**User Manual for Numerical Methods Library**

**Table of Contents**

1. Introduction
2. Getting Started
   * Prerequisites
   * Installation
3. Methods Overview
4. Function Details
   * validate\_function
   * symplified\_function
   * graph
   * Incremental Search
   * Bisection Method
   * False Position Method (regla\_falsa)
   * Fixed-Point Method (punto\_fijo)
   * Newton's Method
   * Utility Functions (sustReg, sustprgr, sustregr, etc.)
5. Troubleshooting
6. FAQs
7. Support Information

**1. Introduction**

This library provides implementations of several numerical methods for solving equations, including root-finding methods, algebraic expression handling, and graphing tools. It's designed for educational purposes and practical use in numerical analysis.

**2. Getting Started**

**Prerequisites**

Ensure you have the following installed on your system:

* Python 3.6+
* Required libraries: numpy, sympy, matplotlib, base64, io, Django

**Installation**

1. Clone or download the project folder.
2. Install dependencies:

bash

pip install numpy sympy matplotlib Django

python manage.py runserver

**3. Methods Overview**

This library includes the following capabilities:

* **Validation and Simplification of Functions**: Check and process algebraic expressions.
* **Graphing**: Generate graphs for algebraic functions.
* **Root-Finding Methods**:
  + Incremental Search
  + Bisection Method
  + False Position Method
  + Fixed-Point Iteration
  + Newton's Method
* **Utility Functions**: Forward and backward substitution for matrix operations.

**4. Function Details**

**validate\_function(function)**

Validates if a given function can be parsed as a symbolic expression.

* **Input**: A string representing an algebraic function (e.g., "x\*\*2 - 4").
* **Output**: True (valid) or False (invalid).

**symplified\_function(fun)**

Simplifies and converts a symbolic expression into a Python function for evaluation.

* **Input**: A string algebraic function.
* **Output**: A callable function (e.g., lambda x: x\*\*2 - 4).

**graph(function)**

Generates a graph of the given function within the range -10 to 10.

* **Input**: A string algebraic function.
* **Output**: Base64-encoded PNG image of the graph.

**Incremental Search**

Identifies intervals containing roots of a function.

* **Function**: incremental\_search(f, x0, delta, N)
* **Inputs**:
  + f: String algebraic function.
  + x0: Starting point.
  + delta: Step size.
  + N: Maximum iterations.
* **Output**: List of intervals where roots exist.

**Bisection Method**

Finds roots of a function within a given interval using the bisection algorithm.

* **Function**: bisection(a, b, function, tolerance, N)
* **Inputs**:
  + a, b: Interval bounds.
  + function: String algebraic function.
  + tolerance: Convergence criterion.
  + N: Maximum iterations.
* **Outputs**:
  + Root approximation.
  + Iterations used.
  + Error estimate.
  + Iteration table (detailed steps).

**False Position Method (regla\_falsa)**

Root-finding method using the false position algorithm.

* **Function**: regla\_falsa(function, a, b, tolerance, N)
* **Inputs**: Same as bisection method.
* **Output**: Same as bisection method.

**Fixed-Point Method (punto\_fijo)**

Finds a root by iterating a given transformation function.

* **Function**: punto\_fijo(function, g\_function, x0, tolerance, N)
* **Inputs**:
  + function: Original function.
  + g\_function: Transformation function.
  + x0: Initial guess.
  + tolerance: Convergence criterion.
  + N: Maximum iterations.
* **Outputs**:
  + Approximation of the root.
  + Iterations and error.
  + Iteration table.

**Newton's Method**

Root-finding method using derivatives.

* **Function**: newton\_method(x0, tolerance, function, N)
* **Inputs**:
  + x0: Initial guess.
  + tolerance: Convergence criterion.
  + function: String algebraic function.
  + N: Maximum iterations.
* **Outputs**:
  + Approximation of the root.
  + Iterations used.
  + Iteration table.

