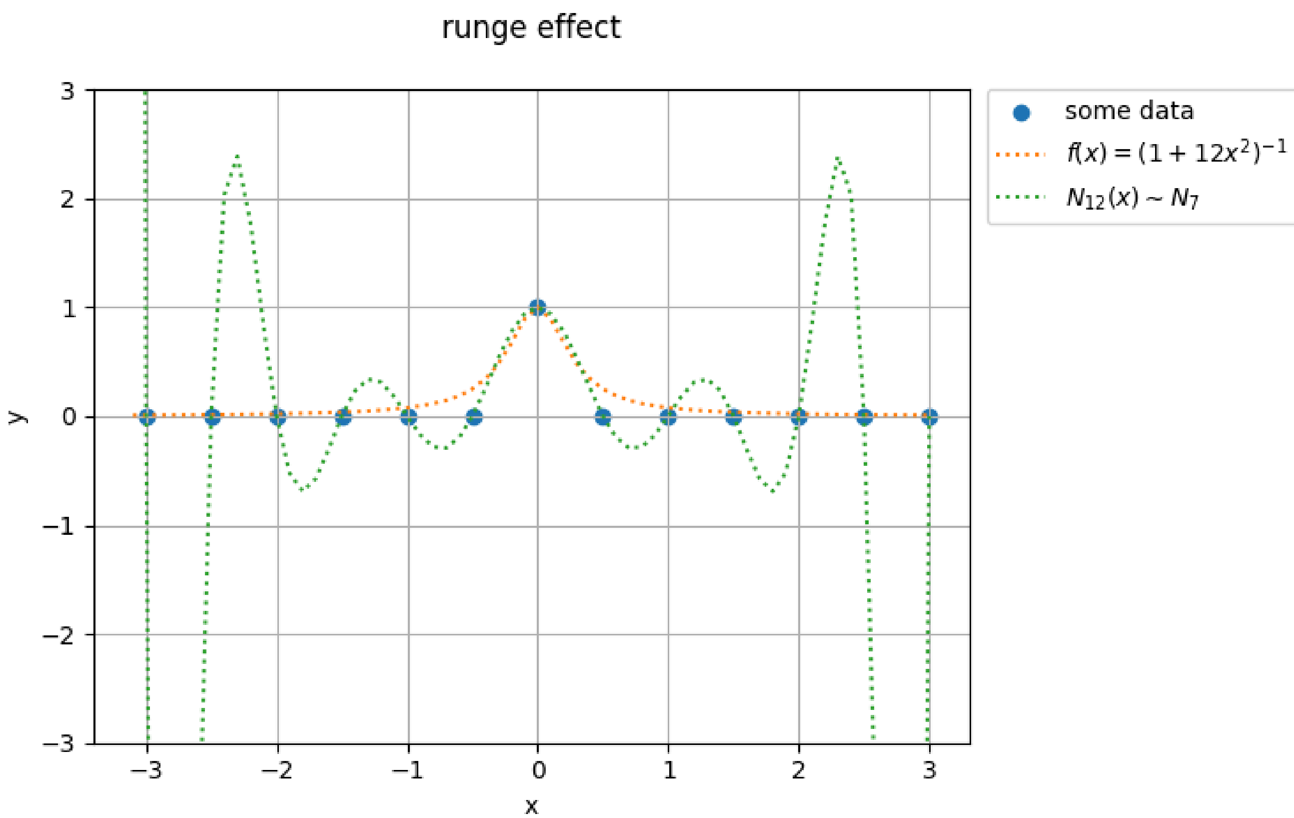
03.03 chebyshev



so that green line is the degree $n-1$ interpolating polynomial. useful, isnt it?

1 chebyshevs theorem

lagrange is simple, newtons divided difference is simpler in implementation and chebyshev is an implementation designed to improve control of the interpolation error.

