Peer-reviewed version available at *Math. Comput. Appl.* **2018**, *23*, 63; doi:10.3390/mca23040063

Article

A Computational Method with MAPLE for a Piecewise Polynomial Approximation to the Trigonometric Functions

Le Phuong Quan [1]*

[1] Department of Mathematics, College of Natural Sciences, Cantho University, 3/2 Street, Cantho City, Vietnam; lpquan@ctu.edu.vn

* Correspondence: lpquan@ctu.edu.vn

Abstract: A complete MAPLE procedure is designed to implement effectively an algorithm for approximating the trigonometric functions. The algorithm gives a piecewise polynomial approximation on an arbitrary interval, presenting a special partition that we can get its parts, subintervals with ending points of finite rational numbers, together with corresponding approximate polynomials. The procedure takes a sequence of pairs of interval-polynomial as its output that we can easily explore in some useful ways. Examples on calculating approximate values of the sine function with arbitrary accuracy for both of rational and irrational arguments as well as drawing the graph of the piecewise approximate functions will be presented. Moreover, from the approximate integration of integrands of the form $x^m \sin x$ on $[a, b]$, another MAPLE procedure is proposed to find the desired polynomial estimates in norm for the best L^2-approximation of the sine function in the vector space $\mathcal{P}_\ell$ of polynomials of degree at most ℓ, a subspace of $L^2(a, b)$.

Keywords: approximation; approximate value; evaluation error, approximation error; piecewise approximate polynomial; rational approximation; Taylor's Theorem

MSC: 41A10, 42A10, 65D17, 65D18

1. Introduction

There would be the two aspects of implementation of an algorithm. One is the illustration for the validity of the algorithm, and another for the efficient exploration of its steps thanks to the choice of suitable tools.

Remez's minimax algorithm is well established (see [4, Sec. 3.5]), but all the steps to proceed it contain the two main obstacles when being performed in practice:

- solving a nonlinear equation and a system of linear equations;
- using the cosine function to compute the initial set of points.

As warned to experienced users, they should have close control of the evaluation error at almost every calculation for the above steps. Therefore, it could be so difficult to implement this algorithm effectively for approximating the trigonometric functions themselves.

Algorithm 2 in [6] that is chosen here to be implemented with MAPLE (or any of Computer Algebra Systems (CAS), if possible) has the great advantage: only using arithmetic calculations on finite rational numbers and comparisons. The choice of MAPLE is due to its powerfulness of symbolic computation and its ability to display exact number of significant digits for the obtained numerical results. The description of **Algorithm 2** is clear, but examples on numerical integration and graphical plots in [6, Sec. 5–6] seem to be as of the first kind of implementation mentioned above. Therefore we come back to the article [6] with aspiration to provide the beginners or occasional users with a complete MAPLE procedure, easy and comfortable to explore its output when implementing **Algorithm 2**.

From [6, Sec. 5], we have that the function $\sin x$ is piecewise approximated on an interval $[a,b]$ by the polynomials $A_i(x)$ on the subintervals $[\alpha_i,\alpha_{i+1}]$, $i=0,\ldots,n$, such that $[a,b]=[a,\alpha_1]\cup[\alpha_1,\alpha_2]\cup\cdots\cup[\alpha_n,b]$, where $\alpha_0=a$ and $\alpha_{n+1}=b$ and

$$A_i(x)=(-1)^{(k_i-1)/2}\sum_{m=0}^{\lfloor n/2\rfloor}\frac{(-1)^m}{(2m)!}(x-k_ip_i)^{2m},$$

or

$$A_i(x)=(-1)^{k_i/2}\sum_{m=0}^{\lfloor (n-1)/2\rfloor}\frac{(-1)^m}{(2m+1)!}(x-k_ip_i)^{2m+1},\quad i=0,\ldots,n.$$

These approximations all have the accuracy of $1/10^r$ in absolute value for a given arbitrary positive interger r. The main purpose of **Algorithm 2** is to give the pairs of $\big[[\alpha_i,\alpha_{i+1}],A_i(x)\big]$, $i=0,\ldots,n$, and as a convention, the notations of this output will be used in the next sections.

The paper is organized as follows. In Section 2, we construct a complete MAPLE procedure by blocks of commands in the order that we can modify them easily. These blocks whose their own purpose will be clearly explained may be useful to design other procedures. Section 3, the longest section, is for exploration of the obtained results from the output of the procedure. The last part of Section 3 is destined for the detailed discussion on how to find a desired estimate p for the best L^2-approximation p_{best} of the sine function in the vector space $\mathcal{P}_\ell$. In particular, it is possible to make a complete MAPLE procedure to find p's with different values of ℓ from the existing materials, and we let the reader do it with his or her close control of evaluation error for the steps containing norms of vectors. Moreover, the crucial components of a procedure that computes approximate values of integration with integrands of the form $x^m\sin x$ ($m\in\mathbb{N}$) are also provided. Section 4 is for the conclusion.

