Facultatea de Matematică și Informatică

UNIVERSITATEA BABEȘ-BOLYAI
BABEȘ-BOLYAI TUDOMÁNYEGYETEM
BABEȘ-BOLYAI UNIVERSITÄT
BABEȘ-BOLYAI UNIVERSITY
TRADITIO ET EXCELLENTIA

Differential Evolution with Adaptive Mutation and Crossover Strategy

for solving Nonlinear Regression problems

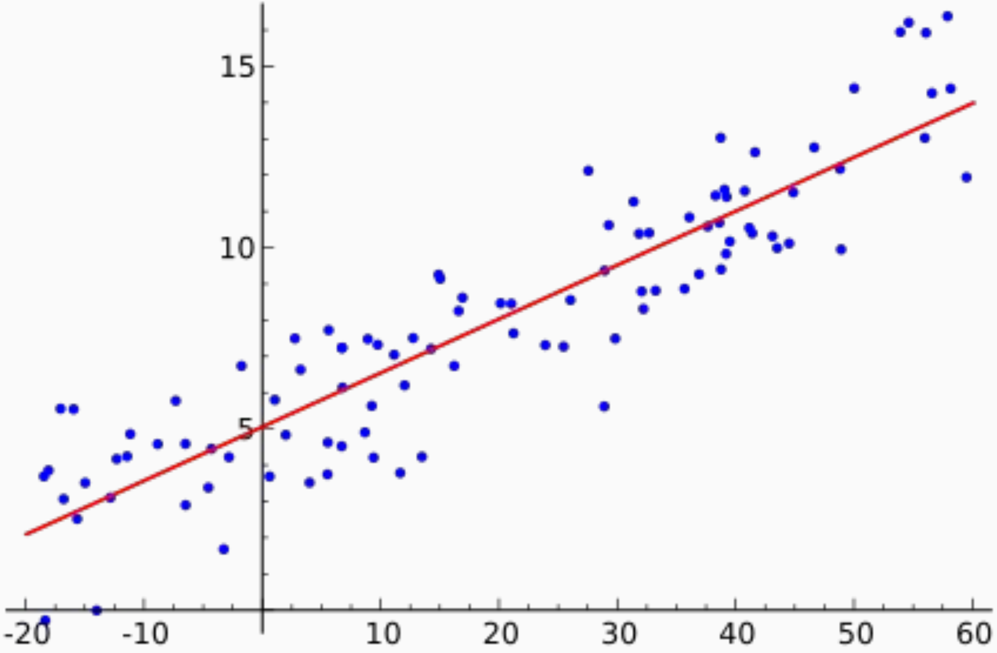
Ivanof Elena-Tamara
2nd year bachelor student in Artificial intelligence
Faculty of Mathematics and Informatics, UBB Cluj-Napoca

What are regression problems?

Regression

- Fitting a mathematical equation to a set of observed data

- Minimizing the error between the predicted values (according to the function) and the actual observed values



Solving a regression problem

General equation:

$$y = f(\vec{x};\vec{\beta}) + \epsilon$$

Method of least squares:

$$RSS = \sum_{i=1}^{n}[y_i - f(\vec{x}_i,\hat{\vec{\beta}})]^2$$

Finding optimal parameters = Minimizing RSS

Linear vs Nonlinear regression

Linear regression

- *linear relationship* between independent and dependent variables
- Analytic solutions
- Example: relationship between house size (x) and price (y)

