The LegendreSobolev Package and Its Applications in Handwriting Recognition

Parisa Alvandi, Stephen M. Watt

David R. Cheriton School of Computer Science
University of Waterloo, Waterloo, Canada
palvandi@uwaterloo.ca, smwatt@uwaterloo.ca

Abstract. The present work is motivated by the problem of mathematical handwriting recognition where symbols are represented as parametric plane curves in a Legendre-Sobolev basis. An early work showed that approximating the coordinate functions as truncated series in a Legendre-Sobolev basis yields fast and effective recognition rates. Furthermore, this representation allows one to study the geometrical features of handwritten characters as a whole. These geometrical features are equivalent to baselines, bounding boxes, loops, and cusps appearing in handwritten characters. The study of these features becomes a crucial task when dealing with two-dimensional math formulas and the large set of math characters with different variations in style and size.

In an early paper, we proposed methods for computing the derivatives, roots, and gcds of polynomials in Legendre-Sobolev bases to find such features without needing to convert the approximations to the monomial basis. Furthermore, in this paper, we propose a new formulation for the conversion matrix for constructing Legendre-Sobolev representation of the coordinate functions from their moment integrals.

Our findings in employing parametrized Legendre-Sobolev approximations for representing handwritten characters and studying the geometrical features of such representation has led us to develop two Maple packages called `LegendreSobolev` and `HandwritingRecognitionTesting`. The methods in these packages rely on Maple's linear algebra routines.

1 Introduction

Orthogonal polynomials have many applications in different recognition problems such as face [15], speech [5], speech emotion [13], and gesture [14] recognition. We are particularly interested in using orthogonal polynomials to represent handwritten characters for the purpose of mathematical handwriting recognition. In fact, modelling handwritten characters as parametrized curves $X(\lambda)$ and $Y(\lambda)$ on an orthogonal basis accurately captures the shape of handwritten mathematical characters using few parameters. This new representation of handwritten characters turns the recognition into a writer and device independent problem as one does not need to deal with the factors such as different device resolutions and the number of points in the sampling process of a handwritten character.

The authors of [6] have found that even without further similarity-processing, the polynomial coefficients from the writing samples form clusters which often contain the same characters written by different test users. This work uses the Chebyshev basis to represent handwritten characters as parametrized curves.

The work [7] is inspired by a similar idea as [6] to represent handwritten data, but uses a different functional basis. The basic idea of [7] is to compute moments of the coordinate curves in real time as the character is being written and then to construct the coefficients of the coordinate curves in the Legendre basis from moments at the time pen is lifted. As a matter of fact, the Legendre representation is just as suitable in practice for representation and analysis of ink traces as the Chebyshev representation, but has the benefit that it can be computed in a small, fixed number of arithmetic operations at the end when a stroke is written, completely. This approach works for any inner product with a linear weight function.

Representing handwritten characters with parametrized coordinate curves in the Chebyshev and Legendre series yields low RMS error rates. However, low RMS error rates do not guarantee the shape similarity of two characters, see Fig. 1. In fact, the problem with the blue and red curves on the right side of Figure 1 is that the corners of these two curves are not in the right places, despite the fact that they demonstrate a lower RMS error rate compared to the curves on the left. One solution to this obstacle might be to work in a jet space to force coordinate and derivative functions of the letters from the same class to have a similar form.

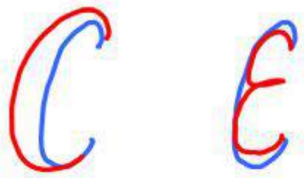

Fig. 1: Despite the fact that the two curves on the right have lower RMS error compared to the ones on the left, they do not have a similar shape.

The work [7, 9] use a special case of Legendre-Sobolev polynomials to represent handwritten characters based on the computation of moment integrals. These special polynomials, which are also called Althammer polynomials, enable us to work in a first jet space. The authors have reported experimental results that demonstrate that representing coordinate curves in a Legendre-Sobolev basis has higher detection rates compared to when these curves are represented in the Legendre basis.

In [2], we have proposed an online method for computing Legendre-Sobolev representations of handwritten characters from their moments by a matrix mul-

tiplication. Furthermore, we have presented methods in [3] for computing the derivatives, roots, and gcd of polynomials in Legendre-Sobolev bases by relying on linear algebra arithmetic operations. The goal of the latter work is to study the geometry and features of handwritten curves by relying on the Legendre-Sobolev coefficients of the approximated parametrized curves corresponding to handwritten characters. In [12], the authors proposed an algorithm to compute some of the important features of the Legendre-Sobolev approximations of handwritten characters by relying on the Newton method. Thus, conversion from a Legendre-Sobolev basis to the monomial basis is required. But the work [3] avoids this conversion because such conversion is known to be ill-conditioned.

In the present work, we give a new formulation for the matrix C in Proposition 1, which is inspired by the work [2], to compute the Legendre-Sobolev coefficients of the truncated parametrized curves of handwritten characters from their moments. The results in [2, 3] and Proposition 1 have led to the development of a MAPLE package called `LegendreSobolev`[1]. This package has all the necessary tools for representing handwritten characters as parametrized curves in a Legendre-Sobolev basis for the purpose of handwriting recognition.

After giving the preliminaries and our new result on the construction of the Legendre-Sobolev coefficients from moments, we explain the structure of the package `LegendreSobolev` with demonstrative examples in Section 4. Finally, we illustrate how to compute Legendre-Sobolev representations of handwritten curves from their corresponding digital inks, by relying on `LegendreSobolev` package, in Section 5.

