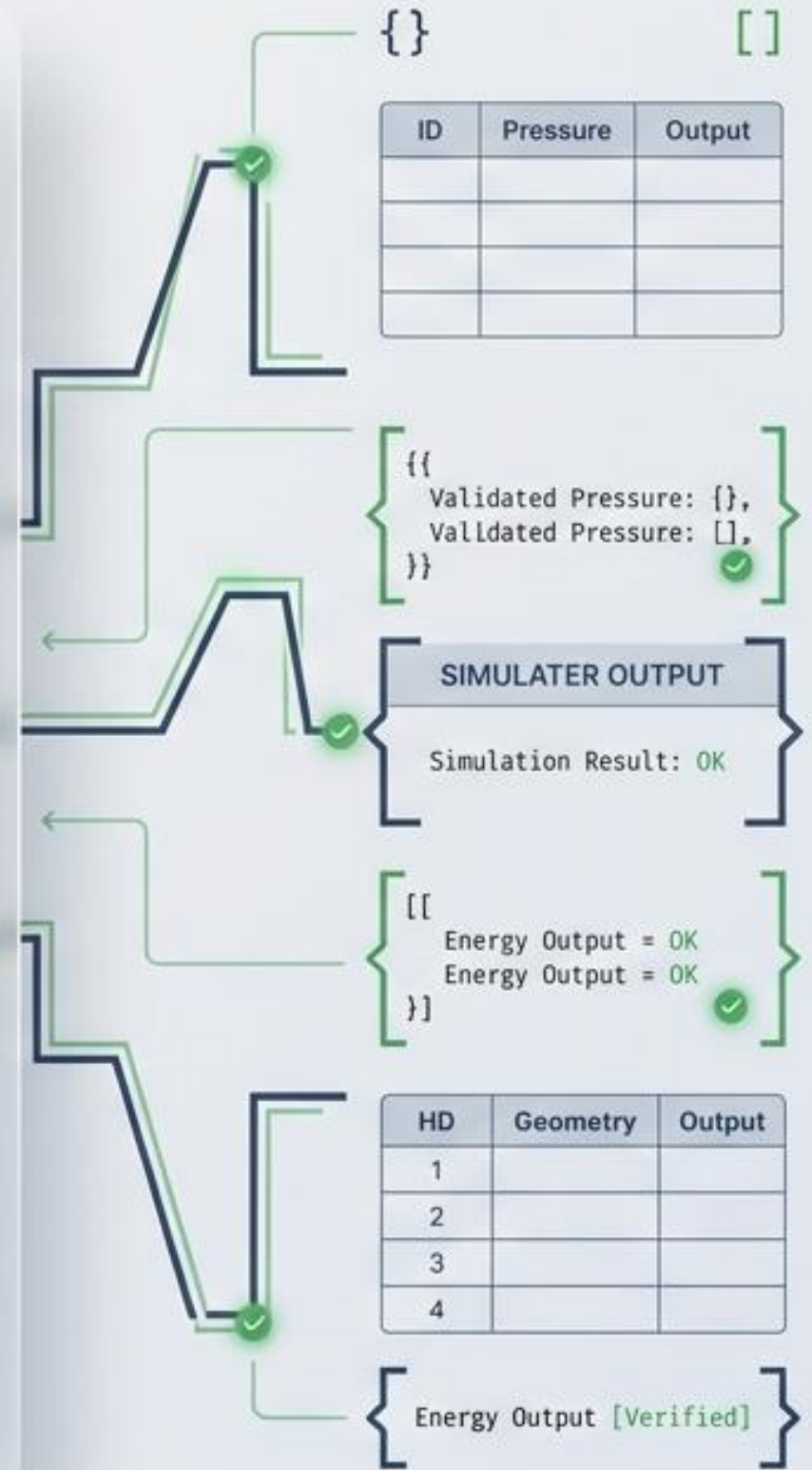


Agentic Engineering with NeqSim

Synthesizing natural language intent with rigorous physical simulation.

```
{  
  Filtering application,  
  logic application  
}
```

```
filter = {  
  filtering application,  
  logic application  
}
```



Topics for presentation :

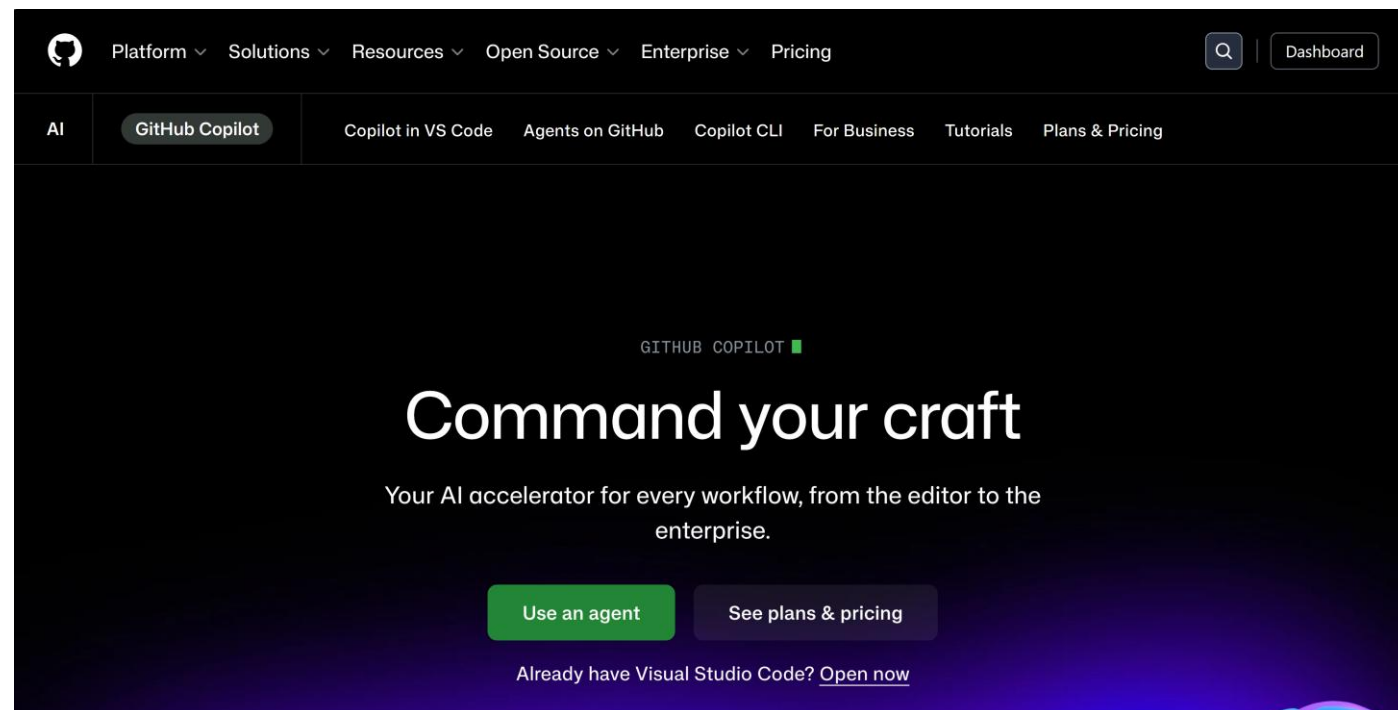
1. Demonstration of tools and steps in solving an engineering task
2. Description of agentic engineering using NeqSim
3. Demonstration of a task – CNG tank technology

Diagnostic Comparison

Aspect	Manual Coding	Commercial Simulators	Agentic NeqSim
Learning Curve	Steep (Learn API)	Moderate (Learn GUI)	Low (Natural Language)
Standards Compliance	Manual lookup	Some built-in	Agent loads applicable standards
Reproducibility	Good (Code)	Poor (GUI state lost)	Excellent (Notebook + Task Folder)
Report Generation	Manual	Manual export	Automated Word + HTML
Physics Rigor	Full Control	Vendor-validated	Full (Same NeqSim Engine)

Tools used in this workflow

- GitHub (with GitHub user account)
- Github Copilot (or Claude Code) with access to LLMs such as ChatGPT, Claude or Gemini
- Visual Studio Code (or other IDE) alternatively access to CodeSpaces for Cloud