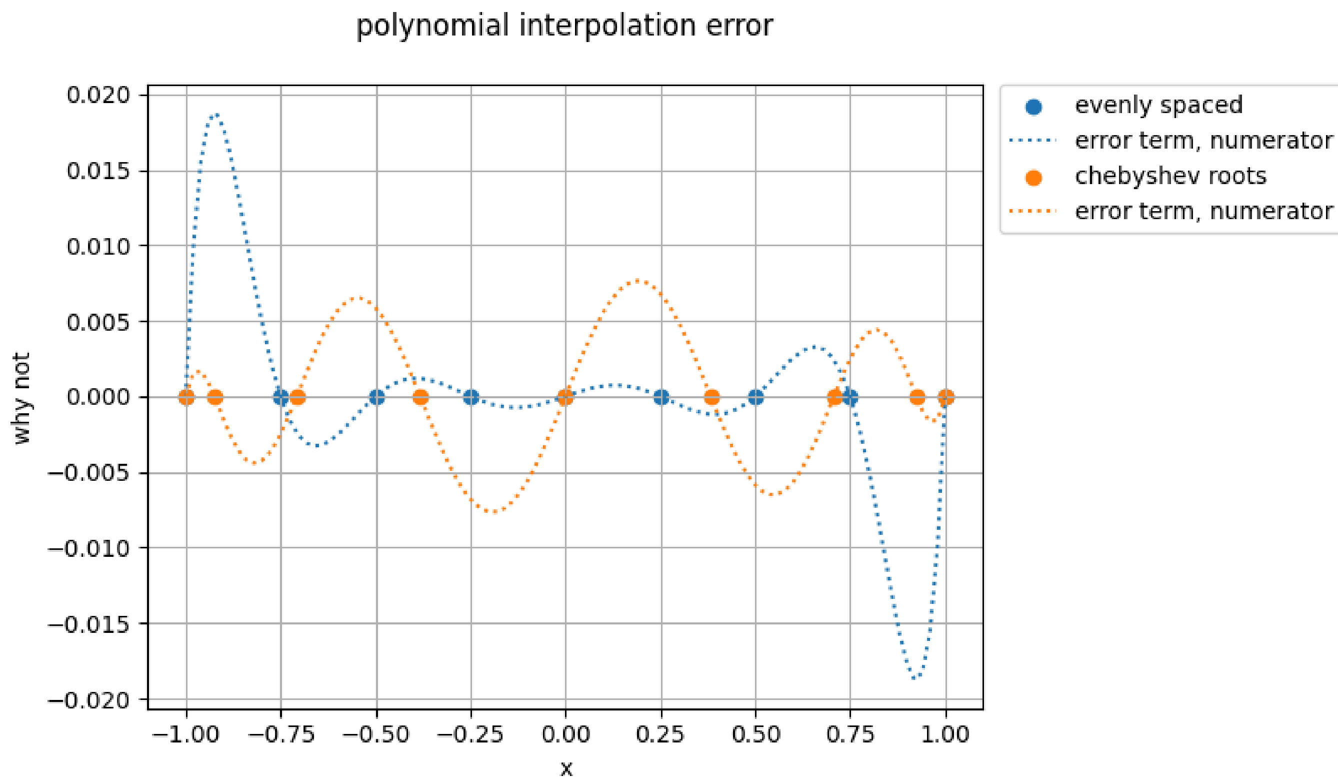
$$\frac{(x-x_1)(x-x_2)\ldots(x-x_n)}{n!} f^{(n)}(c)$$

on the interpolation interval. thats not everything but its a start.

consider interval $[-1,1]$. note that numerator is degree n polynomial with maximum on that interval. is there particular $x_1, \ldots, x_n$ to cause the maximum error to be as small as possible? this is the minimax problem of interpolation.

code, visual, polynomial interpolation error

```
import matplotlib.pyplot as plt
import numpy as np
from numpy import polynomial as npp

def err_num(x,xs): # ~ horners
  rc = 1
  for xi in xs:
    rc = rc*(x-xi)
  return rc

```



theorem 06 chebyshev interpolation

the choice of $x_i \in [-1,1]$ that makes the value of

$$\max_{x\in[-1,1]}|(x-x_1)\dots(x-x_n)|$$

as small as possible is

$$x_i = cos\frac{(2i-1)\pi}{2n} \quad i=1,\dots,n,$$

and its minimum is $\frac{1}{2^{n-1}}$. so the minimum is achieved by

$$(x - x_1)\dots(x - x_n) = \frac{1}{2^{n-1}} T_n(x),$$

where $T_n(x)$ denotes degree n chebyshev polynomial.

