LINMA2171 : Numerical Analysis
Homework 3 : LSPIA algorithm for regression splines

Julie Weverbergh - 46271800

23 November 2021

1 Writing the theorem in mathematical terms

Theorem 2.4 from the article [1] can be rewritten as :

Theorem 1. *Let $d, m, n \in \mathbb{Z}$, $Q := [Q_0, Q_1, ..., Q_m]^T$ with $Q_i \in \mathbb{R}^{d+1} \quad \forall i = 0, ..., m$. Let A be the collocation matrix of the NTP blending basis defined in the equation (2) of [1], with $\{t_j\}_{j=0}^m$ the parameters of $\{Q_j\}_{j=0}^m$, and $\{P^k(t)\}_{k=0}^\infty$ be the sequence of curves generated by the LSPIA method. Then,*

$$\lim_{k\to\infty} P^k(t) = \sum_{i=0}^n B_i(t) P_i^\infty$$

where $[P_0^\infty, ..., P_n^\infty]^T =: P^\infty$ and

$$P^\infty = \arg \min_{P \in \mathbb{R}^{(n+1)\times(d+1)}} f(P) := ||Q - AP||_F^2 \tag{1}$$

Notice that we used the Frobenius norm since Q and P are matrices. In other words, we could rewrite

$$f(P) \quad = \quad \sum_{i=0}^d ||Q_{:,i} - AP_{:,i}||_2^2 \tag{2}$$

with the 2-norm, and we see that this is still a least squares problem.

2 Matrix form of the algorithm and steepest descent

2.1 Matrix form

This reasoning is mostly inspired from [1].

Let $P^k := [P_0^k, ..., P_n^k]^T$ with $P_i^k \quad \forall i = 0, ..., n$ be the i-th control point obtained at step k of LSPIA method, and $Q := [Q_0, ..., Q_m]^T$ with $Q_i \quad \forall i = 0, ..., m$ be the i-th data point to be fitted. By equations (5), (6) and (7) from [1], we can write $\forall i = 0, ..., n$,

$$\begin{aligned} P_i^{k+1} \quad &= \quad P_i^k + \mu \sum_{j=0}^m B_i(t_j)(Q_j - P^k(t_j)) \\ &= \quad P_i^k + \mu \sum_{j=0}^m B_i(t_j) \left[Q_j - \sum_{l=0}^n B_l(t_j) P_l^k \right] \end{aligned}$$

Writing this in a compact matrix form, we obtain

$$P^{k+1} \quad = \quad P^k + \mu A^T (Q - AP^k) \tag{3}$$

where μ is a constant satisfying (4) from [1], and A defined as in theorem 1.

2.2 Steepest descent

Let us apply the steepest descent method on problem (1), and take the same definitions from theorem 1.

Starting with some P^0 and letting $\alpha_k \in \mathbb{R}$, the gradient method on (1) gives $\forall k = 1...$,

$$P^{k+1} \quad = \quad P^k - \alpha_k \nabla f(P^k) \tag{4}$$

where ∇f is defined such that $(\nabla f)_{st} := \frac{\partial f}{\partial P_{st}}$, where P_{st} is the t-th component of P_s, $\quad \forall s = 0, ..., n; \forall t = 0, ..., d$.

Let us compute $\nabla f(P^k)$. By the definition of the Frobenius norm, $f(P)$ in (1) can be rewritten as

$$f(P) = \sum_{i=0}^{m} \sum_{j=0}^{d} (Q_{ij} - (AP)_{ij})^2$$

Then we can compute $\forall s = 0, ..., n; \forall t = 0, ..., d$

$$\begin{aligned} \frac{\partial f}{\partial P_{st}}(P^k) &= -2 \sum_{i=0}^{m} (Q_{it} - (AP^k)_{it}) A_{is} \\ &= -2 [A_{:s}]^T [Q_{:t} - (AP^k)_{:t}] \end{aligned}$$

and under a matrix form, this becomes

$$\nabla f(P^k) \quad = \quad -2A^T (Q - AP^k) \tag{5}$$

By (4), we obtain

$$P^{k+1} = P^k + 2\alpha A^T (Q - AP^k) \tag{6}$$

Denoting $\alpha = \frac{\mu}{2}$, we directly see that 3 is equivalent to 6, which proves that LSPIA method is a steepest gradient method.

