# Taylor Series, Pade Approximants, and Neural Networks

Francis Benistant
Technology Modeling Automation
92 PUBLICATIONS 333 CITATIONS

# Taylor Series, Pade Approximants, and Neural Networks

The Taylor series are well known to be the approximation of a differentiable function, f(x), by a polynomial built-up on the derivatives of f(x).

$$f(x+h) = f(x) + \frac{h}{1!} f^{(1)}(x) + \frac{h^2}{2!} f^{(2)}(x) + \cdots + \frac{h^n}{n!} f^{(n)}(x) + \ldots \qquad (1)$$

Where n! is the factorial of n, and $f^{(n)}(x)$ is the nth derivative of f(x). Taylor series have been described in many papers and that will not be the focus of this article, which is devoted to show the connection between Taylor series and Pade Approximants, then to show how Pade approximants can be linked to Neural Networks.

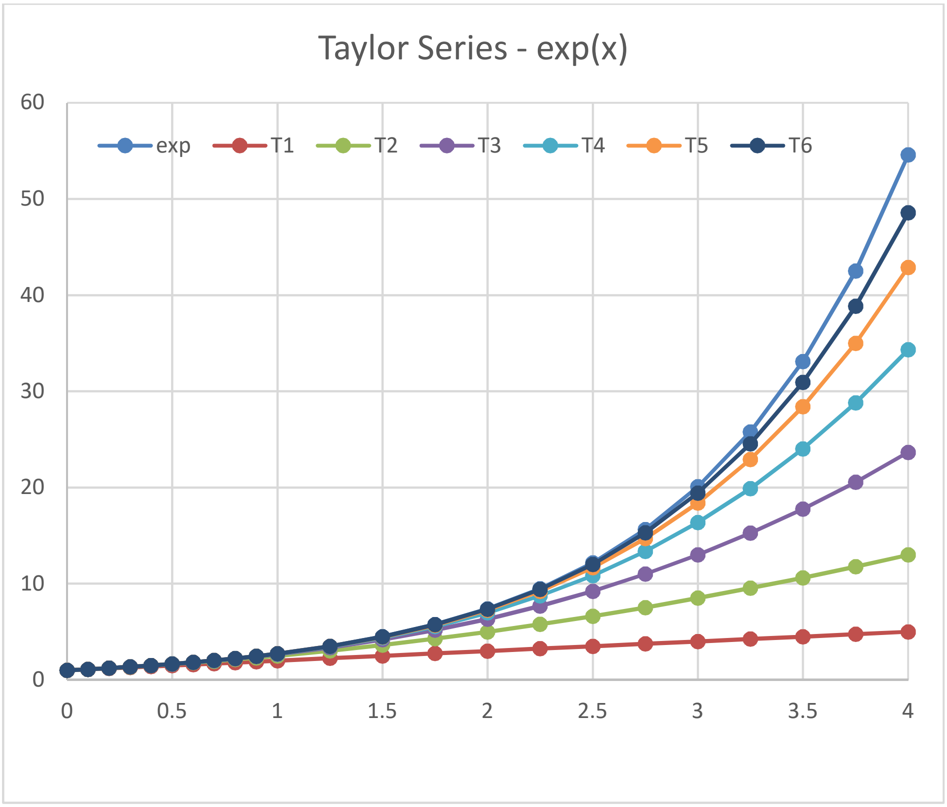
We will give a short example working with f(x) = exp(x). As the nth derivative of exp(x) is exp(x), it makes the calculation simpler.

If we use the formula (1) for exp(x), we can write the Taylor series as:

$$exp(x+h) = exp(x) + \frac{h}{1!} exp\ (x) + \frac{h^2}{2!} exp\ (x) + \cdots + \frac{h^n}{n!} exp(x) + \ldots \qquad (2)$$

$$exp(h) = 1.0 + \frac{h}{1!} + \frac{h^2}{2!} + \cdots + \frac{h^n}{n!} + \ldots \qquad (3)$$

We can plot exp(x) and the Taylor Series (T1 to T6) on [0.0, 4.0] interval:



As x increases, the Taylor Series diverges from the exponential function and we need more terms to reduce the difference between the function and the Taylor polynomials.

