Comprehensive Theory of Everything

Complete Documentation

Theory of Everything: Complete Documentation

The following sections are from DOCUMENTATION.md.

Introduction

Introduction

The Theory of Everything (ToE) is an ambitious computational implementation that unifies all fundamental forces and matter interactions into a single coherent mathematical framework. This project combines advanced physics concepts from quantum field theory, general relativity, particle physics, and cosmology with computational visualization techniques to explore and demonstrate the mathematical structure of the universe. This documentation provides a comprehensive guide to understanding, using, and extending the Theory of Everything codebase.

Table of Contents

Core Concepts

The Theory of Everything presented in this project is built around a unified action principle, expressed as: $$S = S_{\text{gravity}} + S_{\text{matter}} + S_{\text{gauge}} + S_{\text{quantum}}$$ This master equation integrates four fundamental components: - **Gravity Action** (S_{gravity}): Describes how spacetime curves in response to energy and matter - **Matter Action** (S_{matter}): Describes all forms of matter and their interactions with spacetime - **Gauge Field Action** (S_{gauge}): Describes the fundamental forces (electromagnetic, strong, and weak) - **Quantum Corrections** (S_{quantum}): Accounts for quantum fluctuations and virtual particles Each component is implemented as a separate module with classes for different formulations and methods for calculations and visualizations.

Mathematical Framework

Gravity Action The gravity action includes three main formulations: 1. **Einstein-Hilbert Action (Classical Gravity)**: ``` S_gravity^EH = 1/(16πG) ∫d■x √(-g) (R - 2Λ) ``` Where: - G is Newton's gravitational constant - g is the determinant of the metric tensor - R is the Ricci scalar curvature - Λ is the cosmological constant 2. **Loop Quantum Gravity Action**: ``` S_gravity^LQG = 1/(8πG) ∫d■x √(-g) ε^abc E_a^i E_b^j F_ij^c ``` Where: - ε^abc is the Levi-Civita symbol - E_a^i are the densitized triads (gravitational electric field) - F_ij^c is the curvature of the Ashtekar connection 3. **String Theory Gravity Action**: ``` S_gravity^String = 1/(2κ²) ∫d^10x √(-g) e^{-2ϕ} [R + 4(∇ϕ)² - (1/12)H_μνρ H^μνρ] ``` Where: - κ is related to the string tension - ϕ is the dilaton field - H_μνρ is the field strength of the Kalb-Ramond field ### Matter Action The matter action includes two main components: 1. **Fermion Fields (Dirac Action)**: ``` S_fermion = ∫d■x √(-g) ψ■(iγ^μ D_μ - m)ψ ``` Where: - ψ is the fermion field - ψ■ is the Dirac adjoint - γ^μ are the Dirac gamma matrices - D_μ is the covariant derivative - m is the mass of the fermion 2. **Higgs Field (Spontaneous Symmetry Breaking)**: ``` S_Higgs = ∫d■x √(-g) [(D_μϕ)†(D^μϕ) - V(ϕ)] ``` Where: - ϕ is the Higgs field - D_μ is the covariant derivative - V(ϕ) = -μ²|ϕ|² + λ|ϕ|■ is the Higgs potential (the "Mexican hat" potential) ### Gauge Field Action The gauge field action includes two main components: 1. **Yang-Mills Action (Non-Abelian Gauge Fields)**: ``` S_gauge = -1/4 ∫d■x √(-g) F_μν^a F^μν_a ``` Where: - F_μν^a is the field strength tensor - For SU(3): F_μν^a = ∂_μA_ν^a - ∂_νA_μ^a + g f^abc A_μ^b A_ν^c 2. **Supersymmetric Gauge Fields**: ``` S_SUSY-gauge = ∫d■x [-1/4 F_μν F^μν + i λ■γ^μ D_μλ] ``` Where: - λ is the gaugino (supersymmetric partner of the gauge boson) - λ■ is the Dirac adjoint ### Quantum Corrections The quantum corrections include two main components: 1. **Path Integral Formulation**: ``` Z = ∫■ϕ e^{iS[ϕ]} ``` Where: - Z is the partition function - ■ϕ represents the functional integration measure over all field configurations - S[ϕ] is the action functional 2. **Loop Corrections and Renormalization**: ``` S_quantum = ∑_{n=1}^∞ ■^n S_n ``` Where: - ■ is the reduced Planck constant - S_n represents the n-loop quantum correction to the classical action ### Full Master Equation The complete unified action is given by: ``` S = 1/(16πG) ∫d■x √(-g) (R - 2Λ) + ∫d■x √(-g) [ψ■(iγ^μ D_μ - m)ψ + (D_μϕ)†(D^μϕ) - V(ϕ) - 1/4 F_μν^a F^μν_a] + ∑_{n=1}^∞ ■^n S_n ``` This equation combines all fundamental interactions into a single mathematical framework.

Project Structure

The project is organized into several key components: ### Core Modules - **math/toe.py**: Core implementation of the Theory of Everything - `TheoryOfEverything`: Main class implementing the unified action and core physics calculations - `QuantumGeometry`: Class for quantum aspects of spacetime geometry - `UnifiedForces`: Class for modeling the unification of fundamental forces - **math/toe_formulas.py**: Symbolic representation and visualization of formulas - Uses SymPy for symbolic mathematics - Provides methods for displaying and manipulating equations - **math/schumann.py**: Implementation of Schumann resonances - `SchumannResonance`: Class for modeling and visualizing Earth-ionosphere cavity resonances ### Component Formulas - **component_formulas/gravity_action.py**: Implementation of gravity action components - `EinsteinHilbertAction`: Classical gravity through the Einstein-Hilbert action - `LoopQuantumGravity`: Quantum gravity through loop variables - `StringTheoryGravity`: Higher-dimensional gravity from string theory - **component_formulas/matter_action.py**: Implementation of matter action components - `FermionAction`: Dirac action for fermion fields - `HiggsAction`: Higgs field with spontaneous symmetry breaking - **component_formulas/gauge_action.py**: Implementation of gauge field action components - `YangMillsAction`: Non-Abelian gauge fields for strong and weak forces - `SupersymmetricGaugeAction`: Supersymmetric extension with gauginos - **component_formulas/quantum_corrections.py**: Implementation of quantum corrections - `PathIntegral`: Path integral formulation of quantum field theory - `LoopCorrections`: Loop corrections and renormalization - **component_formulas/unified_action.py**: Unified interface for all components - `UnifiedAction`: Combines all components and provides methods for

exploring the complete theory ### User Interfaces - **visualize_toe.py**: Main visualization interface - Menu-driven interface to explore all aspects of the Theory of Everything - **explore_toe_formulas.py**: Interface for exploring component formulas - Provides access to all component formulas and their visualizations - **create_toe_pdf.py**: Tool for generating PDF documentation - Creates a PDF with properly formatted equations - **generate_latex_toe.py**: Tool for generating LaTeX documentation - Creates a LaTeX document with professional typesetting ### Documentation - **README.md**: Overview of the project and key equations - **EXPLANATION.md**: Comprehensive explanation of the Theory of Everything - **DOCUMENTATION.md**: Complete documentation of the codebase (this file) - **FORMULA_IMPROVEMENTS.md**: Summary of improvements to maintain formula integrity - **Everything_ToE.md**: Detailed explanation of the Theory of Everything - **toe.md**: Detailed explanation of the unified action master equation

