Manuel Delgado Mantilla

1

$$\begin{pmatrix} 2 & 1 & 0 \\ 1 & 3 & -2 \\ 0 & 4 & -1 \end{pmatrix} \times \begin{pmatrix} 3 & 2 \\ 0 & -1 \\ 1 & 0 \end{pmatrix}$$

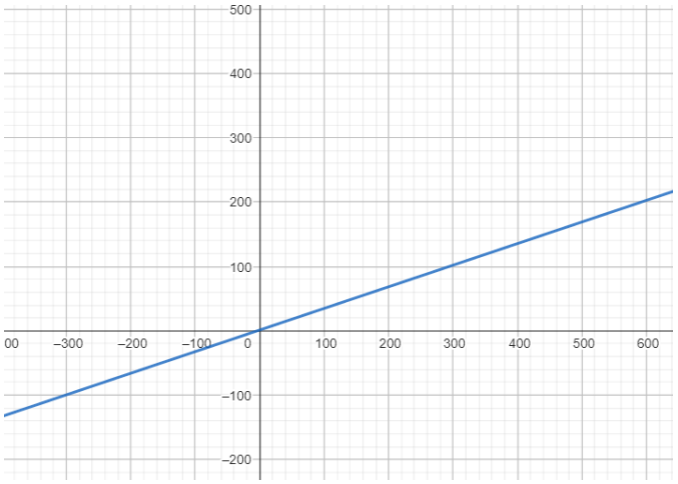
```
run:
Resultado de la multiplicaci□n de matrices:
6 3
1 -1
-1 -4
BUILD SUCCESSFUL (total time: 0 seconds)
```

2

**Aproximación lineal:**

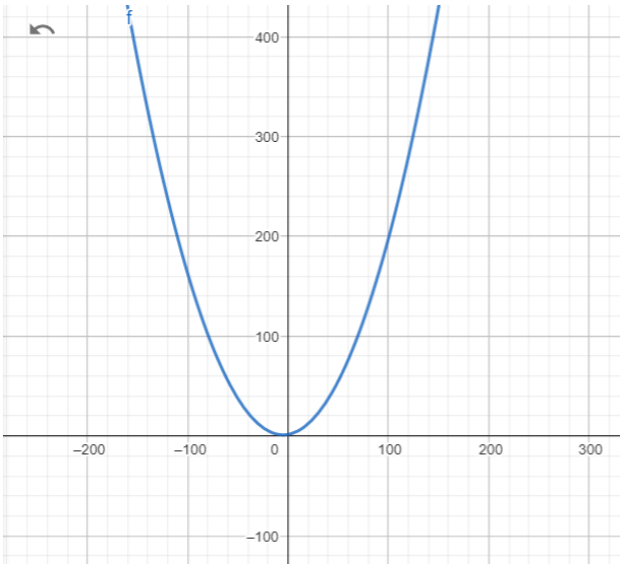
**La mejor**

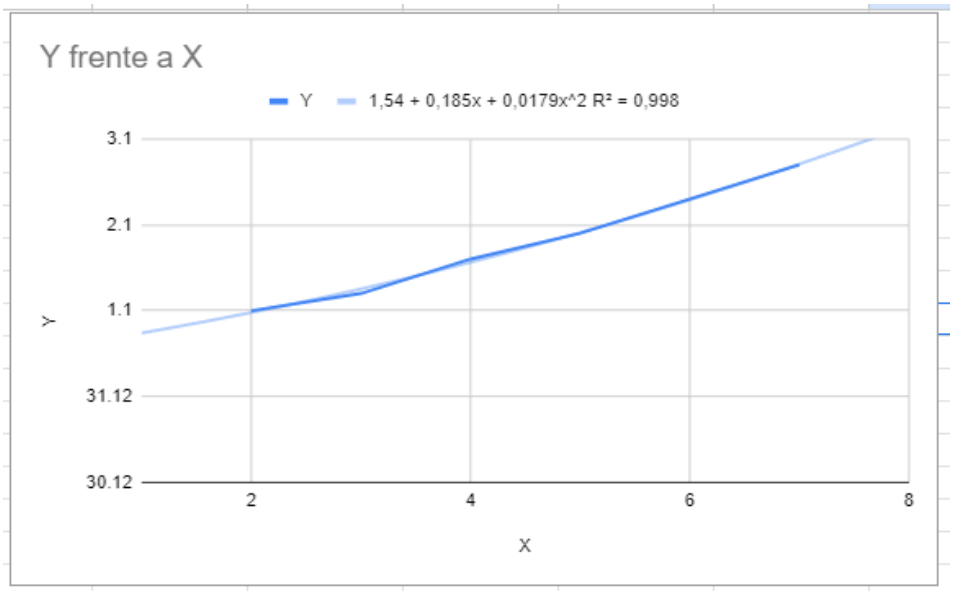
f(x) = 1.3142857142857152 + 0.3357142857142856x



## Aproximación polinómica de grado 2:

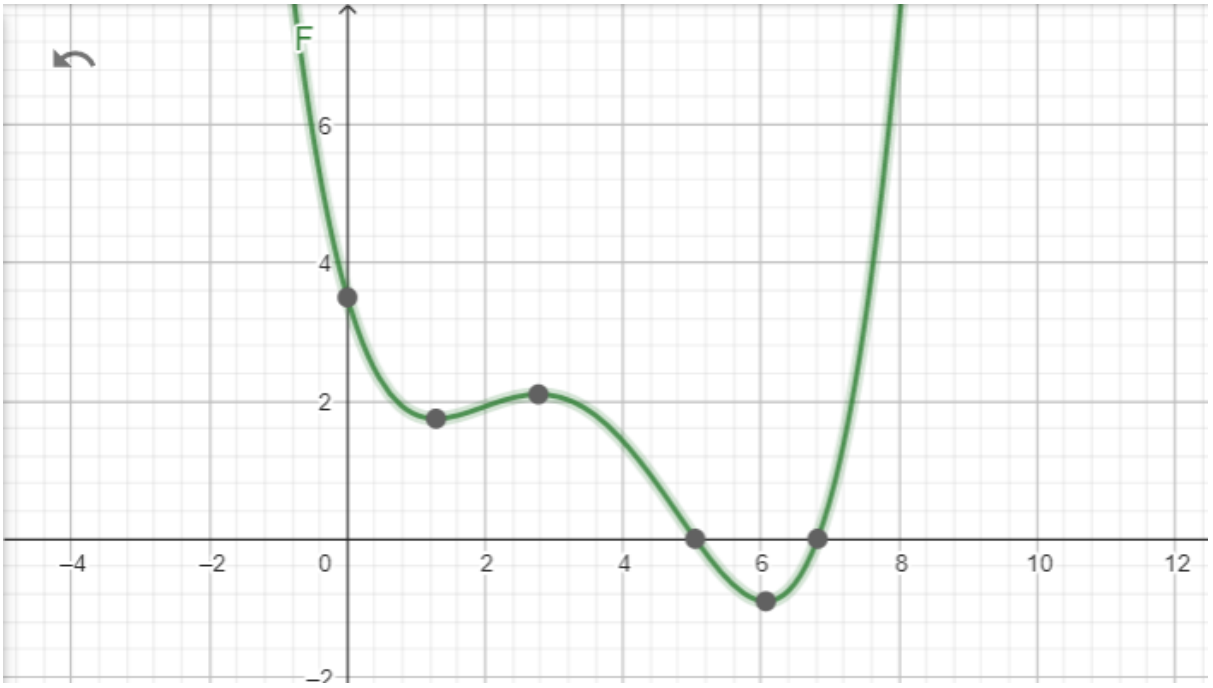
f(x) = 1.5821428571428604 + 0.17499999999999835x + 0.017857142857143026x^2



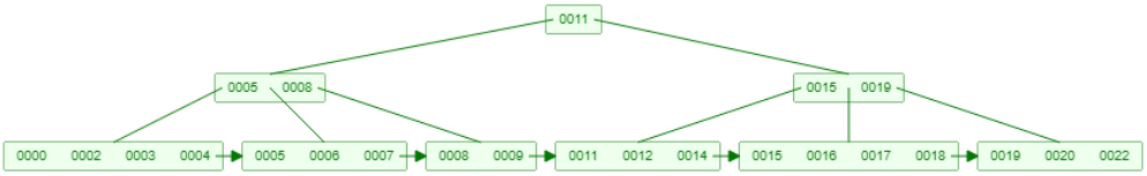



3.

$$F(x)=0.04x^4 - 0.54x^3 + 2.25416x^2 - 3.4583x + 3.5$$




4.




