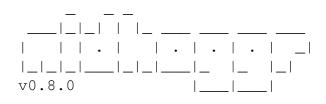
Nidhoggr User Manual

Cody Raskin April 25, 2025



Contents

1	Introduction 1.1 Purpose of Nidhoggr	4 4 4 5
2	Installation	6
3	Getting Started	7
4	Usage	8
5	Core Concepts	9
6	Examples 6.1 Simple test cases	10 10
7	Customization and Extension	11
8	Best Practices	12
9	Troubleshooting	13
10	Reference	14
11	Acknowledgments	15
12	License	15
\mathbf{A}	Appendix A: Glossary	16
В	Appendix B: Additional Resources	16

1 Introduction

1.1 Purpose of Nidhoggr

Nidhoggr is a generic physics simulation framework. It is designed to be used as a base for varied physics simulation methods (FVM,FEM,etc) while keeping helper methods likes equations of state and integrators generic enough to be portable to a wide variety of methods choices. Nidhoggr's major classes and methods are written in C++ and wrapped in Python using pybind11 to enable them to be imported as Python3+ modules inside a runscript. Python holds and passes the pointers to most objects inside the code, while the integration step is always handled by compiled C++ code. Any Python class that returns the expected data types of the compiled C++ classes can substitute for a precompiled package (e.g. a custom equation of state), though speed will suffer.

1.2 Overview of capabilities

As of April 25, 2025, Nidhoggr's capabilities are as follows:

Component	Working	Development	Planned
Physics	N-body gravity	HLL hydro	SPH
	Gravity point sources	FEM	
	Constant direction gravity sources		
	Particle kinetics		
	Acoustic wave solvers		
	Shallow wave equation solvers		
	Chemical reaction solvers		
Equations of State	Ideal gas		Helmholtz
	Polytrope		
Time Integrators	Forward Euler		Symplectic
	2nd Order Runge-Kutta		
	4th Order Runge-Kutta		
Meshing	Eulerian grid	FEM	AMR
Data IO	Silo		
	vtk		
	obj		
	wav		
Custom Data Types	Vectors	Elements	
	Tensors		
	Cosmologies		
	Units		
Parallel	OpenMP		MPI

Table 1: Status of Major Components in Nidhoggr

1.3 Intended audience

Nidhoggr's intended audience is computational scientists who want a toy simulation code to scope simple problems with that's easily driveable and scriptable with a Python interface, and anyone who doesn't mind getting their hands dirty writing their own physics packages in a fully abstracted simulation framework.

2 Installation

- \bullet System requirements
- Dependencies
- Downloading the source code
- \bullet Building and installing

3 Getting Started

 \bullet Basic concepts

• First run: a simple example

4 Usage

- Running Nidhoggr
- Command-line options

5 Core Concepts

- $\bullet\,$ Nodelists and Fields
- Physics methods
- Equations of State
- Mesh/grid handling
- Boundary conditions
- Integrators
- The Controller

6 Examples

The examples folder holds Python runscripts that solve a particular notional physics (or purely calculational) problem.

File	Purpose
cherenkov.py	Simulates a supersonic (or superluminal) point source
	moving through a medium at a speed greater than c.
cosmo.py	Creates an example cosmology (Ω_m, Λ, H_0) and reports
	the properties of that cosmology at the chosen redshift.
diffractionGrating.py	Simulates the transmission of an acoustic wave through
	a diffraction grating.
imageToStringArt.py	Creates the instructions for (and previews) an image made
	from strings stretched across a wheel with a chosen number
	of pins.
oort.py	Simulates a star passing through the Oort cloud
	and dislodging a comet from its orbit.
plinko.py	Simulates the Plinko game.
relativity.py	Calculates the time dilation for a relativistic traveler.
rps.py	Simulates the destruction of chemical mixtures in a
	rock-paper-scissors-like reaction setup, where
	$A \to B \to C \to A$.
tensors.py	Creates some tensors and does some linear algebra with them.
vectors.py	Creates some vectors and does some linear algebra with them.
waveLogo.py	Simulates acoustic waves inside a region with Dirichlet
	boundary conditions arranged in a unique fashion.

Table 2: Examples included in the main branch.

6.1 Simple test cases

Many of the Python scripts inside the tests folder stress single components of Nidhoggr, or a small subset of them. For instance, waveBox.py tests the acoustic wave solver with a single oscillatory source in the center of a box with two openings on either end (using Dirichlet boundaries to create the box).

7 Customization and Extension

- Modifying source code
- Adding new physics modules
- Extending the input parser

8 Best Practices

- $\bullet\,$ Tips for efficient simulation
- Debugging guidance
- Performance tuning

9 Troubleshooting

- \bullet Common errors and solutions
- \bullet FAQ

10 Reference

- Code structure overview
- Important classes and functions
- File organization

11 Acknowledgments

- \bullet Contributors
- Funding and support

12 License

- License terms
- How to cite Nidhoggr

A Appendix A: Glossary

• Terms and definitions

B Appendix B: Additional Resources

- $\bullet\,$ Related software
- Recommended reading