2. Implementation by MAPLE Procedures

We list here the MAPLE commands that appear in our procedures. They all are very important and frequently used in MAPLE programming: `add`, `coeff`, `Digits`, `ERROR`, `evalf`, `floor`, `for`, `irem`, `int`, `nops`, `op`, `piecewise`, `RETURN`, `seq`, `sort`, `while`. A declaration to create a function, say `f`, as `f:=x->F(x)` or `f:=unapply(F(x),x)`, where `F(x)` is an expression in `x`, is very useful and convenient in MAPLE procedures for doing calculations. Also, the conditional structure `if-then` is indispensable in branch programming whereas the type `list` is a flexible ordered arrangement of operands (or things, elements). See [2], [3] and MAPLE help pages in each session to know more details about meaning, syntax and usage of these commands, structures and types.

In the following, we consider in succession the steps to perform **Algorithm 2** in [6, p. 15] with their content and corresponding MAPLE codes. However, we will modify some steps in **Algorithm 2** for convenience and for extracting necessary results from its output. Firstly, we recall the two blocks of commands (inside steps) for getting the degree n of the approximate polynomial $P_n^c(x)$ of the function $\sin x$, which satisfies the accuracy of $1/10^r$, and for finding the approximate values of $\pi/2$, which appear in $P_n^c(x)$.

Block 1

```
n:=0:  d:=1:
while (d>0) do
n:=n+1:
d:=(0.8)^(n+1)*10^(r+1)-(n+1)!;
end do:
```

Block 2

```
m:=r+2:
```

continued on the next page

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 August 2018 doi:10.20944/preprints201808.0231.v1

Peer-reviewed version available at Math. Comput. Appl. 2018, 23, 63; doi:10.3390/mca23040063

Block 2 (continued)

```
while ((b+3.2)*10^(r+3)-10^m>0) do
m:=m+1:
end do:
Digits:=m+4:
q:=evalf[m+3](Pi/2):
T:=x/q-floor(x/q):
if (T=0.5) then
k0:=floor(x/q):
else
while (2.4*10^m*min(T,0.5-T)-abs(x)<0) and
 (2.4*10^m*min(1-T,T-0.5)-abs(x)<0) do
m:=m+1:  Digits:=m+4:
q:=evalf[m+3](Pi/2):
T:=x/q-floor(x/q):
end do:
end if:
if (0.5<T) then
k0:=floor(x/q)+1:
else
k0:=floor(x/q):
end if:
if (irem(k0-1,2)=0) then
F:=unapply((-1)^((k0-1)/2)*add((-1)^s*(t-k0*q)^(2*s)/((2*s)!),
  s=0..floor(n/2)),t):
else
F:=unapply((-1)^(k0/2)*add((-1)^s*(t-k0*q)^(2*s+1)/((2*s+1)!),
  s=0..floor((n-1)/2)),t):
end if:
[k0,q,F];
```

Block 2 is nothing but the full content of `FindPoint`, a procedure to define nodes of the form kp', where $k \in \mathbb{Z}$ and p' is an approximate value of $\pi/2$ (see [6, pp. 11–12]). These nodes are ending points of subintervals in the partition of an arbitrary interval $[a, b]$. The output of `FindPoint` from its argument y also gives the approximate polynomial F. Moreover, we have them all together, k, p' and F from a list of three components; that is, we may write `FindPoint`$(y) = [k, p', P]$ with $p' \approx p = \pi/2$, $|\sin x - P(x)| < 1/10^r$ for all $x \in [kp' - p'/2, kp' + p'/2]$ and $|y - kp'| < 0.8$.

