)}\right)$$

$$Mx^{(k+1)} = (M-A)x^{(k)} + b$$

(1.2) $$Mx^{(k+1)} = Nx^{(k)} + b$$

where $N = M - A$ and so $A = M - N$. Matrix M also called as ***preconditioner***.

Convergence.

Assuming that $x^{(k)} \to x$ as n tends to infinity. Then

$$Mx = Nx + b \quad \Leftrightarrow \quad (M-N)x = Ax = b$$

and the limiting value x is the solution $x = A^{-1} \cdot b$ of eq. (1.1) .

1.1. DEFINITION. We get the **Jacobi** iteration when

$$M_{JA} = \mathrm{D}_A \quad \text{and} \quad N_{JA} = -(L_A + U_A), A = M_{JA} - N_{JA}$$

where

$\mathrm{D}_A =$ diagonal of matrix A (later only D)

$L_A =$ strict lower part of matrix A (later only L)

$U_A =$ strict upper part of matrix A (later only U)

Conducting the basic iteration formulas we split matrix $A = L + \mathrm{D} + U$

$$(L + \mathrm{D} + U)x = b$$

$$Dx = -(L+U)x + b$$

$$x = -\mathrm{D}^{-1} \cdot (L+U)x + \mathrm{D}^{-1}b$$

$$x^{(k+1)} = \mathrm{D}^{-1} \cdot (L+U)x^{(k)} + \mathrm{D}^{-1}b$$

(1.3.) $$x^{(k+1)} = Bx^{(k)} + h$$

if all diagonal elements of A are non-zero then $B = \mathrm{D}^{-1} \cdot (L+U)$ and $h = \mathrm{D}^{-1}b$.

1.2. Definition. Consistency

An iterative method of the form (1.3) is said to be **consistent** with (1.1) $x = Ax$ iff h and B are such that $x = Bx + h$. Equivalently,

$$h = (I - B) \cdot A^{-1} \cdot b \ .$$

Theorem

Let (1.3) be a **consistent** method. Then, the sequence of vectors $\{x^{(k)}\}$ **converges to the solution** of (1.1) for any choice of $x^{(0)}$ iff the spectral radius is less then 1, i.e.

$$\rho(B) < 1.$$

1.1. EXAMPLE. Solution of linear equations with using Jacobi-iteration in case of 1D Laplace-operator.

>

```
restart : with(LinearAlgebra) : with(ArrayTools) : with(Student[NumericalAnalysis]) :
  with(plots) :
```

> $n := 5;\ A, b := BandMatrix([-1, 2, -1], 1, n), Vector\left(n, i \to evalf\left(\frac{\pi^2 \cdot \sin\left(\frac{\pi}{n+1} \cdot i\right)}{(n+1)^2}\right)\right)$

$$n := 5$$

$$A, b := \left[\begin{bmatrix} 2 & -1 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & -1 & 2 \end{bmatrix}, \begin{bmatrix} 0.1371 \\ 0.2375 \\ 0.2742 \\ 0.2375 \\ 0.1371 \end{bmatrix}\right]$$

> $X := \left[seq\left(\frac{i}{n+1}, i = 1..n\right)\right]$

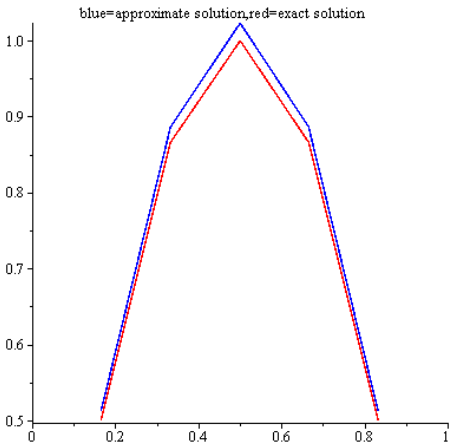
> $approx\_solution := LinearAlgebra:-LinearSolve(A, b) : P\_a := plot(X, approx\_solution, 0..1, color = blue) :$

> $exact\_solution := Vector\left(\left[seq\left(\sin(\pi \cdot x), x = X\right)\right]\right) : P\_e := plot(X, exact\_solution, 0..1, color = red) :$
$X, exact\_solution, approx\_solution;$

> $display(P\_a, P\_e, title = "blue=approximate solution,red=exact solution")$

$$X := \left[\frac{1}{6}, \frac{1}{3}, \frac{1}{2}, \frac{2}{3}, \frac{5}{6}\right]$$

$$\left[\frac{1}{6}, \frac{1}{3}, \frac{1}{2}, \frac{2}{3}, \frac{5}{6}\right], \begin{bmatrix} \frac{1}{2} \\ \frac{1}{2}\sqrt{3} \\ 1 \\ \frac{1}{2}\sqrt{3} \\ \frac{1}{2} \end{bmatrix}, \begin{bmatrix} 0.5117 \\ 0.8863 \\ 1.0234 \\ 0.8863 \\ 0.5117 \end{bmatrix}$$



Jacobi iteration

> $d := Diagonal(A) : diag := Diagonal(d); N := diag - A$

$$diag := \begin{bmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 2 \end{bmatrix} \qquad N := \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

> $Dinverse := diag^{(-1)};$

$$Dinverse := \begin{bmatrix} \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} \end{bmatrix}$$

> $B, h := Dinverse.N, Dinverse.b$

$$B, h := \begin{bmatrix} 0 & \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 \\ 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 \\ 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & 0 & \frac{1}{2} & 0 \end{bmatrix}, \begin{bmatrix} 0.0686 \\ 0.1188 \\ 0.1371 \\ 0.1188 \\ 0.0686 \end{bmatrix}$$

Getting similar formula when we use procedure "*IterativeFormula*" from *Student*[*NumericalAnalysis*] package.

> $Jacobi := IterativeFormula(A, b, method = jacobi, output = ['T', spectralradius], showsteps = true)$

the lower triangular matrix L:

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

the upper triangular matrix U:

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

the diagonal matrix D:

$$\begin{bmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 2 \end{bmatrix}$$

```
the matrix T and the vector c in the iterative formula x^(k+1)=Tx^(k)+c,
T=D^(-1)(L+U), c=D^(-1)b:
```

$$\begin{bmatrix} 0 & \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 \\ 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 \\ 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & 0 & \frac{1}{2} & 0 \end{bmatrix}, \begin{bmatrix} 0.0686 \\ 0.1188 \\ 0.1371 \\ 0.1188 \\ 0.0686 \end{bmatrix}$$

$$Jacobi := \begin{bmatrix} 0 & \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 \\ 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 \\ 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & 0 & \frac{1}{2} & 0 \end{bmatrix}, \frac{1}{2}\sqrt{3}$$

The spectral radius $\varrho(B) < 1, therefore$ the iteration converges for all initial point.

```
> ρJ := evalf(Jacobi[2])
```

$$\rho J := 0.8660$$

Take 50 iteration steps

```
> initial := Vector([seq(0, k = 1..n)]);
  x0 := initial:
  for i from 1 to 50 do
  x0 := B.x0 + h
  end do:
  evalf(approx_solution), evalf(x0), evalf(x0 − approx_solution)
```

$$initial := \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0.5117 \\ 0.8863 \\ 1.0230 \\ 0.8863 \\ 0.5117 \end{bmatrix}, \begin{bmatrix} 0.5113 \\ 0.8856 \\ 1.0230 \\ 0.8856 \\ 0.5113 \end{bmatrix}, \begin{bmatrix} -0.0004 \\ -0.0007 \\ -0.0008 \\ -0.0007 \\ -0.0004 \end{bmatrix}$$

```
> IterativeFormula(A, b, method = jacobi, iterations = 50, initialapprox = Vector([seq(0, k = 1
    ..n)]), digits = 5, output = ['iterates'])
```

$$\left[\begin{bmatrix}0\\0\\0\\0\\0\end{bmatrix},\begin{bmatrix}0.0686\\0.1188\\0.1371\\0.1188\\0.0686\end{bmatrix},\begin{bmatrix}0.1279\\0.2216\\0.2559\\0.2216\\0.1279\end{bmatrix},\begin{bmatrix}0.1793\\0.3106\\0.3587\\0.3106\\0.1793\end{bmatrix},\begin{bmatrix}0.2239\\0.3878\\0.4477\\0.3878\\0.2239\end{bmatrix},\begin{bmatrix}0.2624\\0.4546\\0.5249\\0.4546\\0.2624\end{bmatrix},\begin{bmatrix}0.2958\\0.5124\\0.5917\\0.5124\\0.2958\end{bmatrix},\begin{bmatrix}0.3248\\0.5625\\0.6495\\0.5625\\0.3248\end{bmatrix},\right.$$
$$\begin{bmatrix}0.3498\\0.6059\\0.6996\\0.6059\\0.3498\end{bmatrix},\begin{bmatrix}0.3715\\0.6435\\0.7430\\0.6435\\0.3715\end{bmatrix},\begin{bmatrix}0.3903\\0.6760\\0.7805\\0.6760\\0.3903\end{bmatrix},\begin{bmatrix}0.4065\\0.7042\\0.8131\\0.7042\\0.4065\end{bmatrix},\begin{bmatrix}0.4206\\0.7286\\0.8413\\0.7286\\0.4206\end{bmatrix},\begin{bmatrix}0.4328\\0.7497\\0.8657\\0.7497\\0.4328\end{bmatrix},\begin{bmatrix}0.4434\\0.7680\\0.8868\\0.7680\\0.4434\end{bmatrix},\begin{bmatrix}0.4526\\0.7839\\0.9051\\0.7839\\0.4526\end{bmatrix},$$
$$\begin{bmatrix}0.4605\\0.7976\\0.9209\\0.7976\\0.4605\end{bmatrix},\begin{bmatrix}0.4673\\0.8095\\0.9347\\0.8095\\0.4673\end{bmatrix},\begin{bmatrix}0.4733\\0.8198\\0.9466\\0.8198\\0.4733\end{bmatrix},\begin{bmatrix}0.4784\\0.8287\\0.9569\\0.8287\\0.4784\end{bmatrix},\begin{bmatrix}0.4829\\0.8364\\0.9658\\0.8364\\0.4829\end{bmatrix},\begin{bmatrix}0.4868\\0.8431\\0.9735\\0.8431\\0.4868\end{bmatrix},\begin{bmatrix}0.4901\\0.8489\\0.9802\\0.8489\\0.4901\end{bmatrix},\begin{bmatrix}0.4930\\0.8539\\0.9860\\0.8539\\0.4930\end{bmatrix},$$
$$\begin{bmatrix}0.4955\\0.8582\\0.9910\\0.8582\\0.4955\end{bmatrix},\begin{bmatrix}0.4977\\0.8620\\0.9953\\0.8620\\0.4977\end{bmatrix},\begin{bmatrix}0.4995\\0.8653\\0.9991\\0.8653\\0.4995\end{bmatrix},\begin{bmatrix}0.5012\\0.8681\\1.0024\\0.8681\\0.5012\end{bmatrix},\begin{bmatrix}0.5026\\0.8705\\1.0052\\0.8705\\0.5026\end{bmatrix},\begin{bmatrix}0.5038\\0.8726\\1.0076\\0.8726\\0.5038\end{bmatrix},\begin{bmatrix}0.5049\\0.8745\\1.0097\\0.8745\\0.5049\end{bmatrix},\begin{bmatrix}0.5058\\0.8760\\1.0116\\0.8760\\0.5058\end{bmatrix},$$
$$\begin{bmatrix}0.5066\\0.8774\\1.0131\\0.8774\\0.5066\end{bmatrix},\begin{bmatrix}0.5073\\0.8786\\1.0145\\0.8786\\0.5073\end{bmatrix},\begin{bmatrix}0.5078\\0.8796\\1.0157\\0.8796\\0.5078\end{bmatrix},\begin{bmatrix}0.5084\\0.8805\\1.0167\\0.8805\\0.5084\end{bmatrix},\begin{bmatrix}0.5088\\0.8813\\1.0176\\0.8813\\0.5088\end{bmatrix},\begin{bmatrix}0.5092\\0.8820\\1.0184\\0.8820\\0.5092\end{bmatrix},\begin{bmatrix}0.5095\\0.8826\\1.0191\\0.8826\\0.5095\end{bmatrix},\begin{bmatrix}0.5098\\0.8831\\1.0197\\0.8831\\0.5098\end{bmatrix},$$
$$\begin{bmatrix}0.5101\\0.8835\\1.0202\\0.8835\\0.5101\end{bmatrix},\begin{bmatrix}0.5103\\0.8839\\1.0206\\0.8839\\0.5103\end{bmatrix},\begin{bmatrix}0.5105\\0.8842\\1.0210\\0.8842\\0.5105\end{bmatrix},\begin{bmatrix}0.5107\\0.8845\\1.0213\\0.8845\\0.5107\end{bmatrix},\begin{bmatrix}0.5108\\0.8847\\1.0216\\0.8847\\0.5108\end{bmatrix},\begin{bmatrix}0.5109\\0.8850\\1.0218\\0.8850\\0.5109\end{bmatrix},\begin{bmatrix}0.5110\\0.8851\\1.0221\\0.8851\\0.5110\end{bmatrix},\begin{bmatrix}0.5111\\0.8853\\1.0222\\0.8853\\0.5111\end{bmatrix},$$
$$\left.\begin{bmatrix}0.5112\\0.8854\\1.0224\\0.8854\\0.5112\end{bmatrix},\begin{bmatrix}0.5113\\0.8856\\1.0225\\0.8856\\0.5113\end{bmatrix},\begin{bmatrix}0.5113\\0.8856\\1.0227\\0.8856\\0.5113\end{bmatrix}\right]$$

evalf (*approx_solution*)

$$\begin{bmatrix}0.5117\\0.8863\\1.0230\\0.8863\\0.5117\end{bmatrix}$$

1.2. EXAMPLE. Solution using symbols

> $n := 4;\ b := 'b'$:

> $A, bb := Matrix(\mathrm{n}, \mathrm{n}, symbol = a), \mathrm{Vector}(\mathrm{n}, \mathrm{symbol}= \mathrm{b})$

$$n := 4$$

$$A, bb := \begin{bmatrix}a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4}\\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4}\\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4}\\ a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4}\end{bmatrix}, \begin{bmatrix}b_1\\ b_2\\ b_3\\ b_4\end{bmatrix}$$

> $d := Diagonal(A) : diag := Diagonal(d)$

$$diag := \begin{bmatrix} a_{1,1} & 0 & 0 & 0 \\ 0 & a_{2,2} & 0 & 0 \\ 0 & 0 & a_{3,3} & 0 \\ 0 & 0 & 0 & a_{4,4} \end{bmatrix}$$

> $N := diag - A$

$$N := \begin{bmatrix} 0 & -a_{1,2} & -a_{1,3} & -a_{1,4} \\ -a_{2,1} & 0 & -a_{2,3} & -a_{2,4} \\ -a_{3,1} & -a_{3,2} & 0 & -a_{3,4} \\ -a_{4,1} & -a_{4,2} & -a_{4,3} & 0 \end{bmatrix}$$

> $x0 := 'x0' : initial := Vector(n, symbol = x0)$

$$initial := \begin{bmatrix} x0_1 \\ x0_2 \\ x0_3 \\ x0_4 \end{bmatrix}$$

> $Dinverse := diag^{(-1)};$

> $B, h := Dinverse.N, Dinverse.bb$

$$Dinverse := \begin{bmatrix} \frac{1}{a_{1,1}} & 0 & 0 & 0 \\ 0 & \frac{1}{a_{2,2}} & 0 & 0 \\ 0 & 0 & \frac{1}{a_{3,3}} & 0 \\ 0 & 0 & 0 & \frac{1}{a_{4,4}} \end{bmatrix}$$

$$B, h := \begin{bmatrix} 0 & -\frac{a_{1,2}}{a_{1,1}} & -\frac{a_{1,3}}{a_{1,1}} & -\frac{a_{1,4}}{a_{1,1}} \\ -\frac{a_{2,1}}{a_{2,2}} & 0 & -\frac{a_{2,3}}{a_{2,2}} & -\frac{a_{2,4}}{a_{2,2}} \\ -\frac{a_{3,1}}{a_{3,3}} & -\frac{a_{3,2}}{a_{3,3}} & 0 & -\frac{a_{3,4}}{a_{3,3}} \\ -\frac{a_{4,1}}{a_{4,4}} & -\frac{a_{4,2}}{a_{4,4}} & -\frac{a_{4,3}}{a_{4,4}} & 0 \end{bmatrix}, \begin{bmatrix} \frac{b_1}{a_{1,1}} \\ \frac{b_2}{a_{2,2}} \\ \frac{b_3}{a_{3,3}} \\ \frac{b_4}{a_{4,4}} \end{bmatrix}$$

> $x1 := B.initial + h$

$$x1 := \begin{bmatrix} -\frac{a_{1,2}\,x0_2}{a_{1,1}} - \frac{a_{1,3}\,x0_3}{a_{1,1}} - \frac{a_{1,4}\,x0_4}{a_{1,1}} + \frac{b_1}{a_{1,1}} \\ -\frac{a_{2,1}\,x0_1}{a_{2,2}} - \frac{a_{2,3}\,x0_3}{a_{2,2}} - \frac{a_{2,4}\,x0_4}{a_{2,2}} + \frac{b_2}{a_{2,2}} \\ -\frac{a_{3,1}\,x0_1}{a_{3,3}} - \frac{a_{3,2}\,x0_2}{a_{3,3}} - \frac{a_{3,4}\,x0_4}{a_{3,3}} + \frac{b_3}{a_{3,3}} \\ -\frac{a_{4,1}\,x0_1}{a_{4,4}} - \frac{a_{4,2}\,x0_2}{a_{4,4}} - \frac{a_{4,3}\,x0_3}{a_{4,4}} + \frac{b_4}{a_{4,4}} \end{bmatrix}$$

Algorithm of Jacobi iteration

$$x_i^{(k+1)} = \frac{\left(b_i - \sum_{j=1, j \neq i}^{n} a_{i,j} \cdot x_j^{(k)} \right)}{a_{i,i}}, \; i = 1, 2, \ldots, n$$

1.3. EXAMPLE. Procedure for counting one step of Jacobi-iteration

Following procedure count only one step of Jacobi iteration, i.e. using initial value $x^{(0)}$ count the next approximation

$$x^{(1)} = -\mathrm{D}^{-1} \cdot (L + U) x^{(0)} + \mathrm{D}^{-1} b.$$

```
> GJ := proc(x0, A, b)
      local i, j, n, s, x1 :
    n := LinearAlgebra[Dimension](b) :
   x1 := NULL :
for i from 1 to n do
    s := b_i :
    for j from 1 to n do
        if j ≠ i then s := s − A_{i,j}·x0_j end if:
    end do:
  x1 := x1, s / A_{i,i}
end do:
 RETURN([x1]) :
end proc:
```

1.4. EXAMPLE. Testing of Jacobi-iteration for convergence

```
> n := 5; A, b := BandMatrix([−1, 2, −1], 1, n), Vector(n, i → evalf(π^2 sin(π i/(n + 1)) / (n + 1)^2))
```

$$n := 5$$

$$A, b := \begin{bmatrix} 2 & -1 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & -1 & 2 \end{bmatrix}, \begin{bmatrix} 0.1371 \\ 0.2375 \\ 0.2742 \\ 0.2375 \\ 0.1371 \end{bmatrix}$$