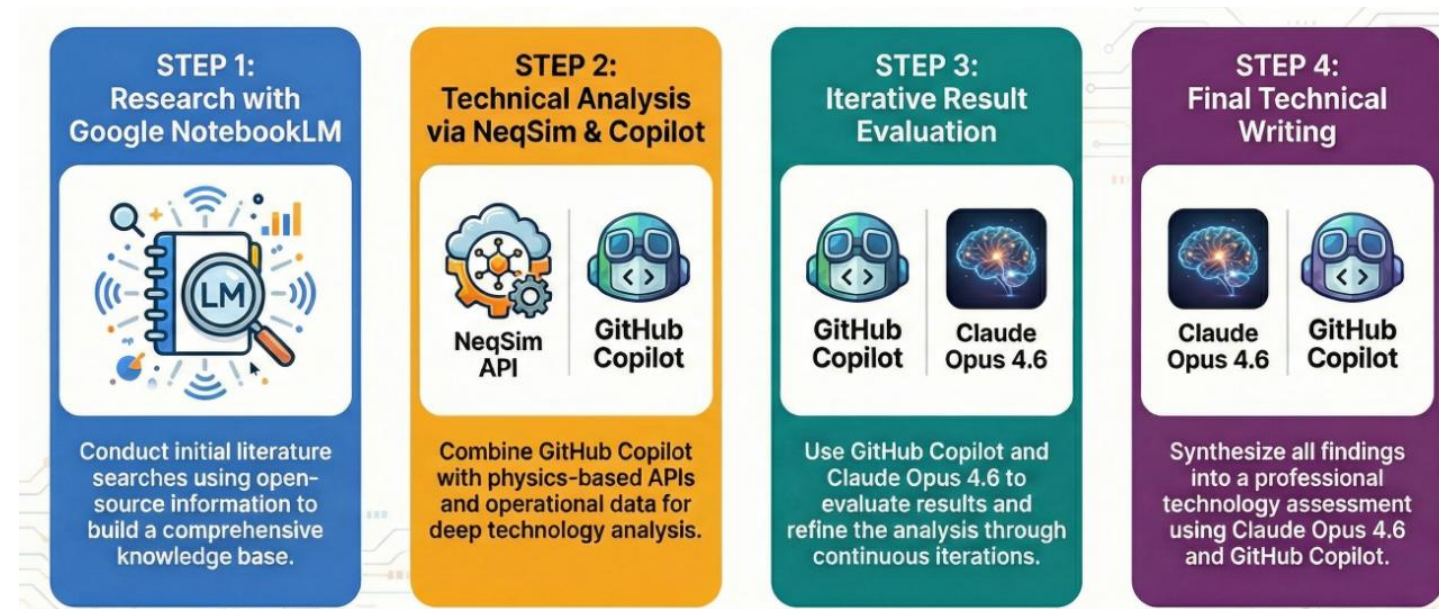
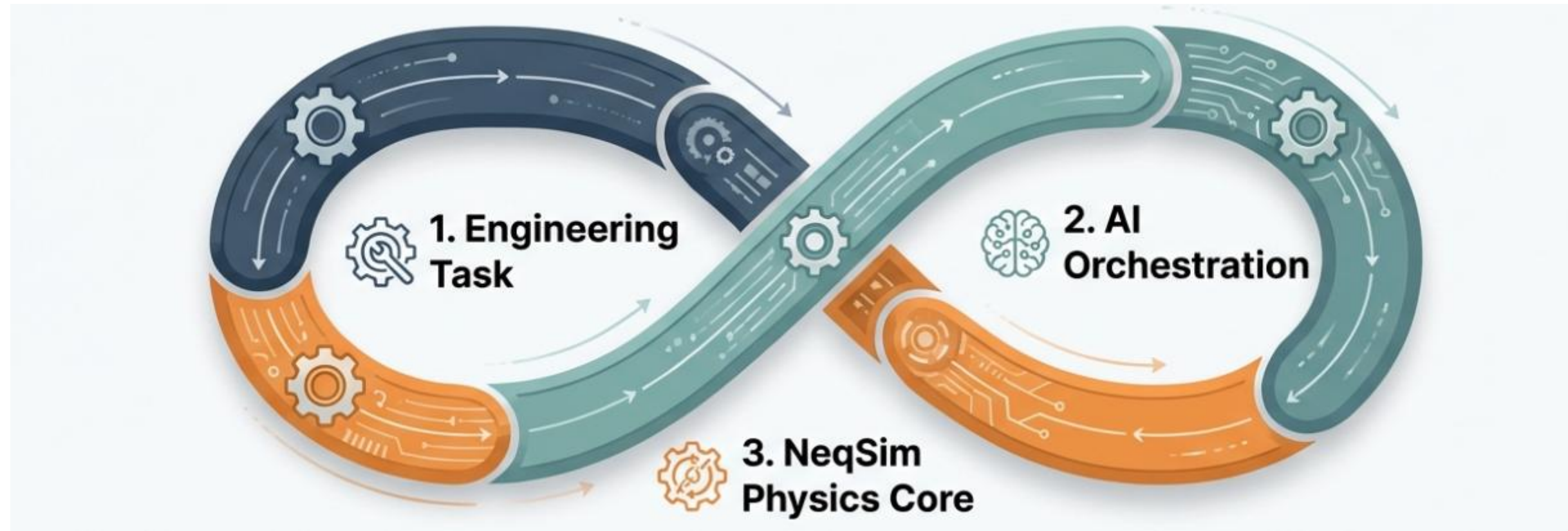


[GitHub Copilot · Your AI pair programmer](#)



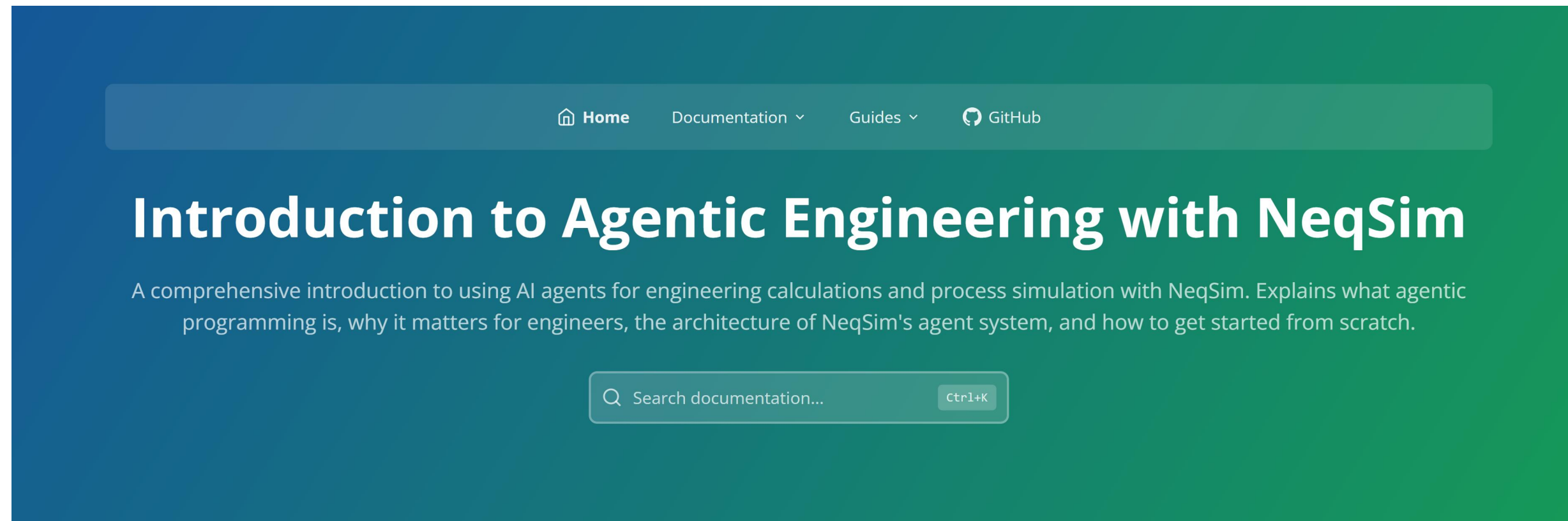
[GitHub](#)

Workflow: Task solving and NeqSim development



Part 1.

Demonstration of tools and steps in solving an engineering task



Introduction to Agentic Engineering with NeqSim

This guide explains how NeqSim combines AI agents with a rigorous physics engine to solve engineering problems. It is written for engineers and scientists who are new to AI-assisted workflows and want to understand both the concepts and the practical tools.

What you will learn:

- What agentic programming means and why it matters

On this page

1. The Problem: AI Alone Cannot Do Engineering
2. What is Agentic Programming?
 - Key Concepts
 - Why Agents Instead of Scripts?

[Web page](#)

Clone NeqSim repo to local machine (and open in Visual Studio Code on local machine) or click Code button and create a new Codespace (Visual Studio Code will open in web browser)

The screenshot shows the GitHub repository page for 'neqsim' by 'equinor'. The repository is public and has 103 stars, 40 forks, and 6 unwatchers. The repository structure is visible, including folders like '.config', '.devcontainer', '.gemini', '.github', '.mvn/wrapper', '.openapi', '.vscode', 'devtools', and 'docs'. A 'Code' button is highlighted in green, and a dropdown menu is open, showing options for 'Local' and 'Codespaces'. The 'Codespaces' section lists several active and inactive workspaces, including 'automatic pancake', 'sturdy broccoli', 'refactored dollop', and 'sturdy zebra'. The 'About' section provides a description of NeqSim as a library for fluid behavior, phase equilibrium, and process simulation, along with a link to the project's home page and various tags like 'processing', 'java', 'thermodynamics', etc.

