4.5 DERIVATIVES USING NEWTON'S BACKWARD DIFFERENCE INTERPOLATION FORMULA

Differentiating Newton's backward difference formula (3.3) w.r.t.p., we have,

First-order

$$f'_p = \frac{1}{h}\frac{df_p}{dp}$$

$$= \frac{1}{h}\left\{\nabla f_0 + \frac{1}{2}(2p+1)\nabla^2 f_0 + \frac{1}{6}(3p^2+6p+2)\nabla^3 f_0 + \frac{1}{12}(2p^3+9p^2+11p+3)\nabla^4 f_0 + \cdots\right\} \quad \ldots (4.11)$$

Second-order

$$f''_p = \frac{1}{h}\frac{df'_p}{dp}$$

$$= \frac{1}{h^2}\left\{\nabla^2 f_0 + (p+1)\nabla^3 f_0 + \frac{1}{12}(6p^2+18p+11)\nabla^4 f_0 + \cdots\right\} \quad \ldots (4.12)$$

Similarly, other higher-order derivatives can be obtained.

Special Cases Formulas for derivatives when

(i) $p = 0$

$$f'_0 = \frac{1}{h}\left\{\nabla f_0 + \frac{1}{2}\nabla^2 f_0 + \frac{1}{3}\nabla^3 f_0 + \frac{1}{4}\nabla^4 f_0 + \cdots\right\} \quad \ldots (4.13)$$

$$f''_0 = \frac{1}{h^2}\left\{\nabla^2 f_0 + \nabla^3 f_0 + \frac{11}{12}\nabla^4 f_0 + \cdots\right\} \quad \ldots (4.14)$$

(ii) $p = \frac{1}{2}$

$$f'_{\frac{1}{2}} = \frac{1}{h}\left\{\nabla f_0 + \nabla^2 f_0 + \frac{23}{24}\Delta^3 f_0 + \frac{11}{24}\nabla^4 f_0 + \cdots\right\} \quad \ldots (4.15)$$

$$f''_{\frac{1}{2}} = \frac{1}{h^2}\left\{\nabla^2 f_0 + \frac{3}{2}\Delta^3 f_0 + \frac{43}{24}\nabla^4 f_0 + \cdots\right\} \quad \ldots (4.16)$$

Example 2 (a) The deflection f(x) measured at various distances x from one end of a cantilever is given in the following table:

x	0.0	0.2	0.4	0.6	0.8	1.0
f(x)	0.0000	0.0456	0.1278	0.3494	0.4027	0.4825

Evaluate f '(0.85) and f ''(1.0) based on Newton's backward difference interpolation formula.

(b) Write a computer program to implement the method for computing the first two derivatives.

Solution (a) The difference table is as follows:

x	f(x)	∇	∇^2	∇^3	∇^4
0.0	0.0000				
		455			
0.2	0.0456		368		
		823		1025	
0.4	0.1278		1393		−4101
		2216		−3076	
0.6	0.3494		−1683		5024
		533		1948	
$x_0 = 0.8$	0.4027		265		
		798			
1.0	0.4825				

(i) $x_0 = 0.8;\ x_p = 0.85,\ h = 0.2.$

$$p = \frac{(x_p - x_0)}{h} = \frac{(0.85 - .08)}{0.2} = 0.25$$

Substituting values of p and the required differences in (4.11), we get,

$$f'(.85) = \frac{1}{0.2}\left\{.0533 + \frac{1}{2}(0.25\times 2+1)\times(-.01683)\right.$$
$$+\frac{1}{6}(3\times.25^2+6\times.25+2)\times(-.3076)$$
$$\left.+\frac{1}{12}(2\times.25^3+9\times.25^2+11\times.25+3)\times(-.4101)\right\}$$
$$= \frac{1}{5}\{0.0533-0.1262-0.1891-0.2170\}$$
$$= -2.3950$$