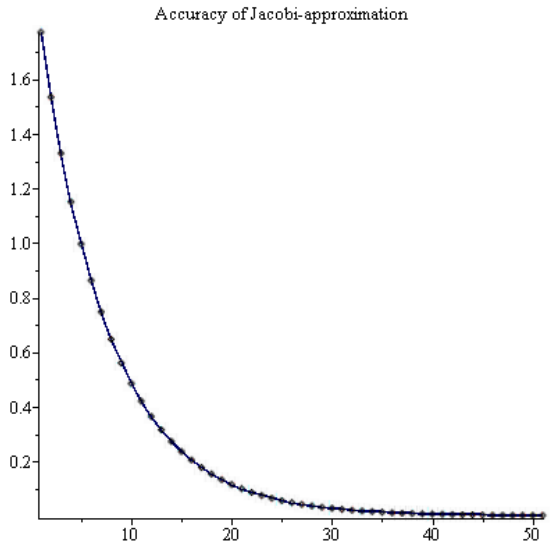
```
> solution:= evalf(LinearAlgebra:-LinearSolve(A, b))
```

$$solution := \begin{bmatrix} 0.5117 \\ 0.8863 \\ 1.0230 \\ 0.8863 \\ 0.5117 \end{bmatrix}$$

```
> x0 := Vector(n);
>
  deviation := LinearAlgebra[Norm](approx_solution - Vector(x0), 2) : differences
       := deviation :
  for i from 1 to 50 do
    x0 := GJ(x0, A, b) :
    deviation := LinearAlgebra[Norm](approx_solution - Vector(x0), 2) :
    differences := differences, evalf(deviation, 8) :
  end do:
> `number of iterartion` = nops([differences]) : convergenceJ := [differences];
> Statistics:-LineChart(convergenceJ, title = "Accuracy of Jacobi-approximation)"
```

$$x0 := \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$convergenceJ := [1.7726, 1.5351, 1.3293, 1.1512, 0.9970, 0.8634, 0.7478, 0.6476, 0.5608, 0.4856, 0.4206, 0.3645, 0.3157, 0.2732, 0.2363, 0.2050, 0.1775, 0.1537, 0.1332, 0.1153, 0.0999, 0.0863, 0.0748, 0.0647, 0.0561, 0.0488, 0.0424, 0.0368, 0.0320, 0.0281, 0.0241, 0.0210, 0.0183, 0.0162, 0.0136, 0.0116, 0.0100, 0.0088, 0.0077, 0.0070, 0.0063, 0.0053, 0.0044, 0.0041, 0.0035, 0.0027, 0.0025, 0.0023, 0.0022, 0.0021, 0.0021]$$



EXAMPLE

1.1. Increase the value n and examine the convergence speed!

1.2. Solve system of linear equations by the 2D Laplacian matrix equations using Jacobi iteration!

2. Gauss-Seidel iteration

Solve system of linear equations

$$A \cdot x = b$$

by splitting matrix

$$A = (L + \mathrm{D} + U), M = L + \mathrm{D}, N = -U, A = M - N.$$

Derive the Gauss-Seidel iteration process in vector-matrix form

$$(L + \mathrm{D} + U)x = b$$
$$(L + \mathrm{D})x = -Ux + b$$
$$x = -(L + \mathrm{D})^{-1} \cdot Ux + (L + \mathrm{D})^{-1} b$$
$$x^{(k+1)} = -(L + \mathrm{D})^{-1} \cdot Ux^{(k)} + (L + \mathrm{D})^{-1} b$$
$$x^{(k+1)} = Bx^{(k)} + h,\ B = -(L + \mathrm{D})^{-1} \cdot U,\ h = (L + \mathrm{D})^{-1} b$$

2.1. DEFINITION. Gauss-Seidel iteration

$$M_{GS} = L_A + \mathrm{D}_A \quad \text{and} \quad N_{GS} = -U_A$$

where

$\mathrm{D}_A =$ diagonal of matrix A (further D)

$L_A =$ strict lower part of matrix A (further L)

$U_A =$ strict upper part of matrix A (further U)

Comparing the matrices of Jacobi and Gauss-Seidel iterations

Jacobi-method	Gauss-Seidel method
$A = M - N$	
$M_{JA} = \mathrm{D}_A$ és $N_{JA} = -\left(L_A + U_A\right)$	$M_{GS} = L_A + \mathrm{D}_A$ és $N_{GS} = -U_A$
$x^{(k+1)} = \left(M^{-1} \cdot N\right) \cdot x^{(k)} + M^{-1} \cdot b = B \cdot x^{(k)} + h$	

2.2. EXAMPLE. Solution of linear equations with using Gauss-Seidel-iteration with 1D Laplace- matrix

```
> restart : with(LinearAlgebra) : with(ArrayTools) : with(Student[NumericalAnalysis]) :
> n := 5; A, b := BandMatrix([-1, 2, -1], 1, n), Vector(n, i -> evalf(Pi^2 * sin(Pi/(n+1) * i) / (n+1)^2))
```

$$n := 5$$

$$A, b := \begin{bmatrix} 2 & -1 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & -1 & 2 \end{bmatrix}, \begin{bmatrix} 0.1371 \\ 0.2375 \\ 0.2742 \\ 0.2375 \\ 0.1371 \end{bmatrix}$$

```
> solution := LinearAlgebra:-LinearSolve(A, b)
```

$$solution := \begin{bmatrix} 0.5117 \\ 0.8863 \\ 1.0234 \\ 0.8863 \\ 0.5117 \end{bmatrix}$$

```
> d := Diagonal(A) : diag := Diagonal(d)
```

$$diag := \begin{bmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 2 \end{bmatrix}$$

```
> L, U := LowerTriangle(A) - Diagonal(d), UpperTriangle(A) - Diagonal(d)
```

$$L, U := \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 \end{bmatrix}, \begin{bmatrix} 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

```
> N, M := -U, L + diag
```

$$N, M := \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 2 & 0 & 0 & 0 & 0 \\ -1 & 2 & 0 & 0 & 0 \\ 0 & -1 & 2 & 0 & 0 \\ 0 & 0 & -1 & 2 & 0 \\ 0 & 0 & 0 & -1 & 2 \end{bmatrix}$$

> $Minverse := M^{(-1)}$;

$$Minverse := \begin{bmatrix} \frac{1}{2} & 0 & 0 & 0 & 0 \\ \frac{1}{4} & \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{8} & \frac{1}{4} & \frac{1}{2} & 0 & 0 \\ \frac{1}{16} & \frac{1}{8} & \frac{1}{4} & \frac{1}{2} & 0 \\ \frac{1}{32} & \frac{1}{16} & \frac{1}{8} & \frac{1}{4} & \frac{1}{2} \end{bmatrix}$$

> $B, h := Minverse.N, Minverse.b$

$$B, h := \begin{bmatrix} 0 & \frac{1}{2} & 0 & 0 & 0 \\ 0 & \frac{1}{4} & \frac{1}{2} & 0 & 0 \\ 0 & \frac{1}{8} & \frac{1}{4} & \frac{1}{2} & 0 \\ 0 & \frac{1}{16} & \frac{1}{8} & \frac{1}{4} & \frac{1}{2} \\ 0 & \frac{1}{32} & \frac{1}{16} & \frac{1}{8} & \frac{1}{4} \end{bmatrix}, \begin{bmatrix} 0.0686 \\ 0.1530 \\ 0.2136 \\ 0.2256 \\ 0.1813 \end{bmatrix}$$

Similar deduction using procedure "IterativeFormula" from $Student[NumericalAnalysis]$ package.

> $G\_S := IterativeFormula(A, b, method = gaussseidel, output = ['T', spectralradius], showsteps = true)$

the lower triangular matrix L:

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

the upper triangular matrix U:

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

the diagonal matrix D:

$$\begin{bmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 2 \end{bmatrix}$$

the matrix T and the vector c in the iterative formula x^(k+1)=Tx^(k)+c, T=(D-L)^(-1)U, c=(D-L)^(-1)b:

$$\begin{bmatrix} 0 & \frac{1}{2} & 0 & 0 & 0 \\ 0 & \frac{1}{4} & \frac{1}{2} & 0 & 0 \\ 0 & \frac{1}{8} & \frac{1}{4} & \frac{1}{2} & 0 \\ 0 & \frac{1}{16} & \frac{1}{8} & \frac{1}{4} & \frac{1}{2} \\ 0 & \frac{1}{32} & \frac{1}{16} & \frac{1}{8} & \frac{1}{4} \end{bmatrix}, \begin{bmatrix} 0.0686 \\ 0.1531 \\ 0.2136 \\ 0.2256 \\ 0.1814 \end{bmatrix}$$

$$G\_S := \begin{bmatrix} 0 & \frac{1}{2} & 0 & 0 & 0 \\ 0 & \frac{1}{4} & \frac{1}{2} & 0 & 0 \\ 0 & \frac{1}{8} & \frac{1}{4} & \frac{1}{2} & 0 \\ 0 & \frac{1}{16} & \frac{1}{8} & \frac{1}{4} & \frac{1}{2} \\ 0 & \frac{1}{32} & \frac{1}{16} & \frac{1}{8} & \frac{1}{4} \end{bmatrix}, \frac{3}{4}$$

Spectral radius of matrix B is smaller than 1, therefore the iteration is converging for all initial value. The spectral radius of Gauss-Seidel iteration is smaller than the spectral radius belongs to Jacobi-iteration, therefore the GS-iteration is faster now.

> $\rho GS := evalf\left(G\_S_2\right)$

$$\rho GS := 0.7500$$

> $initial := Vector(n)$

>
```
x0 := initial :
for i from 1 to 50 do
x0 := B.x0 + h
end do:
evalf (x0), evalf (solution), evalf (x0 − solution)
```

$$initial := \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} 0.5117 \\ 0.8863 \\ 1.0230 \\ 0.8863 \\ 0.5117 \end{bmatrix}, \begin{bmatrix} 0.5117 \\ 0.8863 \\ 1.0230 \\ 0.8863 \\ 0.5117 \end{bmatrix}, \begin{bmatrix} -4.191010^{-7} \\ -6.287010^{-7} \\ -6.287010^{-7} \\ -4.715010^{-7} \\ -2.358010^{-7} \end{bmatrix}$$

2.3. EXAMPLE. The iteration using symbols

> $n := 4; b := 'b':$

> $A, bb := Matrix(\mathrm{n}, \mathrm{n}, symbol = a), \mathrm{Vector}(\mathrm{n}, \mathrm{symbol} = \mathrm{b})$

$$n := 4$$

$$A, bb := \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} \\ a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} \end{bmatrix}, \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix}$$

> $d := Diagonal(A) : diag := Diagonal(d)$

$$diag := \begin{bmatrix} a_{1,1} & 0 & 0 & 0 \\ 0 & a_{2,2} & 0 & 0 \\ 0 & 0 & a_{3,3} & 0 \\ 0 & 0 & 0 & a_{4,4} \end{bmatrix}$$

> $L := LowerTriangle(A) - Diagonal(d) : U := UpperTriangle(A) - Diagonal(d) :$
$N := -U; M := L + diag$

$$N := \begin{bmatrix} 0 & -a_{1,2} & -a_{1,3} & -a_{1,4} \\ 0 & 0 & -a_{2,3} & -a_{2,4} \\ 0 & 0 & 0 & -a_{3,4} \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$M := \begin{bmatrix} a_{1,1} & 0 & 0 & 0 \\ a_{2,1} & a_{2,2} & 0 & 0 \\ a_{3,1} & a_{3,2} & a_{3,3} & 0 \\ a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} \end{bmatrix}$$

> $x0 := 'x0': initial := Vector(n, symbol = x0)$

$$initial := \begin{bmatrix} x0_1 \\ x0_2 \\ x0_3 \\ x0_4 \end{bmatrix}$$

> $Minverse := M^{(-1)} : B := Minverse.N :$

> $h := Minverse.bb :$

Calculating the first coordinate

> $x1[1] := (B.initial + h)[1]$

$$x1_1 := -\frac{a_{1,2}\,x0_2}{a_{1,1}} - \frac{a_{1,3}\,x0_3}{a_{1,1}} - \frac{a_{1,4}\,x0_4}{a_{1,1}} + \frac{b_1}{a_{1,1}}$$

Calculating the second coordinates we use the previously getting better approximation for the first coordinate

> $(B.initial + h)[2] : x1[2] := (collect(\%, a[2,1]))$

$$x1_2 := \left(\frac{a_{1,2}\,x0_2}{a_{1,1}\,a_{2,2}} + \frac{a_{1,4}\,x0_4}{a_{1,1}\,a_{2,2}} - \frac{b_1}{a_{1,1}\,a_{2,2}} + \frac{a_{1,3}\,x0_3}{a_{1,1}\,a_{2,2}}\right) a_{2,1} - \frac{a_{2,3}\,x0_3}{a_{2,2}} + \frac{b_2}{a_{2,2}} - \frac{a_{2,4}\,x0_4}{a_{2,2}}$$

> $coeff\left(x1[2], a_{2,1}\right); simplify\left(\%\cdot a_{2,2} + x1[1]\right)$

$$\frac{a_{1,2}\,x0_2}{a_{1,1}\,a_{2,2}} + \frac{a_{1,4}\,x0_4}{a_{1,1}\,a_{2,2}} - \frac{b_1}{a_{1,1}\,a_{2,2}} + \frac{a_{1,3}\,x0_3}{a_{1,1}\,a_{2,2}}$$

$$0$$

This calculation proves formula for index 2!

$$x1_2 = \frac{1}{a_{2,2}}\left(b_2 - a_{2,1}\cdot x1_1 - \sum_{j=3}^{4} a_{i,j}\cdot x0_j\right)$$

> $(B.initial + h)[3] : x1[3] := (collect(\%, [a[3,1], a[3,2]], distributed))$

$$x1_3 := \left(\frac{a_{1,3}\,x0_3}{a_{1,1}\,a_{3,3}} + \frac{a_{1,2}\,x0_2}{a_{1,1}\,a_{3,3}} - \frac{b_1}{a_{1,1}\,a_{3,3}} + \frac{a_{1,4}\,x0_4}{a_{1,1}\,a_{3,3}}\right) a_{3,1} + \left(-\frac{a_{2,1}\,a_{1,2}\,x0_2}{a_{2,2}\,a_{1,1}\,a_{3,3}} + \left(-\frac{a_{2,1}\,a_{1,3}}{a_{2,2}\,a_{1,1}\,a_{3,3}} + \frac{a_{2,3}}{a_{2,2}\,a_{3,3}}\right) x0_3 + \left(-\frac{a_{2,1}\,a_{1,4}}{a_{2,2}\,a_{1,1}\,a_{3,3}} + \frac{a_{2,4}}{a_{2,2}\,a_{3,3}}\right) x0_4 + \frac{a_{2,1}\,b_1}{a_{2,2}\,a_{1,1}\,a_{3,3}} - \frac{b_2}{a_{2,2}\,a_{3,3}}\right) a_{3,2} + \frac{b_3}{a_{3,3}} - \frac{a_{3,4}\,x0_4}{a_{3,3}}$$

> $coeff\left(x1[3], a_{3,1}\right); simplify\left(\%\cdot a_{3,3} + x1[1]\right)$

$$\frac{a_{1,3}\,x0_3}{a_{1,1}\,a_{3,3}} + \frac{a_{1,2}\,x0_2}{a_{1,1}\,a_{3,3}} - \frac{b_1}{a_{1,1}\,a_{3,3}} + \frac{a_{1,4}\,x0_4}{a_{1,1}\,a_{3,3}}$$

$$0$$

> $coeff\left(x1[3], a_{3,2}\right); simplify\left(\%\cdot a_{3,3} + x1[2]\right)$

$$-\frac{a_{2,1}\,a_{1,2}\,x0_2}{a_{2,2}\,a_{1,1}\,a_{3,3}} + \left(-\frac{a_{2,1}\,a_{1,3}}{a_{2,2}\,a_{1,1}\,a_{3,3}} + \frac{a_{2,3}}{a_{2,2}\,a_{3,3}}\right) x0_3 + \left(-\frac{a_{2,1}\,a_{1,4}}{a_{2,2}\,a_{1,1}\,a_{3,3}} + \frac{a_{2,4}}{a_{2,2}\,a_{3,3}}\right) x0_4 + \frac{a_{2,1}\,b_1}{a_{2,2}\,a_{1,1}\,a_{3,3}} - \frac{b_2}{a_{2,2}\,a_{3,3}}$$

0

So we get relation

$$x1_3 = \frac{\left(b_3 - \sum_{j=1}^{2} a_{3,j} \cdot x1_j - a_{3,4} \cdot x0_4\right)}{a_{3,3}}$$

Finally consider the fourth coordinate

> $(B.initial + h)[4] : x1[4] := (collect(\%, [a[4,1], a[4,2], a[4,3]], distributed))$

$$x1_4 := \left(\frac{a_{1,3}\,x0_3}{a_{1,1}\,a_{4,4}} + \frac{a_{1,2}\,x0_2}{a_{1,1}\,a_{4,4}} - \frac{b_1}{a_{1,1}\,a_{4,4}} + \frac{a_{1,4}\,x0_4}{a_{1,1}\,a_{4,4}}\right) a_{4,1} + \left(\left(-\frac{a_{2,1}\,a_{1,3}}{a_{1,1}\,a_{2,2}\,a_{4,4}} + \frac{a_{2,3}}{a_{2,2}\,a_{4,4}}\right) x0_3 + \left(-\frac{a_{2,1}\,a_{1,4}}{a_{1,1}\,a_{2,2}\,a_{4,4}} + \frac{a_{2,4}}{a_{2,2}\,a_{4,4}}\right) x0_4 + \frac{a_{2,1}\,b_1}{a_{1,1}\,a_{2,2}\,a_{4,4}} - \frac{a_{2,1}\,a_{1,2}\,x0_2}{a_{1,1}\,a_{2,2}\,a_{4,4}} - \frac{b_2}{a_{2,2}\,a_{4,4}}\right) a_{4,2} + \left(\frac{\left(a_{2,1}\,a_{3,2} - a_{2,2}\,a_{3,1}\right) a_{1,2}\,x0_2}{a_{2,2}\,a_{1,1}\,a_{3,3}\,a_{4,4}} + \left(\frac{\left(a_{2,1}\,a_{3,2} - a_{2,2}\,a_{3,1}\right) a_{1,4}}{a_{2,2}\,a_{1,1}\,a_{3,3}\,a_{4,4}} - \frac{a_{3,2}\,a_{2,4}}{a_{3,3}\,a_{2,2}\,a_{4,4}} + \frac{a_{3,4}}{a_{3,3}\,a_{4,4}}\right) x0_4 + \left(\frac{\left(a_{2,1}\,a_{3,2} - a_{2,2}\,a_{3,1}\right) a_{1,3}}{a_{2,2}\,a_{1,1}\,a_{3,3}\,a_{4,4}} - \frac{a_{3,2}\,a_{2,3}}{a_{3,3}\,a_{2,2}\,a_{4,4}}\right) x0_3 + \frac{a_{3,2}\,b_2}{a_{3,3}\,a_{2,2}\,a_{4,4}} - \frac{\left(a_{2,1}\,a_{3,2} - a_{2,2}\,a_{3,1}\right) b_1}{a_{2,2}\,a_{1,1}\,a_{3,3}\,a_{4,4}} - \frac{b_3}{a_{3,3}\,a_{4,4}}\right) a_{4,3} + \frac{b_4}{a_{4,4}}$$

> $coeff\left(x1[4], a_{4,1}\right); simplify\left(\% \cdot a_{4,4} + x1[1]\right)$

$$\frac{a_{1,3}\,x0_3}{a_{1,1}\,a_{4,4}} + \frac{a_{1,2}\,x0_2}{a_{1,1}\,a_{4,4}} - \frac{b_1}{a_{1,1}\,a_{4,4}} + \frac{a_{1,4}\,x0_4}{a_{1,1}\,a_{4,4}}$$

0

> $coeff\left(x1[4], a_{4,2}\right); simplify\left(\% \cdot a_{4,4} + x1[2]\right)$

$$\left(-\frac{a_{2,1}\,a_{1,3}}{a_{1,1}\,a_{2,2}\,a_{4,4}} + \frac{a_{2,3}}{a_{2,2}\,a_{4,4}}\right) x0_3 + \left(-\frac{a_{2,1}\,a_{1,4}}{a_{1,1}\,a_{2,2}\,a_{4,4}} + \frac{a_{2,4}}{a_{2,2}\,a_{4,4}}\right) x0_4 + \frac{a_{2,1}\,b_1}{a_{1,1}\,a_{2,2}\,a_{4,4}} - \frac{a_{2,1}\,a_{1,2}\,x0_2}{a_{1,1}\,a_{2,2}\,a_{4,4}} - \frac{b_2}{a_{2,2}\,a_{4,4}}$$

0

> $coeff\left(x1[4], a_{4,3}\right); simplify\left(\% \cdot a_{4,4} + x1[3]\right)$

$$\frac{\left(a_{2,1}\,a_{3,2} - a_{2,2}\,a_{3,1}\right) a_{1,2}\,x0_2}{a_{2,2}\,a_{1,1}\,a_{3,3}\,a_{4,4}} + \left(\frac{\left(a_{2,1}\,a_{3,2} - a_{2,2}\,a_{3,1}\right) a_{1,4}}{a_{2,2}\,a_{1,1}\,a_{3,3}\,a_{4,4}} - \frac{a_{3,2}\,a_{2,4}}{a_{3,3}\,a_{2,2}\,a_{4,4}} + \frac{a_{3,4}}{a_{3,3}\,a_{4,4}}\right) x0_4 + \left(\frac{\left(a_{2,1}\,a_{3,2} - a_{2,2}\,a_{3,1}\right) a_{1,3}}{a_{2,2}\,a_{1,1}\,a_{3,3}\,a_{4,4}} - \frac{a_{3,2}\,a_{2,3}}{a_{3,3}\,a_{2,2}\,a_{4,4}}\right) x0_3 + \frac{a_{3,2}\,b_2}{a_{3,3}\,a_{2,2}\,a_{4,4}} - \frac{\left(a_{2,1}\,a_{3,2} - a_{2,2}\,a_{3,1}\right) b_1}{a_{2,2}\,a_{1,1}\,a_{3,3}\,a_{4,4}} - \frac{b_3}{a_{3,3}\,a_{4,4}}$$

So

$$x1_4 = \frac{\left(b_4 - \sum_{j=1}^{3} a_{3,j} \cdot x1_j\right)}{a_{3,3}}$$

2.4. EXAMPLE. Gauss-Seidel iteration in component form

The result of the conduct of the Gauss-Seidel iterative algorithm program

$$x_i^{(k+1)} = \frac{\left(b_i - \sum_{j=1}^{i-1} a_{i,j} \cdot x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{i,j} \cdot x_j^{(k)}\right)}{a_{i,i}}, i = 1, 2, \ldots, n$$

2.5. EXAMPLE. One step of Gauss-Seidel-iteration process

Following procedure count only one step of Gauss-Seidel iteration, i.e. using initial value $x^{(0)}$ count the next approximation

$$x^{(1)} = -(L + D)^{-1} \cdot U \cdot x^{(0)} + (L + D)^{-1} \cdot b.$$

```
> GS := proc(x0, A, b)
      local i, j, n , s, x1 :
    n := LinearAlgebra[Dimension](b) :
   x1 := NULL :
for i from 1 to n do
    s := b_i :
    for j from 1 to n do
        if j ≤ (i − 1) then s := s − A_{i,j} · [x1]_j end if:
         if j ≥ (i + 1) then s := s − A_{i,j} · x0_j end if:
    end do:
   x1 := x1, s / A_{i,i}
end do:
 RETURN([x1]) :
end proc:
```

2.6. EXAMPLE. Testing of Gauss-Seidel-iteration for convergence

```
> n := 5; A, b := BandMatrix([−1, 2, −1], 1, n), Vector(n, i→evalf(π^2 · sin(π/(n + 1) · i) / (n + 1)^2))
```

$n := 5$

$$A, b := \begin{bmatrix} 2 & -1 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & -1 & 2 \end{bmatrix}, \begin{bmatrix} 0.1371 \\ 0.2375 \\ 0.2742 \\ 0.2375 \\ 0.1371 \end{bmatrix}$$

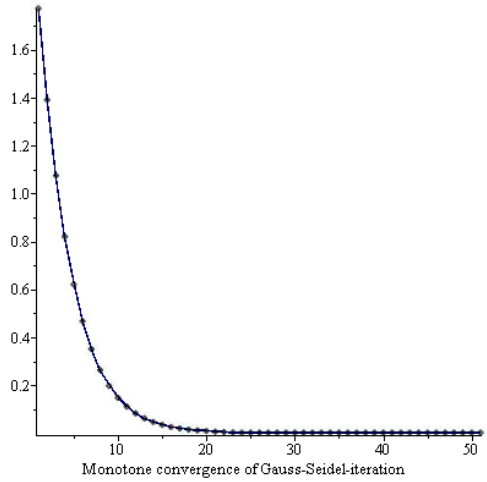
```
> solution:= LinearAlgebra-LinearSolve(A, b)
```

$$solution := \begin{bmatrix} 0.5117 \\ 0.8863 \\ 1.0234 \\ 0.8863 \\ 0.5117 \end{bmatrix}$$

```
> x0 := Vector(n):
>
  deviation := evalf(LinearAlgebra[Norm](solution − Vector(x0), 2)) : differences
       := deviation :
   for i from 1 to 50 do
     x0 := GS(x0, A, b) :
      deviation := evalf(LinearAlgebra[Norm](solution− Vector(x0), 2)) :
     differences := differences , deviation :
   end do:

> konvergenciaGS := [differences ];
  Statistics[LineChart]([differences ], caption
       = "Monotone convergence of Gauss-Seidel-iteration)"
```

konvergenciaGS := [1.7730, 1.3910, 1.0750, 0.8203, 0.6209, 0.4673, 0.3509, 0.2632, 0.1974, 0.1478, 0.1107, 0.0829, 0.0620, 0.0467, 0.0346, 0.0258, 0.0196, 0.0147, 0.0116, 0.0089, 0.0061, 0.0045, 0.0030, 0.0022, 0.0021]



EXAMPLE

1. Increase the value n and examine the convergence speed!

2. Solve system of linear equations by the 2D Laplacian matrix equations using Gauss-Seidel iteration! Compare the speed of convergence wit Jacobi-iteration

3. Relaxation method (SOR)

The relaxation method accelerates the iteration further when relaxation parameter ω is cleverly chosen. The Gauss-Seidel iteration is derived from the use of relaxation.
Let us start with the Gauss-Seidel iteration of the split

$$(L + \mathrm{D})x^{(k+1)} = -Ux^{(k)} + b\ .$$

Transforming the right hand side by adding and subtracting term $\mathrm{D}\cdot x^{(k)}$

$$(L + \mathrm{D})x^{(k+1)} = \mathrm{D}\cdot x^{(k)} - (\mathrm{D} + U)\cdot x^{(k)} + b\ .$$

Multiply the equation from left with the inverse of diagonal matrix D

$$\left(E + \mathrm{D}^{-1}\cdot L\right)\cdot x^{(k+1)} = x^{(k)} - \mathrm{D}^{-1}\cdot(\mathrm{D} + U)\cdot x^{(k)} + \mathrm{D}^{-1}\cdot b$$

$$x^{(k+1)} = x^{(k)} - \mathrm{D}^{-1}\cdot\left(L\cdot x^{(k+1)} + (\mathrm{D} + U)\cdot x^{(k)} - b\right)$$

$$x^{(k+1)} = x^{(k)} - \Delta^{(k)},$$

$$\Delta^{(k)} = \mathrm{D}^{-1}\cdot\left(L\cdot x^{(k+1)} + (\mathrm{D} + U)\cdot x^{(k)} - b\right).$$

This formula is equivalent to the Gauss-Seidel iteration and more evident that the steps (k + 1) -th is used in the previously calculated term $L\cdot x^{(k+1)}$. So

$$\Delta^{(k)} = x^{(k)} - x^{(k+1)}$$

is the difference between the iteration steps. The correction or convergence acceleration idea is using a multiplier factor ω

$$x^{(k+1)} = x^{(k)} - \omega\cdot\Delta^{(k)}\ .$$

It is obtained from the equation backwards to the previous derivation of the Gauss-Seidel iteration relaxation.

$$x^{(k+1)} = x^{(k)} - \omega\cdot\mathrm{D}^{-1}\cdot\left(L\cdot x^{(k+1)} + (\mathrm{D} + U)\cdot x^{(k)} - b\right)$$

$$x^{(k+1)} + \omega\cdot\mathrm{D}^{-1}\cdot\ L\cdot x^{(k+1)} = x^{(k)} - \omega\cdot\left(E + \mathrm{D}^{-1}\cdot U\right)\cdot x^{(k)} + \omega\cdot\mathrm{D}^{-1}\cdot b$$

$$\mathrm{D}^{-1}\cdot\left(\mathrm{D} + \omega\cdot\ L\right)\cdot x^{(k+1)} = \mathrm{D}^{-1}\cdot\left((1-\omega)\cdot\mathrm{D}\ - \omega\cdot U\right)\cdot x^{(k)} + \omega\cdot\mathrm{D}^{-1}\cdot b\ .$$

Multiply the equation from left by matrix $\left(\mathrm{D} + \omega\cdot L\right)^{-1}\cdot\mathrm{D}$

$$x^{(k+1)} = \left(\mathrm{D} + \omega\cdot L\right)^{-1}\cdot\left((1-\omega)\cdot\mathrm{D} - \omega\cdot U\right)\cdot x^{(k)} + \omega\cdot\left(\mathrm{D} + \omega\cdot L\right)^{-1}\cdot b$$

The resulting formula of the successive over relaxation method (SOR), which is the original Gauss-Seidel-iteration when $\omega = 1$. Writing in form $x^{(k+1)} = B\cdot x^{(k)} + h$ we get $B = \left(\mathrm{D} + \omega\cdot L\right)^{-1}\cdot\left((1-\omega)\cdot\mathrm{D} - \omega\cdot U\right), h = \omega\cdot\left(\mathrm{D} + \omega\cdot L\right)^{-1}\cdot b,$ where in splitting $A = M - N$ is $M = \frac{1}{\omega}\mathrm{D} + L$ and $N = \frac{1-\omega}{\omega}\mathrm{D} - U.$

3.1. THEOREM. (Necessary condition of convergence for SOR)
It is necessary to fulfil the condition

$$0 < \omega < 2$$

for the method of SOR.

3.2. THEOREM (Sufficient condition of convergence for SOR)
If the matrix A is symmetric and positive definit (SPD) then the condition $0 < \omega < 2$ is sufficient for the convergence for method SOR.

When the parameter $0 < \omega < 1$ than it is called under-relaxation case and in case $1 < \omega < 2$ is the over-relaxation.

The optimal value of the relaxation parameter $\omega_{optimal} = \frac{2}{1 + \sqrt{1 - (\rho_J)^2}}$, *where* ρ_J is the spectral radius of Jacobi-iteration matrix.

3.3. EXAMPLE. Solution of linear equations with using relaxation method with 1D Laplace-operator matrix

```
> restart : with(LinearAlgebra) : with(ArrayTools) : with(Student[NumericalAnalysis]) :
  with(plots) :
> n := 5; A, b := BandMatrix([-1, 2, -1], 1, n), Vector(n, i -> evalf((Pi^2 * sin(Pi/(n+1) * i))/(n+1)^2))
```

$$n := 5$$

$$A, b := \begin{bmatrix} 2 & -1 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & -1 & 2 \end{bmatrix}, \begin{bmatrix} 0.1371 \\ 0.2375 \\ 0.2742 \\ 0.2375 \\ 0.1371 \end{bmatrix}$$

```
> solution := LinearAlgebra:-LinearSolve(A, b)
```

$$solution := \begin{bmatrix} 0.5117 \\ 0.8863 \\ 1.0234 \\ 0.8863 \\ 0.5117 \end{bmatrix}$$

The speed of convergence depends on the choice parameter $\omega \in (0, 2)$. The optimal choice is $\omega_{opt} := \frac{2}{1 + \sqrt{1 - \rho(J)^2}}$ where $\rho(J)$ is the spectral radius of Jacobi-matrix.

```
> ρJ := 0.8660254040
```

$$\rho J := 0.8660$$

```
> ω := 1.25; ωopt := 2/(1 + sqrt(1 - ρJ^2))
```

$$\omega := 1.2500$$

$$\omega opt := 1.3330$$

```
> ω := ωopt
```

$$\omega := 1.3330$$

```
> d := Diagonal(A) : diag := Diagonal(d);
> L, U := LowerTriangle(A) - Diagonal(d), UpperTriangle(A) - Diagonal(d)
```

$$diag := \begin{bmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 2 \end{bmatrix}$$

$$L, U := \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 \end{bmatrix}, \begin{bmatrix} 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

> $M, N := \frac{1}{\omega} \cdot diag + L, \frac{1-\omega}{\omega} \cdot diag - U$

$$M, N := \begin{bmatrix} 1.5004 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ -1.0000 & 1.5004 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & -1.0000 & 1.5004 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & -1.0000 & 1.5004 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & -1.0000 & 1.5004 \end{bmatrix},$$

$$\begin{bmatrix} -0.4996 & 1.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & -0.4996 & 1.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & -0.4996 & 1.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & -0.4996 & 1.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 & -0.4996 \end{bmatrix}$$

> $Minverz := M^{(-1)};$

$$Minverz := \begin{bmatrix} 0.6665 & 0.0000 & 0.0000 & -0.0000 & -0.0000 \\ 0.4442 & 0.6665 & 0.0000 & -0.0000 & -0.0000 \\ 0.2961 & 0.4442 & 0.6665 & -0.0000 & -0.0000 \\ 0.1973 & 0.2961 & 0.4442 & 0.6665 & -0.0000 \\ 0.1315 & 0.1973 & 0.2961 & 0.4442 & 0.6665 \end{bmatrix}$$

> $B, h := Minverz.N, Minverz.b$

$$B, h := \begin{bmatrix} -0.3330 & 0.6665 & 0.0000 & 0.0000 & 0.0000 \\ -0.2219 & 0.1112 & 0.6665 & 0.0000 & 0.0000 \\ -0.1479 & 0.0741 & 0.1112 & 0.6665 & 0.0000 \\ -0.0986 & 0.0494 & 0.0741 & 0.1112 & 0.6665 \\ -0.0657 & 0.0329 & 0.0494 & 0.0741 & 0.1112 \end{bmatrix}, \begin{bmatrix} 0.0914 \\ 0.2192 \\ 0.3288 \\ 0.3775 \\ 0.3429 \end{bmatrix}$$

Similar deduction using procedure "IterativeFormula" from $Student[NumericalAnalysis]$ package.

> $SOR := IterativeFormula(A, b, method = SOR(\omega), output = ['T', spectralradius], showsteps = true)$

the lower triangular matrix L:

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

the upper triangular matrix U:

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

the diagonal matrix D:

$$\begin{bmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 2 \end{bmatrix}$$

the matrix T and the vector c in the iterative formula x^(k+1)=Tx^(k)+c, T=(D-wL)^(-1)((1-w)D+wU), c=w(D-wL)^(-1)b, w=1.333:

$$\begin{bmatrix} -0.3330 & 0.6665 & 0.0000 & 0.0000 & 0.0000 \\ -0.2219 & 0.1112 & 0.6665 & 0.0000 & 0.0000 \\ -0.1479 & 0.0742 & 0.1112 & 0.6665 & 0.0000 \\ -0.0986 & 0.0494 & 0.0742 & 0.1112 & 0.6665 \\ -0.0657 & 0.0329 & 0.0494 & 0.0742 & 0.1112 \end{bmatrix}, \begin{bmatrix} 0.0914 \\ 0.2193 \\ 0.3289 \\ 0.3775 \\ 0.3428 \end{bmatrix}$$

$$SOR := \begin{bmatrix} -0.3330 & 0.6665 & 0.0000 & 0.0000 & 0.0000 \\ -0.2219 & 0.1112 & 0.6665 & 0.0000 & 0.0000 \\ -0.1479 & 0.0742 & 0.1112 & 0.6665 & 0.0000 \\ -0.0986 & 0.0494 & 0.0742 & 0.1112 & 0.6665 \\ -0.0657 & 0.0329 & 0.0494 & 0.0742 & 0.1112 \end{bmatrix}, 0.3412$$

```
> ρGS := evalf(SOR[2])
```

$$\rho GS := 0.3412$$

3.4. EXAMPLE. Testing of relaxation-iteration for convergence

```
> initial := Vector(n):
  deviation := evalf(LinearAlgebra[Norm](solution - Vector(initial), 2)):
  differences := deviation:

>
  x0 := initial:
  for i from 1 to 50 do
    x0 := B.x0 + h:
    deviation := evalf(LinearAlgebra[Norm](solution - Vector(x0), 2)):
    differences := differences, deviation:
  end do:
  i, evalf(x0), evalf(solution), evalf(deviation)
```

$$51,\ \begin{bmatrix} 0.5117 \\ 0.8863 \\ 1.0230 \\ 0.8863 \\ 0.5117 \end{bmatrix},\ \begin{bmatrix} 0.5117 \\ 0.8863 \\ 1.0230 \\ 0.8863 \\ 0.5117 \end{bmatrix},\ 6.1810\,10^{-16}$$

> *convergenceSOR* := [*differences*]

convergenceSOR := [1.7730, 1.1800, 0.7106, 0.3782, 0.1698, 0.0625, 0.0272, 0.0106, 0.0039, 0.0015, 0.0006, 0.0002, 0.0001, 0.0000, 0.0000, 0.0000, 0.0000, $4.5570\,10^{-7}$, $1.6040\,10^{-7}$, $5.7160\,10^{-8}$, $2.0730\,10^{-8}$, $7.1680\,10^{-9}$, $2.5510\,10^{-9}$, $9.0960\,10^{-10}$, $3.2010\,10^{-10}$, $1.1120\,10^{-10}$, $3.9890\,10^{-11}$, $1.3950\,10^{-11}$, $4.8700\,10^{-12}$, $1.7160\,10^{-12}$, $6.0790\,10^{-13}$, $2.0990\,10^{-13}$, $7.3780\,10^{-14}$, $2.5820\,10^{-14}$, $8.7240\,10^{-15}$, $2.7350\,10^{-15}$, $7.1950\,10^{-16}$, $2.9370\,10^{-16}$, $3.6820\,10^{-16}$, $2.7190\,10^{-16}$, $3.1400\,10^{-16}$, $2.9370\,10^{-16}$, $3.1400\,10^{-16}$, $4.9650\,10^{-16}$, $5.8750\,10^{-16}$, $6.1810\,10^{-16}$, $6.1810\,10^{-16}$, $6.1810\,10^{-16}$, $6.1810\,10^{-16}$, $6.1810\,10^{-16}$, $6.1810\,10^{-16}$]

3.5. EXAMPLE. Compare the speed of convergence for iterative methods Jacobi, Gauss-Seidel and relaxation

The approximations of the three iteration processes are copied here taking 40 steps and starting with the same initial value. The 40 steps of the three iterations are copied here, with the same system of equations and starting value.

