Bulletin of the *Transilvania* University of Braşov • Vol 9(58), No. 2 - 2016
Series III: Mathematics, Informatics, Physics, 153-160

ON THE COMPUTATION OF A TRIGONOMETRIC INTERPOLATION POLYNOMIAL

Ernest SCHEIBER[1]

Abstract

The note presents a method to obtain the trigonometric interpolation polynomial through the polynomial interpolation. In order to make an explicit computation the method will be programmed in a computation environment with polynomial computational facilities. Several examples are given with *Scilab* codes.

2000 *ACM Subject Classication:* G.1.1, I.1.4
Key words: interpolation, computation on polynomials

1 Introduction

The goal of this note is to present an explicit method to compute a trigonometric interpolation polynomial, more precisely the coefficients of the trigonometric polynomial. If the required trigonometric polynomial is

$$T(x) = \frac{a_0}{2} + \sum_{j=1}^{m}(a_j \cos jx + b_j \sin jx) = \sum_{k=-m}^{m} c_k e^{ikx} = \varphi(z) \tag{1}$$

with $z = e^{ix}$ and where $c_k = \frac{a_k - ib_k}{2}$, $c_{-k} = \bar{c}_k$, for $k \in \{1, 2, \ldots, m\}$ then the interpolation constraints are expressed for the polynomial $\Phi(z) = z^m \varphi(z)$.

In this way the trigonometric interpolation problem is reduced to a polynomial interpolation problem. We suppose that we have a function that computes the interpolation polynomial. From the solution of the polynomial interpolation problem the coefficients of the trigonometric interpolation polynomial are found.

The method may be programmed in a computation environment with polynomial computational facilities. *Scilab, Julia, Matlab* have such symbolic facilities. We have used *Scilab,* [6].

In *Matlab,* the function `trigint,` from the package *Interpolation Utilities,* computes the values of the trigonometric interpolation polynomial on a prescribed

[1]Faculty of Mathematics and Informatics, *Transilvania* University of Braşov, Romania, e-mail: scheiber@unitbv.ro

set of points, [5]. Barycentric formulas for some trigonometric interpolation polynomials are given in [1].

By imposing the interpolation constraints into (1), a linear system of algebraic equation results. The method presented in this note uses only direct formulas, avoiding the requirement to solve any additional problem, i.e. to solve a linear algebraic system.

Several examples are presented. To make the results reproducible we provide some code in Appendix.

2 Simple trigonometric interpolation problem

Given $-\pi \le x_1 < \ldots < x_{2m+1} < \pi$ and the numbers $y_1, \ldots, y_{2m+1}$ there exists the trigonometric polynomial (1) which satisfies the equalities $T(x_k) = y_k$, for any $k \in \{1, 2, \ldots, 2m+1\}$, [4]. The expression of the trigonometric interpolation polynomial is given by

$$T(x) = \sum_{k=1}^{2m+1} y_k \frac{\sin\frac{x-x_1}{2}\ldots\sin\frac{x-x_{k-1}}{2}\sin\frac{x-x_{k+1}}{2}\ldots\sin\frac{x-x_{2m+1}}{2}}{\sin\frac{x_k-x_1}{2}\ldots\sin\frac{x_k-x_{k-1}}{2}\sin\frac{x_k-x_{k+1}}{2}\ldots\sin\frac{x_k-x_{2m+1}}{2}},$$

but we shall not use this formula.

If $\Phi(z) = z^m\varphi(z)$ and $z_k = e^{ix_k}$ then $\Phi(z_k) = z_k^m y_k$, for any $k \in \{1, 2, \ldots, 2m+1\}$. It results that $\Phi(z)$ is the Lagrange interpolation polynomial

$$\Phi(z) = L(\mathbb{P}_{2m}; z_1, \ldots, z_{2m+1}; z_1^m y_1, \ldots, z_{2m+1}^m y_{2m+1})(z).$$

The Lagrange interpolation polynomial is computed using the recurrence formula

$$L(\mathbb{P}_k; z_1, \ldots, z_{k+1}; f)(z) =$$

$$= \frac{(z-z_1)L(\mathbb{P}_{k-1}; z_2, \ldots, z_{k+1}; f)(z) - (z-z_{k+1})L(\mathbb{P}_{k-1}; z_1, \ldots, z_k; f)(z)}{z_{k+1}-z_1}.$$

Here $\mathbb{P}_k$ denotes the vector space of all polynomials of degree at most k and the symbol f corresponds to the interpolated values. This is the goal of the function *LagrangePoly* (Appendix 5).