*ANEXOS CODIGOS:*

*CODIGO PARA SOLUCIONAR PRIMER EJERCICIO:*

```
/**
 *
 * @author manue
 */public class PUNTOUNO {
    public static void main(String[] args) {
        int[][] matrixA = { {2, 1, 0}, {1, 3, -2}, {0, 4, -1} };
        int[][] matrixB = { {3, 2}, {0, -1}, {1, 0} };
```

```
        int[][] result = multiplyMatrices(matrixA, matrixB);

        System.out.println("Resultado de la multiplicación de matrices:");
        printMatrix(result);
    }

    private static int[][] multiplyMatrices(int[][] matrixA, int[][] matrixB) {
        int rowsA = matrixA.length;
        int colsA = matrixA[0].length;
        int colsB = matrixB[0].length;

        int[][] result = new int[rowsA][colsB];

        for (int i = 0; i < rowsA; i++) {
            for (int j = 0; j < colsB; j++) {
                for (int k = 0; k < colsA; k++) {
                    result[i][j] += matrixA[i][k] * matrixB[k][j];
                }
            }
        }

        return result;
    }

    private static void printMatrix(int[][] matrix) {
        for (int i = 0; i < matrix.length; i++) {
            for (int j = 0; j < matrix[0].length; j++) {
                System.out.print(matrix[i][j] + " ");
            }
            System.out.println();
        }
    }
}
```