>

convergenceJ := [1.772170160595179621.5347444, 1.3291276, 1.1510583, 0.99684572, 0.86329371, 0.74763429, 0.64747028, 0.56072571, 0.48560271, 0.42054429, 0.36420204, 0.31540821, 0.27315153, 0.23655616, 0.20486364, 0.17741712, 0.15364773, 0.13306284, 0.11523580, 0.099797131, 0.086426850, 0.074847848, 0.064820138, 0.056135886, 0.048615104, 0.042101915, 0.036461328, 0.031576436, 0.027345996, 0.023682327, 0.020509497, 0.017761746, 0.015382123, 0.013321309, 0.011536592, 0.0099909814, 0.0086524440, 0.0074932364, 0.0064893329, 0.0056199270, 0.0048669991, 0.0042149445, 0.0036502488, 0.0031612083, 0.0027376867, 0.0023709063, 0.0020532649, 0.0017781794, 0.0015399488, 0.0013336346]

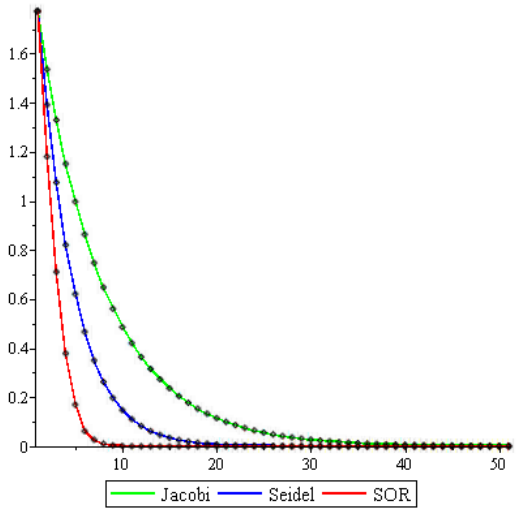
>

convergenceGS := [1.772170161, 1.390912903, 1.074598020, 0.82019672170.620386537, 0.46662024230.35029946160.26280830700.19712716950.14785061270.1108892684, 0.083167278770.062375541270.046781676690.035086262910.026314698940.019736024970.014802018980.011101514240.008326135889.006244602386, 0.00468345215200351258953000.00263444214700.00197583206700.001481874361, 0.001111405611, 0.00083355424850.00062516577950.00046887484450.0003516562411, 0.00026374213300.00019780618490.00014835500480.0001112659944, 0.000083448992890.000062586643970.000046940087790.00003520457176, 0.000026403060980.000019802018900.000014851366510.00001113836842, 0.000008354142406, 0.000006265347599, 0.000004698507793, 0.000003523700908, 0.000002642545853, 0.000001981708880, 0.000001486644825, 0.000001114621634]

> *Statistics*[*LineChart*]([*convergenceJ* , *convergenceGS* , *convergenceSOR*], *color* = [*green* , *blue* , *red*], *legend* = ["Jacobi", "Seidel", "SOR"])

```
convergenceJ := [1.7722, 1.5347, 1.3291, 1.1511, 0.9968, 0.8633, 0.7476, 0.6475, 0.5607, 0.4856,
    0.4205, 0.3642, 0.3154, 0.2732, 0.2366, 0.2049, 0.1774, 0.1536, 0.1331, 0.1152, 0.0998,
    0.0864, 0.0748, 0.0648, 0.0561, 0.0486, 0.0421, 0.0365, 0.0316, 0.0273, 0.0237, 0.0205,
    0.0178, 0.0154, 0.0133, 0.0115, 0.0100, 0.0087, 0.0075, 0.0065, 0.0056, 0.0049, 0.0042,
    0.0037, 0.0032, 0.0027, 0.0024, 0.0021, 0.0018, 0.0015, 0.0013]

convergenceGS := [1.7722, 1.3909, 1.0746, 0.8202, 0.6204, 0.4666, 0.3503, 0.2628, 0.1971,
    0.1479, 0.1109, 0.0832, 0.0624, 0.0468, 0.0351, 0.0263, 0.0197, 0.0148, 0.0111, 0.0083,
    0.0062, 0.0047, 0.0035, 0.0026, 0.0020, 0.0015, 0.0011, 0.0008, 0.0006, 0.0005, 0.0004,
    0.0003, 0.0002, 0.0001, 0.0001, 0.0001, 0.0001, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,
    0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000]
```



>

When we choose correctly the over-relaxation parameter ω then the SOR method gives on a quick way the solution. In general, the convergence speed of Gauss-Seidel -method is slower than SOR and the slowest method is the Jacobi -iteration.

4. Summary

D is the diagonal matrix of the diagonal entries of A,

L is the lower triangular matrix entries of $(-A)$: $l_{i,j}=-a_{i,j}$ if i > j, $l_{i,j}=0$ if i ≤ j, and

U is the upper triangular matrix entries of $(-A)$: $u_{i,j}=-a_{i,j}$ if j > i, $l_{i,j}=0$ if j ≤ i.

As a consequence, $A=\mathrm{D}-(L+U)$.

4.1. Jacobi-method

- in coordinate format

$$x_i^{(k+1)}=\frac{1}{a_{ii}}\left[b_i-\sum_{\substack{j=1\\ j\neq i}}^{n}a_{ij}x_j^{(k)}\right],\quad i=1,\ldots,n.$$

- in matrix format

$$A=\mathrm{D}-(L+U)\quad \cdot B_J=\mathrm{D}^{-1}\cdot(L+U)=I-\mathrm{D}^{-1}\cdot A$$

4.2. Gauss-Seidel -method

- in coordinate format

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left[b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right], \quad i = 1, \ldots, n.$$

- in matrix format

$$A = M - N, \quad M = \mathrm{D} - L \text{ and } \quad N = U, \quad B_{GS} = (\mathrm{D} - L)^{-1} \cdot U$$

4.3. Successive over-relaxation method (or SOR)

- in coordinate format

$$x_i^{(k+1)} = \frac{\omega}{a_{ii}} \left[b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right] + (1 - \omega) x_i^{(k)}$$

- in vector format

$$x^{(k+1)} = x^{(k)} + \left(\frac{1}{\omega} \mathrm{D} - L \right)^{-1} \cdot r^{(k)}$$

5. Exercises

Consider the 3×3 linear systems of the form $A_i \cdot x = b_i$, where b_i is always taken in such a way that the solution of the system will be the unit vector $x = (1, 1, \ldots, 1)^T$, and the matrices A_i are

$$\mathrm{A}_1 = \begin{bmatrix} 3 & 0 & 4 \\ 7 & 4 & 2 \\ -1 & 1 & 2 \end{bmatrix}, \qquad \mathrm{A}_2 = \begin{bmatrix} -3 & 3 & -6 \\ -4 & 7 & -8 \\ 5 & 7 & -9 \end{bmatrix},$$

$$\mathrm{A}_3 = \begin{bmatrix} 4 & 1 & 1 \\ 2 & -9 & 0 \\ 0 & -8 & -6 \end{bmatrix}, \qquad \mathrm{A}_4 = \begin{bmatrix} 7 & 6 & 9 \\ 4 & 5 & -4 \\ -7 & -3 & 8 \end{bmatrix}.$$

(a) It can be checked that the Jacobi method does fail to converge for A_1 (ρ(BJ) =1.33), while the Gauss-Seidel scheme is convergent.

(b) Conversely, in the case of A_2, the Jacobi method is convergent, while the Gauss-Seidel method fails to converge (ρ(BGS) = 1.11).

(c) In the remaining two cases, the Jacobi method is more slowly convergent than the Gauss-Seidel method for matrix A_3 (ρ(BJ) = 0.44 against ρ(BGS) = 0.018), and the converse is true for A_4 (ρ(BJ) = 0.64 while ρ(BGS) = 0.77).

6. Steepest descent method

LARGE-SCALE LINEAR SYSTEMS OF EQUATIONS
Computer Science Engineering Msc
Theoretical and Practical lesson

Consider the system of linear equations

(1) $A \cdot x = b$

with $n \times n$ symmetric positive definite (SPD) matrix A . The aim of the gradient like methods is locating the solution vector $x^* = A^{-1} \cdot b \in \mathbb{R}^n$ of system (1) as the minimum value of the energy function

$$E(x) = \frac{1}{2} \cdot x^T \cdot A \cdot x - x^T \cdot b = \frac{1}{2} \cdot \langle A \cdot x, x \rangle - \langle x, b \rangle \xrightarrow{x \in \mathbb{R}^n} \min$$

where the searching space for vector x is the whole $n-$ dimensional $\mathbb{R}^n$ vector space. The energy function $E(x)$ is quadratic, i.e. second degree in coordinates x_i of vector x . Here and after x^T denote the transposition of the vector x and the (real) scalar product is denoted by $\langle x, y \rangle = \sum_{k=1}^{n} x_k y_k$ between vectors x and y .

1. Gradient vector of a quadratic form and the corresponding linear system of equations

Compute the **gradient vector** $\nabla E(x)$ for the energy function $E(x)$ by using the Maple symbolic derivative capabilities

$$E(x) = \frac{1}{2} \cdot x^T \cdot A \cdot x - x^T \cdot b = \frac{1}{2} \cdot \sum_{i,j=1}^{n} a_{i,j} \cdot x_i \cdot x_j - \sum_{i=1}^{n} b_i \cdot x_i$$

We will show that the **residual** vector r of the linear system (1) is the negative gradient

$$\nabla E(x) = \begin{bmatrix} \frac{\partial E(x)}{\partial x_1} \\ \frac{\partial E(x)}{\partial x_2} \\ \dots \\ \frac{\partial E(x)}{\partial x_{n-1}} \\ \frac{\partial E(x)}{\partial x_n} \end{bmatrix} = A \cdot x - b = -r$$

where the matrix $A = \left(a_{i,j}\right)_{i=1..n}^{j=1..n}$ is symmetric.

Give matrix A in symbolic form and also vectors x and b.

```
> restart : with(LinearAlgebra) :
> n := 3
> A, B, X := Matrix(n, symbol = a, shape = symmetric), Vector(n, symbol = b), Vector(n,
      symbol = x);
  v := op(convert(X, list))
```

$$n := 3$$

$$A, B, X := \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{1,2} & a_{2,2} & a_{2,3} \\ a_{1,3} & a_{2,3} & a_{3,3} \end{bmatrix}, \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}, \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

$$v := x_1, x_2, x_3$$

Define the quadratic energy function $E(x)$!

> $E := unapply\left(simplify\left(\frac{1}{2} \cdot X^{+}.A.X - X^{+}.B\right), v\right)$:

> $E(v)$

$$\frac{1}{2} x_1^2 a_{1,1} + x_1 x_2 a_{1,2} + x_1 x_3 a_{1,3} + \frac{1}{2} x_2^2 a_{2,2} + x_2 x_3 a_{2,3} + \frac{1}{2} x_3^2 a_{3,3} - x_1 b_1 - x_2 b_2 - x_3 b_3$$

Calculate the gradient vector of function $E(x_1, x_2, \ldots, x_n)$ with respect to the variables $x_1, x_2, \ldots, x_n$

$$grad\ E(x) = \nabla E(x) = \begin{bmatrix} \frac{\partial E(x)}{\partial x_1} \\ \frac{\partial E(x)}{\partial x_2} \\ \ldots \\ \frac{\partial E(x)}{\partial x_{n-1}} \\ \frac{\partial E(x)}{\partial x_n} \end{bmatrix}$$

> $gradient := Vector\left(\left[seq\left(\frac{\partial}{\partial x_i} E(v), i = 1..n\right)\right]\right)$

$$gradient := \begin{bmatrix} a_{1,1} x_1 + a_{1,2} x_2 + a_{1,3} x_3 - b_1 \\ a_{1,2} x_1 + a_{2,2} x_2 + a_{2,3} x_3 - b_2 \\ a_{1,3} x_1 + a_{2,3} x_2 + a_{3,3} x_3 - b_3 \end{bmatrix}$$

Create the minus sign to the residual vector $(A \cdot x - b) = -r$ of the linear system (1).

> $linear\_system := A.X - B$

$$linear\_system := \begin{bmatrix} a_{1,1} x_1 + a_{1,2} x_2 + a_{1,3} x_3 - b_1 \\ a_{1,2} x_1 + a_{2,2} x_2 + a_{2,3} x_3 - b_2 \\ a_{1,3} x_1 + a_{2,3} x_2 + a_{3,3} x_3 - b_3 \end{bmatrix}$$

We can see that the two vectors

(1) **the gradient vector** of $\boldsymbol{E(x)}$ and

(2) **the negative residual** $(A \cdot x - b) = -r$ of the linear system (1)

are the same.

> $gradient - linear\_system$

$$\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

2. Minimization of a quadratic function along a line

2.1. Finding the minimum value of quadratic form is equivalent with finding the solution of linear system of equations

Assume that the matrix $A \in \mathbb{R}^{n \times n}$ is symmetric and positive definite (SPD) with size $n \times n$ and $b \in \mathbb{R}^n$ is an $n-$ dimensional vector.

The iterative sequence $x^{(0)}, x^{(1)}, x^{(2)}, ..., x^{(k)}, ...$ is created by **Gradient method** and **Conjugate gradient method** such that the elements will be closer and closer to the exact solution x^* of the system of linear equations

$$A \cdot x = b$$

by finding the minimum value of the energy function

$$E(x) = \frac{1}{2} x^T \cdot A \cdot x - b^T \cdot x$$

along different **lines**

$$L_k = \{x : x = x^{(k)} + t \cdot d^{(k)}, t > 0\}$$

where $x^{(k)}$ is the iterated point, $d^{(k)}$ is the direction at the step k^{th}

$$x^{(k+1)} = \min_{x \in L_k} E(x)$$

THEOREM. 2.2. EXISTENCE AND UNIQUENESS OF THE GLOBAL MINIMUM VALUE OF FUNCTION $E(x)$

Since the Hesse-matrix of the energy function $E(x)$ is

$$Hesse(E(x)) = \left(\frac{\partial^2 E(x)}{\partial x_i \partial x_j} \right)_{i=1..n}^{j=1..n} = Jacobi(\nabla E(x)) = A$$

positive definite matrix, therefore exist exactly one minimum value x^* of the function $E(x)$ which is the unique solution of the system of linear equations $A \cdot x^* = b$.

Elementary Proof.

We give an elegant proof of theorem 2.2 without using differentiation. Denote the solution of system $A \cdot x = b$ by $x = x^*$. Then we have

$$E(x) - E(x^*) = \frac{1}{2} < Ax, x > - \langle b, x\rangle - \frac{1}{2}\langle Ax^*, x^*\rangle + \langle b, x^*\rangle = \frac{1}{2}\langle Ax, x\rangle - (b, x)$$

$$-\frac{1}{2}\langle b, x^*\rangle + \langle b, x^*\rangle =$$

$$= \frac{1}{2}\langle Ax, x\rangle - \frac{1}{2}\cdot\langle b, x\rangle - \frac{1}{2}\cdot\langle b, x\rangle + \frac{1}{2}\cdot\langle b, x^*\rangle = \frac{1}{2}\langle Ax - Ax^*, x\rangle - \frac{1}{2}\cdot\langle b, x - x^*\rangle$$

$$=$$

$$= \frac{1}{2}\langle x - x^*, Ax\rangle - \frac{1}{2}\cdot\langle Ax^*, x - x^*\rangle = \frac{1}{2}\langle A(x - x^*), x - x^*\rangle = \frac{1}{2}\cdot\|x - x^*\|_A.$$

Since $\|x - x^*\|_A \geq 0$ and this norm equals to zero only if $x = x^*$ therefore the relation

$$E(x) \geq E(x^*)$$

is valid for all vector x and the equality holds if and only if $x = x^*$. This means that the minimum value of function $E(x)$ are attained only at the vector $x = x^*$!

2.3. DEFINITION. GRADIENT VECTOR AND THE RESIDUAL VECTOR

The **gradient** vector of function $E(x)$ is

$$\nabla E(x) = A\cdot x - b = -r$$

which is opposite to the residual r **residual** vector.

2.4. DEFINITION. NEW NORM USING MATRIX "A"

With a symmetric and positive definite matrix A can be introduce a new vector norm

$$\|x\|_A := \sqrt{x^T\cdot A\cdot x}$$

the so-called A-norm that fulfils each requirement of the norms e.g. for all $x \in R^n$ $x^T\cdot A\cdot x \geq 0$ and equal to 0 when $x = 0$.

2.5. Linear search for minimum value

The gradient and conjugate gradient method is also based on linear search method in which the search direction is specially chosen. From this reason have to investigate the values $E(x + t\cdot p)$ where $t \in \mathbb{R}_+$ and the vectors x, p are given

The set of points

$$e = \{x + t\cdot p \mid t \in \mathbb{R}\} \subset \mathbb{R}^n$$

in n-dimensional $x \in \mathbb{R}^n$ vector space is line go through point x and parallel with direction vector $p \in \mathbb{R}^n$.

2.5.1. LEMMA.

The quadratic function $E(x + t\cdot p)$ attain their minimum value at location

(2) $$t = t^* = \frac{r^T\cdot p}{p^T\cdot A\cdot p}$$

where $r = b - A\cdot x$ represent the **residual vector** for given vectors x, p and b !

Proof
Consider the second order function for variable t

$$g(t) = E(x + t\cdot p) = \frac{1}{2}(x + t\cdot p)^T\cdot A\cdot(x + t\cdot p) - b^T\cdot(x + t\cdot p) = E(x) + t\cdot(A\cdot x - b)^T\cdot p + \frac{1}{2}t^2\cdot p^T\cdot A\cdot p$$

where the coefficient of member t^2 is $\frac{1}{2}\cdot p^T\cdot A\cdot p > 0,$ if $p \neq 0.$ Therefore the zero root of the derivative with respect to the variable t give the location of the minimum value

$$\frac{\mathrm{d}}{\mathrm{d}\,t}\,g(t) = t\cdot p^T\cdot A\cdot p + (A\cdot x - b)^T\cdot p = 0$$

$$t = \frac{(b - A\cdot x)^T\cdot p}{p^T\cdot A\cdot p} = \frac{r^T\cdot p}{p^T\cdot A\cdot p}$$

Q.e.d.

Exercise 2.6.
Let us prove that the iteration step of the steepest descent method starting from arbitrary vector x_0 and arbitrary direction p_0 and take

$$x_1 = x_0 + \frac{r_0^T\cdot p_0}{p_0^T\cdot A\cdot p_0}\cdot p_0$$

one step of the iteration then we get a point x_1 such that the residual vector $r_1 = b - A\cdot x_1$ is orthogonal to the previous direction p_0 that is

$$r_1 \perp p_0.$$

3. Steepest descent method

It is well known from the multivariate calculus that

(a) **gradient vector** of a multivariate **function** $\boldsymbol{E(x)}$ is pointing to the direction where the function **growing** fastest;

(b) since we are searching for **minimum value** of function $E(x)$ therefore we have to go in **opposite direction** to the gradient vector, which is the **steepest descent** direction.

(c) From definition 2.3 follows that the opposite direction of the gradient vector is the direction of the **residual vector**.

Therefore, the residual vector is the direction pointing to the steepest descent, because of it is opposite to the gradient!
Choose direction vector $p_0 = r_0$ as residual vector at point x_0 and take one step in this direction until function $E(x)$ attain their **minimum value** of along the line

$$\mathrm{p} = r_0 \quad \Rightarrow \quad t^* = \frac{r_0^T\cdot p}{p^T\cdot A\cdot p} = \frac{r_0^T\cdot r_0}{r_0{}^T\cdot A\cdot r_0}.$$

Illustrating this step by using Maple program!