The Taylor Series are polynomial approximation of a function, and we can ask if there is a better way to approximate a function. Pade Approximants answer this question by using ratio of polynomials to approximate a function. The Pade Approximant is a rational function which numerator is a polynomial of order m and the denominator a polynomial of order n:

$$\text{P[m/n]}(x) = \frac{a0+a1\,x+a2\,x^2+\cdots+am\,x^m}{1+b1\,x+b2\,x^2+\,\ldots+bn\,x^n} \qquad (4)$$

Usually all the coefficients are normalized by b0 which is taken at 1.0 The coefficients am and bn are calculated such as P[m/n] matches the Taylor Series of order (m+n).

Let’s make short illustration with the exp(x) function searching for the P[1/1] Pade approximant.

$$\text{P[1/1]}(x) = \frac{a0+a1\,x}{1+b1\,x} \qquad (5)$$

The Taylor Series of order 2 for exp(x) is:

$$T2(x) = 1.0 + \frac{x}{1!} + \frac{x^2}{2!} \qquad (6)$$

As we match P[1/1](x) with T2(x) keeping only the terms of order 2, we have:

$$\frac{a0+a1\,x}{1+b1\,x} = 1.0 + x + \frac{x^2}{2}$$

$$a0 + a1\,x = \left(1 + b1\,x\right) * (1.0 + x + \frac{x^2}{2}) \qquad (7)$$

$$\begin{aligned} a0 &= 1.0 \\ a1\,x &= x + \text{b1x} \qquad (8) \\ 0.0 &= \frac{x^2}{2} + \text{b1}x^2 \end{aligned}$$

$$\begin{aligned} a0 &= 1.0 \\ a1 &= 1 + \text{b1} \qquad (9) \\ \text{b1} &= \frac{-1}{2} \end{aligned}$$

$$\begin{aligned} a0 &= 1.0 \\ a1 &= \frac{1}{2} \qquad (10) \\ \text{b1} &= \frac{-1}{2} \end{aligned}$$

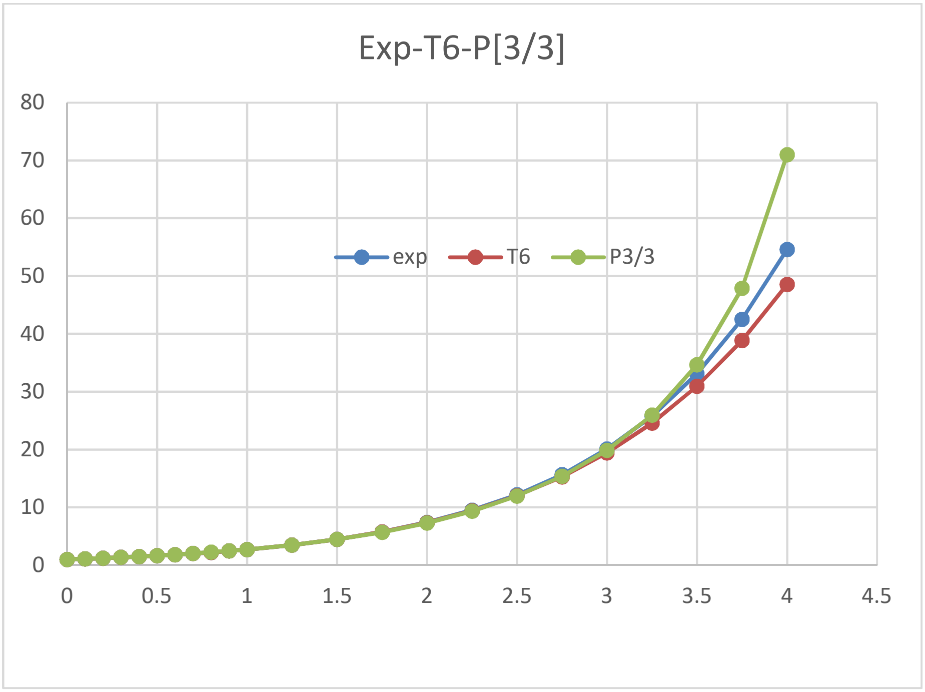
Finally the P[1/1](x) Pade Approximant for the exp(x) function is:

$$\mathrm{P}[1/1](\mathrm{x}) = \frac{1.0+0.5\,x}{1-0.5\,x} \qquad (11)$$

The P[3/3] Pade Approximant for exp(x) is given by (12) and can be compared to the T6(x) Taylor Series.

The P[3/3] shows slightly better match of the exp(x) function up to 3.5 while T6(x) starts to diverge at 3.25.

$$\mathrm{P}[3/3](\mathrm{x}) = \frac{1.0+0.5\,x+0.1x^2+\frac{x^3}{120}}{1-0.5x+0.1x^2-\frac{x^3}{120}} \qquad (12)$$



The Pade Approximants show all their benefits for function like sin(x). The Taylor series of sin(x) is given by the general formula below:

$$\sin(x) = \sum_{k=0}^{\infty} \frac{(-1)^k x^{1+2k}}{(1+2k)!} \qquad (13)$$

The Taylor polynomial up to the order 7 is:

$$\sin(x) = x - \frac{x^3}{6} + \frac{x^5}{120} - \frac{x^7}{65040} \qquad (14)$$

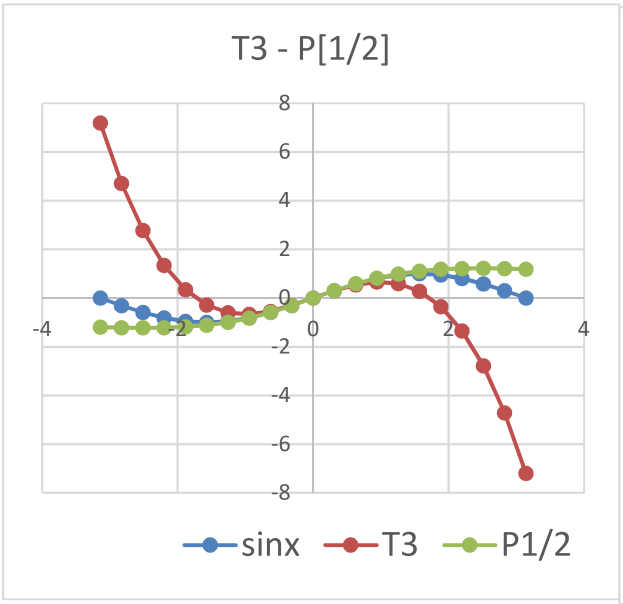
The Pade Approximants based on the Taylor Series can be calculated as done with the P[1/1] Approximant for exp(x).

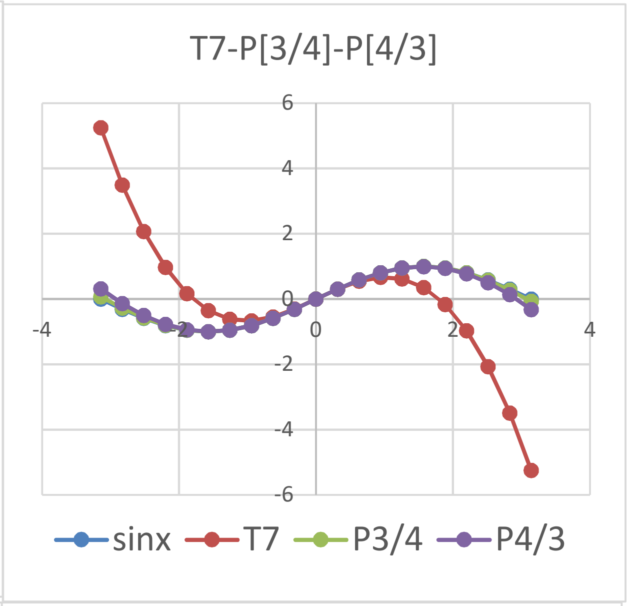
$$\mathrm{P}[1/2](\mathrm{x})=\mathrm{P}[2/3](\mathrm{x}) = \frac{x}{1.0+\frac{x^2}{6}} \qquad (15)$$