3 Another version of the algorithm with μ computed by exact line search

Let μ^k be the step size chosen at iteration k, and let us take the same definitions as in theorem 1. We will propose two different versions of the algorithm : one with a unique $\mu^k \in \mathbb{R}$, and another with $\mu^k \in \mathbb{R}^{d+1}$.

3.1 $\mu^k \in \mathbb{R}$

To obtain μ^k by exact line search, we have to find the minimizer of f along the steepest descent direction from P^k, i.e. taking the recurrence relation (3), we have to solve $\forall k = 0...$

$$\mu^k = \arg \min_{\mu \in \mathbb{R}_{++}} F(\mu) := f(P^k + \mu A^T (Q - AP^k)) = ||Q - A[P^k + \mu A^T (Q - AP^k)]||_F^2 \tag{7}$$

Let us rewrite $F(\mu)$ from (7) :

$$\begin{aligned} F(\mu) &= ||(Q - AP^k) - \mu AA^T (Q - AP^k)]||_F^2 \\ &= \sum_{i=0}^{m} \sum_{j=0}^{d} [(Q - AP^k)_{ij} - \mu (AA^T (Q - AP^k))_{ij}]^2 \end{aligned}$$

Thus, taking the derivative, we obtain

$$\frac{\mathrm{d}F}{\mathrm{d}\mu}(\mu) \quad = \quad -2 \sum_{i=0}^{m} \sum_{j=0}^{d} [(Q - AP^k)_{ij} - \mu (AA^T (Q - AP^k))_{ij}] [AA^T (Q - AP^k)]_{ij}$$

and then $\frac{\mathrm{d}F}{\mathrm{d}\mu}(\mu^k) = 0$ gives

$$\mu^k \quad = \quad \frac{\sum_{i=0}^{m} \sum_{j=0}^{d} (Q - AP^k)_{ij} (AA^T (Q - AP^k))_{ij}}{\sum_{i=0}^{m} \sum_{j=0}^{d} (AA^T (Q - AP^k))_{ij}^2} \tag{8}$$

where the denominator can be rewritten in a compact form using the Frobenius norm.

Now let us try to find a better step size by choosing a different one for each component in $\mathbb{R}^{d+1}$, and this will even lead to a more compact form of μ^k.

3.2 $\mu^k \in \mathbb{R}^{d+1}$

Using (2) and since all $P_{:,i} \quad \forall i = 0, ..., d$ are independent, denoting[1] $P^\infty = [P_{:,0}, ..., P_{:,d}]$, the minimization problem in (1) can be rewritten as

$$P^\infty_{:,i} \quad = \quad \arg \min_{P \in \mathbb{R}^{n+1}} g(P) := ||Q_{:,i} - AP||_2^2$$

$\forall i = 0, ..., d.$

Now let us find $\mu_i^k \quad \forall i = 0, ..., d$ for each component separately. In the same way as in section 3.1, we have to solve $\forall k = 0...$ and $\forall i = 0, ..., d$

$$\mu_i^k = \arg \min_{\mu \in \mathbb{R}_{++}} G(\mu) := g(P^k_{:,i} + \mu A^T(Q_{:,i} - AP^k_{:,i})) = ||Q_{:,i} - A[P^k_{:,i} + \mu A^T(Q_{:,i} - AP^k_{:,i})]||_2^2 \tag{9}$$

Let us rewrite $G(\mu)$ from (9) :

$$\begin{aligned} G(\mu) &= ||(Q_{:,i} - AP^k_{:,i}) - \mu AA^T(Q_{:,i} - AP^k_{:,i})||_2^2 \\ &= ((Q_{:,i} - AP^k_{:,i}) - \mu AA^T(Q_{:,i} - AP^k_{:,i}))^T(Q_{:,i} - AP^k_{:,i}) - \mu AA^T(Q_{:,i} - AP^k_{:,i}) \end{aligned}$$

