Designing a Process Simulator

Baltakatei
Baltakatei Heavy Industries

Nug
Winhill Consulting

2022-07-27

1 Users

- Chemical Engineers simulating a natural gas plant (oxygen refrigeration, oil refineries, corn syrup reactions, plastic production, coal liquefaction, metallugy, pharmaceuticals).
    - pressure drop
    - gas liquid separation
- Heat transfer engineers designing heat exchangers (heat conduction)

2 Strategies and Approaches

2.1 Finite Element Analysis

Network of nodes in space.

Parallelizable.

Pruning of simulation per tick by "sound cone" (see "light cone"). Avoiding $(n * n)$ by instead $(n \cdot k_{\text{sound}})$.

Computational more heavy because of space.

Possibly plausible to perform with modern GPUs and maybe ASICs (if we get that far).

Navier-Stokes Equation – analytically unsolvable – momentum balance.

2.2 Analytical (Traditional)

Network of nodes as needed.

Typical node is the defining characteristic of "Chemical Engineering" as a profession, compared to "Applied Chemistry" or "Chemistry".

AIChE - American Institute of Chemical Engineers - 1960s - "unit process". Not lab bench chemists.

Typical Process Simulation approach. Used because CPUs historically have been slow.

3 Design

- Degrees of Freedom of
    - Energy Balances
    - Mass Balances

- “Equation of State” (e.g. Peng-Robinson EOS)

Helium: k_1, k_2, k_3

$$PV = z\,\frac{\rho}{M}\,RT$$

$$P = k_1\,T^2 + \rho\,k_2(V, z)\,T^3$$

$$\text{output} \;=\; f(\text{input})$$

3.1 Language

C? Rust?

Avoid closed-source libraries.

What fraction of libraries are open source?

3.1.1 C

Blender uses it. GIMP uses it. So, many math libraries available with FOSS licenses enforced by maintainers of Blender/GIMP/etc.

Longer “runway”. Need to build up more complex structures to actually have a runnable program.

3.1.2 Rust

Math library-limited. (May have to spend a lot of time writing libraries. >.<)

More appropriate for small secure “black box” programs. “fast scripts”. (e.g. crypto/authentication).

Similar to C in “runway” aspect.

Used by Mozilla?. Deeper FOSS roots?

3.1.3 Go

Known for computation speed and accessibility.

Overtaken by Rust in some aspects.

By Google. Resembles Java (accessible as a scripting language). Designed with parallelism in mind?

May have significant amount of CompSci math libraries.

3.1.4 Python

Slower: An interpreted language, not compiled.

Larger community than for Lua.

3.1.5 Lua

Slower: An interpreted language.

Scripting language like Javascript. Integration/configuration scripts. (e.g. Scripting API to application).

Similar to Python.

3.1.6 Java

Slower than C, Rust. Compiled to byte-code for Java VM (at runtime).

3.2 License

Avoid DWSIM fate. (educational funnel into commercial product) :(

Debian-compatible?

FSF compatible? (GPLv3)

3.3 Interoperability

- CAPE-OPEN - talking to other process simulation executables (mostly Windows)

- CSV, spreadsheet (FOSS)

- JSON (RDFa?) - Python Dict

- Save as CSV for ease of editability of files used to adjust process parameters.

3.4 User Interface

- Strongly consider using CLI with CSV as inputs and outptus (focus on engine instead of GUI).

4 Action items

- Try coding rudimentary engine with C as practice. (goal: get something compiling)
    - single-component mass balancer
    - converges a single return loop
- Shotgun search broadly for libraries that may be related.
    - Equation-of-State (flash calculation)
    - Fluid-holdup calculations (e.g. T, P, changes over piping)

-