**Utility Functions**

* **sustReg**: Backward substitution for upper triangular matrices.
* **sustprgr**: Forward substitution for lower triangular matrices.
* **sustregr**: Backward substitution for systems with augmented matrices.

**5. Troubleshooting**

| **Problem** | **Possible Cause** | **Solution** |
| --- | --- | --- |
| "invalid function" error | Invalid input expression | Ensure the input is correct. |
| No root found | Incorrect interval or function | Check the interval or function. |
| Slow convergence | Poor initial guess or high tolerance | Adjust initial guess/tolerance. |

**Secant Method**

**Purpose:  
Finds an approximate root of a function using the secant method, which iteratively updates the root estimate using the function values at two initial points.**

**Inputs:**

* **x0: Initial estimate of the root.**
* **x1: Second estimate of the root.**
* **tolerance: The stopping criterion for the error.**
* **N: Maximum number of iterations.**
* **function: A string representing the algebraic function to analyze.**

**Outputs:**

* **A tuple containing:**
  + **The number of iterations performed.**
  + **The approximate root.**
  + **The function value at the root.**
  + **The error.**
  + **A table showing the iteration history (iteration, current estimate, function value, and error).**
* **If unsuccessful, a message such as "ran out of iterations" or "dividing by 0" with the table of iterations.**

**Multiple Roots Method**

**Purpose:  
Finds a root of a function with possible multiplicity using higher-order derivatives for refinement.**

**Inputs:**

* **function: A string representing the algebraic function to analyze.**
* **x0: Initial estimate of the root.**
* **tolerance: The stopping criterion for the error.**
* **N: Maximum number of iterations.**

**Outputs:**

* **A tuple containing:**
  + **The number of iterations performed.**
  + **The approximate root.**
  + **The function value at the root.**
  + **The error.**
  + **A table showing the iteration history (iteration, current estimate, function value, and error).**
* **If unsuccessful, a message such as "ran out of iterations" with the table of iterations.**

**Gaussian Elimination (Simple)**

**Purpose:  
Solves a system of linear equations using simple Gaussian elimination without pivoting.**

**Inputs:**

* **A: Coefficient matrix (list of lists or NumPy array).**
* **b: Right-hand side vector (list or NumPy array).**

**Outputs:**

* **A NumPy array representing the solution vector.**

**Example Usage:**

**pythonSecant Method**

**Purpose:  
Finds an approximate root of a function using the secant method, which iteratively updates the root estimate using the function values at two initial points.**

**Inputs:**

* **x0: Initial estimate of the root.**
* **x1: Second estimate of the root.**
* **tolerance: The stopping criterion for the error.**
* **N: Maximum number of iterations.**
* **function: A string representing the algebraic function to analyze.**

**Outputs:**

* **A tuple containing:**
  + **The number of iterations performed.**
  + **The approximate root.**
  + **The function value at the root.**
  + **The error.**
  + **A table showing the iteration history (iteration, current estimate, function value, and error).**
* **If unsuccessful, a message such as "ran out of iterations" or "dividing by 0" with the table of iterations.**

**Multiple Roots Method**

**Purpose:  
Finds a root of a function with possible multiplicity using higher-order derivatives for refinement.**

**Inputs:**

* **function: A string representing the algebraic function to analyze.**
* **x0: Initial estimate of the root.**
* **tolerance: The stopping criterion for the error.**
* **N: Maximum number of iterations.**

**Outputs:**

* **A tuple containing:**
  + **The number of iterations performed.**
  + **The approximate root.**
  + **The function value at the root.**
  + **The error.**
  + **A table showing the iteration history (iteration, current estimate, function value, and error).**
* **If unsuccessful, a message such as "ran out of iterations" with the table of iterations.**

**Gaussian Elimination (Simple)**

**Purpose:  
Solves a system of linear equations using simple Gaussian elimination without pivoting.**

**Inputs:**

* **A: Coefficient matrix (list of lists or NumPy array).**
* **b: Right-hand side vector (list or NumPy array).**