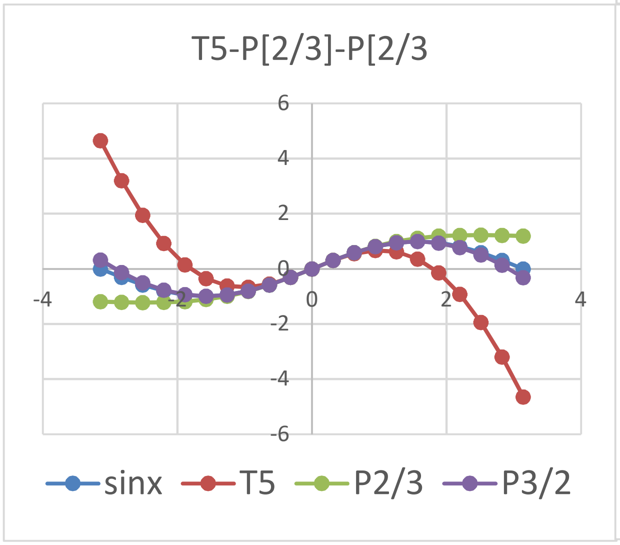
$$\mathrm{P}[3/2](\mathrm{x})=\mathrm{P}[4/3](\mathrm{x}) = \frac{x-\frac{7x^3}{60}}{1.0+\frac{x^2}{20}} \qquad (16)$$

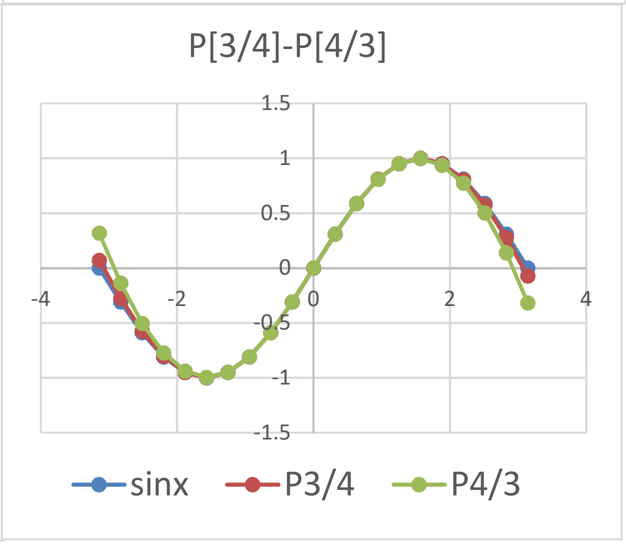
$$\mathrm{P}[3/4](\mathrm{x}) = \frac{x-\frac{31x^3}{294}}{1.0+\frac{3x^2}{49}+\frac{11x^4}{5880}} \qquad (17)$$

The plots below show the better fit of the Pade Approximant compared to the Taylor Series. The value of the x axis are between –pi and +pi.