ie, interpolation error can be minimized if the n interpolation base points in $[-1, 1]$ are chosen to be the roots of the degree n chebyshev interpolating polynomial $T_n(x)$. these roots are

$$x_i = cos\frac{\text{odd } \pi}{2n}$$

where "odd" stands for the odd numbers from 1 to $2n - 1$. that guarantees the absolute value of the error term numerator is less than $\frac{1}{2n-1}$ for all $x \in [-1, 1]$.

choosing the chebyshev roots as the base points for interpolation distributes the interpolation error as evenly as possible across the interval $[-1, 1]$. the interpolating polynomial that uses chebyshev roots as base points is the **chebyshev interpolating polynomial**.

example 10

find worst-case error bound for the difference on $[-1, 1]$ between $f(x) = e^x$ and degree four chebyshev interpolating polynomial.

$$\text{error } \epsilon(x) = f(x) - P_4(x) = \frac{(x - x_1)(x - x_2)(x - x_3)(x - x_4)(x - x_5)}{5!} f^{(5)}(c),$$

where

$$x_1 = cos\frac{\pi}{10}, x_2 = cos\frac{3\pi}{10}, x_3 = cos\frac{5\pi}{10}, x_4 = cos\frac{7\pi}{10}, x_1 = cos\frac{9\pi}{10}$$

are the chebyshev roots and where $-1 \le c \le 1$. by theorem 05, for $-1 \le x \le 1$,

$$|(x - x_1)\dots(x - x_5)| \le \frac{1}{2^4}.$$

also $|f^{(5)}| \le e^1$ on $[-1, 1]$ and interpolation error is

$$|e^x - P_4(x)| \leq \frac{e}{2^4 \cdot 5!} \approx 0.00142 \quad \forall x \in [-1,1].$$

programmatically, chebyshev has slightly worse error in the middle and much better error at the end points.

definition chebyshev polynomials

define the nth **chebyshev polynomial** by $T_n(x) = cos(n\ arcos\ x)$. work through the trig, and itll look more conventional.

$$\begin{aligned}
n = 0 \quad &\mapsto \quad T_0(x) = cos(0 \cdot arcos\ x) = 1 \\
n = 1 \quad &\mapsto \quad T_1(x) = cos(1 \cdot arcos\ x) = x \\
n = 2 \quad &\mapsto \quad T_2(x) = cos(2 \cdot arcos\ x) = cos^2 y, \quad \text{where } y = arcos\ x \\
&\qquad\qquad\qquad = cos^2 y - sin^2 y - 1 \\
&\qquad\qquad\qquad = 2x^2 - 1 \quad \sim \text{ degree 2 polynomial. } \checkmark
\end{aligned}$$

in general,

$$\begin{aligned}
T_{n+1}(x) &= cos(n+1)y = cos(ny + y) = cos\ ny\ cos\ y - sin\ ny\ sin\ y \\
T_{n-1}(x) &= cos(n-1)y = cos(ny - y) = cos\ ny\ cos\ y - sin\ ny\ sin\ (-y)
\end{aligned}$$

$$\Downarrow \quad sin(-y) = -sin\ y$$

$$T_{n+1}(x) + T_{n-1}(x) = 2\ cos\ ny\ cos\ y$$

$$\Downarrow \quad y = arcos\ x$$

$$= 2 \cdot T_n(x) \cdot x = 2xT_n(x)$$

$$\Downarrow$$

$$T_{n+1}(x) = 2xTn(x) - T_{n-1}(x).$$

this is the **recursion relation** for chebyshev polynomials.

