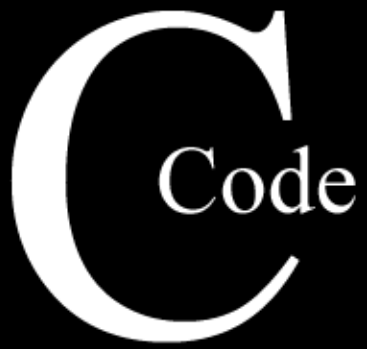
PROGRAMMING TEST PROBLEMS



OUTLINE

- Newton-Raphson Method
- Lagrange Interpolating Polynomial

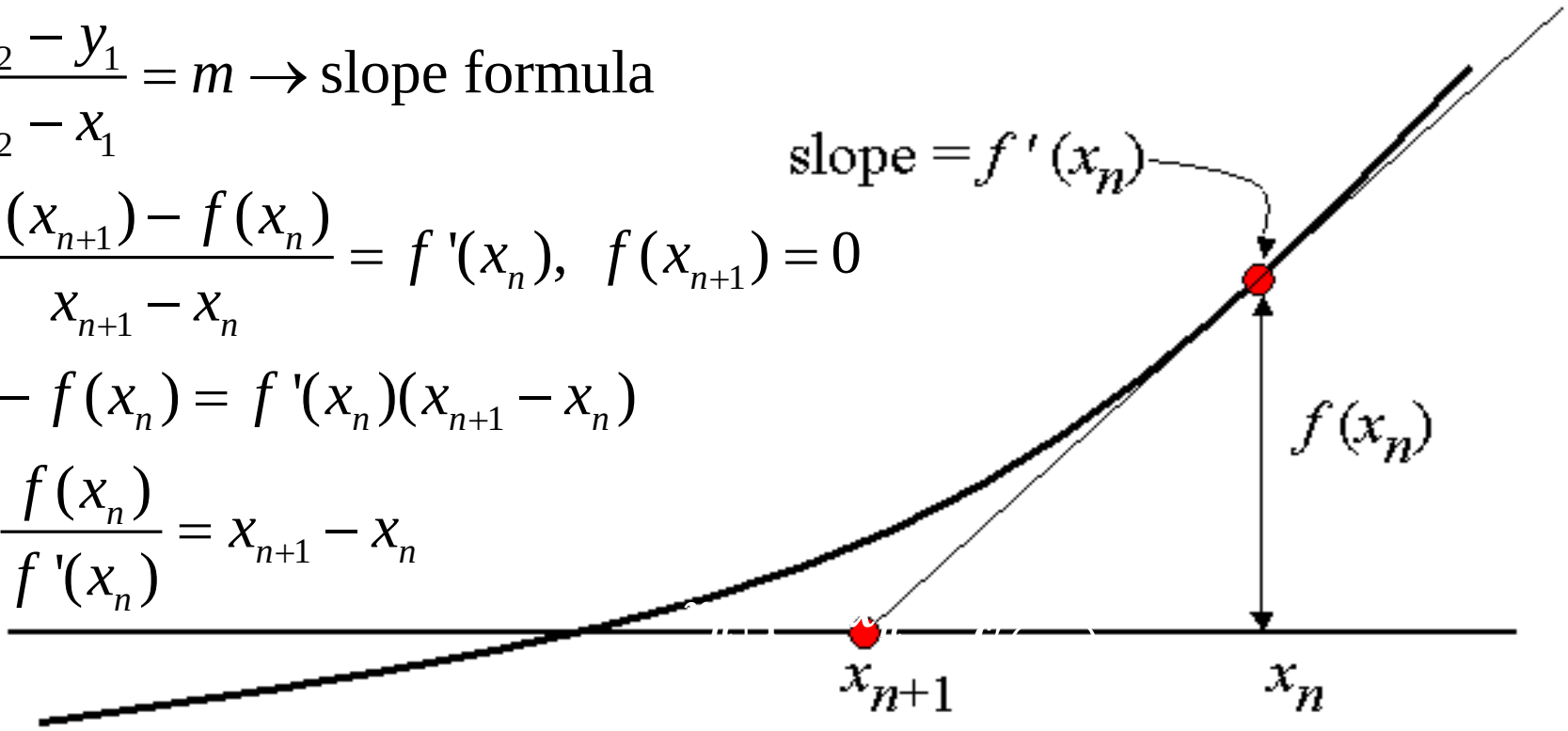
NEWTON-RAPHSON METHOD

$$\frac{y_2 - y_1}{x_2 - x_1} = m \rightarrow \text{slope formula}$$

$$\frac{f(x_{n+1}) - f(x_n)}{x_{n+1} - x_n} = f'(x_n), \quad f(x_{n+1}) = 0$$

$$0 - f(x_n) = f'(x_n)(x_{n+1} - x_n)$$

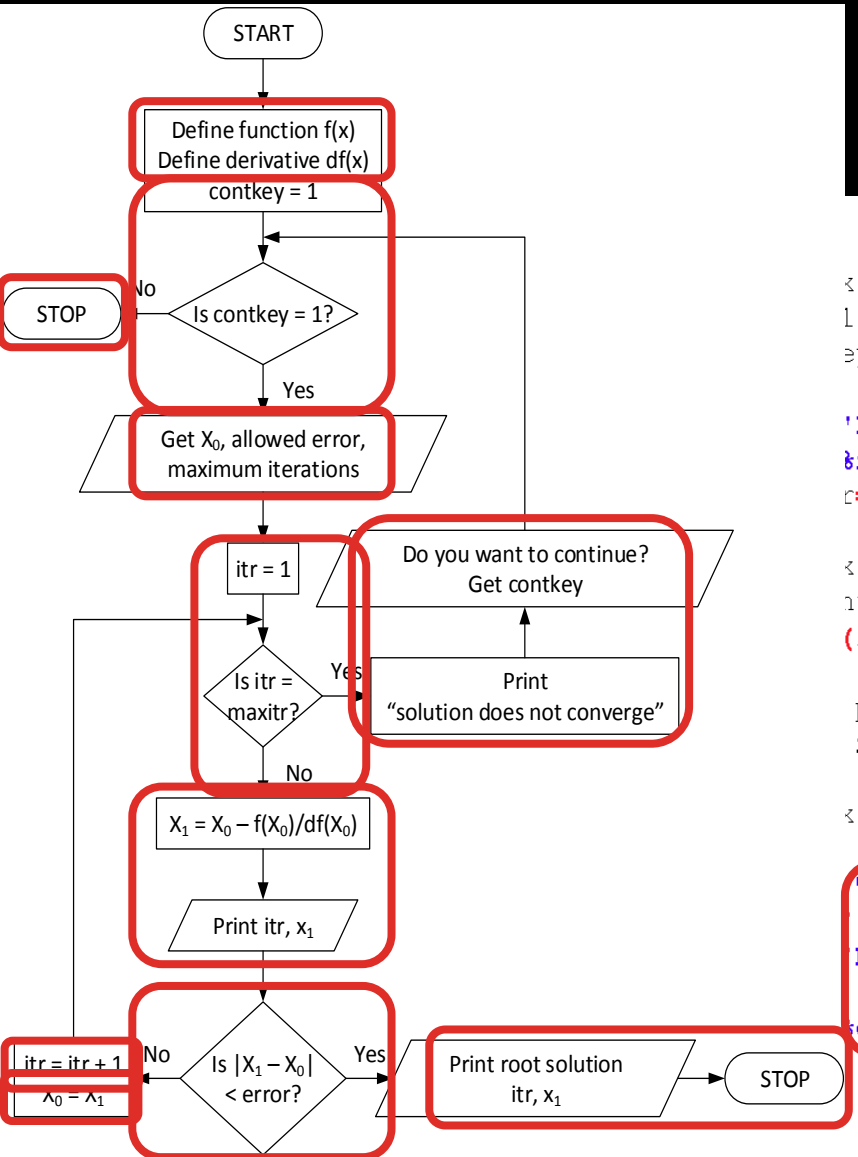
$$-\frac{f(x_n)}{f'(x_n)} = x_{n+1} - x_n$$



$$x_{n+1} = x_n - \frac{f'(x_n)}{f(x_n)}$$

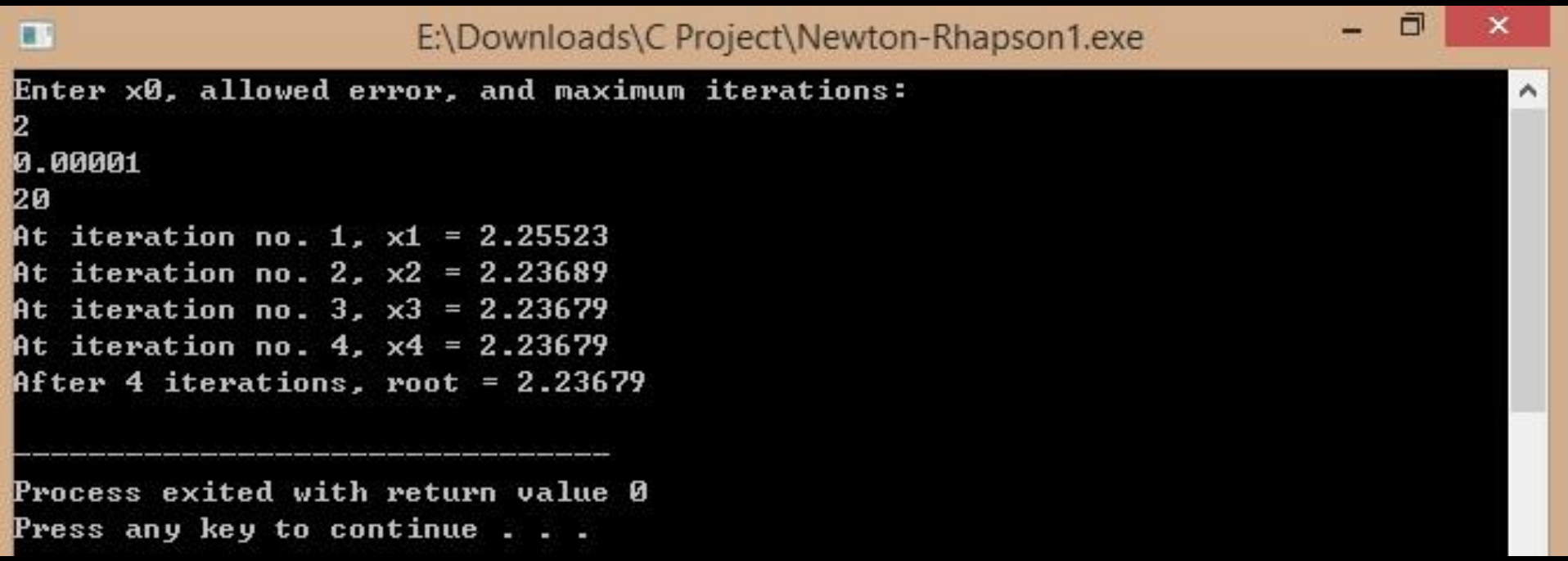
NEWTON-RHAPSON METHOD

Main program




```
itr, contkey=1;
l, err;
ey==1)

'Enter x0, allowed error, and maximum iterations:\n");
%f %f %d", &x0, &err, &maxitr);
r=1; itr<=maxitr; itr++)

x0-f(x0)/df(x0);
ntf("At iteration no. %d, x%d = %0.5f\n", itr, itr, x1);
(fabs(x1-x0) < err)

 printf("After %d iterations, root = %0.5f\n", itr, x1);
 return 0;

x1;

"The solution does not converge or iteration number is
);
"Do you want to continue?\nPress 1 to continue or any other key

d",&contkey);
```

NEWTON-RHAPSON METHOD



NEWTON-RHAPSON METHOD

Test problems

$f_1(x) = x^{2.3} - 6.37 = 0$

$f_2(x) = \sin x - 0.5x = 0$

Termination criterion $|x_{n+1} - x_n| < \varepsilon$, say $\varepsilon = 10^{-6}$

$x_0 = \{0.8, 1, 1.2, 7\}$

Wolfram Demonstrations Project

The Newton-Raphson Method

Dev-C++

$f_1(x)$

$f_2(x)$

REFERENCE WEBSITES

- Newton-Raphson Method

  http://bit.ly/1s4whrh

LAGRANGE INTERPOLATING POLYNOMIAL

- Start with a set of $n+1$ points with different x-coordinates

  $(x_0, y_0), (x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$

- Find a polynomial function that passes through all $n+1$ points

- Interpolating polynomial 

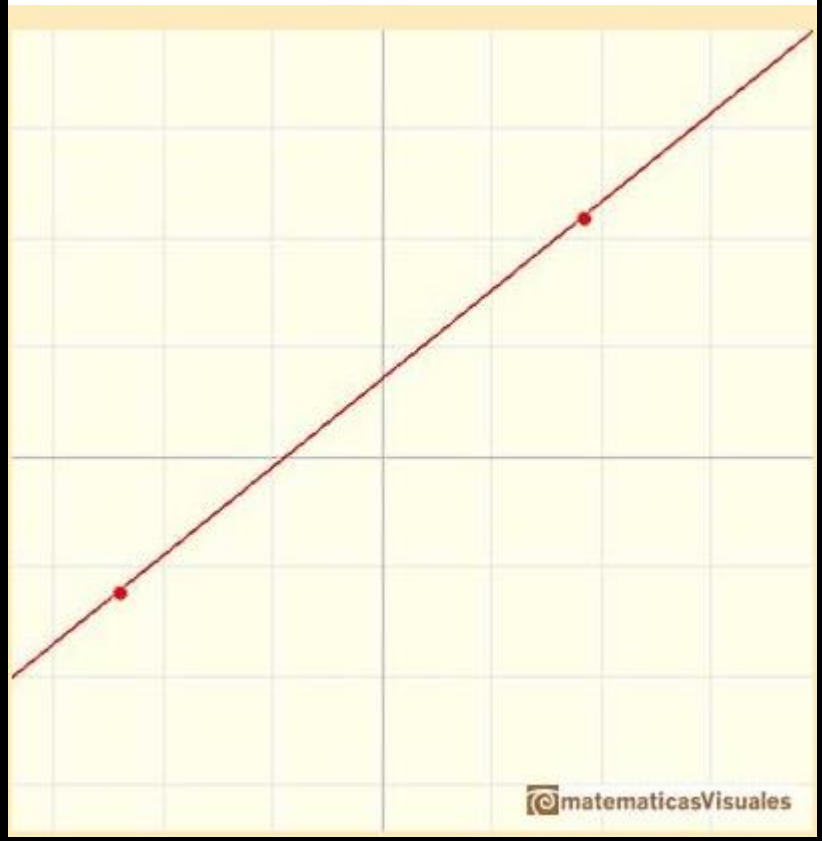
LAGRANGE INTERPOLATING POLYNOMIAL

- Polynomial: $f(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n$
- General formula: $f(x) = \sum_{i=1}^{n+1} L_i(x) y_i$
- Lagrange basis polynomials: $L_i(x) = \prod_{\substack{j=1 \\ i \neq j}}^{n+1} \frac{x - x_j}{x_i - x_j}$
- Expand:

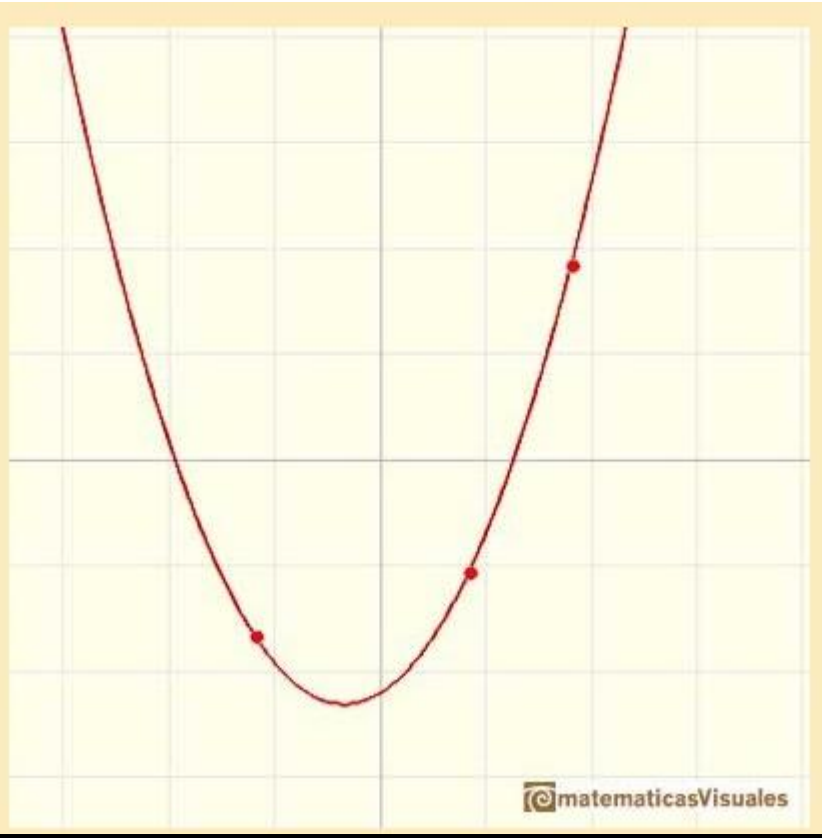
$$L_i(x) = \prod_{\substack{j=1 \\ i \neq j}}^{n+1} \frac{x - x_j}{x_i - x_j} = \frac{x - x_0}{x_1 - x_0} \cdots \frac{x - x_{i-1}}{x_i - x_{i-1}} \cdot \frac{x - x_{i+1}}{x_i - x_{i+1}} \cdots \frac{x - x_{n+1}}{x_i - x_{n+1}}$$

LAGRANGE INTERPOLATING POLYNOMIAL

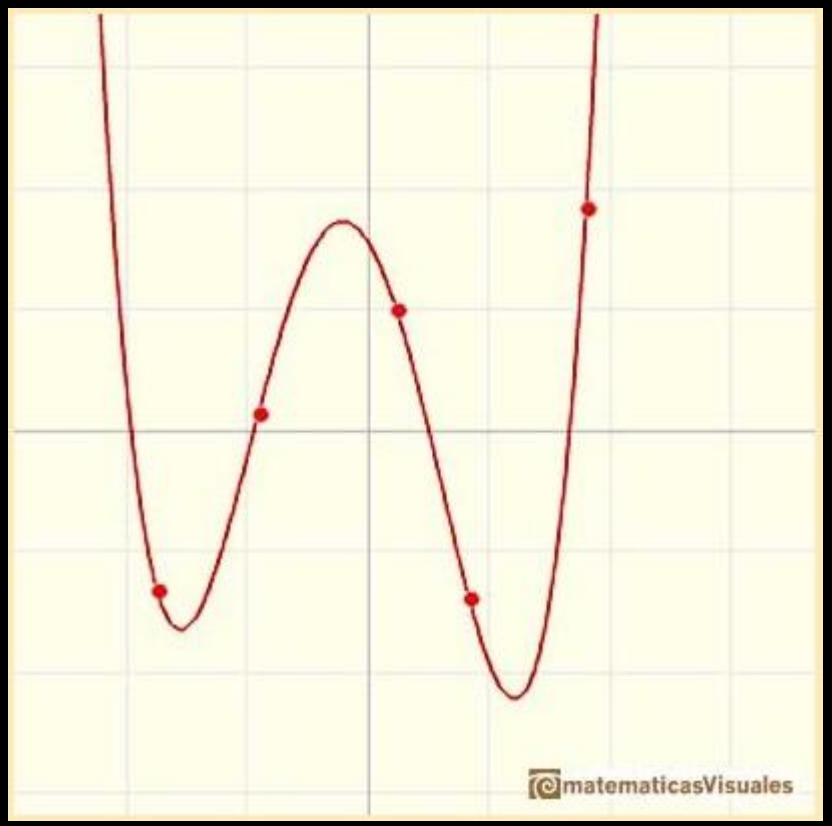
$$f(x) = \frac{x - x_1}{x_0 - x_1} y_0 + \frac{x - x_0}{x_1 - x_0} y_1$$



$$f(x) = \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)} y_0 + \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)} y_1 + \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)} y_2$$



LAGRANGE INTERPOLATING POLYNOMIAL



LAGRANGE INTERPOLATING POLYNOMIAL

x	y
0.1	1.20998
0.3	1.68847
0.45	2.09434
0.62	2.59304
0.75	2.96104
0.86	3.33271
0.91	3.48612
0.93	3.54692
1.15	4.17276
1.30	4.51711

- Determine the values of y with Lagrange interpolation when

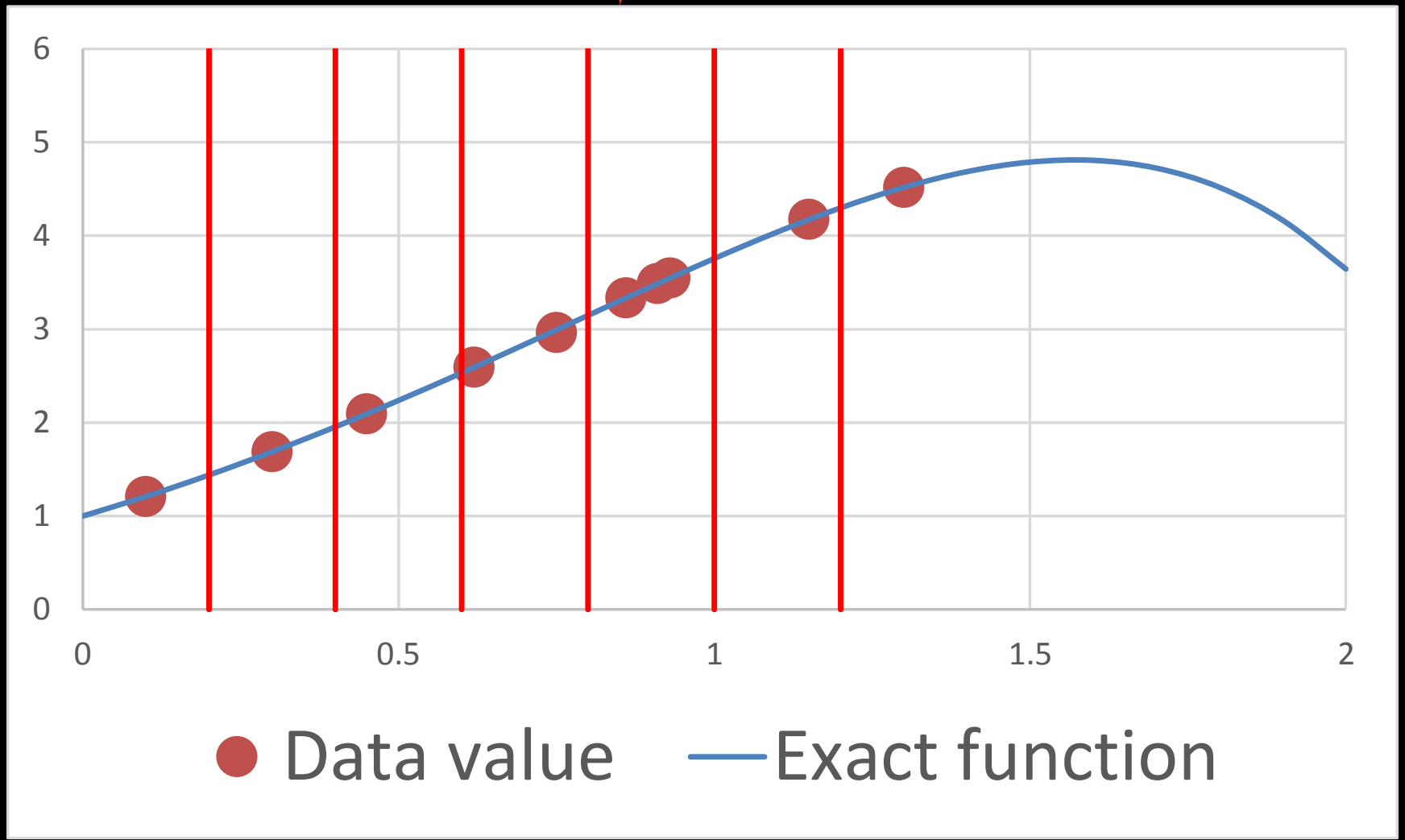
  x = {0.2, 0.4, 0.6, 0.8, 1.0, 1.2}

- Compare interpolated values with exact values according to

  $$y = e^x (\cos x + \sin x)$$

LAGRANGE INTERPOLATING POLYNOMIAL

Lagrange Interpolating Polynomial



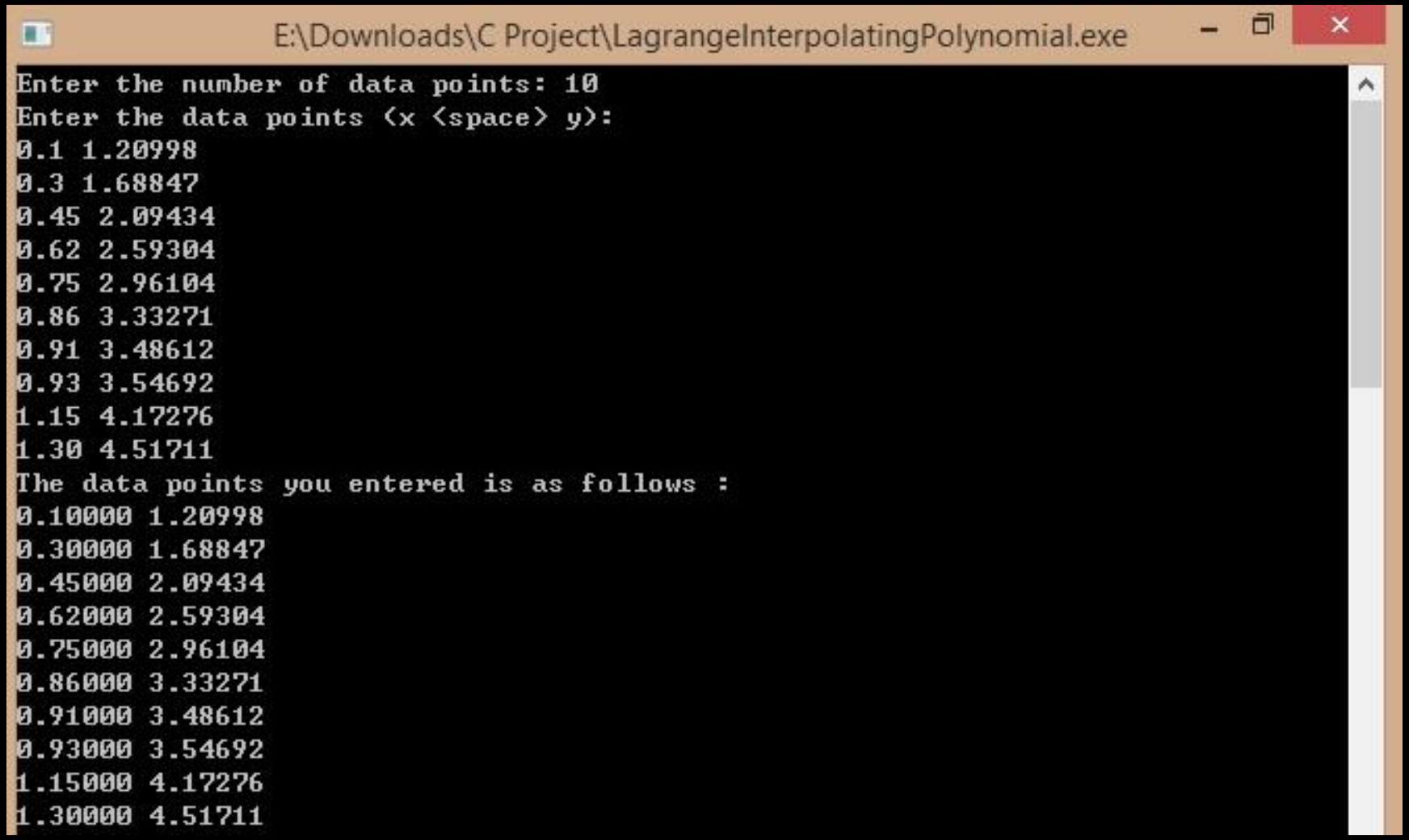
LAGRANGE INTERPOLATING POLYNOMIAL

- For any arbitrary *n+1* data points [x_i , y_i]
- Evaluated at x_{eval}
- Step 1: Let *f* = 0
- Step 2: For *i* = 1 to *n*+1 do steps 3 to 6
- Step 3: Let *s* = 1
- Step 4: For *j* = 1 to *n*+1 do step 5
- Step 5: if *i* and *j* are not equal then let $\boldsymbol{s = s\dfrac{x_{eval} - x_j}{x_i - x_j}}$
- Step 6: $\boldsymbol{f(x_{eval}) = f(x_{eval}) + s \times y_i}$
- Step 7: End

$$\boldsymbol{f(x_{eval}) = \sum_{i=1}^{n+1} L_i(x_{eval}) y_i}$$

$$\boldsymbol{L_i(x_{eval}) = \prod_{\substack{j=1 \\ i \neq j}}^{n+1} \frac{x_{eval} - x_j}{x_i - x_j}}$$

LAGRANGE INTERPOLATING POLYNOMIAL



LAGRANGE INTERPOLATING POLYNOMIAL

```
Enter the number of x points to be evaluated: 6
Enter the data of x points to be evaluated:
0.2 0.4 0.6 0.8 1.0 1.2
The data of x points you entered is as follows :
0.20000
0.40000
0.60000
0.80000
1.00000
1.20000
The value of interpolated y is :
2.90039
1.84721
2.55064
3.13415
3.77633
4.07465
The exact value of y is :
1.43971
1.95501
2.53271
3.14705
3.75605
4.29755
```

LAGRANGE INTERPOLATING POLYNOMIAL

Wolfram Demonstrations Project

Lagrange Interpolating Polynomial

Dev-C++

Lagrange Interpolating Polynomial

LAGRANGE INTERPOLATING POLYNOMIALS

Input data points

```
#include<stdio.h>
#include<math.h>
main()
{
    float x[100],y[100],xeval[100],finter[100],fexact[100],ferror[100],s;
    int n,i,j,neval,ieval,contkey=1;
    printf("Enter the number of data points: ");
    scanf("%d",&n);
    printf("Enter the data points (x <space> y): \n");
    for(i=0; i<n; i++)
    {
        scanf("%f",&x[i]);
        scanf("%f",&y[i]);
    }
    printf("The data points you entered is as follows : \n");
    for(i=0; i<n; i++)
    {
        printf("%0.5f\t%0.5f",x[i],y[i]);
        printf("\n");
    }
```

LAGRANGE INTERPOLATING POLYNOMIALS

Input evaluated data points

```
while(contkey==1)
{
    printf("Enter the number of x points to be evaluated: ");
    scanf("%d",&neval);
    printf("Enter the data of x points to be evaluated: \n");
    for(ieval=0; ieval<neval; ieval++)
    {
        scanf("%f",&xeval[ieval]);
    }
    printf("The data of x points you entered is as follows : \n");
    for(ieval=0; ieval<neval; ieval++)
    {
        printf("%0.5f",xeval[ieval]);
        printf("\n");
    }
```

LAGRANGE INTERPOLATING POLYNOMIALS

Main program & evaluated output

```
float finter[100]={0};
for(ieval=0; ieval<neval; ieval++)
{
    for(i=0; i<n; i++)
    {
        s=1;
        for(j=0; j<n; j++)
        {
            if(j!=i)
            {
                s=s*(xeval[ieval]-x[j])/(x[i]-x[j]);
            }
        }
        finter[ieval]=finter[ieval]+s*y[i];
    }
}
printf("The value of interpolated y is : \n");
for(ieval=0; ieval<neval; ieval++)
{
    printf("%0.5f",finter[ieval]);
    printf("\n");
}
```

LAGRANGE INTERPOLATING POLYNOMIALS

Exact and error output

```
        printf("The exact value of y is : \n");
        for(ieval=0; ieval<neval; ieval++)
        {
fexact[ieval]=exp(xeval[ieval])*(cos(xeval[ieval])+sin(xeval[ieval]));
            printf("%0.5f",fexact[ieval]);
            printf("\n");
        }
        printf("The error value (exact - interpolated) of y is : \n");
        for(ieval=0; ieval<neval; ieval++)
        {
            ferror[ieval]=fexact[ieval]-finter[ieval];
            printf("%0.5f",ferror[ieval]);
            printf("\n");
        }
        printf("Do you want to continue?\nPress 1 to continue or any other key
to exit: ");
        scanf("%d",&contkey);
    }
    return 0;
}
```

REFERENCE WEBSITES

- Interpolating Polynomial

  http://bit.ly/1CYd4Me

  http://bit.ly/12XDGxc

- Lagrange Interpolating Polynomial with Exact Functions

  http://bit.ly/1vEVS9M