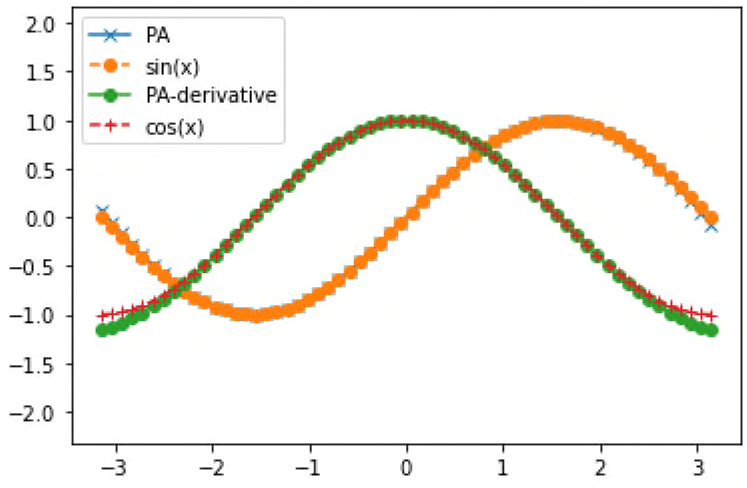


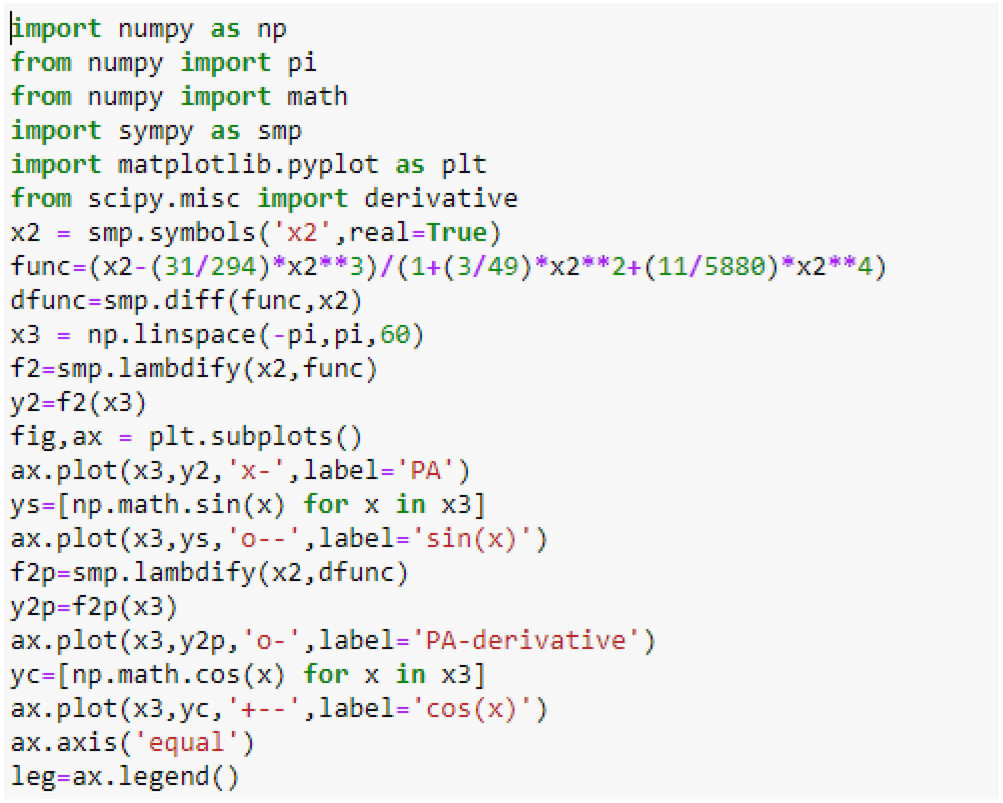




Note that P[3/4] and P[4/3] gives a very good match of the sinus function over all the x axis, P[3/4] gives the best fit of the function.