Installation and Requirements

Prerequisites The project requires the following Python libraries: - NumPy: For numerical calculations - SciPy: For scientific computing and integration - SymPy: For symbolic mathematics - Matplotlib: For visualization - IPython/Jupyter (optional): For interactive features ### Installation 1. Clone the repository: ``` git clone https://github.com/your-username/theoryofeverything.git cd theoryofeverything ``` 2. Install the required dependencies: ``` pip install numpy scipy sympy matplotlib ``` 3. (Optional) For PDF generation with LaTeX: - Install a LaTeX distribution (e.g., TeX Live, MiKTeX) - Ensure pdflatex is available in your PATH

Usage Guide

Exploring the Theory of Everything 1. Run the main visualization interface: ``` python visualize_toe.py ``` This provides a menu-driven interface to explore all aspects of the Theory of Everything. 2. Explore the component formulas: ``` python explore_toe_formulas.py ``` This allows you to explore each component formula individually. 3. Explore specific components: - For formula visualization: `python demonstrate_formulas.py` - For Schumann resonances: `python math/schumann.py` - For Theory of Everything core visualizations: `python math/toe.py` ### Generating Documentation 1. Create a PDF with properly formatted equations: ``` python create_toe_pdf.py ``` This allows you to choose between matplotlib-based or LaTeX-based PDF generation. 2. Generate a LaTeX document: ``` python generate_latex_toe.py ``` This creates a LaTeX document with professional typesetting.

Component Formulas

The component formulas are implemented in the `component_formulas` directory. Each component is implemented as a separate module with classes for different formulations and methods for calculations and visualizations. ### Gravity Action The gravity action is implemented in `component_formulas/gravity_action.py`. It includes three main classes: 1. `EinsteinHilbertAction`: Implements the Einstein-Hilbert action for classical gravity. - Methods for calculating the action, field equations, and visualizing spacetime curvature. 2. `LoopQuantumGravity`: Implements the Loop Quantum Gravity extension. - Methods for visualizing quantum foam, calculating area spectrum, and visualizing area eigenvalues. 3. `StringTheoryGravity`: Implements the String/M-Theory gravity. - Methods for visualizing string worldsheets and compactified extra dimensions. ### Matter Action The matter action is implemented in `component_formulas/matter_action.py`. It includes two main classes: 1. `FermionAction`: Implements the Dirac action for fermion fields. - Methods for calculating the Dirac equation, visualizing spinor fields, and visualizing the Dirac sea. 2. `HiggsAction`: Implements the Higgs field with spontaneous symmetry breaking.

- Methods for calculating the Higgs potential, visualizing the Mexican hat potential, and visualizing symmetry breaking. ### Gauge Field Action The gauge field action is implemented in `component_formulas/gauge_action.py`. It includes two main classes: 1. `YangMillsAction`: Implements the Yang-Mills action for non-Abelian gauge fields. - Methods for calculating the field strength tensor, visualizing field strength, and visualizing force unification. 2. `SupersymmetricGaugeAction`: Implements the supersymmetric extension of gauge fields. - Methods for visualizing supersymmetry multiplets and supersymmetry breaking. ### Quantum Corrections The quantum corrections are implemented in `component_formulas/quantum_corrections.py`. It includes two main classes: 1. `PathIntegral`: Implements the path integral formulation of quantum field theory. - Methods for Monte Carlo calculation of path integrals, visualizing paths, and visualizing Feynman diagrams. 2. `LoopCorrections`: Implements loop corrections and renormalization. - Methods for calculating loop corrections, visualizing quantum corrections, and visualizing renormalization flow. ### Unified Action The unified action is implemented in `component_formulas/unified_action.py`. It includes the `UnifiedAction` class that combines all components and provides methods for exploring the complete theory.

Visualization Techniques

The project employs several advanced visualization techniques: ### 3D Visualizations - **Spacetime Curvature**: Visualizes how mass curves spacetime in general relativity. - **Quantum Foam**: Visualizes spacetime fluctuations at the Planck scale. - **String Worldsheets**: Visualizes the evolution of strings in spacetime. - **Higgs Potential**: Visualizes the Mexican hat potential in 3D. ### 2D Plots - **Force Unification**: Visualizes how the coupling constants of the fundamental forces converge at high energies. - **Quantum Corrections**: Visualizes how quantum corrections modify the classical action. - **Area Spectrum**: Visualizes the discrete spectrum of area eigenvalues in Loop Quantum Gravity. - **Renormalization Flow**: Visualizes how coupling constants change with energy scale. ### Formula Visualization - **Symbolic Representation**: Uses SymPy to display formulas with proper mathematical notation. - **Relationship Diagrams**: Visualizes the relationships between different components of the theory. - **Unified Theory Structure**: Visualizes the hierarchical structure of the unified theory.

Theoretical Implications

The Theory of Everything presented in this project has several profound implications: ### Unified Physical Laws All forces and matter fields are combined into a single framework, showing how they emerge from a common mathematical structure. This unification provides a deeper understanding of the fundamental nature of reality. ### Quantum Gravity Spacetime is quantized and emergent from more fundamental structures, resolving the conflict between general relativity and quantum mechanics. This approach addresses the long-standing problem of reconciling gravity with quantum theory. ### Supersymmetry A fundamental symmetry balances matter and force particles, potentially explaining the hierarchy problem and providing dark matter candidates. Supersymmetry introduces a new symmetry between bosons and fermions. ### Dark Matter/Energy Quantum spacetime fluctuations and supersymmetric particles provide natural explanations for dark matter and dark energy. These phenomena, which constitute most of the universe's energy content, emerge naturally from the theory. ### Origin of the Universe The unified framework provides a mathematical basis for understanding the beginning and evolution of the cosmos. It offers insights into the initial conditions and subsequent evolution of the universe.

