Numerical Anaylasis Proj 4

Prolem 1: Find the polynomial of degree 10 that interpolates the function arctan x at 11 equally spaced points in the interval [1,6]. Print the coefficients in the Newton form of the polynomial. Compute and print the difference between the polynomial and the function at 33 equally spaced points in the interval [0,8]. What conclusion can be drawn?

Prolem 1: Print the coefficients in the Newton form of the polynomial.

In [4]:

```
#import sympy
from sympy import *
import sympy as sym
import numpy as npy
import math
#using init_printing
sym.init_printing(use_latex="mathjax")
init_printing(use_unicode=True,)




# # Function to find the root
def arctan(x):
    return npy.arctan(x)




def populateDatapoints_matrix(f,x,y,a,b,n):
 t = (b-a)/n
 an=a
 for i in range(n):
    y[i,0]= f(an)
    x.append(an)
    an=an+t

def productTerm(i, value, x):
    product = 1
    for j in range(0,i):
        product = product * (value - x[j])
    return product

def dividedDiffTable_matrix(x, y, n):
    for i in range(1,n):
        for j in range(0,n-i):
            y[j,i]= (y[j,i - 1] - y[j + 1,i - 1]) / (x[j] - x[i + j])
```

```
def NewtonDividedDifference(value,x,y,n):
    sum = y[0,0]
    for i in range(1, n):
        sum = sum + (productTerm(i, value, x) * y[0,i])
    return sum

# construct table polynomial 11
a=1
b=6
x=[]
x_array=[]
n=11
ymatrix= sym.Matrix.zeros(n,n)
xmatrix= sym.Matrix.zeros(n)
populateDatapoints_matrix(arctan,x,ymatrix,a,b,n)
dividedDiffTable_matrix(x, ymatrix, n)
xmatrix=ymatrix.row(0)
print(xmatrix)
```

```
Matrix([[0.785398163397448, 0.402843797977465, -0.153274974082992, 0.0
458491398368293, -0.0110733536933921, 0.00216313869198317, -0.00032906
3209962730, 3.26938070460224e-5, 5.03927839154818e-7, -1.2198780505844
6e-6, 3.70644555184973e-7]])
```

Prolem 1: Find the polynomial of degree 10 that interpolates the function arctan x at 11 equally spaced points in the interval [1,6]

In [2]:

```
from sympy import *
init_printing(use_latex=True)


#import sympy
from sympy import *
import sympy as sym
import numpy as npy
import math
#using init_printing
sym.init_printing(use_latex="mathjax")
init_printing(use_unicode=True,)
x=symbols('x')

def populateDatapoints_matrix(f,w,y,a,b,n):
 t = (b-a)/n
#  print('t',t)
#  print('a',a)
```

```
#  print('b',b)
    an=a
    for i in range(n):
        y[i,0]= f(an)
        w.append(an)
        an=an+t

def arctan(w):
    return npy.arctan(w)

def productTerm_string(i, w):
    s=''
    for j in range(0,i):
        temp=w[j]
        s= s + '('+ str(x-temp)+')'
        if(i != i-1):
            s=s+"*"
    #print('s product term:',s)
    return s

def dividedDiffTable_matrix(w, y, n):
    for i in range(1,n):
        for j in range(0,n-i):
            y[j,i]= (y[j,i - 1] - y[j + 1,i - 1]) / (w[j] - w[i + j])

def NewtonDividedDifference_string(w,y,n):

    sum = y[0,0]
    s=''
    s=str(y[0,0])
    p=''
    for i in range(1, n):
        temp=y[0,i]
        p=productTerm_string(i,w)
        s = s +"+"+ p + '('+ str(temp)+')'

    return s


# #polynomial
a2=1
b2=6

n2=11
w=[]
ymatrix= sym.Matrix.zeros(n2,n2)
populateDatapoints_matrix(arctan,w,ymatrix,a2,b2,n2)
dividedDiffTable_matrix(w, ymatrix, n2)
# #string
x=symbols('x')
r= NewtonDividedDifference_string(w,ymatrix,n2)
expand(r)
```

```
# L1 = (x-2)*(x-3)/((0-2)*(0-3))
# L2 = (x-0)*(x-3)/((2-0)*(2-3))
# L3 = (x-0)*(x-2)/((3-0)*(3-2))
# p = 1*L1 + 3*L2 + 0*L3
# expand(p)
```