(ii) $x_0 = 1;\ p = 0.$

Substituting values of p and the required differences in (4.14), we get,

$$f_0'' = \frac{1}{0.2^2}\left\{0.0265 + 0.1948 + \frac{11}{12} \times 0.5025\right\}$$

$$= \frac{1}{.04} \times 0.6819 = 17.0457$$

(b) Computer Program No. 6: Numerical Differentiation

```
# include<iostream.h>
# include<conio.h>

class NewBackDiff
  {
    public:
            NewBackDiff( );
            void input( );
            void result( );
            void get_1st_der( );
            void get_2nd_der( );

    private:
            int degree,values,actual_ degree;
            float nebla[10],xp,p,x[10],fx[10],h;
  };

    NewBackDiff::NewBackDiff( )
    {
        clscr( );
        cout<<"\n\tNEWTON'S BACKWARD DIFFERENCE FORMULA\n\n";
        degree=values=xp=actual_degree=0;
        p= -2;
        for(int i=0; i<10; i++)
              nebla[i]=x[i]=fx[i]=0.0;
    }

    void NewBackDiff::input( )
    {
          cout<<"How many values you want for X?\t";
          cin>>values;
          cout<<"Upto what power of Delta:\t";
```

```
        cin>>degree;
//      degree=degree>4 || degree<1 ? 4 : degree;
        cout<<"\n value of Xp:\t";
        cin>>xp;
        for(int i=0;i!=values;i++)
        {
            cout<<"\nEnter X"<<i+1<<":\t";
            cin>>x[i];
            cout<<"Enter F("<<i+1<<"):\t";
            cin>>fx[i];
        }
    }

void NewBackDiff::result( )
{
        diser( );
        cout<<"\nX\t";
        for(int i=0;i!=values;i++)
              cout<<"\t"<<x[i];

        cout<<endl;
        cout<<"\nF(x)\t";
        for(i=0;i!=values;i++)
              cout<<"\t"<<fx[i];

        cout<<endl;
        h=x[1]-x[0];
        for(int j=values-1,temp= -1;j>=0&&(p<0 || p>1); temp=j,j--)
              p=(xp-x[j])/h;

        cout<<"\nValue of P is :\t"<<p<<"\n";
        for(actual_degree=1, j=values; actual_degree<=degree&&j>1,actual_
            degree++, j--)
        {
            cout<<"\n\nNebla power "<<actual_degree<<":\t";
            for(int k=0;k<j-1;k++)
            {
                fx[k]=fx[k+1]-fx[k];
                cout<<fx[k]<<"\t";
            }
            nebla[actual_degree-1]=fx[temp-actual_degree];
        }
        get_1st_der( );
```

```
        get_2nd_der( );
}

void NewBackDiff::get_1st_der( );
{
        float parray[]={1,2*p+1,3*p*p+6*p+2,2*p*p*p+9*p*p+11p+3},
                        div[]={1,2,6,12},
                        ans=0;

        for(int i=0;i<actual_degree;i++)
                ans+=nebla[i]*parray[i]/div[i];
        cout<<"\n\n\n\nf '("<<xp<<"):\t"<<ans/h;
}

    void NewBackDiff::get_2nd_der( );
    {
        float parray[]={1,p+1,6*p*p+18*p+11},
                        div[]={1,1,12},
                        ans=0;

        for(int i=1;i<actual_degree;i++)
                ans+=nebla[i]*parray[i-1]/div[i-1];

        cout<<"\n\n\n\nf ''("<<xp<<"):\t"<<ans/(h*h);
    }

void main (void)
{
    NewBackDiff obj;
    obj.input( );
    obj.result( );
    getch( );
}
```

Computer Output

FROM NEWTON'S BACKWARD DIFFERENCE FORMULA

How many values you want for X? 6
Upto what power of Nebla: 4

Value of Xp: .85

```
Enter X1:       0
Enter F(1):     0

Enter X2:       .2
Enter F(2):     .0455

Enter X3:       .4
Enter F(3):     .1278

Enter X4:       .6
Enter F(4):     .3494

Enter X5:       .8
Enter F(5):     .4027

Enter X6:       1
Enter F(6):     .4825
```

X	0	0.2	0.4	0.6	0.8	1
F(X)	0	0.0455	0.1278	0.3494	0.4027	0.4825

```
Value of P is : 0.25

Nebla Power 1:      0.0455      0.0823      0.2216      0.0533      0.0798

Nebla Power 2:      0.0368      0.1393      -0.1683     0.0265

Nebla Power 3:      0.1025      -0.3076     0.1948

Nebla Power 4:      -0.4101     0.5024

f'(0.85):           -2.393843

f"(0.85):           -27.383205
```

4.6 DERIVATIVES USING CENTRAL DIFFERENCE INTERPOLATION FORMULAS

The formulas derived in Sections 4.4 and 4.5 are not very accurate. A relatively higher accuracy can be achieved if we use one of the central difference formulas. Consequently, the numerical differentiation is more accurate if the derivatives of an interpolation formula at the centre of the data-points are used.