CMN1

October 20, 2020

1 Resolución de la ecuación diferencial: $-d^2f(x)/dx^2 = 1 + 4x^2$ con f(0)=f(1)=0 y x[0,1] mediante el método de los momentos con diferentes funciones base y peso

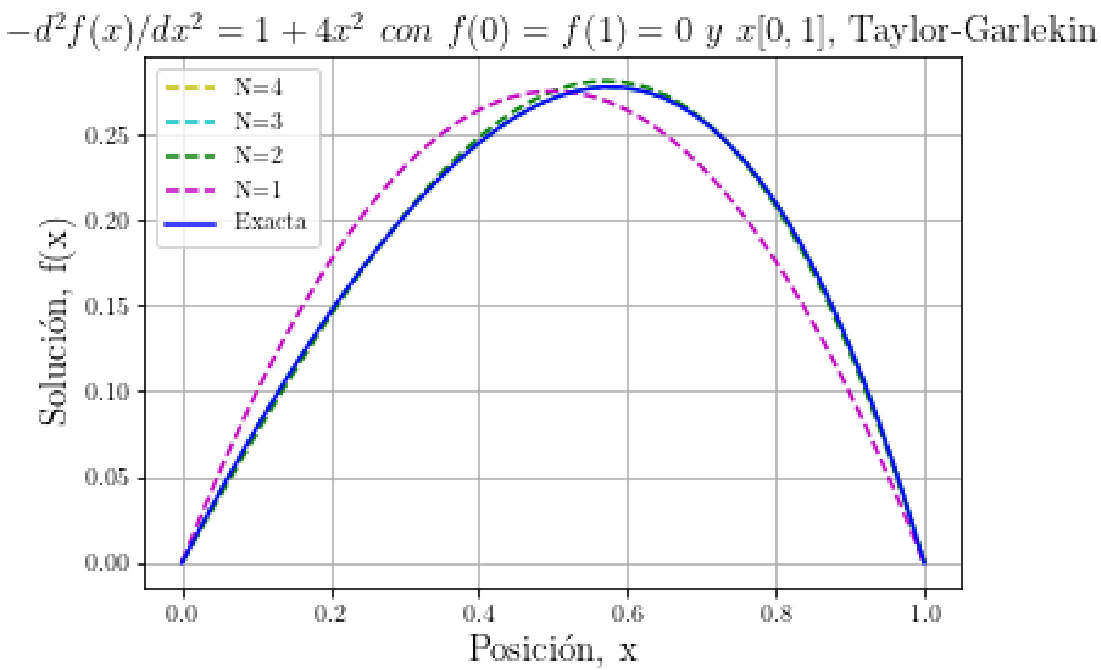
1.0.1 David Freire Fernández

[86]:
```
import numpy as np
import sympy as sp
import matplotlib.pyplot as plt
N=3
x=sp.Symbol('x')
ge=np.ones([N])
ele=np.ones([N,N])

for k in range (0,N):
    for p in range (0,N):
        a=k+1
        b=p+1
        ele[k,p]=(a*b)/(a+b+1)
        ge[p]=((3*b+8)*b)/(2*(b+2)*(b+4))
alpha=np.linalg.solve(ele, ge)
F=0
for i in range (0,N):
    c=i+1
    F=F+alpha[i]*(x-x**(c+1))
F
```

[86]: $-0.333333333333308x^4 - 5.22451129977445 \cdot 10^{-14}x^3 - 0.499999999999968x^2 + 0.833333333333328x$