**Outputs:**

* **A NumPy array representing the solution vector.**

**Gaussian Elimination with Partial Pivoting**

**Purpose:  
Solves a system of linear equations using Gaussian elimination with partial pivoting to reduce numerical errors by swapping rows.**

**Inputs:**

* **A: Coefficient matrix (list of lists or NumPy array).**
* **b: Right-hand side vector (list or NumPy array).**

**Outputs:**

* **A NumPy array representing the solution vector.**

**Example Usage:**

**Gaussian Elimination with Total Pivoting**

**Purpose:  
Solves a system of linear equations using Gaussian elimination with total pivoting, which swaps both rows and columns to ensure the most numerically stable operations.**

**Inputs:**

* **A: Coefficient matrix (list of lists or NumPy array).**
* **b: Right-hand side vector (list or NumPy array).**

**Outputs:**

* **A NumPy array representing the solution vector.**

**Secant Method**

**Purpose:  
Finds an approximate root of a function using the secant method, which iteratively updates the root estimate using the function values at two initial points.**

**Inputs:**

* **x0: Initial estimate of the root.**
* **x1: Second estimate of the root.**
* **tolerance: The stopping criterion for the error.**
* **N: Maximum number of iterations.**
* **function: A string representing the algebraic function to analyze.**

**Outputs:**

* **A tuple containing:**
  + **The number of iterations performed.**
  + **The approximate root.**
  + **The function value at the root.**
  + **The error.**
  + **A table showing the iteration history (iteration, current estimate, function value, and error).**
* **If unsuccessful, a message such as "ran out of iterations" or "dividing by 0" with the table of iterations.**

**Multiple Roots Method**

**Purpose:  
Finds a root of a function with possible multiplicity using higher-order derivatives for refinement.**

**Inputs:**

* **function: A string representing the algebraic function to analyze.**
* **x0: Initial estimate of the root.**
* **tolerance: The stopping criterion for the error.**
* **N: Maximum number of iterations.**

**Outputs:**

* **A tuple containing:**
  + **The number of iterations performed.**
  + **The approximate root.**
  + **The function value at the root.**
  + **The error.**
  + **A table showing the iteration history (iteration, current estimate, function value, and error).**
* **If unsuccessful, a message such as "ran out of iterations" with the table of iterations.**

**Gaussian Elimination (Simple)**

**Purpose:  
Solves a system of linear equations using simple Gaussian elimination without pivoting.**

**Inputs:**

* **A: Coefficient matrix (list of lists or NumPy array).**
* **b: Right-hand side vector (list or NumPy array).**

**Outputs:**

* **A NumPy array representing the solution vector.**

**Gaussian Elimination with Partial Pivoting**

**Purpose:  
Solves a system of linear equations using Gaussian elimination with partial pivoting to reduce numerical errors by swapping rows.**

**Inputs:**

* **A: Coefficient matrix (list of lists or NumPy array).**
* **b: Right-hand side vector (list or NumPy array).**

**Outputs:**

* **A NumPy array representing the solution vector.**

**Gaussian Elimination with Total Pivoting**

**Purpose:  
Solves a system of linear equations using Gaussian elimination with total pivoting, which swaps both rows and columns to ensure the most numerically stable operations.**

**Inputs:**

* **A: Coefficient matrix (list of lists or NumPy array).**
* **b: Right-hand side vector (list or NumPy array).**

**Outputs:**

* **A NumPy array representing the solution vector.**

**Secant Method**

**Purpose:  
Finds an approximate root of a function using the secant method, which iteratively updates the root estimate using the function values at two initial points.**

**Inputs:**

* **x0: Initial estimate of the root.**
* **x1: Second estimate of the root.**
* **tolerance: The stopping criterion for the error.**
* **N: Maximum number of iterations.**
* **function: A string representing the algebraic function to analyze.**