Out[2]:

$$3.70644555184973 \cdot 10^{-7} x^{10} - 1.2507689503945 \cdot 10^{-5} x^{9} + 0.000182979893808116 x^{8} - 0.0184140900756385 x^{5} - 0.000397892201175830 x^{4} + 0.18168736652767 x^{3} - 0.0936294988557575$$

$3.70644555184973 \cdot 10^{-7} x^{10} - 1.2507689503945 \cdot 10^{-5} x^{9} + 0.000182979893808116 x^{8} - 0.0014983173$

Prolem 1: Compute and print the difference between the polynomial and the function at 33 equally spaced points in the interval [0,8]

In [3]:

```
# # Function to find the root
def arctan(x):
    return npy.arctan(x)



def NewtonDividedDifference_string(w,y,n):

    sum = y[0,0]
    s=''
    s=str(y[0,0])
    p=''
    for i in range(1, n):
        temp=y[0,i]
        p=productTerm_string(i,w)
        s = s +"+"+ p + '('+ str(temp)+')'


    return s
def productTerm_string(i, w):
    s=''
    for j in range(0,i):
        temp=w[j]
        s= s + '('+ str(x-temp)+')'
        if(i != i-1):
            s=s+"*"
    #print('s product term:',s)
```

```
    return s
def populateDatapoints_matrix(f,x,y,a,b,n):
 t = (b-a)/n
 an=a
 for i in range(n):
    y[i,0]= f(an)
    x.append(an)
    an=an+t
def fillactual(f,a,b,n,r,v):
    t = (b-a)/n

    an=a
    for i in range(n):
#         print('an',an)
        r.append(f(an))
        v.append(an)
        an=an+t


def getdifs_expr(f,expr,a,b,n,diff):
    xvalues=[]
    realyvalues=[]
    fillactual(f,a,b,n,realyvalues,xvalues)
    for i in range(n):
        x=xvalues[i]
        aprox = eval(expr)
        d=aprox - realyvalues[i]
        diff.append(d)

def productTerm(i, value, x):
    product = 1
    for j in range(0,i):
        product = product * (value - x[j])
    return product


def dividedDiffTable_matrix(x, y, n):
    for i in range(1,n):
        for j in range(0,n-i):
            y[j,i]= (y[j,i - 1] - y[j + 1,i - 1]) / (x[j] - x[i + j])

def NewtonDividedDifference(value,x,y,n):
    sum = y[0,0]
    for i in range(1, n):
        sum = sum + (productTerm(i, value, x) * y[0,i])
    return sum
def main():
#     print('hello world')
    #assumption polynomial is degree the one constructed with 11 terms [0,6]
    #difference between the polynomial and the function at 33 equally spaced points

    #polynomial
    a2=1
    b2=6
```

```python
    n2=11
    xvalues=[]
    ymatrix= sym.Matrix.zeros(n2,n2)
    populateDatapoints_matrix(arctan,xvalues,ymatrix,a2,b2,n2)
    dividedDiffTable_matrix(xvalues, ymatrix, n2)
    expr = NewtonDividedDifference_string(xvalues,ymatrix,n2)
    # diffs populate 33 equally spaced points
    a=0
    b=8
    n=33

    diff=[]
    getdifs_expr(arctan,expr,a,b,n,diff)
    print("diffs:",diff)

    return 0

main()
```

```
diffs: [-0.093629498855757498, -0.029016435273853586, -0.0067801695146
522678, -0.0010229027955719339, -2.5396883236328271e-05, 2.55266510980
06781e-05, -6.6613381477509392e-16, -2.6693240513520067e-06, 2.4169797
874229459e-07, 5.5808495758036258e-07, -1.3803247878030334e-07, -1.925
1866367753223e-07, 9.620486629557945e-08, 1.0016641982524277e-07, -9.2
020687292304615e-08, -7.4547628070575911e-08, 1.2560967577179838e-07,
7.5609929428566147e-08, -2.5561252270378532e-07, -9.2090330472416326e-
08, 8.4102444475320226e-07, 2.2204460492503131e-16, -5.400587742121487
6e-06, 4.3794718580514314e-06, 0.00014443095242966386, 0.0007986653675
1249662, 0.0029723490323052548, 0.0089380159803891246, 0.0233339871953
71719, 0.054929600456944394, 0.11933704277219559, 0.24302732121823101,
0.46909835265987088]
```