$$y = \beta_0 + \beta_1 x + \epsilon$$

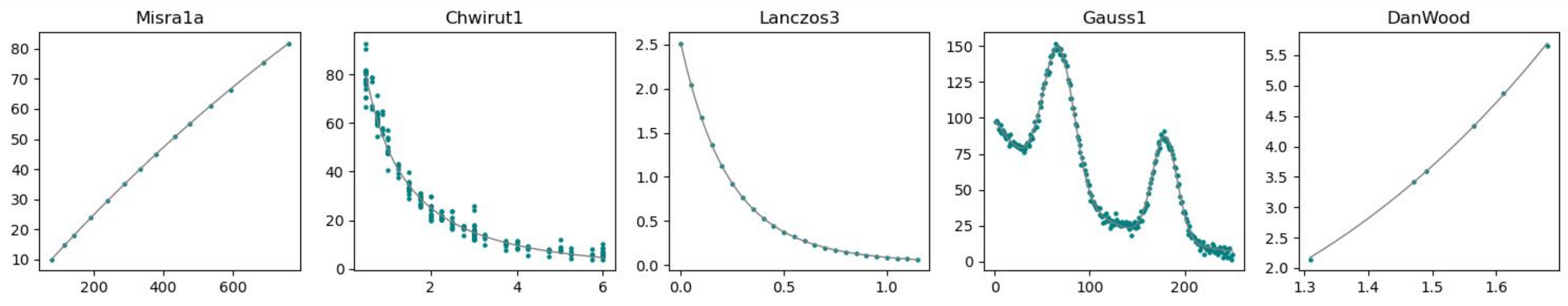
Nonlinear regression

- *nonlinear relationships* between independent and dependent variables
- Iterative optimization
- Example: temperature forecasting

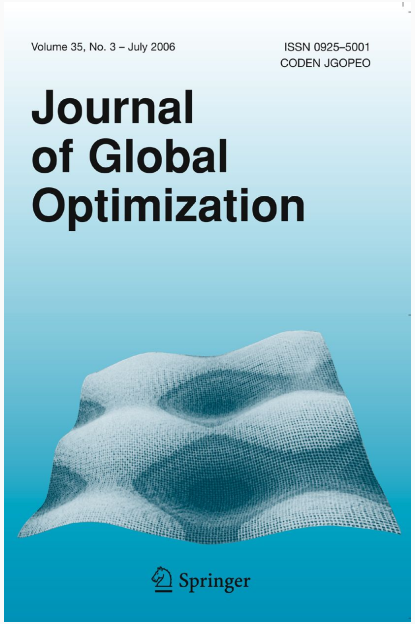
$$y(t) = \beta_0 + \beta_1 \sin\left(\frac{2\pi t}{T}\right) + \beta_2 \cos\left(\frac{2\pi t}{T}\right) + \epsilon$$

NIST models for nonlinear regression

- *The National Institute of Standards and Technology (NIST) proposed 27 datasets used for validating the accuracy and robustness of a certain minimization algorithm for nonlinear regression.*
- *It includes both generated and "real-world" nonlinear least squares problems of varying levels of difficulty.*



What is Differential Evolution?



Differential Evolution (DE) is a population-based stochastic optimization algorithm introduced by Storn and Price in 1997.

Designed for continuous optimization problems, DE is particularly effective for nonlinear, non-differentiable, and multimodal objective functions.

DE belongs to the family of Genetic Algorithms, and has three main operations: **mutation**, **crossover**, and **selection**.

Differential Evolution – steps

1. **Start with a random population of vectors**

$$x_i = [x_{ij}], i = 1, 2, 3, ..., NP; j = 1, 2, 3, ..., D$$

2. **Mutate vectors**

$$xm = x_{r1} + F(x_{r2} - x_{r3})$$

3. **Apply crossover**

$$xc_j = \begin{cases} xm_j & \text{if}(rand_j \leq CR)\text{or}(j = j_{\text{rand}}) \\ x_{i,j} & \text{otherwise} \end{cases}$$

4. **Update vectors with better values (if found)**

$$f(xc) < f(x_i)$$

Differential Evolution – pseudocode

```
1. Initialize population P = {x1, x2, ..., x_NP} randomly within bounds
2. While stopping criterion not met:
   3. For each target vector x_i in P:
      4. Mutation: Generate mutant vector v = x_r₁ + F · (x_r₂ - x_r₃)
      5. Crossover: Create trial vector u by mixing x_i and v (binomial crossover)
      6. Selection: Replace x_i with u if f(u) < f(x_i)
   7. Update best solution x_best
8. Return x_best
```

Differential Evolution - benefits and costs

The DE algorithm is a powerful tool for functional optimization.
It can handle:

- High-dimensional problems
- Multimodal optimization
- Constrained optimization
- Noisy & dynamic optimization

DE also has some disadvantages:

- Not ideal for purely combinatorial problems unless hybridized
- Slower convergence than gradient-based methods for smooth, convex problems

Why improve differential evolution?

The classical DE algorithm has a few **limitations**:

- **Fixed mutation and crossover strategies**
  - Poor exploration vs. exploitation trade-off
- **Static values for scaling factor (F) and crossover rate (CR)**
  - Requires manual tuning for each problem.
- **Premature Convergence**
  - May stagnate in local optima for complex, multimodal functions.

Differential Evolution with Adaptive Mutation and Crossover strategies

Proposed by **Watchara Wongsa**, **Pikul Puphasuk** and **Jeerayut Wetweerapong** from Khon Kaen University, Thailand, in the article:

Differential evolution with adaptive mutation and crossover strategies for nonlinear regression problems

Date published: Mar 6, 2024

The **DEAMC algorithm** improves classical DE in the following ways:

1. **Adaptive Mutation**:
   - Dynamically switches between Classic Mutation (CM) and Sorting Mutation (SM) based on success rates.

$$xm_i = x_{r_1} + F \cdot (x_{r_2} - x_{r_3})$$

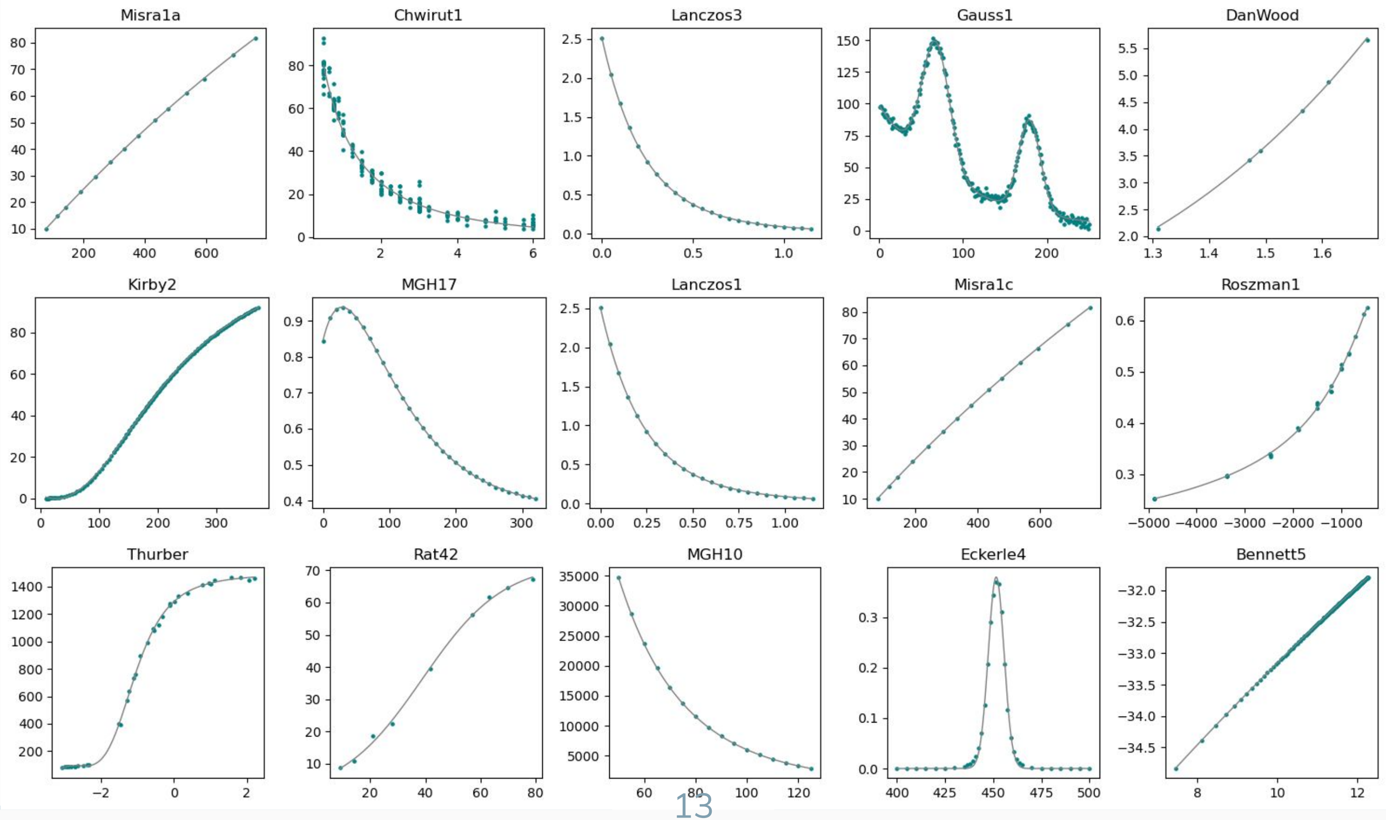
$$xm = x^*_{r_1} + F \cdot (x^*_{r_2} - x^*_{r_3})$$