[NeqSim home page](#)

Step 1. Define a task in the chat window

The screenshot displays the VS Code interface with a chat window open on the right side. The chat window is titled "DESIGNING A METHANE COMPRESSOR SYSTEM" and contains a user message: "make a task doing a design a a compressor compressing methane from 10 bar to 25 bar with flow rate 10000 kg/hr. Make a detaied mechanical deisgn." The chat window shows the AI assistant's response, which includes: "Optimizing tool selection...", "Reviewed 4 files", "I am setting this up as a Type F design task and will now create the mandatory task folder first, then populate scope, analysis notes, and a runnable compressor mechanical-design notebook with verification outputs and report generation. Next I will verify the exact compressor feasibility/mechanical APIs in source before writing the notebook so the workflow runs end-to-end.", "Ran python devtools/new_task.py 'Detailed mechanical d...", "The task workspace is created successfully under task_solve and I will now read its README plus the scaffolded Step 1 files, then build a standards-based compressor design notebook and generate results and report outputs.", "Reviewed 4 files", "I have the scaffold and next I am validating the exact compressor and mechanical feasibility APIs in source code so the notebook uses only confirmed method signatures and can run without guesswork.", "Executed commands in terminal for Compressor design analysis", "Ripgrep is unavailable in this container, so I am switching to find/grep for API verification and then I will fill the task specification and build the notebook with the confirmed signatures.", "Executed multiple grep commands for compressor classes", "I confirmed the compressor and feasibility classes exist; now I am reading the key method blocks to safely script the notebook outputs and detailed design tables without API mismatches." The chat window also shows a text input field with the prompt: "Describe the engineering task — e.g., 'JT cooling for rich gas at 100 bara', 'TEG dehydration sizing for 50 MMSCFD wet gas', 'hydrate form". The VS Code interface also shows the Explorer view on the left with the project structure for "NEQSIM [CODESPACES: AUTOMATIC PANCAKE]". The bottom status bar shows "Codespaces: automatic pancake" and "Layout: Norwegian".

EXPLORER

- NEQSIM [CODESPACES: AUTOMATIC PANCAKE]
- .github
- .mvn
- .openapi
- .venv
- .vscode
- devtools
- docs
- examples
- scripts
- src
- target
- task_solve
 - 2026-03-25_detailed_mechanical_design_of_met...
 - figures
 - step1_scope_and_research
 - step2_analysis
 - step3_report
 - _eq_images
 - generate_report.py
 - Report.docx
 - Report.html
 - README.md
 - results.json
 - TASK_TEMPLATE
 - README.md
 - .flattened-pom.xml
 - .gitignore
 - AGENTS.md
 - ! azure-pipelines.yml
 - OUTLINE
 - TIMELINE
 - JAVA PROJECTS
 - MAVEN

CHAT

DESIGNING A METHANE COMPRESSOR SYSTEM

make a task doing a design a a compressor compressing methane from 10 bar to 25 bar with flow rate 10000 kg/hr. Make a detaied mechanical deisgn.

Optimizing tool selection...

Reviewed 4 files

I am setting this up as a Type F design task and will now create the mandatory task folder first, then populate scope, analysis notes, and a runnable compressor mechanical-design notebook with verification outputs and report generation. Next I will verify the exact compressor feasibility/mechanical APIs in source before writing the notebook so the workflow runs end-to-end.

Ran `python devtools/new_task.py "Detailed mechanical d...`

The task workspace is created successfully under task_solve and I will now read its README plus the scaffolded Step 1 files, then build a standards-based compressor design notebook and generate results and report outputs.

Reviewed 4 files

I have the scaffold and next I am validating the exact compressor and mechanical feasibility APIs in source code so the notebook uses only confirmed method signatures and can run without guesswork.

Executed commands in terminal for Compressor design analysis

Ripgrep is unavailable in this container, so I am switching to find/grep for API verification and then I will fill the task specification and build the notebook with the confirmed signatures.

Executed multiple grep commands for compressor classes

I confirmed the compressor and feasibility classes exist; now I am reading the key method blocks to safely script the notebook outputs and detailed design tables without API mismatches.

Describe the engineering task — e.g., 'JT cooling for rich gas at 100 bara', 'TEG dehydration sizing for 50 MMSCFD wet gas', 'hydrate form

Open Chat `Ctrl + Alt + I`

Show All Commands `Ctrl + Shift + P`

Toggle Terminal `Ctrl + ``

Codespaces: automatic pancake

master* Launchpad 0 8 10 Git Graph Java: Lightweight Mode

Layout: Norwegian

Step 2. Add data and literature and background info

The screenshot displays the Visual Studio Code (VS Code) interface within a Codespaces environment. The browser address bar shows the URL `https://automatic-pancake-pgvqrpg94gqhrp65.github.dev`. The VS Code window title is `neqsim [Codespaces: automatic pancake]`. The Explorer sidebar on the left shows the project structure for `NEQSIM [CODESPACES: AUTOMATIC PANC...`, with the `references` folder under `step1_scope_and_research` selected. The main editor area displays the `README.md` file with the following content:

```
1 # References Folder
2
3 Place literature papers, standards documents, and other
4 reference material here.
5
6 ## What to put in this folder
7
8 - **PDF papers** -- journal articles, conference papers,
9 technical reports
10
11 - **Standards excerpts** -- relevant sections from ASME, API,
12 DNV, ISO, NORSOK, etc.
13
14 - **Company documents** -- TR documents, design basis,
15 operating philosophy
16
17 - **Data sheets** -- equipment data sheets, material
18 certificates
19
20 - **Lab reports** -- PVT reports, fluid analysis, corrosion
21 test results
22
23
24 ## How the AI uses these files
25
26 1. **Google NotebookLM (recommended for PDFs):** Upload the
27 PDFs from this
28 folder to NotebookLM. It can read, cross-reference, and
29 cite multiple
30 documents at once. Ask it targeted questions and paste the
31 answers into
32 notes.md.
33
34 2. **VS Code Copilot Chat:** Copilot can read text-based files
35 (.txt, .md,
36 .csv) placed here. For PDFs, extract key sections as text
37 or summarise
38 them in notes.md so the AI can reference the content.
39
40 3. **Manual notes:** Read the papers yourself and capture key
41 equations,
42 data points, and design rules in notes.md under the
43 "Literature &
```

The Chat window on the right is titled `DESIGNING A METHANE COMPRESSOR SYSTEM`. It contains a user prompt: `make a task doing a design a a compressor compressing methane from 10 bar to 25 bar with flow rate 10000 kg/hr. Make a detaied mechanical deisgn.` The chat shows the AI's response: `Optimizing tool selection... Reviewed 4 files. I am setting this up as a Type F design task and will now create the mandatory task folder first, then populate scope, analysis notes, and a runnable compressor mechanical-design notebook with verification outputs and report generation. Next I will verify the exact compressor feasibility/mechanical APIs in source before writing the notebook so the workflow runs end-to-end.` Below this, it shows a command being run: `Ran python devtools/new_task.py "Detailed mechanical d...` and another response: `The task workspace is created successfully under task_solve and I will now read its README plus the scaffolded Step 1 files, then build a standards-based compressor design notebook and generate results and report outputs.` The chat also shows a second `Reviewed 4 files` status and a final response: `I have the scaffold and next I am validating the exact compressor and mechanical feasibility APIs in source code so the notebook uses only confirmed method signatures and can run without guesswork.` At the bottom of the chat, it indicates `Executed commands in terminal for Compressor design analysis` and shows a terminal output starting with `...`. The input field at the bottom of the chat contains `+ README.md` and the prompt `Describe the engineering task — e.g., 'JT cooling for rich gas at 100 bara', 'TEG c`. The status bar at the bottom of VS Code shows `Codespaces: automatic pancake`, `master*`, `Launchpad`, `0` errors, `8` warnings, `10` Git Graph items, and `Java: Lightweight Mode`. The chat window status bar shows `Ln 1, Col 1`, `Spaces: 3`, `UTF-8`, `LF`, `{ }` Markdown, and `Layout: Norwegian`.

Step 3. Jupyter (python code) will be made with calculations and documentation. Iterate add more calculations. Use chat to iterate and add new evaluations. Alos ask to improve or implement new NeqSim functionality.

The screenshot displays a GitHub Codespaces environment. The main window shows a Jupyter notebook titled "01_methane_compressor_mechanical_design.ipynb". The notebook content includes a title "Methane Compressor Thermodynamic and Mechanical Design", a description of the notebook's purpose, a design basis list, and a code cell with Python imports.

Methane Compressor Thermodynamic and Mechanical Design

This notebook performs a detailed screening-level mechanical design for methane compression from 10 bara to 25 bara at 10000 kg/hr using NeqSim.

Design basis:

- Gas: methane 100 mol%
- Suction pressure: 10 bara
- Discharge pressure: 25 bara
- Flow rate: 10000 kg/hr
- Inlet temperature: 25 C
- Base isentropic efficiency: 0.78
- Design references: API 617, ASME VIII Div 1, ASME B16.5, NACE MR0175 screening

```
import importlib
import subprocess
import sys
import os
import pathlib
import json
```

The chat interface on the right is titled "DESIGNING A METHANE COMPRESSOR SYSTEM" and contains a user prompt: "make a task doing a design a a compressor compressing methane from 10 bar to 25 bar with flow rate 10000 kg/hr. Make a detaied mechanical deisgn." The chat shows the AI's response, including file reviews and terminal output.

make a task doing a design a a compressor compressing methane from 10 bar to 25 bar with flow rate 10000 kg/hr. Make a detaied mechanical deisgn.