fact 01

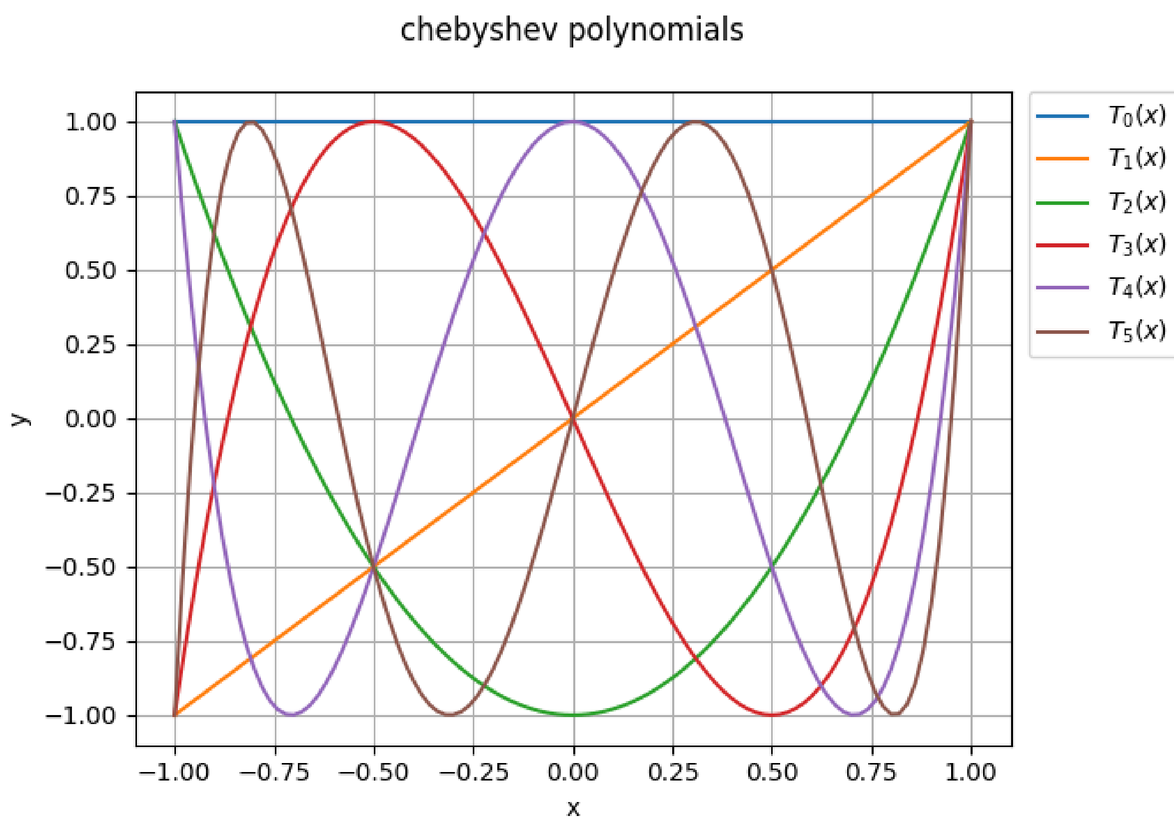
T_ns are polynomials. this was shown explicitly for T_0, T_1, T_2 and $T_3 = 2xT_2(x) - T_1(x) = 2x(2x^2 - 1) - x = 4x^3 - 3x$. usw.

fact 02

deg(T_n) = n and leading coefficient is 2^{n-1}. this is clear for $n = 1, n = 2$ and by recursion relation to all n.

code, visual, chebyshev polynomial degrees

```
import matplotlib.pyplot as plt
import numpy as np
from numpy import polynomial as npp

def main():
  t0 = lambda x: 1 #lol
  t1 = lambda x: x
  t2 = lambda x: 2*pow(x,2) - 1
```



fact 03

$T_n(1) = 1, T_n(-1) = (-1)^n$. clear for $n = 1, 2$ and in general,

$$T_{n+1}(1) = 2(1)T_n(1) - T_{n-1}(1) = 2(1) - 1 = 1 \text{ and}$$

$$\begin{aligned} T_{n+1}(-1) &= 2(-1)T_n(-1) - T_{n-1}(-1) \\ &= -2(-1) - (-1)^{n-1} \\ &= (-1)^{n-1}(2-1) = (-1)^{n-1} = (-1)^{n+1}. \end{aligned}$$

fact 04

maximum absolute value of $T_n(x)$ for $-1 \leq x \leq 1$ is 1. bc $T_n(x) = cos\ y$ for some y.

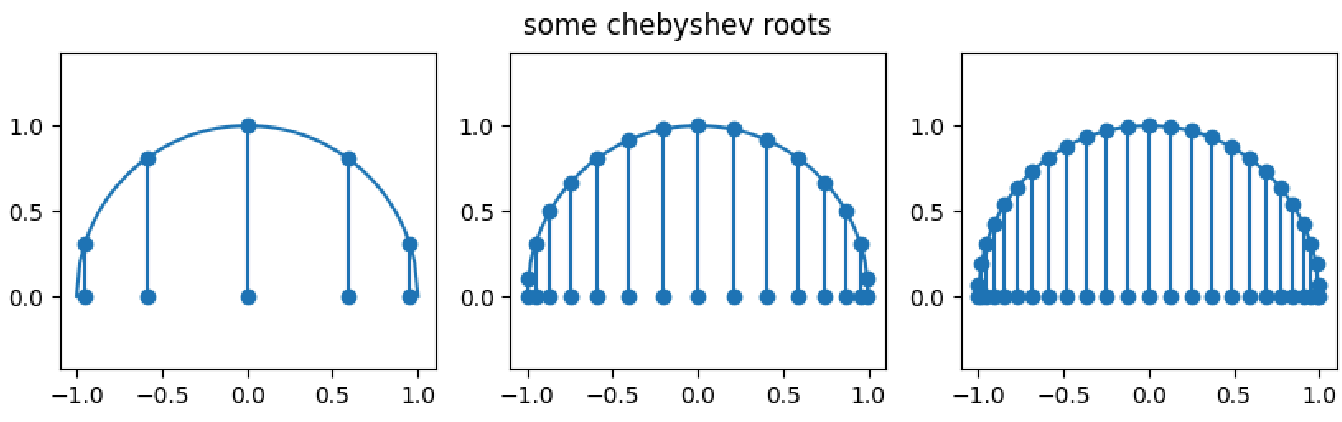
fact 05

all zeros of $T_n(x)$ are located between -1 and 1. the zeros are solutions of $0 = cos(n\ arcos\ x)$. bc $cos\ y = 0$ iif $y =$ odd integer $\cdot(\frac{\pi}{2})$,

$$\begin{aligned} n\ arcos\ x &= \text{odd} \cdot \frac{\pi}{2} \\ x &= cos \frac{\text{odd} \cdot \pi}{2n}. \end{aligned}$$

visual, chebyshev root spacing

```
import matplotlib.pyplot as plt
import numpy as np

def main():

  fig,axs = plt.subplots(1,3)
  fig.suptitle('some chebyshev roots\n')
```



fact 06

$T_n(x)$ alternates between -1 and 1 a total of $n+1$ times. this happens at $cos\ 0, cos\frac{\pi}{n}, \ldots, cos(n-1)\frac{\pi}{n}, cos\ \pi$.

usw

$\frac{T_n(x)}{2^{n-1}}$ is monic from fact 02. all roots of $T_n(x)$ are real from fact 05. so $\frac{T_n(x)}{2^{n-1}}$ in factored form is $(x-x_1)\ldots(x-x_n)$ where x_i are chebyshev nodes as described in theorem 08.

proof of theorem 06

let $P_n(x)$ be a monic polynomial with an even smaller absolute maximum on $[-1,1]$; ie, $|P_n(x)| < \frac{1}{2^{n-1}}$ for $-1 \leq x \leq 1$. this assumption leads to a contradiction. bc $T_n(x)$ alternates between -1 and 1 a total of $n+1$ times (fact 06), at these $n+1$ points the difference $P_n - T_n/2^{n-1}$ is alternately positive and negative. therefore, $P_n - T_n/2^{n-1}$ must cross zero at least n times; that is, it must have at least n roots. this contradicts the fact that, because $P_n, T_n/2^{n-1}$ are monic, their difference is of degree $\leq n-1$. ■

3.3.3 change of interval

so far our discussion of chebyshev interpolation has been restricted to the interval $[-1,1]$, because theorem 06 is most easily stated for this interval. next, scale to general interval $[a,b]$.

the base points are moved so that they have the same relative positions in $[a,b]$ that they had in $[-1,1]$. (1) stretch the points by the factor $(b-a)/2$; (2) translate the points by $(b+a)/2$ to move the center of mass from 0 to the midpoint of $[a,b]$. ie,

$$cos\frac{\text{odd }\pi}{2n} \quad \mapsto \quad \frac{b-a}{2}cos\frac{\text{odd }\pi}{2n} + \frac{b+a}{2}.$$

this also changes the numerator of the interpolation error term bc its upper bound will stretch by $\frac{b-a}{2}$ on each factor $x - x_i$. so replace the minimax value

$$\frac{1}{2^{n-1}} \quad \mapsto \quad \frac{(\frac{b-a}{2})^n}{2^{n-1}}.$$

chebyshev interpolation nodes

on interval $[a,b]$,

$$x_i = \frac{b+a}{2} + \frac{b-a}{2} cos \frac{(2i-1)\pi}{2n}$$

for $i = 1, \ldots, n$. the inequality

$$|(x-x_1)\ldots(x-x_n)| \leq \frac{(\frac{b-a}{2})^n}{2^{n-1}}$$

holds on $[a, b]$.

example 11

continues example 07, which used evenly spaced points.

find the four chebyshev base points for interpolation on the interval $[0, \frac{\pi}{2}]$ and find an upper bound for chebyshev interpolation error for $f(x) = sin\ x$ on the interval.

the chebyshev base points are

$$\frac{\frac{\pi}{2}-0}{2} cos\left(\frac{\text{odd } \pi}{2(4)}\right) + \frac{\frac{\pi}{2}+0}{2}.$$

$$\Downarrow$$

$$\begin{aligned}
x_1 &= \frac{\pi}{4} + \frac{\pi}{4} cos \frac{\pi}{8}, \\
x_2 &= \frac{\pi}{4} + \frac{\pi}{4} cos \frac{3\pi}{8}, \\
x_3 &= \frac{\pi}{4} + \frac{\pi}{4} cos \frac{5\pi}{8}, \\
x_4 &= \frac{\pi}{4} + \frac{\pi}{4} cos \frac{7\pi}{8}.
\end{aligned}$$

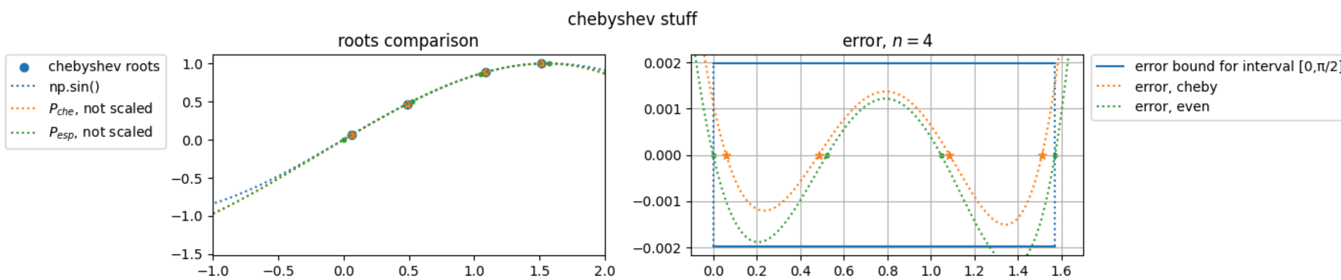
the worst-case interpolation error for $0 \leq x \leq \frac{\pi}{2}$ is

$$|sin\ x - P_3(x)| = \frac{|(x-x_1)(x-x_2)(x-x_3)(x-x_4)|}{4!} |f''''(c)| \leq \frac{\left(\frac{\frac{\pi}{2}-0}{2}\right)^4}{4!2^3} \cdot 1 \approx 0.00198$$