**Outputs:**

* **A tuple containing:**
  + **The number of iterations performed.**
  + **The approximate root.**
  + **The function value at the root.**
  + **The error.**
  + **A table showing the iteration history (iteration, current estimate, function value, and error).**
* **If unsuccessful, a message such as "ran out of iterations" or "dividing by 0" with the table of iterations.**

**Multiple Roots Method**

**Purpose:  
Finds a root of a function with possible multiplicity using higher-order derivatives for refinement.**

**Inputs:**

* **function: A string representing the algebraic function to analyze.**
* **x0: Initial estimate of the root.**
* **tolerance: The stopping criterion for the error.**
* **N: Maximum number of iterations.**

**Outputs:**

* **A tuple containing:**
  + **The number of iterations performed.**
  + **The approximate root.**
  + **The function value at the root.**
  + **The error.**
  + **A table showing the iteration history (iteration, current estimate, function value, and error).**
* **If unsuccessful, a message such as "ran out of iterations" with the table of iterations.**

**Gaussian Elimination (Simple)**

**Purpose:  
Solves a system of linear equations using simple Gaussian elimination without pivoting.**

**Inputs:**

* **A: Coefficient matrix (list of lists or NumPy array).**
* **b: Right-hand side vector (list or NumPy array).**

**Outputs:**

* **A NumPy array representing the solution vector.**

**Example Usage:**

**Gaussian Elimination with Partial Pivoting**

**Purpose:  
Solves a system of linear equations using Gaussian elimination with partial pivoting to reduce numerical errors by swapping rows.**

**Inputs:**

* **A: Coefficient matrix (list of lists or NumPy array).**
* **b: Right-hand side vector (list or NumPy array).**

**Outputs:**

* **A NumPy array representing the solution vector.**

**Gaussian Elimination with Total Pivoting**

**Purpose:  
Solves a system of linear equations using Gaussian elimination with total pivoting, which swaps both rows and columns to ensure the most numerically stable operations.**

**Inputs:**

* **A: Coefficient matrix (list of lists or NumPy array).**
* **b: Right-hand side vector (list or NumPy array).**

**Outputs:**

* **A NumPy array representing the solution vector.**

**Secant Method**

**Purpose:  
Finds an approximate root of a function using the secant method, which iteratively updates the root estimate using the function values at two initial points.**

**Inputs:**

* **x0: Initial estimate of the root.**
* **x1: Second estimate of the root.**
* **tolerance: The stopping criterion for the error.**
* **N: Maximum number of iterations.**
* **function: A string representing the algebraic function to analyze.**

**Outputs:**

* **A tuple containing:**
  + **The number of iterations performed.**
  + **The approximate root.**
  + **The function value at the root.**
  + **The error.**
  + **A table showing the iteration history (iteration, current estimate, function value, and error).**
* **If unsuccessful, a message such as "ran out of iterations" or "dividing by 0" with the table of iterations.**

**Multiple Roots Method**

**Purpose:  
Finds a root of a function with possible multiplicity using higher-order derivatives for refinement.**

**Inputs:**

* **function: A string representing the algebraic function to analyze.**
* **x0: Initial estimate of the root.**
* **tolerance: The stopping criterion for the error.**
* **N: Maximum number of iterations.**

**Outputs:**

* **A tuple containing:**
  + **The number of iterations performed.**
  + **The approximate root.**
  + **The function value at the root.**
  + **The error.**
  + **A table showing the iteration history (iteration, current estimate, function value, and error).**
* **If unsuccessful, a message such as "ran out of iterations" with the table of iterations**

**Gaussian Elimination (Simple)**

**Purpose:  
Solves a system of linear equations using simple Gaussian elimination without pivoting.**

**Inputs:**

* **A: Coefficient matrix (list of lists or NumPy array).**
* **b: Right-hand side vector (list or NumPy array).**

**Outputs:**

* **A NumPy array representing the solution vector.**

**Gaussian Elimination with Partial Pivoting**

**Purpose:  
Solves a system of linear equations using Gaussian elimination with partial pivoting to reduce numerical errors by swapping rows.**