2 Preliminaries

Digital ink is generated by sampling points from handwritten characters and is a collection of points (x, y, t) with position (x, y) and timestamp t. We use these points to compute moment integrals and from them, we approximate the coefficients of the coordinate curves $X(\lambda)$ and $Y(\lambda)$ on an orthogonal basis, where λ is either time or length of handwritten curves. In this paper, we assume that sample values of $X(\lambda)$ and $Y(\lambda)$ are received as a real time signal and λ corresponds to the length of handwritten curves. We assume the sample points are equally spaced with $\Delta t = 1$.

The works [6, 7, 2] showed that the coordinate curves $X(\lambda)$ and $Y(\lambda)$ for handwritten characters can be modelled by truncated Chebyshev, Legendre, and Legendre-Sobolev series, respectively. The coefficients of such series can be used for classification and recognition of characters. In this paper, we are interested in Legendre-Sobolev approximations for representing handwritten characters.

For two functions $f, g : [-1, 1] \to \mathbb{R}$, consider the following inner product which is given as

$$\langle f(\lambda), g(\lambda) \rangle = \int_{-1}^{1} f(\lambda)\, g(\lambda) d\lambda + \mu \int_{-1}^{1} f'(\lambda) g'(\lambda) d\lambda, \tag{1}$$

[1] This package is publicly available at `www.maplesoft.com/applications/view.aspx?SID=154553`

where $\mu \in \mathbb{R}_{\geq 0}$. This inner product is a special case of the Legendre-Sobolev inner product. In fact, the Legendre-Sobolev inner product may involve terms corresponding to higher order derivatives, but for the purpose of this paper we restrict ourselves to the first order derivative. Systems of orthogonal polynomials corresponding to the above inner product can be computed by applying the Gram-Schmidt orthogonalization process to the monomial basis. We denote the Legendre-Sobolev polynomials corresponding to the inner product given by Eq. (1) (which are also called Althammer polynomials, see [1]) of degree n by $S_n^\mu(\lambda)$. When $\mu = 0$ in Eq 1, we denote the polynomial $S_n^0(\lambda)$ by $P_n(\lambda)$, as well. In fact, the polynomial $P_n(\lambda)$ is the Legendre polynomial of degree n.

A function $f : [-1, 1] \to \mathbb{R}$, when the integrals involved in the inner product are well-defined for $f(\lambda)$, can be represented by an infinite linear combination of orthogonal polynomials $\{S_0^\mu(\lambda), S_1^\mu(\lambda), \ldots\}$ as

$$f(\lambda) = \sum_{i=0}^{\infty} \alpha_i S_i^\mu(\lambda).$$

The coefficients of the series in the new basis can be computed by the formula

$$\alpha_i = \frac{\langle f(\lambda), S_i^\mu(\lambda)\rangle}{\langle S_i^\mu(\lambda), S_i^\mu(\lambda)\rangle} \ , \ i = 0, 1, \ldots,$$

where $\langle ., . \rangle$ stands for the Legendre-Sobolev inner product given by Eq. (1). For representing handwritten characters, we use truncated linear combinations of Legendre-Sobolev polynomials to represent the function $f(\lambda)$. In fact, the closest polynomial of degree d to function $f(\lambda)$ with respect to Euclidean norm induced by the given inner product is the following series

$$f(\lambda) \simeq \sum_{i=0}^{d} \alpha_i S_i^\mu(\lambda).$$

Such approximation allows to think of functions as points $(\alpha_0, \ldots, \alpha_d)$ in $(d+1)$-dimensional vector space. That means that one can establish a method for measuring how close two functions are to each other in such $(d+1)$-dimensional vector space. In other words, for two functions $f, g : [-1, 1] \to \mathbb{R}$, if we approximate $f(\lambda)$ and $g(\lambda)$ as following

$$f(\lambda) \simeq \sum_{i=0}^{d} \alpha_i S_i^\mu(\lambda) \ , \ g(\lambda) \simeq \sum_{i=0}^{d} \beta_i S_i^\mu(\lambda),$$

then one can measure how close $f(\lambda)$ and $g(\lambda)$ are by computing the quantity

$$\| f(\lambda) - g(\lambda) \| \simeq \sqrt{\sum_{i=0}^{d} (\alpha_i - \beta_i)^2}.$$

This method of measuring the distance of two functions is the basic and important rule in the handwriting recognition method used in [8].

The moments of a function $f(\lambda)$ defined on the interval $[a, b]$ are the integrals:

$$\int_a^b \lambda^k \, f(\lambda) d\lambda.$$

A key aspect of the approach used in [6] for the purpose of interpolating the coordinate curves $X(\lambda)$ and $Y(\lambda)$ corresponding to handwritten strokes is to recover these curves from their moments. This is the Hausdorff moment problem [10, 11], known to be ill-conditioned. For the purpose of this paper, the moments of a function f are defined over an unbounded half-line since the curve may be traced over an arbitrary length:

$$m_k(f(\lambda), \ell) = \int_0^\ell \lambda^k \, f(\lambda) d\lambda.$$

In our application, we assume that discrete sample values of $f(\lambda)$ are received as a real-time signal. We use these values to compute approximate values for the moment integrals. After a curve is traced out, we will have computed its moments over some length L, with L known only at the time the pen is lifted. The problem is now to scale L to a standard interval and compute the truncated Legendre-Sobolev series coefficients for the scaled function from the moments of the unscaled function, $m_k(f(\lambda), L)$.