Next, Block 3 that may be the most important one is designed to determine approximate polynomials $A_j(x)$, corresponding to intervals $[\alpha_{j-1}, \alpha_j]$, where α_j, $j = 1, \ldots, i$, are the nodes mentioned above. This block gives a so-called speading technique that takes nodes together with the polynomials $A_j(x)$, by using Block 2 successively, as clearly described in [6, Sec. 5]. We choose the output of Block 3 as a function of a finite sequence of lists given in the form of

$$H := x \to \big[[\alpha_0, \alpha_1], A_0(x)\big],\ \big[[\alpha_1, \alpha_2], A_1(x)\big],\ \ldots,\ \big[[\alpha_{i-1}, \alpha_i], A_{i-1}(x)\big].$$

This output has the advantages of a function itself or a sequence of terms, because we can take its values $H(x)$ and $H(-x)$, or its components (or operands) $h_j(x) = \big[[\alpha_{j-1}, \alpha_j], A_{j-1}(x)\big]$ as we want.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 August 2018 doi:10.20944/preprints201808.0231.v1

Peer-reviewed version available at Math. Comput. Appl. 2018, 23, 63; doi:10.3390/mca23040063

We will design a MAPLE procedure named `ApproxFunct` to give the approximate function P for the sine function on an interval $[a,b]$ for $a \geq 0$. In the case of $0 \leq a < b \leq 0.8$ and $0 \leq a < 0.8 < b$, we may set $P = G$ on $[a,b]$ and $P = G$ on $[a, 0.8]$, respectively, where

$$G(x) = \sum_{m=0}^{\lfloor (n-1)/2 \rfloor} \frac{(-1)^m}{(2m+1)!} x^{2m+1}$$

and n is determined by Block 1. Before giving the MAPLE codes of Block 3 chosen as the full content of `TempApproxFunct`, the procedure to give the approximation to the sine function on $[a,b]$ only for $a \geq 0.8$, we recall here (from [6, p. 12]) the important notice: if $\alpha, \beta \in [a,b]$ are numbers such that

$$\texttt{FindPoint}(\alpha)\texttt{[1]} = \texttt{FindPoint}(\beta)\texttt{[1]} = k$$

with $p' = \texttt{FindPoint}(\alpha)\texttt{[2]}$, $p'' = \texttt{FindPoint}(\beta)\texttt{[2]}$ then we have $|\alpha - kp''| < 0.8$ and $|\beta - kp'| < 0.8$.

Block 3

```
n0:=FindPoint(b)[1]:
p0:=FindPoint(b)[2]:
B0:=FindPoint(b)[3]:
i:=0:
k[i]:=FindPoint(a)[1]:
p[i]:=FindPoint(a)[2]:
A[i]:=FindPoint(a)[3]:
u:=k[i]:
if n0<=u then
H:=x->[[a,b],A[0](x)]:
RETURN(H);
end if:
while u<n0 do
i:=i+1:
k[i]:=FindPoint(k[i-1]*p[i-1]+p[i-1])[1]:
p[i]:=FindPoint(k[i-1]*p[i-1]+p[i-1])[2]:
A[i]:=FindPoint(k[i-1]*p[i-1]+p[i-1])[3]:
u:=k[i]:
end do:
if i=1 then
 if b<=k[0]*p[0]+p[0]/2 then
H:=x->[[a,b],A[0](x)]:
RETURN(H);
 elif b<=k[0]*p[0]+p[0] then
H:=x->([[a,k[0]*p[0]+p[0]/2],A[0](x)],[[k[0]*p[0]+p[0]/2,b],A[1](x)]):
 RETURN(H);
 else
H:=x->([[a,k[0]*p[0]+p[0]/2],A[0](x)],[[k[0]*p[0]+p[0]/2,k[0]*p[0]+p[0]],
A[1](x)],[[k[0]*p[0]+p[0],b],B0(x)]):
 RETURN(H);
 end if:
elif i=2 then
 if b<=k[1]*p[1]+p[1]/2 then
H:=x->([[a,k[0]*p[0]+p[0]/2],A[0](x)],[[k[0]*p[0]+p[0]/2,b],A[1](x)]):
```

continued on the next page

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 August 2018 doi:10.20944/preprints201808.0231.v1

Peer-reviewed version available at Math. Comput. Appl. 2018, 23, 63; doi:10.3390/mca23040063