The *Scilab* function *TrigInterpPoly* (Appendix 5) computes the coefficients of the trigonometrical interpolation polynomial using the coefficients of the Lagrange interpolation polynomial.

Example 2.1. *For* $x\ :\ -\frac{2\pi}{3} < -\frac{\pi}{2} < 0 < \frac{\pi}{6} < \frac{\pi}{2}$ *and* $y\ :\ \frac{1}{2} - \frac{\sqrt{3}}{2}, 1, 4, 1 - \sqrt{3}, -4$ *compute the trigonometric interpolation polynomial. The data correspond to* $T(t) = 1 + \cos t - 2\sin t + 2\cos 2t - 3\sin 2t$.

We have obtained:

```
x=[-2*%pi/3,-%pi/2,0,%pi/6,%pi/2]
y=[-0.5-sqrt(3)/2,1,4,1-sqrt(3),-3]
[a,b]=TrigInterpPoly(x,y);
[a',b']
```

```
 1.   - 1.110D-16
 1.   - 2.
 2.   - 3.
```

The data in the columns are the coefficients of the computed trigonometric polynomial (1). It may be observed that we retrieve the starting trigonometric polynomial.

Example 2.2. *Compute the trigonometric interpolation polynomial of the function* $f(x) = x^2,\ x \in [-\pi, \pi]$, *for* $x_j = -\frac{\pi}{2} + j\frac{\pi}{n-1},\ j \in \{0, 1, \ldots, n-1\}$ *and* n *odd.*

For $n = 7$ we get

```
n=7
x1=linspace(-%pi/2,%pi/2,n)
y=x.^2
[a,b]=TrigInterpPoly(x,y);
[a',b']

    2.8687929  - 2.220D-16
  - 3.2277726    4.441D-16
    0.4013918  - 4.441D-16
  - 0.0424120  - 3.331D-16
```

Thus, the trigonometric interpolation polynomial is

$$T(x) \approx \tilde{T}(x) = 2.8687929 - 3.2277726 \cos x + 0.4013918 \cos 2x - 0.0424120 \cos 3x.$$

We shall verify the accuracy of the interpolation constraints computing the absolute error $e = \max_i |\tilde{T}(x_i) - f(x_i)|$. For different values of n, the obtained absolute error values are given in the next table:

n	e
7	$3.553\ 10^{-15}$
15	$1.654\ 10^{-12}$
21	$1.584\ 10^{-9}$
25	$4.355\ 10^{-8}$
31	0.0000081

The decrease of the accuracy is due to the roundoff errors and floating point arithmetic.

3 Osculatory trigonometric interpolation problem

The existence of the osculatory trigonometric polynomial is stated in the following theorem, [2],

Theorem 1. *Given two sets of* n *complex numbers,* $w_1, \ldots, w_n$ *and* $w'_1, \ldots, w'_n$ *there exists a trigonometric polynomial* f *of the form* $f(e^{ix}) = \sum_{k=-n}^{n} a_k e^{ikx}$ *with* $a_0 = 0$, *so that* $f(e^{ix_j}) = w_j$ *and* $f'(e^{ix_j}) = w'_j$ *for* $j = 1, 2, \ldots, n$, *where* $-\pi \leq x_1 < x_2 < \ldots < x_n < \pi$.

We shall use the Hermite polynomial, [3],

$$H_{2n-1}(z)=\sum_{k=1}^{n}f(z_k)\left(1-(z-z_k)\frac{u''_k}{u'_k}\right)l_k^2(z)+\sum_{k=1}^{n}f'(z_k)(z-z_k)l_k^2(z), \qquad (2)$$

where

$$u(z)=\prod_{k=1}^{n}(z-z_k),\quad u'_k=u'(z_k),\quad u''_k=u''(z_k),\quad l_k(z)=\tfrac{u(z)}{(z-z_k)u'_k} \qquad (3)$$

and where $z_1,z_2,\ldots,z_n$ are distinct complex points. Here $l_1(z),\ldots,l_n(z)$ are the Lagrange fundamental polynomials.

The polynomial H_{2n-1} satisfies the constraints $H_{2n-1}(z_k)=f(z_k)$ and $H'_{2n-1}(z_k)=f'(z_k)$, for any $k=1,2,\ldots,n$.