Having represented handwritten characters as Legendre-Sobolev coefficients of parametrized coordinate curves, we can study the geometrical features of handwritten curves by means of linear algebra calculations, such as matrix multiplication, and solving Diophantine equations. The work [3] gives methods for computing the derivatives, roots, and gcd of polynomials in Legendre-Sobolev bases based on their coefficient matrix.

3 Construction of Handwritten curves from Moments

The goal of this section is to present our new result on computing the representations of handwritten characters as approximated parametrized curves in Legendre-Sobolev bases from their corresponding digital inks, see Proposition 1.

To find such representation, one needs to compute moment integrals as a curve is being traced out, first, and then use Proposition 1 to compute the Legendre-Sobolev coefficients of the parametrized coordinate curves in a Legendre-Sobolev basis. In Section V in [2], we have explained how to compute moment integrals from the input digital ink.

Proposition 1 *Suppose that $m_i(f(\lambda), L)$ is defined as $\int_0^L f(\lambda)\lambda^i d\lambda$ where L is the length of a given curve and $f(\lambda)$ is either $X(\lambda)$ or $Y(\lambda)$, for $i = 0, \ldots, d$. Let also $\hat{f}(\lambda) = \sum_{i=0}^d \alpha_i S_i^\mu(\lambda)$ be the corresponding scaled function of $f(\lambda)$ in the interval $[-1, 1]$. Then for $i = 0, \ldots, d$, one may compute α_i as*

$$\alpha_i = \sum_{j=0}^{d} \frac{1}{L^{j+1}} \, C_{ij} \; m_j(f(\lambda), L),$$

where

$$(-1)^{i+j}C_{ij} = \frac{1}{2i!(i+1)!}\left(B_{ij}\left(\lfloor\tfrac{d-j}{2}\rfloor\right) - B_{ij}\left(\max(0,\lceil\tfrac{i-j}{2}-1\rceil)\right)\right)\left(\frac{1}{a_j(\mu)} - \frac{1}{a_{j+2}(\mu)}\right)$$
$$+ (2j+1)\binom{j}{i}\binom{i+j}{j}\frac{1}{a_j(\mu)},$$
$$B_{ij}(k) = \frac{(i+j+2k+2)!}{(j-i+2k)!}, \quad \textit{taking } \frac{1}{(-n)!} = 0, \textit{ for } k,n \in \mathbb{Z}^+,$$

$$a_0(\mu) = 1,\ a_i(\mu) = \sum_{k=0}^{\lfloor\frac{i-1}{2}\rfloor}\left(\frac{\mu}{4}\right)^k \frac{(i+2k-1)!}{(2k)!(i-2k-1)!},\ \textit{for } i \geq 1. \tag{2}$$

Proof We have simplified the following summation as

$$\sum_{\ell=k_1}^{k_2}(2\,j+1+4\,\ell)\binom{j+2\,\ell}{i}\binom{i+j+2\,\ell}{j+2\,\ell} = \frac{1}{2i!(i+1)!}\left(B_{ij}(k_2) - B_{ij}(k_1-1)\right),$$

for $k_1, k_2 \in \mathbb{Z}_{\geq 0}$, while taking $\frac{1}{(-n)!} = 0$, for $n \in \mathbb{Z}^+$. Then, the proof of this proposition is followed by substituting the above simplified form in the counterpart representation of matrix C in [2]. □

Note that the coefficients C_{ij} are independent of the problem and may be computed as constants, in advance.

We have developed the MAPLE's package `HandwritingRecognitionTesting`[2] which implements the idea of Proposition 1 for computing Legendre-Sobolev approximations of handwritten curves. Section 5 explains how to use this package to compute such representations.

4 LegendreSobolev package

We have developed a MAPLE package for applying different mathematical operations in Legendre-Sobolev bases called `LegendreSobolev` (see Fig. 2). The work [2, 3] and Proposition 1 support the theory behind the commands in this package. The operations in this package rely on linear algebra arithmetic operations such as matrix multiplication, and solving Diophantine equations.

In this section, we explain how one can use the commands in `LegendreSobolev` package to compute Legendre-Sobolev polynomials of a given degree and parameter μ, change the representation of polynomials with respect to different bases, and find roots and gcds of polynomials in Legendre-Sobolev bases.

[2] This package is publicly available at `www.maplesoft.com/applications/view.aspx?SID=154553`

```
> read "LS.mpl" :
  with(LegendreSobolev);
      [ ComradeMatrix, DerivativeInLS, DerivativeMatrixInLS, GcdInLS,
        LSToLegendreMatrix, LSToMonomialMatrix, LegendreToLSMatrix,
        MomentsToLSMatrix, MonomialToLSMatrix, P, S, α ]
```

Fig. 2: The functions of `LegendreSobolev` package.

