# Sutra Decomposition Engine

A Python framework for mathematical formalization-deformalization using Vedic sutras and hypercube transformations.

## Overview

The Sutra Decomposition Engine is a sophisticated mathematical system that combines ancient Vedic mathematical techniques with modern computational approaches to transform, simplify, and analyze mathematical expressions. The engine implements a complete pipeline for processing mathematical expressions using pattern recognition, hypercube transformations, and machine learning.

### Key Features

- \*\*Pattern-based Transformation\*\*: Apply Vedic sutra patterns to simplify mathematical expressions

- \*\*Hypercube Transformations\*\*: Map expressions to higher-dimensional spaces using hypercube operators

- \*\*Pattern Learning\*\*: Discover new transformation patterns from examples

- \*\*Advanced Visualizations\*\*: Generate interactive visualizations of the transformation process

- \*\*Command-line Interface\*\*: Access all functionality from the terminal

- \*\*Interactive Notebook\*\*: Explore the system through a Jupyter notebook

## Installation

### Prerequisites

- Python 3.7+

- NumPy

- SymPy

- Matplotlib

- NetworkX

- scikit-learn (for pattern learning)

- docopt (for CLI)

### Installing from Source

```bash

# Clone the repository

git clone https://github.com/yourusername/sutra-decomposition-engine.git

cd sutra-decomposition-engine

# Install the package in development mode

pip install -e .

```

### Package Installation (when available)

```bash

pip install sutra-decomposition-engine

```

## Quick Start

### Using the Python API

```python

from sutra\_engine import EnhancedSutraDecompositionEngine

# Initialize the engine

engine = EnhancedSutraDecompositionEngine()

# Process an expression

result = engine.process\_expression("(x + a)\*(x - a)")

# Display the result

print(f"Original: {result['original']['str']}")

print(f"Transformed: {result['transformed']['str']}")

```

### Using the Command-Line Interface

```bash

# Transform an expression

sutra transform "(x + a)\*(x - a)" --output=markdown --save=result.md

# Visualize a transformation

sutra visualize "(x + a)\*(x - a)" --type=transform --save=visualization.png

# Process a batch of expressions

sutra batch expressions.txt --output=json --save=results.json

# List available sutras

sutra list-sutras

```

### Using the Interactive Notebook

```bash

# Launch the interactive notebook

jupyter notebook demo\_notebook.ipynb

```

## Documentation

### Core Components

1. \*\*EnhancedSutraDecompositionEngine\*\*: The main engine for processing mathematical expressions

2. \*\*SutraLibrary\*\*: Collection of Vedic sutras and transformation patterns

3. \*\*SutraPatternLearner\*\*: System for learning new transformation patterns from examples

4. \*\*SutraTransformationVisualizer\*\*: Tools for visualizing the transformation process

5. \*\*SutraCLI\*\*: Command-line interface for the engine

### Available Sutras

The engine implements several Vedic sutras, including:

- \*\*S1: Ekadhikena Purvena\*\* - For squaring numbers ending in 5

- \*\*S2: Nikhilam Navatashcaramam Dashatah\*\* - All from 9 and the last from 10

- \*\*S3: Urdhva-Tiryagbhyam\*\* - Vertically and crosswise multiplication

- \*\*S11: Purna-Purna\*\* - Difference of squares pattern

### Hypercube Transformation

The hypercube transformation maps mathematical operations to higher-dimensional spaces using the following parameters:

- \*\*dimension\*\*: Dimension of the hypercube (default: 4)

- \*\*chi\*\*: Invariant parameter controlling matrix scaling (default: 0.5)

- \*\*alpha\*\*: Alloy weights for diagonal elements (default: uniform weights)

- \*\*lambda0\*\*: Scalar parameter for adjacency matrix (default: 1.0)

## Examples

### Difference of Squares (Purna-Purna Sutra)

```python

engine = EnhancedSutraDecompositionEngine()

result = engine.process\_expression("(x + a)\*(x - a)")

# Result: x\*\*2 - a\*\*2

```

### Square of Number Ending in 5 (Ekadhikena Sutra)

```python

engine = EnhancedSutraDecompositionEngine()

result = engine.process\_expression("25\*\*2")

# Result: 625

```

### Multiplication Near Base (Nikhilam Sutra)

```python

engine = EnhancedSutraDecompositionEngine()

result = engine.process\_expression("98\*97")

# Result: 9506

```

## Project Structure

```

sutra\_engine/

├── \_\_init\_\_.py

├── enhanced\_sutra\_engine.py # Core engine implementation

├── sutra\_advanced\_components.py # Pattern learning and visualization

├── sutra\_testing\_framework.py # Testing framework

├── sutra\_cli.py # Command-line interface

│

├── examples/

│ ├── example\_transformations.json # Example transformations

│ └── expressions.txt # Sample expressions for batch processing

│

├── notebooks/

│ └── demo\_notebook.ipynb # Interactive demonstration notebook

│

├── tests/

│ ├── \_\_init\_\_.py

│ ├── test\_engine.py

│ ├── test\_pattern\_learning.py

│ └── test\_visualization.py

│

├── docs/

│ ├── sutras.md # Documentation of implemented sutras

│ ├── hypercube\_transformation.md # Details on hypercube transformations

│ ├── pattern\_learning.md # Guide to pattern learning

│ └── cli\_usage.md # Command-line interface usage guide

│

├── setup.py # Package setup script

└── README.md # Project README

```

## Development

### Running Tests

```bash

# Run all tests

pytest tests/

# Run specific test module

pytest tests/test\_engine.py

```

### Building Documentation

```bash

# Install documentation dependencies

pip install sphinx sphinx\_rtd\_theme

# Build documentation

cd docs

make html

```

## Contributing

Contributions are welcome! Please feel free to submit a Pull Request.

## License

This project is licensed under the MIT License - see the LICENSE file for details.

## Acknowledgments

- The Vedic mathematics tradition for the sutra patterns

- The SymPy project for symbolic mathematics capabilities

- The scikit-learn project for machine learning components