Block 3 (continued)

```
 RETURN(H);
 elif b<=k[1]*p[1]+p[1] then
H:=x->([[a,k[0]*p[0]+p[0]/2],A[0](x)],[[k[0]*p[0]+p[0]/2,k[1]*p[1]+p[1]/2],
A[1](x)],[[k[1]*p[1]+p[1]/2,b],A[2](x)]):
 RETURN(H);
 else
H:=x->([[a,k[0]*p[0]+p[0]/2],A[0](x)],[[k[0]*p[0]+p[0]/2,k[1]*p[1]+p[1]/2],
A[1](x)],[[k[1]*p[1]+p[1]/2,k[1]*p[1]+p[1]],A[2](x)],[[k[1]*p[1]+p[1],b],B0(x)]):
 RETURN(H);
 end if:
else
 if b<=k[i-1]*p[i-1]+p[i-1]/2 then
H:=x->([[a,k[0]*p[0]+p[0]/2],A[0](x)],seq([[k[m-1]*p[m-1]+p[m-1]/2,
k[m]*p[m]+p[m]/2],A[m](x)],m=1..i-2),[[k[i-2]*p[i-2]+p[i-2]/2,b],A[i-1](x)]):
 RETURN(H);
 elif b<=k[i-1]*p[i-1]+p[i-1] then
H:=x->([[a,k[0]*p[0]+p[0]/2],A[0](x)],seq([[k[m-1]*p[m-1]+p[m-1]/2,
k[m]*p[m]+p[m]/2],A[m](x)],m=1..i-1),[[k[i-1]*p[i-1]+p[i-1]/2,b],A[i](x)]):
 RETURN(H);
 else
H:=x->([[a,k[0]*p[0]+p[0]/2],A[0](x)],seq([[k[m-1]*p[m-1]+p[m-1]/2,
k[m]*p[m]+p[m]/2],A[m](x)],m=1..i-1),[[k[i-1]*p[i-1]+p[i-1]/2,k[i]*p[i-1]],
A[i](x)],[[k[i]*p[i-1],b],B0(x)]):
 RETURN(H);
 end if:
end if:
```

Note that `TempApproxFunct` takes three arguments in order as a, b and r. Next, we combine Block 1 and Block 3 with some conditional commands to form Block 4, which is the content of the `ApproxFunct`.

Block 4

```
G:=unapply(add((-1)^s*t^(2*s+1)/((2*s+1)!),s=0..floor((n-1)/2)),t):
if (a<0) then ERROR(‘1st argument must be nonegative‘);
elif (a<0.8) then
 if (b<=0.8) then
 [[a,b],G];
 else
 ([[a,0.8],G],TempApproxFunct(0.8,b,r));
 end if:
else
TempApproxFunct(a,b,r);
end if:
```

Now, we may call `ApproxFunct(`a`,`b`,`r`)` to fully access all subintervals together with approximate polynomials for the accuracy of $1/10^r$ on an interval $[a,b]$, $a \geq 0$. In particular, if we want to extract the jth interval and its corresponding approximate polynomial, use the calling sequence `ApproxFunct(`a`,`b`,`r`)(`x`)[`j`]`, where j is chosen from the output `ApproxFunct(`a`,`b`,`r`)(`x`)`.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 August 2018 doi:10.20944/preprints201808.0231.v1

Peer-reviewed version available at Math. Comput. Appl. 2018, 23, 63; doi:10.3390/mca23040063

Finally, from the above analysis, we obtain the desired procedure named `PiecewiseFunct`, which gives a special partition of an arbitrary interval $[a,b]$ into subintervals $[\alpha_{j-1},\alpha_j]$ together with corresponding approximate polynomials $A_j(x)$, where

$$|\sin x - A_j(x)| < \frac{1}{10^r} \text{ for all } x \in [\alpha_{j-1},\alpha_j],\ j = 1,\ldots,N,$$

with $N =$ `nops([PiecewiseFunct(`a`,`b`,`r`)])`. Here, there is a warning that in MAPLE, if `A:=a,b,c,d` or `A:=(a,b,c,d)`, we cannot define `nops(A)`; but, we can if `A:=[a,b,c,d]`, and `nops(A)` $= 4$. For all three cases, we can select ordered elements of `A`, namely `A[3]` $=$ `c`, for example.

Thus, a complete MAPLE procedure to perform **Algorithm 2** in [6, p. 15] is suggested to be

The `PiecewiseFunct` procedure

```
PiecewiseFunct:=proc(a::realcons,b::realcons,r::posint)
local ApproxFunct,j,n,d,G,num,funct,intrv;
ApproxFunct:=proc(a::realcons,b::realcons,r::posint)
local TempApproxFunct,n,d,G;option remember;
TempApproxFunct:=proc(a::realcons,b::realcons,r::posint)
local i,p0,n0,B0,u,p,k,A,H,FindPoint;option remember;
FindPoint:=proc(x::realcons)
local m,q,n,d,k0,F,T;option remember;
Block 1
Block 2
end proc:
Block 3
end proc:
Block 1
Block 4
end proc:
Block 1
G:=unapply(add((-1)^s*t^(2*s+1)/((2*s+1)!),s=0..floor((n-1)/2)),t):
if (0<=a) then
ApproxFunct(a,b,r)(x);
elif (-0.8<=a) then
 if (b<=0) then
 [[a,b],G(x)];
 else
 ([[a,0],G(x)],ApproxFunct(0,b,r)(x));
 end if:
else
 if (b<=0) then
 num:=nops([ApproxFunct(-b,-a,r)(-x)]):
 for j from 1 to num do
 funct[j]:=op(2,(-1)*ApproxFunct(-b,-a,r)(-x)[j]):
 intrv[j]:=sort(op(1,(-1)*ApproxFunct(-b,-a,r)(-x)[j])):
 end do:
 seq([intrv[num+1-i],funct[num+1-i]],i=1..num);
 else
 num:=nops([ApproxFunct(0,-a,r)(-x)]):
 for j from 1 to num do
 funct[j]:=op(2,(-1)*ApproxFunct(0,-a,r)(-x)[j]):
```