Optimizing tool selection...

Reviewed 4 files

I am setting this up as a Type F design task and will now create the mandatory task folder first, then populate scope, analysis notes, and a runnable compressor mechanical-design notebook with verification outputs and report generation. Next I will verify the exact compressor feasibility/mechanical APIs in source before writing the notebook so the workflow runs end-to-end.

Ran `python devtools/new_task.py "Detailed mechanical d...`

The task workspace is created successfully under task_solve and I will now read its README plus the scaffolded Step 1 files, then build a standards-based compressor design notebook and generate results and report outputs.

Reviewed 4 files

I have the scaffold and next I am validating the exact compressor and mechanical feasibility APIs in source code so the notebook uses only confirmed method signatures and can run without guesswork.

Executed commands in terminal for Compressor design analysis

Ripgrep is unavailable in this container, so I am switching to find/grep for API verification

+ 01_methane_compressor_mechanical_design.ipynb • Cell 1

Describe the engineering task — e.g., 'JT cooling for rich gas at 100 bara', 'TEG dehydratic

+ solve engineering task Auto ↕

Local Autopilot (Preview)

Step 4. Reports are made in word or html format. Iterate to improve the report and add sections and calculations to improve the report.

The screenshot displays a GitHub Codespaces IDE interface. The main window shows a document titled "Report.docx" with the following content:

Detailed mechanical design of methane compressor 10 to 25 bara at 10000 kg/hr

Author: GitHub Copilot
Date: 2026-03-25

The left sidebar shows the Explorer view with the following file structure:

- NEQSIM [CODESPACES: AUTOMATIC PANC...
- task_solve
 - 2026-03-25_detailed_mechanical_design_of_methane_...
 - step1_scope_and_research
 - analysis.md
 - neqsim_improvements.md
 - notes.md
 - task_spec.md
 - step2_analysis
 - .gitkeep
 - 01_methane_compressor_mechanical_design.ipynb
 - notes.md
 - step3_report
 - _eq_images
 - generate_report.py
 - Report.docx (selected)
 - Report.html
 - README.md
 - results.json
 - TASK_TEMPLATE
 - README.md
 - .flattened-pom.xml
 - .gitignore
 - AGENTS.md
 - azure-pipelines.yml
 - CHANGELOG_AGENT_NOTES.md
 - CODE_OF_CONDUCT.md
 - CODEOWNERS
 - CONTEXT.md

The right sidebar shows a chat window titled "DESIGNING A METHANE COMPRESSOR SYSTEM" with the following content:

make a task doing a design a a compressor compressing methane from 10 bar to 25 bar with flow rate 10000 kg/hr. Make a detaied mechanical deisgn.

Optimizing tool selection...

Reviewed 4 files

I am setting this up as a Type F design task and will now create the mandatory task folder first, then populate scope, analysis notes, and a runnable compressor mechanical-design notebook with verification outputs and report generation. Next I will verify the exact compressor feasibility/mechanical APIs in source before writing the notebook so the workflow runs end-to-end.

Ran `python devtools/new_task.py "Detailed mechanical d...`

The task workspace is created successfully under task_solve and I will now read its README plus the scaffolded Step 1 files, then build a standards-based compressor design notebook and generate results and report outputs.

Reviewed 4 files

I have the scaffold and next I am validating the exact compressor and mechanical feasibility APIs in source code so the notebook uses only confirmed method signatures and can run without guesswork.

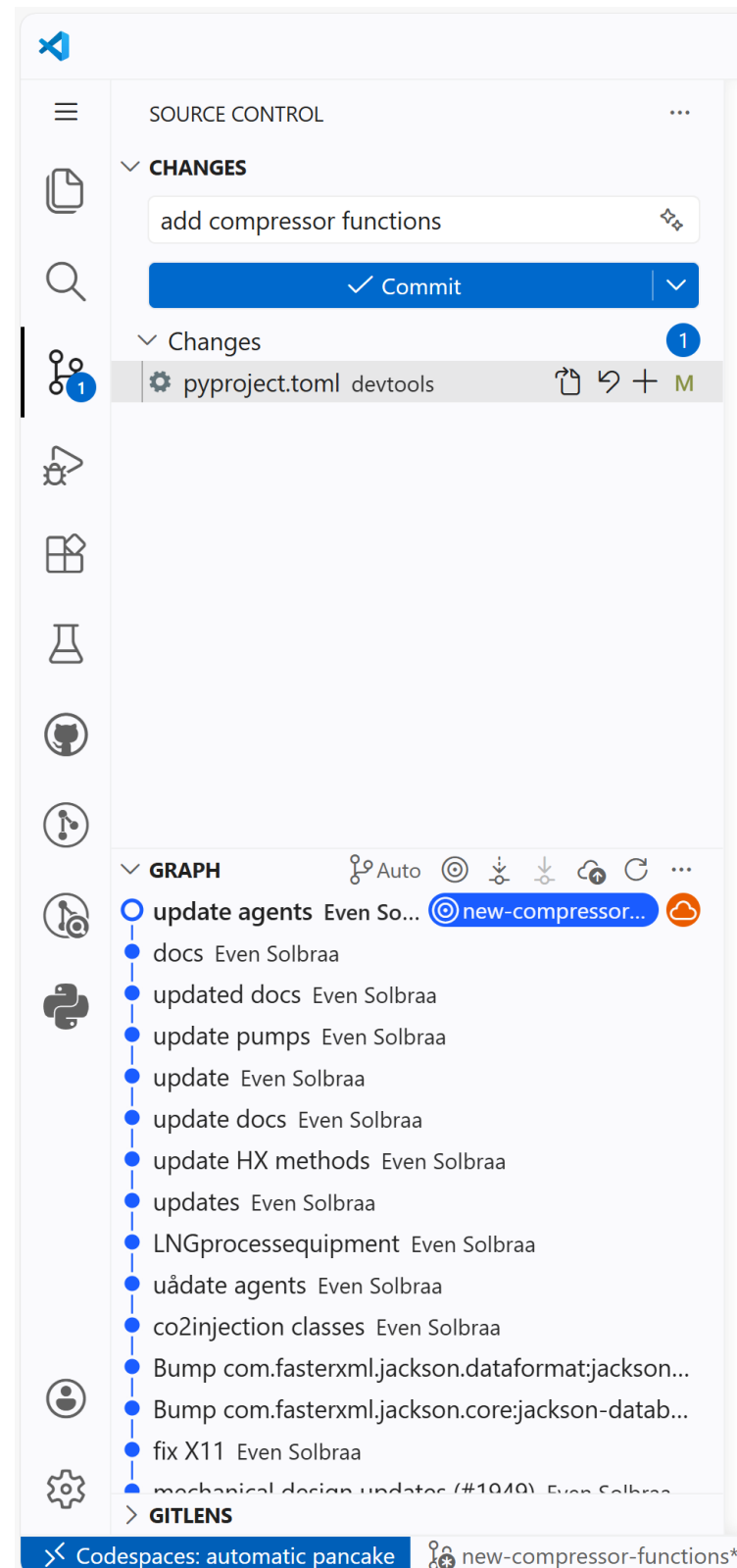
Executed commands in terminal for Compressor design analysis

Describe the engineering task — e.g., 'JT cooling for rich gas at 100 bara', 'TEG c

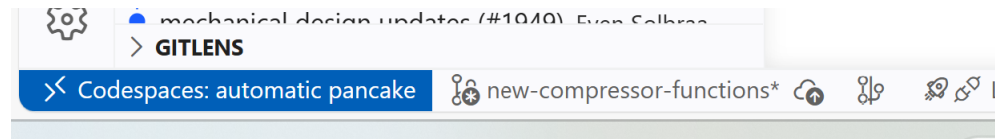
At the bottom, the status bar shows "Codespaces: automatic pancake" and "Layout: Norwegian".

Step 5. Make a PR (pull request to update neqsim with neqsim functionality).

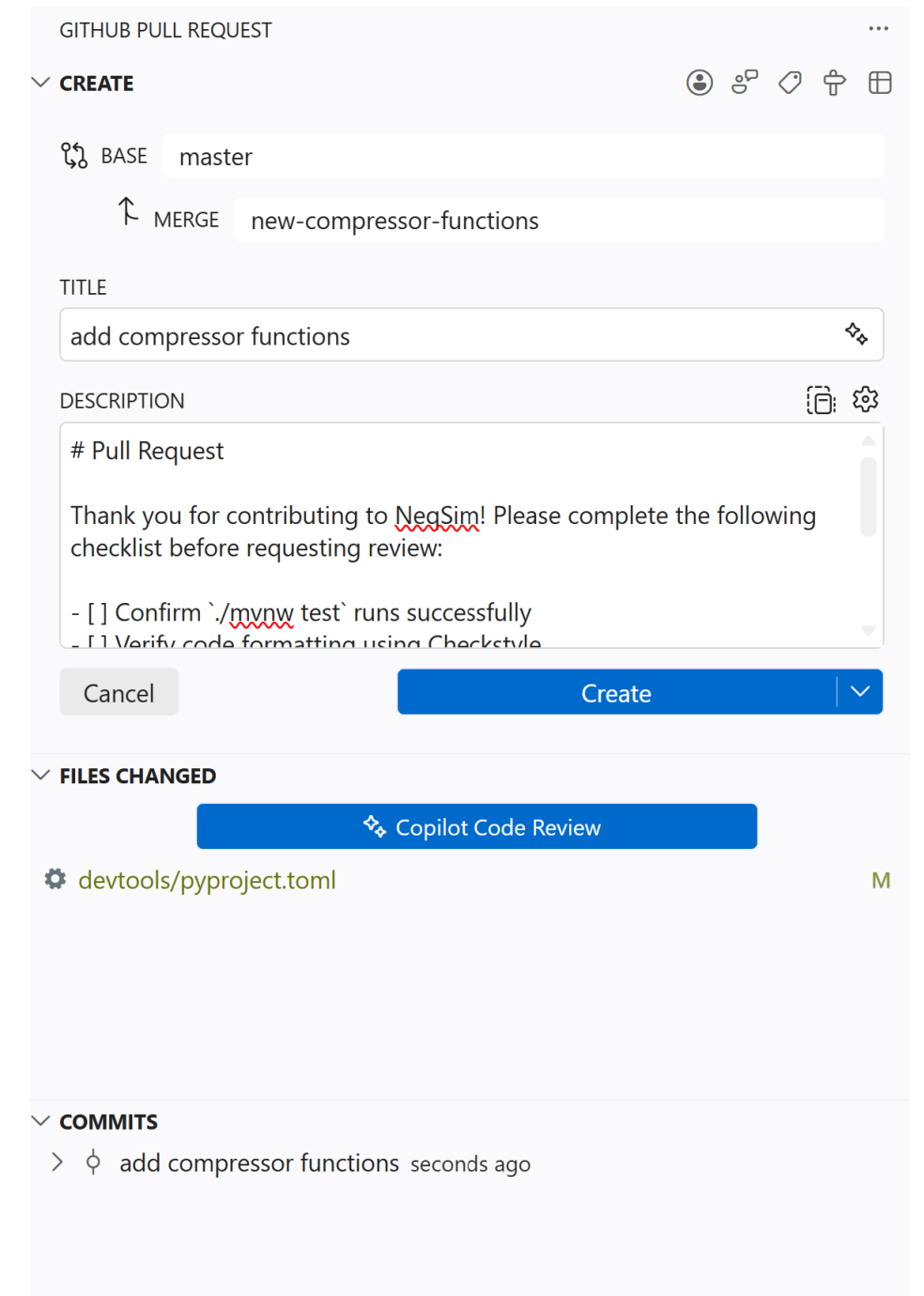
2. Add files that should be included in pull request



1. Make a new branch (give name)

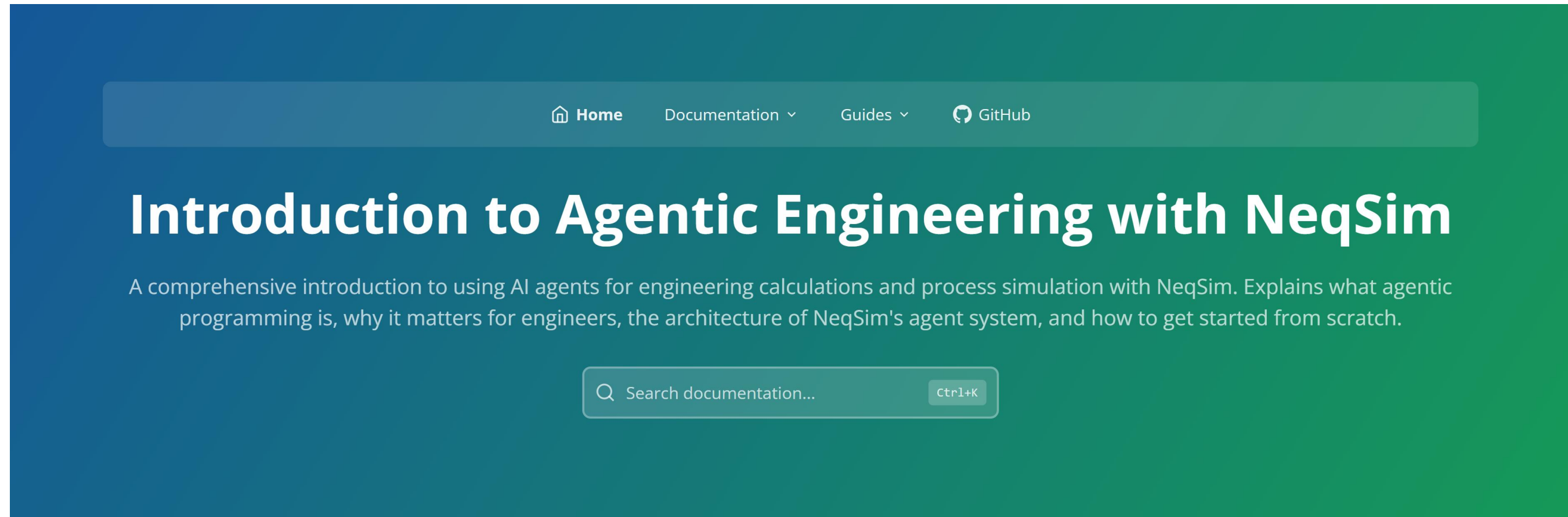


3. Give som info about what you add



Part 2.

Description of agentic engineering using NeqSim



Introduction to Agentic Engineering with NeqSim

This guide explains how NeqSim combines AI agents with a rigorous physics engine to solve engineering problems. It is written for engineers and scientists who are new to AI-assisted workflows and want to understand both the concepts and the practical tools.

What you will learn:

- What agentic programming means and why it matters

On this page

1. The Problem: AI Alone Cannot Do Engineering
2. What is Agentic Programming?
 - Key Concepts
 - Why Agents Instead of Scripts?

[Web page](#)

> _

What is the density of methane at 200 bara and 25°C?

The Hallucination Gap



Density \approx X kg/m³

LLMs hallucinate physics. They cannot solve a cubic equation of state or evaluate phase stability.

The Physics Engine

Equation of State

Compressibility Factor

Fugacity

Phase Stability

Validated: 142.3 kg/m³

Thermodynamic rigor requires mathematical iteration, not linguistic prediction.

The AI agent writes the code.
NeqSim runs the physics.
You get validated results.

Comparing Approaches: Intent vs. Rigor

Standalone LLMs

Great at understanding intent, terrible at thermodynamic math.

Agentic NeqSim

AI interprets the intent. NeqSim calculates the physics.

Standalone NeqSim / Manual Coding

Perfect physics, but requires complex Java/Python API knowledge.

Y-Axis
Natural Language Intent & Speed

High

Low

Low

X-Axis

Physics Rigor & Accuracy

High

The Agent Catalog

1 Routing & Discovery

@neqsim.help



Router: Analyzes requests and delegates to specialists.

@capability.scout



Discovery: Assesses feasibility before complex tasks.

2 Engineering Specialists

@thermo.fluid

Properties & Flash



@process.model

Flowsheets



@flow.assurance

Hydrates, Wax, Pipe DP



@mechanical.design

ASME/DNV Sizing & Cost



@gas.quality

ISO 6976



@safety.depressuring

API 520/521



3 Workflow & Productivity

@solve.task

End-to-End 3-Step workflow



@solve.process

Quick notebooks



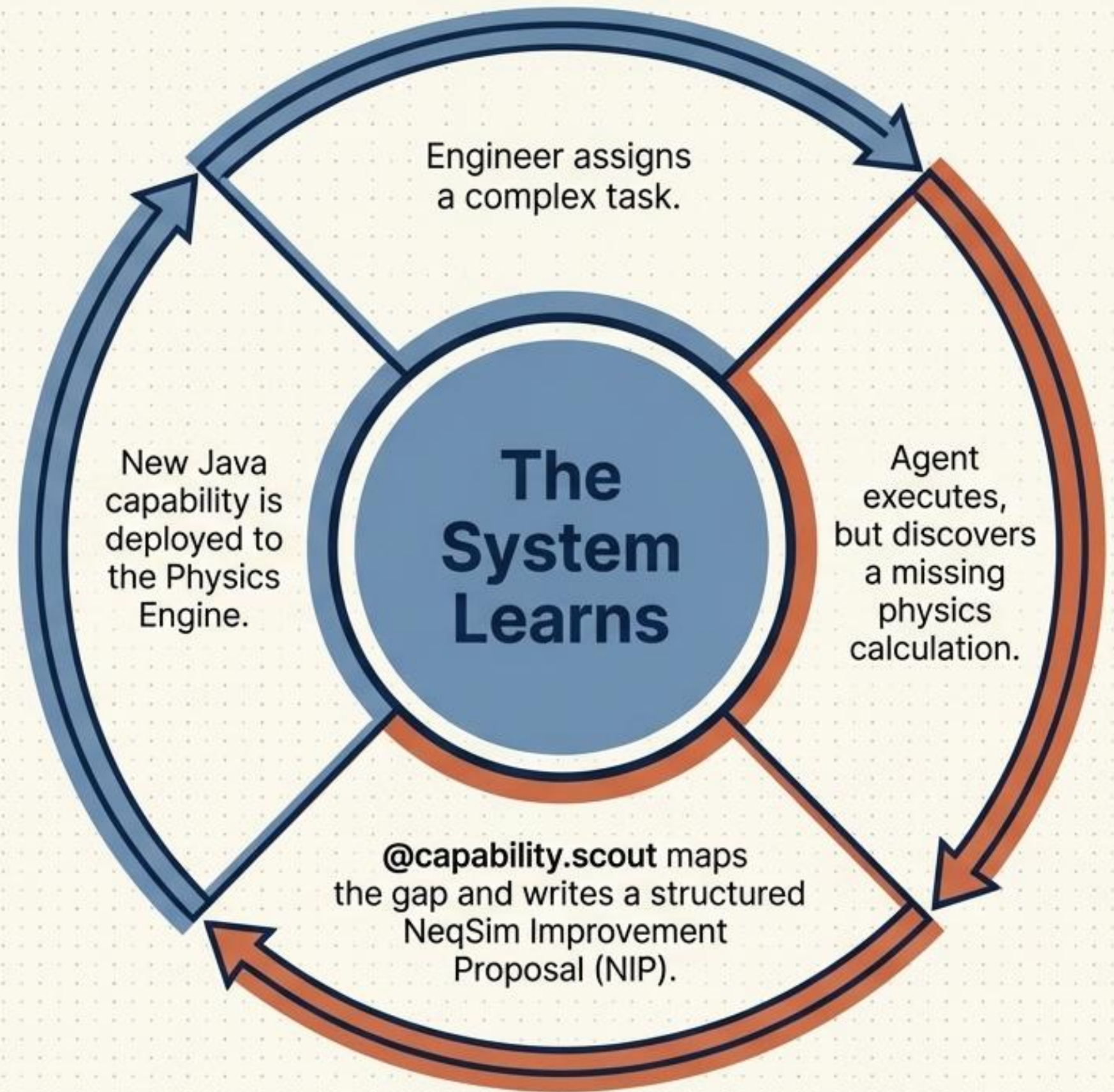
@neqsim.test

Unit testing

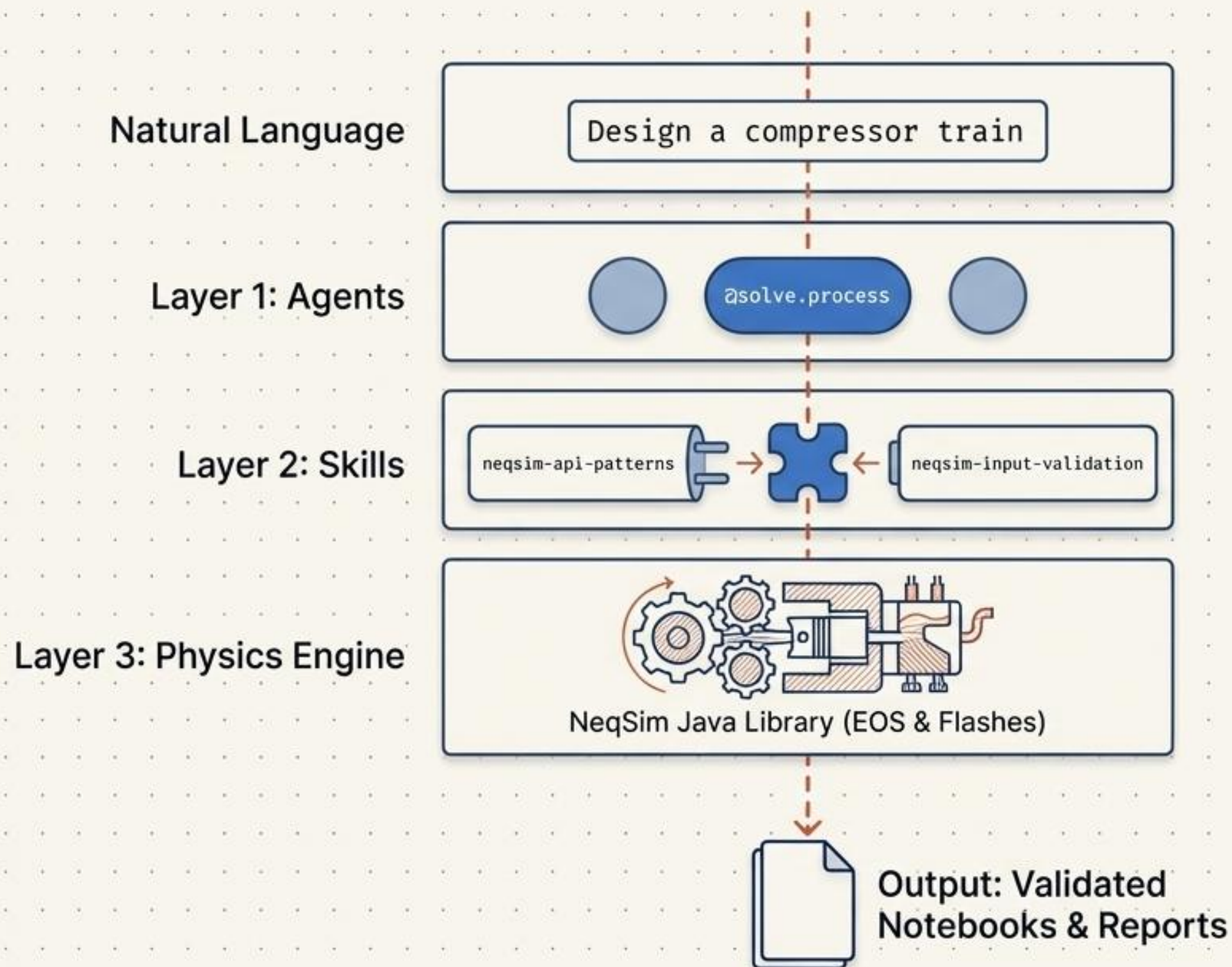


The Development Flywheel

The ultimate advantage of agentic engineering. Agents don't just solve isolated tasks—they map capability gaps in real-time. Every solved problem leaves behind structured code, updated baselines, and a smarter NeqSim engine for the next engineer.

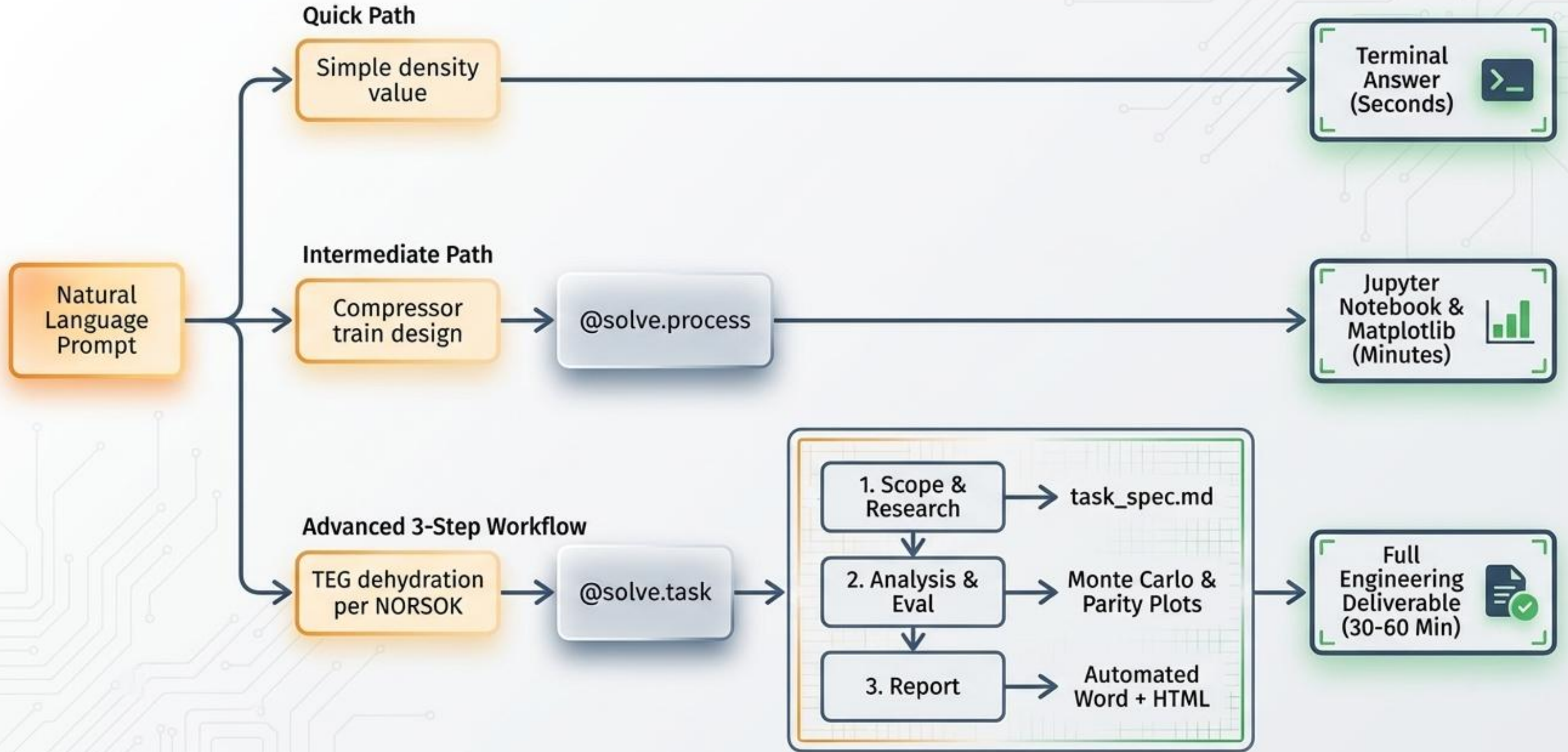


The Agentic Translation Stack



Key Insight

Agents do not guess physics. They load domain knowledge (Skills) to write executable code that calls the Physics Engine.