It is worth noting that the derivative of P[3/4] matches the derivative of sin(x) as shown on the graph below. The python script is also given for reference.





```
import numpy as np
from numpy import pi
from numpy import math
import sympy as smp
import matplotlib.pyplot as plt
from scipy.misc import derivative
x2 = smp.symbols('x2',real=True)
func=(x2-(31/294)*x2**3)/(1+(3/49)*x2**2+(11/5880)*x2**4)
dfunc=smp.diff(func,x2)
x3 = np.linspace(-pi,pi,60)
f2=smp.lambdify(x2,func)
y2=f2(x3)
fig,ax = plt.subplots()
ax.plot(x3,y2,'x-',label='PA')
ys=[np.math.sin(x) for x in x3]
ax.plot(x3,ys,'o--',label='sin(x)')
f2p=smp.lambdify(x2,dfunc)
y2p=f2p(x3)
ax.plot(x3,y2p,'o-',label='PA-derivative')
yc=[np.math.cos(x) for x in x3]
ax.plot(x3,yc,'+--',label='cos(x)')
ax.axis('equal')
leg=ax.legend()
```

The Pade Approximant can be much superior to the Taylor Series to capture nonlinear behavior. This property makes them interesting for Neural Networks modeling.

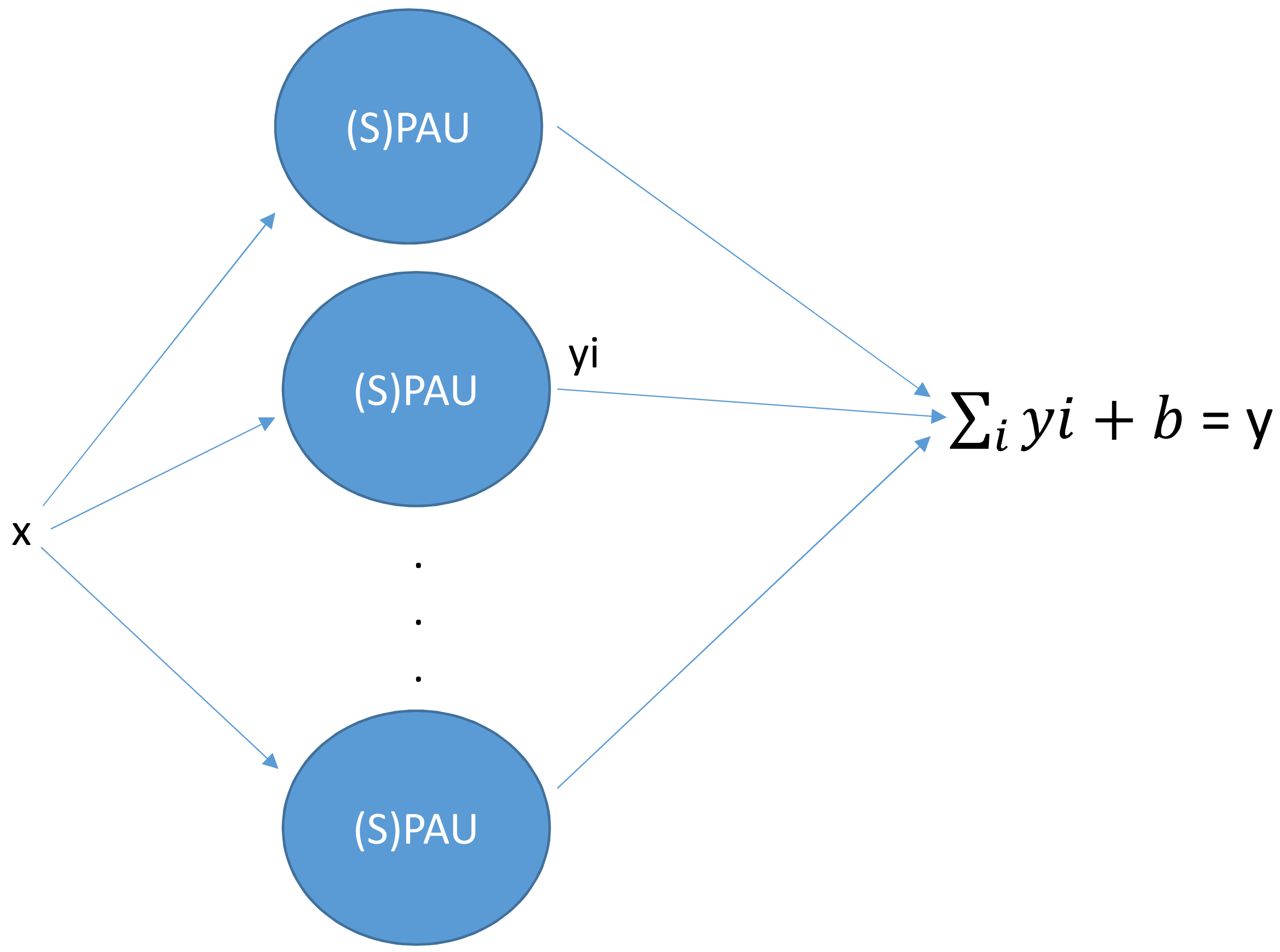
In 2020, Alejandro Molina has published a paper on PADE Activation Unit (PAU) as activation function for Deep Network. Instead of using a polynomial for the denominator, the absolute value of the polynomial, except the constant term, was used to avoid poles. In 2021, Chi-Chun Zhou published a paper were the activation function is based on PADE Approximants generalized to ratio net. These two papers show the property of universal approximation of Neural Networks with PAU nodes. In the following part, we will review this property on the matching of nonlinear functions.