continued on the next page

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 August 2018 doi:10.20944/preprints201808.0231.v1

Peer-reviewed version available at Math. Comput. Appl. 2018, 23, 63; doi:10.3390/mca23040063

The PiecewiseFunct procedure (continued)

```
 intrv[j]:=sort(op(1,(-1)*ApproxFunct(0,-a,r)(-x)[j])):
 end do:
 (seq([intrv[num+1-i],funct[num+1-i]],i=1..num),ApproxFunct(0,b,r)(x));
 end if:
end if:
end proc:
```

The last part of the PiecewiseFunct procedure might need to be explained in more details. In the case $a < b \leq 0$, we make a partion first for the interval $[-b, -a]$; then, from the result of the partition, we take a sample of the form $\left[[\beta_{j-1}, \beta_j], B_j(x)\right]$ and convert it into its symetric part in the partition of $[a, b]$: $\left[[-\beta_j, -\beta_{j-1}], -B_j(-x)\right]$. Such a sample in MAPLE language is given by the declaration: `[intrv[j],funct[j]]`, where

```
intrv[j]:=sort(op(1,(-1)*ApproxFunct(-b,-a,r)(-x)[j])),
funct[j]:=op(2,(-1)*ApproxFunct(-b,-a,r)(-x)[j]).
```

Because the converted intervals should be arranged in the correct order on the real axis, we use the calling sequence `seq([intrv[num+1-i],funct[num+1-i]],i=1..num)`.

In fact, PiecewiseFunct contains a technique that indirectly solves a difficult problem "How to reduce values of arguments when doing calculations with the trigonometric functions". There was a great attempt to solve the problem and perhaps, [4, Chap. 9] would be one of the best reference books on this fact. However, this technique is only a suitable remedy for applying Taylor's Theorem, which has not been considered a good way in approximation theory.

PiecewiseFunct can be also used to give pointwise approximate values of the sine function, so we do not need to recall here **Algorithm 1** (see [6, p. 10]). A hint: combining Block 1, Block 2 and the illustrated MAPLE codes for the notice of "r-th decimal digit" and "r significant digits" on [6, p. 10] to make a procedure, say Sine, to implement **Algorithm 1**.

3. Exploration of the Output of the PiecewiseFunct Procedure

Firstly, we save the result from performing PiecewiseFunct in a variable chosen as a list of lists `A:=[PiecewiseFunct(a,b,r)]`, then put `num:=nops(A)`. Now, we number the ending points of subintervals and corresponding approximate polynomials, for instance, by a for-loop:

```
for i from 1 to num do
 a[i]:=op(1,op(1,A[i])):
 b[i]:=op(2,op(1,A[i])):
 F[i]:=op(2,A[i]):
end do:
```
(1)

For a given number $\alpha \in [a, b]$, we can choose the index `i` such that the interval `[a[i],b[i]]` contains α, hence we put $g :=$ `unapply(F[i],x)`. Note that `F[i]` is still an expression in `x` as default, not a function; besides, we cannot set $g :=$ `x->F[i]`, because `x` here and `x` in `F[i]` are not the same by MAPLE's rule. Then, $\sin\alpha \approx g(\alpha)$ with the accuracy of $1/10^r$. In MAPLE, we can use the command piecewise for a sequence of conditional settings to get the piecewise polynomial approximation to the sine function on $[a, b]$ by putting

```
f := x->piecewise(seq([a[j]<=x and x<=b[j],F[j]][],j=1..num),NULL):
```
(2)

Now, we have $\sin x \approx f(x)$ for all $x \in [a, b]$ with the accuracy of $1/10^r$.

In the following, we take examples on approximating values of the sine function at rational and irrational arguments and getting the graph of approximate functions on an arbitrary intervals with various accuracy. Moreover, the approximation of integration on $[a,b]$ with integrands of the form $x^m \sin x$ ($m \in \mathbb{N}$) will be considered in both theoretical and practical aspects.