4.1 Legendre-Sobolev polynomials

As Fig. 3 demonstrates, one can use the command `S` in `LegendreSobolev` package to compute a Legendre-Sobolev polynomial $S_n^{\mu}(\lambda)$. Furthermore, we can use both $S_n^0(\lambda)$ and $P_n(\lambda)$ to compute the Legendre polynomial of degree n, see Fig. 4.

```
> n := 20 :
  μ := 0.125 :
  S[n, μ](x);
      5.87545823986209778959007681 10^13 − 1.11772423971922692903741 6 10^16 x^2
      +3.52438029997315411930387 4 10^17 x^4 − 4.32684548233029055116120 81 10^18 x^6
      +2.70617540023482520158424 51 10^19 x^8 − 9.74805902439812478072269 21 10^19 x^10
      +2.14272436481779852545311 52 10^20 x^12 − 2.92106975571110920158241 62 10^20 x^14
      +2.41083585404532176254424308 10^20 x^16 − 1.10338344683885758917477137 10^20 x^18
      +2.14936627054372451057874677 10^19 x^20
```

Fig. 3: Legendre-Sobolev polynomial computation with `LegendreSobolev` package.

```
> S[5, 0](x);
      15/8 x − 35/4 x^3 + 63/8 x^5
```

```
> P[5](x);
      15/8 x − 35/4 x^3 + 63/8 x^5
```

Fig. 4: The Legendre polynomial computation with `LegendreSobolev` package with two different functions.

Legendre-Sobolev polynomials of a given degree n can also be computed as a function in μ (see Figure 5). The function α in `LegendreSobolev` package gives a polynomial with respect to degree n and parameter μ. In fact, the command α is used to compute Legendre-Sobolev polynomials with respect to the Legendre polynomials and implements $a_n(\mu)$ given by Eq. (2). The formula for computing Legendre-Sobolev polynomials from $a_n(\mu)$ and the Legendre polynomials is given by the relation: $S_n^{\mu}(\lambda) = S_{n-2}^{\mu}(\lambda) + a_n(\mu)\ (P_n(\lambda) - P_{n-2}(\lambda))$.

```
> S[5, μ](x);
      x + (1 + 3 μ)(−5/2 x + 5/2 x^3) + (105μ^2 + 45μ + 1)(27/8 x − 45/4 x^3 + 63/8 x^5)
> alpha[5, μ];
      105μ^2 + 45μ + 1
```

Fig. 5: Legendre-Sobolev polynomial computation as a function in μ.

4.2 Changing a polynomial representation w.r.t different bases: Legendre-Sobolev, Legendre, and monomial

When a polynomial $f(\lambda)$ is given in the monomial basis, one can compute the coefficients of the polynomial in any Legendre-Sobolev basis, with parameter μ, by using `MonomialToLSMatrix` command of `LegendreSobolev` package (see Fig. 6). In fact, one needs to multiply the coefficient matrix of the given polynomial in the monomial basis and the matrix in the output of the command `MonomialToLSMatrix`(degree(f), μ).

```
> f := (x − 1)^2 (x + 2)^3 :
  fcoeffs := Matrix([seq(coeff(f, x, i), i = 0..degree(f))]) :
  μ := 0.125 :
  fcoeffsInLS := Multiply(fcoeffs, MonomialToLSMatrix(degree(f), μ));
      fcoeffsInLS := [ 5.46666666666667  − 2.70649350661521  − 3.78467908902692
                       0.691130587508162 0.318012422360248 0.0153629189534213 ]
```

Fig. 6: Changing the representation of f from the monomial to Legendre-Sobolev basis with $\mu = 0.125$.

When a polynomial $f(\lambda)$ is given in the monomial basis, it is possible to convert this polynomial to the Legendre basis, as well, by a matrix multiplication. In fact, to compute the Legendre representation of $f(\lambda)$, one needs to multiply the coefficient matrix of $f(\lambda)$ in the monomial basis and the matrix in the output of the command `MonomialToLSMatrix`(degree(f), μ), where $\mu = 0$ (see Fig. 7).

```
> f := (x − 1)^2 (x + 2)^3 :
  fcoeffs := Matrix([seq(coeff(f, x, i), i = 0..degree(f))]) :
  fcoeffsInL := Multiply(fcoeffs, MonomialToLSMatrix(degree(f), 0));
      fcoeffsInL := [ 82/15  − 104/35  − 92/21  38/45  32/35  8/63 ]
```

Fig. 7: Changing the representation of f from the monomial to Legendre basis.

Furthermore, when a polynomial $f(\lambda)$ is given in the Legendre basis, one can change its representation to a Legendre-Sobolev one by multiplying the coefficient matrix of $f(\lambda)$ in the Legendre basis and the matrix from the output of the command `LegendreToLSMatrix`(degree(f), μ), where μ is given (see Fig. 8a).

```
> fcoeffsInL := Matrix([[1, 1, 1, 1, 1, 1]]) :
  μ := 1/5 :
  fcoeffsInLS := Multiply(fcoeffsInL,
   LegendreToLSMatrix(5, μ))
    fcoeffsInLS := [1  7/4  7/4  335/284  1/4  5/71 ]
```

(a)

```
> fcoeffsInLS := Matrix([[1, 1, 1, 1, 1, 1]]) :
  μ := 1/5 :
  fcoeffsInL := Multiply(fcoeffsInLS,
   LSToLegendreMatrix(5, μ))
    fcoeffsInLS := [1  −1/5  −2  −11  4  71/5 ]
```

(b)

Fig. 8: Changing the representation of f from (a) the Legendre to Legendre-Sobolev and (b) Legendre-Sobolev to Legendre basis with $\mu = \frac{1}{5}$.

Finally, for a polynomial $f(\lambda)$ in a Legendre-Sobolev basis, we can compute the Legendre representation of this polynomial by multiplying the corresponding coefficients in the Legendre-Sobolev basis and the matrix in the output of the command `LSToLegendreMatrix`(degree(f), μ), where μ is the same parameter as the one in the Legendre-Sobolev representation of $f(\lambda)$, see Fig. 8b.

