



AI Powered Lyapunov Function Discovery

Scaling to Real Power Systems on Barkla2

Abhiram Bhat | N8 Research Intern | University of Liverpool

a.b.bhat@liverpool.ac.uk



Agenda

1. Background & Overview
2. Changes
3. New System
4. Results
5. Q&A

Background & Overview

Lyapunov Functions - $V(x)$

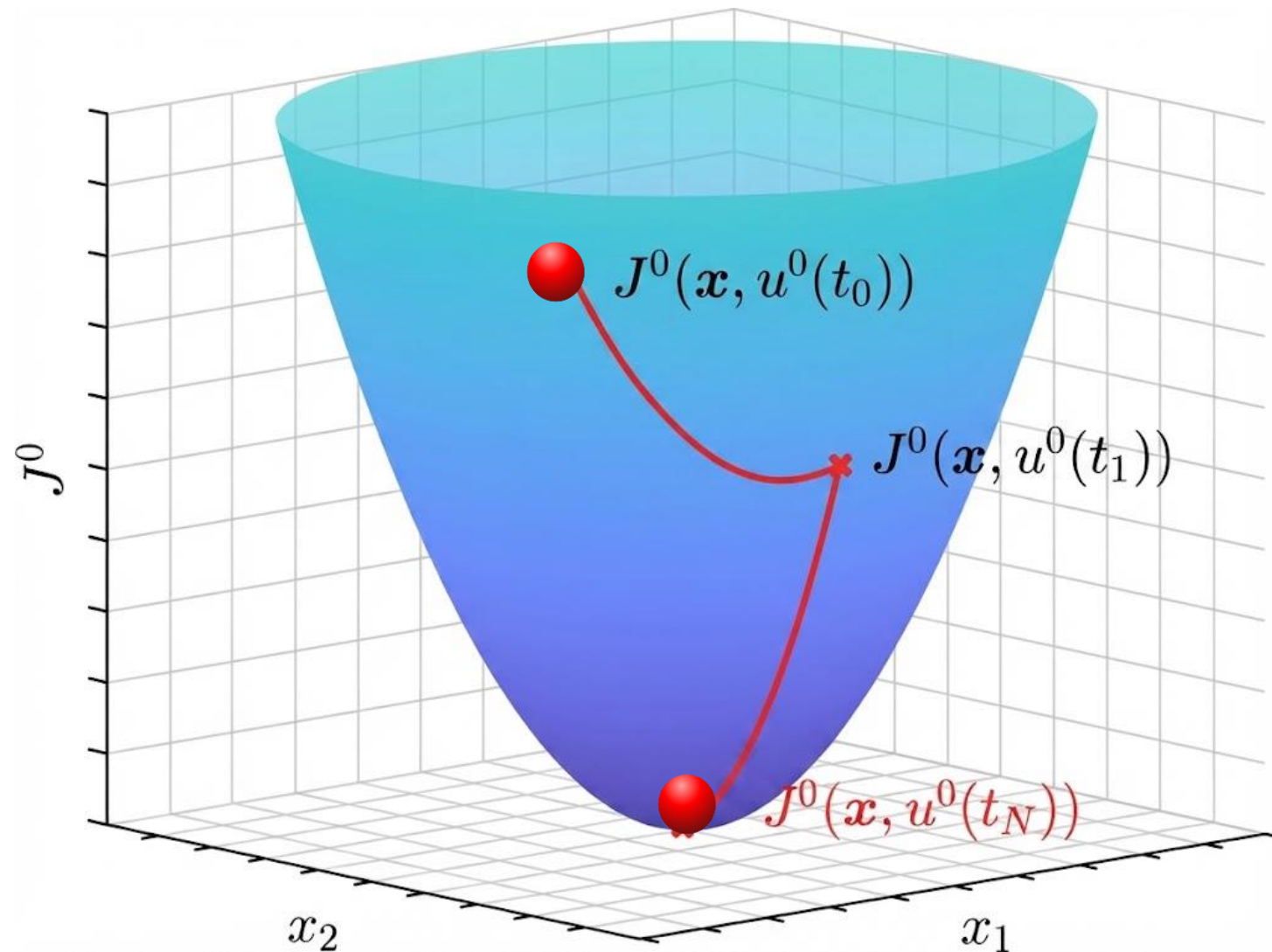


Fig. 1 Stable Grid
Will always reach equilibrium

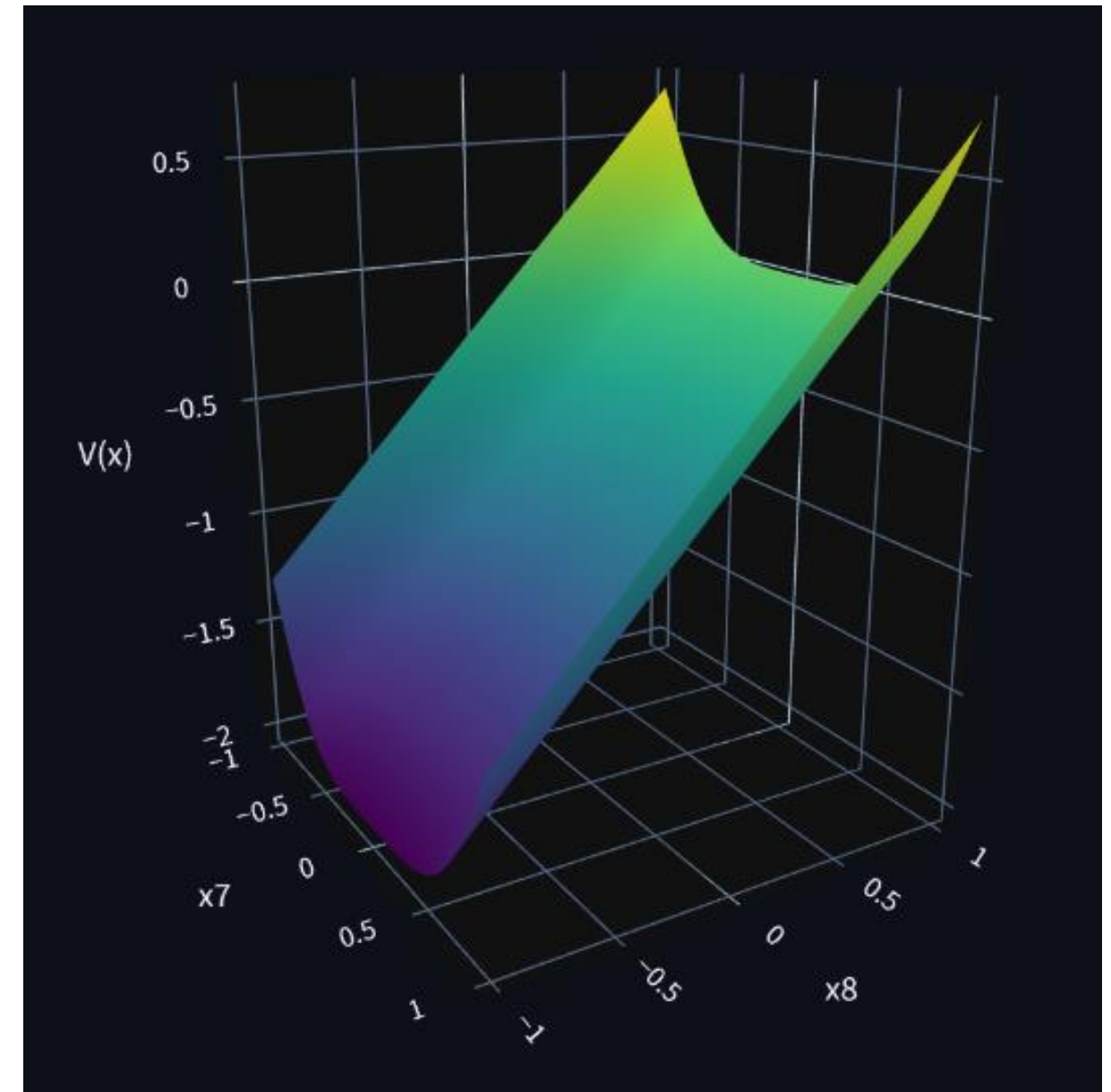


Fig. 2 Unstable Grid
No equilibrium exists

The Goal: Find a "Lyapunov Function" - a mathematical formula that proves the grid is safe.

