Polynomial Regression and Visualization Using LAPACK and Cairo in C with Python Data Generation

Amaan Rahman

Introduction

In a system where n sample points (x_i, y_i) are stored in a text file, the objective is to compute an optimal n-degree polynomial that best fits the given data in the least-squares sense.
This is achieved through:

- **Polynomial Regression** for modeling the data
- **BLAS/LAPACK** for high-performance numerical computation
- **Cairo Graphics Library** for plotting the data points and the fitted curve

Objective

To compute an n-degree polynomial regression that best fits a noisy dataset using optimized numerical routines and visualize the dataset along with the true polynomial curve.

Mathematical Formulation

Polynomial model:

$$y = c_0 x^n + c_1 x^{n-1} + \cdots + c_{n-1} x + c_n.$$

Matrix form:

$$Y = X\beta + E.$$

Normal Equation:

$$X^T X \beta = X^T Y.$$

OLS Formulation

Error vector:

$$E = Y - X\beta$$

Minimizing squared error:

$$E^T E = (Y - X\beta)^T (Y - X\beta)$$

Optimal solution:

$$\beta = (X^T X)^{-1} X^T Y$$

LAPACK solves this without explicit inversion.

LAPACK Implementation

Steps:

1. Compute $X^T X$ using `dgemm`.
2. Compute $X^T Y$ using `dgemv`.
3. Solve $(X^T X)\beta = X^T Y$ using `dposv`.

Advantages:

- Efficient
- Numerically stable
- Integration-friendly with C

Python Code: Random Point Generation

Code Snippet

```
1  def generate_points_and_polynomial():
2      n = int(input("Enter order n: "))
3      x_vals = np.array([random.uniform(-10,10) for
           _ in range(n)])
4      y_vals = np.array([random.uniform(-10,10) for
           _ in range(n)])
5      coeffs = np.polyfit(x_vals, y_vals, n)
6      ...
```

Python Code: Synthetic Data Curve Fitting

Code Snippet

```
1 def generate_synthetic_data(poly_expr, x_range
    =(-5,5), n_points=100):
2     x = np.linspace(x_range[0], x_range[1],
        n_points)
3     y_true = np.array([poly_expr.subs('x',v) for
        v in x], dtype=float)
4     y_noisy = y_true + np.random.normal(0,1,len(x
        ))
5     return x, y_noisy, y_true
```

Python Code: Main Function

Code Snippet

```
popt, pcov = curve_fit(model, x_data, y_noisy)
y_fit = model(x_data, *popt)
with open("synthetic_polynomial_data.txt", "w")
    as f:
     ...
plt.scatter(x_data, y_noisy)
plt.plot(x_data, y_true)
plt.plot(x_data, y_fit)
```

C Code: Reading Data

Code Snippet

```
1  int read_data(const char *filename, int *order,
       int *N,
2                  double **x, double **y_noisy,
                       double **y_true)
3  {
4       FILE *f = fopen(filename, "r");
5       ...
6  }
```

C Code: Vandermonde Matrix

Code Snippet

```
1  void build_vandermonde(int N, int deg, const
       double *x, double *X)
2  {
3       for (int j = 0; j < deg+1; j++)
4           for (int i = 0; i < N; i++)
5               X[j*N + i] = pow(x[i], deg - j);
6  }
```

C Code: LAPACK Solution

Code Snippet

```
1 cblas_dgemm(...);
2 cblas_dgemv(...);
3 dposv_(&uplo, &n, &nrhs, XtX, &lda, Xty, &ldb, &
      info);
```

Saving Optimal Coefficients to File

C Code Snippet

```c
FILE *f = fopen("optimal_coeffs.txt", "w");
if (!f) {
    printf("Error writing file!\n");
    return;
}
for (int i = 0; i < degree+1; i++)
    fprintf(f, "%lf\n", beta[i]);
fclose(f);
```

C Code: Cairo Plotting

Cairo Visualization Code

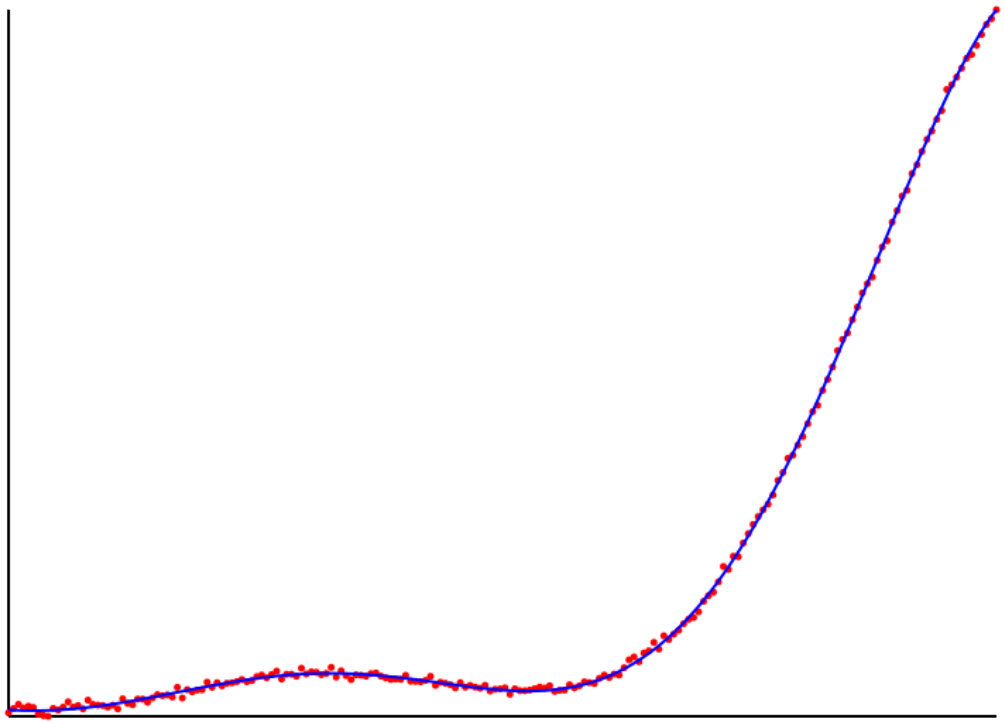
```
1  cairo_surface_t *surface =
2      cairo_image_surface_create(
           CAIRO_FORMAT_ARGB32, 800, 600);
3  cairo_t *cr = cairo_create(surface);
4
5  // draw axes, points, curve ...
6  cairo_set_source_rgb(cr, 1, 0, 0);
7  cairo_stroke(cr);
8
9  cairo_surface_write_to_png(surface, "example-plot
       .png");
10 cairo_destroy(cr);
11 cairo_surface_destroy(surface);
```

Cairo Visualization

Steps:

- Create surface & context
- Plot sample points
- Draw best-fit polynomial curve
- Annotate and label graph
- Export PNG

Plot Example



An example plot for an 8th order equation generated using the Cairo library in C

Input/Output Format

Input:

- Text file containing n sample points (x_i, y_i).

Output:

- Optimal polynomial coefficients
- Visualization of the best-fit curve

Thank You!