interpol_Newton

April 1, 2025

[1]:
```
reset()
```

[2]:
```
%display typeset
```

1 Interpolação e aproximação de funções

1.1 Diferenças divididas

[27]:
```
def dif_divid(xi, fxi, alg=6):
    """
    Calcula a tabela das diferencas divididas

    Input:
    xi  -> nos de interpolacao
    fxi -> valores nodais

    Output:
    -> matriz das diferencas divididas
    -> diagonal superior da tabela

    Versao:
    """
    n = len(xi)
    q = zero_matrix(RDF,n)
    for i in range(n):
        q[i,i] = fxi[i]
    for j in range(1,n):
        for iter in range(0,n-j):
            a = iter
            b = iter+j
            q[a,b] = (q[a,b-1]-q[a+1,b])/(xi[a]-xi[b])

    #for j in range(0,n):
    #    for i in range(j,-1,-1):
    #        print (N(q[i,j],digits=alg)),

    return N(q,digits=alg),N(q.row(0),digits=alg)
```

[30]:
```
xi = vector(RDF, [0.2, 0.6, 1.0, 1.4, 1.8]); xi
```

[30]:

(0.2, 0.6, 1.0, 1.4, 1.8)

[31]:
```
fxi = vector(RDF, [1.02, 1.19, 1.54, 2.15, 3.11]); fxi
```

[31]:

(1.02, 1.19, 1.54, 2.15, 3.11)

[32]:
```
dif_divid(xi, fxi)
```

[32]:

$$\left(\begin{pmatrix} 1.02000 & 0.425000 & 0.562500 & 0.208333 & 0.0162760 \\ 0.000000 & 1.19000 & 0.875000 & 0.812500 & 0.234375 \\ 0.000000 & 0.000000 & 1.54000 & 1.52500 & 1.09375 \\ 0.000000 & 0.000000 & 0.000000 & 2.15000 & 2.40000 \\ 0.000000 & 0.000000 & 0.000000 & 0.000000 & 3.11000 \end{pmatrix}, (1.02000, 0.425000, 0.562500, 0.208333, 0.0162760)\right)$$

[4]:
```
def diferencas_divid(xi,fxi,alg=6):
    """
    Versao mais simples da tabela das diferencas
    divididas com os elementos na diagonal

    Versao:
    """
    n = len(xi)
    d = matrix(RDF,n)
    for i in range(n):
        d[i,0] = fxi[i]
    for j in [1..n-1]:
        for i in [j..n-1]:
            d[i,j] = (d[i-1,j-1]-d[i,j-1]) / (xi[i-j]-xi[i])
    return N(d,digits=alg)
```

[33]:
```
diferencas_divid(xi, fxi)
```

[33]:

$$\begin{pmatrix} 1.02000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 \\ 1.19000 & 0.425000 & 0.000000 & 0.000000 & 0.000000 \\ 1.54000 & 0.875000 & 0.562500 & 0.000000 & 0.000000 \\ 2.15000 & 1.52500 & 0.812500 & 0.208333 & 0.000000 \\ 3.11000 & 2.40000 & 1.09375 & 0.234375 & 0.0162760 \end{pmatrix}$$

1.1.1 Forma de Newton das diferenças divididas do polinómio interpolador

[5]:
```
def poli_Newton(xi,fxi,alg=6):
    """
    Forma de Newton das diferencas divididas do
    polinómio interpolador

    Input:
    xi  -> nos de interpolacao
    fxi -> valores nodais
```

```
    alg -> quantidade de algarismos na saida

    Output:
    soma -> polinómio interpolador das diferencas divididas

    Versão:
    """
    a, b = dif_divid(xi, fxi, alg)
    x = var('x')
    n = len(xi)
    prod = 1
    soma = b[0]
    #print(soma)
    for i in range(n-1):
        prod = prod*(x-xi[i])
        soma = soma +b [i+1]*prod
    #print(soma)
    return soma.expand()
```

[34]:
```
poli_Newton(xi, fxi)
```

[34]:

$$0.0162760\,x^{4} + 0.156250\,x^{3} + 0.243490\,x^{2} + 0.143750\,x + 0.980235$$

[6]:
```
xi = vector(RDF, [0.2, 0.5, 1.0, 1.2, 2.3]); xi
```

[6]:

$$(0.2, 0.5, 1.0, 1.2, 2.3)$$

[7]:
```
fxi = vector(RDF, [1.020067, 1.127626, 1.543081, 1.810656, 5.037221]); fxi
```