Give the symmetric matrix A, the right hand side vector b and the solution vector x.

```
> restart : with(LinearAlgebra ) :
> n := 3
> A, B, X := Matrix(n, symbol = a, shape = symmetric), Vector(n, symbol = b), Vector(n,
      symbol = x);
> v := op(convert(X, list))
```

$$n := 3$$

$$A, B, X := \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{1,2} & a_{2,2} & a_{2,3} \\ a_{1,3} & a_{2,3} & a_{3,3} \end{bmatrix}, \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}, \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

$$v := x_1, x_2, x_3$$

Define quadratic energy function by $F(x) = \frac{1}{2} x^T \cdot A \cdot x - x^T . b$ and the residual vector $r = b - A \cdot x$!

```
> F := unapply(simplify(1/2 · X^+ .A.X − X^+ .B), v) : F(v);
> r := B − A.X
```

$$\frac{1}{2} x_1^2 a_{1,1} + x_1 x_2 a_{1,2} + x_1 x_3 a_{1,3} + \frac{1}{2} x_2^2 a_{2,2} + x_2 x_3 a_{2,3} + \frac{1}{2} x_3^2 a_{3,3} - x_1 b_1 - x_2 b_2 - x_3 b_3$$

$$r := \begin{bmatrix} -a_{1,1} x_1 - a_{1,2} x_2 - a_{1,3} x_3 + b_1 \\ -a_{1,2} x_1 - a_{2,2} x_2 - a_{2,3} x_3 + b_2 \\ -a_{1,3} x_1 - a_{2,3} x_2 - a_{3,3} x_3 + b_3 \end{bmatrix}$$

Take one step into the direction of the residual

$$x^{(1)} = x^{(0)} + \alpha_0 \cdot r^{(0)}$$

where the step size is given by the parameter $\alpha_0 = \frac{\left(r^{(0)}\right)^T \cdot r^{(0)}}{\left(r^{(0)}\right)^T \cdot A \cdot r^{(0)}}$ for which $F\left(x^{(0)} + \alpha \cdot r^{(0)}\right)$ will be minimal for $\alpha = \alpha_0 > 0$!

Give the **line** as a Maple sequence because of we can get the value F easily restricted onto the line.

```
> line := [seq(x_i, i = 1 ..n)] + convert(α·r, list) : u := op(simplify(line))
```

$$u := -\alpha a_{1,1} x_1 - \alpha a_{1,2} x_2 - \alpha a_{1,3} x_3 + \alpha b_1 + x_1, -\alpha a_{1,2} x_1 - \alpha a_{2,2} x_2 - \alpha a_{2,3} x_3 + \alpha b_2 + x_2, -\alpha a_{1,3} x_1 - \alpha a_{2,3} x_2 - \alpha a_{3,3} x_3 + \alpha b_3 + x_3$$

Restrict the value of F onto the line $u = x + \alpha \cdot r$, where α is a positive parameter!

```
> S := F(u);
```

$$S := \frac{1}{2}\left(-\alpha a_{1,1} x_1 - \alpha a_{1,2} x_2 - \alpha a_{1,3} x_3 + \alpha b_1 + x_1\right)^2 a_{1,1} + \left(-\alpha a_{1,1} x_1 - \alpha a_{1,2} x_2\right.$$
$$\left. - \alpha a_{1,3} x_3 + \alpha b_1 + x_1\right)\left(-\alpha a_{1,2} x_1 - \alpha a_{2,2} x_2 - \alpha a_{2,3} x_3 + \alpha b_2 + x_2\right) a_{1,2} + \left(\right.$$
$$-\alpha a_{1,1} x_1 - \alpha a_{1,2} x_2 - \alpha a_{1,3} x_3 + \alpha b_1 + x_1\left)\right(-\alpha a_{1,3} x_1 - \alpha a_{2,3} x_2 - \alpha a_{3,3} x_3$$
$$\left. + \alpha b_3 + x_3\right) a_{1,3} + \frac{1}{2}\left(-\alpha a_{1,2} x_1 - \alpha a_{2,2} x_2 - \alpha a_{2,3} x_3 + \alpha b_2 + x_2\right)^2 a_{2,2} + \left(\right.$$
$$-\alpha a_{1,2} x_1 - \alpha a_{2,2} x_2 - \alpha a_{2,3} x_3 + \alpha b_2 + x_2\left)\right(-\alpha a_{1,3} x_1 - \alpha a_{2,3} x_2 - \alpha a_{3,3} x_3$$
$$\left. + \alpha b_3 + x_3\right) a_{2,3} + \frac{1}{2}\left(-\alpha a_{1,3} x_1 - \alpha a_{2,3} x_2 - \alpha a_{3,3} x_3 + \alpha b_3 + x_3\right)^2 a_{3,3} - \left(\right.$$
$$-\alpha a_{1,1} x_1 - \alpha a_{1,2} x_2 - \alpha a_{1,3} x_3 + \alpha b_1 + x_1\left)\right. b_1 - \left(-\alpha a_{1,2} x_1 - \alpha a_{2,2} x_2\right.$$
$$\left. - \alpha a_{2,3} x_3 + \alpha b_2 + x_2\right) b_2 - \left(-\alpha a_{1,3} x_1 - \alpha a_{2,3} x_2 - \alpha a_{3,3} x_3 + \alpha b_3 + x_3\right) b_3$$

We can find the minimum value of function $S(\alpha)$ where the derivative equals to 0. Take the derivative $\frac{d}{d\alpha} S(\alpha)$ with respect to variable α.

```
> derivative := diff(S, α);
```

$$derivative := \left(-\alpha a_{1,1} x_1 - \alpha a_{1,2} x_2 - \alpha a_{1,3} x_3 + \alpha b_1 + x_1\right) a_{1,1}\left(-a_{1,1} x_1 - a_{1,2} x_2\right.$$
$$\left. - a_{1,3} x_3 + b_1\right) + \left(-a_{1,1} x_1 - a_{1,2} x_2 - a_{1,3} x_3 + b_1\right)\left(-\alpha a_{1,2} x_1 - \alpha a_{2,2} x_2\right.$$
$$\left. - \alpha a_{2,3} x_3 + \alpha b_2 + x_2\right) a_{1,2} + \left(-\alpha a_{1,1} x_1 - \alpha a_{1,2} x_2 - \alpha a_{1,3} x_3 + \alpha b_1 + x_1\right)\left(\right.$$
$$-a_{1,2} x_1 - a_{2,2} x_2 - a_{2,3} x_3 + b_2\left)\right. a_{1,2} + \left(-a_{1,1} x_1 - a_{1,2} x_2 - a_{1,3} x_3 + b_1\right)\left(\right.$$
$$-\alpha a_{1,3} x_1 - \alpha a_{2,3} x_2 - \alpha a_{3,3} x_3 + \alpha b_3 + x_3\left)\right. a_{1,3} + \left(-\alpha a_{1,1} x_1 - \alpha a_{1,2} x_2\right.$$
$$\left. - \alpha a_{1,3} x_3 + \alpha b_1 + x_1\right)\left(-a_{1,3} x_1 - a_{2,3} x_2 - a_{3,3} x_3 + b_3\right) a_{1,3} + \left(-\alpha a_{1,2} x_1\right.$$
$$\left. - \alpha a_{2,2} x_2 - \alpha a_{2,3} x_3 + \alpha b_2 + x_2\right) a_{2,2}\left(-a_{1,2} x_1 - a_{2,2} x_2 - a_{2,3} x_3 + b_2\right) + \left(\right.$$
$$-a_{1,2} x_1 - a_{2,2} x_2 - a_{2,3} x_3 + b_2\left)\right(-\alpha a_{1,3} x_1 - \alpha a_{2,3} x_2 - \alpha a_{3,3} x_3 + \alpha b_3$$
$$\left. + x_3\right) a_{2,3} + \left(-\alpha a_{1,2} x_1 - \alpha a_{2,2} x_2 - \alpha a_{2,3} x_3 + \alpha b_2 + x_2\right)\left(-a_{1,3} x_1 - a_{2,3} x_2\right.$$
$$\left. - a_{3,3} x_3 + b_3\right) a_{2,3} + \left(-\alpha a_{1,3} x_1 - \alpha a_{2,3} x_2 - \alpha a_{3,3} x_3 + \alpha b_3 + x_3\right) a_{3,3}\left(\right.$$
$$-a_{1,3} x_1 - a_{2,3} x_2 - a_{3,3} x_3 + b_3\left)\right. - \left(-a_{1,1} x_1 - a_{1,2} x_2 - a_{1,3} x_3 + b_1\right) b_1 - \left(\right.$$
$$-a_{1,2} x_1 - a_{2,2} x_2 - a_{2,3} x_3 + b_2\left)\right. b_2 - \left(-a_{1,3} x_1 - a_{2,3} x_2 - a_{3,3} x_3 + b_3\right) b_3$$

Solve equation $\frac{d}{d\alpha} S(\alpha) = 0$ for scalar variable α.

```
> step_size1 := solve(derivative, α)
```

$$step\_size1 := \left(a_{1,1}^2 x_1^2 + 2\,a_{1,1}\,a_{1,2}\,x_1\,x_2 + 2\,a_{1,1}\,a_{1,3}\,x_1\,x_3 + a_{1,2}^2 x_1^2 + a_{1,2}^2 x_2^2 + 2\,a_{1,2}\,a_{1,3}\,x_2\,x_3 + 2\,a_{1,2}\,a_{2,2}\,x_1\,x_2 + 2\,a_{1,2}\,a_{2,3}\,x_1\,x_3 + a_{1,3}^2 x_1^2 + a_{1,3}^2 x_3^2 + 2\,a_{1,3}\,a_{2,3}\,x_1\,x_2 + 2\,a_{1,3}\,a_{3,3}\,x_1\,x_3 + a_{2,2}^2 x_2^2 + 2\,a_{2,2}\,a_{2,3}\,x_2\,x_3 + a_{2,3}^2 x_2^2 + a_{2,3}^2 x_3^2 + 2\,a_{2,3}\,a_{3,3}\,x_2\,x_3 + a_{3,3}^2 x_3^2 - 2\,a_{1,1}\,b_1\,x_1 - 2\,a_{1,2}\,b_1\,x_2 - 2\,a_{1,2}\,b_2\,x_1 - 2\,a_{1,3}\,b_1\,x_3 - 2\,a_{1,3}\,b_3\,x_1 - 2\,a_{2,2}\,b_2\,x_2 - 2\,a_{2,3}\,b_2\,x_3 - 2\,a_{2,3}\,b_3\,x_2 - 2\,a_{3,3}\,b_3\,x_3 + b_1^2 + b_2^2 + b_3^2\right) \Big/ \left(a_{1,1}^3 x_1^2 + 2\,a_{1,1}^2\,a_{1,2}\,x_1\,x_2 + 2\,a_{1,1}^2\,a_{1,3}\,x_1\,x_3 + 2\,a_{1,1}\,a_{1,2}^2\,x_1^2 + a_{1,1}\,a_{1,2}^2\,x_2^2 + 2\,a_{1,1}\,a_{1,2}\,a_{1,3}\,x_2\,x_3 + 2\,a_{1,1}\,a_{1,2}\,a_{2,2}\,x_1\,x_2 + 2\,a_{1,1}\,a_{1,2}\,a_{2,3}\,x_1\,x_3 + 2\,a_{1,1}\,a_{1,3}^2\,x_1^2 + a_{1,1}\,a_{1,3}^2\,x_3^2 + 2\,a_{1,1}\,a_{1,3}\,a_{2,3}\,x_1\,x_2 + 2\,a_{1,1}\,a_{1,3}\,a_{3,3}\,x_1\,x_3 + 2\,a_{1,2}^3\,x_1\,x_2 + 2\,a_{1,2}^2\,a_{1,3}\,x_1\,x_3 + a_{1,2}^2\,a_{2,2}\,x_1^2 + 2\,a_{1,2}^2\,a_{2,2}\,x_2^2 + 2\,a_{1,2}^2\,a_{2,3}\,x_2\,x_3 + 2\,a_{1,2}\,a_{1,3}^2\,x_1\,x_2 + 2\,a_{1,2}\,a_{1,3}\,a_{2,2}\,x_2\,x_3 + 2\,a_{1,2}\,a_{1,3}\,a_{2,3}\,x_1^2 + 2\,a_{1,2}\,a_{1,3}\,a_{2,3}\,x_2^2 + 2\,a_{1,2}\,a_{1,3}\,a_{2,3}\,x_3^2 + 2\,a_{1,2}\,a_{1,3}\,a_{3,3}\,x_2\,x_3 + 2\,a_{1,2}\,a_{2,2}^2\,x_1\,x_2 + 2\,a_{1,2}\,a_{2,2}\,a_{2,3}\,x_1\,x_3 + 2\,a_{1,2}\,a_{2,3}^2\,x_1\,x_2 + 2\,a_{1,2}\,a_{2,3}\,a_{3,3}\,x_1\,x_3 + 2\,a_{1,3}^3\,x_1\,x_3 + 2\,a_{1,3}^2\,a_{2,3}\,x_2\,x_3 + a_{1,3}^2\,a_{3,3}\,x_1^2 + 2\,a_{1,3}^2\,a_{3,3}\,x_3^2 + 2\,a_{1,3}\,a_{2,2}\,a_{2,3}\,x_1\,x_2 + 2\,a_{1,3}\,a_{2,3}^2\,x_1\,x_3 + 2\,a_{1,3}\,a_{2,3}\,a_{3,3}\,x_1\,x_2 + 2\,a_{1,3}\,a_{3,3}^2\,x_1\,x_3 + a_{2,2}^3\,x_2^2 + 2\,a_{2,2}^2\,a_{2,3}\,x_2\,x_3 + 2\,a_{2,2}\,a_{2,3}^2\,x_2^2 + a_{2,2}\,a_{2,3}^2\,x_3^2 + 2\,a_{2,2}\,a_{2,3}\,a_{3,3}\,x_2\,x_3 + 2\,a_{2,3}^3\,x_2\,x_3 + a_{2,3}^2\,a_{3,3}\,x_2^2 + 2\,a_{2,3}^2\,a_{3,3}\,x_3^2 + 2\,a_{2,3}\,a_{3,3}^2\,x_2\,x_3 + a_{3,3}^3\,x_3^2 - 2\,a_{1,1}^2\,b_1\,x_1 - 2\,a_{1,1}\,a_{1,2}\,b_1\,x_2 - 2\,a_{1,1}\,a_{1,2}\,b_2\,x_1 - 2\,a_{1,1}\,a_{1,3}\,b_1\,x_3 - 2\,a_{1,1}\,a_{1,3}\,b_3\,x_1 - 2\,a_{1,2}^2\,b_1\,x_1 - 2\,a_{1,2}^2\,b_2\,x_2 - 2\,a_{1,2}\,a_{1,3}\,b_2\,x_3 - 2\,a_{1,2}\,a_{1,3}\,b_3\,x_2 - 2\,a_{1,2}\,a_{2,2}\,b_1\,x_2 - 2\,a_{1,2}\,a_{2,2}\,b_2\,x_1 - 2\,a_{1,2}\,a_{2,3}\,b_1\,x_3 - 2\,a_{1,2}\,a_{2,3}\,b_3\,x_1 - 2\,a_{1,3}^2\,b_1\,x_1 - 2\,a_{1,3}^2\,b_3\,x_3 - 2\,a_{1,3}\,a_{2,3}\,b_1\,x_2 - 2\,a_{1,3}\,a_{2,3}\,b_2\,x_1 - 2\,a_{1,3}\,a_{3,3}\,b_1\,x_3 - 2\,a_{1,3}\,a_{3,3}\,b_3\,x_1 - 2\,a_{2,2}^2\,b_2\,x_2 - 2\,a_{2,2}\,a_{2,3}\,b_2\,x_3 - 2\,a_{2,2}\,a_{2,3}\,b_3\,x_2 - 2\,a_{2,3}^2\,b_2\,x_2 - 2\,a_{2,3}^2\,b_3\,x_3 - 2\,a_{2,3}\,a_{3,3}\,b_2\,x_3 - 2\,a_{2,3}\,a_{3,3}\,b_3\,x_2 - 2\,a_{3,3}^2\,b_3\,x_3 + a_{1,1}\,b_1^2 + 2\,a_{1,2}\,b_1\,b_2 + 2\,a_{1,3}\,b_1\,b_3 + a_{2,2}\,b_2^2 + 2\,a_{2,3}\,b_2\,b_3 + a_{3,3}\,b_3^2\right)$$

This value of α determine the step size into the direction of residual! This ratio is the same as we calculated by using the residual vector!

> $step\_size2 := \dfrac{r^+ . r}{r^+ . A . r}$

$$step\_size2 := \left(\left(-a_{1,1}x_1 - a_{1,2}x_2 - a_{1,3}x_3 + b_1\right)^2 + \left(-a_{1,2}x_1 - a_{2,2}x_2 - a_{2,3}x_3 + b_2\right.\right.$$
$$\left.\left.-a_{1,3}x_1 - a_{2,3}x_2 - a_{3,3}x_3 + b_3\right)^2\right) \Big/ \left(\left(\left(-a_{1,1}x_1 - a_{1,2}x_2 - a_{1,3}x_3 + b_1\right)a_{1,1}\right.\right.$$
$$+ \left(-a_{1,2}x_1 - a_{2,2}x_2 - a_{2,3}x_3 + b_2\right)a_{1,2} + \left(-a_{1,3}x_1 - a_{2,3}x_2 - a_{3,3}x_3\right.$$
$$\left.\left.+ b_3\right)a_{1,3}\right)\left(-a_{1,1}x_1 - a_{1,2}x_2 - a_{1,3}x_3 + b_1\right) + \left(\left(-a_{1,1}x_1 - a_{1,2}x_2 - a_{1,3}x_3\right.\right.$$
$$\left.\left.+ b_1\right)a_{1,2} + \left(-a_{1,2}x_1 - a_{2,2}x_2 - a_{2,3}x_3 + b_2\right)a_{2,2} + \left(-a_{1,3}x_1 - a_{2,3}x_2\right.\right.$$
$$\left.\left.- a_{3,3}x_3 + b_3\right)a_{2,3}\right)\left(-a_{1,2}x_1 - a_{2,2}x_2 - a_{2,3}x_3 + b_2\right) + \left(\left(-a_{1,1}x_1 - a_{1,2}x_2\right.\right.$$
$$\left.\left.- a_{1,3}x_3 + b_1\right)a_{1,3} + \left(-a_{1,2}x_1 - a_{2,2}x_2 - a_{2,3}x_3 + b_2\right)a_{2,3} + \left(-a_{1,3}x_1\right.\right.$$
$$\left.\left.- a_{2,3}x_2 - a_{3,3}x_3 + b_3\right)a_{3,3}\right)\left(-a_{1,3}x_1 - a_{2,3}x_2 - a_{3,3}x_3 + b_3\right)\Big)$$

> $simplify(step\_size1 - step\_size2)$

$$0$$

We proved by using Maple symbolic tools when we take step size $\alpha = \dfrac{r^T \cdot r}{r^T \cdot A \cdot r}$ in the direction of the residual then we get to minimum position along this direction.
So we take one step into the steepest descent direction.

The algorithm of the **steepest descent**

Step 1. For arbitrary initial value $x^{(0)}$ calculate the value of the initial residual $r^{(0)} = b - A \cdot x^{(0)}$ and let $k = 0$.

Step 2. Continue the following calculations

while $\left|r^{(k)}\right| > eps = 10^{-m}$ **and** $k \le maxiter$

$$\alpha_k = \frac{r^{(k)T} \cdot r^{(k)}}{r^{(k)T} \cdot A \cdot r^{(k)}}$$

$$x^{(k+1)} = x^{(k)} + \alpha_k \cdot r^{(k)}$$

$$k = k + 1$$

$$r^{(k)} = b - A \cdot x^{(k)}$$

end of calculations and return to examine the conditions in **step 2**.

Result is $x^{(k+1)}$

Here is the procedure which is based on the steepest descent method.

```
> GM := proc(A, b, x0, eps, maxiter)
  local r0, r1, difference, k, x, alfa :
    r0 := b - A.x0 : x := x0 :
    difference := LinearAlgebra[Norm](r0) : k := 0 :
   while (difference > eps) and (k ≤ maxiter) do
       alfa := evalf( (r0.r0) / ((r0^+.A).r0) );
       x := x + alfa·r0;
       r0 := b - A.x :
     difference := LinearAlgebra[Norm](r0) :
     k := k + 1 :
    end do:
  RETURN(x, difference, k)
 end proc:
```

Test the procedure for the following symmetric matrix obtained from 1D PDE discretization.

```
> with(LinearAlgebra) : with(ArrayTools) :
> n := 4
> A := BandMatrix([-1, 2, -1], 1, n, storage = rectangular);
```

$$n := 4$$

$$A := \begin{bmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{bmatrix}$$

```
> s := Vector(n, 1); b := A.s
```

$$s := \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

$$b := \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

```
> initial := Vector([seq(k + 1, k = 1..n)])
```

$$initial := \begin{bmatrix} 2 \\ 3 \\ 4 \\ 5 \end{bmatrix}$$

Check the positive definitiveness of A calculating the eigenvalues. Calculate the condition number, too.

```
> ev := evalf(Eigenvalues(A))
```

$$ev := \begin{bmatrix} 0.3820 \\ 2.6180 \\ 1.3820 \\ 3.6180 \end{bmatrix}$$

> $condition\ number = \frac{\max(ev)}{\min(ev)}$; $evalf\,(ConditionNumber\,(A, 2))$

$$condition\ number = 9.4710$$

$$9.4720$$

> $start := time(\,) : sol, deviation, iter := GM(A, b, initial, 0.00001, 100)$; $ellapsed\,time = time(\,) - start$

$$sol, deviation, iter := \begin{bmatrix} 1.0000 \\ 1.0000 \\ 1.0000 \\ 1.0000 \end{bmatrix}, 0.0000, 59$$

$$ellapsed\ time = 0.3300$$

> $start := time(\,) : LinearAlgebra\,[LinearSolve\,](A, b)$; $ellapsed\,time = time(\,) - start$

$$\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

$$ellapsed\ time = 0.0100$$

EXERCISES.

(a) Modify the procedure GM to get the norm of errors in each step and draw these errors.

(b) Modify the procedure GM to get the approximate points of each step and draw the convergence of the sequence of points if n = 2!

(c) Modify the procedure to obtain the residuals of each step and verify that the residuals are orthogonal to each other!

We will give you the solution of part **(b)** of this exercise!

> $restart : with(LinearAlgebra\,) :$

```
> GMb := proc(A, b, x0, eps, maxiter)
  local r0, r1, differences, k, x, alfa, points :
    r0 := b - A.x0 : x := x0 :
    differences := evalf(LinearAlgebra[Norm](r0)) : points := x : k := 0 :
   while (differences > eps) and (k <= maxiter) do
       alfa := evalf( (r0.r0) / (r0^+.A).r0 );
       x := x + alfa*r0;
       r0 := b - A.x :
     differences := evalf(LinearAlgebra[Norm](r0)) :
    points := points, x :
     k := k + 1 :
    end do:
   points := [points] :
  RETURN(x, points, k)
 end proc:
```

Give matrix A and right vector b such that the solution vector is $x^* = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$.

```
> n := 2 :
> A, s := BandMatrix([-1, 2, -1], 1, n, storage = rectangular), Vector(n, 1);
> b := A.s
```

$$A, s := \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$b := \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

The initial value can be selected arbitrarily.

```
> init := Vector([2, 2.5])
```

$$init := \begin{bmatrix} 2 \\ 2.5000 \end{bmatrix}$$

Calculate the approximate solutions for each step.

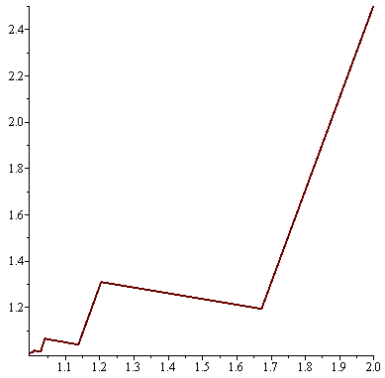
```
> so, points, iter := GMb(A, b, init, 0.001, 100)
```

$$so, points, iter := \begin{bmatrix} 1.0004 \\ 1.0006 \end{bmatrix}, \left[\begin{bmatrix} 2 \\ 2.5000 \end{bmatrix}, \begin{bmatrix} 1.6731 \\ 1.1924 \end{bmatrix}, \begin{bmatrix} 1.2062 \\ 1.3091 \end{bmatrix}, \begin{bmatrix} 1.1386 \\ 1.0395 \end{bmatrix}, \begin{bmatrix} 1.0424 \\ 1.0636 \end{bmatrix}, \begin{bmatrix} 1.0286 \\ 1.0082 \end{bmatrix}, \begin{bmatrix} 1.0087 \\ 1.0131 \end{bmatrix}, \begin{bmatrix} 1.0059 \\ 1.0017 \end{bmatrix}, \begin{bmatrix} 1.0018 \\ 1.0027 \end{bmatrix}, \begin{bmatrix} 1.0012 \\ 1.0003 \end{bmatrix}, \begin{bmatrix} 1.0004 \\ 1.0006 \end{bmatrix} \right], 10$$

Join the points of each consecutive approximation with straight lines on the plane!

```
> map(convert, points, list)
> fig1 := plot(%, thickness = 2) : fig1
```

[[2, 2.5000], [1.6731, 1.1924], [1.2062, 1.3091], [1.1386, 1.0395], [1.0424, 1.0636], [1.0286, 1.0082], [1.0087, 1.0131], [1.0059, 1.0017], [1.0018, 1.0027], [1.0012, 1.0003], [1.0004, 1.0006]]



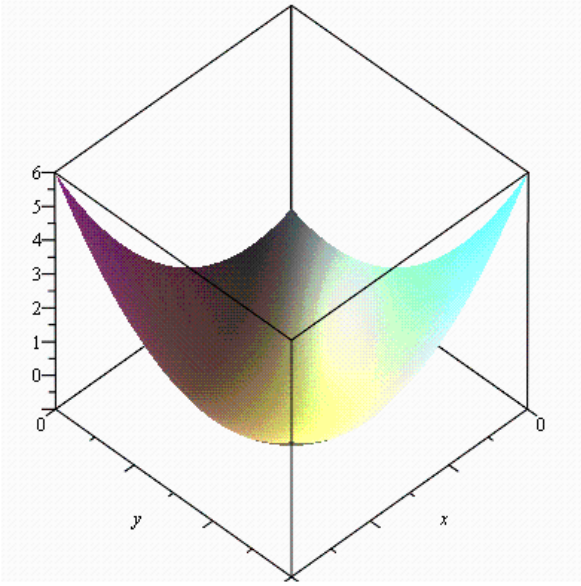
The location and the value of the minimum of the function $F(x,y)$ at point $x=y=1$ and $F(1,1)=-1$.

```
> X := Vector([x,y]):
> F := unapply(expand((1/2)·X^+.A.X - X^+.b), x, y);
  Student[Precalculus]:-CompleteSquare(F(x,y),x): tag1 := op(1,%): maradek := %% - tag1:
  F = tag1 + Student[Precalculus]:-CompleteSquare(maradek)
```

$$F := (x,y) \to x^2 - x\,y + y^2 - x - y$$

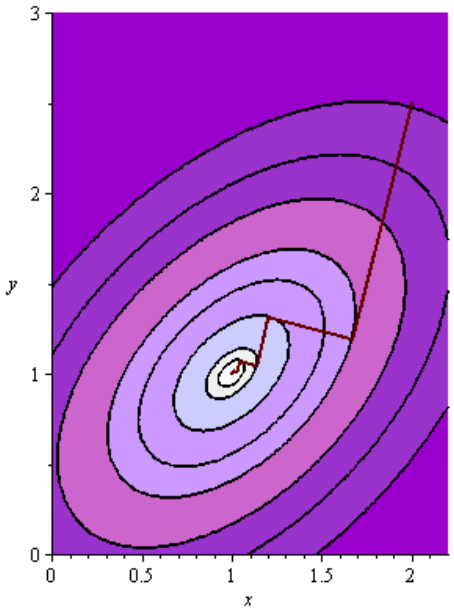
$$F = \left(x - \frac{1}{2}\,y - \frac{1}{2}\right)^2 + \frac{3}{4}\,(y-1)^2 - 1$$

```
> plot3d(F(x,y), x=0..3, y=0..3, axes=boxed)
```



More detailed picture we can get if the contour lines of the function and the pieces of each approximate step are drawn in a common figure!

```
> with(plots):
  fig2 := contourplot(F(x,y), x=0..2.2, y=0..3, contours=[-0.9956,-0.985,-0.923,-0.8,
      -0.645,-0.3, 0.1, 0.7], grid=[100,100], filledregion=true, coloring=["White",
      "DarkViolet"]):
> display(fig2, fig1, scaling=constrained)
```



>

The **geometric interpretation** of the steepest descent method can be understand based on this contour diagram.

(i) The **initial direction is perpendicular** to the **contour line** at the starting point: thanks to choose **residual** direction which is opposite to the direction of **gradient**.

(ii) We take so long step into the **steepest decline** direction (opposite to the gradient) until this straight line **touches one from the contour lines.** The **touch point** will be the **next point** of the iteration.

(iii) From this new point we start in the opposite direction to the gradient. Since the **previous** direction touches the contour passing through the new point and the new direction is **perpendicular** to it, the directions of the two steps are perpendicular to each other.

(iv) We continue the geometric construction from (ii) until the deviations of the successive steps are sufficiently close to each other.

Criticism of the Steepest Descent Method

(a) The opposite direction of the gradient often does not show to the location of the minimum point. The explanation of this is that the contour lines are elongated ellipses as shown in the figure and are not circular.

(b) Each step is perpendicular to the previous step and parallel to the two before. Therefore, the approximation is a series of perpendicular zigzag lines that are ineffective.

Let's experiment with changing the starting point!

7. Conjugate gradient method for systems with a symmetric positive definite matrix

LARGE-SCALE LINEAR SYSTEMS OF EQUATIONS
Computer Science Engineering Msc
Theoretical and Practical lesson

1. Introduction

1.1. DEFINITION. A− conjugacy of two directions

We call the non-zero vectors x and y are **conjugate to each other** with respect to the matrix A if and only if the relation

$$x^T \cdot A \cdot y = 0$$

is fulfilled

1.2. DEFINITION. Set of conjugate directions

We say that the set of directions

$$\{p_0, p_1, p_2, \ldots, p_j\}$$

is a conjugate set of directions with respect to the matrix A if and only if

$$p_i^T \cdot A \cdot p_k = 0$$

for all i and k integers for which

$$0 \le k < i \le j$$

1.3. EXERCISE. Calculate the conjugate direction

Prove that for any vectors $x, p \in \mathbb{R}^n$ (which are not A-conjugate) we can choose the vector

$$y = x + \alpha \cdot p$$

with parameter $\alpha = -\frac{x^T \cdot A \cdot x}{x^T \cdot A \cdot p}$ then the vector y will be $A-$ conjugate to the vector x, i.e.

$$x^T \cdot A \cdot y = 0$$

Solution.

Because of the linearity of scalar product

$$x^T \cdot A \cdot y = x^T \cdot A \cdot (x + \alpha \cdot p) = x^T \cdot A \cdot x + \alpha \cdot x^T \cdot A \cdot p.$$

The vectors x and y are $A-$conjugated, iff $x^T \cdot A \cdot y = 0$. This relation is fulfilled when $\alpha = -\frac{x^T \cdot A \cdot x}{x^T \cdot A \cdot p}$, where the denominator is not zero because of we assume that vectors p and x are not $A-$ conjugated. Q.e.d.

Illustrate this calculation by the model band matrix A!

```
> restart : with(LinearAlgebra) :
> n := 4; A := BandMatrix([-1, 2, -1], 1, n, storage = rectangular);
```

$$n := 4$$

$$A := \begin{bmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{bmatrix}$$

> $evalf(Eigenvalues(A))$

$$\begin{bmatrix} 0.3820 \\ 2.6180 \\ 1.3820 \\ 3.6180 \end{bmatrix}$$

Generate vector p randomly.

> $randomize() : x, p := Vector(n, 1), RandomVector(n, generator = rand(1..10))$

$$x, p := \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 3 \\ 9 \\ 5 \\ 6 \end{bmatrix}$$

Calculate value α and vector y !

> $s1, s2 := x^{+}.A.p, x^{+}.A.x$

$$s1, s2 := 9, 2$$

> $\alpha := -\frac{s2}{s1}$

$$\alpha := -\frac{2}{9}$$

> $y := x + \alpha \cdot p$

$$y := \begin{bmatrix} \frac{1}{3} \\ -1 \\ -\frac{1}{9} \\ -\frac{1}{3} \end{bmatrix}$$

Check the $A-conjugacy$ of vectors x and !

> $x^{+}.A.y$

$$0$$

1.4.DEFINITION A-inner product & A-norm

When the matrix A is symmetric and positive definite then we can define the A-inner product by

$$(y, z)_A = y^T \cdot A \cdot z$$

and the A-norm by $\|y\|_A = \sqrt{(y, y)_A} = \sqrt{y^T \cdot A \cdot y}$.

Exercise. Set of conjugate direction vectors are also linearly independent
Show that the conjugate directions $\{p_0, p_1, p_2, ..., p_j\}$ are linearly independent.

2. A brief summary of the conjugate gradient (CG) method and the Maple procedure

Collect the mathematical requirements in steps.

0. Criterion. Choose a starting vector $x_0 \in \mathbb{R}^n$ arbitrarily and create

$$r_0 = b - A \cdot x_0 \quad \text{and} \quad p_0 = r_0$$

starting residual vector and direction vector. We can assume that $r_0 \neq 0$ because then we found the solution of the system of linear equations $A \cdot x_0 = b$.

Let's suppose we've got the step $j-$ th and compute the following sequences

$x_0, x_1, ..., x_j$ approximating vectors;

$r_0, r_1, ..., r_j$ residual vectors;

$p_0, p_1, ..., p_{j-1}$ direction vectors

such that

$$x_{k+1} = x_k + \alpha_k \cdot p_k \quad \text{for all } k = 0, 1, 2, ..., j-1.$$

We can find an important relation between $r_{j+1} = b - A \cdot x_{j+1}$ and $r_j = b - A \cdot x_j$ i.e. the consecutive residuals

(2.1) $$r_{j+1} = b - A \cdot x_{j+1} = b - A \cdot (x_j + \alpha_j \cdot p_j) = (b - A \cdot x_j) - \alpha_j \cdot (A \cdot p_j) = r_j - \alpha_j \cdot (A \cdot p_j).$$

1. Criterion. Select the direction of the steps so that the residual r_{j+1} will be **perpendicular to the previous** r_j residual. That is, their scalar product should be zero

(*) $$r_{j+1}^T \cdot r_j = r_j^T \cdot r_j - \alpha_j \cdot p_j^T \cdot A \cdot r_j = 0.$$

From which

(2.2) $$\alpha_j = \frac{r_j^T \cdot r_j}{p_j^T \cdot A \cdot r_j}.$$

2. Criterion. Choose the direction of the next step p_{j+1} such that be a linear combination of the previous step p_j and the residual vector r_{j+1}

(2.3) $$p_{j+1} = r_{j+1} + \beta_j \cdot p_j \quad .$$

The residual r_j is similarly written as

$$r_j = p_j - \beta_{j-1} \cdot p_{j-1}$$

The denominator in the expression (2.2) can be written in the following form by substituting r_j from the previous equality

$$p_j^T \cdot A \cdot r_j = p_j^T A \cdot p_j - \beta_{j-1} \cdot p_j^T \cdot A \cdot p_{j-1}.$$

3. Criterion. Make the directions p_j and p_{j-1} such that **will be conjugated to the matrix A,** that is

$$p_j^T \cdot A \cdot p_{j-1} = 0.$$

Therefore

$$p_j^T \cdot A \cdot r_j = p_j^T A \cdot p_j$$

and so the denominator in the formula (2.2) can be rewrite into the form

(2.4) $$\alpha_j = \frac{r_j^T \cdot r_j}{p_j^T \cdot A \cdot p_j}.$$

4. criterion. Determine the value β_j from the formula (2.3) such that the directions p_{j+1} and p_j become A conjugated

$$p_j^T \cdot A \cdot p_{j+1} = p_j^T \cdot A \cdot \left(r_{j+1} + \beta_j \cdot p_j\right) = p_j^T \cdot A \cdot r_{j+1} + \beta_j \cdot p_j^T \cdot A \cdot p_j = 0$$

From which

$$\beta_j = -\frac{p_j^T \cdot A \cdot r_{j+1}}{p_j^T \cdot A \cdot p_j} = -\frac{r_{j+1}^T \cdot A \cdot p_j}{p_j^T \cdot A \cdot p_j}.$$

Using the recursion (2.1)

$$r_{j+1} = r_j - \alpha_j \cdot \left(A \cdot p_j\right) \quad \Rightarrow \quad A \cdot p_j = -\frac{1}{\alpha_j}\left(r_{j+1} - r_j\right).$$

Therefore

$$\beta_j = \frac{1}{\alpha_j} \cdot \frac{\left(r_{j+1} - r_j\right)^T \cdot r_{j+1}}{\left(A \cdot p_j, p_j\right)} = \frac{1}{\alpha_j} \cdot \frac{r_{j+1}^T \cdot r_{j+1}}{p_j^T \cdot A \cdot p_j}$$

since r_j orthogonal to r_{j+1} (see (*) equality).

The equality (2.4) is transformed into relation $\alpha_j \cdot \left(p_j^T \cdot A \cdot p_j\right) = r_j^T \cdot r_j = \|r_j\|^2$ and therefore

(2.5) $$\beta_j = \frac{r_{j+1}^T \cdot r_{j+1}}{r_j^T \cdot r_j} = \frac{\|r_{j+1}\|^2}{\|r_j\|^2}.$$

Summarizing the steps of the conjugate gradient method

(i) $x_0 = 0, r_0 = b, p0 := r0, k := 0$ initialization

(ii) **while** $\|r_k\| > \varepsilon$ **and** $n \le maxiter$ **loop** termination criterion

(iii) $k := k + 1$ counting the number of iterations

(iv) $\alpha_k := \frac{\left(r_{k-1}, r_{k-1}\right)}{\left(A \cdot p_{k-1}, p_{k-1}\right)}$ step size

(v) $x_k := x_{k-1} + \alpha_k \cdot p_{k-1}$ update iterate vector

(vi) $r_k := r_{k-1} - \alpha_k \cdot \left(A \cdot p_{k-1}\right)$ update residual vector

(vii) $\beta_k := \frac{\left(r_k, r_k\right)}{\left(r_{k-1}, r_{k-1}\right)}$ updating the step

(viii) $p_k := r_k + \beta_k \cdot p_{k-1}$ update direction

(ix) check convergence and continue at step (ii) if necessary

Result is x_k.

Procedure for the conjugate gradient method

```
> restart
>
  CG := proc(A, b, x0, eps, maxiter)
     local p0, r0, r1, r0_norm, j, x, alfa, beta :
     r0 := b - A.x0; p0 := r0 : x := x0 :
     r0_norm := LinearAlgebra[Norm](r0) : j := 0 :
    while (r0_norm > eps) and (j ≤ maxiter) do
       alfa := (r0.r0)/(A.p0).p0;
       x := x + alfa·p0;
       r1 := r0 - alfa·(A.p0) :
       beta := (r1.r1)/(r0.r0);
       p0 := r1 + beta·p0;
       r0 := r1 :
      r0_norm := LinearAlgebra[Norm](r0) :
       j := j + 1 :
    end do:
     RETURN(x, r0_norm, j)
  end proc:
```

Try the procedure to the symmetric positive definite tridiagonal matrix obtained from the discretization of 1D elliptic differential equation.

```
>   with(LinearAlgebra) : with(ArrayTools) :
> n := 9;
> A, s := BandMatrix([-1, 2, -1], 1, n, storage = rectangular), Vector(n, 1); b := A.s
```

$$n := 9$$

$$A, s := \begin{bmatrix} 2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 2 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

$$b := \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

The initial vector is generated randomly.

> $randomize(\,) : init := RandomVector(\mathrm{n}, \mathrm{generator} = -5..5)$

$$init := \begin{bmatrix} -2 \\ -4 \\ 5 \\ 2 \\ -5 \\ 2 \\ -4 \\ 5 \\ -4 \end{bmatrix}$$

Check the positive definiteness of matrix A by calculating the eigenvalues.

> $ev := evalf(Eigenvalues(A)); condition\ number = \frac{\max(ev)}{\min(ev)}$

$$ev := \begin{bmatrix} 2.0000 \\ 0.3820 \\ 2.6180 \\ 1.3820 \\ 3.6180 \\ 0.8240 \\ 0.0980 \\ 3.1760 \\ 3.9020 \end{bmatrix}$$

$$condition\ number = 39.8200$$

> $\mathrm{st} := \mathrm{time}(\,) : sol, \mathrm{difference}, iter := CG(A, b, init, 0.0001, 100); T := time(\,) - st$

$$sol, difference, iter := \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}, 0, 9$$

$$T := 0.0800$$

> st := time( ) : *LinearAlgebra* [*LinearSolve*](*A*, *b*); *T2* := *time*( ) − *st*

$$\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

$$T2 := 0.0200$$

> residual= b − A.sol

$$residual = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

2.1. EXERCISE
Modify the procedure CG to get back the following vectors
(1) Check that $r_0, r_1, r_2, \ldots$ sequence of residuals is pair wisely orthogonal to each other!
(2) Check that $p_0, p_1, p_2, \ldots$ sequence of steps directions is pair wisely $A-$ orthogonal (A conjugate) to each other!
(3) Visualize the steps in $n = 2$ dimensional case!

Solution
The modified procedure of the Conjugate Gradient method

> *restart* : *with*(*LinearAlgebra*) :

```
> CGm := proc(A, b, x0, eps, maxiter)
    local p0, r0, r1, res0_norm, j, x, alfa, beta, points, directions, residuals :
    r0 := b - A.x0; p0 := r0 : x := x0 : points := x : directions := p0;
    residuals := r0 : res0_norm := LinearAlgebra[Norm](r0) : j := 0 :
    while (res0_norm > eps) and (j <= maxiter) do
      alfa := (r0.r0)/(A.p0).p0;
      x := x + alfa·p0;
      r1 := r0 - alfa·(A.p0) :
      beta := (r1.r1)/(r0.r0);
      p0 := r1 + beta·p0;
      r0 := r1 :
      res0_norm := LinearAlgebra[Norm](r0) :
      points := points, x :
      directions := directions, p0 :
      residuals := residuals, r0 :
      j := j + 1 :
    end do:
    RETURN(x, [points], [directions], [residuals])
  end proc:
```

Give matrix A and right vector b such that the solution vector will be $x = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ vector.

```
> n := 2 :
> A, s := BandMatrix([-1, 2, -1], 1, n, storage = rectangular), Vector(n, 1);
> b := A.s
```

$$A, s := \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$b := \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

We can choose an initial value vector arbitrarily.

```
> init := Vector([2, 2.5])
```

$$init := \begin{bmatrix} 2 \\ 2.5000 \end{bmatrix}$$

Calculate approximate solutions for each step!

```
> so, points, directions, residuals := CGm(A, b, init, 0.0001, 20)
```

$$so, points, directions, residuals := \begin{bmatrix} 1.0000 \\ 1.0000 \end{bmatrix}, \left[\begin{bmatrix} 2 \\ 2.5000 \end{bmatrix}, \begin{bmatrix} 1.6731 \\ 1.1924 \end{bmatrix}, \begin{bmatrix} 1.0002 \\ 1.0002 \end{bmatrix}, \begin{bmatrix} 1.0000 \\ 1.0000 \end{bmatrix} \right],$$
$$\left[\begin{bmatrix} -0.5000 \\ -2.0000 \end{bmatrix}, \begin{bmatrix} -1.3202 \\ -0.3771 \end{bmatrix}, \begin{bmatrix} -0.0002 \\ -0.0002 \end{bmatrix}, \begin{bmatrix} 0.0000 \\ -0.0001 \end{bmatrix} \right], \left[\begin{bmatrix} -0.5000 \\ -2.0000 \end{bmatrix}, \begin{bmatrix} -1.1538 \\ 0.2883 \end{bmatrix}, \right.$$
$$\left. \begin{bmatrix} -0.0002 \\ -0.0002 \end{bmatrix}, \begin{bmatrix} 0.0001 \\ -0.0001 \end{bmatrix} \right]$$

The perpendicularity is checked for the first two residual vectors because the third residual vector is zero.

> $residuals_1 . residuals_2$

$$3.0000\ 10^{-10}$$

The A- conjugacy should be checked for the first two directions, because of the third direction vector is the zero vector.

> $directions_1 . A . directions_2$

$$7.0000\ 10^{-10}$$

Connect the points of approximations with line segments on the plane!

> $map(convert, points, list)$

> $fig1 := plot(\%, thickness = 3, color = red):$

$$[[2, 2.5000], [1.6731, 1.1924], [1.0002, 1.0002], [1.0000, 1.0000]]$$

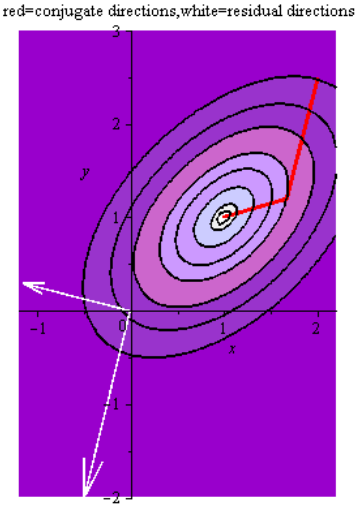
> $X := Vector([x, y]) : E := unapply\left(expand\left(\frac{1}{2} \cdot X^{+} . A . X - X^{+} . b\right), x, y\right)$

$$E := (x, y) \rightarrow x^2 - x\,y + y^2 - x - y$$

>
```
with(plots):
fig2 := contourplot(E(x,y), x=-1.2..2.2, y=-2..3, contours=[-0.9956,-0.985,-0.923,
    -0.8,-0.645,-0.3, 0.1, 0.7], grid=[100,100], filledregion=true, coloring=["White",
    "DarkViolet"]):
```

>
```
arrows := plots[arrow](convert(residuals, list), color=white, shape=arrow, scaling
    =constrained):
```

>
```
display(fig1, fig2, arrows, scaling=constrained, title
    ="red=conjugate directions,white=residual directions")
```



In the second step, the extreme value was precisely received thanks to the choice of conjugate directions.

2.2. EXERCISE

(a) Calculate the sequence of iteration vectors $\boldsymbol{x}^{(1)}$, $\boldsymbol{x}^{(2)}$, $\boldsymbol{x}^{(3)}$ of the linear system of equation

$$A\,\boldsymbol{x} = \begin{bmatrix} 2 & 1 & 0 \\ 1 & 2 & -1 \\ 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} = \boldsymbol{b}$$

using the conjugate gradient method starting from the $\mathbf{x}^{(\mathbf{0})} = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T$ origin.

(b) Show that matrix A is symmetric and positive definite

(c) Demonstrate that the obtained residual vectors $\boldsymbol{r}^{(0)}$, $\boldsymbol{r}^{(1)}, \boldsymbol{r}^{(2)}$ are perpendicular to each other!

(d) Demonstrate that the obtained direction vectors $\boldsymbol{p}^{(0)}$, $\boldsymbol{p}^{(1)}, \boldsymbol{p}^{(2)}$ are $A-$ conjugate to each other!

Tip. Fill the following table

k	$z_k = A \cdot p_k$	$\|r_k\|^2$	$\alpha_k = \dfrac{\|r_k\|^2}{p_k^T \cdot z_k}$	$x_{k+1} = x_k + \alpha_k \cdot p_k$
	$r_{k+1} = r_k - \alpha_k \cdot z_k$	$\|r_{k+1}\|^2$	$\beta_k = \dfrac{\|r_{k+1}\|^2}{\|r_k\|^2}$	$p_{k+1} = r_{k+1} + \beta_k \cdot p_k$

8. Convergence and preconditioning of conjugated gradient method

LARGE-SCALE LINEAR SYSTEMS OF EQUATIONS
Computer Science Engineering Msc
Theoretical and Practical lesson

1. Convergence of conjugate gradient iteration

Consider system of linear equation

$$A \cdot x = b$$

with **symmetric positive definite** matrix A (SPD, i.e. $A^T = A$ and $(Ax, x) > c \cdot \|x\|_2$ where $c > 0$) and the size $N \times N$.

1.1. The steps of conjugate gradient (CG) algorithm

(i) $x_0 = 0, r_0 = b, p0 := r0$ initialize

(ii) loop for $n = 1, 2, ...$ while $\|r_n\|_A > \varepsilon$

(iii) $\alpha_n := \dfrac{(r_{n-1}, r_{n-1})}{(A \cdot p_{n-1}, p_{n-1})}$ step size

(iv) $x_n := x_{n-1} + \alpha_n \cdot p_{n-1}$ approximate solution

(v) $r_n := r_{n-1} - \alpha_n \cdot (A \cdot p_{n-1})$ residual vector

(vi) $\beta_n := \dfrac{(r_n, r_n)}{(r_{n-1}, r_{n-1})}$ improving the step

(vii) $p_n := r_n + \beta_n \cdot p_{n-1}$ new direction

(viii) continue the cycle with condition test at step (ii)

Result in x_n.

Denote the exact solution by $x^* = A^{-1} b$ and the approximate solutions obtained in the conjugate gradient iteration by $x_0, x_1, x_2, ..., x_n, ...$ and the error by $e_n = x^* - x_n$. The method CG is minimizing in each step the A-norm of errors $\|e_n\|_A = \sqrt{e_n^T \cdot A \cdot e_n}$.

In each step of the conjugate gradient method, the A-norms of the errors create a strictly monotonous decrease sequence, the last of which equals to zero.

1.2. THEOREM. Krylov-subspace, the perpendicularity of the residuals and the A-conjugacy of the directions

If during the CG iteration in one step is $r_{n-1} \neq 0$, i.e. the next iteration step can be executed, then the following subspaces will be the same **Krylov-subspace**

$$K_n = span\left(r_0, A \cdot r_0, A^2 \cdot r_0, ..., A^{n-1} \cdot r_0\right) = span\left(x_1, x_2, ..., x_n\right) = span\left(p_0, p_1, ..., p_{n-1}\right)$$
$$= span\left(r_0, r_1, ..., r_{n-1}\right)$$

Furthermore, residuals are perpendicular to the previously calculated residuals

$r_n^T \cdot r_j = 0$, when $j < n$,

and the directions are $A-conjugated$ to the previously calculated directions

$$p_n^T A p_j = 0 \text{ , when } j < n.$$

where span$(x_1, x_2,..., x_n)$ *denote* the subspace spanned by the vectors $x_1, x_2,..., x_n$.

1.3. THEOREM. A monotonicity of A-norm

If during the CG iteration in one step is $r_{n-1} \neq 0$, i.e. the next iteration step can be executed, then the vector x_n is the only vector in subspace K_n , which is minimizing the error $\|e_n\|_A$ $A-$ norm. The convergence is monotone decreasing in $A-$ norm

$$\|e_n\|_A \leq \|e_{n-1}\|_A$$

and equality $e_n = 0$ fulfils for some $n \leq N$, where N is the size of the matrix.

1.4. THEOREM. The order of the convergence

Let us denote by $\kappa = \kappa(A)$ the condition number of symmetric positive definite matrix A in 2-norm. Then we get an estimation on $A-$ norm of the error vector in the n^{th} step

$$\frac{\|e_n\|_A}{\|e_0\|_A} \leq \frac{2}{\left(\frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1}\right)^n + \left(\frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1}\right)^{-n}} \leq 2 \cdot \left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^n$$

1.5. EXAMPLE. Demonstration of the convergence order

Choose symmetric matrix A and vector b in system $A \cdot x = b$ such that

(i) elements of the diagonal of A will be $DIAGONAL(A) = [1, 2, 3,..., n]$

(ii) elements below and above diagonals will be $1 = a_{i,i-1} = a_{i-1,i}$ $(i = 2, 3,..., n)$

(iii) right hand side is $b = [1, 1,..., 1]^T$.

Solve system $A \cdot x = b$ by using **conjugate gradient** solver and

(a) calculate the sequence $x_1, x_2,..., x_n$ and $s_k = \|e_k\|_A = \|x^* - x_k\|_A$ the $A-$ norms of the error vectors, $k = 1, 2, 3,..., n$;

(b) calculate the upper bounds $t_k = \dfrac{2 \cdot \|e_0\|}{\left(\frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1}\right)^k + \left(\frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1}\right)^{-k}}$ with $k = 1, 2, 3,..., n$,

where $\varkappa$ is the condition number of matrix A in 2 norm!

(c) Draw the sequences s_k and t_k in a common figure!

(d) Choose the values for size n the following integers 10, 15, 20 and 25!

(e) How does the condition number and convergence rate change by increasing n?

SOLUTION

```
> restart : with(LinearAlgebra) : with(ArrayTools) :
>
CGstep := proc(A, b, maxiter)
    local p0, r0, r1, rhiba, eps, iter, x, alfa, beta, x0, n, lepesek :
     n := nops(convert(b, list)) :
     x0 := Vector(n, 0) : iter := 0 : r0 := b - A.x0; p0 := r0 : x := x0 :
    rhiba := evalf(sqrt((A.r0).r0)) : eps := 10^(-6) : lepesek := NULL :
   while (rhiba > eps) and (iter ≤ maxiter) do
      alfa := (r0.r0)/(A.p0).p0;
      x := x + alfa·p0;
      r1 := r0 - alfa·(A.p0) :
      beta := (r1.r1)/(r0.r0);
      p0 := r1 + beta·p0;
      r0 := r1 :
     rhiba := evalf(sqrt((A.r0).r0)) :
    iter := iter + 1 :
     lepesek := lepesek, evalf(x)
    end do:
  RETURN([lepesek], evalf(x))
 end proc:
```

Generate matrix A and the right hand side vector b .

```
> n := 20
> A := BandMatrix([1, 0, 1], 1, n, storage = rectangular) :
> for i from 1 to n do A[i, i] := i end do: b := Vector(n, 1) : A, b
```

$$n := 20$$

$$\left[\begin{array}{c} 20\ x\ 20\ \ Matrix \\ Data\ Type:\ anything \\ Storage:\ rectangular \\ Order:\ Fortran\_order \end{array}\right], \left[\begin{array}{c} 1\ ..\ 20\ \ Vector_{column} \\ Data\ Type:\ anything \\ Storage:\ rectangular \\ Order:\ Fortran\_order \end{array}\right]$$

Calculate the condition number as the ratio between the greatest and the smallest eigenvalues.

```
> st := time( ) : ϰ := evalf(ConditionNumber(A, 2)); "ellapsed time" = time( ) - st
```

$$ϰ := 81.7404$$

$$"ellapsed\ time" = 3.9470$$

```
> se := evalf(Eigenvalues(A)); big := max(se); small := min(se); ϰ := big/small;
> q := (sqrt(ϰ) + 1)/(sqrt(ϰ) - 1)
```

$$se := \left[\begin{array}{c} 1\ ..\ 20\ \ Vector_{column} \\ Data\ Type:\ anything \\ Storage:\ rectangular \\ Order:\ Fortran\_order \end{array}\right]$$

$$big := 20.7462$$

$$small := 0.2538$$

$$\varkappa := 81.7404$$

$$q := 1.2487$$

Give the sequence of approximations and the exact solution.

> approx, solCG := $CGstep(A, b, 50)$:

Calculating the error vectors and their $A-norms$.

> $errors := [seq([k, (A.(solCG - approx[k])).(solCG - approx[k])], k = 1..numelems(approx))]$

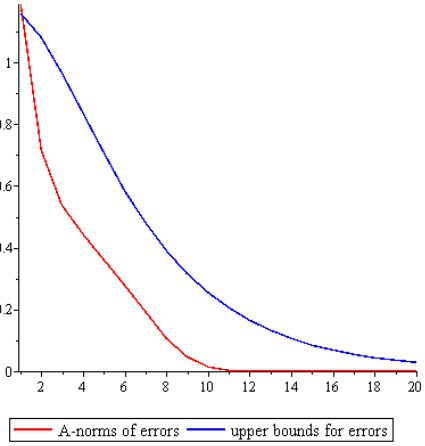
$$errors := [[1, 1.1886], [2, 0.7154], [3, 0.5376], [4, 0.4411], [5, 0.3622], [6, 0.2795], [7, 0.1901], [8, 0.1062], [9, 0.0462], [10, 0.0155], [11, 0.0041], [12, 0.0009], [13, 0.0002], [14, 0.0000], [15, 0.0000], [16, 2.8116\,10^{-7}], [17, 1.9638\,10^{-8}], [18, 8.9948\,10^{-10}], [19, 2.0273\,10^{-11}], [20, 0.0000]]$$

Calculating the upper bounds for the $A-norms$ of errors.

> $bounds := \left[seq\left(\left[n, \frac{2 \cdot errors[1][2]}{q^n + q^{-n}}\right], n = 1..nops(approx)\right)\right]$

$$bounds := [[1, 1.1599], [2, 1.0803], [3, 0.9661], [4, 0.8363], [5, 0.7064], [6, 0.5862], [7, 0.4807], [8, 0.3909], [9, 0.3162], [10, 0.2549], [11, 0.2050], [12, 0.1646], [13, 0.1320], [14, 0.1058], [15, 0.0848], [16, 0.0680], [17, 0.0544], [18, 0.0436], [19, 0.0349], [20, 0.0280]]$$

> $plot([errors, bounds], color = [red, blue], legend = [$ "A−norms of errors", "upper bounds for errors"])



> $solution := evalf(LinearAlgebra[LinearSolve](A, b))$; differences = convert(solCG − solution, list)

$$solution := \begin{bmatrix} 1 .. 20 \ \ Vector_{column} \\ Data\ Type:\ anything \\ Storage:\ rectangular \\ Order:\ Fortran\_order \end{bmatrix}$$

$$differences = [0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000]$$

1.6. EXERCISE. Convergence studies depending on condition number and size

Generate random spares and symmetric matrix with size $n \times n$ based on the following algorithm.

(i) The value of diagonal elements should be 1.