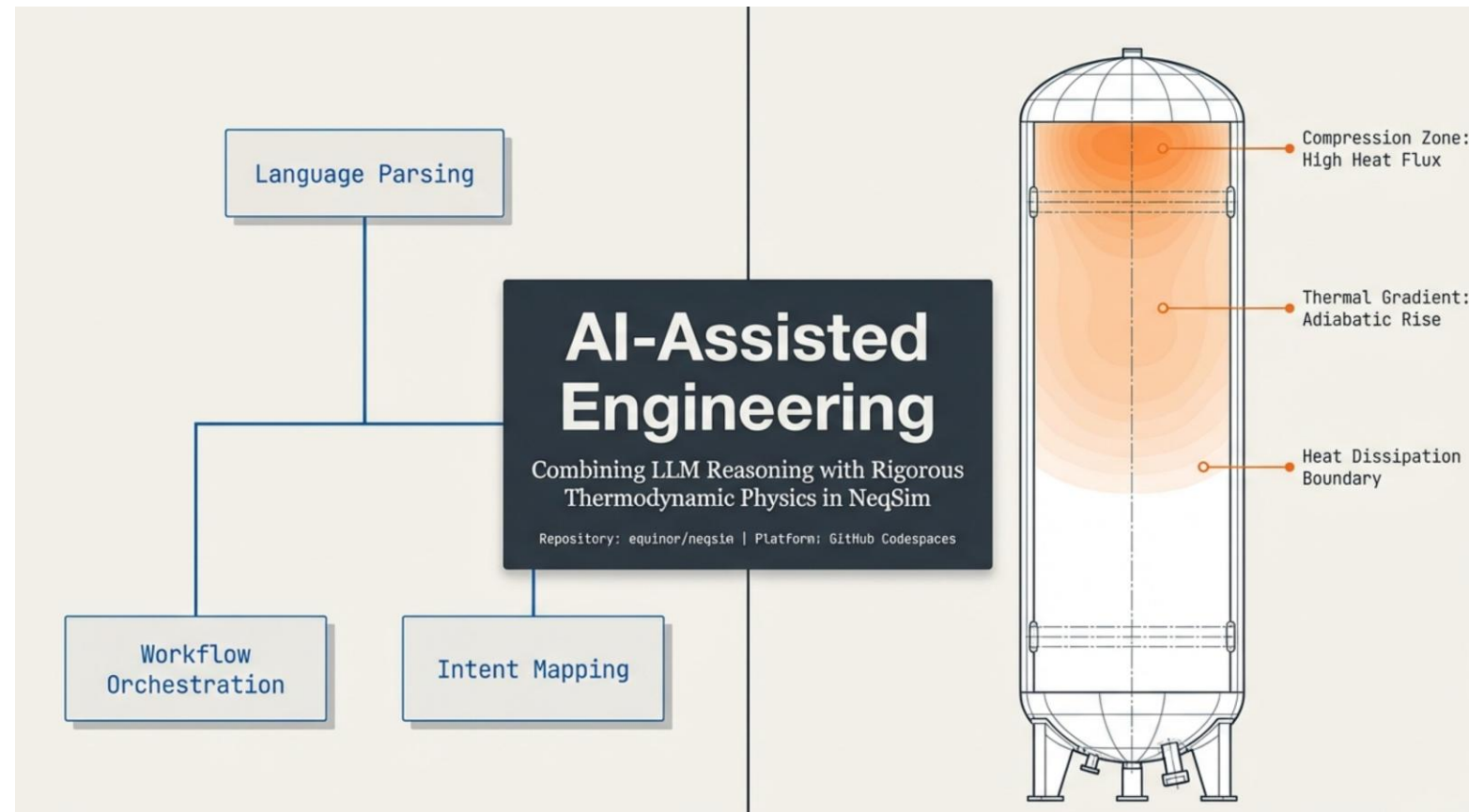
You are no longer just coding or clicking.

You are orchestrating an automated, multi-disciplinary engineering firm that documents its own work, checks its own standards, and continuously expands its own capabilities

Agentic Engineering with NeqSim. High Intent. High Rigor.

Part 3.

Demonstration of a task – filling and emptying a CNG tank



1. Background and Objectives

1.1 Context

Compressed Natural Gas (CNG) is stored in high-pressure cylindrical tanks at pressures up to 250 bara. During filling (pressurisation), the gas heats up through compression work and frictional dissipation. During emptying (depressurisation), the gas cools via Joule-Thomson expansion. These temperature excursions are critical for:

- **Structural integrity:** The tank wall material (X80 carbon steel) has a minimum design metal temperature (MDMT) of approximately $-46\text{ }^{\circ}\text{C}$. Gas temperatures must remain above this limit to avoid brittle fracture.
- **Operational safety:** Overpressure risk during filling in hot ambient conditions, and potential liquid dropout during emptying at low pressure.
- **Dynamic modelling:** Engineers need internal heat transfer coefficient (HTC) values for control system design and transient thermal models.

1.2 Objectives

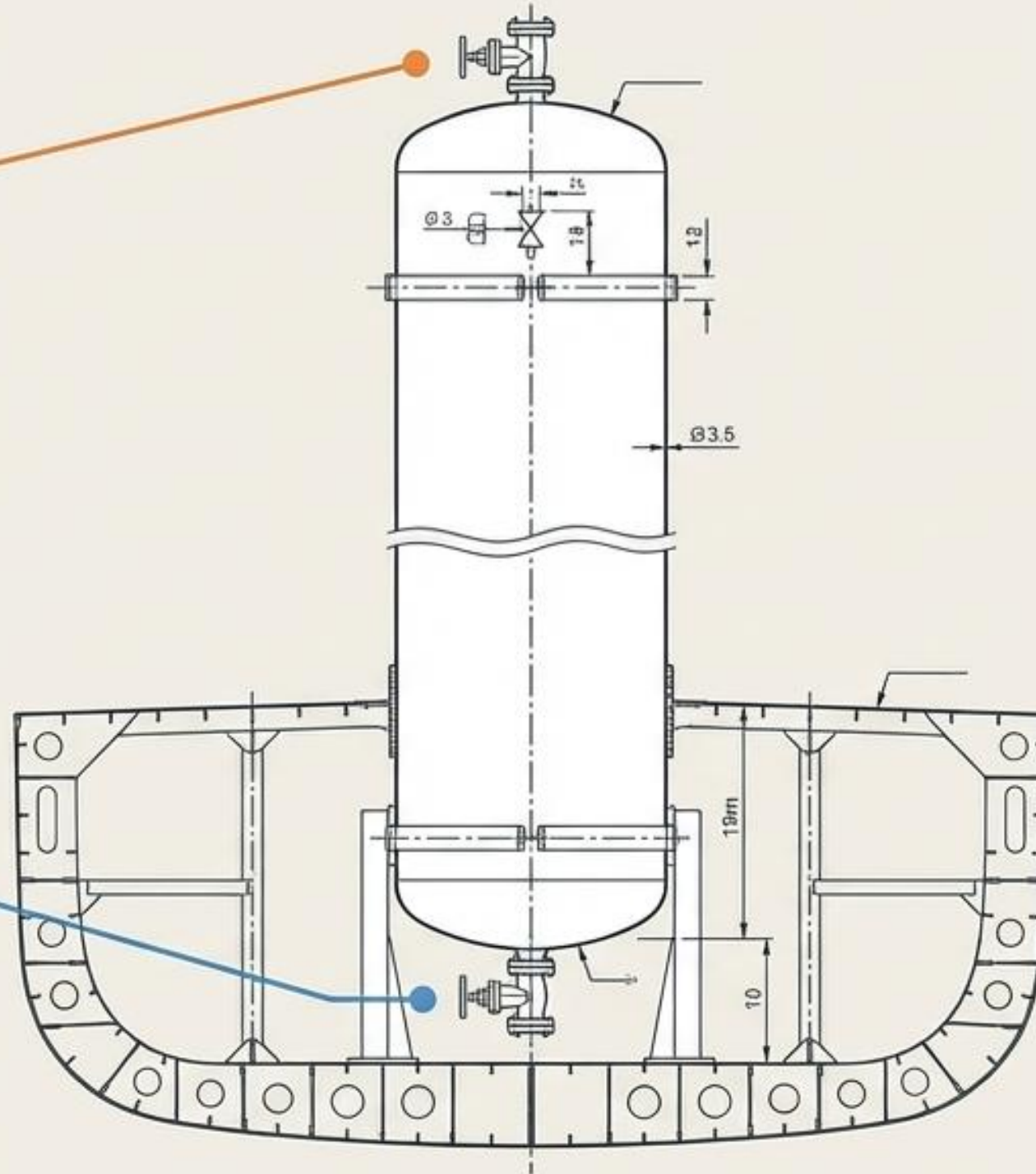
1. Determine gas and wall temperature profiles during tank filling (20 \rightarrow 250 bar) and emptying (250 \rightarrow 20 bar)
2. Derive internal heat transfer coefficients as a function of pressure, temperature difference, and operating mode
3. Assess liquid dropout risk during depressurisation
4. Compare heat transfer model assumptions (adiabatic, calculated HTC, transient wall conduction)
5. Validate simulation inputs against independent reference data (NIST, literature)
6. Verify the physical correctness of the gas-wall temperature coupling
7. Quantify uncertainty in the minimum gas temperature through Monte Carlo simulation
8. Identify and assess risks associated with CNG tank operations