Let be $-\pi\le x_1<x_2<\ldots<x_n<\pi$ and the required trigonometric polynomial

$$T(x)=\sum_{j=1}^{n}(a_j\cos jx+b_j\sin jx)=\sum_{\substack{k=-n\\k\neq 0}}^{n}c_ke^{ikx}.$$

If the sets $w_1,\ldots,w_n$ and $w'_1,\ldots,w'_n$ are given then the interpolation constraints are $T(x_k)=w_k$ and $T'(x_k)=w'_k$, for any $k=1,\ldots,n$.

Denoting $z=e^{ix}, z_k=e^{ix_k}$ and $\varphi(z)=\sum_{\substack{k=-n\\k\neq 0}}^{n}c_kz^k$, the above interpolation constraints become $\varphi(z_k)=w_k$ and $\varphi'(z_k)=\frac{w'_k}{iz_k}$.

The polynomial

$$\Phi(z)=z^n\varphi(z)=\sum_{\substack{k=0\\k\neq n}}^{2n}c_{k-n}z^k=\sum_{\substack{k=0\\k\neq n}}^{2n}b_kz^k,\qquad (b_k=c_{k-n}), \qquad (4)$$

satisfies the equalities

$$\begin{aligned}\Phi(z_k)&=z_k^nw_k,\\ \Phi'(z_k)&=z_k^{n-1}(nw_k-iw'_k)\end{aligned}$$

for $k=1,2,\ldots,n$.

Taking into account (2) we compute the polynomial

$$H_{2n-1}(z)=\sum_{k=1}^{n}z_k^nw_k\left(1-(z-z_k)\frac{u''_k}{u'_k}\right)l_k^2(z)+\sum_{k=1}^{n}z_k^{n-1}(nw_k-iw'_k)(z-z_k)l_k^2(z).$$

This is a $2n-1$ degree polynomial while the degree of Φ is $2n$. Because the two polynomials and their derivatives take the same values on $z_1,\ldots,z_n$ there exists $k\in\mathbb{C}$ such that

$$\Phi(z)=ku^2(z)+H_{2n-1}(z). \qquad (5)$$

The constant k will be computed from the requirement that the coefficient of z^n of Φ must be 0.

The *Scilab* function $HermitePoly(x, y, z)$ (Appendix 6) computes the Hermite interpolation polynomial satisfying $H(x_k) = y_k$ and $H'(x_k) = z_k$ and the function $OsculatorTrigInterpPoly(x, y, z)$ (Appendix 6) computes the osculatory trigonometric interpolation polynomial satisfying $T(x_k) = y_k$ and $T'(x_k) = z_k$. .

Example 3.1. *For* $x \ : \ -\frac{2\pi}{3} < -\frac{\pi}{2} < 0 < \frac{\pi}{2}$ *we retrieve the trigonometric polynomial* $T(t) = \cos t + 2\sin t + 3\cos 3t + 10\sin 3t$ *when* y *and* z *are the values of* T *and* T' *on the given points.*

We have obtained:

```
x=[-2*%pi/3,-%pi/2,0,%pi/2]
y=[-2-sqrt(3),5,4,-11]
z=[29-5*sqrt(3)/2,1,32,-1]
[a,b]=OsculatorTrigInterpPoly(x,y,z);
[a',b']

    1.          2.
    3.          9.636D-15
    1.784D-14   10.
    8.910D-15   1.916D-14
```

Example 3.2. *Compute the osculatory trigonometric interpolation polynomial of the function* $f(x) = x^2, \ x \in [-\pi, \pi]$, *for* $x_j = -\frac{\pi}{2} + j\frac{\pi}{n}, \ j \in \{0, 1, \ldots, n\}$.

The results are:

```
n=5
x=linspace(-%pi/2,%pi/2,n)
y=x.^2
z=2*x
[a,b]=OsculatorTrigInterpPoly(x,y,z);
[a',b']

    1.895028   - 8.942D-14
  - 3.2361061    6.810D-14
    1.948192   - 4.481D-14
  - 0.7687050    2.014D-14
    0.1615910  - 4.723D-15
```

As above, the error of the interpolation constraints is computed and it is $1.297\ 10^{-13}$.

4 Conclusions

The coefficients of the trigonometric interpolation polynomial are computed via a Lagrange interpolation problem instead of computing its value in an arbitrary point.

APPENDIX

5 Codes for a simple trigonometric interpolation

```
function lag=LagrangePoly(x,y)
    z=poly(0,'X')
    n=length(x)
    if n~=length(y) then
        lag="The arguments must have the same length"
        return
    end
    v=zeros(1,n)
    w=zeros(1,n)
    v=y
    for k=1:n-1 do
        for i=1:n-k do
            w(i)=((z-x(i))*v(i+1)-(z-x(i+k))*v(i))/(x(i+k)-x(i))
        end
        for i=1:n-k do
            v(i)=w(i)
        end
    end
    lag=v(1)
endfunction
```

It may be observed that the transition from numeric to symbolic computation is made by introducing the polynomial $z = X$ through the function `poly`. In *Julia* this goal is achieved by using the functions `poly / Poly` from the package `Polynomials` while in *Matlab* through the function `poly2sym`.It is important to extract the coefficients of a polynomial, too.

```
function [a,b]=TrigInterpPoly(x,y)
    n=length(x)
    m=round((n-1)/2)
    a=zeros(1,m+1)
    b=zeros(1,m+1)
    if n~=length(y) then
        disp("The arguments must have the same length")
        return
    end
    if 2*m+1~=n then
        disp("The length of the arguments must be odd")
        return
    end
    x1=zeros(1,n)
    y1=zeros(1,n)
    x1=exp(%i*x)
    y1=x1.^m.*y;
    L=LagrangePoly(x1,y1)
    c=coeff(L)
    for j=1:m+1 do
        a(j)=2*real(c(m+j))
        b(j)=-2*imag(c(m+j))
    end
    a(1)=0.5*a(1)
endfunction
```

6 Codes for the osculator trigonometric interpolation

```
function p=HermitePoly(x,y,z)
    n=length(x)
    if n~=length(y) | n~=length(z) then
        p="The arguments must have the same length"
        return
    end
    X=poly(0,'X')
    w=poly(1,'X','coeff')
    for i=1:n do
        w=w*(X-x(i))
    end
    dw=derivat(w)
    d2w=derivat(dw)
    w1=zeros(1,n)
    w2=zeros(1,n)
    w1=horner(dw,x)
    w2=horner(d2w,x)
    p0=poly(0,'X','coeff')
    for i=1:n do
        p0=p0+(y(i)*(1-(X-x(i))*w2(i)/w1(i))+
          z(i)*(X-x(i)))*w^2/(X-x(i))^2/w1(i)^2
    end
    p=pdiv(numer(p0),denom(p0))
endfunction
```

```
function [a,b]=OsculatorTrigInterpPoly(x,y,z)
    n=length(x);
    a=zeros(1,n)
    b=zeros(1,n)
    if n~=length(y)| n~=length(z) then
        disp("The arguments must have the same length")
        return
    end
    x1=zeros(1,n)
    y1=zeros(1,n)
    z1=zeros(1,n)
    x1=exp(%i*x)
    y1=x1.^n.*y
    z1=x1.^(n-1).*(n*y-%i*z)
    H=HermitePoly(x1,y1,z1)
    X=poly(0,'X')
    U=poly(1,'X','coeff')
    for j=1:n do
        U=U*(X-x1(j))
    end
    cU=coeff(U^2)
    cH=coeff(H)
    k=-cH(n+1)/cU(n+1)
    p=k*U^2+H
    cp=coeff(p)
    for j=1:n do
        a(j)=2*real(cp(n+1+j))
        b(j)=-2*imag(cp(n+1+j))
    end
endfunction
```

References

[1] Berrut J.P., Trefethen L.N., *Barycentric Lagrange Interpolation.* SIAM Review **46**, no. 3, (2004), 501-517.

[2] Newman D.J., Rubel L.A., *On Osculatory Interpolation by Trigonometric Polynomials.* Internat. J. Math. & Math. Sci. **2**, no. 4, (1979), 717-720.

[3] Coman Gh., *Numerical Analysis.* Ed.Libris, Cluj (Romanian), 1995.

[4] Stancu D.D., Coman Gh., Agratini O., Trîmbiţaş R., *Numerical Analysis and Approximation Theory.* Vol. 1, Presa Universitară Clujeană, Cluj-Napoca (Romanian), 2001.

[5] * * *, `http://www.mathworks.com/matlabcentral/fileexchange/36800-interpolation-utilities` (2016).

[6] * * *, `www.scilab.org`