[92]:
```
f5=np.ones([30])
f3=f5=np.ones([30])

teo=np.ones([30])
def Ff6(x):
    Ff6=-0.333333332950675*x**4-1.32511631667498*10-10*x**3-0.
 ↪49999999997909*x**2+0.833333333332264*x
    return Ff6
def Ff5(x):
```

```python
    Ff=7.73552901703218*10**-12*x**6-2.34612277519766*10-11*x**5-0.333333333306913*x**4-1.34348186896381*10**-11*x**3-0.499999999997058*x**2+0.833333333333132*x
    return Ff
def Ff4(x):
    Ff4=-3.87551969682549*10**-13*x**5-0.333333333332341*x**4-8.77849644827697*10**-13*x**3-0.499999999999694*x**2+0.8333333333333*x
    return Ff4
def Ff3(x):
    Ff3=-0.333333333333308*x**4-5.22451129977445*(10**-14)*x**3-0.499999999999968*x**2+0.833333333333328*x
    return Ff3
def Ff2(x):
    Ff2=-0.666666666666668*x**3-0.0999999999999987*x**2+0.766666666666666*x
    return Ff2
def Ff1(x):
    Ff1=-1.1*x**2+1.1*x
    return Ff1
def teo(x):
    teo=(5/6)*x-(1/2)*x**2-(1/3)*x**4
    return teo

x=np.linspace(0,1,50)
f3=Ff3(x)
f4=Ff4(x)
f2=Ff2(x)
f6=Ff6(x)
f5=Ff5(x)
f1=Ff1(x)
t=teo(x)
#plt.plot(x,f6,'r--', label="N=6")
#plt.plot(x,f5,'r--', label="N=5")
plt.plot(x,f4,'y--', label="N=4")
plt.plot(x,f3,'c--', label="N=3")
plt.plot(x,f2,'g--', label="N=2")
plt.plot(x,f1,'m--', label="N=1")
plt.plot(x,t,'b',label="Exacta")
plt.rc('text', usetex=True)
plt.rc('font', family='serif')
plt.grid('on')
plt.title('$-{d{{}^2}f(x)}/{dx{{}^2}}=1+4x{{}^2} \\ con\\ f(0)=f(1)=0\\ y \\ x[0,1]$, Taylor-Garlekin',fontsize=15)
plt.xlabel('Posición, x',fontsize=15)
plt.ylabel('Solución, f(x)',fontsize=15)
plt.legend(loc="best")
plt.show()
```

