Implementación de código Python con librería numpy.

```python
import numpy as np
import matplotlib.pyplot as plt
import mibiblioteca as bib

# Paso 1: Definir parámetros del modelo real
# Generamos 100 puntos igualmente espaciados en el intervalo [-1, 1]
x = np.linspace(-1, 1, 100)

# Coeficientes del polinomio real
a, b, c = 1, 2, 150

# Evaluamos el polinomio exacto usando los coeficientes
y_real = a + b * x + c * x**2

# Paso 2: Simular datos con ruido
# Número de datos simulados
m = 20
# Generamos datos X aleatorios en el intervalo [-1, 1]
X = 1 - 2 * np.random.rand(m)

# Calculamos Y usando el polinomio real y añadiendo ruido aleatorio
Y = a + b * X + c * X**2 + 2 * np.random.randn(m)
# Paso 3: Construcción explícita de la matriz A usando arrays de numpy
# Matriz A con términos constantes, lineales y cuadráticos
A = np.array([np.ones(m), X, X**2]).T
print("Matriz A:\n", A)

# Paso 4: Calcular la matriz A^T A y el vector A^T Y
AtA = np.dot(A.T, A)
AtY = np.dot(A.T, Y)
print("Matriz A^T A:\n", AtA)
print("Vector A^T Y:\n", AtY)

# Paso 5: Resolver el sistema usando eliminación Gaussiana con pivoteo parcial

# Asegurarse de que AtY sea una columna
b_columna = AtY.reshape(-1, 1)

# Utilizamos una función personalizada GaussElimPiv para resolver el sistema
sol = bib.GaussElimPiv(AtA, b_columna)
print("Solución (coeficientes del polinomio):", sol.flatten())

```

```python
# Paso 6: Evaluar el polinomio ajustado
# Calculamos los valores ajustados del polinomio
y_ajustado = sol[0] + sol[1] * x + sol[2] * x**2
# Paso 7: Graficar los resultados
fig, ax = plt.subplots(figsize=(12, 4))
# Graficar los datos simulados con ruido (puntos verdes)
ax.plot(X, Y, 'go', alpha=0.5, label='Datos simulados')
# Graficar el polinomio real (línea roja)
ax.plot(x, y_real, 'r', lw=2, label='Valor real $y = 1 + 2x + 150x^2$')
# Graficar el polinomio ajustado por mínimos cuadrados (línea azul)
ax.plot(x, y_ajustado, 'b', lw=2,
label=f'Ajuste de mínimos cuadrados $y = {sol[0][0]:.2f} + {sol[1][0]:.2f}x + {sol[2][0]:.2f}x^2$')
# Etiquetas de los ejes
ax.set_xlabel(r"$x$", fontsize=18)
ax.set_ylabel(r"$y$", fontsize=18)
# Mostrar la leyenda
ax.legend(loc=2)
# Mostrar el gráfico
plt.title('AJUSTE POR MINIMOS CUADRADOS EN MODELOS: Y = a + bX + cX²')
plt.show()
```

```
PS C:\Users\Usuario\Desktop\matematica_computacional> & C:/Users/Usuario/AppData/Local/Microsoft/WindowsApps/python3.12.exe c:/Use
rs/Usuario/Desktop/matematica_computacional/miprogramaprincipal.py
Matriz A:
 [[ 1.         -0.90337102  0.81607919]
 [ 1.          0.39312095  0.15454408]
 [ 1.         -0.81400299  0.66260086]
 [ 1.         -0.44820661  0.20088917]
 [ 1.         -0.97357259  0.94784359]
 [ 1.          0.4536383   0.2057877 ]
 [ 1.         -0.71864967  0.51645735]
 [ 1.          0.11984205  0.01436212]
 [ 1.         -0.24296815  0.05903352]
 [ 1.         -0.84895026  0.72071654]
 [ 1.         -0.28506228  0.08126051]
 [ 1.          0.11342674  0.01286563]