code

```
import matplotlib.pyplot as plt
import numpy as np
import scipy as sp
from tabulate import tabulate

def main():
  # known
  x = [0,np.pi/2] # interval
  n = 4 # roots
```

```
polynomial sine, degree 3:

         3           2
-0.1139 x - 0.06547 x + 1.02 x


polynomial sine, degree 3 with chebyshev roots:

         3           2
-0.1143 x - 0.06648 x + 1.023 x - 0.001131
```



x (rad)	np.sin()	P,che	error
1	0.841471	0.840831	0.000639605
2	0.909297	0.909747	0.000449781
3	0.14112	0.142019	0.000899202
4	-0.756802	-0.755505	0.00129727
14	0.990607	0.991736	0.00112865
1000	0.82688	0.826072	0.000807298

example 12

design a sine key that will give output correct to ten decimal places.

continues example 07.

with error bound 1e-10, calculate number of base points n required.

$$|sin\ x - P_{n-1}(x)| = \frac{|(x-x_1)\dots(x-x_n)|}{n!}\,|f^{(n)}(c)|$$

$$\leq \frac{\left(\frac{\frac{\pi}{2}-0}{2}\right)^n}{n!2^{n-1}} \cdot 1$$

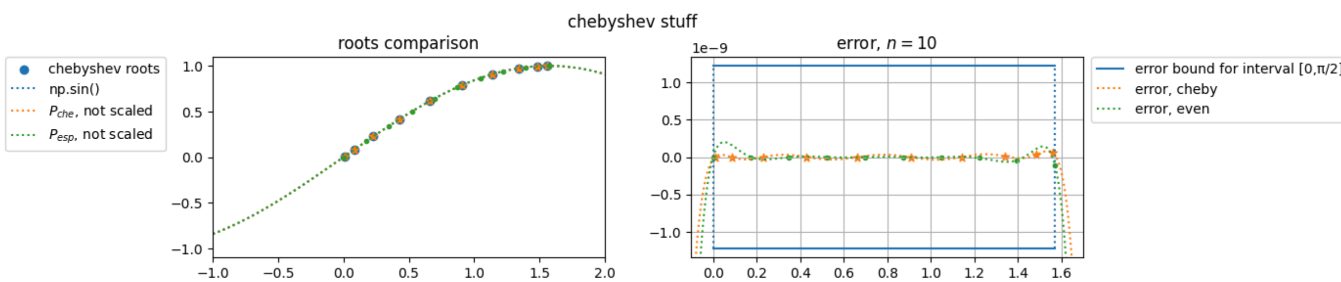
eventually this coughs up error bound $\approx 1.224e-9$ for $n=9$ and error bound $\approx 4.807e-10$ for $n=10$. with $n=10$, the chebyshev base points on $[0,\frac{\pi}{2}]$ are $\frac{\pi}{4}+(\frac{\pi}{4})\, cos(\frac{\text{odd}\,\pi}{20})$.

code

```
# copy first code from example 11 and set n = 10 ~ so difficult!
```

```
import matplotlib.pyplot as plt
import numpy as np
import scipy as sp
from tabulate import tabulate

def main():…

if __name__ == "__main__":…

```

```
polynomial sine, degree 9:

           9             8             7             6            5
1.926e-06 x + 3.677e-06 x - 0.0002062 x + 9.592e-06 x + 0.008326 x
              4           3             2
 + 3.534e-06 x - 0.1667 x + 1.566e-07 x + 1 x


polynomial sine, degree 9 with chebyshev roots:

           9             8             7             6            5
1.921e-06 x + 3.657e-06 x - 0.0002059 x + 8.877e-06 x + 0.008327 x
              4           3            2
 + 2.725e-06 x - 0.1667 x + 8.26e-08 x + 1 x + 3.104e-11
```



x (rad)	np.sin()	P,che	error
1	0.841471	0.841471	3.32245e-11
2	0.909297	0.909297	1.39222e-13
3	0.14112	0.14112	3.09897e-11
4	-0.756802	-0.756802	1.90662e-11
14	0.990607	0.990607	1.11141e-11
1000	0.82688	0.82688	2.70227e-11