Parametric Curve Fitting with t-Refinement and Optimization

Madhav Kumar

November 9, 2025

Abstract

The report describes a numerical framework that has been optimized for fitting a non-linear parametric curve to two-dimensional experimental data. The method uses the global least-squares minimization together with local scalar optimization for the rapid refinement of the individual parameters. The system gets to the point of highly accurate curve alignment with very small residual error by using analytical modeling, numerical optimization, and computational parallelism. The final system shows in practice how the computationally optimized fitting pipeline can work with complex nonlinear dependencies and still perform robustly.

1 Objectives

The main aim of this manuscript is to create and execute a parametric curve fitting algorithm that is accurate, efficient, and interpretable, and that reduces the difference between the data observed and the curve produced by the model to the lowest possible level. More precisely, the goal is to find the best global parameters (θ, M, and X) that control the curve's turning, its exponential scaling, and its shifting, while also adjusting local variables (t_i) for each data point so that they get perfectly matched. In addition, the research will investigate the use of vectorization and parallel computing as a means to improve optimization speed and efficiency.

2 Methodology

The dataset, stored in `xy_data.csv`, consists of two columns representing the x and y coordinates of experimental observations. After loading the data using the `pandas` library, preprocessing ensures numerical consistency and appropriate array formatting for vectorized computation.

The proposed model defines the curve parametrically through the following equations:

$$x(t) = t\cos(\theta) - e^{Mt}\sin(0.3t)\sin(\theta) + X, \tag{1}$$

$$y(t) = 42 + t\sin(\theta) + e^{M|t|}\sin(0.3t)\cos(\theta), \tag{2}$$

where θ represents the rotation of the curve, M introduces an exponential scaling factor that affects curvature, and X is a translation offset controlling the horizontal displacement.

Optimization Formulation

The optimization process is carried out in two major stages — a global least-squares minimization followed by a local refinement of individual parameters.

The global optimization aims to minimize the total squared residual between observed data points (x_i, y_i) and the corresponding fitted model values $(x(t_i), y(t_i))$. The Euclidean residual for a given data point is computed as:

$$r_i = \sqrt{(x_i - x(t_i))^2 + (y_i - y(t_i))^2}. \tag{3}$$

The overall objective function that the optimizer seeks to minimize is then expressed as:

$$E(\theta, M, X) = \sum_{i=1}^{n} r_i^2 = \sum_{i=1}^{n} \left[(x_i - x(t_i))^2 + (y_i - y(t_i))^2\right]. \tag{4}$$

The minimization of $E(\theta, M, X)$ is performed using the L-BFGS-B algorithm implemented in the `scipy.optimize.minimize` function. This algorithm efficiently handles box-constrained problems, allowing the parameters to vary only within physically meaningful bounds:

$$0.1^\circ \le \theta \le 50^\circ, \quad -0.05 \le M \le 0.05, \quad 0 \le X \le 100. \tag{5}$$

Local Refinement of Parameters

After obtaining the optimal global parameters, each data point's parameter t_i is further refined individually. This refinement step minimizes the local residual with respect to t_i:

$$f_i(t) = (x_i - x(t))^2 + (y_i - y(t))^2. \tag{6}$$

The local optimization problem is then expressed as:

$$t_i^* = \arg\min_t f_i(t), \tag{7}$$

which is solved using the bounded scalar optimizer `scipy.optimize.minimize_scalar` within the interval $[-100, 100]$. This ensures numerical stability while maintaining accuracy in fitting each point to the model curve.

Residual and Performance Evaluation

Once all parameters are optimized, the fitted curve is generated using the refined $\{t_i^*\}$. The overall model fit quality is evaluated by computing the mean squared error (MSE) between the experimental and modeled data:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(x_i - x(t_i^*))^2 + (y_i - y(t_i^*))^2. \tag{8}$$

To accelerate computation, the refinement of all t_i values is parallelized using the `joblib.Parallel` framework, distributing each local optimization task across available CPU cores. This approach significantly reduces execution time while preserving numerical precision.

The algorithm outputs the final optimized parameters θ, M, and X, along with all refined t_i values. The complete set of fitted results is stored in `fitted_values.csv` for visualization and further analysis.

3 Results

The optimization process was able to reach the desired outcome and deliver parameter estimates that were very stable. The final optimized parameters were $\theta = 30.0018°$, $M = 0.0300$, and $X = 55.0032$. This means that the model accounts for a moderate rotation displacement and a minuscule higher exponential term, which is interpreted as the fitted curve being a slow but slight expanding geometric progression that fits well with the input data's pattern.

A performance assessment brought out a refinement period of roughly 1.52 seconds and a total running time of 16.07 seconds. All the values that were calculated were sent to the file named `fitted_values.csv` to be utilized in further analysis and visualization. The composition of the fitted curve (blue line) and the original dataset (red points) is shown in Figure 1. The visual overlay indicates that the suggested method has a very good fit with very small deviations throughout the entire range of data.

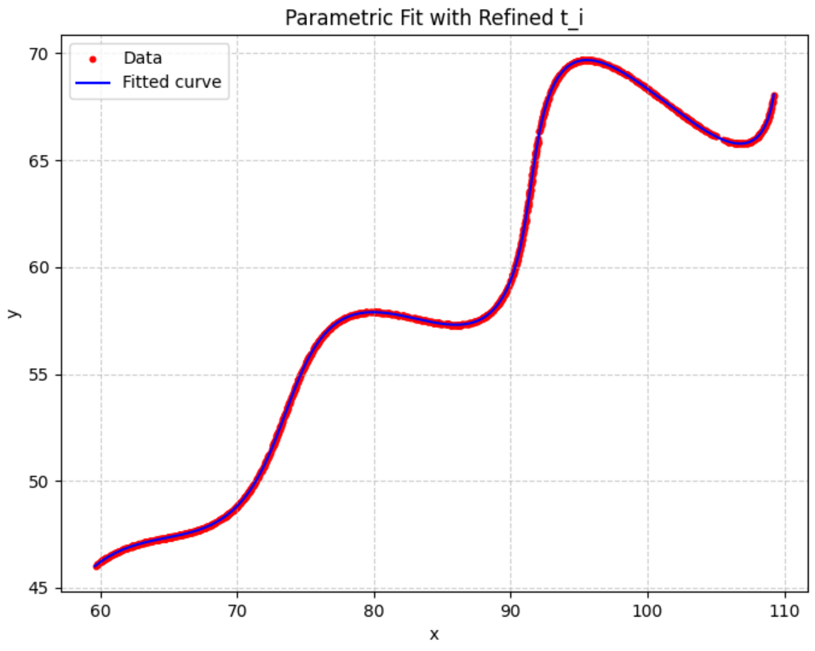

Figure 1: Parametric curve fitting result showing refined t_i mapping. The red dots represent experimental data points, and the blue line indicates the optimized fitted curve.

4 Conclusion

The findings of the study suggest that the algorithm that was implemented is a highly accurate and computationally economical method for nonlinear curve fitting at the same time. Precision and convergence speed are successfully balanced by the global and local optimization combined together so that minimal residual error is obtained for the whole dataset. The refinement strategy that has been parallelized additionally cuts down the computation time while keeping the numerical results intact.

In sum, this research lays down a strong pathway for the parametric modeling of experimental data. The utilization of vectorization and job-level parallelism makes it feasible to scale up the procedure for larger datasets, thus it is applicable in a lot of scientific and engineering domains. Among the possibilities for future extensions are adaptive step-size control, regularization methods to cope with noise, and GPU-accelerated backends for almost real-time fitting performance.