Out[3]:

0

0

What conclusion can be drawn? Numerically we can see that the interpolation can be very accurate within the range that the interpolating polynomial is constructed and becomes increasingly un-accurate outside the bound in which it was constructed

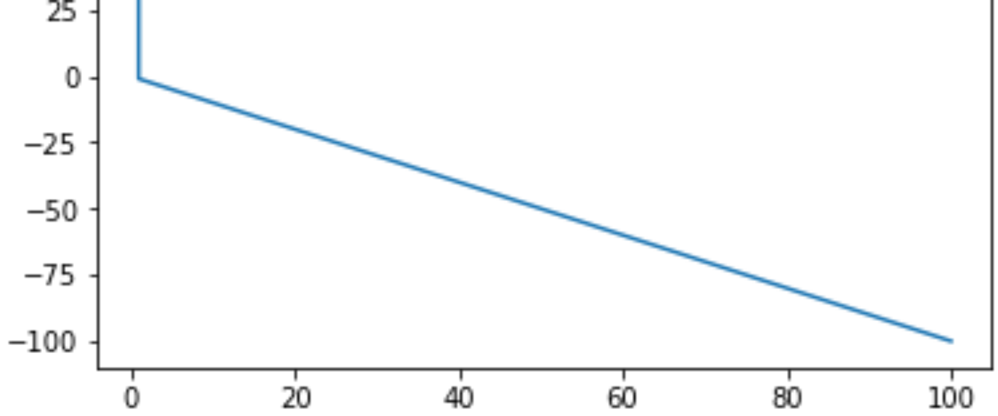
Problem 2 (3 pts) Let f (x) = max{0, 1, −x}. Sketch the function f . Then find interpolating polynomials p of degrees 2, 4, 8, 16, and 32 to f on the interval [-4,4], using equally spaced nodes. Calculate the discrepancy f(x) − p(x) at 64 equally spaced points for each p and plot it. Then redo the problem using Chebyshev nodes. Compare.

Problem 2: Let f (x) = max{0, 1, −x}. Sketch the function f

In [29]:

```
import matplotlib.pyplot as plt
from sympy import *
init_printing(use_latex=True)


#import sympy
from sympy import *
import sympy as sym
import numpy as npy
import math
#using init_printing
sym.init_printing(use_latex="mathjax")
init_printing(use_unicode=True,)

x=symbols('x')
def populateDatapoints_matrix(f,w,y,a,b,n):
 t = (b-a)/n
#  print('t',t)
#  print('a',a)
#  print('b',b)
 an=a
 for i in range(n):
    y[i,0]= f(an)
    w.append(an)
    an=an+t

def arctan(w):
    return npy.arctan(w)

def productTerm_string(i, w):
    s=''
    for j in range(0,i):
        temp=w[j]
        s= s + '('+ str(x-temp)+')'
        if(i != i-1):
            s=s+"*"
    #print('s product term:',s)
    return s

def dividedDiffTable_matrix(w, y, n):
    for i in range(1,n):
        for j in range(0,n-i):
            y[j,i]= (y[j,i - 1] - y[j + 1,i - 1]) / (w[j] - w[i + j])
def NewtonDividedDifference_string(w,y,n):

    sum = y[0,0]
    s=''
    s=str(y[0,0])
```

```
    p=''
    for i in range(1, n):
        temp=y[0,i]
        p=productTerm_string(i,w)
        s = s +"+"+ p + '('+ str(temp)+')'

    return s

def f(x):
    x=x*(-1)
    return max(0, 1, x)





k=[]
y=[]
for i in range (-100,100):
    k.append(f(i))
    y.append(i)

 #plot(p,(x,0,4),xlabel='x',ylabel='y',title='Intepolating polynomial for (0,-2), (
plt.plot(k, y)
plt.show()
```



find interpolating polynomials p of degrees 2, 4, 8, 16, and 32 to f on the interval [-4,4], using equally spaced nodes

