Journal of Chemical, Biological and Physical Sciences

An International Peer Review E-3 Journal of Sciences

Available online at www.jcbsc.org

Section C: Physical Sciences

CODEN (USA): JCBPAT Research Article

Estimation of Errors in Newton's Divided Difference Formula for Polynomial Interpolation of Functions $x^{1/2}$, $x^{1/3}$ and $x^{1/4}$ in the interval [1, 2]

Dilli Raj Neupane

Department of mathematics, Patan Multiple Campus, Tribhuvan University, Kathmandu, Nepal

Received: 01 January 2016**; Revised:** 16 January 2016**; Accepted:** 20 January 2016

Abstract: The interval [1, 2] has been divided into 10 equal parts and the Newton's divided difference formula has been obtained. With the help of this formula, the values of the functions $x^{1/2}$, $x^{1/3}$ and $x^{1/4}$ have been calculated at the points 1.01, 1.02, 1.03, …., 2.00. Percentage errors in the values calculated using Newton's divided difference formula at each point have been calculated. Average percentage error in the interpolation of each function has been found out which shows that $x^{1/3}$ is better interpolated as compared to $x^{1/3}$ and $x^{1/4}$.

Keywords: Divided difference, interpolation, polynomial, approximation theory, difference triangle.

INTRODUCTION

There are many situations that call for fitting a common function to a collection of data. Determining a function that agrees precisely with the data, at least to within round-off tolerances, is called interpolation [1-2]. A polynomial approximation to a collection of data is easy to determine in various ways, and polynomials have easily computed derivatives and integrals that might be useful if the derivative or integral [3-5] of the function underlying the data is needed.

However, a polynomial of degree n-1 is generally required to satisfy a set of n conditions, and polynomials of even moderately high degree are likely to have a high degree of variation except where explicitly constrained. This means that in order to obtain stable approximating polynomials either the number of specified conditions must be kept small, which may not be a reasonable. The problem of approximating [6] a known function with a simpler function follows a pattern similar to that of fitting a function to a collection of data. There are many reasons why such an approximation might be required.

There are three basic types of error: systematic, gross and random. Systematic errors always either overestimate or underestimate the results of measurements and arise for specific reasons (incorrect set-up of measuring equipment, the effect of environment, etc.), which systematically affect the measurements and alter them in one direction. The estimation of systematic errors is achieved using methods which go beyond the confines of mathematical statistics. Gross errors (often called outliers) arise from miscalculations, incorrect reading of the measuring equipment, etc.

The results of measurements which contain gross errors differ greatly from other results of measurements and are therefore often easy to identify. Random errors [7-9] arise from various reasons which have an unforeseen effect on each of the measurements, both in overestimating and in underestimating results.

The main contents of approximation theory concern the approximation of functions. Its foundations are laid by the work of P. L. Chebyshev[10] (1854–1859) on best uniform approximation of functions by polynomials and by K. Weierstrass[11,] who in 1885 established that in principle it is possible to approximate a continuous function on a finite interval by polynomials with arbitrary pre-given error.

Data interpolation [12-15] is one of the most important tasks in geophysical data processing. Its importance is increasing with the development of 3-D seismics, since most of the modern 3-D acquisition geometries carry non-uniform spatial distribution of seismic records. Without a careful interpolation, acquisition irregularities may lead to unwanted artifacts at the imaging step [16, 17].

The interpolation problem in geophysics implies interpolating irregularly sampled data to a regular grid. In general, this problem requires a regularized inversion scheme, such as the method of inversion to zero offset [18, 19].

Newton's forward formula [12] is appropriate to apply only when the unknown value lies near the beginning of the table while Newton's backward formula is usually applied when the unknown value to be interpolated lies near the end of the table [20]. Lagrange's formula for interpolation is applicable for any type of observation [4] but when the observations are given at equi-spaced values of arguments, the formulas using various order differences are found to be more convenient and easy to apply rather than Lagrange's interpolation formula. In the case, the values of the arguments are unequally spaced, we should use Newton's divided difference formula to represent the function[21-29].

MATERIAL AND METHOD

Newton's divided difference formula: The Lagrange interpolation formula involves very considerable computation and its use can be quite risky. It is much more efficient to use the divided differences method for interpolation [30-32].

The divided differences for a function f[x] are defined as follows:

$$f[x_{i-1}, x_i] = \frac{f[x_i] - f[x_{i-1}]}{x_i - x_{i-1}}$$

$$f[x_{i-2}, x_{i-1}, x_i] = \frac{f[x_{i-1}, x_i] - f[x_{i-2}, x_{i-1}]}{x_i - x_{i-2}}$$

$$f[x_{i-3}, x_{i-2}, x_{i-1}, x_i] = \frac{f[x_{i-2}, x_{i-1}, x_i] - f[x_{i-3}, x_{i-2}, x_{i-1}]}{x_i - x_{i-3}}$$

$$f[x_{i-j}, x_{i-j+1}, \ldots, x_i] = \frac{f[x_{i-j+1}, \ldots, x_i] - f[x_{i-j}, \ldots, x_{i-1}]}{x_i - x_{i-j}}$$

The coefficient a_i of the Newton polynomial $P_n(x)$ is $a_i = f[x_0, x_1, \ldots, x_n]$ and it is the top element in the column of the i-th divided differences. The Newton polynomial of degree $\leq n$ that passes through the n+1 points $(x_k, y_k) = (x_k, f(x_k))$, for k = 1, 2, …, n is

$$P_n(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + a_3(x - x_0)(x - x_1)(x - x_2) + \ldots + a_n(x - x_0)(x - x_1)(x - x_2) \ldots (x - x_{n-1})$$

The interval [a, b] has been divided into 10 equal parts with the help of the points

$a=x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}=b$

to calculate $y_i = f(x_i)$ where i=0,1, 2, …. , 10. Difference triangle has been drawn by using the formula

$d^n y_i = d^{n-1} y_i - d^{n-1} y_{i-1}$ where n, i = 0,1, 2, …. , 10

Now, the Newton's divided difference formula

$f(x) = a_0 + a_1(x-x_0) + a_2(x-x_0)(x-x_1) + \ldots. + a_{10}(x-x_0)(x-x_1)\ldots.(x-x_9)$

Where $a_n=(d^n y_0)/(n!h^n)$; n = 0,1, 2, …. , 10

reduces to

$f(x) = c_0 + c_1 x + c_2 x^2 + \ldots.. + c_{10} x^{10}$

Where

c0=a0-a1*p(1,1)+a2*p(2,2)-a3*p(3,3)+a4*p(4,4)-a5*p(5,5)+a6*p(6,6)-a7*p(7,7)

+a8*p(8,8) -a9*p(9,9)+a10*p(10,10);

c1=a1-a2*p(2,1)+a3*p(3,2)-a4*p(4,3)+a5*p(5,4)-a6*p(6,5)+a7*p(7,6)-a8*p(8,7)

+a9*p(9,8) -a10*p(10,9);

c2=a2-a3*p(3,1)+a4*p(4,2)-a5*p(5,3)+a6*p(6,4)-a7*p(7,5)+a8*p(8,6)-a9*p(9,7)

+a10*p(10,8);

c3=a3-a4*p(4,1)+a5*p(5,2)-a6*p(6,3)+a7*p(7,4)-a8*p(8,5)+a9*p(9,6)-a10*p(10,7);

c4=a4-a5*p(5,1)+a6*p(6,2)-a7*p(7,3)+a8*p(8,4)-a9*p(9,5)+a10*p(10,6);

c5=a5-a6*p(6,1)+a7*p(7,2)-a8*p(8,3)+a9*p(9,4)-a10*p(10,5);

c6=a6-a7*p(7,1)+a8*p(8,2)-a9*p(9,3)+a10*p(10,4);

c7=a7-a8*p(8,1)+a9*p(9,2)-a10*p(10,3);

c8=a8-a9*p(9,1)+a10*p(10,2);

c9=a9-a10*p(10,1);

c10=a10;

p(i,j)=summation of the product of j elements in all combinations among $x_0,x_1,..,x_{i-1}$

Computer program in C++ for obtaining the polynomial interpolation of the functions $x^{1/2}$, $x^{1/3}$ and $x^{1/4}$ using Newton's divided difference formula in the interval [1, 2] has been developed and given in Program-1.

Program-1: Program in C++ for polynomial interpolation using Newton's divided difference formula

```
#include<conio.h>
#include<stdio.h>
#include<math.h>
//f(x)= square root of x
float x[11];
void main(void)
{FILE *fpt;
float y[11],dy[10],d2y[9],d3y[8],d4y[7],d5y[6],d6y[5];
```

```
float d7y[4],d8y[3],d9y[2],d10y[1];
float aa[11],c[11];
float a,b; //interval
float h,ax,fx,xx;
float p(int c,int d);
float factorial(int i);
int i;
clrscr();
fpt=fopen("draj1.txt", "w");
printf("Lower bound of interval, a= ");
scanf("%f", &a);
printf("Upper bound of interval, b= ");
scanf("%f", &b);
h=(b-a)/10.0;
fprintf(fpt,"a= %5.2f\n",a);
fprintf(fpt,"b= %5.2f\n",b);
fprintf(fpt,"h= %5.2f\n",h);
fprintf(fpt,"f(x)=x^.5\n");
for(i=0; i<11; i++) {x[i]=a+i*h;
y[i]=sqrt((double) x[i]);
printf("x[i]=%5.2f\n",x[i]);}
// divided difference
for(i=0; i<10; i++)
dy[i]=(y[i+1]-y[i]);
for(i=0; i<9; i++)
d2y[i]=(dy[i+1]-dy[i]);
```

```
for(i=0; i<8; i++)
d3y[i]=(d2y[i+1]-d2y[i]);
for(i=0; i<7; i++)
d4y[i]=(d3y[i+1]-d3y[i]);
for(i=0; i<6; i++)
d5y[i]=(d4y[i+1]-d4y[i]);
for(i=0; i<5; i++)
d6y[i]=(d5y[i+1]-d5y[i]);
for(i=0; i<4; i++)
d7y[i]=(d6y[i+1]-d6y[i]);
for(i=0; i<3; i++)
d8y[i]=(d7y[i+1]-d7y[i]);
for(i=0; i<2; i++)
d9y[i]=(d8y[i+1]-d8y[i]);
for(i=0; i<1; i++)
d10y[i]=(d9y[i+1]-d9y[i]);
aa[0]=y[0];
aa[1]=dy[0]/h;
aa[2]=d2y[0]/(2*h*h);
aa[3]=d3y[0]/(6*h*h*h);
aa[4]=d4y[0]/(24*pow( (double) h, (double) 4.0));
aa[5]=d5y[0]/(factorial(5)*pow( (double) h, (double) 5.0));
aa[6]=d6y[0]/(factorial(6)*pow( (double) h, (double) 6.0));
aa[7]=d7y[0]/(factorial(7)*pow( (double) h, (double) 7.0));
aa[8]=d8y[0]/(factorial(8)*pow( (double) h, (double) 8.0));
aa[9]=d9y[0]/(factorial(9)*pow( (double) h, (double) 9.0));
```

```
aa[10]=d10y[0]/(factorial(10)*pow( (double) h, (double) 10.0));
c[0]=aa[0]-aa[1]*p(1,1)+aa[2]*p(2,2)-aa[3]*p(3,3)+aa[4]*p(4,4)
-aa[5]*p(5,5)+aa[6]*p(6,6)-aa[7]*p(7,7)+aa[8]*p(8,8)-aa[9]
*p(9,9)+aa[10]*p(10,10);
c[1]=aa[1]-aa[2]*p(2,1)+aa[3]*p(3,2)-aa[4]*p(4,3)+aa[5]*p(5,4)-aa[6]
*p(6,5)+aa[7]*p(7,6)-aa[8]*p(8,7)+aa[9]*p(9,8)-aa[10]*p(10,9);
c[2]=aa[2]-aa[3]*p(3,1)+aa[4]*p(4,2)-aa[5]*p(5,3)+aa[6]*p(6,4)-aa[7]
*p(7,5)+aa[8]*p(8,6)-aa[9]*p(9,7)+aa[10]*p(10,8);
c[3]=aa[3]-aa[4]*p(4,1)+aa[5]*p(5,2)-aa[6]*p(6,3)+aa[7]*p(7,4)-aa[8]
*p(8,5)+aa[9]*p(9,6)-aa[10]*p(10,7);
c[4]=aa[4]-aa[5]*p(5,1)+aa[6]*p(6,2)-aa[7]*p(7,3)+aa[8]*p(8,4)-aa[9]
*p(9,5)+aa[10]*p(10,6);
c[5]=aa[5]-aa[6]*p(6,1)+aa[7]*p(7,2)-aa[8]*p(8,3)+aa[9]*p(9,4)
-aa[10]*p(10,5);
c[6]=aa[6]-aa[7]*p(7,1)+aa[8]*p(8,2)-aa[9]*p(9,3)+aa[10]*p(10,4);
c[7]=aa[7]-aa[8]*p(8,1)+aa[9]*p(9,2)-aa[10]*p(10,3);
c[8]=aa[8]-aa[9]*p(9,1)+aa[10]*p(10,2);
c[9]=aa[9]-aa[10]*p(10,1);
c[10]=aa[10];
fprintf(fpt,"difference triangle\n");
fprintf(fpt,"%11.8f %11.8f\n",x[0],y[0]);
fprintf(fpt," %11.8f\n",dy[0]);
fprintf(fpt,"%11.8f %11.8f %11.8f\n",x[1],y[1],d2y[0]);
fprintf(fpt,"%11.8f %11.8f\n",
dy[1],d3y[0]);
fprintf(fpt,"%11.8f  %11.8f%11.8f %11.8f\n",
```

```
x[2],y[2],d2y[1],d4y[0]);
fprintf(fpt," %11.8f %11.8f %11.8f\n",
dy[2],d3y[1],d5y[0]);
fprintf(fpt,"%11.8f %11.8f %11.8f %11.8f %11.8f\n",
x[3],y[3],d2y[2],d4y[1],d6y[0]);
fprintf(fpt," %11.8f %11.8f %11.8f %11.8f\n",
dy[3],d3y[2],d5y[1],d7y[0]);
fprintf(fpt,"%11.8f %11.8f %11.8f  %11.8f %11.8f %11.8f\n",
x[4],y[4],d2y[3],d4y[2],d6y[1],d8y[0]);
fprintf(fpt," %11.8f %11.8f %11.8f  %11.8f %11.8f\n",
dy[4],d3y[3],d5y[2],d7y[1],d9y[0]);
fprintf(fpt,"%11.8f %11.8f  %11.8f  %11.8f %11.8f  %11.8f  %11.8f\n",
x[5],y[5],d2y[4],d4y[3],d6y[2],d8y[1],d10y[0]);
fprintf(fpt," %11.8f  %11.8f   %11.8f   %11.8f %11.8f\n",
dy[5],d3y[4],d5y[3],d7y[2],d9y[1]);
fprintf(fpt,"%11.8f %11.8f  %11.8f %11.8f %11.8f  %11.8f\n",
x[6],y[6],d2y[5],d4y[4],d6y[3],d8y[2]);
fprintf(fpt," %11.8f %11.8f  %11.8f %11.8f\n",
dy[6],d3y[5],d5y[4],d7y[3]);
fprintf(fpt,"%11.8f %11.8f %11.8f  %11.8f   %11.8f\n",
x[7],y[7],d2y[6],d4y[5],d6y[4]);
fprintf(fpt," %11.8f  %11.8f %11.8f\n",
dy[7],d3y[6],d5y[5]);
fprintf(fpt,"%11.8f %11.8f %11.8f %11.8f\n",
x[8],y[8],d2y[7],d4y[6]);
fprintf(fpt," %11.8f %11.8f\n",
```

```
dy [8],d3y[7]);
fprintf (fpt,"%11.8f %11.8f %11.8f\n",x[9],y[9],d2y[8]);
fprintf (fpt," %11.8f\n",dy[9]);
fprintf (fpt,"%11.8f %11.8f\n",x[10],y[10]);
for (i=0; i<11; i++)
{fprintf (fpt,"aa[%2d]=%18.8f\n",i,aa[i]);
Fprintf (fpt,"c [%2d] =%18.8f\n", i,c [i]);}
xx =a;
printf ("xx=%10.4f\n",xx);
while (xx<=b)
{fx =0;
For (i=0; i<11; i++)
fx=fx+c[i]*pow( (double) xx, (double) i);
ax=sin( (double) xx);
fprintf(fpt,"%12.4f %18.8f %18.8f\n", xx,ax,fx);
xx=xx+0.01;}
fprintf(fpt,"xx=%10.4f\n",xx);
getch();
fclose(fpt);}
float factorial(int n)
{int i;
float f=1;
for(i=1; i<=n; i++)
f=f*i;
return(f);}
float p(int c,int d)
```

```
{float pr=0;
int i,j,k,ii,jj,kk,iii,jjj,kkk,iiii;
switch (d)
{case 1 : for(i=0; i<c; i++)
pr=pr+x[i];
break;
case 2 : for(i=0; i<c-1; i++)
{for(j=i+1; j<c; j++)
pr=pr+x[i]*x[j];}
break;
case 3 : for(i=0; i< c-2; i++)
{for(j=i+1; j<c-1; j++)
{for(k=j+1; k<c; k++)
pr=pr+x[i]*x[j]*x[k];}}
break;
case 4 : for(i=0; i< c-3; i++)
{for(j=i+1; j<c-2; j++)
{for(k=j+1; k<c-1; k++)
{for(ii=k+1; ii<c; ii++)
pr=pr+x[i]*x[j]*x[k]*x[ii];}}}
break;
case 5 : for(i=0; i< c-4; i++)
{for(j=i+1; j<c-3; j++)
{for(k=j+1; k<c-2; k++)
{for(ii=k+1; ii<c-1; ii++)
{for(jj=ii+1; jj<c; jj++)
```

```
pr=pr+x[i]*x[j]*x[k]*x[ii]*x[jj];}}}}
break;
case 6 : for(i=0; i< c-5; i++)
{for(j=i+1; j<c-4; j++)
{for(k=j+1; k<c-3; k++)
{for(ii=k+1; ii<c-2; ii++)
{for(jj=ii+1; jj<c-1; jj++)
{for(kk=jj+1; kk<c; kk++)
pr=pr+x[i]*x[j]*x[k]*x[ii]*x[jj]*x[kk];}}}}}
break;
case 7 : for(i=0; i< c-6; i++)
{for(j=i+1; j<c-5; j++)
{for(k=j+1; k<c-4; k++)
{for(ii=k+1; ii<c-3; ii++)
{for(jj=ii+1; jj<c-2; jj++)
{for(kk=jj+1; kk<c-1; kk++)
{for(iii=kk+1; iii<c; iii++)
pr=pr+x[i]*x[j]*x[k]*x[ii]*x[jj]*x[kk]*x[iii];}}}}}}
break;
case 8 : for(i=0; i< c-7; i++)
{for(j=i+1; j<c-6; j++)
{for(k=j+1; k<c-5; k++)
{for(ii=k+1; ii<c-4; ii++)
{for(jj=ii+1; jj<c-3; jj++)
{for(kk=jj+1; kk<c-2; kk++)
{for(iii=kk+1; iii<c-1; iii++)
```

```
{for(jjj=iii+1; jjj<c; jjj++)
{pr=pr+x[i]*x[j]*x[k]*x[ii]*x[jj]*x[kk]*x[iii]*x[jjj];}}}}}}}}
break;
case 9 : for(i=0; i< c-8; i++)
{for(j=i+1; j<c-7; j++)
{for(k=j+1; k<c-6; k++)
{for(ii=k+1; ii<c-5; ii++)
{for(jj=ii+1; jj<c-4; jj++)
{for(kk=jj+1; kk<c-3; kk++)
{for(iii=kk+1; iii<c-2; iii++)
{for(jjj=iii+1; jjj<c-1; jjj++)
{for(kkk=jjj+1; kkk<c; kkk++)
{pr=pr+x[i]*x[j]*x[k]*x[ii]*x[jj]*x[kk]*x[iii]*x[jjj]*x[kkk];}}}}}}}}}
break;
case 10 : for(i=0; i< c-9; i++)
{for(j=i+1; j<c-8; j++)
{for(k=j+1; k<c-7; k++)
{for(ii=k+1; ii<c-6; ii++)
{for(jj=ii+1; jj<c-5; jj++)
{for(kk=jj+1; kk<c-4; kk++)
{for(iii=kk+1; iii<c-3; iii++)
{for(jjj=iii+1; jjj<c-2; jjj++)
{for(kkk=jjj+1; kkk<c-1; kkk++)
{for(iiii=kkk+1; iiii<c; iiii++)
{pr=pr+x[i]*x[j]*x[k]*x[ii]*x[jj]*x[kk]*x[iii]*x[jjj]*x[kkk]*x[iiii];}}}}}}}}}}
break;}
```

```
//printf("p(%2d,%2d)=%12.4f\n",c,d,pr);

return(pr);}
```

RESULT AND DISCUSSION

Interpolation of $f(x)=x^{1/2}$ in the interval [1, 2]: The interval has been divided into 10 equal parts with the help of the points $x_0, x_1, \ldots, x_{10}$ such that

$x_i=x_0+ih$, $i=0,1,2,\ldots, 10$, $h=(b-a)/n$, $n=10$. Polynomial obtained by Newton's divided difference formula using the developed C++ program is given below-

$f(x) = c_0 + c_1 x + c_2 x^2 + \ldots + c_{10} x^{10}$ Where

$c_0 = -1.944630265236$

$c_1 = 16.710096359253$

$c_2 = -49.732860565186$

$c_3 = 91.514305114746$

$c_4 = -110.665145874023$

$c_5 = 91.462081909180$

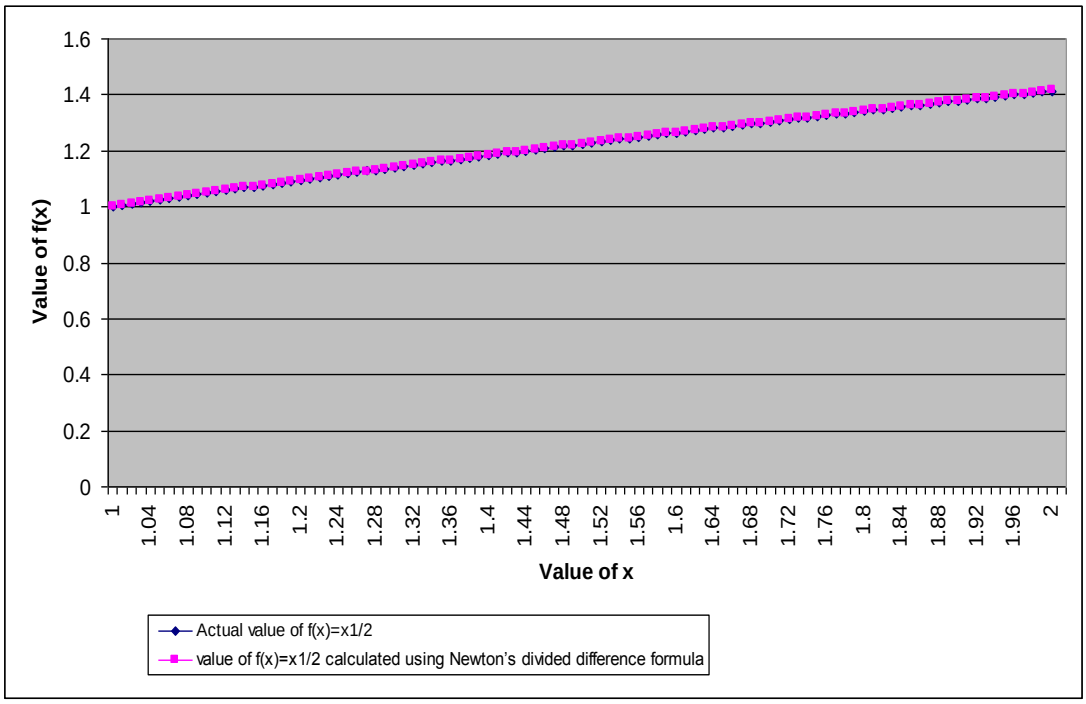
$c_6 = -52.255199432373$

$c_7 = 20.369894027710$

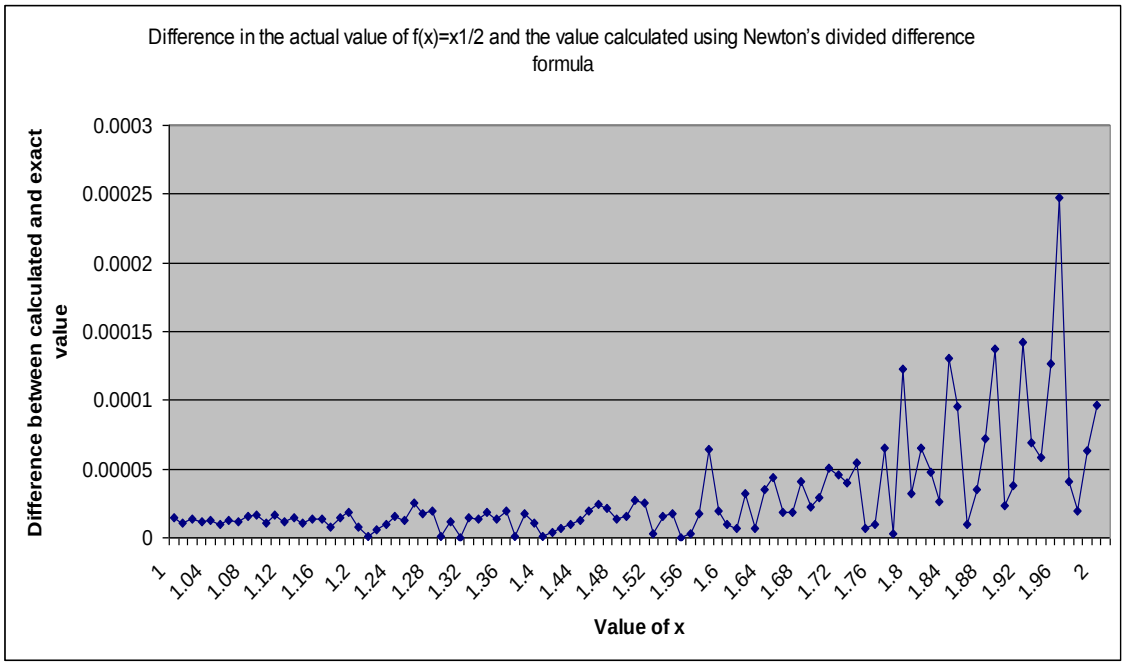
$c_8 = -5.184066295624$

$c_9 = 0.777744591236$

$c_{10} = -0.052232898772$



Graph-1: Graph between actual values of f(x) = x1/2 and the values calculated by Newton's interpolating polynomial at different points in the interval [1, 2] separated by the distance 0.1



Graph-2: Difference between actual and calculated values of the function f(x) = $x^{1/2}$ by Newton's interpolating polynomial.

Table 1: Difference triangle for the function $f(x) = x^{1/2}$ at points of interval [1, 2]

x	y=f(x)	Δy	$\Delta^2 y$	$\Delta^3 y$	$\Delta^4 y$	$\Delta^5 y$	$\Delta^6 y$	$\Delta^7 y$	$\Delta^8 y$	$\Delta^9 y$	$\Delta^{10} y$
1.00000000	1.00000000										
		0.04880881									
1.10000002	1.04880881		-0.00217247								
		0.04663634		0.00026643							
1.20000005	1.09544516		-0.00190604		-0.00005019						
		0.04473031		0.00021625		0.00001216					
1.29999995	1.14017546		-0.00168979		-0.00003803		-0.00000358				
		0.04304051		0.00017822		0.00000858		0.00000179			
1.39999998	1.18321598		-0.00151157		-0.00002944		-0.00000179		-0.00000286		
		0.04152894		0.00014877		0.00000679		-0.00000107		0.00000739	
1.50000000	1.22474492		-0.00136280		-0.00002265		-0.00000286		0.00000453		-0.00001895
		0.04016614		0.00012612		0.00000393		0.00000346		-0.00001156	
1.60000002	1.26491106		-0.00123668		-0.00001872		0.00000060		-0.00000703		
		0.03892946		0.00010741		0.00000453		-0.00000358			
1.70000005	1.30384052		-0.00112927		-0.00001419		-0.00000298				
		0.03780019		0.00009322		0.00000155					
1.79999995	1.34164071		-0.00103605		-0.00001264						
		0.03676414		0.00008059							
1.89999998	1.37840486		-0.00095546								
		0.03580868									
2.00000000	1.41421354										

Table 2: Actual value of $f(x) = x^{1/2}$, Calculated value of $f(x)=x^{1/2}$ by Newton's interpolating polynomial, Difference between actual and calculated values of $f(x)=x^{1/2}$ by Newton's interpolating polynomial, Percentage error in the values of $f(x)=x^{1/2}$ calculated by Newton's interpolating polynomial at different values of x

Value of x	Actual value of $f(x)=x^{1/2}$	value of $f(x)=x^{1/2}$ calculated using Newton's divided difference formula	Difference in the actual value of $f(x)=x^{1/2}$ and the value calculated using Newton's divided difference formula	Percentage error in the value calculated using Newton's divided difference formula
1.00	1.00000000	0.99998528	0.00001472	0.00147223
1.01	1.00498760	1.00497699	0.00001061	0.00105570
1.02	1.00995052	1.00993705	0.00001347	0.00133379
1.03	1.01488912	1.01487720	0.00001192	0.00117460
1.04	1.01980388	1.01979160	0.00001228	0.00120401
1.05	1.02469504	1.02468491	0.00001013	0.00098886
1.06	1.02956295	1.02954996	0.00001299	0.00126207
1.07	1.03440797	1.03439629	0.00001168	0.00112939
1.08	1.03923047	1.03921461	0.00001585	0.00152563
1.09	1.04403067	1.04401422	0.00001645	0.00157571
1.10	1.04880881	1.04879856	0.00001025	0.00097749
1.11	1.05356538	1.05354846	0.00001693	0.00160671
1.12	1.05830050	1.05828881	0.00001168	0.00110389
1.13	1.06301451	1.06299996	0.00001454	0.00136814
1.14	1.06770778	1.06769753	0.00001025	0.00096019
1.15	1.07238042	1.07236683	0.00001359	0.00126726
1.16	1.07703292	1.07701898	0.00001395	0.00129499
1.17	1.08166528	1.08165741	0.00000787	0.00072738
1.18	1.08627796	1.08626378	0.00001419	0.00130592
1.19	1.09087110	1.09085298	0.00001812	0.00166104
1.20	1.09544504	1.09543693	0.00000811	0.00073999
1.21	1.09999990	1.09999859	0.00000131	0.00011921
1.22	1.10453606	1.10453069	0.00000536	0.00048567
1.23	1.10905349	1.10904360	0.00000989	0.00089215
1.24	1.11355281	1.11353695	0.00001585	0.00142381
1.25	1.11803389	1.11802137	0.00001252	0.00111955
1.26	1.12249708	1.12247193	0.00002515	0.00224082
1.27	1.12694263	1.12692499	0.00001764	0.00156556
1.28	1.13137078	1.13135087	0.00001991	0.00175963
1.29	1.13578153	1.13578272	0.00000119	0.00010496
1.30	1.14017534	1.14016378	0.00001156	0.00101417
1.31	1.14455223	1.14455187	0.00000036	0.00003125

Value of x	Actual value of $f(x)=x^{1/2}$	value of $f(x)=x^{1/2}$ calculated using Newton's divided difference formula	Difference in the actual value of $f(x)=x^{1/2}$ and the value calculated using Newton's divided difference formula	Percentage error in the value calculated using Newton's divided difference formula
1.32	1.14891243	1.14889753	0.00001490	0.00129698
1.33	1.15325618	1.15324283	0.00001335	0.00115772
1.34	1.15758359	1.15756488	0.00001872	0.00161680
1.35	1.16189492	1.16188109	0.00001383	0.00119015
1.36	1.16619027	1.16617095	0.00001931	0.00165598
1.37	1.17046988	1.17047131	0.00000143	0.00012222
1.38	1.17473388	1.17471647	0.00001740	0.00148157
1.39	1.17898250	1.17897201	0.00001049	0.00088979
1.40	1.18321574	1.18321478	0.00000095	0.00008060
1.41	1.18743408	1.18743837	0.00000429	0.00036141
1.42	1.19163740	1.19163036	0.00000703	0.00059023
1.43	1.19582593	1.19583559	0.00000966	0.00080747
1.44	1.19999981	1.20001245	0.00001264	0.00105302
1.45	1.20415926	1.20417917	0.00001991	0.00165327
1.46	1.20830441	1.20832837	0.00002396	0.00198303
1.47	1.21243536	1.21241379	0.00002158	0.00177963
1.48	1.21655238	1.21656597	0.00001359	0.00111708
1.49	1.22065532	1.22067058	0.00001526	0.00125005
1.50	1.22474468	1.22477233	0.00002766	0.00225815
1.51	1.22882032	1.22884560	0.00002527	0.00205664
1.52	1.23288262	1.23288012	0.00000250	0.00020305
1.53	1.23693144	1.23691595	0.00001550	0.00125288
1.54	1.24096715	1.24094939	0.00001776	0.00143132
1.55	1.24498975	1.24498940	0.00000036	0.00002873
1.56	1.24899936	1.24899673	0.00000262	0.00020998
1.57	1.25299621	1.25301385	0.00001764	0.00140806
1.58	1.25698030	1.25704503	0.00006473	0.00514969
1.59	1.26095176	1.26097143	0.00001967	0.00155990
1.60	1.26491082	1.26492095	0.00001013	0.00080107
1.61	1.26885748	1.26886439	0.00000691	0.00054491
1.62	1.27279198	1.27282429	0.00003231	0.00253818
1.63	1.27671432	1.27672136	0.00000703	0.00055089
1.64	1.28062463	1.28065991	0.00003529	0.00275537
1.65	1.28452301	1.28456652	0.00004351	0.00338736
1.66	1.28840959	1.28842819	0.00001860	0.00144338
1.67	1.29228461	1.29230285	0.00001824	0.00141138
1.68	1.29614794	1.29618931	0.00004137	0.00319143
1.69	1.29999971	1.30002189	0.00002217	0.00170561

Value of x	Actual value of $f(x)=x^{1/2}$	value of $f(x)=x^{1/2}$ calculated using Newton's divided difference formula	Difference in the actual value of $f(x)=x^{1/2}$ and the value calculated using Newton's divided difference formula	Percentage error in the value calculated using Newton's divided difference formula
1.70	1.30384028	1.30386996	0.00002968	0.00227659
1.71	1.30766940	1.30771983	0.00005043	0.00385614
1.72	1.31148744	1.31153333	0.00004590	0.00349951
1.73	1.31529438	1.31533432	0.00003994	0.00303621
1.74	1.31909037	1.31914496	0.00005460	0.00413905
1.75	1.32287538	1.32286859	0.00000679	0.00051365
1.76	1.32664967	1.32664025	0.00000942	0.00070987
1.77	1.33041322	1.33047867	0.00006545	0.00491922
1.78	1.33416617	1.33416867	0.00000250	0.00018764
1.79	1.33790851	1.33803117	0.00012267	0.00916852
1.80	1.34164047	1.34167218	0.00003171	0.00236350
1.81	1.34536207	1.34542727	0.00006521	0.00484684
1.82	1.34907341	1.34912097	0.00004756	0.00352572
1.83	1.35277462	1.35274851	0.00002611	0.00192987
1.84	1.35646570	1.35659647	0.00013077	0.00964069
1.85	1.36014676	1.36024237	0.00009561	0.00702908
1.86	1.36381781	1.36380804	0.00000978	0.00071675
1.87	1.36747909	1.36751378	0.00003469	0.00253678
1.88	1.37113059	1.37120223	0.00007164	0.00522523
1.89	1.37477243	1.37491024	0.00013781	0.01002391
1.90	1.37840462	1.37838101	0.00002360	0.00171237
1.91	1.38202715	1.38198936	0.00003779	0.00273434
1.92	1.38564038	1.38578224	0.00014186	0.01023780
1.93	1.38924408	1.38917506	0.00006902	0.00496833
1.94	1.39283848	1.39289713	0.00005865	0.00421090
1.95	1.39642370	1.39654994	0.00012624	0.00904042
1.96	1.39999962	1.40024686	0.00024724	0.01766001
1.97	1.40356660	1.40352559	0.00004101	0.00292170
1.98	1.40712440	1.40714371	0.00001931	0.00137244
1.99	1.41067326	1.41073632	0.00006306	0.00447033
2.00	1.41421318	1.41430986	0.00009668	0.00683622
Average percentage error				0.00234432

Difference triangle for the function $f(x)=x^{1/2}$ at the points of interval [1, 2] is given in **Table 1**. Actual value of $f(x)=x^{1/2}$, Calculated value of $f(x)=x^{1/2}$ by Newton's interpolating polynomial, Difference between actual and calculated values of $f(x)=x^{1/2}$ by Newton's interpolating polynomial, Percentage error in the values of $f(x)=x^{1/2}$ calculated by Newton's interpolating polynomial at different values of x

is given in **Table 2**. Graph between actual values of $f(x) = x^{1/2}$ and the values calculated by Newton's interpolating polynomial at different points in the interval [1, 2] separated by the distance 0.1 is given in **Graph-1**. Difference between actual and calculated values of the function $f(x) = x^{1/2}$ by Newton's interpolating polynomial is shown in **Graph-2**. Average percentage error in the values obtained by Newton's interpolating polynomial is 0.00234432%.

Interpolation of $f(x) = x^{1/3}$ in the interval [1, 2]: Average percentage error in the values of the function $f(x)=x^{1/3}$ as obtained by Newton's interpolating polynomial is 0.00222263%.

Interpolation of $f(x)=x^{1/4}$ in the interval [1, 2]: Average percentage error in the values of the function $f(x)=x^{1/4}$ as obtained by Newton's interpolating polynomial is 0.00240526 %.

CONCLUSION

Average percentage error in the interpolation of functions $x^{1/2}$, $x^{1/3}$ and $x^{1/4}$ using Newton's divided difference formulas are 0.00234432%, 0.00222263% and 0.00240526% respectively. It means $x^{1/3}$ is better interpolated as compared to $x^{1/2}$ and $x^{1/4}$. The order of accuracy of these functions is as follows-

$x^{1/3} > x^{1/2} > x^{1/4s}$

REFERENCES

1. E.J. McShane. *American Mathematical Monthly*. 1946, **53** (5), 259.
2. P.M. Hummel. *American Mathematical Monthly*. 1947, **54** (4), 218.
3. B. Fischer. Lothar Reichel. *Math. of Comput*. 1989, **53**, (187), 265.
4. Monatsh. *Math.* 2000, 131, 215.
5. Michael I. Ganzburg. *Bull. Austral. Math. Soc*. 2004, **70**, 475.
6. Kunkle Thomas. *Journal of approximation theory*. 1992, **71** (1), 94.
7. C.R. Kannappan Pl. *Math. Rep. Acad. Sci. Canada*. 1994, **16** (5), 187.
8. Schwaiger. *J. Aequationes mathematicae*. 1994, **48** (2/3), 317.
9. M. Neamtu. *SIAM J. on Numerical Analysis*. 1992, **29** (5), 1435.
10. P. L. Chebyshev, *Me m. Acad. Sci. St. Pe tersb*. 1859, **1**(7), 124.
11. K. Weierstrass, *Mathematische Werke. 1985,* 3, 1.
12. E. Egecioglu, Omer Gallopoulos, K. Koc Cetin. *Journal of complexity*. 1989, **5** (4), 417.
13. Goodyear, H. William. *J. Astronaut. Sci*. 1986, **34** (3), 287.
14. Hama, Hiromitsu, Xing, Chunfeng, Liu, Zhongkan. 1998, **81** (12), 2688.
15. R. Howell, J. Mathews. The AMATYC Review. 1992, **14** (1), 20.
16. G. H. F. Gardner. A. Canning. *64th Annual International Meeting, SEG, Expanded Abstracts*. 1994, 1553–1556
17. N. Chemingui. B. Biondi. *66th Annual International Meeting, SEG, Expanded Abstracts*. 1996., 32–35.
18. Ronen *et al*. *Geophysical Journal International*. 1991, 1, 503.
19. Ronen *et al*. *65th Annual International Meeting, SEG, Expanded Abstracts*. 1995, 743-746.
20. G. Mühlbach. *Numer. Math*. 1979, **32** (4), 393.
21. R.J. Pan. *Journal- Fujian Teachers University Natural Science Edition*. 2001, **17** (2), 28.

22. M. Higashi, K. Amada. *Journal- Japan Society for Precision Engineering*. 2001, **67** (5), 749.
23. Argyros, K. Ioannis, Catinas, Emil, Pavaloiu, Ion Adv. Nonlinear Var. Inequal. 2000, **3** (1), 7.
24. Rabut, Christophe. *SIAM J. Numer. Anal*. 2000, **38** (4), 1294.
25. M.P. Cullinan. *IMA J. of numerical analysis*. 1990, **10** (4), 583.
26. E.T.Y. Lee. *American Mathematical Monthly*. 1989, **96** (7), 618.
27. A. McCurdy, K.C. Ng, B.N. Parlett. *Mathematics of Computation*. 1984, **43** (168), 501.
28. J.I. Maeztu. *SIAM J. on Numerical Analysis*. 1982, **19** (5), 1032.
29. F.T. Krogh. *Mathematics of Computation*. 1979, **33** (148), 1265.
30. Al-Hussainan, A. Adel, Al-Eideh, M. Basel, Al-Zalzalah, S.H. Yousef. *Int. J. Appl. Math*. 2001, **7** (3), 325.
31. A.E. Al-Ayyoub. *Comput. & Structures*. 1996, **58** (4), 689.
32. Herbert E. Salzer. *Proceedings of the American Mathematical Society*. 1962, **13** (2), 210.

*** Corresponding author: Dilli Raj Neupane;**

Department of mathematics, Patan Multiple Campus, Tribhuvan University, Kathmandu, Nepal.
Email: dil.n@hotmail.com