Tanh Summation Method

Jeffrey Severino
Dr. Ray Hixon
Toledo, OH 43606
email: jseveri@rockets.utoledo.edu

June 20, 2022

1 Introduction

KnUpp's Code Verification by the Method of Manufactured Solution (MMS) provides "guidelines" for creating a manufactured solution (MS) such that the observed order of accuracy will approach a theoretical order of accuracy as the number of grid points are reduced from one iteration to the next. While these guidelines offer a road map, there are choices that are left to the investigator that would benefit from additional examples. The first guideline gives the user a free choice of the MS as long as it s smooth. The benefit of the tanh summation method (TSM) reduces the difficuly in defining a sufficient MS by providing a general summation formulation that allows the user to Vary the number of terms in the MS, and the MS behavior without manually changing terms in the MS symbolic expression.

The general form of the MS will be a summation of *tanh* bounded between zero and one. A MS created with the TSM can provide a significant result for a numerical differencing/integration technique by having inflection points of each *tanh* at various locations along the domain, giving a stair like slope. While the TSM can add a layer of complexity to the MS that may not be needed, writing the formulation in a summation lends itself to iterative loops that can be coded, thus reducing the need for manual adjust of the MS, which can be an initial hurdle when performing MMS.

2 General form of a Hyperbolic Tangent

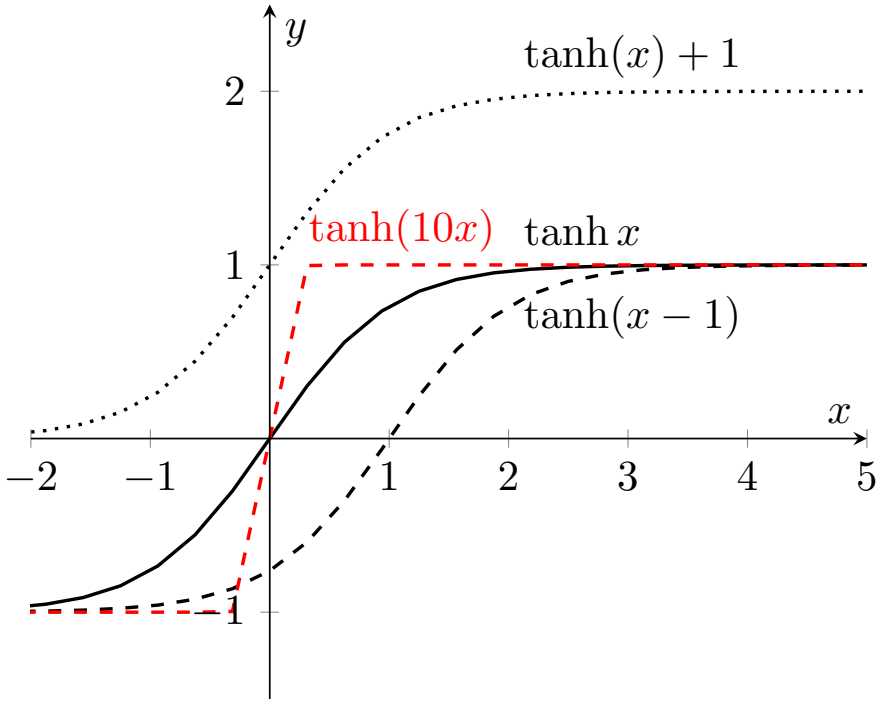
$$R = Atanh(B(x - C)) \tag{1}$$

$$L = Atanh(B(C - E)) \tag{2}$$

$$y = R + L + D \tag{3}$$

where

- $R \equiv$ The value of the hyperbolic tangent. The variable R represnts a "right" facing hyperbolic tangent kink.
- $A \equiv$ magnitude factor that increases or decreases the asymptotic limits $\lim_{x\to-\infty} = -1$ $\lim_{x\to\infty} = 1$
- $B \equiv$ "steepness" of the hyperbolic tangent
- $C \equiv$ The shift in inflection point of the hyperbolic tangent along the x axis
- $D \equiv$ The shift in inflection point of the hyperbolic tangent along the y axis
- $E \equiv x_{i=i_{max}}$
- x The domain. x_i is used to indicate grid point indicies.



The idea is to sum up an arbitrary amount of tangents that will be bounded by zero and one.