The Context

This project is based off an existing paper. The paper used Reinforcement Learning to find the functions for small, basic systems and was run on a commercial PC.

Analytical Lyapunov Function Discovery: An RL-based Generative Approach

Haohan Zou^{*1} Jie Feng^{*1} Hao Zhao² Yuanyuan Shi¹

Fig. 3 Key Paper

The Question

How can we scale the reinforcement learning framework onto a high-performance supercomputer to automatically discover valid Lyapunov Functions for complex power grids?

The Solution

1. Scale to run on Barkla2, Uni of Liverpool's HPC that uses datacentre grade GPUs.
2. Implement and run complex power systems: A real 6-dimensional DC microgrid.

How the AI Discovers Safety Equations

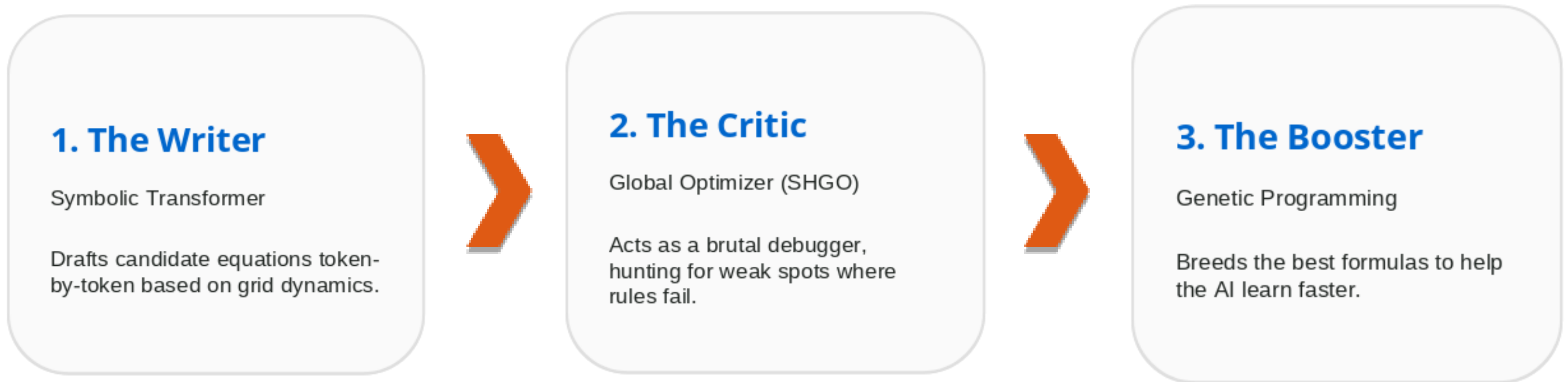


Fig. 4 Paper's Framework

Changes

Codebase Changes

- Streamlined Dimension Selection

Restructured the code to simplify and streamline running a different dimension.

- Caching

The code recompiled identical sub-expressions every time.

- Visualiser

Needed a way to visualise clunky log files (>90k lines).

Fig. 5 Changing dimension code

```
# 2d, 3d, 4d, 6d_poly, 6d_quad, 8d, 10d, 12d, 6d_dc, 20d_kron, 49d_full  
SELECT_DIMENSION = "6d_dc"
```

```
def CURRENT_CONFIG():  
    CONFIGS = {  
        "2d": "example_config.config_2d_simple_pendulum",  
        "3d": "example_config.config_3d_trig",  
        "4d": "example_config.config_4d_2bus_lossy_power",  
        "6d_poly": "example_config.config_6d_poly",  
        "6d_quad": "example_config.config_6d_quadrator",  
        "8d": "example_config.config_8d_poly",  
        # "9d": "example_config.config_9d_syn",  
        "10d": "example_config.config_10d_poly",  
        "12d": "example_config.config_12d_bus",  
        "6d_dc": "example_config.config_6d_dc_microgrid",  
        "20d_kron": "example_config.config_20d_ieee39_kron",  
        "49d_full": "example_config.config_49d_ieee39_full",  
    }  
  
    module_name = CONFIGS[SELECT_DIMENSION]  
    return importlib.import_module(module_name)
```

```
# Gets the benchmark name based on the dimension for Lyapunov file
```

```
def get_benchmark_name():  
    dimension_benchmark_name = {  
        "2d": "fn_d_all_y",  
        "3d": "fn_d_all_t",  
        "4d": "fn_d_all_o",  
        "6d_poly": "fn_d_all_m",  
        "6d_quad": "fn_d_all_z",  
        "8d": "fn_d_all_n",  
        # "9d": "fn_d_all_l",  
        "10d": "fn_d_all_x",  
        "12d": "fn_d_all_12d",  
        "6d_dc": "fn_d_all_6dc",  
        "20d_kron": "fn_d_all_kron",  
        "49d_full": "fn_d_all_full",  
    }  
  
    return dimension_benchmark_name[SELECT_DIMENSION]
```

New System

6D DC Microgrid

- Non-polynomial ODE system modelling:
 - 3 inductor currents,
 - 3 capacitor voltages (6 dimensions).
- Evaluated 50,000 grid samples, Domain: $[-0.3, 0.3]^6$
- Updated function set to only use "add", "n2".
- Changed token range to 12 - 20, enabled compact forms.

```
def dynamics():  
  
    # Base energy ODEs in z:  
    dz1 = (-R[0] / L[0]) * z1 - inv_sqrt_L1C1 * z4  
    dz2 = (-R[1] / L[1]) * z2 + inv_sqrt_L2C1 * z4 - inv_sqrt_L2C2 * z5  
    dz3 = (-R[2] / L[2]) * z3 + inv_sqrt_L3C1 * z4 - inv_sqrt_L3C3 * z6  
    dz4 = inv_sqrt_L1C1 * z1 - inv_sqrt_L2C1 * z2 - inv_sqrt_L3C1 * z3  
    dz5 = inv_sqrt_L2C2 * z2 - inv_sqrt_C2 * (  
        900.0 / (v_e2 + z5 * (1.0 / inv_sqrt_C2)) - 900.0 / v_e2  
    )  
    dz6 = inv_sqrt_L3C3 * z3 - inv_sqrt_C3 * (  
        600.0 / (v_e3 + z6 * (1.0 / inv_sqrt_C3)) - 600.0 / v_e3  
    )  
  
    # Transformed ODEs dy = R * dz:  
    dynamics_ode = [  
        dz1 + c * dz4,  
        dz2 + c * dz5,  
        dz3 + c * dz6,  
        s * dz4,  
        s * dz5,  
        s * dz6,  
    ]  
  
    return state_variables, dynamics_ode
```

Fig. 6 Sample ODEs of 6D system

Results

Success: Certified 6D Lyapunov Discovered!

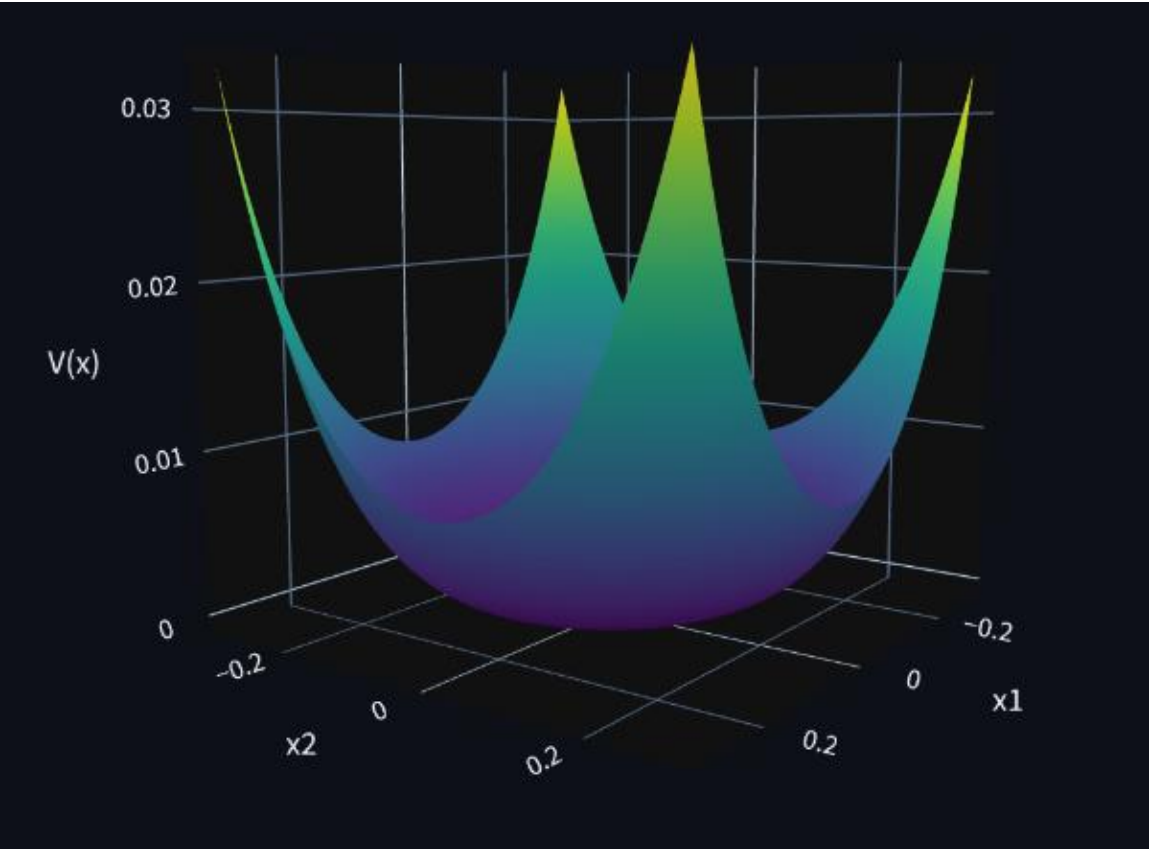
VERIFIED SOLUTION

Dataset: fn_d_all_6dc_5 | Dimension: 6D | Seed: 5 | Evaluations: 1,553,142 | Walltime: 26645.5s

$V(x) = (x1**2 + x2**2 + x3**2 + x4**2 + x5**2 + x6**2)**2$

Maximum Reward (r_best)	Test NMSE	Cache Hit Rate	Early Stopping
0.99608	0.000	31.8%	Triggered

Fig. 7 Screenshot 1 of result



$$V(x) = \left(\sum_{i=1}^6 x_i^2\right)^2$$

Fig. 8 Discovered function and its energy bowl

The 4-Stage Equation Evolution

Stage	Epoch & Time	Reward r	Candidate Equation $V(x)$	What the AI Learned / Did
1. Blind Guess	Epoch 1 (00:02:45)	$r = 0.0564$	$(2x_1 + x_2 + x_3 + x_4 + x_6)^2$	Incomplete: equation is missing the capacitor voltage x_5
2. Finding All States	Epoch 13 (00:47:15)	$r = 0.1013$	$(x_5^2 + (x_3^2 + x_6 + (x_1^2 + x_2^2 + x_4^2)^2)^2)^4$	Discovers all 6 variables but strings them into a convoluted polynomial
3. Learning Coupling	Epoch 40 (02:20:01)	$r = 0.6842$	$(x_5 + x_6 + (x_4 + (x_1 + x_3 + (x_1 + x_2)^2)^2)^2)^8$	Large Reward Jump: The agent begins pairing currents (x_1, x_2, x_3) with voltage (x_4, x_5, x_6)
4. Convergence (Success)	Epoch 113 (07:23:58)	$r = 0.9961$ (nmse=0)	$(x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 + x_6^2)^2$	Analytical Solution: Discovers the clean function.

Table 1 Evolution of the Equation

Thank you

Dr Lin Jiang ljjiang@liverpool.ac.uk
Academic Supervisor, University of Liverpool

Dr Manhui Wang mhwang@liverpool.ac.uk
HPC | Research IT, University of Liverpool

Dr Zitian Qiu z.qiu@imperial.ac.uk
Research Associate, Imperial College London

Q & A