4.3 Computing the derivatives of polynomials in Legendre-Sobolev bases

When a polynomial is given in a Legendre-Sobolev basis, one can compute the derivative of such polynomial by a matrix multiplication (see [3]). Figure 11 shows how to compute the derivative of a polynomial by using the command `DerivativeInLS` of `LegendreSobolev` package.

```
> f := (x1)2(x + 2)3;
  fcoeffs := Matrix([seq(coeff(f, x, i), i = 0..degree(f))]) :
  μ := .125 :
  fcoeffsInLS := fcoeffsInL · LegendreToLSMatrix(degree(f), );
  der := DerivativeInLS(, convert(fcoeffsInLS, list))
  fcoeffsInLS := [ 5.46666666666667  − 2.70649350661521
  −3.78467908902692  .691130587508162  .318012422360248
  0.0153629189534213 ]
  der := [ −2.00000000015363, 8.65454545474783, 5.60248447177297
  4.65454545570186, .397515527919776 ]
```

Fig. 9: Computing the derivative of the polynomial f by a matrix multiplication.

```
> f := randpoly(x, degree = 12, terms = 10);
  fcoeffs := Matrix([seq(coeff(f, x, i), i = 0..degree(f))]) :
  μ := 0.125 :
  fcoeffsInLS := Multiply(fcoeffs, MonomialToLSMatrix(degree(f), μ)) :
  C := ComradeMatrix(degree(f), μ, convert(fcoeffsInLS, list)) :
  LinearAlgebra : −Eigenvalues(C) :
  rootList := [seq(%[i], i = 1..degree(f))]
  f := 75 x^12 − 92 x^10 + 6 x^9 + 74 x^8 + 72 x^7 + 37 x^6 − 23 x^5 + 87 x^4 + 44 x^3 + 29 x
  rootList := [ 1.141014116 + 0.529676826 I, 1.141014116 − 0.529676826 I,
   −1.471711477 10^−8 + 0 I, 0.563935206 + 0.629828559 I, 0.563935206 − 0.629828559 I,
  −0.4044380673 + 0.900856959 I, −0.404438067 − 0.900856959 I,
  0.1365209364 + 0.583947922 I, 0.136520936 − 0.583947922 I,
  −1.0327465955 + 0.373007812 I, −1.0327465955 + 0.373007812 I,
  −0.808571018 + 0 I ]
> ResidualError := norm(eval(f, x = rootList[1]), 2);
      ResidualError := 0.0129507410089049
```

Fig. 10: Computation of the roots of the polynomial f by finding the eigenvalues of the corresponding comrade matrix.

4.4 Computing the roots of polynomials in Legendre-Sobolev bases

Here, we have computed the roots of a polynomial $f(\lambda)$ of degree 12, which is randomly created by using the command `randpoly`. We have first computed the coefficient matrix corresponding to $f(\lambda)$ in the Legendre-Sobolev basis corresponding to $\mu = 0.125$; then we have used this matrix to compute the comrade matrix C of $f(\lambda)$ in the Legendre-Sobolev basis (see [4]), and the roots of $f(\lambda)$ by computing the eigenvalues of matrix C (see Fig. 10).

4.5 Computing gcds of polynomials in Legendre-Sobolev bases

When two polynomials $f(\lambda)$ and $h(\lambda)$ are given in a Legendre-Sobolev basis, then one can find the monic gcd of these polynomials in the same basis by using the command `GcdInLS` as illustrated in Fig. 11. The work [3] explains the theory behind `GcdInLS` command which is essentially based on solving Diophantine equations arising from a comrade matrix.

5 Handwriting recognition with LegendreSobolev package

In this section, we explain how to compute the parametrized approximations of handwritten curves in a Legendre-Sobolev basis using `LegendreSobolev` package. To compute such approximations, we have implemented a package called `HandwritingRecognitionTesting` (see Fig. 12). The functions in the latter package are implemented based on Section 3 and [2].

Handwritten curves are presented as points (x, y, t) with coordinates (x, y) and timestamp t. To compute the parametrized approximation of a handwritten curve in a Legendre-Sobolev basis, we first compute the arc-lengths at each

```
> Digits := 17 :
  μ := 0.125 :
  f1 := randpoly(x, degree = 10);
  g := randpoly(x, degree = 9);
  h1 := randpoly(x, degree = 8);
      f1 := 40 x^9 − 81 x^7 + 91 x^3 + 68 x^2 − 10 x + 31
      g := 55 x^8 − 28 x^6 + 16 x^4 + 30 x^3 − 27 x^2 − 15 x
      h1 := 72 x^8 − 87 x^7 + 47 x^6 − 90 x^4 + 43 x^3 + 92 x
> f := f1 g :
  fcoeffs := Matrix([seq(coeff(f, x, i), i = 0..degree(f))]) :
  fcoeffsInLS := Multiply(fcoeffs, MonomialToLSMatrix(degree(f), mu)) :
  h := h1 g :
  hcoeffs := Matrix([seq(coeff(h, x, i), i = 0..degree(h))]) :
  hcoeffsInLS := Multiply(hcoeffs, MonomialToLSMatrix(degree(h), mu)) :
  gcdLS := GcdInLS(convert(fcoeffsInLS, list), convert(hcoeffsInLS, list), μ);
      gcdLS := [ −0.67070707067873429, 0.11404958676877247, 0.23366585106067264,
  0.15867768593779621, 0.12037071477726489, −7.6571048185969832 10^−13,
  0.0039091068982669860, −4.9471370550988164 10^−17, 0.00003412524698281 5868 ]
> gcoeffs := Matrix([seq(gcdLS[i], i = 1..nops(gcdLS))]) :
  gcoeffsInMonomial := Multiply(gcoeffs, LSToMonomialMatrix(nops(gcdLS) − 1, μ)) :
  computedGcd := add(gcoeffsInMonomial[1][i] · x^(i−1), i = 1..nops(gcdLS)) :
  RelativePolynomialError := evalf( norm(simplify(%− g/lcoeff(g) ,2)) / norm( g/lcoeff(g) ,2) )
                RelativePolynomialError := 4.8855925361391707 10^−11
```