CODIGO PAR ASOLUCIONAR PUNTO 2

```
public class PUNTO2 {
    public static void main(String[] args) {
        // Puntos de muestra
        double[] x = {1, 2, 3, 4, 5,6,7,8};
        double[] y = {1.8, 2, 2.2, 2.6, 2.9, 3.3, 3.7, 4.1};

        // Aproximación lineal
        double[] linearCoefficients = fitLinear(x, y);
        System.out.println("Aproximación lineal:");
        System.out.println("f(x) = " + linearCoefficients[0] + " + " +
linearCoefficients[1] + "x");

        // Aproximación polinómica de grado 2
        double[] polynomialCoefficients = fitPolynomial(x, y, 2);
        System.out.println("Aproximación polinómica de grado 2:");
        System.out.println("f(x) = " + polynomialCoefficients[0] + " + " +
polynomialCoefficients[1] + "x + " + polynomialCoefficients[2] + "x^2");
    }

    private static double[] fitLinear(double[] x, double[] y) {
        int n = x.length;

        double sumX = 0.0;
        double sumY = 0.0;
        double sumXY = 0.0;
        double sumXX = 0.0;

        for (int i = 0; i < n; i++) {
            sumX += x[i];
            sumY += y[i];
            sumXY += x[i] * y[i];
            sumXX += x[i] * x[i];
        }

        double meanX = sumX / n;
        double meanY = sumY / n;
```

```
        double slope = (sumXY - n * meanX * meanY) / (sumXX - n *
meanX * meanX);
        double intercept = meanY - slope * meanX;

        double[] coefficients = {intercept, slope};
        return coefficients;
    }

    private static double[] fitPolynomial(double[] x, double[] y, int degree) {
        int n = x.length;

        int numCoefficients = degree + 1;
        int numEquations = numCoefficients;

        double[][] matrixA = new double[numEquations][numCoefficients];
        double[] vectorB = new double[numEquations];

        for (int i = 0; i < numEquations; i++) {
            for (int j = 0; j < numCoefficients; j++) {
                matrixA[i][j] = sumPower(x, j + i);
            }
            vectorB[i] = sumProduct(x, y, i);
        }

        double[] coefficients = solveLinearSystem(matrixA, vectorB);
        return coefficients;
    }

    private static double sumPower(double[] x, int power) {
        double sum = 0.0;
        int n = x.length;

        for (int i = 0; i < n; i++) {
            sum += Math.pow(x[i], power);
        }

        return sum;
```

```
    }

    private static double sumProduct(double[] x, double[] y, int power) {
        double sum = 0.0;
        int n = x.length;

        for (int i = 0; i < n; i++) {
            sum += Math.pow(x[i], power) * y[i];
        }

        return sum;
    }

    private static double[] solveLinearSystem(double[][] matrixA, double[]
vectorB) {
        int numEquations = matrixA.length;
        int numVariables = matrixA[0].length;

        double[] coefficients = new double[numVariables];

        for (int i = 0; i < numEquations; i++) {
            double pivot = matrixA[i][i];

            for (int j = i + 1; j < numVariables; j++) {
                double factor = matrixA[j][i] / pivot;

                for (int k = 0; k < numVariables; k++) {
                    matrixA[j][k] -= factor * matrixA[i][k];
                }

                vectorB[j] -= factor * vectorB[i];
            }
        }

        for (int i = numVariables - 1; i >= 0; i--) {
            double sum = 0.0;
```

```
            for (int j = i + 1; j < numVariables; j++) {
                sum += matrixA[i][j] * coefficients[j];
            }

            coefficients[i] = (vectorB[i] - sum) / matrixA[i][i];
        }

        return coefficients;
    }
}

codigo para el 3
import java.util.Scanner;

public class TERCERPUNTO {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        // Ingresa los puntos
        /*
        System.out.print("Ingrese el número de puntos: ");
        int n = sc.nextInt();
        */
        double[] x = {1,2,3,4,5};
        double[] y = {1.8, 2, 2.5, 2.8, 3.5};
        /*
        for (int i = 0; i < n; i++) {
            System.out.print("Ingrese el valor de x" + (i+1) + ": ");
            x[i] = sc.nextDouble();
            System.out.print("Ingrese el valor de y" + (i+1) + ": ");
            y[i] = sc.nextDouble();
        }*/

        // Aqui ingresamos el X q nos dan en nuestro caso es 2,5
        System.out.print("Ingrese el valor de x a estimar: ");
        double xIngresado = sc.nextDouble();
```

```
        /*
        // Estimación para el grado 1, llamamos la funcion q cramos
        double yEst1 = lagrangeInterpolation(x, y, xIngresado, 1);
        System.out.println("La estimación para el grado 2 es: " + yEst1);

        // Estimación para el grado 2, llamamos la funcion q creamos
        double yEst2 = lagrangeInterpolation(x, y, xIngresado, 2);
        System.out.println("La estimación para el grado 2 es: " + yEst2);
        double yEst3 = lagrangeInterpolation(x, y, xIngresado, 3);
        System.out.println("La estimación para el grado 1 es: " + yEst3);*/
        // Estimación para el grado 3, llamamos la funcion q creamos

        double yEst4 = lagrangeInterpolation(x, y, xIngresado, 4);
        System.out.println("La estimación para el grado 1 es: " + yEst4);
    }
    //dependiendo de el grado solicitado va ir avanzando en los puntos
    public static double lagrangeInterpolation(double[] x, double[] y,
double xIngresado, int grado) {
        double yEstimacion = 0;

        //el primer for es para asignar el primer valor de Y y con este usar
la formula
        double[] coeficientes = new double[x.length ];
        for (int i = 0; i <= grado; i++) {
            double yi = y[i];
            double xi = x[i];
            //este segundo For es para con la formula ir hallando el valor de
Y
            //pero colcoamos ese if pues segun la formula j tiene q ser
diferente de I
            double[] li = {1};
            double denominador = 1;
            for (int j = 0; j <= grado; j++) {
                if(j!=i){
                    double xj = x[j];
                    double[] polAux = {-xj,1};
                    li = multiplicarPolinomios(li, polAux);
```

```
                denominador *= xi-xj;
                /*yi *= (xIngresado - x[j]) / (x[i] - x[j]);
                System.out.println("yi "+yi+" xJ "+xj);*/
            }
        }
        for (int j = 0; j < li.length; j++) {
            li[j] *= (yi / denominador);

            coeficientes[j] += li[j];
        }

        //sumatoria de los valores de term con todos los puntos de Y
        yEstimacion += yi;
    }
    for (int i = 0; i < coeficientes.length; i++) {
        double coeficiente = coeficientes[i];
        System.out.println("coeifcientes son "+coeficiente);

    }
    System.out.println(" ");
    return yEstimacion;
  }

  private static double[] multiplicarPolinomios(double[] pol1, double[]
pol2) {
    double[] respuesta = new double[pol1.length + 1];
    double coef = 0;
    for (int i = 0; i < pol1.length; i++) {
        for (int j = 0; j < pol2.length; j++) {
            respuesta[i+j] += pol1[i] * pol2[j];
        }
    }

    return respuesta;
  }
}
```