Solving the system of linear equations - shaping the polynomial

Tomasz Chwiej

March 12, 2025

1 Introduction

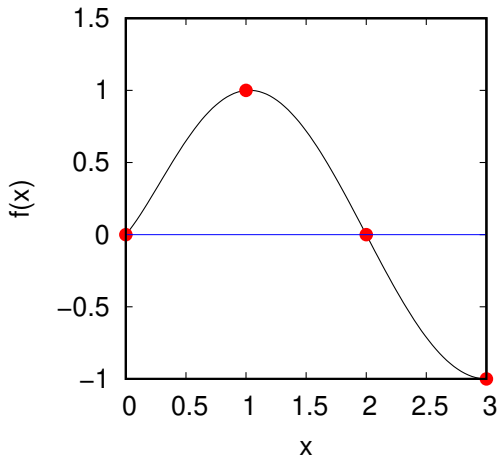

Figure 1: Polynomial of 5-th degree (black curve) which at the nodes (red dots) has defined properties: value and/or first derivative.

In many numerical applications we use basis of functions such that each of them must fullfill set of conditions cast on the function values and/or values of its derivative at some distinct points. These points are called collocation nodes. The simplest basis functions are polynomials defined as linear combination of monomials

$$f(x) = \sum_{i=0}^{N} a_i \, x^i \tag{1}$$

because we may easily manipulate their shapes and integrate or compute their derivatives. Let's define set of conditions, four function values and two first derivative values, for a polynomial displayed in

Fig.1

$$f(x_0) = 0 \tag{2}$$

$$\left.\frac{df(x)}{dx}\right|_{x=x_0} = 1 \tag{3}$$

$$f(x_1) = 1 \tag{4}$$

$$f(x_2) = 0 \tag{5}$$

$$f(x_3) = -1 \tag{6}$$

$$\left.\frac{df(x)}{dx}\right|_{x=x_3} = 0 \tag{7}$$

Because there are six conditions, the polynomial we are looking for must have exactly six linear coefficients so as to be uniquely determined. This implies 5-th degree polynomial:

$$f(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + a_4 x^4 + a_5 x^5 \tag{8}$$

Now, by implementing explicitly six conditions i.e. substituting definition of polynomial to each of equations (2)-(7) we get SLE

$$1 \cdot a_0 + \quad x_0 \cdot a_1 + \quad x_0^2 \cdot a_2 + \quad x_0^3 \cdot a_3 + \quad x_0^4 \cdot a_4 + \quad x_0^5 \cdot a_5 = \quad 0 \tag{9}$$

$$0 \cdot a_0 + \quad 1 \cdot a_1 + \quad 2x_0^1 \cdot a_2 + \quad 3x_0^2 \cdot a_3 + \quad 4x_0^3 \cdot a_4 + \quad 5x_0^4 \cdot a_5 = \quad 1 \tag{10}$$

$$1 \cdot a_0 + \quad x_1 \cdot a_1 + \quad x_1^2 \cdot a_2 + \quad x_1^3 \cdot a_3 + \quad x_1^4 \cdot a_4 + \quad x_1^5 \cdot a_5 = \quad 1 \tag{11}$$

$$1 \cdot a_0 + \quad x_2 \cdot a_1 + \quad x_2^2 \cdot a_2 + \quad x_2^3 \cdot a_3 + \quad x_2^4 \cdot a_4 + \quad x_2^5 \cdot a_5 = \quad 0 \tag{12}$$

$$1 \cdot a_0 + \quad x_3 \cdot a_1 + \quad x_3^2 \cdot a_2 + \quad x_3^3 \cdot a_3 + \quad x_3^4 \cdot a_4 + \quad x_3^5 \cdot a_5 = \quad -1 \tag{13}$$

$$0 \cdot a_0 + \quad 1 \cdot a_1 + \quad 2x_3^1 \cdot a_2 + \quad 3x_3^2 \cdot a_3 + \quad 4x_3^3 \cdot a_4 + \quad 5x_3^4 \cdot a_5 = \quad 0 \tag{14}$$

By rewriting SLE as single matrix equation we separate matrix elements $a_{i,j}$ from the solution vector elements $\vec{a}$ which we want to find

$$A\vec{a} = \vec{b} \tag{15}$$

$$\begin{bmatrix} 1 & x_0^1 & x_0^2 & x_0^3 & x_0^4 & x_0^5 \\ 0 & x_0 & 2x_0^1 & 3x_0^2 & 4x_0^3 & 5x_0^4 \\ 1 & x_1^1 & x_1^2 & x_1^3 & x_1^4 & x_1^5 \\ 1 & x_2^1 & x_2^2 & x_2^3 & x_2^4 & x_2^5 \\ 1 & x_3^1 & x_3^2 & x_3^3 & x_3^4 & x_3^5 \\ 0 & 1 & 2x_3^1 & 3x_3^2 & 4x_3^3 & 5x_3^4 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ -1 \\ 0 \end{bmatrix} \tag{16}$$

After finding the coefficients a_i we get the polynomial with embedded properties at collocation nodes. Remarks:

- we consider a standard SLE problem, the number of columns is the same as for the rows, hence SLE has single (unique) solution
- we might decrease the degree of polynomial but then the number of rows will be larger than the number of columns, SLE would be overdetermined and we wouldn't expect exact reconstruction of the initial conditions in polynomial but only approximately

2 Practical part

Tasks to do

1. Write the computer program solving the SLE with matrix A defined in Eqs. 15 and 16. To solve the problem numerically use the routine **lapacke_dgetrf()** to find the LU decomposition of matrix A and then the routine **lapacke_dgetrs()** to solve SLE using this LU.
2. In calculations use following positions of collocation nodes: $x_0 = 0$, $x_1 = 1$, $x_2 = 2$ and $x_3 = 3$.
3. Print the matrix elements on screen and check their correctness.
4. After computing LU decomposition calculate determinant of matrix A as the product of diagonal elements of matrix U (remeber: A is overwritten by routine **lapacke_dgetrf()** with the elements of L - below the diagonal, and, with the elements of U - an upper triangle + the diagonal)

$$det(A) = det(LU) = det(L)det(U) = 1 \cdot det(U) = \prod_{i=0}^{n-1} u_{ii} \tag{17}$$

5. Find coefficients a_i solving SLE and write them to file.
6. Use these coefficients to plot the polynomial (Eq.8).
7. At home prepare a report. Make a short analysis of stability of solving SLE for a case when x_2 node is shifted towards the x_1 node, namely, for $x_2 = 1.1, 1.01, 1.001, 1.0001$ calculate the determinant and solve SLE for each case. Based on definition of nonsingular matrix predict what would happen if two nodes coincide at one point, check this prediction with your program.

3 Hints

In this and some of the following projects we use **LAPACKE** package so we need to include appropriate header files

```
#include<lapacke.h>
```

and compile the source code with **liblapacke.a** and other needed libraries

```
g++ -I/path_to_header_files_directory -I/path_to_library   *.cpp -llapacke
-llapack -lblas -lm
```

In order to find the LU decomposition of matrix A we use **lapacke_dgetrf()** routine

```
lapack_int LAPACKE_dgetrf (int matrix_layout, lapack_int m, lapack_int n,
        double * a, lapack_int lda, lapack_int * ipiv);
```

One-dimensional array **a** on input stores the elements of matrix A (on output is overwritten by LU), **m** and **n** is the number of rows and columns of A, respectively, **lda** is the leading dimension of A (**a**) in direction indicated by **matrix_layout** while **ipiv** is an array storing indices of permuted rows/columns performed during factorization.

If the array **a** is declared as 1d vector object in **C++** then pass the pointer to its first element as an argument **&a[0]**.

In calculations use the following parametrization

- for **matrix_layout** assume const value $LAPACK\_ROW\_MAJOR$ - elements of A are then written in array **a** in a row-wise order (rows are written one by another)
- declare array **a** as one-dimensional array of length n^2, then **m=n** (square matrix A) and **lda=n**, for row-wise order the relation between elements of matrix A and array **a** is following (k-row index, i-column index)

$$a[k * n + i] \equiv A_{k,i}, \qquad k, i = 0, 1, 2 \ldots, n - 1 \tag{18}$$

- declare **ipiv** as one-dimensional array of length n

Calculated LU decomposition is next used for solving SLE by the routine **lapacke_dgetrs()**. It is important to not change any elements in arrays **a** and **ipiv** between calling first and second routine.

```
lapack_int LAPACKE_dgetrs (int matrix_layout, char trans, lapack_int n,
        lapack_int      nrhs, const double * a, lapack_int lda,
        const lapack_int * ipiv, double * b, lapack_int ldb);
}
```

Arguments: **matrix_layout**, **n**, **a**, **lda**, **ipiv** are the same as on output from **dgetrf**, do not change them. Meaning of other arguments

- **trans='N'** - do not transpose the matrix
- **nrhs=1** - only one right-hand-side vector $\vec{b}$
- declare **b** as array of length **n** it stores the right-hand-side vector $\vec{b}$ of SLE
- set **ldb=1** (row-major ordering)