[7]:

$$(1.020067, 1.127626, 1.543081, 1.810656, 5.037221)$$

[8]:
```
dif_divid(xi, fxi)
```

[8]:

$$\left(\left(\begin{array}{rrrrr} 1.02007 & 0.358530 & 0.590475 & 0.133761 & 0.0693650 \\ 0.000000 & 1.12763 & 0.830910 & 0.724236 & 0.279427 \\ 0.000000 & 0.000000 & 1.54308 & 1.33788 & 1.22720 \\ 0.000000 & 0.000000 & 0.000000 & 1.81066 & 2.93324 \\ 0.000000 & 0.000000 & 0.000000 & 0.000000 & 5.03722 \end{array}\right), (1.02007, 0.358530, 0.590475, 0.133761, 0.0693650)\right)$$

[9]:
```
diferencas_divid(xi,fxi)
```

[9]:

$$\left(\begin{array}{rrrrr} 1.02007 & 0.000000 & 0.000000 & 0.000000 & 0.000000 \\ 1.12763 & 0.358530 & 0.000000 & 0.000000 & 0.000000 \\ 1.54308 & 0.830910 & 0.590475 & 0.000000 & 0.000000 \\ 1.81066 & 1.33788 & 0.724236 & 0.133761 & 0.000000 \\ 5.03722 & 2.93324 & 1.22720 & 0.279427 & 0.0693650 \end{array}\right)$$

[10]:
```
poli_Newton(xi, fxi)
```

[10]:

$$0.0693650\,x^{4} - 0.0673977\,x^{3} + 0.560078\,x^{2} - 0.0213208\,x + 1.00236$$

1.1.2 Exemplo 3.13

[11]:
```
xi = vector(RDF, [0.1, 0.5, 1.5, 2.2, 2.5]); xi
```

[11]:

(0.1, 0.5, 1.5, 2.2, 2.5)

[12]:
```
fxi = vector(RDF, [0.099669, 0.463648, 0.982794, 1.133169, 1.190290])
```

[13]:
```
dif_divid(xi,fxi)
```

[13]:

$$\left(\begin{pmatrix} 0.0996690 & 0.909948 & -0.279144 & 0.0476807 & 0.0123406 \\ 0.000000 & 0.463648 & 0.519146 & -0.179014 & 0.0772982 \\ 0.000000 & 0.000000 & 0.982794 & 0.214821 & -0.0244181 \\ 0.000000 & 0.000000 & 0.000000 & 1.13317 & 0.190403 \\ 0.000000 & 0.000000 & 0.000000 & 0.000000 & 1.19029 \end{pmatrix}, (0.0996690, 0.909948, -0.279144, 0.0476807,\right.$$

[14]:
```
poli_Newton(xi, fxi)
```

[14]:

$0.0123406\,x^4 - 0.00538395\,x^3 - 0.310536\,x^2 + 1.09601\,x - 0.00682280$

1.1.3 Outro exemplo

[15]:
```
xi = vector(RDF, [0.2, 0.6, 1.0, 1.4, 1.8]); xi
```

[15]:

(0.2, 0.6, 1.0, 1.4, 1.8)

[16]:
```
fxi = vector(RDF, [1.02, 1.19, 1.54, 2.15, 3.11]); fxi
```

[16]:

(1.02, 1.19, 1.54, 2.15, 3.11)

[17]:
```
poli_Newton(xi, fxi)
```

[17]:

$0.0162760\,x^4 + 0.156250\,x^3 + 0.243490\,x^2 + 0.143750\,x + 0.980235$

[18]:
```
P4(x) = poli_Newton(xi, fxi)
```

[19]:
```
P4(1.1)
```

[19]:

1.66478

1.2 Diferenças progressivas

[20]:
```
def tabela_dif(xi, fxi, tabela=True, alg=6):
    """
    Calcula a tabela das diferencas

    Input:
    xi  -> nos de interpolacao
    fxi -> valores nodais
    alg -> quantidade de algarismos na saida
```

```
    Output:
    q -> matriz das diferencas

    Versão: 2025
    """
    n = len(xi)
    h = xi[1]-xi[0]
    q = zero_matrix(RDF,n)

    for i in range(n):
        q[i,i] = fxi[i]
    for j in range(1,n):
        for iter in range(0,n-j):
            a = iter
            b = iter+j
            q[a,b] = (q[a+1,b]-q[a,b-1])
    if tabela:
        for j in range(0,n):
            for i in range(j,-1,-1):
                print (N(q[i,j],digits=alg)),
            print()
    return N(q,digits=alg)
```

1.2.1 Polinómio interpolador das diferenças progressivas de Newton

[21]:
```
def poliNewton_progres(xi, fxi, alg=6):
    """
    Determina o polinomio interpolador na forma
    das diferenças progressivas de Newton
    usando a tabela das diferencas.

    Input:
    xi  -> nos de interpolação
    fxi -> valores nodais
    alg -> quantidade de algarismos no output

    Output:
    soma -> polinomio interpolador das diferenças progressivas

    Versão:
    """
    progr = tabela_dif(xi,fxi,alg)
    dif = progr.row(0)

    x = var('x')
    n = len(xi)
    prod = 1
```

```python
    soma = dif[0]

    h = xi[1]-xi[0]
    s = (x-xi[0])/h

    for i in range(n-1):
        prod = prod*(s-i)

        soma += prod*dif[i+1]/factorial(i+1)

    return soma.expand()
```