Extending the Project

The modular structure of the project makes it easy to extend: ### Adding New Physics Components To add a new physics component, create a new class that implements the relevant physics. For example, to add a new gravity formulation: ```python class NewGravityFormulation: def __init__(self): # Initialize parameters def calculate_action(self, parameters): # Calculate the action def visualize_results(self): # Visualize the results ``` ### Implementing Additional Visualization Methods To add a new visualization method, add a new method to the relevant class: ```python def visualize_new_aspect(self, parameters): # Create the visualization plt.figure() # ... visualization code ... plt.show() ``` ### Exploring Different Parameter Regimes To explore different parameter regimes, modify the parameters in the existing methods: ```python result = calculate_something(param1=1.0, param2=2.0) result = calculate_something(param1=10.0, param2=20.0) ```

Formula Improvements

We have made several improvements to maintain the mathematical integrity of the formulas in the Theory of Everything codebase: ### Gravity Action Components - Added detailed notes explaining each formula - Clarified the meaning of variables and parameters - Ensured proper representation of mathematical symbols ### Matter Action Components - Enhanced the description of the Higgs potential - Explained the role of spontaneous symmetry breaking - Ensured proper representation of Dirac gamma matrices ### Gauge Field Action Components - Included the explicit form of the field strength tensor - Clarified the role of gauge symmetry - Explained the relationship between gauge fields and forces ### Quantum Corrections Components - Clarified the meaning of the path integral measure - Explained the role of loop corrections in quantum field theory - Ensured proper representation of quantum corrections ### Unified Action - Enhanced the full master equation with proper quantum corrections term - Added explanatory notes for each component of the unified action - Ensured mathematical consistency across all terms

References

1. Weinberg, S. (1995). The Quantum Theory of Fields. Cambridge University Press. 2. Rovelli, C. (2004). Quantum Gravity. Cambridge University Press. 3. Green, M. B., Schwarz, J. H., & Witten, E. (1987). Superstring Theory. Cambridge University Press. 4. Peskin, M. E., & Schroeder, D. V. (1995). An Introduction to Quantum Field Theory. Westview Press. 5. Wald, R. M. (1984). General Relativity. University of Chicago Press. 6. Zwiebach, B. (2009). A First Course in String Theory. Cambridge University Press. 7. Srednicki, M. (2007). Quantum Field Theory. Cambridge University Press. 8. Ashtekar, A., & Lewandowski, J. (2004). Background Independent Quantum Gravity: A Status Report. Classical and Quantum Gravity, 21(15), R53. 9. Polchinski, J. (1998). String Theory. Cambridge University Press. 10. Witten, E. (1995). String Theory Dynamics in Various Dimensions. Nuclear Physics B, 443(1-2), 85-126.

The Theory of Everything: Project Explanation

The following sections are from EXPLANATION.md.

Introduction

Overview

This project is an ambitious computational implementation of a Grand Unified Theory of Everything (ToE), which attempts to unify all fundamental forces and matter interactions into a single coherent mathematical framework. The project combines advanced physics concepts from quantum field theory, general relativity, particle physics, and cosmology with computational visualization techniques to explore and demonstrate the mathematical structure of the universe.

Core Concepts

The Theory of Everything presented in this project is built around a unified action principle, expressed as: $$S = S_{\text{gravity}} + S_{\text{matter}} + S_{\text{gauge}} + S_{\text{quantum}}$$ This master equation integrates four fundamental components: - **Gravity Action** (S_{gravity}): Describes how spacetime curves in response to energy and matter - **Matter Action** (S_{matter}): Describes all forms of matter and their interactions with spacetime - **Gauge Field Action** (S_{gauge}): Describes the fundamental forces (electromagnetic, strong, and weak) - **Quantum Corrections** (S_{quantum}): Accounts for quantum fluctuations and virtual particles

Project Structure

The project is organized into several key components: ### Core Physics Implementation (`math/toe.py`) This file contains the primary implementation of the Theory of Everything, with three main classes: - **`TheoryOfEverything`**: Implements the unified action and core physics calculations - Initializes physical constants and fields - Implements methods for calculating quantum corrections - Provides visualization methods for spacetime curvature and quantum effects - **`QuantumGeometry`**: Handles quantum aspects of spacetime geometry - Implements quantum metric operators - Calculates eigenvalues and eigenvectors of quantum spacetime - Visualizes quantum foam (spacetime fluctuations at Planck scale) - **`UnifiedForces`**: Models the unification of fundamental forces - Implements running coupling constants - Calculates force unification at high energies - Visualizes the relative strengths of forces ### Formula Visualization (`math/toe_formulas.py`) This module provides comprehensive visualization and exploration of all mathematical formulas in the Theory of Everything: - Uses symbolic mathematics (SymPy) to represent and manipulate equations - Displays formulas with proper mathematical notation - Visualizes relationships between different components of the theory - Provides an interactive formula explorer with explanations ### Schumann Resonance Implementation (`math/schumann.py`) This module explores Schumann resonances, which are global electromagnetic resonances in the Earth-ionosphere cavity: - Calculates resonant frequencies and wave properties - Visualizes resonance modes in time and frequency domains - Creates 3D visualizations of the Earth-ionosphere cavity - Models wave propagation and standing waves ### Visualization Interface (`visualize_toe.py`) This script provides a unified interface for exploring all aspects of the Theory of Everything: - Offers a menu-driven interface to access all visualizations - Organizes visualizations into logical categories - Provides access to formula exploration, force unification, quantum gravity, and Schumann resonances ### PDF Generation Tools The project includes tools for generating properly formatted PDF documentation: - `generate_toe_pdf.py`: Creates a PDF using matplotlib for rendering equations - `generate_latex_toe.py`: Creates a LaTeX document with professional typesetting - `create_toe_pdf.py`: Provides a user-friendly interface for choosing PDF generation methods ### Documentation The project includes several Markdown files that explain different aspects of the theory: - `README.md`: Overview of the project and key equations - `Everything_ToE.md`: Comprehensive explanation of the Theory of Everything - `toe.md`: Detailed explanation of the unified action master equation - `Smatter/Smatter.md`: Explanation of the matter action component