(ii) The non-diagonal elements are formed uniformly distributed from the interval $[-1, 1]$!

(iii) Replace with 0 all non-diagonal elements to which $|a_{i,j}| > \tau$.

(iv) Let the right hand side vector $b = [1, 1,..., 1]$.

Choose value τ from interval $(0,1)$!

(a) Observe what happens

with the number of zero elements of the resulting matrix;

with the condition number of the matrix

when τ is increasing.

(b) Investigate the speed of convergence of the residuals in $A-$ norm when values are $\tau = 0.2,0.15$ and 0.01

(c) Let's change value n between 120 and 200. Calculate the elapsed computer times!

Tips. Generating random matrices using the following command

```
> with(RandomTools) : A := Matrix(n, n, shape = symmetric, Generate(float(range = -1 ..1,
      digits = 7, method = uniform), makeproc = true));
```

$$A := \begin{bmatrix} 20 \text{ x } 20 \text{ Matrix} \\ \text{Data Type: anything} \\ \text{Storage: triangular}_{upper} \\ \text{Order: Fortran\_order} \end{bmatrix}$$

2. Preconditioning the conjugate gradient method and the convergence

Consider system of linear equation with $n \times n$ size

(2.1) $$A \cdot x = b$$

and multiply the system with an inverse matrix M^{-1} from left hand side

(2.2) $$M^{-1}A \cdot x = M^{-1} \cdot b$$

The solutions are the same for these two systems.

However it is more convenient to solve (2.2) than (2.1) in numerical point of view when the relation between condition numbers

$$\kappa\left(M^{-1}A\right) < \kappa(A).$$

is fulfilled. That is, this modification is resulted an improvement in condition number, so it is reasonable to perform such a transformation before applying the CG method.

2.1. DEFINITION.

Matrix $\boldsymbol{M}$ is called a **preconditioning matrix** if the condition number $\kappa(M^{-1}A)$ of the transformed system (2.2) is smaller than the condition number $\kappa(A)$ of the original linear system (2.1).

A good preconditioning matrix is expected to accelerate the approximation solution of a large linear system of equations, with fewer steps to achieve the same accuracy.
Matrix M must be symmetric and positive definite, and easily invertable.
Two extreme cases for selecting the matrix M
(a) $M = A$
(b) $M = I$
are uninteresting because in case (a) it is difficult to determine the inverse of M, (b) in this case we don't use preconditioning. Really, you have to select a matrix M whose invertation contributes to the inverse calculation of A .
Note that finding a good preconditioning matrix does not have a clearly defined method, this can be determined by knowing matrix A .
Let the preconditioning matrix M is given in the form

$$M^{-1} = \mathrm{L} \cdot \mathrm{L}^T$$

which is the so called Cholesky-factorization form, where the lower triangular matrix L is invertible then L^T the transpose of L is an upper triangular matrix.
So the form of system (2.2)

$$M^{-1} \cdot A \cdot x = L \cdot L^T \cdot A \cdot x = L \cdot L^T \cdot b,$$

and multiply here by matrix L^{-1} from left hand side

$$L^T \cdot A \cdot x = L^T \cdot b$$

$$\left(L^T \cdot A \cdot L\right) \cdot \left(L^{-1} x\right) = L^T b$$

Therefore get another

$$\tilde{A} \cdot \tilde{x} = \tilde{b} \qquad (2.3)$$

linear system of equation, where the matrix $\tilde{A} = L^T A L$ is symmetric and positive definite, with new vectors $\tilde{x} = L^{-1} x$ and $\tilde{b} = L^T b$.
The algorithm bellow is the application of conjugate gradient method to the modified system (2.3)

2.2. Algorithmic Steps of the Preconditioned Conjugate Gradient (PCG) method

(i) $x_0 = 0, r_0 = b, z_0 = M^{-1} r_0, p0 = z0$ initialize

(ii) loop for $k = 1, 2, \ldots, maxiter$ while $\|r_k\|_A > \varepsilon$

(iii) $\alpha_k := \dfrac{\left(r_{k-1}, z_{k-1}\right)}{\left(A \cdot p_{k-1}, p_{k-1}\right)}$ step size

(iv) $x_k := x_{k-1} + \alpha_k \cdot p_{k-1}$ improving the approximate solution

(v) $r_k := r_{k-1} - \alpha_k \cdot \left(A \cdot p_{k-1}\right)$ improving residual vector

(vi) $z_k = M^{-1} \cdot r_k$ preconditioning

(vii) $\beta_k := \dfrac{\left(z_k, r_k\right)}{\left(z_{k-1}, r_{k-1}\right)}$ correction factor of gradient

(viii)	$p_k := z_k + \beta_k \cdot p_{k-1}$	select new direction
(ix)	end of the cycle check condition in step (ii)	

The result is x_k.

2.3. EXAMPLE

Write the solver program (procedure) in Maple for preconditioning of conjugate gradient method. Let's try the solver for the following preconditioners

(a) with matrix $M = I$ with identity matrix, which is the original conjugate gradient method without preconditioning

(b) with matrix $M = \mathrm{D} = diag(A)$:which is the **Jacobi-type** preconditioner

(c) with matrix $M = (L + \mathrm{D}) \cdot \mathrm{D}^{-1} \cdot (L^T + \mathrm{D})$: which is the symmetric **Gauss-Seidel-type** preconditioner

(d) with matrix $M = \left(L + \frac{\mathrm{D}}{\omega}\right) \cdot \frac{\omega}{2-\omega} \cdot \mathrm{D}^{-1} \cdot \left(L^T + \frac{\mathrm{D}}{\omega}\right), 0 < \omega < 2,$ which is the symmetric successive relaxation-type preconditioner.

Here L is the lower triangular matrix and the D is the diagonal matrix in splitting $A = L + \mathrm{D} + L^T$!

Choose size of the matrix $n = 50$.

SOLUTION

```
> restart
>
  PCG := proc(A, b, M)
      local n, r, iter, maxiter, x, z, p, eps, rhiba, alfa, beta, S, Mi :
      Digits := 20 : n := nops(convert(b, list)) : maxiter := max(100, n) :
      x := Vector(n, 0) : iter := 0 :
      r := evalf(b) : Mi := evalf(M^(-1)) :
      z := Mi.r :
      p := z :
      eps := 10^(-7) : rhiba := evalf(sqrt(r.r)) :
    while (rhiba > eps) and (iter ≤ maxiter) do
      iter := iter + 1 :
      S := r.z :
      alfa := S/(A.p).p;
      x := x + alfa·p;
      r := r - alfa·(A.p) :
      z := Mi.r :
      beta := r.z/S;
      p := z + beta·p;
      rhiba := evalf(sqrt(r.r)) :
    end do:
   RETURN(x, rhiba, iter)
  end proc:

> with(LinearAlgebra) : with(ArrayTools) :
> n := 50
```

```
> A := BandMatrix([1,-4, 6,-4, 1], 2, n, storage = rectangular)
```

$$n := 50$$

$$A := \begin{bmatrix} 50\ x\ 50\ \ Matrix \\ Data\ Type:\ anything \\ Storage:\ rectangular \\ Order:\ Fortran\_order \end{bmatrix}$$

```
> #kA :=evalf(ConditionNumber(A, 2))# this calculation takes so many time!!
> b := Vector([1, seq(0, k = 1 ..n - 1)])
```

$$b := \begin{bmatrix} 1\ ..\ 50\ \ Vector_{column} \\ Data\ Type:\ anything \\ Storage:\ rectangular \\ Order:\ Fortran\_order \end{bmatrix}$$

(a) with matrix $M = I$ with identity matrix, which is the original conjugate gradient method without preconditioning

```
> M0 := Matrix(n, n, shape = identity)
```

$$M0 := \begin{bmatrix} 50\ x\ 50\ \ Matrix \\ Data\ Type:\ anything \\ Storage:\ empty \\ Order:\ Fortran\_order \end{bmatrix}$$

```
> mo0, r_norm0, iter0 := PCG(A, b, M0)
```

$$mo0, r\_norm0, iter0 := \begin{bmatrix} 1\ ..\ 50\ \ Vector_{column} \\ Data\ Type:\ sfloat \\ Storage:\ rectangular \\ Order:\ Fortran\_order \end{bmatrix}, 9.7643\,10^{-8}, 83$$

```
> Norm(evalf(LinearSolve(A, b)) - mo0, 2)
```

$$1.0567\,10^{-7}$$

(b) with matrix $M = \mathrm{D} = diag(A)$:which is the **Jacobi-type** preconditioner

```
> M1 := Matrix(n, shape = diagonal, Diagonal(A))
```

$$M1 := \begin{bmatrix} 50\ x\ 50\ \ Matrix \\ Data\ Type:\ anything \\ Storage:\ diagonal \\ Order:\ Fortran\_order \end{bmatrix}$$

```
> sol1, r_norm1, iter1 := PCG(A, b, M1)
```

$$sol1, r\_norm1, iter1 := \begin{bmatrix} 1\ ..\ 50\ \ Vector_{column} \\ Data\ Type:\ sfloat \\ Storage:\ rectangular \\ Order:\ Fortran\_order \end{bmatrix}, 7.0139\,10^{-8}, 83$$

```
> Norm(evalf(LinearSolve(A, b)) - sol1, 2)
```

$$5.0408\,10^{-8}$$

The speed of convergence has **not changed**.

(c) with matrix M= $(L+\mathrm{D})\cdot \mathrm{D}^{-1}\cdot\left(L^{T}+\mathrm{D}\right)$: which is the symmetric **Gauss-Seidel-type** preconditioner

> $M2 := LowerTriangle(A).Matrix(n, shape = diagonal, Diagonal(A))^{(-1)}$
> $.LowerTriangle(A)^{+}$

$$M2 := \begin{bmatrix} 50\ x\ 50\ \ Matrix \\ Data\ Type:\ anything \\ Storage:\ rectangular \\ Order:\ Fortran\_order \end{bmatrix}$$

> $sol2, r\_norm2, iter2 := PCG(A, b, M2)$

$$sol2, r\_norm2, iter2 := \begin{bmatrix} 1\ ..\ 50\ \ Vector_{column} \\ Data\ Type:\ sfloat \\ Storage:\ rectangular \\ Order:\ Fortran\_order \end{bmatrix}, 1.2548\,10^{-9}, 48$$

> $Norm(evalf(LinearSolve(A, b)) - sol2, 2)$

$$1.7821\,10^{-9}$$

With this preconditioner, we received the same accuracy results within fewer steps: $48 << 83,$. when $n = 50$.

(d) with matrix $M=\left(L+\frac{\mathrm{D}}{\omega}\right)\cdot\frac{\omega}{2-\omega}\cdot\mathrm{D}^{-1}\cdot\left(L^{T}+\frac{\mathrm{D}}{\omega}\right), 0<\omega<2,$ which is the symmetric successive relaxation type preconditioner.

> $\omega := 1.5; diag := Matrix(n, shape = diagonal, Diagonal(A)) : L := LowerTriangle(A)$
> $- diag$

$$\omega := 1.5000$$

$$L := \begin{bmatrix} 50\ x\ 50\ \ Matrix \\ Data\ Type:\ anything \\ Storage:\ rectangular \\ Order:\ Fortran\_order \end{bmatrix}$$

> $M3 := \frac{\omega}{2-\omega}\cdot\left(L+\frac{diag}{\omega}\right).diag^{-1}.\left(L^{+}+\frac{diag}{\omega}\right)$

$$M3 := \begin{bmatrix} 50\ x\ 50\ \ Matrix \\ Data\ Type:\ float_{8} \\ Storage:\ rectangular \\ Order:\ Fortran\_order \end{bmatrix}$$

> $sol3, r\_norm3, iter3 := PCG(A, b, M3)$

$$sol3, r\_norm3, iter3 := \begin{bmatrix} 1 .. 50 \ \ Vector_{column} \\ Data\ Type:\ sfloat \\ Storage:\ rectangular \\ Order:\ Fortran\_order \end{bmatrix}, 3.5034\,10^{-8}, 39$$

Fewer steps had to be taken to achieve the same result: $39 < 48 << 83$.

Let's look at an example with large-scale and spare matrices!

2.4. EXAMPLE. Preconditioning of system with large -size and spare symmetric positive definite matrix

Consider matrix A $n \times n$ for large-scale n getting

(i) the elements in the main diagonal are defined by $a_{i,\,i} = 0.5 + \sqrt{i}$;

(ii) there are 1 values below and upper of the main diagonal, i.e. $a_{i,\,j} = 1$ if $|i - j| = 1$;

(iii) there are 1 values below and upper of the main diagonal shifted the indices by 100 , i.e. $a_{i,\,j} = 1$ if $|i - j| = 100$;

(iv) the right hand side vector $b = [1, 1, \ldots, 1]$

Investigate the

(a) CG iteration without preconditioning, when matrix M=E is the identity,

(b) CG iteration preconditioning with matrix $\mathrm{M}=(L + \mathrm{D}) \cdot \mathrm{D}^{-1} \cdot \left(L^T + \mathrm{D}\right)$.

Compare the **elapsed times** and **number of iteration** when choose the size $n = 200, 400$ and 1000.

Solution

```
> restart : with(LinearAlgebra) : with(ArrayTools) :
```

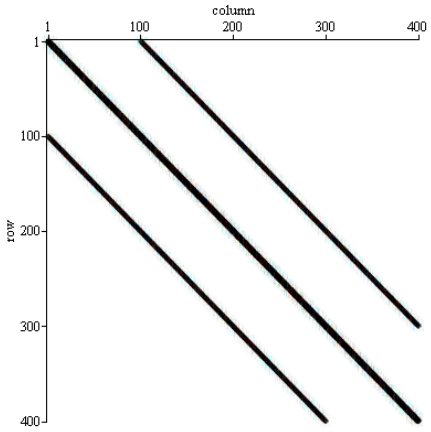
```
PCGstep := proc(A, b, M, maxiter)
    local n, r, iter, x, z, p, eps, rhiba, alfa, beta, zregi, nb, S, Mi, steps :
    n := nops(convert(b, list)) :
    x := Vector(n, 0) : iter := 0 :
     r := evalf(b) :
     z := evalf(LinearSolve(M, r)) :
     p := z :
     eps := 10^(-8) : rhiba := evalf(sqrt(r.r)) :
     steps := NULL :
  while (rhiba > eps) and (iter ≤ maxiter) do
      iter := iter + 1 :
      S := r.z :
      alfa := evalf(S/(A.p).p);
      x := x + alfa·p;
      r := r - alfa·(A.p) :
      z := LinearSolve(M, r) :
      beta := evalf(r.z/S);
      p := z + beta·p;
      rhiba := evalf(sqrt(r.r)) :
      steps := steps, [iter, rhiba]
    end do:
 RETURN(steps)
end proc:
```

```
> n := 400;
> A := Matrix(n, n) :
>
  for i from 1 to n do
    for j from 1 to n do
      if i = j then A[i,j] := evalf(0.5 + sqrt(i)) end if:
      if abs(i - j) = 1 or abs(i - j) = 100 then A[i,j] := 1 end if:
    end do:
  end do:
> A
```

$$n := 400$$

$$\begin{bmatrix} 400 \text{ x } 400 \text{ Matrix} \\ \text{Data Type: anything} \\ \text{Storage: rectangular} \\ \text{Order: Fortran\_order} \end{bmatrix}$$

```
> plots[sparsematrixplot](A, 'matrixview');
```




```
> b := Vector(n, 1)
```

$$b := \begin{bmatrix} 1 .. 400 \text{ Vector}_{column} \\ \text{Data Type: anything} \\ \text{Storage: rectangular} \\ \text{Order: Fortran\_order} \end{bmatrix}$$

(a) CG iteration **without preconditioning**, when matrix $M = E$ is the identity matrix

```
> M0 := Matrix(n, n, shape = identity) :
> ev := evalf(Eigenvalues(A)); big := max(Re(ev)); small := min(Re(ev)); ϰ := big/small
```

$$ev := \begin{bmatrix} 1 .. 400 \text{ Vector}_{column} \\ \text{Data Type: complex}_{8} \\ \text{Storage: rectangular} \\ \text{Order: Fortran\_order} \end{bmatrix}$$

$$big := 22.6732$$

$$small := 0.1990$$

$$\varkappa := 113.9142$$

> st := time( ) : *approx0* := [*PCGstep*(*A*, *b*, *M0*, 100)]; *time0* := *time*( ) − *st*

approx0 := [[1, 5.5280], [2, 2.3476], [3, 1.1209], [4, 0.6320], [5, 0.4265], [6, 0.3604], [7, 0.3422], [8, 0.3341], [9, 0.3182], [10, 0.3029], [11, 0.2800], [12, 0.2503], [13, 0.2113], [14, 0.1703], [15, 0.1303], [16, 0.0960], [17, 0.0688], [18, 0.0480], [19, 0.0325], [20, 0.0215], [21, 0.0141], [22, 0.0090], [23, 0.0057], [24, 0.0035], [25, 0.0022], [26, 0.0013], [27, 0.0008], [28, 0.0005], [29, 0.0003], [30, 0.0002], [31, 0.0001], [32, 0.0001], [33, 0.0000], [34, 0.0000], [35, 0.0000], [36, 0.0000], [37, 0.0000], [38, 0.0000], [39, $7.3155\,10^{-7}$], [40, $3.8507\,10^{-7}$], [41, $2.0099\,10^{-7}$], [42, $1.0330\,10^{-7}$], [43, $5.3352\,10^{-8}$], [44, $2.7244\,10^{-8}$], [45, $1.3846\,10^{-8}$], [46, $6.9119\,10^{-9}$]]

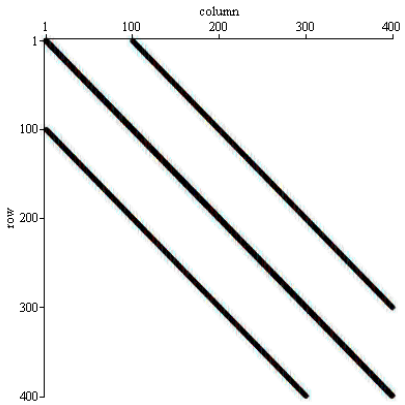
time0 := 3.1200

(b) CG iteration **preconditioning** with matrix $M = (L + \mathrm{D}) \cdot \mathrm{D}^{-1} \cdot (L^T + \mathrm{D})$

> *M1* := *LowerTriangle*(*A*).*Matrix*(*n*, *shape* = *diagonal*, *Diagonal*(*A*))^(−1) .*LowerTriangle*(*A*)^+

$$M1 := \begin{bmatrix} 400\ x\ 400\ \ Matrix \\ Data\ Type:\ float_8 \\ Storage:\ \ rectangular \\ Order:\ Fortran\_order \end{bmatrix}$$

> *plots*[sparsematrixplot](M1, 'matrixview')



> st := time( ) : *approx1* := [*PCGstep*(*A*, *b*, *M1*, 200)]; *time1* := *time*( ) − *st*

approx1 := [[1, 0.4259], [2, 0.1681], [3, 0.0562], [4, 0.0050], [5, 0.0006], [6, 0.0001], [7, 0.0000], [8, $5.5900\,10^{-7}$], [9, $3.5998\,10^{-8}$], [10, $2.0921\,10^{-9}$]]

time1 := 0.5930

> "improvement in time rate" = $\frac{time0}{time1}$

"improvement in time rate" = 5.2614

```
> plots[logplot]([approx0, approx1], axis[2] = [gridlines], color = [blue, red], legend
    = ["Conjugate gradient", "Preconditioned CG"], thickness = 3, title
    = "Speed of convergence in logplot view")
```