2. **Dynamic Crossover**:
   - Alternates between low (CR ∈ [0, 0.1]) and high (CR ∈ [0.9, 1]) ranges.
3. **Self-Tuning Probabilities**:
   - Automatically adjusts strategy probabilities (pm1, pc1) using success histories.

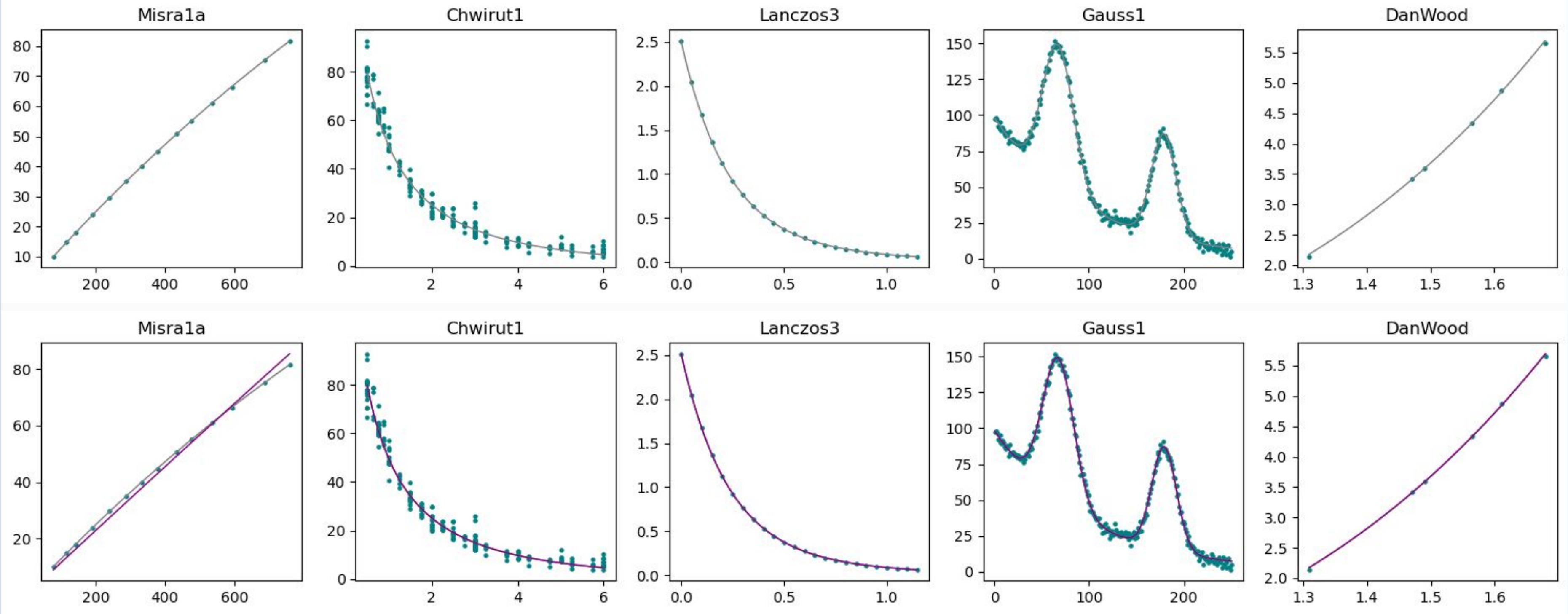
Differential Evolution with Adaptive Mutation and Crossover – pseudocode

```
1. Initialize:
  - Population P = {x_1, x_2, ..., x_NP} within bounds (NP = 10D)
  - Probabilities: pm1 = pm2 = 0.5 (CM/SM), pc1 = pc2 = 0.5 (low/high CR)
  - Counters: nm1 = nm2 = 0 (CM/SM successes), nc1 = nc2 = 0 (CR successes)

2. While stopping criterion not met (log(fw/fb) < eps or nf >= maxnf):
  3. For each target vector x_i in P:
    4. Mutation:
      - Generate F ~ U[0.5, 0.7]
      - With probability pm1:
        v = x_r_1 + F * (x_r_2 - x_r_3)
      - Else:
        Sort {x_r_1, x_r_2, x_r_3} by fitness (ascending) -> {x*_r_1, x*_r_2, x*_r_3}
        v = x*_r_1 + F * (x*_r_2 - x*_r_3)

    5. Crossover:
      - With probability pc1: CR ~ U[0, 0.1] (low range)
        Else: CR ~ U[0.9, 1] (high range)
      - Generate trial vector u via binomial crossover:
        u_j = v_j if (rand < CR or j = j_rand) else x_i_j

    6. Selection:
      - If f(u) < f(x_i):
        Replace x_i with u
        Update success counters (nm1/nm2 or nc1/nc2)
        If f(u) < f(x_best): x_best = u

7. Adaptation (every 100 successes):
  - Update pm1 = 0.9*pm1 + 0.1*(nm1/(nm1 + nm2))
  - Update pc1 = 0.9*pc1 + 0.1*(nc1/(nc1 + nc2))
  - Reset counters nm1, nm2, nc1, nc2 = 0
8. Return x_best
```

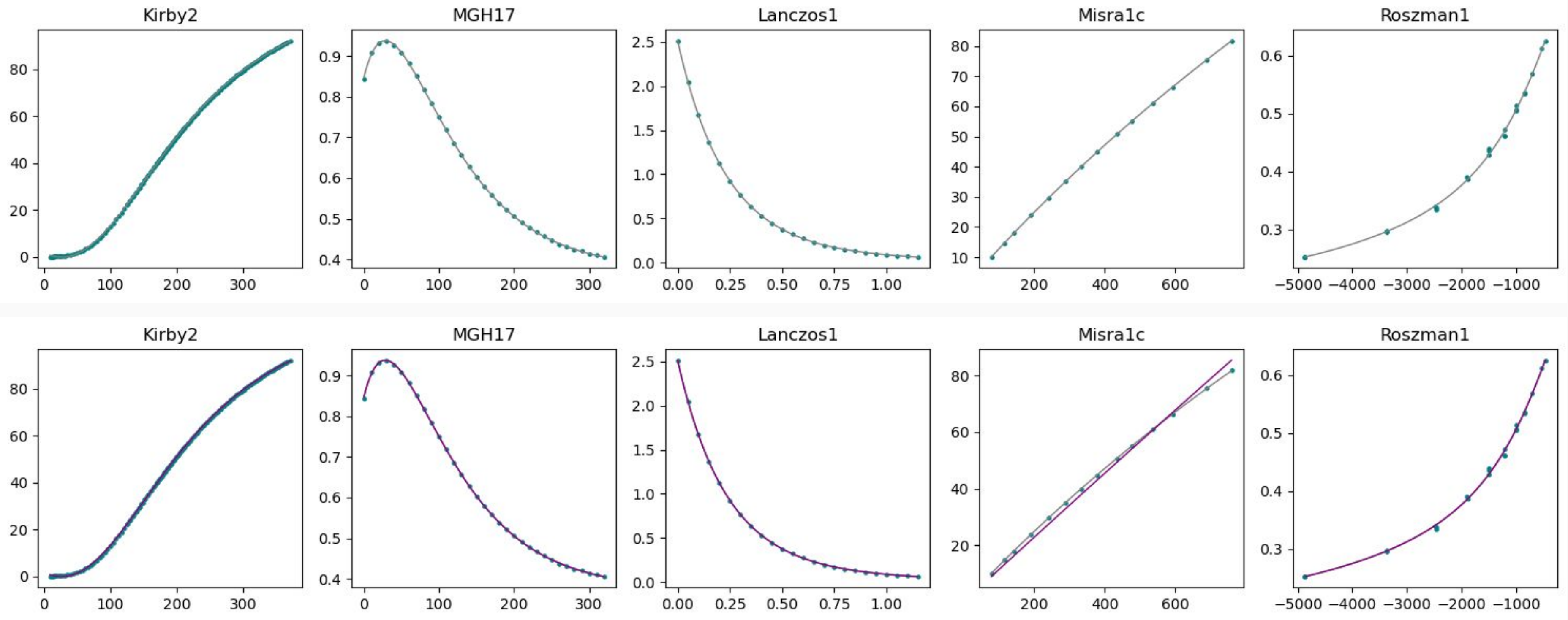
Applying DEAMC to nonlinear regression problems



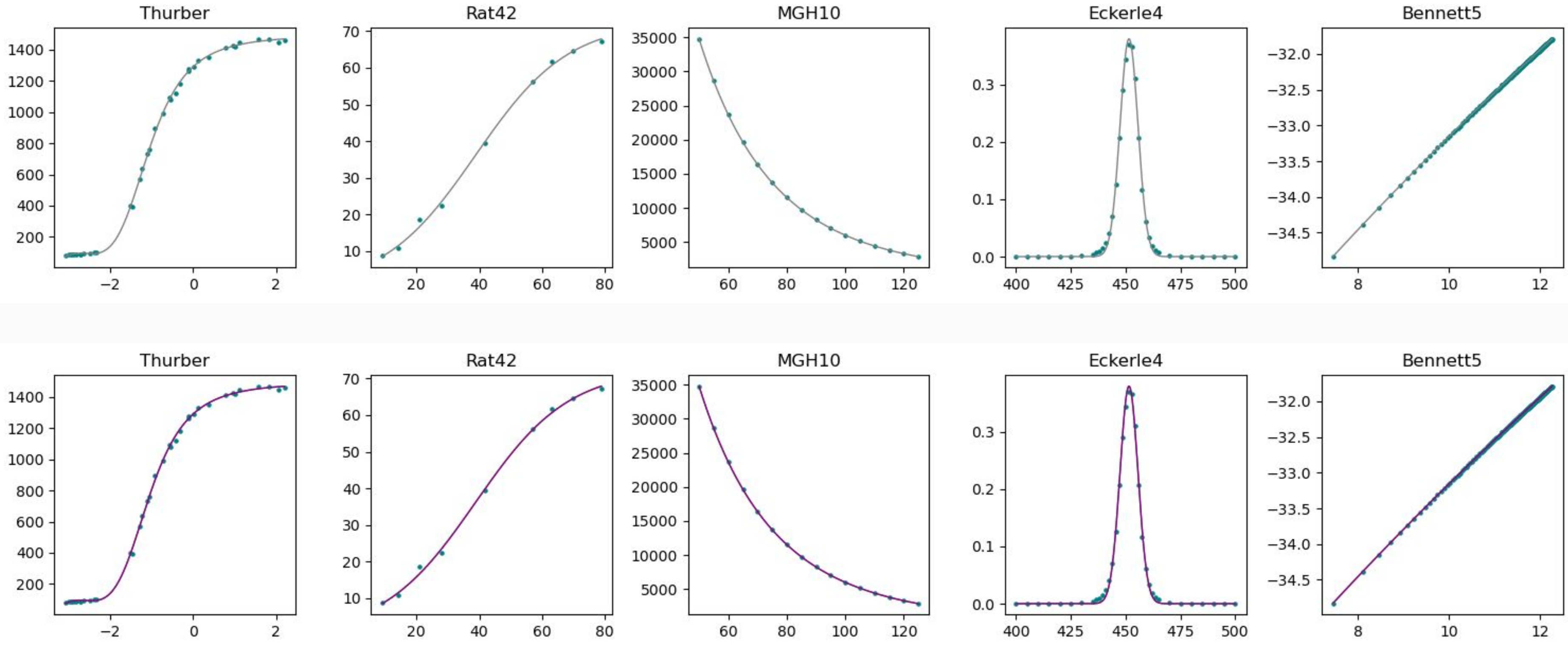
Preliminary results - Lower difficulty models