Thus, taking the derivative, we obtain

$$\frac{\mathrm{d}F}{\mathrm{d}\mu}(\mu) \quad = \quad -2((Q_{:,i} - AP^k_{:,i}) - \mu AA^T(Q_{:,i} - AP^k_{:,i}))^T(AA^T(Q_{:,i} - AP^k_{:,i}))$$

and then $\frac{\mathrm{d}G}{\mathrm{d}\mu}(\mu_i^k) = 0$ gives

$$\begin{aligned} \mu_i^k &= \frac{(Q_{:,i} - AP^k_{:,i})^T AA^T(Q_{:,i} - AP^k_{:,i})}{||AA^T(Q_{:,i} - AP^k_{:,i})||_2^2} \\ &= \frac{||A^T(Q_{:,i} - AP^k_{:,i})||_2^2}{||AA^T(Q_{:,i} - AP^k_{:,i})||_2^2} \\ &= \frac{||\nabla f_{:,i}(P^k)||_2^2}{||A\nabla f_{:,i}(P^k)||_2^2} \end{aligned} \tag{10}$$

where the last equality comes from (5). Finally, we will denote $\mu^k = [\mu_0^k, ..., \mu_d^k]^T$ from now on.

3.3 Comparison and conclusion

The two approaches use the exact line search, but we may guess that the second one will be better since we have more degrees of freedom to choose μ^k since it has a higher dimension.

In conclusion, the algorithm can be rewritten as, at step k :

1. compute μ^k using (8) or (10) ;
2. compute P_{k+1} using, by (3),

$$P^{k+1} \quad = \quad P^k + \mu^k A^T(Q - AP^k)$$

if $\mu^k \in \mathbb{R}$, or

$$P^{k+1}_{:,i} \quad = \quad P^k_{:,i} + \mu_i^k A^T(Q_{:,i} - AP^k_{:,i}) \quad \forall i = 0, ..., d$$

if $\mu^k \in \mathbb{R}^{d+1}$.

1. The notation $C_{:,i}$ means that we take the i-th column of the matrix C.

4 Comparison between LSPIA and solving the normal equations

4.1 Results and error analysis

Let us now apply the LSPIA method on the letter J, with $m = 200$, $n = 50$ and μ computed as described in section 3.2 from [1]. The steps 0, 1, 3, 5 and 7 are shown at figures 1, 2, 3, 4 and 5 respectively (Q in blue, P in orange and the B-spline curve in red). And the Whitney-Schoenberg condition has been checked in the implementation of the algorithm, by checking that each column of A has at least 1 non-zero element.

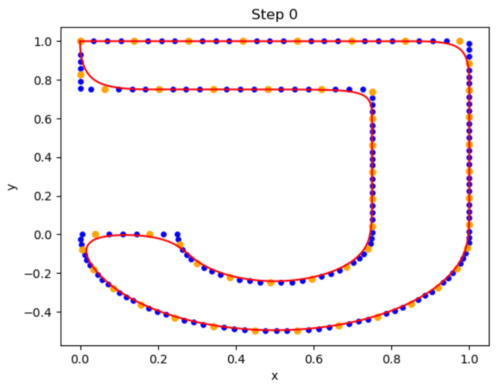

FIGURE 1 – Step 0 - Initial shape

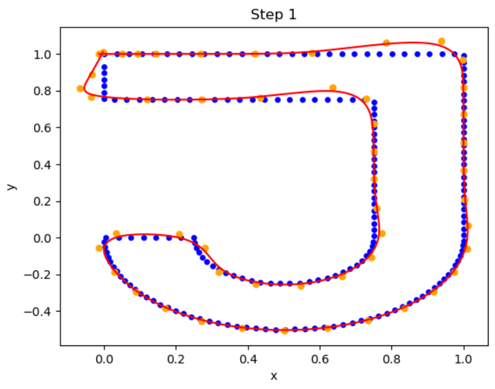

FIGURE 2 – Step 1

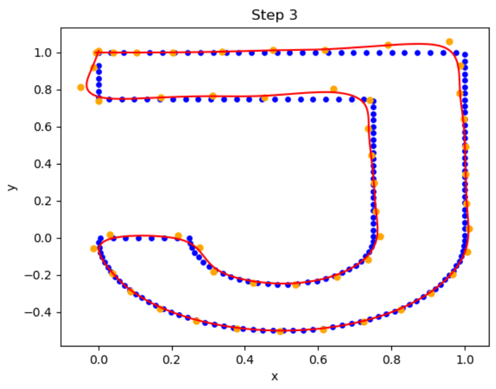

FIGURE 3 – Step 3

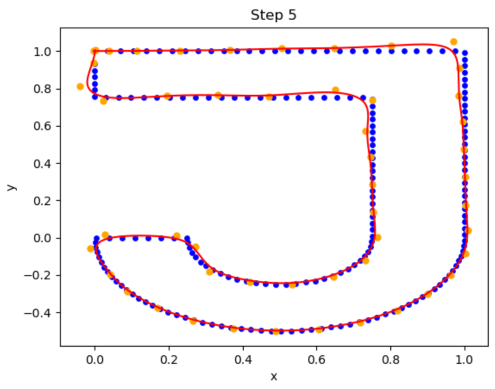

FIGURE 4 – Step 5

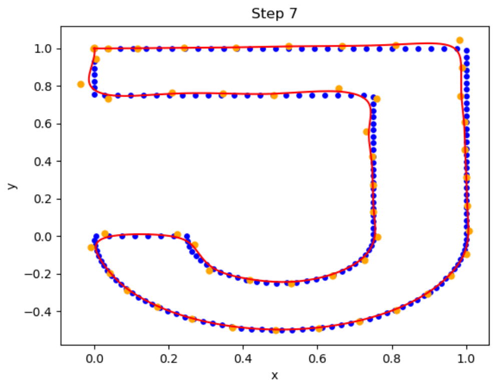

FIGURE 5 – Step 7

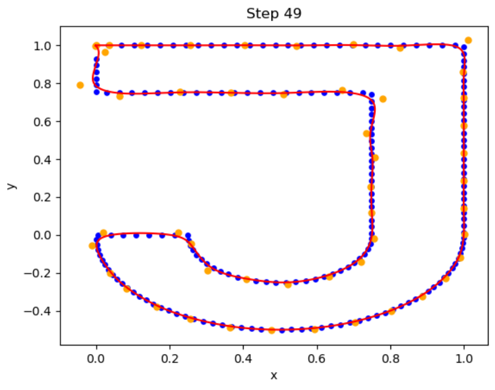

FIGURE 6 – Step 49 - Fitting curve

We can see that, as the number of steps increases, the B-spline curves tend to become very close to the J-shape. Let us define the same error as in section 4.2 from [1] :

$$E_k := \sum_{j=0}^{m} ||Q_j - \sum_{i=0}^{n} B_i(t_j) P_i^k||^2$$

Using the stop criterion $|E_{k+1} - E_k| < 10^{-7}$, we obtain the fitting curve at figure 6 after 49 iterations. We see that it is seemingly nearly the same as the least squares solution (i.e. the curve corresponding to the control points solutions of normal equations $A^T AP = A^T Q$) shown at figure 7. It makes sense since the curves produced by LSPIA method converge to the least squares solution, by theorem 2.4 from [1].

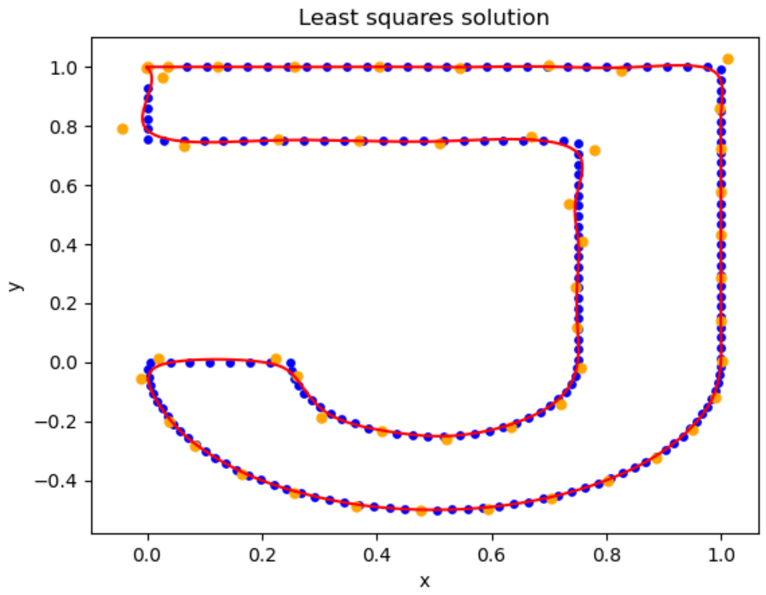

FIGURE 7 – Least squares solution

Moreover, as we see on figure 8, the error tends to decrease at each iteration, and converges to the error of the least squares solution, equal to 0.004052. Furthermore, this also means that we will never do better than this precision of 0.004052 with LSPIA method.