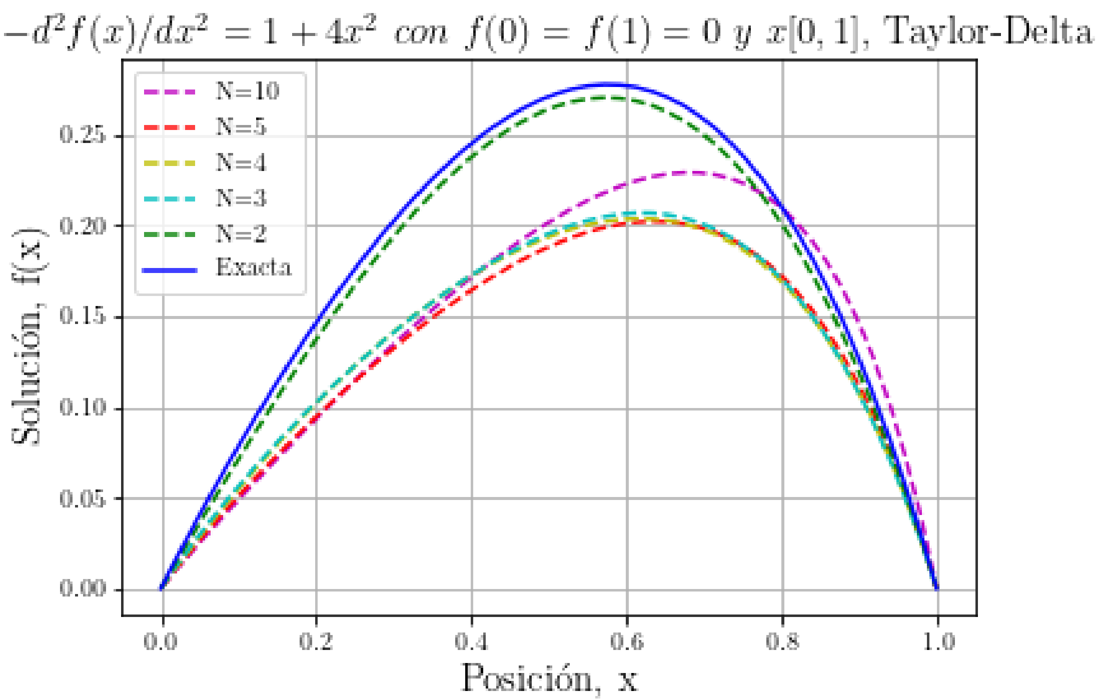


[87]:
```python
import numpy as np
import sympy as sp
import matplotlib.pyplot as plt
N=3
x=sp.Symbol('x')
ge=np.ones([N])
ele=np.ones([N,N])

for k in range (0,N):
    for p in range (0,N):
        a=k+1
        b=p+1
        ele[k,p]=(a*(a+1))*(b/(N+1))**(a-1)
        ge[p]=1+4*(b/(N+1))**2
alpha=np.linalg.solve(ele, ge)
F=0
for i in range (0,N):
    c=i+1
    F=F+alpha[i]*(x-x**(c+1))
F
```

[87]: $-0.791666666666667x^4 + 0.875x^3 - 0.708333333333333x^2 + 0.625x$

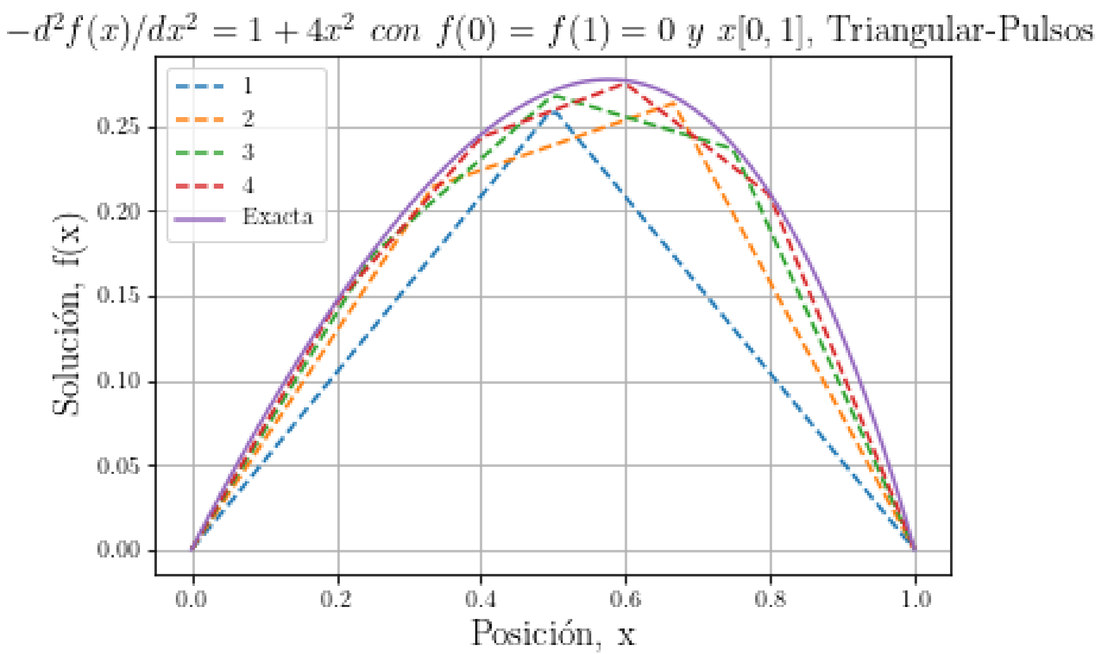
[91]:
```python
def Ff10(x):
```

```
    Ff10=-0.306826846048574*x**11+1.20693295346655*x**10-2.81807179374623*x**9+3.81481781619486*x**8-2.96454781349953*x**7+0.391520933474278*x**6+1.4367339023369*x**5-1.74952421507793*x**4+0.829847052367201*x**3-0.357410915087355*x**2+0.516528925619835*x
    return Ff10
def Ff5(x):
    Ff5=-0.596296296296309*x**6+1.29259259259265*x**5-1.68888888888898*x**4+0.996296296296365*x**3-0.559259259259278*x**2+0.555555555555556*x
    return Ff5
def Ff4(x):
    Ff4=-0.550555555555555*x**5+0.579999999999998*x**4-0.294999999999997*x**3-0.314444444444446*x**2+0.58*x
    return Ff4
def Ff3(x):
    Ff3=-0.791666666666667*x**4+0.875*x**3-0.708333333333333*x**2+0.625*x
    return Ff3
def Ff2(x):
    Ff2=-0.666666666666667*x**3-0.0555555555555556*x**2+0.722222222222222*x
    return Ff2
def teo(x):
    teo=(5/6)*x-(1/2)*x**2-(1/3)*x**4
    return teo

x=np.linspace(0,1,50)
f3=Ff3(x)
f4=Ff4(x)
f5=Ff5(x)
f10=Ff10(x)
f2=Ff2(x)
t=teo(x)
plt.plot(x,f10,'m--', label="N=10")
plt.plot(x,f5,'r--', label="N=5")
plt.plot(x,f4,'y--', label="N=4")
plt.plot(x,f3,'c--', label="N=3")
plt.plot(x,f2,'g--', label="N=2")
plt.plot(x,t,'b',label="Exacta")
plt.rc('text', usetex=True)
plt.rc('font', family='serif')
plt.grid('on')
plt.title('$-{d{{}^2}f(x)}/{dx{{}^2}}=1+4x{{}^2} \\ con\\ f(0)=f(1)=0\\ y \\ x[0,1]$, Taylor-Delta',fontsize=15)
plt.xlabel('Posición, x',fontsize=15)
plt.ylabel('Solución, f(x)',fontsize=15)
plt.legend(loc="best")
plt.show()
```