Preliminary results - Average difficulty models



Preliminary results - Higher difficulty models



Performance analysis

fb - best value obtained from the algorithm

c - certified value for a problem

λ - number of decimal places that fb matches the certified value

NS - number of successful runs, calculated as the number of runs where the algorithm reaches early convergence and λ > 4

nf - number of evaluations before the algorithm stops

Performance analysis - Lower difficulty models

Problem	NS	Mean nf	Mean λ
Misra1a	4	1550.0	3.0128800100578625
Chwirut1	10	3420.0	6.060057889293306
Lanczos3	5	58308.0	3.537158354598259
Gauss1	4	26936.0	2.9226720302463045
DanWood	6	1494.0	4.804842280430851

My results (10 experiments)

Problem	NS	Mean nf	Mean λ
Misra1a	100	2,892	10.4
Chwirut1	100	5,328	11.0
Lanczos3	100	68,101	10.8
Gauss1	100	28,562	10.7
Danwood	100	2,062	11.0

Paper results (100 experiments)

Performance analysis - Average difficulty models

Problem	NS	Mean nf	Mean λ
Kirby2	10	10985.0	6.536159829516862
MGH17	10	14820.0	6.243980396984648
Lanczos1	0	29640.0	0.09366064699010357
Misra1c	3	1546.0	2.498285873480339
Roszman1	10	6756.0	6.461049863373413

My results (10 experiments)

Problem	NS	Mean nf	Mean λ
Kirby2	100	15,537	11.0
Mgh17	100	18,260	11.0
Lanczos1	0	240,000	0.0
Misra1c	100	3,571	11.0
Roszman1	100	6,734	11.0

Paper results (100 experiments)

Performance analysis - Higher difficulty models

Problem	NS	Mean nf	Mean λ
Thurber	9	23317.0	6.452229711197411
Rat42	10	4062.0	7.399562335683347
MGH10	10	17709.0	8.189990606719089
Eckerle4	8	3093.0	7.381809585085401
Bennett5	1	73551.0	1.2764681933464765

My results (10 experiments)

Problem	NS	Mean nf	Mean λ
Thurber	100	25,751	10.9
Rat42	100	4,002	11.0
Mgh10	100	39,696	11.0
Eckerle4	100	2,995	10.7
Bennett5	100	81,797	11.0

Paper results (100 experiments)

Conclusions + Possible adjustments

- The algorithm generally performs well, yielding maximum successful runs for 6 out of 15 experiments.
- The algorithm handles complex data very well, perhaps even better than simple data. This indicates that it is more suitable for complex regression problems.
- Possible adjustments for improving performance include:
    - modifying the rate of adaptation for balancing exploration and exploitation;
    - adjusting the bounds for the initial values of vectors.

References

- R. Storn and K. Price, *Differential Evolution–A simple and efficient heuristic for global optimization over continuous spaces*, Journal of Global Optimization, vol. 11, no. 4, pp. 341-359, 1997, doi: 10.1023/A:1008202821328.
- Wongsa, Watchara & Puphasuk, Pikul & Wetweerapong, Jeerayut. (2024). *Differential evolution with adaptive mutation and crossover strategies for nonlinear regression problems.* Bulletin of Electrical Engineering and Informatics. 13. 3503-3514. 10.11591/eei.v13i5.6417.
- - National Institute of Standards and Technology (NIST). (n.d.). *Nonlinear Least Squares Regression.* Retrieved May 10, 2025, from https://www.itl.nist.gov/div898/handbook/pmd/section1/pmd142.htm
- - National Institute of Standards and Technology (NIST). (n.d.). *Nonlinear Least Squares Regression Background Information*. Retrieved May 10, 2025, from https://www.itl.nist.gov/div898/strd/nls/nls_info.shtml
- - National Institute of Standards and Technology (NIST). (n.d.). *Statistical Reference Datasets (STRD)*. Retrieved May 10, 2025, from https://www.itl.nist.gov/div898/strd/frames.html

Thank you!

Do you have any questions?