To find the approximate value of $\sin(123.45)$ with the accuracy of $1/10^{20}$, we put $A :=$ `PiecewiseFunct(123,124,20)`. Then, `nops(`A`)` $= 2$ and the interval $[123,124]$ is partitioned into the subintervals $[123, 123.3075117]$ and $[123.3075117, 124]$. Thus, we take $f :=$ `unapply(op(2,`A`[2]),x)` and obtain

$$\begin{aligned} f(x) = &-1 + \frac{(x-x_0)^2}{2} - \frac{(x-x_0)^4}{24} + \frac{(x-x_0)^6}{720} - \frac{(x-x_0)^8}{40320} \\ &+ \frac{(x-x_0)^{10}}{3628800} - \frac{(x-x_0)^{12}}{479001600} + \frac{(x-x_0)^{14}}{87178291200} - \frac{(x-x_0)^{16}}{20922789888000} \\ &+ \frac{(x-x_0)^{18}}{6402373705728000} - \frac{(x-x_0)^{20}}{2432902008176640000}, \end{aligned}$$

where $x_0 = 123.09290981679683291927441363 6$. Hence, we have the desired approximation

$$\sin(123.45) \approx \texttt{evalf[20]}(f(123.45)) = -0.80035463532671180916.$$

Now, with an irrational number x, how do we approximate the value of $\sin x$ with the accuracy of $1/10^r$? In principle, we can find a rational number x' such that $|x - x'| < 1/10^{r+1}$ and use the `PiecewiseFunct` procedure to determine a polynomial P that satisfies $|\sin(x') - P(x')| < 1/10^{r+1}$. Then, we have the estimate $|\sin x - P(x')| < 1/10^r$. For example, we approximate the value of $\sin(\sqrt{3})$ with the accuracy of $1/10^{50}$ by taking first

$$\sqrt{3} \approx x' = 1.7320508075688772935274463415058723669428052538103 81.$$

Since `nops(`A`)` $= 1$, with $A :=$ `[PiecewiseFunct(1.73,2,51)]`, we set $f :=$ `unapply(op(2,`A`[1]),x)`. Then, we have the needed estimate

$$\sin(\sqrt{3}) \approx \texttt{evalf[51]}(f(x')) = 0.98702664499035378399332439243967038895709261414 4764.$$

There have been many algorithms in the literature to find rational approximations to an irrational number with the desired accuracy, mostly using continued fractions. The theoretical basis of this classic problem can be found in the two great books [1, Chap. 10–11] and [5, Chap. 6–7]. CAS of course have their built-in commands to compute such approximations. Here, we may take x' by the calling sequence `evalf[51](sqrt(3))`.

Besides, from (1) with the declaration (2), we derive the graph of piecewise polynomial approximation to the sine function by the command `plot(f(x),x=a..b,numpoints=500)`. We chose the large values $r = 150$ and $r = 500$ for the intervals $[-200, -50]$ and $[-100, 600]$, respectively. Let us try the case $r = 500$ to see the display of 448 pairs of interval-polynomial with the accuracy of $1/10^{500}$. The corresponding graphs are depicted in Figure 1.

We may also explore `PiecewiseFunct` from the other side of our approximation. We have used Taylor polynomials with larger degrees when higher accuracy required so far. If we want to confine approximate polynomials to a fixed degree, we might relate this determination to the vector space $\mathcal{P}_\ell$, a subspace of $L^2(a,b)$. We know that in $\mathcal{P}_\ell$ there is the best polynomial approximation p_{best} of the sine function in L^2-norm $\|\cdot\| = \langle\cdot,\cdot\rangle^{1/2}$ of $L^2(a,b)$, which is endowed with the inner product

$$\langle \varphi, \psi \rangle = \int_a^b \varphi(x)\psi(x)dx.$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 August 2018 doi:10.20944/preprints201808.0231.v1

Peer-reviewed version available at Math. Comput. Appl. 2018, 23, 63; doi:10.3390/mca23040063