Fig. 11: Monic gcd computation using `LegendreSobolev` package.

```
> read "HandwrittingRectesting.mpl" :
  with(HandwritingRecognitionTesting);
  [ ApproximateCurveFromCurves, ApproximateCurveFromPoints,
  BoundingBox, MomentIntegrals, NormalizeArcLength ]
```

Fig. 12: The functions of `HandwritingRecognitionTesting` package.

time for which a data point (x, y) is collected. To do so, we use the command `NormalizeArcLength`(x, y, m), where x and y are the tables containing all x_i and y_i, respectively, which are sampled at t_i, for $i = 0, \ldots, m$, where m is the number of times a data point is collected (see Fig. 13).

The next step is to compute the matrix of moment integrals corresponding to the points which are given by x and y. To do so, we use the command `MomentIntegrals`$(x, y, ArcLength, L, numSteps)$, where $ArcLength$ is the table of arc lengths corresponding to the data points given by x and y at a time, L

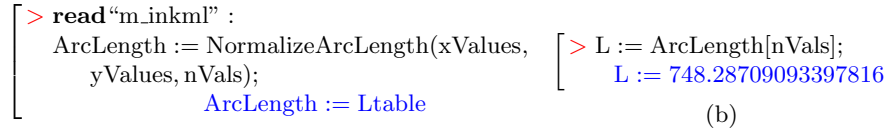

Fig. 13: (a) Arc-lengths computation of handwritten characters at each time a point is collected; (b) computing the total arc-length of a handwritten character.

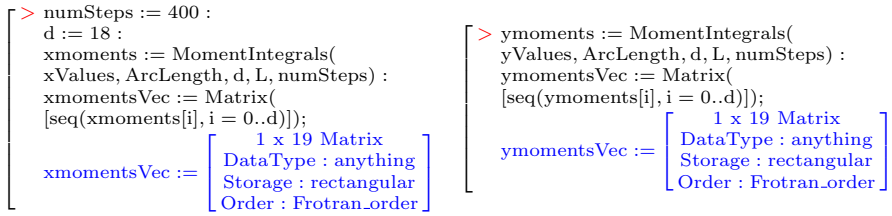

Fig. 14: Moments computation with `HandwritingRecognitionTesting` package.

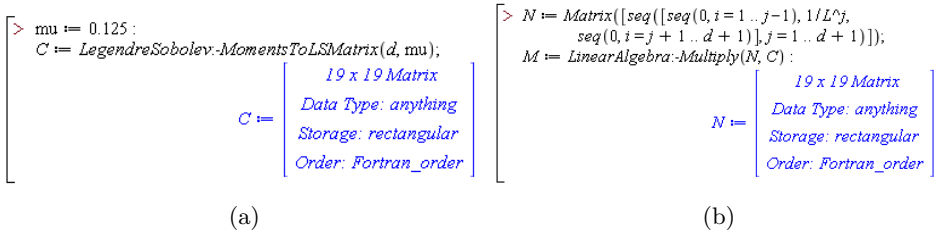

Fig. 15: (a) Matrix C computation with `LegendreSobolev` package. (b) Computation of matrix N to rescale the parametrized curves from $[0, L]$ to $[-1, 1]$.

the total arc length, and *numSteps* the number of steps in numerical integration for computing moment integrals (see Fig. 14).

Fig. 15a illustrates how to compute coefficients of an approximated curve in the Legendre-Sobolev basis with $\mu = 0.125$ from moments. The matrix which is computed by the command `MomentsToLSMatrix` is given by Proposition 1.

Fig 15b shows how to compute matrix N which scales a handwritten curve to be defined over $[-1, 1]$ instead of $[0, L]$. To scale the approximations to be defined over the interval $[-1, 1]$, we multiply two matrices N and C and compute the conversion matrix M. Note that the matrix C is independent of the problem and can be computed, in advance, but the matrix N is only known at the time a handwritten curve is completely written and pen is lifted up.

