TUTORIAL 6

1 An algorithm for computing the characteristic polynomial

Let $A \in \mathcal{M}_n(\mathbb{K})$, the goal of the following method is to compute the characteristic polynomial of A with a cost better than $O(n^4)$.

1. Let T be the transvection which acts on the left of a matrix A through $L_i \leftarrow L_i + \alpha L_j$, i.e., $T = I_n + \alpha E_{ij}$. Describe the action of T^{-1} on the right of A in terms of column operations.

2. Using question 1, show that one can find a matrix R such that

$$RAR^{-1} = \begin{bmatrix} a_{1,1} & a'_{1,2} & \cdots & a'_{1,n} \\ l_2 & a'_{2,2} & \ddots & a'_{2,n} \\ 0 & a'_{3,2} & \ddots & a'_{3,n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & a'_{n,2} & \cdots & a'_{n,n} \end{bmatrix}.$$

   (Hint: perform row operations by multiplying on the left by some transvection matrices T_i and see what happens on the columns when you multiply on the right by T_i^{-1}).

3. Give an algorithm to compute the matrices R_n and M such that

$$R_n A R_n^{-1} = M = \begin{bmatrix} m_{1,1} & m_{1,2} & m_{1,3} & \cdots & m_{1,n} \\ \ell_2 & m_{2,2} & m_{2,3} & \ddots & m_{2,n} \\ 0 & \ell_3 & m_{3,3} & \ddots & m_{3,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \ell_n & m_{n,n} \end{bmatrix}$$

   using $O(n^3)$ operations in $\mathbb{K}$.

   Remark: such an "almost triangular" shape matrix is called an *upper Hessenberg matrix*; we have shown how to reduce any matrix into (upper) Hessenberg form.

4. Deduce an algorithm to compute the characteristic polynomial of A, with a complexity bound $O(n^3)$.

5. Could it be possible to find R such that $R^{-1}AR = M$ is upper triangular by (arbitrarily many) elementary operations in $\mathbb{K}$? If yes, explain how. If not, explain why.

2 Toeplitz linear systems

Let $M \in \mathcal{M}_n(K)$ be a Toeplitz matrix, that is,

$$M = \begin{bmatrix} m_0 & m_{-1} & \cdots & m_{-n+2} & m_{-n+1} \\ m_1 & m_0 & \ddots & \ddots & m_{-n+2} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ m_{n-2} & \ddots & \ddots & \ddots & m_{-1} \\ m_{n-1} & m_{n-2} & \cdots & m_1 & m_0 \end{bmatrix}$$

for some $m_{-n+1}, \dots, m_0, \dots, m_{n-1} \in K$. The goal of this exercise is to solve the linear system $M\vec{x} = \vec{y}$ more efficiently than with general-purpose algorithms.

1. What is the size of the input of our problem? And the size of the output?

   For $k \in [n]$, we denote M_k the upper left sub-matrix of M of size $k \times k$. *We shall assume that M_k is non-singular (invertible) for all k.*

   We denote by $e_k^{(1)} \in K^k$ the vector $(1, 0, \cdots, 0)^T$ and by $e_k^{(k)} \in K^k$ the vector $(0, \cdots, 0, 1)^T$ of size k. For $k \in [n]$, we define $\vec{f_k} \in K^k$ by $M_k \vec{f_k} = e_k^{(1)}$, and $\vec{b}_k \in K^k$ by $M_k \vec{b}_k = e_k^{(k)}$.

2. Find $\vec{f_1}$ and $\vec{b}_1$.

3. Let $\vec{f}'_k = (\vec{f}^T_{k-1}, 0)^T$, and $\vec{b}'_k = (0, \vec{b}^T_{k-1})^T$. Compute $M_k \vec{f}'_k$ and $M_k \vec{b}'_k$. Deduce $\vec{f_k}$ and $\vec{b}_k$.

   For $k \in [n]$, let $\vec{y}^{(k)} = (y_1, \cdots, y_k)$ and define $\vec{x}^{(k)} \in K^k$ by $M_k \vec{x}^{(k)} = \vec{y}^{(k)}$. Note that we have $\vec{x} = \vec{x}^{(n)}$.

4. Give an algorithm, which on input $\vec{x}^{(k-1)}$, y_k and $\vec{b}_k$, computes $\vec{x}^{(k)}$.

5. Deduce an algorithm to solve a Toeplitz linear system. Give a complexity bound for your algorithm.

3 Hensel-type strategy for solving linear system

In this exercise, we study algorithms to solve $Mx = b$, $M \in \mathcal{M}_n(K[X]), b \in K[X]^n$. We shall assume that the degree of all coordinates of M, b is $\leqslant d$.

Cramer's formulas show that if x is a solution of $Mx = b$, $(\det M) \cdot x \in K[X]^n$, and the coefficients of $(\det M) \cdot x$ have degree $\leqslant nd$. We'll also assume that $\det M(u) \neq 0$ *for all $u \in K$.*

1. What is the complexity of computing $B := (M \bmod X)^{-1}$?

   Let $y_i \in K[X]^n$ be a solution of $My_i = b \bmod X^i$, and define $r_i = b - My_i$.

2. Prove that $r_i = \lambda_i X^i$ for some $\lambda_i \in K[X]^n$. If $z_i = B\lambda_i \bmod X$, prove that $y_{i+1} = y_i + X^i z_i$ and $r_{i+1} = r_i - X^i M z_i$.

3. What is the complexity of computing y_{nd+1} using this method? Assuming that $\det M$ is given as input or precomputed, deduce an algorithm for solving $Mx = b$.

4. If we need to compute $\det M$ beforehand, then this computation is going to dominate the complexity of linear system solving. Can we avoid computing the determinant? (Hint: use rational reconstruction.)