The PAU nodes can take 2 forms: either a classical rational function (PAU below) or a “safe” rational function (SPAU below) design to avoid the poles such as the denominator is never zero.

$$\text{PAU}[m/n](x) = \frac{a0+a1\,x+a2\,x^2+\cdots+am\,x^m}{1+b1\,x+b2\,x^2+\ \ldots+bn\,x^n} \qquad (18)$$

$$\text{SPAU}[m/n](x) = \frac{a0+a1\,x+a2\,x^2+\cdots+am\,x^m}{1+\left|b1\,x+b2\,x^2+\ \ldots+bn\,x^n\right|} \qquad (19)$$

The Neural network is built up on PAU replacing the activation function at each node of the network. The figure below shows a 1 hidden layer neural network for 1 input layer and 1 output layer.



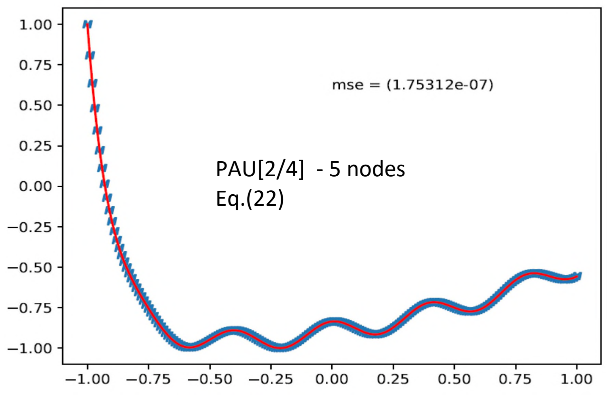
The model can be written for p PAU[M/N], or SPAU nodes as:

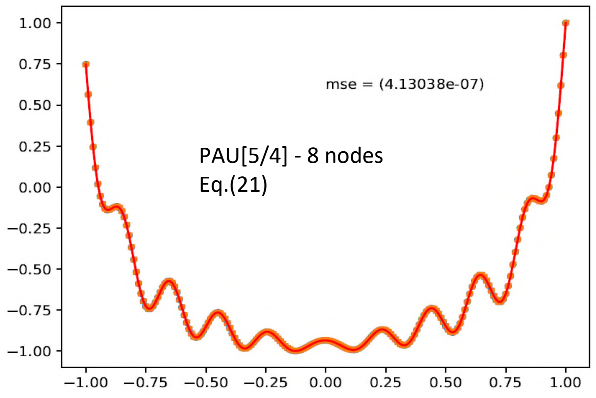
$$y(x) = \sum_{i=1}^{P} \left( \frac{a0 + a1\,x + a2\,x^2 + \cdots + am\,x^m}{1 + b1\,x + b2\,x^2 + \ldots + bn\,x^n} \right)_i + b \quad (20)$$

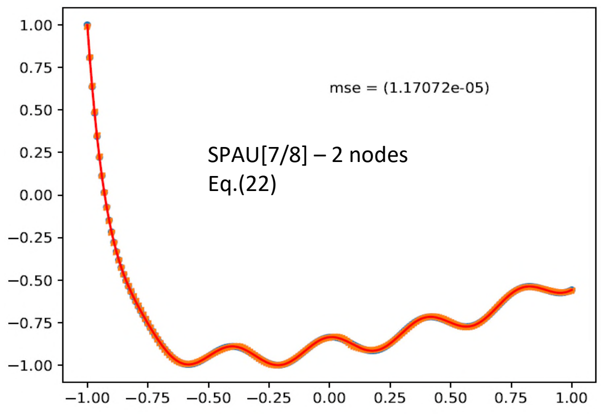
This network with one hidden layer was used to approximate the following nonlinear functions:

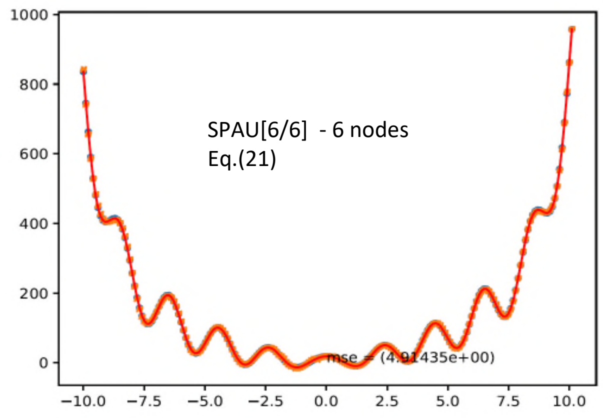
$$y(x) = 0.025\exp(-x) + 0.025\exp(x) + 2.8\,x^2 + 1.4\,x + 15.0\cos(\pi x) + 10.0\,x\sin(\pi x) + 3.5 \quad (21)$$

$$y(x) = 0.005\exp(-x) + 0.1\,x^2 + 1.2\,x + 3.5\cos(0.5\pi x) \quad (22)$$









The two top graphs show the matching of eq. 21 and 22 using PAU units, and the two bottom graphs show the same matching using SPAU units. While the matching is good, it is not unique, and the SPAU units show more sensitivity to the scaling process, that is why the SPAU of eq 21 is shown without scaling. The mse represents the mean squared error calculated on each points using the Scikit-Learn subroutine, the scaling of the data is also done using Scikit-Learn. The optimization process is done using

Scipy BFGS or L-BFGS-B subroutines using complex derivatives to estimate the jacobian. The Scipy code and the complex derivative will be detailed in some next papers.

In conclusion, the Pade Approximants are very interesting and powerful tools to be used as universal approximants being much superior to polynomial approximants. The usual activation functions can be replaced with PAU, which opens the research for new Artificial Neural network architectures.

## References:

- https://en.wikipedia.org/wiki/Pad%C3%A9_approximant
- https://mathworld.wolfram.com/PadeApproximant.html
- https://www.wolframalpha.com/input/?i=%5B5%2F5%5D+pade+of+sin%28x%29
- https://arxiv.org/abs/2005.06678v2
- https://arxiv.org/abs/2105.11309v1
- https://arxiv.org/abs/1907.06732