Now one can compute the coefficients of parametrized approximations of handwritten characters in the Legendre-Soboelv basis, with $\mu = 0.125$, by multiplying the moment integral matrices and conversion matrix M, see Fig 16.

```
> xCoeffsVec := xmomentsVec · M
                                  1 x 19 Matrix
xCoeffsVec :=                   DataType : anything
                                Storage : rectangular
                                Order : Frotran_order

> yCoeffsVec := ymomentsVec · M
                                  1 x 19 Matrix
yCoeffsVec :=                   DataType : anything
                                Storage : rectangular
                                Order : Frotran_order
```

Fig. 16: Computing the Legendre-Sobolev coefficients of the coordinate curves from moment integrals.

```
> xCoeffs := convert(xCoeffsVec, list);
  yCoeffs := convert(yCoeffsVec, list);
  XLS := add(xCoeffs[kk]S[kk1, mu](x), kk = 1..d + 1);
  YLS := add(yCoeffs[kk]S[kk1, mu](x), kk = 1..d + 1);
  XLS := 189.7334700734497143174.6127062900x^17 + 195134.26543686008x^15
  372294.97257625631x^13 + 392139.51651262814x^11253289.82060430780x^9
  +107985.69508828471x^7 + 259085.21825307995x^18
  1124537.0937011107000000x^16 + 2009145.0748245059000000x^14
  1899073.9014249837000000x^12 + 1016510.0846958727000000x^10
  309032.67958172588x^832230.336065891532x^5 + 54268.035000655102x^6
  +6111.3437078230404x^37124.1885409608675x^4 + 788.36849480813250x^2
  440.31476515699463x
  YLS := 287.6474665675798841062.137598260x^17 + 188484.7429259304x^15
  356524.0808428460x^13 + 357612.16332286471x^11204629.82886696718x^9
  +67770.765572894417x^7 + 285020.31716937217x^181270274.4758773500000000x^16
  +2379548.2831858383000000x^142430565.7566198024000000x^12
  +1463461.9788810968000000x^10519221.25154222754x^812983.274033705085x^5
  +99749.532330325308x^6 + 1403.1811689782580x^37549.097513537004x^4
  151.99583336566333x^2 + 39.1334243192727x
```

Fig. 17: Recovering the monomial representation of the coordinate curves.

After the coefficient matrices corresponding to Legendre-Sobolev approximations of the given handwritten curve are computed, we can recover the corresponding monomial representation and then plot the handwritten curve, see Figs. 17 and 18, respectively.

5.1 Baselines and cusps

One can compute the cusps and baselines of handwritten characters by computing the critical points of the parametrized approximations of the handwritten curves in a Legendre-Sobolev basis. The experimental results in [3], suggests to use quadruple precision for the calculations. To do so, one can compute the points corresponding to values of λ for which either $X'(\lambda) = 0$ or $Y'(\lambda) = 0$, but here we restrict ourselves to the latter case. We apply our calculations for the example which is given in Section 5 for the handwritten letter “m”, where the

```
> plt := plot([XLS, YLS, x = −1..1], color = black,
  legend = "The approximated handwritten curve in LS basis with d = 18, μ = 1/8") :
```

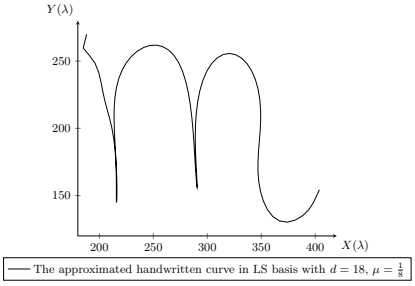

Fig. 18: The approximated curve which is constructed from moment integrals.

```
> Digits := 35 :
  yder := DerivativeInLS(mu, yCoeffs);
```

$$
\begin{aligned}
yder := [&-57.80172489055115070,\ 241.95681501221907573147241310639816,\\
&-72.00758972940440400672553544459136 8,\ 492.00134475978062028773451382443144,\\
&63.75423428512149041797064603850885 5,\ 58.99816519068351673720139527266 2948,\\
&7.11192325070875765571005103861763 82,\ 1.59685483656958251903351209653896 95,\\
&-0.90261179600677726273432558825127989,\ 0.15759274822840982821587390190311778,\\
&0.00493824811802938845646294027403994 83,\ 0.00138123499769584964169920792368 46833,\\
&-0.00011054971636315838410370568755104992,\ 0.00002317647414312603663887681642 6525092,\\
&-2.03446281050242111304257663768078 1\ 10^{-8},\ 4.29727128179245603791119269561544 63\ 10^{-7},\\
&-8.20863850624813849681799408813580 09\ 10^{-9},\ 2.15042115681719797899922415283294 32\ 10^{-9}]
\end{aligned}
$$

Fig. 19: Derivative computation using `LegendreSobolev` package.

degree of the approximation is $d = 18$ and $\mu = 0.125$. To do so, we first compute the coefficients of $Y'(\lambda)$ in the given Legendre-Sobolev basis. The command `DerivativeInLS` implements this functionality, see Fig. 19.

Then, we need to compute the comrade matrix corresponding to $Y'(\lambda)$. In fact, the roots of the system $\{Y'(\lambda) = 0\}$ are equivalent to the eigenvalues of the comrade matrix corresponding to $Y'(\lambda)$, Fig. 20. Now, we can compute the critical points on the handwritten curve as given in Fig. 21.

5.2 Regions of the characters in a handwritten math expression

Using the coefficients of the Legendre-Sobolev approximations of handwritten characters, one can find the corresponding regions of individual characters, automatically, by relying on computation of critical points of the corresponding parametrized approximations. In Fig. 22, we have approximated the individual

```
> C := ComradeMatrix( d − 1, mu, yder) :
  evalf(LinearAlgebra:-Eigenvalues(C)) :
  yroots := [seq(%[i], i = 1 ..d − 1)] :
  realyroots := [ ] :
   for i from 1 to d − 1 do
       if Im(yroots[i]) = 0 and Re(yroots[i]) > −1 and Re(yroots[i]) < 1 then
          realyroots := [op(realyroots), Re(yroots[i])];
       end if;
   end do:
   realyroots
[0.897228485150036392692957065316063777, 0.479466194110766822599786514520407 03,
   0.135636873906447859346942600194031 17, −0.226552286620585639752847634730784 15,
   −0.629065703169637588806049990032562 38]
```

Fig. 20: Critical point computation using `LegendreSobolev` package.

```
> pnts := [seq(eval([XLS, YLS], x = realyroots[i]), i = 1..nops(realyroots))] :
  b := plots : −pointplot(pnts, symbol = solidbox, color = red,
  legend = "Points on the approximated curve in LS basis with Y′(λ) = 0") :
  a1 := plot([XLS, YLS, x = −1..1], color = black,
  legend = "The approximated handwritten curve in LS basis with d = 18, μ = 1/8") :
  plots : −display([b, a1]);
```

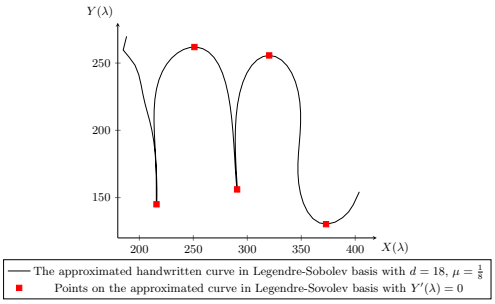

Fig. 21: Critical points corresponding to the approximated curve in Legendre-Sobolev basis constructed from moment integrals, with $\mu = 0.125$.

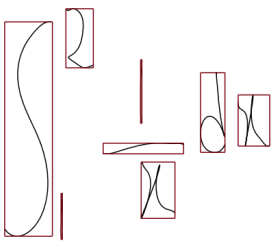

Fig. 22: Finding the regions of the characters in a math expression.

characters in a handwritten math expression by degree 10 polynomials in the Legendre-Sobolev basis with $\mu = \frac{1}{5}$. The command `BoundingBox`(Cx, Cy, μ) implements this method, where Cx and Cy are the $X(\lambda)$ and $Y(\lambda)$ coordinate approximation coefficients in the Legendre-Sobolev basis.

6 Concluding Remarks

The new MAPLE package `LegendreSobolev` performs various operations on polynomials in Legendre-Sobolev bases. All these operations rely on linear algebra arithmetic operations.

The package `LegendreSobolev` offers a command for recovering the coefficients of polynomials in Legendre-Sobolev bases from their moment integrals. This functionality is very useful in the problem of on-line handwriting recognition, when having the ability of real-time encoding of handwritten characters from their digital inks is crucial. It is also possible to study the geometrical features of handwritten characters by relying on computations of critical and singular points in Legendre-Sobolev bases.

Investigation of how these features might improve the mathematical handwritting recognition rates is a work in progress.

References

[1] P. Althammer. Eine Erweiterung des Orthogonalitätsbegriffes bei Polynomen und deren Anwendung auf die beste approximation. *J. Reine Ang. Math.*, 211:192–204, 1962.

[2] P. Alvandi and S. M. Watt. Real-Time Computation of Legendre-Sobolev Approximations. In *SYNASC*, pages 67–74, 2018.

[3] P. Alvandi and S. M. Watt. Handwriting Feature Extraction via Legendre-Sobolev Matrix Representation. 2019 (preprint).

[4] S. Barnett. A companion matrix analogue for orthogonal polynomials. *Linar Algebr. Appl*, 12(8):197–202, 1975.

[5] G. Carballo, R. lvarez Nodarse, and J.S. Dehesa. Chebychev polynomials in a speech recognition model. *Applied Mathematics Letters*, 14(5):581 – 585, 2001.

[6] B. W. Char and S. M. Watt. Representing and Characterizing Handwritten Mathematical Symbols through Succinct Functional Approximation. In *ICDAR*, volume 2, pages 1198–1202, 2007.

[7] O. Golubitsky and S. M. Watt. Online stroke modeling for handwriting recognition. In *CASCON*, pages 72–80, 2008.

[8] O. Golubitsky and S. M. Watt. Online computation of similarity between handwritten characters. In *Document Recognition and Retrieval XVI, part of the IS&T-SPIE Electronic Imaging Symposium*, pages C1–C10, 2009.

[9] O. Golubitsky and S. M. Watt. Online Recognition of Multi-Stroke Symbols with Orthogonal Series. In *ICDAR*, pages 1265–1269, 2009.

[10] F. Hausdorff. Summationsmethoden und Momentfolgen. I. 9:74–109, 1921.

[11] F. Hausdorff. Summationsmethoden und Momentfolgen. II. 9:74–109, 1921.

[12] R. Hu and S. M. Watt. Identifying Features via Homotopy on Handwritten Mathematical Symbols. In *SYNASC*, pages 61–67, 2013.

[13] K. Wang, N. An, B. N. Li, Y. Zhang, and L. Li. Speech emotion recognition using fourier parameters. *IEEE Transactions on Affective Computing*, 6(1):69–75, 2015.

[14] Y. Zhiqi. Gesture learning and recognition based on the chebyshev polynomial neural network. In *2016 IEEE Information Technology, Networking, Electronic and Automation Control Conference*, pages 931–934, 2016.

[15] L. Zhu and S. Zhu. Face recognition based on orthogonal discriminant locality preserving projections. *Neurocomputing*, 70(7):1543 – 1546, 2007. Advances in Computational Intelligence and Learning.