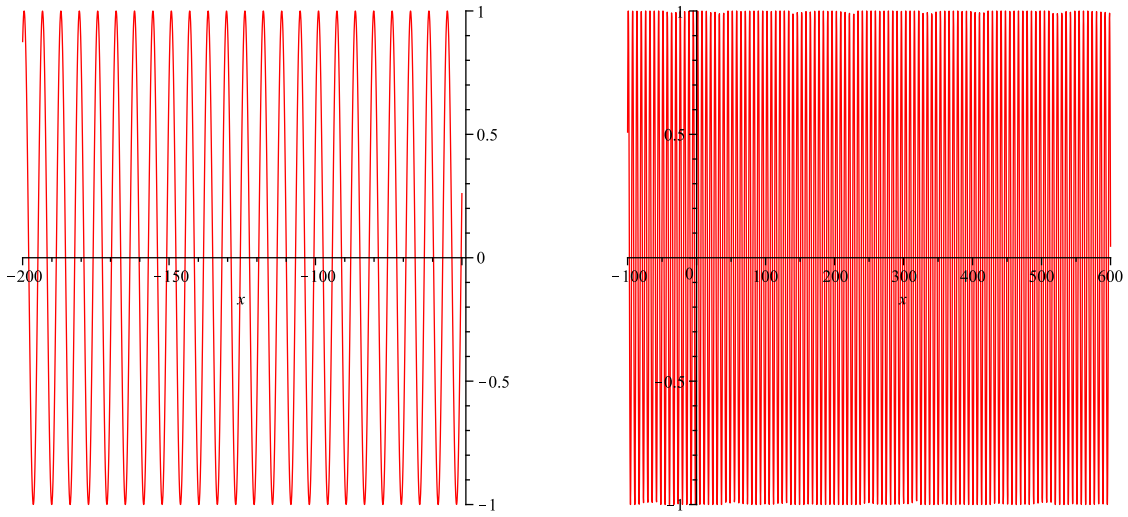

Figure 1. The graphs of f with $r = 150$ (left) and $r = 500$ (right).

Now, we use the Gram-Schmidt procedure to find an orthonormal basis $\{p_0, p_1, \ldots, p_\ell\}$ for $\mathcal{P}_\ell$ from the basis $\{1, x, x^2, \ldots, x^\ell\}$, according to the recursion

$$p_0 := 1/\|1\| = 1/\sqrt{b-a}, \quad q_k := x^k - \sum_{i=0}^{k-1} \langle x^k, p_i \rangle p_i, \quad p_k := q_k/\|q_k\|, \ k = 1, \ldots, \ell. \tag{3}$$

Once the orthonormal basis has been found, we obtain

$$p_{\text{best}} = \langle \sin, p_0 \rangle p_0 + \langle \sin, p_1 \rangle p_1 + \cdots + \langle \sin, p_\ell \rangle p_\ell = \sum_{k=0}^{\ell} \langle \sin, p_k \rangle p_k.$$

To give an estimate for $\|\sin - F\|$, where F is our piecewise polynomial approximation to the sine function on $[a, b]$, we first approximate $\langle \sin, p_k \rangle$ by

$$\langle \sin, p_k \rangle = \int_a^b p_k(x) \sin x dx \approx \langle F, p_k \rangle = \int_a^b p_k(x) F(x) dx.$$

These estimates have the absolute error

$$|\langle \sin, p_k \rangle - \langle F, p_k \rangle| = |\langle \sin - F, p_k \rangle| \leq \|\sin - F\|,$$

where

$$\|\sin - F\| = \Big(\int_a^b |\sin x - F(x)|^2 dx \Big)^{1/2} \leq \Big(\frac{b-a}{10^{2r}} \Big)^{1/2} = \frac{\sqrt{b-a}}{10^r}.$$

Therefore, if we choose

$$p_{\text{best}}^{\text{c}} := \sum_{k=0}^{\ell} \langle F, p_k \rangle p_k \tag{4}$$

as an approximation of p_{best} then we have the following estimation for the error norm

$$\|p_{\text{best}} - p_{\text{best}}^{\text{c}}\| = \Big\| \sum_{k=0}^{\ell} \langle \sin - F, p_k \rangle p_k \Big\| \leq \sum_{k=0}^{\ell} |\langle \sin - F, p_k \rangle| \leq \frac{(\ell+1)\sqrt{b-a}}{10^r}. \tag{5}$$

Hence, we may take $r = \lfloor \lg((\ell+1)\sqrt{b-a}) \rfloor + 1 + u$ to obtain the estimate

$$\|p_{\text{best}} - p^{\text{c}}_{\text{best}}\| < \frac{1}{10^u}.$$

To determine $p^{\text{c}}_{\text{best}}$ with MAPLE, it would be better to write the polynomial p_k in the form of $p_k := \sum_{s=0}^{k} a_{ks} x^s$, hence we have

$$\langle F, p_k \rangle = \sum_{s=0}^{k} a_{ks} \langle F, x^s \rangle, \text{ with } \langle F, x^s \rangle = \int_a^b x^s F(x) dx = \sum_{i=0}^{n} \int_{\alpha_i}^{\alpha_{i+1}} x^s A_i(x) dx. \tag{6}$$

