**Genetic Algorithm Solver for Systems of Equations (2x2, 3x3, 4x4)**

**Overview**

This project uses a **Genetic Algorithm (GA)** to solve systems of nonlinear equations involving 2, 3, or 4 variables. The algorithm evolves a population of candidate solutions over several generations to minimize the total error (difference between left-hand and right-hand sides of the equations).

**Project Phases & Development Journey**

**🔹 Phase 1: 2x2 System Solver**

* **Goal**: Solve 2 equations with 2 unknowns using GA.
* **Challenges**:
  + Finding a proper loss function that doesn't just minimize error globally but finds correct variable values.
  + Ensuring convergence under mutation and crossover settings.
* **Outcome**: Achieved convergence within 10 seconds and L2 loss < 1e-6.

**🔹 Phase 2: 3x3 System Solver**

* **Goal**: Extend the model to 3 equations and 3 variables.
* **Challenges**:
  + Parsing and evaluating symbolic equations dynamically.
  + Some solutions had low loss but wrong values (false convergence).
  + Ensuring accuracy and speed.
* **Solutions**:
  + Used sympy for symbolic parsing and lambdify for fast numerical evaluation.
  + Introduced convergence threshold and stall detection.
* **Performance**: Reliable results within ~20 generations.

**🔹 Phase 3: 4x4 System Solver**

* **Goal**: Handle 4 equations with 4 unknowns.
* **Challenges**:
  + Managing increased computational load due to higher dimensionality.
  + Maintaining generalizability for arbitrary symbolic input.
* **Enhancements**:
  + Made population size, mutation parameters, and elite count configurable.
  + Generalized the entire codebase to work for N-variable systems.
  + Added protection against inf/nan solutions.

**Key Features**

* Dynamic equation parsing via user input
* Loss minimization using L1 norm (|f1| + |f2| + ...)
* Configurable GA parameters: mutation probability, elite count, population size, etc.
* Early stopping via convergence detection
* Clear generation-wise logs of best individual

**Technologies Used**

* **Python**
* **Libraries**:
  + sympy for symbolic math and equation handling
  + numpy for population and mutation operations
  + random for parent selection and crossover randomness

**Sample Execution**

Equation 1: x + y + z + w = 10

Equation 2: x - y + z - w = 2

Equation 3: x\*y - z\*w = 5

Equation 4: x\*\*2 + y\*\*2 + z\*\*2 + w\*\*2 = 50

Gen 21: x=1.2356, y=2.1009, z=3.4890, w=3.1745, Error=0.00000239

**Challenges Faced**

* **False convergence**: Population gets stuck in a local minima with low error but invalid values.
* **Symbolic handling**: Users can enter any nonlinear symbolic equations.
* **Performance tuning**: Balancing between speed and accuracy.

**Lessons Learned**

* Symbolic math is powerful but requires careful handling.
* Genetic algorithms are a strong alternative when analytical solutions are hard or impossible.
* Precision control and convergence criteria are crucial for reliable results.

**Future Work**

* Support GUI for input and visualization
* Integrate constraint handling
* Add support for arbitrary number of equations (n >= 2)

**Credits & Resources**

* OpenAI ChatGPT for logic design and debugging
* [SymPy Documentation](https://docs.sympy.org/)
* [NumPy Documentation](https://numpy.org/doc/)