In a log-log scale, we can say that the error decreases approximately in $\mathcal{O}(\frac{1}{k^2})$ as we see on figure 9 (even if it seems to behave as $\mathcal{O}(\frac{1}{k})$ during the first iterations). To go further, we could investigate some results about convergence of gradient method and see if we can apply these results to this example.

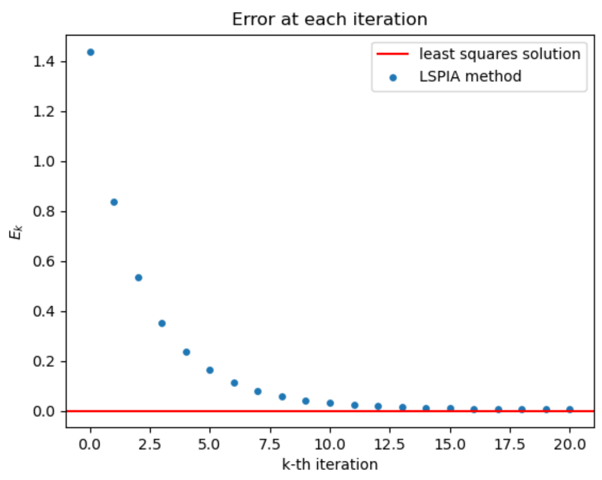

FIGURE 8 – Error at each iteration

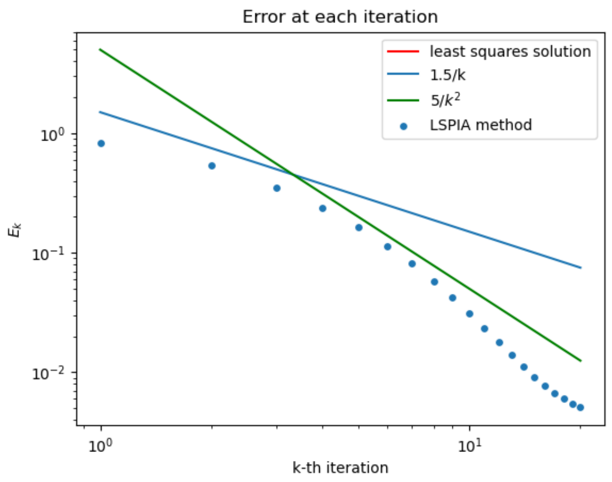

FIGURE 9 – Error at each iteration - Log-log scale

4.2 Time and space complexities

4.2.1 Time complexity

To solve the normal equations, we need to make the gaussian elimination and the computation of A^TA and A^TQ. A^TA and A^TQ require $\mathcal{O}(mn^2)$ and $\mathcal{O}(mnd)$ operations respectively. And since A^TA has a dimension $n \times n$, the time complexity of the gaussian elimination is $\mathcal{O}(n^3)$. Thus the time complexity to find the least squares solution is $\mathcal{O}(mn^2)$ since $m > n$.

For the LSPIA method, at each iteration, we compute the new iterate (3) and μ. The computation of (3) requires matrix multiplication, which gives the time complexity $\mathcal{O}(mnd)$, while μ needs $\mathcal{O}(mn)$ operations (assuming we compute it with the approximation described in section 3.2 from [1]). For k iterations, without counting the initializations and other calculations which are negligeable compared to the total complexity, we obtain $\mathcal{O}(kmnd)$ operations for LSPIA method.

In conclusion, assuming $d, k \ll n$ (if $k = 20$, we already reach a good precision, as we see on figure 8), LSPIA method becomes much faster than solving normal equations.

4.2.2 Space complexity

Since A has a lot of zeros, this matrix can be implemented as a sparse matrix. But if m is extremely large, i.e. the data set contains a lot of points, then even the sparse matrix can not be stored anymore, because of lack of memory. Thus solving normal equations becomes impossible.

However, if we implement the LSPIA method as described in section 2.1 from [1] (and not using the compact form 3), we do not need A anymore. In this case, we only need to store the data points, the control points and some intermediate matrices which are of size $m \times d$ or $n \times d$. Since $d \ll m, n$, there are no more issues.

In conclusion, assuming $d \ll m, n$, LSPIA method allows to deal with problems of very large size, which are impossible to solve directly with normal equations.

Références

[1] Chongyang Deng and Hongwei Lin. Progressive and iterative approximation for least squares b-spline curve and surface fitting. *Elsevier*, 2012.