1.3 Scope

The analysis covers a single vertical CNG tank with the geometry and gas composition specified in Section 2. The filling and emptying rate is fixed at $1783.4\text{ Sm}^3/\text{day}$. Ambient temperature is $15\text{ }^{\circ}\text{C}$ (base case). No external insulation is present. The analysis uses the NeqSim VesselDepressurization module with the SRK equation of state.

The Engineering Problem: CNG Tank Estimation

Compression Heating
How hot does the gas get during filling (20 to 250 bar)?



Expansion Cooling
How cold does the gas get during emptying (250 to 20 bar)?

DNV GASA Constraints

Limits: -10°C to $+50^{\circ}\text{C}$

MDMT: -46°C

Specs: 19m tall | 1m dia |
33.5mm X80 steel

Anatomy of the Engineer's Prompt

Estimate gas temperatures during CNG tank filling and emptying. Use SRK EOS for natural gas (85% methane, 5.4% CO₂, 4.6% ethane). Tank is 19m vertical cylinder, 33.5mm X80 steel. Check compliance with DNV limits (-10°C to +50°C).

Agent Routing

Invokes `@solve.task` & `@thermo.fluid`.

Physics Parameters

Directs Engine to use `SystemSrkJos` with specific component fractions.

Acceptance Criteria

Sets hard validation boundaries for Step 2 execution.

Executing Step 1: Scope & Research

Workspace Generated in Seconds

```
task_solve/  
└─ 2026-03-07_cng_tank/  
   └─ step1_scope_and_research/  
      └─ task_spec.md  
      └─ references/
```

Task Specification

Standards Applied

EN 13530 (transportable equipment) & DNV GASA (Ship approval limits).

Methods Selected

Churchill-Chu natural convection correlation, 1-D transient wall conduction.

Acceptance Criteria

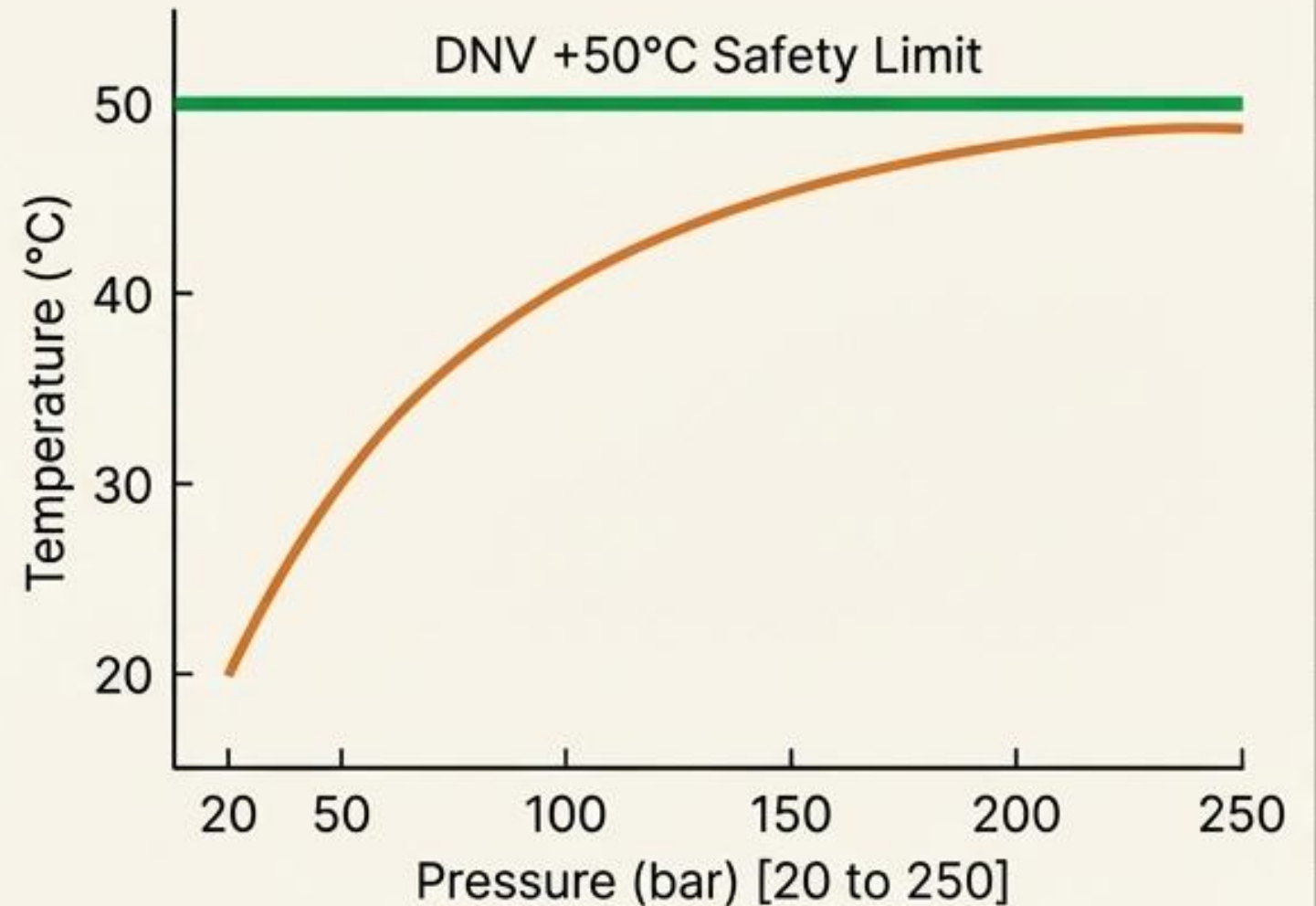
Mass balance tolerance < 0.5%. Strict adherence to the -10°C to $+50^{\circ}\text{C}$ thermal envelope.

Executing Step 2: Analysis & Evaluation

```
from neqsim.thermo import fluid

cng = fluid('srk')
cng.addComponent('methane', 0.85)
cng.addComponent('CO2', 0.054)
cng.setTemperature(20.0, 'C')
cng.setPressure(250.0, 'bara')
```

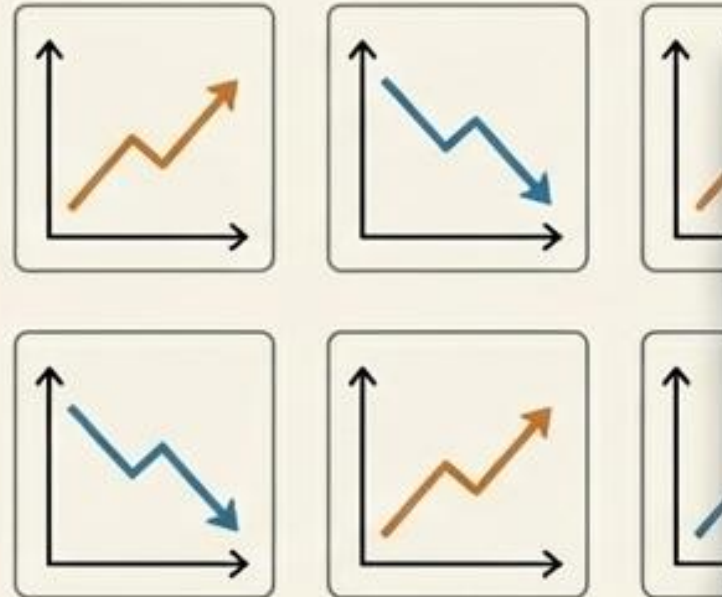
AI loads neqsim-api-patterns to write flawless engine calls.



KPI: Mass Balance Tolerance Achieved at 0.12% (Passes < 0.5% criteria).

Executing Step 3: The Automated Deliverable

```
{  
  "max_temp_C": 48.2,  
  "min_temp_C": -35.1,  
  "dnv_pass": true  
}
```



19 Automatically
Generated Figures
Georgia in Deep Charcoal

CNG Tank
Transient
Thermal
Analysis

Actionable Insights

- Outputs PASS/FAIL matrix against DNV criteria.
- Automatically sizes required 20°C heater for pressure letdown phase.