Now the goal is to generalize this formulation such that we can add up terms. A is determined by setting a maximum amplitude for each *tanh* function by $A = A_{max}/n$. Note that amplitude can be different for each term but is chosen to be the same. A parameter $\hat{x} = (x - x_{min})/(x_{max} - x_{min}$ scales the domain to be between the minimum and maximum bounds.

$$R_{ij} = Atanh(B(x_i - C_j)) \tag{4}$$

$$L_j = Atanh(B(C_j - E)) \tag{5}$$

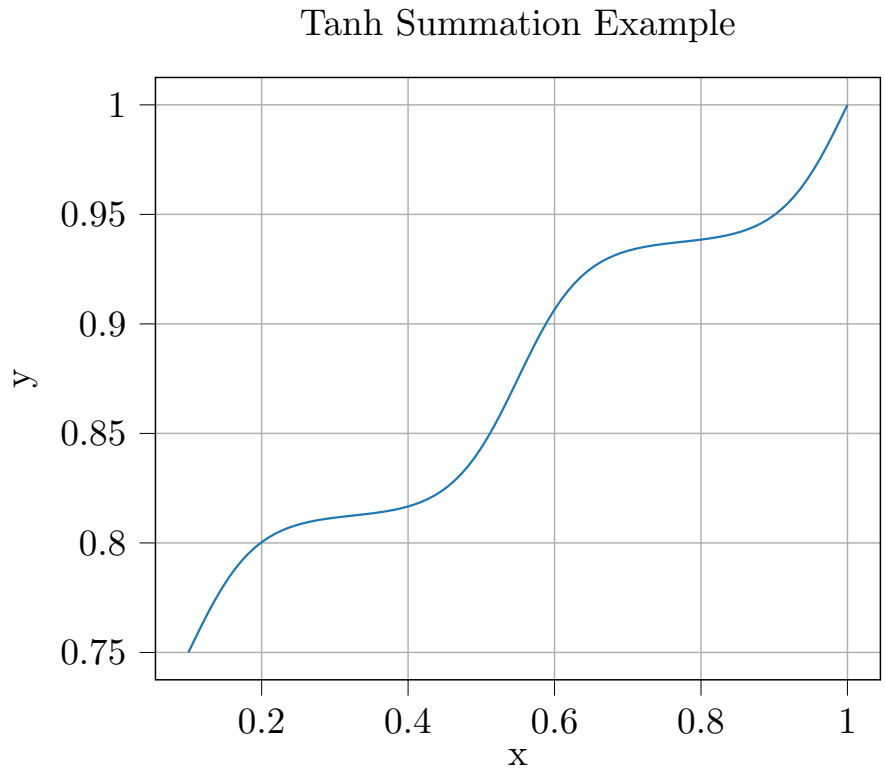
$$y = \sum_{j=1}^{n} R_{ij} + \sum_{j=1}^{n} L + D \tag{6}$$

The function `TanhMethod` does this procedure.

Setting $A = 1/16$ and $C_1 = 0$, $C_2 = 0.75$, $C_3 = 1$, $D = 1$, $E = 1$ and $B = 10$

$$\sum_{j=1}^{3} R_{ij} = 1/16tanh(10(\hat{x}_i)) + 1/16tanh(10(\hat{x}_i - 0.75)) + 1/16tanh(10(\hat{x}_i - 1)) \tag{7}$$

$$\sum_{j=1}^{3} L_j = 1/16tanh(10(-1)) + 1/16tanh(10(0.75-1)) + 1/16tanh(10(1-1)) \tag{8}$$



The simplified expression becomes,

$$y = \frac{1}{16}\tanh\left(\frac{100}{9}r - \frac{100}{9}\right) + \frac{1}{16}\tanh\left(\frac{100}{9}r - \frac{55}{9}\right) + \frac{1}{16}\tanh\left(\frac{100}{9}r - \frac{10}{9}\right) + \frac{7}{8} \tag{9}$$

3 Appendix

```
# ========= Packages ========
import sympy as sp
import numpy as np
def TanhMethod(n,B,r_min,r_max):
    # inputs:
    #   n - number of tanh functions
    #   B - the slope around the inflection point

    # outputs:
    # initialize lists for the tanh function
    RightKink  = []
    LeftKink   = []

    # symbolic variables needed for this function
    r = sp.Symbol('r')
```

```
# rescaling the radius (redundant but needed for BC Fairing Function)
r_hat = (r - r_min)/(r_max - r_min)

# amplitude for each wave
A          = []

# maximum allowed amplitude
max_amplitude = 0.25

# vertical shift along the y axis
# one is chosed to keep the inflection points above
# zero
S_vertical = 1

# rj is the list of inflection point locations
rj = list(np.linspace(1,0,n))



# messages for error and warning checking
warning_mssg = {1:'Warning: Total Amplitude exceeds maximum ', \
                2:'Warning: Function is negative'}

# getting amplitude for each kink
for i in range(len(rj)):
    #
    A.append(max_amplitude/(len(rj)+1))
    if sum(A) > max_amplitude:
        sys.exit(str(warning_mssg[1]))

# defining kinks and antikinks
for j in range(len(rj)):
    RightKink.append( A[j]*sp.tanh(B*( r_hat    - rj[j] )) )
    LeftKink.append(  A[j]*sp.tanh(B*( rj[j]- rj[0] )) )

f = sum(LeftKink) + sum(RightKink) + S_vertical
print(A)
return f
```