In [107]:

```
import matplotlib.pyplot as plt
init_printing(use_latex=True)
import sympy as sym
import numpy as npy
import math
#using init_printing
sym.init_printing(use_latex="mathjax")
init_printing(use_unicode=True,)
x=symbols('x')

def populateDatapoints_matrix(f,w,y,a,b,n):
 t = (b-a)/n
 an=a
 for i in range(n):
    y[i,0]= f(an)
    w.append(an)
    an=an+t

def arctan(w):
    return npy.arctan(w)

def productTerm_string(i, w):
    s=''
    for j in range(0,i):
        temp=w[j]
        s= s + '('+ str(x-temp)+')'
        if(i != i-1):
            s=s+"*"
    #print('s product term:',s)
    return s

def dividedDiffTable_matrix(w, y, n):
    for i in range(1,n):
        for j in range(0,n-i):
            y[j,i]= (y[j,i - 1] - y[j + 1,i - 1]) / (w[j] - w[i + j])
def NewtonDividedDifference_string(w,y,n):
```

```
    sum = y[0,0]
    s=''
    s=str(y[0,0])
    p=''
    for i in range(1, n):
        temp=y[0,i]
        p=productTerm_string(i,w)
        s = s +"+"+ p + '('+ str(temp)+')'

    return s

def fw(w):
    w=w*(-1)
    return max(0, 1, w)


#Then find interpolating polynomials p of degrees 2, 4, 8, 16, and 32 to f on the i
# using equally spaced nodes.

#degrees 2
#---polynomial
a2=-4
b2=4
n2=2
w2=[]
ymatrix= sym.Matrix.zeros(n2,n2)
populateDatapoints_matrix(arctan,w2,ymatrix,a2,b2,n2)
dividedDiffTable_matrix(w2, ymatrix, n2)
#---printing  degrees 2

r2= NewtonDividedDifference_string(w2,ymatrix,n2)
expand(r2)
```

Out[107]:

$$0.331454415917008x + 1.99840144432528 \cdot 10^{-15}$$

$$0.331454415917008x + 1.99840144432528 \cdot 10^{-15}$$

In [103]:

```
import matplotlib.pyplot as plt
init_printing(use_latex=True)
import sympy as sym
import numpy as npy
import math
#using init_printing
sym.init_printing(use_latex="mathjax")
init_printing(use_unicode=True,)
```

```
x=symbols('x')

def populateDatapoints_matrix(f,w,y,a,b,n):
 t = (b-a)/n
#  print('t',t)
#  print('a',a)
#  print('b',b)
 an=a
 for i in range(n):
    y[i,0]= f(an)
    w.append(an)
    an=an+t

def arctan(w):
    return npy.arctan(w)

def productTerm_string(i, w):
    s=''
    for j in range(0,i):
        temp=w[j]
        s= s + '('+ str(x-temp)+')'
        if(i != i-1):
            s=s+"*"
    #print('s product term:',s)
    return s

def dividedDiffTable_matrix(w, y, n):
    for i in range(1,n):
        for j in range(0,n-i):
            y[j,i]= (y[j,i - 1] - y[j + 1,i - 1]) / (w[j] - w[i + j])
def NewtonDividedDifference_string(w,y,n):

    sum = y[0,0]
    s=''
    s=str(y[0,0])
    p=''
    for i in range(1, n):
        temp=y[0,i]
        p=productTerm_string(i,w)
        s = s +"+"+ p + '('+ str(temp)+')'

    return s

def fw(w):
    w=w*(-1)
    return max(0, 1, w)


#Then find interpolating polynomials p of degrees 2, 4, 8, 16, and 32 to f on the in
# using equally spaced nodes.


#degrees 4
a2=-4
```

```
b2=4
n2=4
w4=[]
ymatrix= sym.Matrix.zeros(n2,n2)
populateDatapoints_matrix(fw,w4,ymatrix,a2,b2,n2)
dividedDiffTable_matrix(w4, ymatrix, n2)
#---printing  degrees 4
r4= NewtonDividedDifference_string(w4,ymatrix,n2)
expand(r4)
```

Out[103]:

$0.125x^2 - 0.25x + 1.0$

$$0.125x^2 - 0.25x + 1.0$$

In [104]:

```
import matplotlib.pyplot as plt
init_printing(use_latex=True)
import sympy as sym
import numpy as npy
import math
#using init_printing
sym.init_printing(use_latex="mathjax")
init_printing(use_unicode=True,)
x=symbols('x')

def populateDatapoints_matrix(f,w,y,a,b,n):
 t = (b-a)/n
#  print('t',t)
#  print('a',a)
#  print('b',b)
 an=a
 for i in range(n):
    y[i,0]= f(an)
    w.append(an)
    an=an+t

def arctan(w):
    return npy.arctan(w)

def productTerm_string(i, w):
    s=''
    for j in range(0,i):
        temp=w[j]
        s= s + '('+ str(x-temp)+')'
        if(i != i-1):
            s=s+"*"
    #print('s product term:',s)
    return s
```

```python
def dividedDiffTable_matrix(w, y, n):
    for i in range(1,n):
        for j in range(0,n-i):
            y[j,i]= (y[j,i - 1] - y[j + 1,i - 1]) / (w[j] - w[i + j])
def NewtonDividedDifference_string(w,y,n):

    sum = y[0,0]
    s=''
    s=str(y[0,0])
    p=''
    for i in range(1, n):
        temp=y[0,i]
        p=productTerm_string(i,w)
        s = s +"+"+ p + '('+ str(temp)+')'

    return s

def fw(w):
    w=w*(-1)
    return max(0, 1, w)


#Then find interpolating polynomials p of degrees 2, 4, 8, 16, and 32 to f on the i
# using equally spaced nodes.

#degrees 16
a2=-4
b2=4
n2=16
w16=[]
ymatrix= sym.Matrix.zeros(n2,n2)
populateDatapoints_matrix(fw,w16,ymatrix,a2,b2,n2)
dividedDiffTable_matrix(w16, ymatrix, n2)
#---printing  degrees 16
r16= NewtonDividedDifference_string(w16,ymatrix,n2)
expand(r16)
```

Out[104]:

$$1.61249685059209 \cdot 10^{-5} x^{15} + 4.6514332228618 \cdot 10^{-5} x^{14} - 0.000618640618$$
$$+ 0.00937053070386402 x^{11} + 0.0197354497354495 x^{10} - 0.0718682287729915 x^{9} - 0.$$
$$+ 0.293641975308649 x^{6} - 0.616966490299811 x^{5} - 0.241865079365044 x^{4} + 0.467$$
$$- 0.0825896325895933 x + 1.000000000000$$

$1.61249685059209 \cdot 10^{-5} x^{15} + 4.6514332228618 \cdot 10^{-5} x^{14} - 0.000618640618640618 x^{13} - 0.001571268$

In [13]:

```python
import matplotlib.pyplot as plt
init_printing(use_latex=True)
```

```
import sympy as sym
import numpy as npy
import math
#using init_printing
sym.init_printing(use_latex="mathjax")
init_printing(use_unicode=True,)
x=symbols('x')

def populateDatapoints_matrix(f,w,y,a,b,n):
 t = (b-a)/n
 an=a
 for i in range(n):
    y[i,0]= f(an)
    w.append(an)
    an=an+t

def arctan(w):
    return npy.arctan(w)

def productTerm_string(i, w):
    s=''
    for j in range(0,i):
        temp=w[j]
        s= s + '('+ str(x-temp)+')'
        if(i != i-1):
            s=s+"*"
    return s

def dividedDiffTable_matrix(w, y, n):
    for i in range(1,n):
        for j in range(0,n-i):
            y[j,i]= (y[j,i - 1] - y[j + 1,i - 1]) / (w[j] - w[i + j])
def NewtonDividedDifference_string(w,y,n):

    sum = y[0,0]
    s=''
    s=str(y[0,0])
    p=''
    for i in range(1, n):
        temp=y[0,i]
        p=productTerm_string(i,w)
        s = s +"+"+ p + '('+ str(temp)+')'

    return s

def fw(w):
    w=w*(-1)
    return max(0, 1, w)



#Then find interpolating polynomials p of degrees 2, 4, 8, 16, and 32 to f on the i
# using equally spaced nodes.

#degrees 32
```

```
a2=-4
b2=4
n2=32
w32=[]
ymatrix= sym.Matrix.zeros(n2,n2)
populateDatapoints_matrix(fw,w32,ymatrix,a2,b2,n2)
dividedDiffTable_matrix(w32, ymatrix, n2)
#---printing  degrees 32
r32= NewtonDividedDifference_string(w32,ymatrix,n2)
expand(r32)
```

Out[13]:

$$4.85087465772575 \cdot 10^{-9}x^{31} + 1.42598987783145 \cdot 10^{-8}x^{30} - 3.912212489505$$
$$+ 1.40464334535301 \cdot 10^{-5}x^{27} + 3.66401543325998 \cdot 10^{-5}x^{26} - 0.0002967737$$
$$+ 0.00410931699787995x^{23} + 0.00901396325067296x^{22} - 0.0393033127915205x^{21} -$$
$$+ 0.452866228986612x^{18} - 1.29534019295516x^{17} - 1.85135278530192x^{16} + 4.$$
$$- 10.9959026817321x^{13} - 9.67916314376179x^{12} + 18.3688323680897x^{11} + 11$$
$$- 7.86432852663257x^{8} + 12.9152231861739x^{7} + 2.79778448621379x^{6} - 4.42$$
$$+ 0.65422122705294x^{3} + 0.0172139869253196x^{2} - 0.0264694521$$

$4.85087465772575 \cdot 10^{-9}x^{31} + 1.42598987783145 \cdot 10^{-8}x^{30} - 3.91221248950511 \cdot 10^{-7}x^{29} - 1.089227$

Calculate the discrepancy f(x) – p(x) at 64 equally spaced points for each p and plot it.

f(x) – p2(x)

In [29]:

```
import matplotlib.pyplot as plt

init_printing(use_latex=True)
import sympy as sym
import numpy as npy
import math
#using init_printing
sym.init_printing(use_latex="mathjax")
init_printing(use_unicode=True,)
x=symbols('x')
sym.init_printing(use_latex="mathjax")
init_printing(use_unicode=True,)

def populateDatapoints_matrix(f,w,y,a,b,n):
 t = (b-a)/n
 an=a
 for i in range(n):
```

```
    y[i,0]= f(an)
    w.append(an)
    an = an + t
#(arctan,w,ymatrix,a,b,ndegree_poly)
def populateDatapoints_matrix_Chebyshev(f,nodes,y,a,b,n):

 for j in range(n):
     nodes.append(npy.cos((2*(j+1)-1)/(2*n)*npy.pi))
 for i in range(n):
    y[i,0]= f(nodes[i])



def arctan(w):
    return npy.arctan(w)

def productTerm_string(i, w):
    s=''
    for j in range(0,i):
        temp=w[j]
        s= s + '('+ str(x-temp)+')'
        if(i != i-1):
            s=s+"*"
    #print('s product term:',s)
    return s

def dividedDiffTable_matrix(w, y, n):
    for i in range(1,n):
        for j in range(0,n-i):
            y[j,i]= (y[j,i - 1] - y[j + 1,i - 1]) / (w[j] - w[i + j])
def NewtonDividedDifference_string(w,y,n):

    sum = y[0,0]
    s=''
    s=str(y[0,0])
    p=''
    for i in range(1, n):
        temp=y[0,i]
        p=productTerm_string(i,w)
        s = s +"+"+ p + '('+ str(temp)+')'

    return s
# fillactual(fw,a2,b2,n2,realvalues,w2)
def fillactual(f,a,b,n,r,v):
    t = (b-a)/n

    an=a
    for i in range(n):
#         print('an',an)
        r.append(f(an))
        v.append(an)
        an=an+t
```

```
def getdifs_expr(f,expr,a,b,n,diff,xvalues):
    realyvalues=[]
    fillactual(f,a,b,n,realyvalues,xvalues)
    for i in range(n):
        x=xvalues[i]
        aprox = eval(expr)
        d=abs(aprox - realyvalues[i]) #should i use abs???????????????????????????
        diff.append(d)
def fw(w):
    w=w*(-1)
    return max(0, 1, w)

def NewtonDividedDifference(value,w,y,n):
    sum = y[0,0]
    for i in range(1, n):
        sum = sum + (productTerm(i, value, w) * y[0,i])
    return sum
def difplot(ndegree_poly,nPlaces):
    a=-4
    b=4
    n=2
    w=[]
    createxvalues=[]
    diff=[]
    print('difference Poly Degree',ndegree_poly,"Red is without Chebyshev nodes and
    ymatrix= sym.Matrix.zeros(ndegree_poly,ndegree_poly)
    populateDatapoints_matrix(arctan,w,ymatrix,a,b,ndegree_poly)
    dividedDiffTable_matrix(w, ymatrix, ndegree_poly)
    expr= NewtonDividedDifference_string(w,ymatrix,n)
    getdifs_expr(fw,expr,a,b,nPlaces,diff,createxvalues)
    plt.plot(createxvalues,diff,color='red')           #plot1
    #plt.show()

    title=str(ndegree_poly)+' diff without Chebyshev nodes'
    plt.savefig(title)
    plt.close()
    w=[]
    createxvalues=[]
    diff=[]
    ymatrix= sym.Matrix.zeros(ndegree_poly,ndegree_poly)
    populateDatapoints_matrix_Chebyshev(arctan,w,ymatrix,a,b,ndegree_poly)##che
    dividedDiffTable_matrix(w, ymatrix, ndegree_poly)
    expr= NewtonDividedDifference_string(w,ymatrix,n)
    getdifs_expr(fw,expr,a,b,nPlaces,diff,createxvalues)

    plt.plot(createxvalues,diff,color='green')
#     plt.show()
    title=str(ndegree_poly)+' diff with Chebyshev nodes'
    plt.savefig(title)
    plt.close()

ndegree_poly2=2
```

```
ndegree_poly4=4
ndegree_poly8=8
ndegree_poly16=16
ndegree_poly32=32
nPlaces=64
difplot(ndegree_poly2,nPlaces)
difplot(ndegree_poly4,nPlaces)
difplot(ndegree_poly8,nPlaces)
difplot(ndegree_poly16,nPlaces)
difplot(ndegree_poly32,nPlaces)
```

```
difference Poly Degree 2 Red is without Chebyshev nodes and green is w
ith Chebyshev nodes
difference Poly Degree 4 Red is without Chebyshev nodes and green is w
ith Chebyshev nodes
difference Poly Degree 8 Red is without Chebyshev nodes and green is w
ith Chebyshev nodes
difference Poly Degree 16 Red is without Chebyshev nodes and green is
with Chebyshev nodes
difference Poly Degree 32 Red is without Chebyshev nodes and green is
with Chebyshev nodes
```

Type *Markdown* and LaTeX: α^2 α^2

In [ ]:

```

```

Numerical Anaylasis Proj 4

Prolem 1: Find the polynomial of degree 10 that interpolates the function arctan x at 11 equally spaced points in the interval [1,6]. Print the coefficients in the Newton form of the polynomial. Compute and print the difference between the polynomial and the function at 33 equally spaced points in the interval [0,8]. What conclusion can be drawn?

Prolem 1: Print the coefficients in the Newton form of the polynomial.

In [4]:

```
Matrix([[0.785398163397448, 0.402843797977465, -0.153274974082992, 0.0
458491398368293, -0.0110733536933921, 0.00216313869198317, -0.00032906
3209962730, 3.26938070460224e-5, 5.03927839154818e-7, -1.2198780505844
6e-6, 3.70644555184973e-7]])
```

Type *Markdown* and LaTeX: α^2

Prolem 1: Find the polynomial of degree 10 that interpolates the function arctan x at 11 equally spaced points in the interval [1,6]

In [2]:

Out[2]:

$$3.70644555184973 \cdot 10^{-7}x^{10} - 1.2507689503945 \cdot 10^{-5}x^{9} + 0.000182979893808116x^{8} - 0.00149831739$$

Prolem 1: Compute and print the difference between the polynomial and the function at 33 equally spaced points in the interval [0,8]

In [3]:

```
diffs: [-0.093629498855757498, -0.029016435273853586, -0.0067801695146
522678, -0.0010229027955719339, -2.5396883236328271e-05, 2.55266510980
06781e-05, -6.6613381477509392e-16, -2.6693240513520067e-06, 2.4169797
874229459e-07, 5.5808495758036258e-07, -1.3803247878030334e-07, -1.925
1866367753223e-07, 9.620486629557945e-08, 1.0016641982524277e-07, -9.2
020687292304615e-08, -7.4547628070575911e-08, 1.2560967577179838e-07,
7.5609929428566147e-08, -2.5561252270378532e-07, -9.2090330472416326e-
08, 8.4102444475320226e-07, 2.2204460492503131e-16, -5.400587742121487
6e-06, 4.3794718580514314e-06, 0.00014443095242966386, 0.0007986653675
1249662, 0.0029723490323052548, 0.0089380159803891246, 0.0233339871953
71719, 0.054929600456944394, 0.11933704277219559, 0.24302732121823101,
0.46909835265987088]
```

Out[3]:

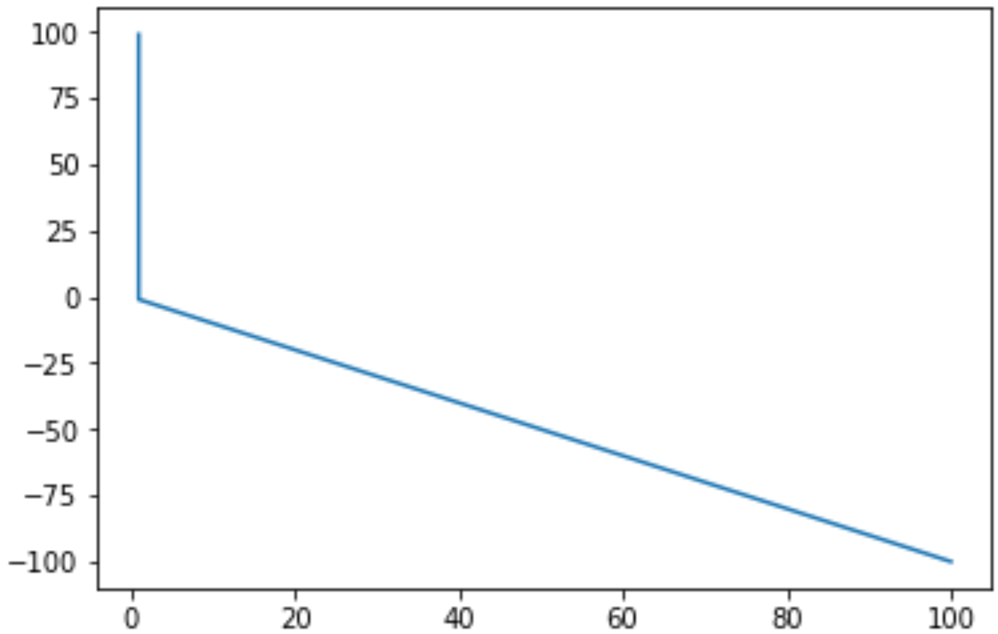
$$0$$

What conclusion can be drawn? Numerically we can see that the interpolation can be very accurate within the range that the interpolating polynomial is constructed and becomes increasingly un-accurate outside the bound in which it was constructed

Problem 2 (3 pts) Let f (x) = max{0, 1, −x}. Sketch the function f . Then find interpolating polynomials p of degrees 2, 4, 8, 16, and 32 to f on the interval [-4,4], using equally spaced nodes. Calculate the discrepancy f(x) − p(x) at 64 equally spaced points for each p and plot it. Then redo the problem using Chebyshev nodes. Compare.

Problem 2: Let f (x) = max{0, 1, −x}. Sketch the function f

In [29]:



find interpolating polynomials p of degrees 2, 4, 8, 16, and 32 to f on the interval [-4,4], using equally spaced nodes

In [107]:

Out[107]:

$$0.331454415917008x + 1.99840144432528 \cdot 10^{-15}$$

In [103]:

Out[103]:

$$0.125x^2 - 0.25x + 1.0$$

In [104]:

Out[104]:

$$1.61249685059209 \cdot 10^{-5}x^{15} + 4.6514332228618 \cdot 10^{-5}x^{14} - 0.000618640618640618x^{13} - 0.001571268$$

In [13]:

Out[13]:

$$4.85087465772575 \cdot 10^{-9}x^{31} + 1.42598987783145 \cdot 10^{-8}x^{30} - 3.91221248950511 \cdot 10^{-7}x^{29} - 1.08922708971857 \cdot 10^{-6}x$$

Calculate the discrepancy f(x) – p(x) at 64 equally spaced points for each p and plot it.

f(x) – p2(x)

In [29]:

```
difference Poly Degree 2 Red is without Chebyshev nodes and green is w
ith Chebyshev nodes
difference Poly Degree 4 Red is without Chebyshev nodes and green is w
ith Chebyshev nodes
difference Poly Degree 8 Red is without Chebyshev nodes and green is w
ith Chebyshev nodes
difference Poly Degree 16 Red is without Chebyshev nodes and green is
with Chebyshev nodes
difference Poly Degree 32 Red is without Chebyshev nodes and green is
with Chebyshev nodes
```

Type *Markdown* and LaTeX: α^2

In [ ]: