FACULTAD DE CIENCIAS

Tarea 5

ANÁLISIS NÚMERICO

Oscar Andrés Rosas Hernandez

Alarcón Alvarez Aylin

Pahua Castro Jesús Miguel Ángel

Diciembre 2018

Índice

1. Problemas de Computadora

Una nota importante es que al inicio de CADA script se incluyen los algoritmos, porfavor cambia el valor de la variable Directory para que sea un string que apunte desde donde estas a donde estan la carpeta de los algoritmos

Esta linea:

```
getd(pwd() + Directory);
```

Para ejecutar cada uno basta con hacer algo como:

```
exec("/Users/mac/Documents/Projects/Learning/UNAM/NumericalAnalysis/Homework4/Code/1.sce", -1)
```

1.1. 12

Ejecuta los scripts que esta dentro de Code llamado: 12a.sce y 12b.sce

En este código muestra justo lo que se nos pide, eso si, calcular este interpolante toma un par de segundos, ten paciencia.

```
// @Author: Rosas Hernandez Oscar Andres
// @Author: Alarcón Alvarez Aylin Yadira Guadalupe
// @Author: Pahua Castro Jesús Miguel Ángel

getd(pwd() + Directory);
clc;

points = [
    -0.5;
    -0.25;
    0;
]

valuations = [
    -0.024;
    0.334;
    1.101;
]

derivatives = [
    0.751;
    2.189;
    4.002;
]

pointsToEvaluate = linspace(-1, 1, 20)'

pointsEvaluated = HermiteInterpolant(points, valuations, derivatives, pointsToEvaluate)

plot(points, valuations, "*red")
plot(pointsToEvaluate, pointsEvaluated, "-blue")

xtitle("Hermite Interpolant", "x", "f(x)");
```

```
// @Author: Rosas Hernandez Oscar Andres
// @Author: Alarcón Alvarez Aylin Yadira Guadalupe
// @Author: Pahua Castro Jesús Miguel Ángel

getd(pwd() + Directory);
clc;

points = [
    0.1;
    0.2;
    0.3;
    0.4;
]

valuations = [
    -0.620;
    -0.283;
    0.006;
    0.248;
]

derivatives = [
    3.585;
    3.140;
    2.666;
    2.165;
]

pointsToEvaluate = linspace(-0.1, 0.6, 20)'

pointsEvaluated = HermiteInterpolant(points, valuations, derivatives, pointsToEvaluate)

plot(points, valuations, "*red")
plot(pointsToEvaluate, pointsEvaluated, "-blue")

xtitle("Hermite Interpolant", "x", "f(x)");
```

1.2. 13

Ejecuta los scripts que esta dentro de Code llamado: 13a.sce, 13b.sce y 13c.sce

Creo que es obvio que gracias a la información en la derivada Hermite será mejor

```
// @Author: Rosas Hernandez Oscar Andres
// @Author: Alarcón Alvarez Aylin Yadira Guadalupe
// @Author: Pahua Castro Jesús Miguel Ángel

getd(pwd() + Directory);
clc;


function [x] = f(x)
    x = cos(x)
endfunction

points = [
    0;
    0.6;
    0.9;
]

valuations = [
    f(points(1));
    f(points(2));
    f(points(3));
]

derivatives = [
    ( f(points(1) + 0.00001) - f(points(1)) ) / (0.00001);
    ( f(points(2) + 0.00001) - f(points(2)) ) / (0.00001);
    ( f(points(3) + 0.00001) - f(points(3)) ) / (0.00001);
]

pointsToEvaluate = linspace(-0.1, 1, 30)'

pointsEvaluated1 = HermiteInterpolant(points, valuations, derivatives, pointsToEvaluate)
pointsEvaluated2 = LagrangeInterpolant(points, valuations, pointsToEvaluate)

plot(points, valuations, "*red")
plot(pointsToEvaluate, pointsEvaluated1, "-blue")
plot(pointsToEvaluate, pointsEvaluated2, "-green")
plot(pointsToEvaluate, f(pointsToEvaluate), "-m")

hl=legend(['Points'; 'Hermite'; 'Lagrange'; 'Real']);
xtitle("Hermite Interpolant", "x", "f(x)");
```

```
// @Author: Rosas Hernandez Oscar Andres
// @Author: Alarcón Alvarez Aylin Yadira Guadalupe
// @Author: Pahua Castro Jesús Miguel Ángel

getd(pwd() + Directory);
clc;


function [x] = f(x)
    x = log(x + 1)
endfunction

points = [
    0;
    0.6;
    0.9;
]

valuations = [
    f(points(1));
    f(points(2));
    f(points(3));
]

derivatives = [
    ( f(points(1) + 0.00001) - f(points(1)) ) / (0.00001);
    ( f(points(2) + 0.00001) - f(points(2)) ) / (0.00001);
    ( f(points(3) + 0.00001) - f(points(3)) ) / (0.00001);
]

pointsToEvaluate = linspace(-0.1, 1, 30)'

pointsEvaluated1 = HermiteInterpolant(points, valuations, derivatives, pointsToEvaluate)
pointsEvaluated2 = LagrangeInterpolant(points, valuations, pointsToEvaluate)
```

```
plot(points, valuations, "*red")
plot(pointsToEvaluate, pointsEvaluated1, "-blue")
plot(pointsToEvaluate, pointsEvaluated2, "-green")
plot(pointsToEvaluate, f(pointsToEvaluate), "-m")

hl=legend(['Points'; 'Hermite'; 'Lagrange'; 'Real']);
xtitle("Hermite Interpolant", "x", "f(x)");
```

```
// @Author: Rosas Hernandez Oscar Andres
// @Author: Alarcón Alvarez Aylin Yadira Guadalupe
// @Author: Pahua Castro Jesús Miguel Ángel

getd(pwd() + Directory);
clc;


function [x] = f(x)
    x = sqrt(x + 1)
endfunction

points = [
    0;
    0.6;
    0.9;
]

valuations = [
    f(points(1));
    f(points(2));
    f(points(3));
]

derivatives = [
    ( f(points(1) + 0.00001) - f(points(1)) ) / (0.00001);
    ( f(points(2) + 0.00001) - f(points(2)) ) / (0.00001);
    ( f(points(3) + 0.00001) - f(points(3)) ) / (0.00001);
]

pointsToEvaluate = linspace(-0.1, 1, 30)'

pointsEvaluated1 = HermiteInterpolant(points, valuations, derivatives, pointsToEvaluate)
pointsEvaluated2 = LagrangeInterpolant(points, valuations, pointsToEvaluate)

plot(points, valuations, "*red")
plot(pointsToEvaluate, pointsEvaluated1, "-blue")
plot(pointsToEvaluate, pointsEvaluated2, "-green")
plot(pointsToEvaluate, f(pointsToEvaluate), "-m")

hl=legend(['Points'; 'Hermite'; 'Lagrange'; 'Real']);
xtitle("Hermite Interpolant", "x", "f(x)");
```

1.3. 14

Ejecuta los scripts que esta dentro de Code llamado: 14a.sce y 14b.sce

En este código muestra justo lo que se nos pide, eso si, grafico ambos interpolantes de Newton, para que veas que son iguales e incluso hice que dentro del interpolante de Newton nos genere un string con la representación del interpolante, se ve bonito.

```
// @Author: Rosas Hernandez Oscar Andres
// @Author: Alarcón Alvarez Aylin Yadira Guadalupe
// @Author: Pahua Castro Jesús Miguel Ángel

getd(pwd() + Directory);
clc;

points = [
    -0.5;
    -0.25;
    0;
]

valuations = [
    -0.024;
    0.334;
    1.101;
]

pointsToEvaluate = linspace(-1, 1)'

pointsEvaluated = NewtonHomogeneousInterpolant(points, valuations, pointsToEvaluate)
pointsEvaluated2 = NewtonInterpolant(points, valuations, pointsToEvaluate)

plot(points, valuations, "*red")
plot(pointsToEvaluate, pointsEvaluated2, "-green")
plot(pointsToEvaluate, pointsEvaluated, "-blue")

xtitle("Newton Homogeneous Interpolant", "x", "f(x)");
```

```
// @Author: Rosas Hernandez Oscar Andres
// @Author: Alarcón Alvarez Aylin Yadira Guadalupe
// @Author: Pahua Castro Jesús Miguel Ángel

getd(pwd() + Directory);
clc;

points = [
    0.1;
    0.2;
    0.3;
    0.4;
]

valuations = [
    -0.620;
    -0.283;
    0.006;
    0.248;
]

pointsToEvaluate = linspace(-1, 1)'

pointsEvaluated = NewtonHomogeneousInterpolant(points, valuations, pointsToEvaluate)
pointsEvaluated2 = NewtonInterpolant(points, valuations, pointsToEvaluate)

plot(points, valuations, "*red")
plot(pointsToEvaluate, pointsEvaluated2, "-green")
plot(pointsToEvaluate, pointsEvaluated, "-blue")

xtitle("Newton Homogeneous Interpolant", "x", "f(x)");
```

1.4. 15

Ejecuta los scripts que esta dentro de Code llamado: 15.sce

En este código muestra justo lo que se nos pide. Se puso todo en un solo archivo justo para comparar:

Ahora si:

- En $[1,5]$ El mejor tiene que ser el polinomio por lo cerca que esta con respecto a la verdadera función gamma, digo no hay mucho mas que decir, fácil de deducir porque el polinomio tiene un grado mas que el Spline en este Castro
- En $[1,2]$ Tiene mucho mejor rendimiento el Spline, por la misma razón, al solo representar un fragmento lo peude hacer mucho mejor, mientras que el polinomio tiene que ser el mismo para este fragmento y para todo lo demas, por lo que en giros bruscos es mcuho mejor el spline. Como en este caso.

```

getd(pwd() + Directory);
clc;

x = [1; 2; 3; 4; 5;];
y = [1; 1; 2; 6; 24;];

A = [
    1 x(1) x(1)^2 x(1)^3 x(1)^4
    1 x(2) x(2)^2 x(2)^3 x(2)^4
    1 x(3) x(3)^2 x(3)^3 x(3)^4
    1 x(4) x(4)^2 x(4)^3 x(4)^4
    1 x(5) x(5)^2 x(5)^3 x(5)^4
]

Coefficients = inv(A) * y
a0 = Coefficients(1)
a1 = Coefficients(2)
a2 = Coefficients(3)
a3 = Coefficients(4)
a4 = Coefficients(5)

disp(Coefficients)

function [x] = f(x)
    x = a0 + a1*x + a2*x^2 + a3*x^3 + a4*x^4
endfunction

points = linspace(1, 5)';

[estimations] = f(points);
plot(points, estimations, "black-");

disp(f(2))

[estimations] = Spline3Interpolant(x, y, points, 0);
plot(points, estimations, "blue-");
plot(points, gamma(points), "red-");
plot(x, y, "m*");

legend(['Poly'; 'Spline'; 'Real gamma' ;'Points'])

xtitle("Gamma", "x", "y");
```

1.5. 16

- a) En el inciso A hay solo que calcular condicionales y como dice que lo hagamos a través de una libreria lo hice usando la función de scilab. Donde si bien a todas las fue de la patada, tengo que admitir que la versión 4) es la mejor y por mucho.
- c) Pues hace lo que tiene que hacer :v
- d) Ahi notamos la ventaja del Spline, porque en los polinomio en los extremos tienen unas pendientes terriblemente feas. Pero el spline no :)

  No esta extremadamente cerca pero la idea esta bien.
- e) Toma su tiempo, no te preocupes, pero obviamente da el mismo polinomio.
- f) Da obviamente da el mismo polinomio.

```
// @Author: Rosas Hernandez Oscar Andres
// @Author: Alarcón Alvarez Aylin Yadira Guadalupe
// @Author: Pahua Castro Jesús Miguel Ángel

getd(pwd() + Directory);
clc;

x = [
    1900;
    1910;
    1920;
    1930;
    1940;
    1950;
    1960;
    1970;
    1980;
];

y = [
    076212168;
    092228496;
    106021537;
    123202624;
    132164569;
    151325798;
    179323175;
    203302031;
    226542199;
];

function [y] = f1(t, j)
    y = t^(j -1)
endfunction

function [y] = f2(t, j)
    y = (t - 1900)^(j -1)
endfunction

function [y] = f3(t, j)
    y = (t - 1940)^(j -1)
endfunction

function [y] = f4(t, j)
    y = ( (t - 1940) / 40 )^(j -1)
endfunction

A1 = [
    f1(x(1), 1) f1(x(1), 2) f1(x(1), 3) f1(x(1), 4) f1(x(1), 5) f1(x(1), 6) f1(x(1), 7)
     f1(x(1), 8) f1(x(1), 9)
    f1(x(2), 1) f1(x(2), 2) f1(x(2), 3) f1(x(2), 4) f1(x(2), 5) f1(x(2), 6) f1(x(2), 7)
     f1(x(2), 8) f1(x(2), 9)
    f1(x(3), 1) f1(x(3), 2) f1(x(3), 3) f1(x(3), 4) f1(x(3), 5) f1(x(3), 6) f1(x(3), 7)
     f1(x(3), 8) f1(x(3), 9)
    f1(x(4), 1) f1(x(4), 2) f1(x(4), 3) f1(x(4), 4) f1(x(4), 5) f1(x(4), 6) f1(x(4), 7)
     f1(x(4), 8) f1(x(4), 9)
    f1(x(5), 1) f1(x(5), 2) f1(x(5), 3) f1(x(5), 4) f1(x(5), 5) f1(x(5), 6) f1(x(5), 7)
     f1(x(5), 8) f1(x(5), 9)
    f1(x(6), 1) f1(x(6), 2) f1(x(6), 3) f1(x(6), 4) f1(x(6), 5) f1(x(6), 6) f1(x(6), 7)
     f1(x(6), 8) f1(x(6), 9)
    f1(x(7), 1) f1(x(7), 2) f1(x(7), 3) f1(x(7), 4) f1(x(7), 5) f1(x(7), 6) f1(x(7), 7)
     f1(x(7), 8) f1(x(7), 9)
    f1(x(8), 1) f1(x(8), 2) f1(x(8), 3) f1(x(8), 4) f1(x(8), 5) f1(x(8), 6) f1(x(8), 7)
     f1(x(8), 8) f1(x(8), 9)
```

```
    f1(x(9), 1)  f1(x(9), 2)  f1(x(9), 3)  f1(x(9), 4)  f1(x(9), 5)  f1(x(9), 6)  f1(x(9), 7)  f1(x(9), 8)  f1(x(9), 9)
]


A2 = [
    f2(x(1), 1)  f2(x(1), 2)  f2(x(1), 3)  f2(x(1), 4)  f2(x(1), 5)  f2(x(1), 6)  f2(x(1), 7)  f2(x(1), 8)  f2(x(1), 9)
    f2(x(2), 1)  f2(x(2), 2)  f2(x(2), 3)  f2(x(2), 4)  f2(x(2), 5)  f2(x(2), 6)  f2(x(2), 7)  f2(x(2), 8)  f2(x(2), 9)
    f2(x(3), 1)  f2(x(3), 2)  f2(x(3), 3)  f2(x(3), 4)  f2(x(3), 5)  f2(x(3), 6)  f2(x(3), 7)  f2(x(3), 8)  f2(x(3), 9)
    f2(x(4), 1)  f2(x(4), 2)  f2(x(4), 3)  f2(x(4), 4)  f2(x(4), 5)  f2(x(4), 6)  f2(x(4), 7)  f2(x(4), 8)  f2(x(4), 9)
    f2(x(5), 1)  f2(x(5), 2)  f2(x(5), 3)  f2(x(5), 4)  f2(x(5), 5)  f2(x(5), 6)  f2(x(5), 7)  f2(x(5), 8)  f2(x(5), 9)
    f2(x(6), 1)  f2(x(6), 2)  f2(x(6), 3)  f2(x(6), 4)  f2(x(6), 5)  f2(x(6), 6)  f2(x(6), 7)  f2(x(6), 8)  f2(x(6), 9)
    f2(x(7), 1)  f2(x(7), 2)  f2(x(7), 3)  f2(x(7), 4)  f2(x(7), 5)  f2(x(7), 6)  f2(x(7), 7)  f2(x(7), 8)  f2(x(7), 9)
    f2(x(8), 1)  f2(x(8), 2)  f2(x(8), 3)  f2(x(8), 4)  f2(x(8), 5)  f2(x(8), 6)  f2(x(8), 7)  f2(x(8), 8)  f2(x(8), 9)
    f2(x(9), 1)  f2(x(9), 2)  f2(x(9), 3)  f2(x(9), 4)  f2(x(9), 5)  f2(x(9), 6)  f2(x(9), 7)  f2(x(9), 8)  f2(x(9), 9)
]


A3 = [
    f3(x(1), 1)  f3(x(1), 2)  f3(x(1), 3)  f3(x(1), 4)  f3(x(1), 5)  f3(x(1), 6)  f3(x(1), 7)  f3(x(1), 8)  f3(x(1), 9)
    f3(x(2), 1)  f3(x(2), 2)  f3(x(2), 3)  f3(x(2), 4)  f3(x(2), 5)  f3(x(2), 6)  f3(x(2), 7)  f3(x(2), 8)  f3(x(2), 9)
    f3(x(3), 1)  f3(x(3), 2)  f3(x(3), 3)  f3(x(3), 4)  f3(x(3), 5)  f3(x(3), 6)  f3(x(3), 7)  f3(x(3), 8)  f3(x(3), 9)
    f3(x(4), 1)  f3(x(4), 2)  f3(x(4), 3)  f3(x(4), 4)  f3(x(4), 5)  f3(x(4), 6)  f3(x(4), 7)  f3(x(4), 8)  f3(x(4), 9)
    f3(x(5), 1)  f3(x(5), 2)  f3(x(5), 3)  f3(x(5), 4)  f3(x(5), 5)  f3(x(5), 6)  f3(x(5), 7)  f3(x(5), 8)  f3(x(5), 9)
    f3(x(6), 1)  f3(x(6), 2)  f3(x(6), 3)  f3(x(6), 4)  f3(x(6), 5)  f3(x(6), 6)  f3(x(6), 7)  f3(x(6), 8)  f3(x(6), 9)
    f3(x(7), 1)  f3(x(7), 2)  f3(x(7), 3)  f3(x(7), 4)  f3(x(7), 5)  f3(x(7), 6)  f3(x(7), 7)  f3(x(7), 8)  f3(x(7), 9)
    f3(x(8), 1)  f3(x(8), 2)  f3(x(8), 3)  f3(x(8), 4)  f3(x(8), 5)  f3(x(8), 6)  f3(x(8), 7)  f3(x(8), 8)  f3(x(8), 9)
    f3(x(9), 1)  f3(x(9), 2)  f3(x(9), 3)  f3(x(9), 4)  f3(x(9), 5)  f3(x(9), 6)  f3(x(9), 7)  f3(x(9), 8)  f3(x(9), 9)
]


A4 = [
    f4(x(1), 1)  f4(x(1), 2)  f4(x(1), 3)  f4(x(1), 4)  f4(x(1), 5)  f4(x(1), 6)  f4(x(1), 7)  f4(x(1), 8)  f4(x(1), 9)
    f4(x(2), 1)  f4(x(2), 2)  f4(x(2), 3)  f4(x(2), 4)  f4(x(2), 5)  f4(x(2), 6)  f4(x(2), 7)  f4(x(2), 8)  f4(x(2), 9)
    f4(x(3), 1)  f4(x(3), 2)  f4(x(3), 3)  f4(x(3), 4)  f4(x(3), 5)  f4(x(3), 6)  f4(x(3), 7)  f4(x(3), 8)  f4(x(3), 9)
    f4(x(4), 1)  f4(x(4), 2)  f4(x(4), 3)  f4(x(4), 4)  f4(x(4), 5)  f4(x(4), 6)  f4(x(4), 7)  f4(x(4), 8)  f4(x(4), 9)
    f4(x(5), 1)  f4(x(5), 2)  f4(x(5), 3)  f4(x(5), 4)  f4(x(5), 5)  f4(x(5), 6)  f4(x(5), 7)  f4(x(5), 8)  f4(x(5), 9)
    f4(x(6), 1)  f4(x(6), 2)  f4(x(6), 3)  f4(x(6), 4)  f4(x(6), 5)  f4(x(6), 6)  f4(x(6), 7)  f4(x(6), 8)  f4(x(6), 9)
    f4(x(7), 1)  f4(x(7), 2)  f4(x(7), 3)  f4(x(7), 4)  f4(x(7), 5)  f4(x(7), 6)  f4(x(7), 7)  f4(x(7), 8)  f4(x(7), 9)
    f4(x(8), 1)  f4(x(8), 2)  f4(x(8), 3)  f4(x(8), 4)  f4(x(8), 5)  f4(x(8), 6)  f4(x(8), 7)  f4(x(8), 8)  f4(x(8), 9)
    f4(x(9), 1)  f4(x(9), 2)  f4(x(9), 3)  f4(x(9), 4)  f4(x(9), 5)  f4(x(9), 6)  f4(x(9), 7)  f4(x(9), 8)  f4(x(9), 9)
]

disp(cond(A1))
disp(cond(A2))
disp(cond(A3))
disp(cond(A4))

Coefficients = inv(A4) * y
a0 = Coefficients(1)
a1 = Coefficients(2)
a2 = Coefficients(3)
a3 = Coefficients(4)
a4 = Coefficients(5)
a5 = Coefficients(6)
a6 = Coefficients(7)
a7 = Coefficients(8)
a8 = Coefficients(9)

disp(Coefficients)

```

```
function [x] = f(x)
    x = a0*f4(x, 1) + a1*f4(x, 2) + a2*f4(x, 3) + a3*f4(x, 4) + a4*f4(x, 5) + a5*f4(x, 6) + a6*f4(x, 7) + a7*f4(x, 8) + + a8*f4(x, 9)
endfunction



points = linspace(1900, 1980)';

[estimations] = f(points);
plot(points, estimations, "black-");
plot(x, y, "m*");


legend(['Poly' ;'Points'])

xtitle("Population data", "year", "population");
```

```

getd(pwd() + Directory);
clc;

x = [
    1900;
    1910;
    1920;
    1930;
    1940;
    1950;
    1960;
    1970;
    1980;
];

y = [
    076212168;
    092228496;
    106021537;
    123202624;
    132164569;
    151325798;
    179323175;
    203302031;
    226542199;
];

function [y] = f4(t, j)
    y = ( (t - 1940) / 40 )^(j -1)
endfunction

A4 = [
    f4(x(1), 1) f4(x(1), 2) f4(x(1), 3) f4(x(1), 4) f4(x(1), 5) f4(x(1), 6) f4(x(1), 7) f4(x(1), 8) f4(x(1), 9)
    f4(x(2), 1) f4(x(2), 2) f4(x(2), 3) f4(x(2), 4) f4(x(2), 5) f4(x(2), 6) f4(x(2), 7) f4(x(2), 8) f4(x(2), 9)
    f4(x(3), 1) f4(x(3), 2) f4(x(3), 3) f4(x(3), 4) f4(x(3), 5) f4(x(3), 6) f4(x(3), 7) f4(x(3), 8) f4(x(3), 9)
    f4(x(4), 1) f4(x(4), 2) f4(x(4), 3) f4(x(4), 4) f4(x(4), 5) f4(x(4), 6) f4(x(4), 7) f4(x(4), 8) f4(x(4), 9)
    f4(x(5), 1) f4(x(5), 2) f4(x(5), 3) f4(x(5), 4) f4(x(5), 5) f4(x(5), 6) f4(x(5), 7) f4(x(5), 8) f4(x(5), 9)
    f4(x(6), 1) f4(x(6), 2) f4(x(6), 3) f4(x(6), 4) f4(x(6), 5) f4(x(6), 6) f4(x(6), 7) f4(x(6), 8) f4(x(6), 9)
    f4(x(7), 1) f4(x(7), 2) f4(x(7), 3) f4(x(7), 4) f4(x(7), 5) f4(x(7), 6) f4(x(7), 7) f4(x(7), 8) f4(x(7), 9)
    f4(x(8), 1) f4(x(8), 2) f4(x(8), 3) f4(x(8), 4) f4(x(8), 5) f4(x(8), 6) f4(x(8), 7) f4(x(8), 8) f4(x(8), 9)
    f4(x(9), 1) f4(x(9), 2) f4(x(9), 3) f4(x(9), 4) f4(x(9), 5) f4(x(9), 6) f4(x(9), 7) f4(x(9), 8) f4(x(9), 9)
]

disp(cond(A1))
disp(cond(A2))
disp(cond(A3))
disp(cond(A4))
// @Author: Rosas Hernandez Oscar Andres
// @Author: Alarcón Alvarez Aylin Yadira Guadalupe
// @Author: Pahua Castro Jesús Miguel Ángel

Coefficients = inv(A4) * y
a0 = Coefficients(1)
a1 = Coefficients(2)
a2 = Coefficients(3)
a3 = Coefficients(4)
a4 = Coefficients(5)
a5 = Coefficients(6)
```

```
a6 = Coefficients(7)
a7 = Coefficients(8)
a8 = Coefficients(9)

disp(Coefficients)

function [x] = f(x)
    x = a0*f4(x, 1) + a1*f4(x, 2) + a2*f4(x, 3) + a3*f4(x, 4) + a4*f4(x, 5) + a5*f4(x, 6) + a6*f4(x, 7) + a7*f4(x, 8) + + a8*f4(x, 9)
endfunction

points = linspace(1900, 1980)';

[estimations] = f(points);
plot(points, estimations, "black-");
plot(x, y, "m*");

[estimations2] = Spline3Interpolant(x, y, points, 0);
plot(points, estimations2, "m-");

legend(['Poly' ;'Points'; 'Spline'])

xtitle("Population data", "year", "population");
```

```
// @Author: Rosas Hernandez Oscar Andres
// @Author: Alarcón Alvarez Aylin Yadira Guadalupe
// @Author: Pahua Castro Jesús Miguel Ángel

getd(pwd() + Directory);
clc;

x = [
    1900;
    1910;
    1920;
    1930;
    1940;
    1950;
    1960;
    1970;
    1980;
];

y = [
    076212168;
    092228496;
    106021537;
    123202624;
    132164569;
    151325798;
    179323175;
    203302031;
    226542199;
];

function [y] = f4(t, j)
    y = ( (t - 1940) / 40 )^(j -1)
endfunction

A4 = [
    f4(x(1), 1)  f4(x(1), 2)  f4(x(1), 3)  f4(x(1), 4)  f4(x(1), 5)  f4(x(1), 6)  f4(x(1), 7)  f4(x(1), 8)  f4(x(1), 9)
    f4(x(2), 1)  f4(x(2), 2)  f4(x(2), 3)  f4(x(2), 4)  f4(x(2), 5)  f4(x(2), 6)  f4(x(2), 7)  f4(x(2), 8)  f4(x(2), 9)
    f4(x(3), 1)  f4(x(3), 2)  f4(x(3), 3)  f4(x(3), 4)  f4(x(3), 5)  f4(x(3), 6)  f4(x(3), 7)  f4(x(3), 8)  f4(x(3), 9)
    f4(x(4), 1)  f4(x(4), 2)  f4(x(4), 3)  f4(x(4), 4)  f4(x(4), 5)  f4(x(4), 6)  f4(x(4), 7)  f4(x(4), 8)  f4(x(4), 9)
    f4(x(5), 1)  f4(x(5), 2)  f4(x(5), 3)  f4(x(5), 4)  f4(x(5), 5)  f4(x(5), 6)  f4(x(5), 7)  f4(x(5), 8)  f4(x(5), 9)
    f4(x(6), 1)  f4(x(6), 2)  f4(x(6), 3)  f4(x(6), 4)  f4(x(6), 5)  f4(x(6), 6)  f4(x(6), 7)  f4(x(6), 8)  f4(x(6), 9)
    f4(x(7), 1)  f4(x(7), 2)  f4(x(7), 3)  f4(x(7), 4)  f4(x(7), 5)  f4(x(7), 6)  f4(x(7), 7)  f4(x(7), 8)  f4(x(7), 9)
    f4(x(8), 1)  f4(x(8), 2)  f4(x(8), 3)  f4(x(8), 4)  f4(x(8), 5)  f4(x(8), 6)  f4(x(8), 7)  f4(x(8), 8)  f4(x(8), 9)
    f4(x(9), 1)  f4(x(9), 2)  f4(x(9), 3)  f4(x(9), 4)  f4(x(9), 5)  f4(x(9), 6)  f4(x(9), 7)  f4(x(9), 8)  f4(x(9), 9)
]

disp(cond(A1))
disp(cond(A2))
disp(cond(A3))
disp(cond(A4))

Coefficients = inv(A4) * y
```

```
a0 = Coefficients(1)
a1 = Coefficients(2)
a2 = Coefficients(3)
a3 = Coefficients(4)
a4 = Coefficients(5)
a5 = Coefficients(6)
a6 = Coefficients(7)
a7 = Coefficients(8)
a8 = Coefficients(9)

disp(Coefficients)

function [x] = f(x)
    x = a0*f4(x, 1) + a1*f4(x, 2) + a2*f4(x, 3) + a3*f4(x, 4) + a4*f4(x, 5) + a5*f4(x, 6) + a6*f4(x, 7) + a7*f4(x, 8) + + a8*f4(x, 9)
endfunction

points = linspace(1900, 1990)';

[estimations] = f(points);
plot(points, estimations, "black-");
plot(x, y, "m*");

[estimations2] = Spline3Interpolant(x, y, points, 0);
plot(points, estimations2, "m-");

legend(['Poly' ;'Points'; 'Spline'])

xtitle("Population data", "year", "population");

disp("Poly:" + string(estimations(100)))
disp("Spline:" + string(estimations2(100)))
```

```
// @Author: Rosas Hernandez Oscar Andres
// @Author: Alarcón Alvarez Aylin Yadira Guadalupe
// @Author: Pahua Castro Jesús Miguel Ángel

getd(pwd() + Directory);
clc;

x = [
    1900;
    1910;
    1920;
    1930;
    1940;
    1950;
    1960;
    1970;
    1980;
];

y = [
    076212168;
    092228496;
    106021537;
    123202624;
    132164569;
    151325798;
    179323175;
    203302031;
    226542199;
];

function [y] = f4(t, j)
    y = ( (t - 1940) / 40 )^(j -1)
endfunction

A4 = [
    f4(x(1), 1)  f4(x(1), 2)  f4(x(1), 3)  f4(x(1), 4)  f4(x(1), 5)  f4(x(1), 6)  f4(x(1), 7)  f4(x(1), 8)  f4(x(1), 9)
    f4(x(2), 1)  f4(x(2), 2)  f4(x(2), 3)  f4(x(2), 4)  f4(x(2), 5)  f4(x(2), 6)  f4(x(2), 7)  f4(x(2), 8)  f4(x(2), 9)
    f4(x(3), 1)  f4(x(3), 2)  f4(x(3), 3)  f4(x(3), 4)  f4(x(3), 5)  f4(x(3), 6)  f4(x(3), 7)  f4(x(3), 8)  f4(x(3), 9)
    f4(x(4), 1)  f4(x(4), 2)  f4(x(4), 3)  f4(x(4), 4)  f4(x(4), 5)  f4(x(4), 6)  f4(x(4), 7)  f4(x(4), 8)  f4(x(4), 9)
    f4(x(5), 1)  f4(x(5), 2)  f4(x(5), 3)  f4(x(5), 4)  f4(x(5), 5)  f4(x(5), 6)  f4(x(5), 7)  f4(x(5), 8)  f4(x(5), 9)
    f4(x(6), 1)  f4(x(6), 2)  f4(x(6), 3)  f4(x(6), 4)  f4(x(6), 5)  f4(x(6), 6)  f4(x(6), 7)  f4(x(6), 8)  f4(x(6), 9)
    f4(x(7), 1)  f4(x(7), 2)  f4(x(7), 3)  f4(x(7), 4)  f4(x(7), 5)  f4(x(7), 6)  f4(x(7), 7)  f4(x(7), 8)  f4(x(7), 9)
    f4(x(8), 1)  f4(x(8), 2)  f4(x(8), 3)  f4(x(8), 4)  f4(x(8), 5)  f4(x(8), 6)  f4(x(8), 7)  f4(x(8), 8)  f4(x(8), 9)
    f4(x(9), 1)  f4(x(9), 2)  f4(x(9), 3)  f4(x(9), 4)  f4(x(9), 5)  f4(x(9), 6)  f4(x(9), 7)
```

```
    f4(x(9), 8)  f4(x(9), 9)
]

disp(cond(A1))
disp(cond(A2))
disp(cond(A3))
disp(cond(A4))

Coefficients = inv(A4) * y
a0 = Coefficients(1)
a1 = Coefficients(2)
a2 = Coefficients(3)
a3 = Coefficients(4)
a4 = Coefficients(5)
a5 = Coefficients(6)
a6 = Coefficients(7)
a7 = Coefficients(8)
a8 = Coefficients(9)

disp(Coefficients)

function [x] = f(x)
    x = a0*f4(x, 1) + a1*f4(x, 2) + a2*f4(x, 3) + a3*f4(x, 4) + a4*f4(x, 5) + a5*f4(x, 6) +
     a6*f4(x, 7) + a7*f4(x, 8) + + a8*f4(x, 9)
endfunction

points = linspace(1900, 1990)';

[estimations] = f(points);
plot(points, estimations, "black-");
plot(x, y, "m*");

[estimations2] = LagrangeInterpolant(x, y, points);
plot(points, estimations2, "m-");

legend(['Poly' ;'Points'; 'Lagrange'])

xtitle("Population data", "year", "population");
```

```
// @Author: Rosas Hernandez Oscar Andres
// @Author: Alarcón Alvarez Aylin Yadira Guadalupe
// @Author: Pahua Castro Jesús Miguel Ángel

getd(pwd() + Directory);
clc;

x = [
    1900;
    1910;
    1920;
    1930;
    1940;
    1950;
    1960;
    1970;
    1980;
];

y = [
    076212168;
    092228496;
    106021537;
    123202624;
    132164569;
    151325798;
    179323175;
    203302031;
    226542199;
];

function [y] = f4(t, j)
    y = ( (t - 1940) / 40 )^(j -1)
endfunction

A4 = [
    f4(x(1), 1)  f4(x(1), 2)  f4(x(1), 3)  f4(x(1), 4)  f4(x(1), 5)  f4(x(1), 6)  f4(x(1), 7)
     f4(x(1), 8)  f4(x(1), 9)
    f4(x(2), 1)  f4(x(2), 2)  f4(x(2), 3)  f4(x(2), 4)  f4(x(2), 5)  f4(x(2), 6)  f4(x(2), 7)
     f4(x(2), 8)  f4(x(2), 9)
    f4(x(3), 1)  f4(x(3), 2)  f4(x(3), 3)  f4(x(3), 4)  f4(x(3), 5)  f4(x(3), 6)  f4(x(3), 7)
     f4(x(3), 8)  f4(x(3), 9)
    f4(x(4), 1)  f4(x(4), 2)  f4(x(4), 3)  f4(x(4), 4)  f4(x(4), 5)  f4(x(4), 6)  f4(x(4), 7)
     f4(x(4), 8)  f4(x(4), 9)
    f4(x(5), 1)  f4(x(5), 2)  f4(x(5), 3)  f4(x(5), 4)  f4(x(5), 5)  f4(x(5), 6)  f4(x(5), 7)
     f4(x(5), 8)  f4(x(5), 9)
    f4(x(6), 1)  f4(x(6), 2)  f4(x(6), 3)  f4(x(6), 4)  f4(x(6), 5)  f4(x(6), 6)  f4(x(6), 7)
```

```
    f4(x(6), 8)  f4(x(6), 9)
    f4(x(7), 1)  f4(x(7), 2)  f4(x(7), 3)  f4(x(7), 4)  f4(x(7), 5)  f4(x(7), 6)  f4(x(7), 7)  f4(x(7), 8)  f4(x(7), 9)
    f4(x(8), 1)  f4(x(8), 2)  f4(x(8), 3)  f4(x(8), 4)  f4(x(8), 5)  f4(x(8), 6)  f4(x(8), 7)  f4(x(8), 8)  f4(x(8), 9)
    f4(x(9), 1)  f4(x(9), 2)  f4(x(9), 3)  f4(x(9), 4)  f4(x(9), 5)  f4(x(9), 6)  f4(x(9), 7)  f4(x(9), 8)  f4(x(9), 9)
]

disp(cond(A1))
disp(cond(A2))
disp(cond(A3))
disp(cond(A4))

Coefficients = inv(A4) * y
a0 = Coefficients(1)
a1 = Coefficients(2)
a2 = Coefficients(3)
a3 = Coefficients(4)
a4 = Coefficients(5)
a5 = Coefficients(6)
a6 = Coefficients(7)
a7 = Coefficients(8)
a8 = Coefficients(9)

disp(Coefficients)

function [x] = f(x)
    x = a0*f4(x, 1) + a1*f4(x, 2) + a2*f4(x, 3) + a3*f4(x, 4) + a4*f4(x, 5) + a5*f4(x, 6) + a6*f4(x, 7) + a7*f4(x, 8) + + a8*f4(x, 9)
endfunction

points = linspace(1900, 1990)';

[estimations] = f(points);
plot(points, estimations, "black-");
plot(x, y, "m*");

[estimations2] = NewtonInterpolant(x, y, points);
plot(points, estimations2, "m--");

legend(['Poly' ;'Points'; 'Newton'])

xtitle("Population data", "year", "population");
```

1.6. 17

Ejecuta los scripts que esta dentro de Code llamado: 17.sce

En este código muestra justo lo que se nos pide, eso si, el snoppy se ve mas bonito se mueves la grafica, haciendola mas horizontal

```
// @Author: Rosas Hernandez Oscar Andres
// @Author: Alarcón Alvarez Aylin Yadira Guadalupe
// @Author: Pahua Castro Jesús Miguel Ángel

getd(pwd() + Directory);
clc;

x1 = [
     1;
     2;
     5;
     6;
     7;
     8;
     10;
     13;
     17;
];

y1 = [
     3.0;
     3.7;
     3.9;
     4.2;
     5.7;
     6.6;
     7.1;
     6.7;
     4.5;
];

x2 = [
     17;
     20;
     23;
     24;
     25;
     27;
     27.7;
];

y2 = [
     4.5;
     7.0;
     6.1;
     5.6;
     5.8;
     5.2;
     4.1;
];

x3 = [
     27.7;
     28;
     29;
     30;
];

y3 = [
     4.1;
     4.3;
     4.1;
     3.0;
];

points1 = linspace(1, 17)';
plot(0, 0, "*red")

[estimations1] = Spline3Interpolant(x1, y1, points1, [-1; +0.67]);
plot(points1, estimations1, "blue-");
plot(x1, y1, "m*");

points2 = linspace(17, 27.7)';

[estimations2] = Spline3Interpolant(x2, y2, points2, [-3; +4]);
plot(points2, estimations2, "red-");
plot(x2, y2, "m*");
```

```
points3 = linspace(27.7, 30, 20)';

[estimations3] = Spline3Interpolant(x3, y3, points3, [-0.33; 1.5]);
plot(points3, estimations3, "m-");
plot(x3, y3, "m*");

xtitle("Estimate a Snoppy", "x", "y");
```

1.7. 18

Ejecuta los scripts que esta dentro de Code llamado: 18.sce

En este código muestra justo lo que se nos pide, eso si, el snoppy se ve mas bonito se mueves la grafica, haciendola mas horizontal y debo de admitir que con este spline me gusto mas como se ve

```
// @Author: Rosas Hernandez Oscar Andres
// @Author: Alarcón Alvarez Aylin Yadira Guadalupe
// @Author: Pahua Castro Jesús Miguel Ángel

getd(pwd() + Directory);
clc;

x1 = [
    1;
    2;
    5;
    6;
    7;
    8;
    10;
    13;
    17;
];

y1 = [
    3.0;
    3.7;
    3.9;
    4.2;
    5.7;
    6.6;
    7.1;
    6.7;
    4.5;
];

x2 = [
    17;
    20;
    23;
    24;
    25;
    27;
    27.7;
];

y2 = [
    4.5;
    7.0;
    6.1;
    5.6;
    5.8;
    5.2;
    4.1;
];

x3 = [
    27.7;
    28;
    29;
    30;
];

y3 = [
    4.1;
    4.3;
    4.1;
    3.0;
];

points1 = linspace(1, 17)';
plot(0, 0, "*red")

[estimations1] = Spline3Interpolant(x1, y1, points1, 0);
plot(points1, estimations1, "blue-");
plot(x1, y1, "m*");

points2 = linspace(17, 27.7)';

[estimations2] = Spline3Interpolant(x2, y2, points2, 0);
plot(points2, estimations2, "red-");
plot(x2, y2, "m*");
```

```
points3 = linspace(27.7, 30, 20)';

[estimations3] = Spline3Interpolant(x3, y3, points3, 0);
plot(points3, estimations3, "m-");
plot(x3, y3, "m*");

xtitle("Estimate a Snoppy", "x", "y");
```

2. Anexo

2.1. HermiteInterpolant

```
// Function to aproximate a function using the Lagrange method
// @param: points a vector of points
// @param: valuations a vector such valuations(i) = f(points(i))
// @param: pointsToEvaluate a vector of points to evaluate
// @return: pointsEvaluated such pointsEvaluated(i) = f(pointsToEvaluate(i))

// @Author: Rosas Hernandez Oscar Andres
// @Author: Alarcón Alvarez Aylin Yadira Guadalupe
// @Author: Pahua Castro Jesús Miguel Ángel

function [pointsEvaluated] = HermiteInterpolant(points, valuations, derivatives, pointsToEvaluate)
    n = length(points)

    numberOfEvaluations = length(pointsToEvaluate)
    pointsEvaluated = zeros(numberOfEvaluations, 1)

    HermiteEvaluations = zeros(numberOfEvaluations, n)
    HermiteEvaluationsHat = zeros(numberOfEvaluations, n)

    for evaluation = (1 : numberOfEvaluations)
        for i = (1 : n)
            HermiteEvaluationsHat(:, i) = HermiteHatPolynomial(points, i, pointsToEvaluate)
            HermiteEvaluations(:, i) = HermitePolynomial(points, i, pointsToEvaluate)
        end
    end

    for evaluation = (1 : numberOfEvaluations)
        temporal = 0
        for i = (1 : n)
            temporal = temporal + valuations(i) * HermiteEvaluations(evaluation, i)
            temporal = temporal + derivatives(i) * HermiteEvaluationsHat(evaluation, i)
        end
        pointsEvaluated(evaluation) = temporal
    end
endfunction

function [pointsEvaluated] = HermitePolynomial(points, j, pointsToEvaluate)
    numberOfEvaluations = length(pointsToEvaluate)
    pointsEvaluated = zeros(numberOfEvaluations, 1)

    for evaluation = (1 : numberOfEvaluations)
        temporal = (LagrangePolynomial(points, j, points(j) + 0.0001) - LagrangePolynomial(points, j, points(j))) / (0.0001)
        pointsEvaluated(evaluation) = 1 - 2 * (pointsToEvaluate(evaluation) - points(j)) * temporal
        pointsEvaluated(evaluation) = pointsEvaluated(evaluation) * LagrangePolynomial(points, j, pointsToEvaluate(evaluation))^2
    end
endfunction

function [pointsEvaluated] = HermiteHatPolynomial(points, j, pointsToEvaluate)
    numberOfEvaluations = length(pointsToEvaluate)
    pointsEvaluated = zeros(numberOfEvaluations, 1)

    numberOfEvaluations = length(pointsToEvaluate)
    for evaluation = (1 : numberOfEvaluations)
        pointsEvaluated(evaluation) = (pointsToEvaluate(evaluation) - points(j)) * LagrangePolynomial(points, j, pointsToEvaluate(evaluation))^2
    end
endfunction
```

2.2. LagrangeInterpolant

```
// Function to aproximate a function using the Lagrange method
// @param: points a vector of points
// @param: valuations a vector such valuations(i) = f(points(i))
// @param: pointsToEvaluate a vector of points to evaluate
// @return: pointsEvaluated such pointsEvaluated(i) = f(pointsToEvaluate(i))

// @Author: Rosas Hernandez Oscar Andres
// @Author: Alarcón Alvarez Aylin Yadira Guadalupe
// @Author: Pahua Castro Jesús Miguel Ángel

function [pointsEvaluated] = LagrangeInterpolant(points, valuations, pointsToEvaluate)
    n = length(points)

    numberOfEvaluations = length(pointsToEvaluate)
    pointsEvaluated = zeros(numberOfEvaluations, 1)

    LangrangeEvaluations = zeros(numberOfEvaluations, n)

    for evaluation = (1 : numberOfEvaluations)
        for i = (1 : n)
            LangrangeEvaluations(:, i) = LagrangePolynomial(points, i, pointsToEvaluate)
        end
    end

    for evaluation = (1 : numberOfEvaluations)
        temporal = 0
        for i = (1 : n)
            temporal = temporal + valuations(i) * LangrangeEvaluations(evaluation, i)
        end
        pointsEvaluated(evaluation) = temporal
    end
endfunction

function [pointsEvaluated] = LagrangePolynomial(points, k, pointsToEvaluate)
    n = length(points)

    numberOfEvaluations = length(pointsToEvaluate)
    for evaluation = (1 : numberOfEvaluations)
        temporal = 1
        for i = (1 : n)
            if (i ~= k)
                temporal = temporal * (pointsToEvaluate(evaluation) - points(i)) / (points(k) -
 points(i))
            end
        end
        pointsEvaluated(evaluation) = temporal
    end

endfunction
```

2.3. NewtonInterpolant

```
// Function to aproximate a function using the Newton method
// @param: points a vector of points
// @param: valuations a vector such valuations(i) = f(points(i))
// @param: pointsToEvaluate a vector of points to evaluate
// @return: pointsEvaluated such pointsEvaluated(i) = f(pointsToEvaluate(i))

// @Author: Rosas Hernandez Oscar Andres
// @Author: Alarcón Alvarez Aylin Yadira Guadalupe
// @Author: Pahua Castro Jesús Miguel Ángel

function [pointsEvaluated] = NewtonInterpolant(points, valuations, pointsToEvaluate)
    n = length(points)
    Differences = NewtonInterpolantCoefficients(points, valuations)

    data = "I(x) = " + string(Differences(1))
    for i = (2 : n)
        data = data + " + " + string(Differences(i))
        for j = (1 : i - 1)
            if (points(j) > 0)
                data = data + "(x - " + string(points(j)) + ")"
            else
                data = data + "(x + " + string(-1 * points(j)) + ")"
            end
        end
    end

    disp(data)

    numberOfEvaluations = length(pointsToEvaluate)
    pointsEvaluated = zeros(numberOfEvaluations, 1)

    for evaluation = (1 : numberOfEvaluations)
        temporal = Differences(n)
        for j = (n - 1 : -1 : 1)
            temporal = temporal * ( pointsToEvaluate(evaluation) - points(j) ) + Differences(j)
        end
        pointsEvaluated(evaluation) = temporal
    end
endfunction

// Function to aproximate a function using the Newton method but the points should be
// from a homogenues partition
// @param: points a vector of points
// @param: valuations a vector such valuations(i) = f(points(i))
// @param: pointsToEvaluate a vector of points to evaluate
// @return: pointsEvaluated such pointsEvaluated(i) = f(pointsToEvaluate(i))

function [pointsEvaluated] = NewtonHomogeneousInterpolant(points, valuations, pointsToEvaluate)
    n = length(points)
    Differences = NewtonInterpolantCoefficients(points, valuations)

    numberOfEvaluations = length(pointsToEvaluate)
    pointsEvaluated = zeros(numberOfEvaluations, 1)
    h = points(2) - points(1)

    for evaluation = (1 : numberOfEvaluations)
        s = ( pointsToEvaluate(evaluation) - points(1) ) / h

        temporal = Differences(1)
        for k = (2 : n)
            temporal2 = Differences(k) * h^(k-1)
            for term = (0 : k - 2)
                temporal2 = temporal2 * (s - term)
            end
            temporal = temporal + temporal2
        end
        pointsEvaluated(evaluation) = temporal
    end
endfunction


// Function get all the Divided Differences for the Newton interpolant
// @param: points a vector of points
// @param: valuations a vector such valuations(i) = f(points(i))
// @return: Differences such Differences(i) = f[x_0 ... x_i]

function [Differences] = NewtonInterpolantCoefficients(points, valuations)
    n = length(points)

    for i = (1 : n)
        Differences(i) = valuations(i)
    end

    for order = (1 : n - 1)
        for i = (n : -1 : order + 1)
```

```
            numerator = Differences(i) - Differences(i-1)
            denominator = points(i) - points(i-order)
            Differences(i) = numerator / denominator
        end
    end

endfunction
```

2.4. Spline3Interpolant

```
// Function to aproximate a function using the Spline (cubic) method
// @param: x a vector of points
// @param: y a vector such valuations(i) = f(x(i))
// @param: pointsToEvaluate a vector of points to evaluate
// @param: type if 0 then Spline natural if [a, b] then spline complete where
//          f(x_0)' = -a and f(x_n)' = -b
// @return: pointsEvaluated such pointsEvaluated(i) = f(pointsToEvaluate(i))

// @Author: Rosas Hernandez Oscar Andres
// @Author: Alarcón Alvarez Aylin Yadira Guadalupe
// @Author: Pahua Castro Jesús Miguel Ángel


function [pointsEvaluated] = Spline3Interpolant(x, y, pointsToEvaluate, type)
    [z, h] = Spline3Coefficients(x, y, type)

    numberOfEvaluations = length(pointsToEvaluate)
    pointsEvaluated = zeros(numberOfEvaluations, 1)

    for evaluation = (1 : numberOfEvaluations)
        point = pointsToEvaluate(evaluation)
        i = FindPoint(point, x)

        sum1 = z(i + 1) / (6 * h(i)) * (point - x(i))^3
        sum2 = z(i) / (6 * h(i)) * (x(i+1) - point)^3
        sum3 = ( y(i+1)/h(i) - z(i+1)/6 * h(i) ) * (point - x(i))
        sum4 = ( y(i)/h(i) - z(i)/6 * h(i) ) * (x(i+1) - point)

        pointsEvaluated(evaluation) = sum1 + sum2 + sum3 + sum4
    end

endfunction


// Function to get the Z and H for the Spline method :v
// @param: t a vector of points
// @param: y a vector such valuations(i) = f(t(i))
// @return: z a vector of ?? well, of number to the next algorithm step
// @return: h a vector of distances

function [z, h] = Spline3Coefficients(t, y, type)
    n = length(t);

    h = zeros(n - 1, 1);
    b = zeros(n - 1, 1);
    u = zeros(n - 2, 1);
    v = zeros(n - 2, 1);
    z = zeros(n, 1);

    for i = (1 : n - 1)
        h(i) =  t(i+1) - t(i);
        b(i) = 6 * (y(i+1) - y(i)) / h(i);
    end

    u(2) = 2 * (h(1) + h(2));
    v(2) = b(2) - b(1);

    for i = (3 : n - 1)
        u(i) = 2 * (h(i) + h(i-1)) - h(i-1)**2 / u(i-1);
        v(i) = b(i) - b(i-1) - h(i-1) * v(i-1) / u(i-1);
    end

    if (length(type) == 1) z(n) = 0; else z(n) = type(2); end

    for i = (n - 1 : -1 : 2)
        z(i) = (v(i) - h(i) * z(i+1)) / u(i);
    end

    if (length(type) == 1) z(1) = 0; else z(1) = type(1); end

endfunction

// Function to find a interval in a distribution
// @param: point a point :v
// @param: data a vector of values
// @return: i a number such data(i) <= point <= data(i+1)

function [i] = FindPoint(point, data)
    middle = 1, start = 1
    final = length(data)

    while (start < final)
        middle = start + floor((final - start) / 2)

        if (point < data(middle)) then final = middle;
```

```
        else start = middle + 1;
        end
    end

    i = start - 1;

endfunction
```