[90]:
```
import scipy.interpolate as spi
import numpy as np
import matplotlib.pyplot as plt

for N in range (1,5):

    ge=np.ones([N])
    F=np.zeros([N+2])
    alpha2=np.zeros([N+2])
    ele=np.ones([N,N])
    for k in range (0,N):
        for p in range (0,N):
            a=k+1
            b=p+1
            ele[k,p]=0

            if a==b:
                ele[k,p]=2*(N+1)
            if abs(a-b)==1:
                ele[k,p]=-(N+1)
            ge[p]=(1+(4*b**2+1/3)/(N+1)**2)/(N+1)

    alpha=np.linalg.solve(ele, ge)
    for i in range (0,N+2):
```

```
        F[i]=i/(N+1)

    for j in range (1,N+1):
        alpha2[j]=alpha[j-1]

    H=spi.interp1d(F,alpha2)
    x = np.linspace(0,1,100)
    plt.plot(x,H(x),'--', label=N)

def teo(x):
    teo=(5/6)*x-(1/2)*x**2-(1/3)*x**4
    return teo
t=teo(x)
plt.plot(x,t,label='Exacta')
plt.rc('text', usetex=True)
plt.rc('font', family='serif')
plt.grid('on')
plt.title('$-{d{{}^2}f(x)}/{dx{{}^2}}=1+4x{{}^2} \\ con\\ f(0)=f(1)=0\\ y \\␣
 ↪x[0,1]$, Triangular-Pulsos',fontsize=15)
plt.xlabel('Posición, x',fontsize=15)
plt.ylabel('Solución, f(x)',fontsize=15)
plt.legend(loc="best")
plt.show()
```




```
[ ]:
```