 [ 1.          0.4180825   0.17479298]
 [ 1.         -0.61372023  0.37665252]
 [ 1.         -0.18542983  0.03438422]
 [ 1.          0.46967262  0.22059237]
 [ 1.          0.25496195  0.0650056 ]
 [ 1.          0.10239414  0.01048456]
 [ 1.          0.86590199  0.74978625]
 [ 1.         -0.6059662   0.36719504]]
```

```
Matriz A^T A:
 [[20.         -3.44885859  6.39133381]
 [-3.44885859  6.39133381 -2.76910947]
 [ 6.39133381 -2.76910947  3.83025234]]
Vector A^T Y:
 [ 964.54964746 -401.30863046  570.94598438]
Matriz escalonada pivoteada con pivot A:
 [[  20.           -3.44885859    6.39133381  964.54964746]
 [   0.            5.79660253   -1.66696914 -234.97886381]
 [   0.           -1.66696914    1.78779494  262.70804573]]
Matriz escalonada pivoteada con pivot A:
 [[ 2.00000000e+01 -3.44885859e+00  6.39133381e+00  9.64549647e+02]
 [ 0.00000000e+00  5.79660253e+00 -1.66696914e+00 -2.34978864e+02]
 [ 0.00000000e+00  2.22044605e-16  1.30841308e+00  1.95133546e+02]]
Matriz escalonada pivoteada con pivot A:
 [[ 2.00000000e+01 -3.44885859e+00  6.39133381e+00  9.64549647e+02]
 [ 0.00000000e+00  5.79660253e+00 -1.66696914e+00 -2.34978864e+02]
 [ 0.00000000e+00  2.22044605e-16  1.30841308e+00  1.95133546e+02]]
Matriz Eliminacion pivoteada aumentada Ab:
 [[ 2.00000000e+01 -3.44885859e+00  6.39133381e+00  9.64549647e+02]
 [ 0.00000000e+00  5.79660253e+00 -1.66696914e+00 -2.34978864e+02]
 [ 0.00000000e+00  2.22044605e-16  1.30841308e+00  1.95133546e+02]]
```

```
Matriz pivoteda A1:
 [[ 2.00000000e+01 -3.44885859e+00  6.39133381e+00]
 [ 0.00000000e+00  5.79660253e+00 -1.66696914e+00]
 [ 0.00000000e+00  2.22044605e-16  1.30841308e+00]]
vector b1:
 [ 964.54964746 -234.97886381  195.13354625]
 [ 0.00000000e+00  5.79660253e+00 -1.66696914e+00 -2.34978864e+02]
 [ 0.00000000e+00  2.22044605e-16  1.30841308e+00  1.95133546e+02]]
Matriz Eliminacion pivoteada aumentada Ab:
 [[ 2.00000000e+01 -3.44885859e+00  6.39133381e+00  9.64549647e+02]
 [ 0.00000000e+00  5.79660253e+00 -1.66696914e+00 -2.34978864e+02]
 [ 0.00000000e+00  2.22044605e-16  1.30841308e+00  1.95133546e+02]]
Matriz pivoteda A1:
 [[ 2.00000000e+01 -3.44885859e+00  6.39133381e+00]
 [ 0.00000000e+00  5.79660253e+00 -1.66696914e+00]
 [ 0.00000000e+00  2.22044605e-16  1.30841308e+00]]
vector b1:
 [ 964.54964746 -234.97886381  195.13354625]
 [ 0.00000000e+00  2.22044605e-16  1.30841308e+00  1.95133546e+02]]
vector b1:
 [ 964.54964746 -234.97886381  195.13354625]
 [ 0.00000000e+00  2.22044605e-16  1.30841308e+00  1.95133546e+02]]
Matriz pivoteda A1:
 [[ 2.00000000e+01 -3.44885859e+00  6.39133381e+00]
 [ 0.00000000e+00  5.79660253e+00 -1.66696914e+00]
 [ 0.00000000e+00  2.22044605e-16  1.30841308e+00]]
vector b1:
 [ 964.54964746 -234.97886381  195.13354625]
vector b1:
 [ 964.54964746 -234.97886381  195.13354625]
Solución (coeficientes del polinomio): [  0.97352753   2.35118098 149.13756907]
PS C:\Users\Usuario\Desktop\matematica_computacional>
```