According to the initial settings in (1), the coefficients a_{ks} and the inner products $\langle F, x^s \rangle$ can be computed by the following commands

$$a_{ks} \rightarrow \texttt{coeff(p[k],x,s)}$$
$$\langle F, x^s \rangle \rightarrow \texttt{add(int(x\^{}s*F[i],x=a[i]..b[i]),i=1..num)}$$

We can make a procedure named `PowerIntApprox` only to compute $\langle F, x^s \rangle$, $s = 0, \ldots, \ell$. This procedure that has one more argument, say "`deg`", for the chosen degree of p_{best} may take all the blocks of `PiecewiseFunct`, but replacing the output of Block 3, Block 4 and the last part of `PiecewiseFunct` with the commands recapped in the following. For the output `RETURN` of Block 3, we take the sum of integrals of $x^s A_i(x)$ on $[\alpha_i, \alpha_{i+1}]$, instead of giving the sequence of $[[\alpha_i, \alpha_{i+1}], A_i]$; and we do similarly for the output of Block 4. The last part of `PiecewiseFunct` may be replaced with the commands

```
if 0<=a then
evalf[r](ApproxInt(a,b,deg,r));
elif b<=0 then
evalf[r]([seq((-1)^(i+1)*ApproxInt(-b,-a,deg,r)[i+1],i=0..deg)]);
else
evalf[r]([seq((-1)^(i+1)*ApproxInt(0,-a,deg,r)[i+1],i=0..deg)]
    +ApproxInt(0,b,deg,r));
end if:
```

and the reason why we do so can be easily recognized. Here, `ApproxInt` has a similar role to `ApproxFunct`, but simpler to use.

Now, we have enough materials derived from (3), (4), (5) and (6) to make a procedure for finding a desired approximation on $[a, b]$ in L^2-norm to the best approximation p_{best} of the sine function in $\mathcal{P}_\ell$ with a given positive integer ℓ. The procedure takes four arguments a, b, ℓ and u to give $p \in \mathcal{P}_\ell$ as its output such that $\|p - p_{\text{best}}\| < 1/10^u$. As mentioned above, if we named the procedure `BestApprox`, hence it takes `PowerIntApprox` as its local variable, then it would be interesting to run `BestApprox` with different values of its input. But, as warned in Section 1, we should pay much attention to the cases of large values of ℓ and u in order to take an appropriate regulation.

4. Conclusion

There could be some more useful ways to explore the output of `PiecewiseFunct` and its corollary, `PowerIntApprox`, as well as `Sine`. Although these procedures are designed only for the sine function, we can make some appropriate changes to the blocks of commands to obtain their "cosine" version. It is hoped that some application results of our procedures would supply some more improved computational tools in applied mathematics.

Acknowledgments: The author is very grateful to Professor Henk Pijls (University of Amsterdam, the Netherlands) for his valuable advice and constant encouragement since 1998, and also wants to express his

thankfulness to the Maplesoft experts for their great work on developing MAPLE. The paper could not be well illustrated without MAPLE, their powerful and user-friendly product.

Conflicts of Interest: The author declares no conflict of interest.

References

1. Hardy, G.H.; Wright, E.M. (revised by Heath-Brown, D.R.; Silverman, J.H.). *An Introduction to the Theory of Numbers*; 6th ed.; Oxford University Press: Oxford, UK, 2008.
2. Monagan, M.B.; Geddes, K.O.; Heal, K.M.; Labahn, G.; Vorkoetter, S.M.; McCarron, J.; DeMarco, P. *MAPLE Introductory Programming Guide*; Copyright © Maplesoft. Waterloo MAPLE Inc.: Ontario, ON, Canada, 2008.
3. Monagan, M.B.; Geddes, K.O.; Heal, K.M.; Labahn, G.; Vorkoetter, S.M.; McCarron, J.; DeMarco, P. *MAPLE Advanced Programming Guide*; Copyright © Maplesoft. Waterloo MAPLE Inc.: Ontario, ON, Canada, 2008.
4. Muller, J.M. *Elementary Functions–Algorithms and Implementation*; 2nd ed.; Birkhäuser Boston: New York, NY, USA, 2006.
5. Niven, I.; Zuckerman, H.S.; Montgomery, H.L. *An Introduction to the Theory of Numbers*; 5th ed.; John Wiley & Sons, Inc.: New York, NY, USA, 1991.
6. Quan, L.P.; Nhan, T.A. Applying Computer Algebra Systems in Approximating the Trigonometric Functions. *Math. Comput. Appl.* **2018**, 23, 37. https://doi.org/10.3390/mca23030037