Mathematical Framework

Gravity Action The gravity action includes three main formulations: **Einstein-Hilbert Action** (Classical Gravity): $$S_{\text{gravity}}^{\text{EH}} = \frac{1}{16\pi G} \int d^4x \, \sqrt{-g} \, (R - 2\Lambda)$$ **Loop Quantum Gravity Extension**: $$S_{\text{gravity}}^{\text{LQG}} = \frac{1}{8\pi G} \int d^4x \, \sqrt{-g} \, \epsilon^{abc} E_a^i E_b^j F_{ij}^c$$ **String/M-Theory Gravity**: $$S_{\text{gravity}}^{\text{String}} = \frac{1}{2\kappa^2} \int d^{10}x \, \sqrt{-g} \, e^{-2\phi} \left(R + 4 (\nabla \phi)^2 - \frac{1}{12} H_{\mu\nu\rho} H^{\mu\nu\rho}\right)$$ ### Matter Action The matter action describes fermions and the Higgs field: **Fermion Fields** (Dirac Action): $$S_{\text{fermion}} = \int d^4x \, \sqrt{-g} \, \bar{\psi} (i \gamma^\mu D_\mu - m) \psi$$ **Higgs Field** (Spontaneous Symmetry Breaking): $$S_{\text{Higgs}} = \int d^4x \, \sqrt{-g} \, \left[(D_\mu \phi)^\dagger (D^\mu \phi) - V(\phi) \right]$$ ### Gauge Field Action The gauge field action describes the fundamental forces: **Yang-Mills Action** (Non-Abelian Gauge Fields): $$S_{\text{gauge}} = -\frac{1}{4} \int d^4x \, \sqrt{-g} \, F_{\mu\nu}^a F^{\mu\nu}_a$$ **Supersymmetric Gauge Fields**: $$S_{\text{SUSY-gauge}} = \int d^4x \, \left[-\frac{1}{4} F_{\mu\nu} F^{\mu\nu} + i \bar{\lambda} \gamma^\mu D_\mu \lambda \right]$$ ### Quantum Corrections The quantum corrections account for quantum fluctuations: **Path Integral Formulation**: $$Z = \int \mathcal{D}\phi \, e^{i S[\phi]}$$ **Loop Corrections and Renormalization**: $$S_{\text{quantum}} = \sum_{n=1}^{\infty} \hbar^n S_n$$

Computational Implementation

Physical Constants The implementation uses accurate physical constants: - Gravitational constant (G): 6.67430 × 10■¹¹ m³/(kg·s²) - Speed of light (c): 299,792,458 m/s - Reduced Planck constant (■): 1.054571817 × 10■³■ J·s - Cosmological constant (Λ): 1.089 × 10■■² ### Key Computational Methods **Quantum Metric Calculation**: - Creates a quantum metric operator based on distances between points - Applies quantum corrections at the Planck scale - Computes eigenvalues and eigenvectors to analyze quantum spacetime structure **Force Unification Calculation**: - Implements renormalization group equations for coupling constants - Calculates running couplings across energy scales - Identifies the unification point where forces converge **Quantum Corrections**: - Implements loop integrals using Monte Carlo methods - Calculates n-loop quantum corrections to classical action - Accounts for combinatorial factors in Feynman diagrams **Schumann Resonance Calculation**: - Models the Earth-ionosphere cavity as a spherical waveguide - Calculates resonant frequencies based on Earth's circumference - Implements wave equations with damping for realistic modeling ### Visualization Techniques The project employs several advanced visualization techniques: **3D Visualizations**: - Spacetime curvature due to mass - Quantum foam (spacetime fluctuations) - Earth-ionosphere cavity for Schumann resonances - Higgs potential "Mexican hat" shape **2D Plots**: - Force unification across energy scales - Quantum corrections to classical action - Schumann resonance modes in time and frequency domains - Quantum metric eigenspectrum **Formula Visualization**: - Symbolic representation of equations - Relationship diagrams between different formulas - Interactive exploration of mathematical structure

Theoretical Implications

The Theory of Everything presented in this project has several profound implications: **Unified Physical Laws**: All forces and matter fields are combined into a single framework, showing how they emerge from a common mathematical structure. **Quantum Gravity**: Spacetime is quantized and emergent from more fundamental structures, resolving the conflict between general relativity and quantum mechanics. **Supersymmetry**: A fundamental symmetry balances matter and force particles, potentially explaining the

hierarchy problem and providing dark matter candidates. **Dark Matter/Energy**: Quantum spacetime fluctuations and supersymmetric particles provide natural explanations for dark matter and dark energy. **Origin of the Universe**: The unified framework provides a mathematical basis for understanding the beginning and evolution of the cosmos.

How to Use This Project

Exploring Visualizations Run the main visualization interface: ``` python visualize_toe.py ``` This provides a menu-driven interface to explore all aspects of the Theory of Everything. Explore specific components: - For formula visualization: `python demonstrate_formulas.py` - For Schumann resonances: `python math/schumann.py` - For Theory of Everything core visualizations: `python math/toe.py` ### Generating Documentation Create a PDF with properly formatted equations: ``` python create_toe_pdf.py ``` This allows you to choose between matplotlib-based or LaTeX-based PDF generation. ### Extending the Project The modular structure of the project makes it easy to extend: - Add new physics components by extending the existing classes - Implement additional visualization methods - Explore different parameter regimes or alternative formulations

Technical Requirements

The project requires the following Python libraries: - NumPy: For numerical calculations - SciPy: For scientific computing and integration - SymPy: For symbolic mathematics - Matplotlib: For visualization - IPython/Jupyter (optional): For interactive features

Conclusion

This project represents an ambitious attempt to computationally implement and visualize a Grand Unified Theory of Everything. While the complete unification of all physical forces remains an open challenge in theoretical physics, this implementation provides a framework for exploring the mathematical structure that might underlie such a unified theory. The combination of rigorous mathematical formulation with interactive visualization tools makes complex physics concepts more accessible and provides insights into the fundamental nature of reality. Whether used for educational purposes or as a starting point for further theoretical exploration, this project offers a unique computational perspective on the quest for a Theory of Everything.

Theory of Everything: User Guide

The following sections are from USER_GUIDE.md.

Introduction

This guide provides step-by-step instructions for using the Theory of Everything codebase, with a focus on the component formulas and their visualizations.

Table of Contents

Getting Started

Installation 1. Clone the repository: ``` git clone https://github.com/your-username/theoryofeverything.git cd theoryofeverything ``` 2. Install the required dependencies: ``` pip install numpy scipy sympy matplotlib ``` ### Quick Start To quickly get started with the Theory of Everything, run the main exploration script: ``` python explore_toe_formulas.py ``` This will launch the main menu interface that allows you to explore all aspects of the Theory of Everything.

Exploring Component Formulas

The Theory of Everything is composed of several component formulas, each implemented in a separate module. Here's how to explore each component: ### Unified Action To explore the unified action (master equation): 1. Run the exploration script: ``` python explore_toe_formulas.py ``` 2. Select option 1 from the main menu: ``` 1. Unified Action (Master Equation) ``` 3. This will display the unified action in LaTeX format: ``` S = S_gravity + S_matter + S_gauge + S_quantum ``` 4. Press Enter to see the full master equation, which expands all components. ### Gravity Action To explore the gravity action components: 1. Run the exploration script: ``` python explore_toe_formulas.py ``` 2. Select option 2 from the main menu: ``` 2. Gravity Action ``` 3. This will display the three main formulations of gravity: - Einstein-Hilbert Action (Classical Gravity) - Loop Quantum Gravity Extension - String/M-Theory Gravity 4. Select a visualization to display: - Spacetime Curvature (Einstein-Hilbert) - Quantum Foam (Loop Quantum Gravity) - Area Spectrum (Loop Quantum Gravity) - String Worldsheet (String Theory) - Extra Dimensions (String Theory) ### Matter Action To explore the matter action components: 1. Run the exploration script: ``` python explore_toe_formulas.py ``` 2. Select option 3 from the main menu: ``` 3. Matter Action ``` 3. This will display the two main components of matter: - Fermion Fields (Dirac Action) - Higgs Field (Spontaneous Symmetry Breaking) 4. Select a visualization to display: - Spinor Field (Fermion) - Dirac Sea (Fermion) - Higgs Potential 1D - Higgs Potential 2D (Mexican Hat) - Spontaneous Symmetry Breaking ### Gauge Field Action To explore the gauge field action components: 1. Run the exploration script: ``` python explore_toe_formulas.py ``` 2. Select option 4 from the main menu: ``` 4. Gauge Field Action ``` 3. This will display the two main components of gauge fields: - Yang-Mills Action (Non-Abelian Gauge Fields) - Supersymmetric Gauge Fields 4. Select a visualization to display: - Gauge Field Strength (Yang-Mills) - Force Unification (Yang-Mills) - Supersymmetry Multiplet (SUSY) - Supersymmetry Breaking (SUSY) ### Quantum Corrections To explore the quantum corrections components: 1. Run the exploration script: ``` python explore_toe_formulas.py ``` 2. Select option 5 from the main menu: ``` 5. Quantum Corrections ``` 3. This will display the two main components of quantum corrections: - Path Integral Formulation - Loop Corrections and Renormalization 4. Select a visualization to display: - Path Integral Paths - Feynman Diagram (Propagator) - Feynman Diagram (Vertex) - Feynman Diagram (Loop) - Loop Corrections - Renormalization Flow

Visualizing the Theory

The Theory of Everything provides several ways to visualize the unified theory and its implications: ### Unified Theory Structure To visualize the structure of the unified theory: 1. Run the exploration script: ``` python explore_toe_formulas.py ``` 2. Select option 6 from the main menu: ``` 6. Visualize Unified Theory Structure ``` 3. This will display a hierarchical visualization of the unified theory, showing how all components fit together. ### Theory Implications To visualize the implications of the Theory of Everything: 1. Run the exploration script: ``` python explore_toe_formulas.py ``` 2. Select option 7 from the main menu: ``` 7. Visualize Theory Implications ``` 3. This will display a visualization of the key implications of the Theory of Everything, including unified physical laws, quantum gravity, supersymmetry, dark matter/energy, and the origin of the universe.

Understanding the Mathematics

Each component of the Theory of Everything is based on rigorous mathematical formulations. Here's a guide to understanding the key mathematical concepts: ### Gravity Action The gravity action describes how spacetime curves in response to energy and matter: - **Einstein-Hilbert Action**: $S_{\text{gravity}}^{\text{EH}} = \frac{1}{16\pi G} \int d^4x \, \sqrt{-g} \, (R - 2\Lambda)$ - G is Newton's gravitational constant - g is the determinant of the metric tensor - R is the Ricci scalar curvature - Λ is the cosmological constant - **Loop Quantum Gravity**: $S_{\text{gravity}}^{\text{LQG}} = \frac{1}{8\pi G} \int d^4x \, \sqrt{-g} \, \epsilon^{abc} E_a^i E_b^j F_{ij}^c$ - ε^abc is the Levi-Civita symbol - E_a^i are the densitized triads (gravitational electric field) - F_ij^c is the curvature of the Ashtekar connection - **String Theory Gravity**: $S_{\text{gravity}}^{\text{String}} = \frac{1}{2\kappa^2} \int d^{10}x \, \sqrt{-g} \, e^{-2\phi} \left(R + 4 (\nabla \phi)^2 - \frac{1}{12} H_{\mu\nu\rho} H^{\mu\nu\rho}\right)$ - κ is related to the string tension - ϕ is the dilaton field - H_μνρ is the field strength of the Kalb-Ramond field ### Matter Action The matter action describes all forms of matter and their interactions with spacetime: - **Fermion Fields**: $S_{\text{fermion}} = \int d^4x \, \sqrt{-g} \, \bar{\psi} (i \gamma^\mu D_\mu - m) \psi$ - ψ is the fermion field - ψ■ is the Dirac adjoint - γ^μ are the Dirac gamma matrices - D_μ is the covariant derivative - m is the mass of the fermion - **Higgs Field**: $S_{\text{Higgs}} = \int d^4x \, \sqrt{-g} \, \left[(D_\mu \phi)^\dagger (D^\mu \phi) - V(\phi) \right]$ - ϕ is the Higgs field - D_μ is the covariant derivative - V(ϕ) = -μ²|ϕ|² + λ|ϕ|■ is the Higgs potential ### Gauge Field Action The gauge field action describes the fundamental forces (electromagnetic, strong, and weak): - **Yang-Mills Action**: $S_{\text{gauge}} = -\frac{1}{4} \int d^4x \, \sqrt{-g} \, F_{\mu\nu}^a F^{\mu\nu}_a$ - F_μν^a is the field strength tensor - For SU(3): F_μν^a = ∂_μA_ν^a - ∂_νA_μ^a + g f^abc A_μ^b A_ν^c - **Supersymmetric Gauge Fields**: $S_{\text{SUSY-gauge}} = \int d^4x \, \left[-\frac{1}{4} F_{\mu\nu} F^{\mu\nu} + i \bar{\lambda} \gamma^\mu D_\mu \lambda \right]$ - λ is the gaugino (supersymmetric partner of the gauge boson) - λ■ is the Dirac adjoint ### Quantum Corrections The quantum corrections account for quantum fluctuations and virtual particles: - **Path Integral Formulation**: $Z = \int \mathcal{D}\phi \, e^{i S[\phi]}$ - Z is the partition function - ■ϕ represents the functional integration measure over all field configurations - S[ϕ] is the action functional - **Loop Corrections**: $S_{\text{quantum}} = \sum_{n=1}^{\infty} \hbar^n S_n$ - ■ is the reduced Planck constant - S_n represents the n-loop quantum correction to the classical action

Generating Documentation

The Theory of Everything provides tools for generating documentation with properly formatted equations: ### Creating a PDF To create a PDF with properly formatted equations: 1. Run the PDF generation script: ``` python create_toe_pdf.py ``` 2. Choose between matplotlib-based or LaTeX-based PDF generation: ``` 1. Generate PDF using matplotlib 2. Generate PDF using LaTeX ``` 3. The PDF will be generated in the current directory. ### Generating LaTeX Documentation To generate a LaTeX document with professional typesetting: 1. Run the LaTeX generation script: ``` python generate_latex_toe.py ``` 2. If pdflatex is

available, the script will compile the LaTeX document to PDF. 3. The LaTeX document and PDF will be generated in the current directory.

Troubleshooting

Common Issues 1. **ImportError: No module named 'numpy'** - Solution: Install NumPy with `pip install numpy` 2. **ImportError: No module named 'sympy'** - Solution: Install SymPy with `pip install sympy` 3. **ImportError: No module named 'matplotlib'** - Solution: Install Matplotlib with `pip install matplotlib` 4. **ModuleNotFoundError: No module named 'component_formulas.unified_action'** - Solution: Make sure you're running the script from the project root directory 5. **RuntimeError: No pdflatex executable found** - Solution: Install a LaTeX distribution (e.g., TeX Live, MiKTeX) and ensure pdflatex is in your PATH ### Getting Help If you encounter any issues not covered in this guide, please: 1. Check the documentation in the `DOCUMENTATION.md` file 2. Look for similar issues in the project's issue tracker 3. Contact the project maintainers for assistance

Next Steps

After exploring the component formulas and visualizations, you might want to: 1. Extend the project with your own physics components 2. Implement additional visualization methods 3. Explore different parameter regimes 4. Contribute to the project by improving the code or documentation For more information, see the complete documentation in the `DOCUMENTATION.md` file.

Formula Improvements in the Theory of Everything

The following sections are from FORMULA_IMPROVEMENTS.md.

Introduction

This document summarizes the improvements made to maintain the mathematical integrity of the formulas in the Theory of Everything codebase.

Gravity Action Components

Einstein-Hilbert Action ``` S_gravity^EH = 1/(16πG) ∫d■x √(-g) (R - 2Λ) ``` - Added detailed notes explaining the formula - Clarified that R is the Ricci scalar curvature - Clarified that Λ is the cosmological constant ### Loop Quantum Gravity Action ``` S_gravity^LQG = 1/(8πG) ∫d■x √(-g) ε^abc E_a^i E_b^j F_ij^c ``` - Added detailed notes explaining the formula - Clarified that E_a^i are the densitized triads - Clarified that F_ij^c is the curvature of the Ashtekar connection ### String Theory Gravity Action ``` S_gravity^String = 1/(2κ²) ∫d^10x √(-g) e^{-2ϕ} [R + 4(∇ϕ)² - (1/12)H_μνρ H^μνρ] ``` - Added detailed notes explaining the formula - Clarified that ϕ is the dilaton field - Clarified that H_μνρ is the field strength of the Kalb-Ramond field

Matter Action Components

Fermion (Dirac) Action ``` S_fermion = ∫d■x √(-g) ψ■(iγ^μ D_μ - m)ψ ``` - Added detailed notes explaining the formula - Ensured proper representation of the Dirac gamma matrices - Clarified the meaning of the covariant derivative D_μ ### Higgs Action ``` S_Higgs = ∫d■x √(-g) [(D_μϕ)†(D^μϕ) - V(ϕ)] ``` - Added detailed notes explaining the formula - Enhanced the description of the Higgs potential V(ϕ) = -μ²|ϕ|² + λ|ϕ|■ - Explained the "Mexican hat" potential and its role in spontaneous symmetry breaking

Gauge Field Action Components

Yang-Mills Action ``` S_gauge = -1/4 ∫d■x √(-g) F_μν^a F^μν_a ``` - Added detailed notes explaining the formula - Included the explicit form of the field strength tensor for SU(3): F_μν^a = ∂_μA_ν^a - ∂_νA_μ^a + g f^abc A_μ^b A_ν^c ### Supersymmetric Gauge Action ``` S_SUSY-gauge = ∫d■x [-1/4 F_μν F^μν + i λ■γ^μ D_μλ] ``` - Added detailed notes explaining the formula - Clarified that λ is the gaugino (supersymmetric partner of the gauge boson)

Quantum Corrections Components

Path Integral Formulation ``` Z = ∫■ϕ e^{iS[ϕ]} ``` - Added detailed notes explaining the formula - Clarified that ■ϕ represents the functional integration measure over all field configurations ### Loop Corrections ``` S_quantum = ∑_{n=1}^∞ ■^n S_n ``` - Added detailed notes explaining the formula - Clarified that S_n represents the n-loop quantum correction to the classical action

Unified Action

Master Equation ``` S = S_gravity + S_matter + S_gauge + S_quantum ``` - Added detailed notes explaining how this combines all fundamental interactions - Enhanced the full master equation with proper quantum corrections term - Added explanatory notes for each component of the unified action ### Full Master Equation ``` S = 1/(16πG) ∫d■x √(-g) (R - 2Λ) + ∫d■x √(-g) [ψ■(iγ^μ D_μ - m)ψ + (D_μϕ)†(D^μϕ) - V(ϕ) - 1/4 F_μν^a F^μν_a] + ∑_{n=1}^∞ ■^n S_n ``` - Expanded the master equation to show all components explicitly - Added detailed notes explaining each term in the equation - Ensured mathematical consistency across all terms

General Improvements

- Fixed import statements to include all necessary dependencies - Added proper documentation for all mathematical formulas - Ensured consistent notation across all components - Maintained the integrity of the mathematical expressions - Enhanced readability with detailed explanatory notes

Enhanced 3D and 4D Visualizations for the Theory of Everything

The following sections are from ENHANCED_VISUALIZATIONS.md.

Introduction

This module provides advanced visualizations that maximize 3D representations and visualize multi-dimensional spaces in 4D where possible.

Overview

The enhanced visualizations module extends the Theory of Everything codebase with state-of-the-art visualizations that help users understand complex physical concepts through interactive 3D and 4D representations. These visualizations are designed to provide intuitive insights into the mathematical structures underlying the unified theory.

Features

- **4D Spacetime Curvature**: Visualize 4D spacetime around massive objects using a 3D surface with color representing the time dimension. - **Quantum Foam in 3D**: Explore the quantum fluctuations of spacetime at the Planck scale with detailed 3D voxel representations. - **String Worldsheet in 3D**: Visualize the evolution of a string through spacetime with multiple oscillation modes. - **Extra Dimensions (String Theory)**: See how extra dimensions are compactified at each point in our 3D space using Calabi-Yau manifold representations. - **4D Higgs Field**: Understand spontaneous symmetry breaking with a 3D visualization of the Higgs field, using color to represent the 4th dimension. - **4D Gauge Field Configuration**: Explore non-Abelian gauge fields with a 3D vector field visualization, using color to represent field strength.

Usage

To run the enhanced visualizations: ```bash python run_enhanced_vis.py ``` This script will display a menu of available visualizations. Select a visualization by entering the corresponding number.

Technical Details

4D Spacetime Curvature This visualization shows 4D spacetime around a massive object using the Schwarzschild metric. The 3D surface represents spatial curvature, while the color represents time dilation (the 4th dimension). The visualization is based on the Schwarzschild metric: ``` g_tt = -(1 - 2GM/rc²) ``` ### Quantum Foam in 3D This visualization shows quantum foam - the quantum fluctuations of spacetime at the Planck scale. The foam represents the probabilistic nature of spacetime at quantum scales. The visualization uses a 3D voxel representation with multiple frequency components to simulate quantum fluctuations. ### String Worldsheet in 3D This visualization shows a string worldsheet in 3D. The string evolves through time, with multiple oscillation modes. The worldsheet represents the string's path through spacetime. The visualization includes: - Multiple oscillation modes with different amplitudes and phases - Time evolution of the string - Connections between time steps to show the worldsheet ### Extra Dimensions (String Theory) This visualization represents a multi-dimensional space with compactified extra dimensions. The large transparent sphere represents our visible 3D space, while the small complex shapes represent the extra dimensions compactified at each point in our 3D space. The visualization uses simplified Calabi-Yau manifolds to represent the extra dimensions. ### 4D Higgs Field This visualization shows the 4D Higgs field with spontaneous symmetry breaking: - The Mexican hat potential in the complex ϕ plane - The Higgs field in

3D space, with the 4th dimension (field magnitude) represented by color The visualization includes the circle of minima representing the vacuum expectation value. ### 4D Gauge Field Configuration This visualization shows a 4D non-Abelian gauge field. The arrows represent the field direction in 3D space, while the color represents the field strength (4th dimension). The configuration is similar to a magnetic monopole in SU(2) gauge theory.

Implementation Notes

The enhanced visualizations are implemented using: - NumPy for numerical calculations - Matplotlib for 3D plotting - Custom colormaps for representing the 4th dimension - Advanced 3D techniques like voxels, vector fields, and surface plots The visualizations are designed to be both scientifically accurate and visually appealing, providing insights into the complex mathematical structures of the Theory of Everything.

Future Enhancements

Planned future enhancements include: - Interactive visualizations with sliders for parameter adjustment - Animated visualizations showing time evolution - VR/AR support for immersive exploration of multi-dimensional spaces - Higher-dimensional visualizations using advanced projection techniques

Agent-Friendly Visualizations for the Theory of Everything

The following sections are from AGENT_VISUALIZATIONS.md.

Introduction

This module provides enhanced 3D and 4D visualizations for the Theory of Everything that can be called programmatically by AI agents. It includes a clean API for generating visualizations and accessing formulas without requiring user interaction.

Overview

The agent-friendly visualizations extend the Theory of Everything codebase with a programmatic interface that allows AI agents to: 1. Generate high-quality 3D and 4D visualizations 2. Access mathematical formulas and their LaTeX representations 3. Customize visualization parameters 4. Save visualizations to disk for later use

Components

The agent-friendly visualization system consists of two main components: 1. **`toe_api.py`**: Clean API for interacting with the Theory of Everything, with built-in visualization capabilities 2. **`agent_example.py`**: Example script demonstrating how an AI agent can use the API

API Usage

Basic Usage ```python from toe_api import ToEAPI api = ToEAPI() visualizations = api.list_visualizations() formulas = api.list_formulas() unified_action = api.get_formula("unified_action") params = {"mass": 2.0, "grid_size": 30} vis_path = api.generate_visualization("4d_spacetime_curvature", params) ``` ### Available Visualizations The API provides the following visualizations: 1. **4D Spacetime Curvature** (`4d_spacetime_curvature`): Visualizes 4D spacetime around massive objects using a 3D surface with color representing the time dimension. Parameters: - `mass`: Mass of the central object (in solar masses) - `grid_size`: Size of the grid for visualization 2. **Quantum Foam in 3D** (`quantum_foam_3d`): Provides a detailed 3D voxel representation of quantum fluctuations at the Planck scale. Parameters: - `grid_size`: Size of the grid for visualization - `amplitude`: Amplitude of quantum fluctuations - `frequency`: Frequency of quantum fluctuations 3. **Extra Dimensions** (`extra_dimensions_3d`): Visualizes how extra dimensions are compactified at each point in our 3D space using Calabi-Yau manifold representations. Parameters: - `num_dimensions`: Total number of dimensions to represent - `grid_size`: Size of the grid for visualization 4. **4D Higgs Field** (`4d_higgs_field`): Represents the 4D Higgs field with spontaneous symmetry breaking, using color to show the 4th dimension. Parameters: - `grid_size`: Size of the grid for visualization 5. **4D Gauge Field** (`gauge_field_4d`): Displays non-Abelian gauge fields with a 3D vector field visualization, using color to represent field strength. Parameters: - `grid_size`: Size of the grid for visualization 6. **All Visualizations** (`all`): Generates all available visualizations. ### Available Formulas The API provides access to the following formulas: 1. **Unified Action** (`unified_action`): The unified action (master equation) 2. **Gravity Action** (`gravity_action`): The gravity action components 3. **Matter Action** (`matter_action`): The matter action components 4. **Gauge Field Action** (`gauge_action`): The gauge field action components 5. **Quantum Corrections** (`quantum_corrections`): The quantum corrections components 6. **Full Master Equation** (`full_master_equation`): The full master equation

Command Line Interface

The API also provides a command line interface for generating visualizations and accessing formulas: ```bash python toe_api.py visualize 4d_spacetime_curvature --params '{"mass": 2.0, "grid_size": 30}' python toe_api.py formula unified_action python toe_api.py list visualizations python toe_api.py list formulas ```

Example Agent Interaction

The `agent_example.py` script demonstrates how an AI agent can interact with the Theory of Everything API: ```bash python agent_example.py ``` This script shows how to: 1. List available visualizations and formulas 2. Get formula information including LaTeX representations 3. Generate visualizations with custom parameters 4. Save visualizations to disk

Testing the API

The `test_api.py` script provides a simple way to test that the API is working correctly: ```bash python test_api.py ``` This script performs basic tests of the API functionality: 1. Importing the API 2. Creating an API instance 3. Listing available visualizations 4. Retrieving a formula 5. Generating a simple visualization

Implementation Notes

The agent-friendly visualization system is designed to handle the common issues that arise when working with the Theory of Everything codebase: 1. **Math Module Conflict**: The system automatically handles the conflict between the project's `math` directory and Python's built-in `math` module. 2. **Visualization Performance**: The system optimizes visualization performance by adjusting parameters based on the complexity of the visualization. 3. **Error Handling**: The system includes robust error handling to ensure that AI agents can recover from errors. 4. **Clean API**: The system provides a clean, well-documented API that is easy for AI agents to use.