[22]:
```python
poliNewton_progres(xi, fxi)
```

```
1.02000

1.19000
0.170000

1.54000
0.350000
0.180000

2.15000
0.610000
0.260000
0.0800000

3.11000
0.960000
0.350000
0.0900000
0.0100000
```

[22]:

$$0.0162760\,x^4 + 0.156250\,x^3 + 0.243490\,x^2 + 0.143750\,x + 0.980234$$

[23]:
```python
P4(x) = poliNewton_progres(xi, fxi, False)
```

[24]:
```python
P4(1.1)
```

[24]:

1.66478

[25]:
```python
def plot_poli_interp_Newton_progress(xi,fxi,a,b):
    '''
    Representação gráfica do polinómio
    interpolador

    '''
```

```
    nodes=[]
    x = var('x')
    for i in range(len(xi)):
        nodes.append((xi[i],fxi[i]))
    f(x)=poliNewton_progres(xi, fxi)
    P = plot(f(x),a,b)
    Q = line(nodes, marker='o', linestyle="", axes_labels=['$x$','$y=f(x)$'])
    (P+Q).show()
    return f
```

[26]:

```
plot_poli_interp_Newton_progress(xi,fxi,0,2)
```

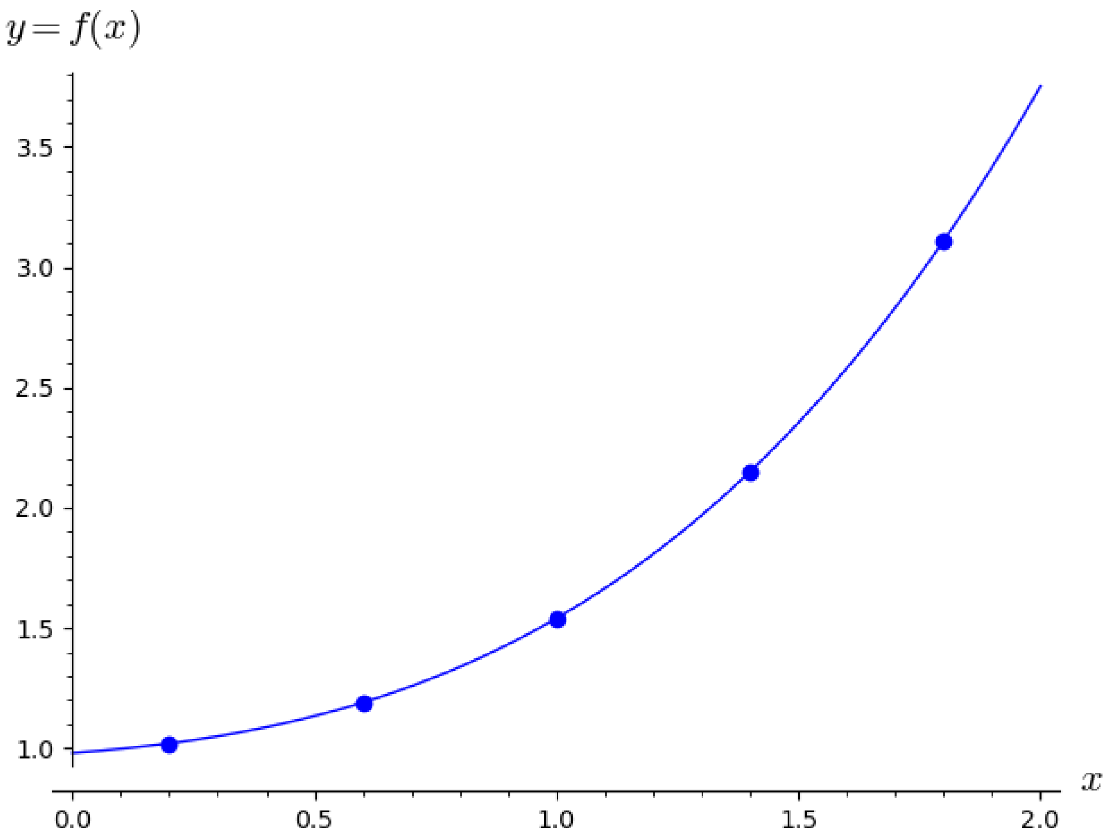
```
1.02000

1.19000
0.170000

1.54000
0.350000
0.180000

2.15000
0.610000
0.260000
0.0800000

3.11000
0.960000
0.350000
0.0900000
0.0100000
```



[26]:

$$x \mapsto 0.0162760\,x^4 + 0.156250\,x^3 + 0.243490\,x^2 + 0.143750\,x + 0.980234$$

[ ]: