NumCSE exercise sheet 2
LU decomposition, sparsity, least squares

soumil.gurjar@sam.math.ethz.ch
oliver.rietmann@sam.math.ethz.ch

October 10, 2018

Exercise 2.1. *LU decomposition and pivoting.*

We consider the family of *invertible* matrices

$$\mathbf{A}_\epsilon := \begin{pmatrix} \epsilon & 1 \\ 1 & 1 \end{pmatrix}, \qquad 0 \leq \epsilon < 1$$

and the permutation matrix

$$\mathbf{P} := \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

We will examine the equation

$$\mathbf{A}_\epsilon \cdot \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \tag{1}$$

with solution

$$x_1 = \frac{-1}{1-\epsilon}, \qquad x_2 = \frac{1}{1-\epsilon}. \tag{2}$$

(a) Show that $\mathbf{A}_0$ has no LU decomposition, while $\mathbf{P} \cdot \mathbf{A}_0$ does.

(b) If $0 < \epsilon < 1$, then $\mathbf{A}_\epsilon$ admits the LU decomposition $\mathbf{A}_\epsilon = \mathbf{L}_\epsilon \cdot \mathbf{U}_\epsilon$, where

$$\mathbf{L}_\epsilon := \begin{pmatrix} 1 & 0 \\ \frac{1}{\epsilon} & 1 \end{pmatrix}, \qquad \mathbf{U}_\epsilon := \begin{pmatrix} \epsilon & 1 \\ 0 & 1 - \frac{1}{\epsilon} \end{pmatrix}.$$

However for sufficiently[1] small $\epsilon > 0$, cancellation will occur: Instead of $\mathbf{U}_\epsilon$, we will end up with

$$\tilde{\mathbf{U}}_\epsilon := \begin{pmatrix} \epsilon & 1 \\ 0 & -\frac{1}{\epsilon} \end{pmatrix}.$$

Solve (1) using the erroneous LU decomposition $(\mathbf{L}_\epsilon, \tilde{\mathbf{U}}_\epsilon)$ and compare this solution with (2).

(c) Now we do the same, but with pivoting: For all $0 < \epsilon < 1$, we have $\mathbf{P} \cdot \mathbf{A}_\epsilon = \mathbf{L}_\epsilon^{\mathbf{P}} \cdot \mathbf{U}_\epsilon^{\mathbf{P}}$, where

$$\mathbf{L}_\epsilon^{\mathbf{P}} := \begin{pmatrix} 1 & 0 \\ \epsilon & 1 \end{pmatrix}, \qquad \mathbf{U}_\epsilon^{\mathbf{P}} := \begin{pmatrix} 1 & 1 \\ 0 & 1 - \epsilon \end{pmatrix}.$$

Again, for sufficiently small $\epsilon > 0$, we have to consider $\tilde{\mathbf{U}}_\epsilon^{\mathbf{P}}$, where

$$\tilde{\mathbf{U}}_\epsilon^{\mathbf{P}} := \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}.$$

Solve (1) using the erroneous LU decomposition $(\mathbf{L}_\epsilon^{\mathbf{P}}, \tilde{\mathbf{U}}_\epsilon^{\mathbf{P}})$ of $\mathbf{P} \cdot \mathbf{A}_\epsilon$. Compare this solution with (2) and the one from (b).

[1] This issue occurs for instance if ϵ is of type `float` and has value `1.0e-8f`.

Exercise 2.2. *LU decomposition and sparsity.*

Fix $n \in \mathbb{N}$ and let $\mathbf{T} \in \mathbb{R}^{n \times n}$ be the tridiagonal matrix

$$\mathbf{T} = \begin{pmatrix} 2 & -1 & 0 & \dots & 0 \\ -1 & 2 & -1 & \ddots & \vdots \\ 0 & -1 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & 2 & -1 \\ 0 & \dots & 0 & -1 & 2 \end{pmatrix}.$$

(a) How many real numbers are needed to store $\mathbf{T}$ for given n in CCS format?

(b) Write a function

```
std::vector<Eigen::Triplet<double>> MakeTripletList(int n)
```

which for given dimension `n` returns a vector of triplets representing $\mathbf{T}$.

(c) Write a function

```
double Runtime(const std::function<void(void)> &f)
```

that returns the runtime of a function `void f()` in seconds.

(d) Compare sparse vs. dense LU decomposition of $\mathbf{T}$ in terms of execution time as follows: Measure the runtime of `Eigen::SparseLU::compute` and `Eigen::FullPivLU::compute` for all $n \in \{64, 128, 256, 512\}$. What are the asymptotic complexities (large n)?

Exercise 2.3. *least squares.*

We consider the (scaled) *Runge*[2] function $r : [0,1] \to \mathbb{R}$ defined by

$$r(x) = \frac{1}{1+25x^2}$$

for all $x \in [0,1]$. Fix $m \in \mathbb{N}$ and suppose that r is given only by sampled data (x_i, y_i), where

$$x_i = \frac{2i}{m-1} - 1, \qquad y_i = r(x_i),$$

for all $i \in \{0, \dots, m-1\}$. For given $n \in \mathbb{N}$ we shall use this data to approximate r by a polynomial

$$p(x) = \sum_{i=0}^{n-1} a_i x^i,$$

where the coefficients $a_0, \dots, a_{n-1} \in \mathbb{R}$ are to be determined. To this end, we introduce the linear system

$$\underbrace{\begin{pmatrix} 1 & x_0^1 & \dots & x_0^{n-1} \\ 1 & x_1^1 & \dots & x_1^{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{m-1}^1 & \dots & x_{m-1}^{n-1} \end{pmatrix}}_{\mathbf{V}(x,n) :=} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{m-1} \end{pmatrix}, \tag{3}$$

which is equivalent to $p(x_i) = y_i$ for all $i \in \{0, \dots, m-1\}$. The matrix $\mathbf{V}(x,n)$ is invertible and is called *Vandermonde*[3] matrix.

(a) Implement a function

```
Eigen::MatrixXd Vandermonde(const Eigen::VectorXd &x, int n)
```

which takes the vector `x` of sample points $x_0, \dots, x_{m-1}$ and the number `n` of coefficients of p and returns the associated Vandermonde matrix $\mathbf{V}(x,n)$.

(b) We set $m = n$, so that (3) admits a unique solution. Use `C++/Eigen` to find this solution and print the coefficients of p for $n = 11$.

(c) Now we consider $m = 3n$ data points (yielding a tall $\mathbf{V}(x,n)$). In general, Equation (3) will no longer be solvable and we resort to a least squares solution. Write a short `C++/Eigen` code to obtain the least squares solution. Print the coefficients of p again for $n = 11$.

(d) Plot the exact Runge function r and the polynomial p obtained in Tasks (b) and (c). What do you observe?

[2]Named after the German mathematician and physicist Carl David Tolmé Runge.
[3]Named after the French mathematician Alexandre-Théophile Vandermonde.