Future Enhancements

Planned future enhancements include: 1. **Interactive Visualizations**: Support for generating interactive visualizations that can be embedded in web applications. 2. **Animation Support**: Support for generating animations that show time evolution of physical systems. 3. **3D Model Export**: Support for exporting visualizations as 3D models that can be viewed in VR/AR applications. 4. **Real-time Collaboration**: Support for real-time collaboration between AI agents and human users.

Run the interactive UI

The following sections are from README_MODULAR.md.

Introduction

y and # Theory of Everything - Modular API This repository contains a modular API for interacting with the Theory of Everything, providing both human-friendly interfaces and advanced tools for AI agents.

Overview

The Theory of Everything API is designed with a modular architecture, consisting of small, focused components that can be used independently or combined through the unified interface. This approach provides flexibility, maintainability, and extensibility.

Modules

The API consists of the following modules: ### Core Module (`toe_core.py`) The core module provides essential functionality used by all other modules, including: - Math module conflict handling - File operations - Script execution utilities - JSON handling ### Formula Module (`toe_formulas.py`) The formula module provides tools for working with mathematical formulas, including: - Formula retrieval - Formula exploration - Formula comparison - Formula search - Formula dependency analysis - LaTeX export ### Visualization Module (`toe_vis.py`) The visualization module provides tools for generating visualizations, including: - Visualization generation - Parameter validation - Parameter suggestions - Batch processing ### Agent Tools Module (`toe_agent.py`) The agent tools module provides specialized tools for AI agents, including: - Session management - Theory exploration - Visualization sequence generation - Insight extraction ### UI Module (`toe_ui.py`) The UI module provides a command-line interface for human users, including: - Interactive menus - Formula exploration - Visualization generation - Formula comparison ### Unified API

(`toe_unified.py`) The unified API combines all modules into a single interface, providing: - Access to all functionality through a consistent interface - Agent mode for advanced features - Command-line interface for scripting

Usage

For Human Users Human users can interact with the API through the command-line interface: ```bash python toe_ui.py python toe_unified.py formula get unified_action python toe_unified.py visualization generate 4d_spacetime_curvature --show ``` ### For AI Agents AI agents can interact with the API programmatically: ```python from toe_unified import ToEUnified api = ToEUnified(agent_mode=True) exploration = api.explore_theory() result = api.generate_visualization_for_formula("gravity_action") insights = api.extract_insights(formula_name="gravity_action") ``` See `agent_advanced_example.py` for more examples of how AI agents can use the API.

Advanced Features for Agents

The API provides several advanced features specifically designed for AI agents: ### Session Management Agents can create and manage sessions to track their interactions with the API: ```python session_info = api.get_session_info() ``` ### Theory Exploration Agents can explore the Theory of Everything at a high level: ```python exploration = api.explore_theory() results = api.explore_theory(query="gravity") ``` ### Visualization Sequences Agents can generate sequences of visualizations by varying parameters: ```python param_ranges = { "mass": [0.5, 1.0, 2.0, 5.0] } sequence = api.generate_visualization_sequence("4d_spacetime_curvature", param_ranges) ``` ### Insight Extraction Agents can extract insights from formulas and visualizations: ```python insights = api.extract_insights(formula_name="gravity_action") insights = api.extract_insights(visualization_name="4d_spacetime_curvature") ``` ### Batch Processing Agents can generate multiple visualizations in batch: ```python configs = [{"name": "4d_spacetime_curvature", "params": {"mass": 1.0}}, {"name": "quantum_foam_3d", "params": {"amplitude": 0.7}}] results = api.batch_generate_visualizations(configs) ``` ### Formula Exploration Agents can explore formulas and their relationships: ```python exploration = api.explore_formula("unified_action") dependencies = api.get_formula_dependencies("unified_action") comparison = api.compare_formulas(["gravity_action", "matter_action"]) ``` ### LaTeX and PDF Generation Agents can generate LaTeX and PDF documentation: ```python from latexagent import LaTeXAgent from pdfagent import PDFAgent latex_agent = LaTeXAgent() latex_content = latex_agent.generate_latex(formula="unified_action", include_components=True) latex_agent.save_latex(latex_content, "unified_action.tex") pdf_agent = PDFAgent() pdf_path = pdf_agent.generate_pdf(formula="unified_action", output="unified_action.pdf") ``` The LaTeX and PDF agents can also be used from the command line: ```bash python latexagent.py --formula unified_action --output unified_action.tex python pdfagent.py --formula unified_action --output unified_action.pdf python latexagent.py --list-formulas python latexagent.py --help python pdfagent.py --help ```

Examples

The repository includes several example scripts: - `agent_advanced_example.py`: Demonstrates how AI agents can use the advanced features of the API - `toe_ui.py`: Provides an interactive command-line interface for human users - `toe_unified.py`: Provides a command-line interface for scripting

Tests

The repository includes a comprehensive test suite in the `tests` directory. These tests are designed to verify the functionality of each module and can be run individually or as a complete suite. ```bash python tests/run_tests.py python tests/run_tests.py imports python tests/run_tests.py core python tests/run_tests.py modules python tests/run_tests.py formula python tests/run_tests.py visualization python tests/run_tests.py agent ``` The test suite includes: - **Import Tests**: Test importing all modules - **Core Tests**: Test the core functionality like math module conflict handling and file operations - **Module Tests**: Test importing and creating instances of all modules - **Formula Tests**: Test the formula functionality including retrieval, exploration, and comparison - **Visualization Tests**: Test the visualization functionality including parameter validation and generation - **Agent Tests**: Test the agent-specific features like session management and insight extraction - **LaTeX Tests**: Test the LaTeX agent functionality for generating LaTeX documentation - **PDF Tests**: Test the PDF agent functionality for generating PDF documentation **Note**: Some tests may require additional setup or dependencies. If you encounter issues running the tests, please check the individual test files for specific requirements.

Installation

No installation is required. Simply clone the repository and run the scripts.

Requirements

- Python 3.6 or higher - NumPy - Matplotlib

License

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