**Inputs:**

* **A: Coefficient matrix (list of lists or NumPy array).**
* **b: Right-hand side vector (list or NumPy array).**

**Outputs:**

* **A NumPy array representing the solution vector.**

**Gaussian Elimination with Total Pivoting**

**Purpose:  
Solves a system of linear equations using Gaussian elimination with total pivoting, which swaps both rows and columns to ensure the most numerically stable operations.**

**Inputs:**

* **A: Coefficient matrix (list of lists or NumPy array).**
* **b: Right-hand side vector (list or NumPy array).**

**Outputs:**

* **A NumPy array representing the solution vector.**

**Secant Method**

**Purpose:  
Finds an approximate root of a function using the secant method, which iteratively updates the root estimate using the function values at two initial points.**

**Inputs:**

* **x0: Initial estimate of the root.**
* **x1: Second estimate of the root.**
* **tolerance: The stopping criterion for the error.**
* **N: Maximum number of iterations.**
* **function: A string representing the algebraic function to analyze.**

**Outputs:**

* **A tuple containing:**
  + **The number of iterations performed.**
  + **The approximate root.**
  + **The function value at the root.**
  + **The error.**
  + **A table showing the iteration history (iteration, current estimate, function value, and error).**
* **If unsuccessful, a message such as "ran out of iterations" or "dividing by 0" with the table of iterations**

**Multiple Roots Method**

**Purpose:  
Finds a root of a function with possible multiplicity using higher-order derivatives for refinement.**

**Inputs:**

* **function: A string representing the algebraic function to analyze.**
* **x0: Initial estimate of the root.**
* **tolerance: The stopping criterion for the error.**
* **N: Maximum number of iterations.**

**Outputs:**

* **A tuple containing:**
  + **The number of iterations performed.**
  + **The approximate root.**
  + **The function value at the root.**
  + **The error.**
  + **A table showing the iteration history (iteration, current estimate, function value, and error).**
* **If unsuccessful, a message such as "ran out of iterations" with the table of iterations.**

**Gaussian Elimination (Simple)**

**Purpose:  
Solves a system of linear equations using simple Gaussian elimination without pivoting.**

**Inputs:**

* **A: Coefficient matrix (list of lists or NumPy array).**
* **b: Right-hand side vector (list or NumPy array).**

**Outputs:**

* **A NumPy array representing the solution vector.**

**Gaussian Elimination with Partial Pivoting**

**Purpose:  
Solves a system of linear equations using Gaussian elimination with partial pivoting to reduce numerical errors by swapping rows.**

**Inputs:**

* **A: Coefficient matrix (list of lists or NumPy array).**
* **b: Right-hand side vector (list or NumPy array).**

**Outputs:**

* **A NumPy array representing the solution vector.**

**Gaussian Elimination with Total Pivoting**

**Purpose:  
Solves a system of linear equations using Gaussian elimination with total pivoting, which swaps both rows and columns to ensure the most numerically stable operations.**

**Inputs:**

* **A: Coefficient matrix (list of lists or NumPy array).**
* **b: Right-hand side vector (list or NumPy array).**

**Outputs:**

* **A NumPy array representing the solution vector.**

**Gaussian Elimination with Partial Pivoting**

**Purpose:  
Solves a system of linear equations using Gaussian elimination with partial pivoting to reduce numerical errors by swapping rows.**

**Inputs:**

* **A: Coefficient matrix (list of lists or NumPy array).**
* **b: Right-hand side vector (list or NumPy array).**

**Outputs:**

* **A NumPy array representing the solution vector.**

**Gaussian Elimination with Total Pivoting**

**Purpose:  
Solves a system of linear equations using Gaussian elimination with total pivoting, which swaps both rows and columns to ensure the most numerically stable operations.**

**Inputs:**

* **A: Coefficient matrix (list of lists or NumPy array).**
* **b: Right-hand side vector (list or NumPy array).**

**Outputs:**

* **A NumPy array representing the solution vector.**