```
begin
    using Pkg
    Pkg.activate(mktempdir())
    Pkg.add("PyPlot")
    Pkg.add("PlutoUI")
    using PlutoUI
    using PyPlot
    using LinearAlgebra
end
```

Tridiagonal systems

In the previous lecture (nb08) we introudced the discretization matrix for the 1D heat conduction problem. In general form it can be written as a tridiagonal matrix

$$A = \begin{pmatrix} b_1 & c_1 & & & \\ a_2 & b_2 & c_2 & & \\ & a_3 & b_3 & \ddots & \\ & & \ddots & \ddots & c_{n-1} \\ & & & a_N & b_N \end{pmatrix}$$

and stored in three arrays a,b,c.

Gaussian elimination using arrays a, b, c as matrix storage ?

- From what we have seen, this question arises in a quite natural way, and historically, the answer has been given several times and named differently
- TDMA (tridiagonal matrix algorithm)
- "Thomas algorithm" (Llewellyn H. Thomas, 1949 (?))
- "Progonka method" (from Russian "прогонка": "run through"; Gelfand, Lokutsievski, 1952, published 1960)

Прогонка: derivation

Write solution of $Au = f$ as

$$a_i u_{i-1} + b_i u_i + c_i u_{i+1} = f_i \quad (i = 1 \ldots N)$$

where we define $a_1 = 0$, $c_N = 0$.

- For $i = 1 \ldots N-1$, assume there are coefficients α_i, β_i such that

$$u_i = \alpha_{i+1} u_{i+1} + \beta_{i+1}$$

.

- Re-arranging, we can express u_{i-1} and u_i via u_{i+1}:

$$(a_i \alpha_i \alpha_{i+1} + b_i \alpha_{i+1} + c_i) u_{i+1} + (a_i \alpha_i \beta_{i+1} + a_i \beta_i + b_i \beta_{i+1} - f_i) = 0$$

- This is true for arbitrary u if $\begin{cases} a_i \alpha_i \alpha_{i+1} + b_i \alpha_{i+1} + c_i & = 0 \\ a_i \alpha_i \beta_{i+1} + a_i \beta_i + b_i \beta_{i+1} - f_i & = 0 \end{cases}$
- Re-arranging gives for $i = 1 \ldots N-1$:

$$\begin{cases} \alpha_{i+1} & = -\frac{c_i}{a_i \alpha_i + b_i} \\ \beta_{i+1} & = \frac{f_i - a_i \beta_i}{a_i \alpha_i + b_i} \end{cases}$$

Прогонка: realization

- Initialization of forward sweep:

$$\begin{cases} \alpha_2 & = -\frac{c_1}{b_1} \\ \beta_2 & = \frac{f_1}{b_1} \end{cases}$$

- Forward sweep: for $i = 2 \dots N-1$:

$$\begin{cases} \alpha_{i+1} &= -\frac{c_i}{a_i\alpha_i + b_i} \\ \beta_{i+1} &= \frac{f_i - a_i\beta_i}{a_i\alpha_i + b_i} \end{cases}$$

- Initialization of backward sweep: $u_N = \frac{f_N - a_N\beta_N}{a_N\alpha_N + b_N}$
- Backward sweep: for $i = N-1 \dots 1$:

$$u_i = \alpha_{i+1}u_{i+1} + \beta_{i+1}$$

Прогонка: properties

- N unknowns, one forward sweep, one backward sweep $\Rightarrow O(N)$ operations vs. $O(N^{2.75})$ for algorithm using full matrix
- No pivoting $\Rightarrow$ stability issues
- Stability for diagonally dominant matrices where $|b_i| > |a_i| + |c_i|$
- Stability for symmetric positive definite matrices
- In fact, this is a realization of Gaussian elimination on a particular data structure.

Tridiagonal matrices in Julia

In Julia, solution of a tridiagonal system is based on the LU factorization in the LAPACK routine `dgtsv` which also does pivoting.

```
N = 5
```

```
N=5
```

- LU Factorization in the case of a tridiagonal matrix with random diagonal entries

```
A = 5×5 Tridiagonal{Float64,Array{Float64,1}}:
 0.186793  0.985412   ⋅         ⋅         ⋅
 0.201945  0.336733  0.671338   ⋅         ⋅
  ⋅        0.518801  0.191529  0.416203   ⋅
  ⋅         ⋅        0.889957  0.147218  0.769763
  ⋅         ⋅         ⋅        0.726546  0.470719
```

```
A=Tridiagonal(rand(N-1),rand(N),rand(N-1))
```

```
LU{Float64,Tridiagonal{Float64,Array{Float64,1}}}
L factor:
5×5 Array{Float64,2}:
 1.0       0.0       0.0       0.0       0.0
 0.924968  1.0       0.0       0.0       0.0
 0.0       0.0       1.0       0.0       0.0
 0.0       0.0       0.0       1.0       0.0
 0.0       0.769797  0.752337  0.420408  1.0
U factor:
5×5 Array{Float64,2}:
 0.201945  0.336733   0.671338  0.0        0.0
 0.0       0.673945  -0.620967  0.0        0.0
 0.0       0.0        0.889957  0.147218   0.769763
 0.0       0.0        0.0       0.726546   0.470719
 0.0       0.0        0.0       0.0       -0.777014
```

```
lu(A)
```

```
Int64[2, 1, 4, 5, 3]
```

```
lu(A).p
```

```
Float64[1.5038, 0.729746, 0.671173, 1.18418, 0.296653]
```

```
A\ones(N)
```

Solving this system with a positive right hand side can yield negative solution components.

We see that the in order to maintain stability, pivoting is performed: the LU factorization is performed as $PA = LU$ where P is a permutation matrix. The underlying permutation can be obtained as `lu(A).p`)

- Define a diagonally dominant matrix with random entries with positive main diagonal and nonpositive off-diagonal elements:

```
A1 = 5×5 Tridiagonal{Float64,Array{Float64,1}}:
       2.04343   -0.907038    ⋅          ⋅          ⋅
      -0.265936   2.25515   -0.597263    ⋅          ⋅
        ⋅        -0.739934   2.24558   -0.790491    ⋅
        ⋅          ⋅        -0.701657   2.47384   -0.233202
        ⋅          ⋅          ⋅        -0.662966   2.22106
```

```
A1=Tridiagonal(-rand(N-1),rand(N).+2,-rand(N-1))
```

```
LU{Float64,Tridiagonal{Float64,Array{Float64,1}}}
L factor:
5×5 Array{Float64,2}:
  1.0       0.0        0.0        0.0       0.0
 -0.130142  1.0        0.0        0.0       0.0
  0.0      -0.346232   1.0        0.0       0.0
  0.0       0.0       -0.344154   1.0       0.0
  0.0       0.0        0.0       -0.301103  1.0
U factor:
5×5 Array{Float64,2}:
 2.04343  -0.907038   0.0        0.0        0.0
 0.0       2.1371    -0.597263   0.0        0.0
 0.0       0.0        2.03879   -0.790491   0.0
 0.0       0.0        0.0        2.20179   -0.233202
 0.0       0.0        0.0        0.0        2.15084
```

```
lu(A1)
```

```
Int64[1, 2, 3, 4, 5]
```

```
lu(A1).p
```

Here we see, that no permutation is needed to maintain stability, confirming the statement made. In this case, the underlying algorithm is equivalent to Progonka, and the resulting LU factorization can be stored in three diagonals.

```
Float64[0.844489, 0.800026, 0.97042, 0.742815, 0.671959]
```

```
A1\ones(N)
```

Here we get only nonnegative solution values, though the matrix off-diagonal elements are nonpositive. Later we will see that this is a theorem for this type of matrices.

```
5×5 Array{Float64,2}:
 0.519495    0.231457   0.0686099  0.0225583  0.00236853
 0.0678612   0.52144    0.154568   0.0508208  0.00533598
 0.024921    0.191491   0.55307    0.181845   0.019093
 0.00727301  0.0558852  0.16141    0.469004   0.0492435
 0.00217093  0.0166812  0.0481793  0.139993   0.464934
```

```
inv(A1)
```

The inverse is a nonnegative full matrix! This is a theorem as well.