“CORAL: Compressive Orbit Recovery Algorithm”

By Randy Miller, 2025

Synopsis

CORAL is an exploratory and analytical model that generalizes and transforms the classical Collatz conjecture operators, defined by modular arithmetic:

If f(n): {3n+1 if n =1mod2, n/2 if n = 0mod2

Into if f(n): {mn+b if n=rmodz, n/z if n = 0modz, b,r =(1,z-1),

with the former defined as the expansion class of operators, and the latter as the compression operators, into dynamical systems theorized to exhibit similar properties of convergence.

In this generalized form, b and z are adjusted to consider all range outside of [-1, 1] for z. Variable m is constricted more in the form z < m < (z^2), and when gcd(z, m)=1. This guarantees full representation of modular residues under z and avoids divergent behavior as you might see in {3n+1 if n =1mod2, n/2 if n = 0mod2. Divergent behavior and support for negative integer ranges will require more sophisticated lines of analysis.

CORAL is designed as a tool not only to create these new systems, but to analyze their properties, and while some heuristics regarding behavior have been adopted and implemented into the codebase, development is underway to evaluate branching patterns between different numbers and convergences agnostically. No assumption regarding behavior of a novel dynamical system is strictly held. The framework is purely exploratory.

Regarding general software design, CORAL is built in Python 3.11.9, is heavily modularized, and primarily leverages Ray processes and a [I forget the fancy term for this] hybridized data storage model between SQLite for fast reads and DuckDB for fast writes. Upon public release, the system will use JupyterLabs for implement of a frontend and be packaged as a zero-login desktop app, with an ancillary of prebuilt and sophisticated analysis and visualization tools that have already been sketched out and prototyped for open-science experimentation. Simple extension of the algorithm to federate data for cluster-scale analysis by porting chunked files of relevant system data to cold storage based on dynamic monitoring of system capacities has also been mapped out for implementation.

Benefits to Community

In the general math community, the Collatz conjecture has captured many minds since its introduction in 1937 and has gained some notoriety in its particular draw for hobbyists. Its behavior is indeed fascinating, and much of its underlying mechanisms remain elusive. Most of the more successful approaches to the problem take a more probabilistic, rather than deterministic, approach.

Integration with SageMath could leverage a sophisticated knowledge base of mathematical visualization techniques and analytical tools for Python to bolster and expand the capabilities of this system, dramatically improving the project's capabilities and utility of output. SageMath mentorship could provide guidelines for standardization of CORAL notation and data analysis towards existing mathematical frameworks and fields.

Deliverables

CORAL deliverables have been subdivided into project phases. This algorithm is a personal project developed independently. To respect the time and resources of SageMath mentors, deliverables have been organized first by utility and then more theoretical applications of the framework.

Phase 1: Core pipeline logic refinement

CORAL's framework was originally developed around exploring the quirks of the classical permutation of the Collatz conjecture. Its algorithm was first delimited to examining what happened when multiplying odd integers n by 3, which in some cases, shortens the total Collatz cycle length by exactly one step for 3n relative to n.

A core refactor of pipeline logic is, as of April 8, in its final phases of debugging. As mentioned above, this refactor adopts the framework

$f(n)$: {$mn+b$ if $n=r \bmod z$, n/z if $n = 0 \bmod z$, $b,r =(1,z-1)$,

and we can abstract the multiplicative factor described above into factor F. F is determined dynamically to be an integer that will produce the most reliable short changes in sequence lengths, for deeper analysis of behavior.

Users will be permitted to input Z and the integer range (and optionally F), and M is given as a short range of choices before initialization. Polarity of B can be toggled by the user (think of this as the transformative operator between modular residues) to be positive, negative, or both.

Factor F is determined in a more complicated fashion, and much more computationally expensive but it returns the projected best possible number based on simple statistical comparisons.

Finally, one more simple feature for general system analysis has been added. The Collatz conjecture in its classical permutation essentially asks if all positive integers >= 1 will fall into an infinitely repeating loop of the form (4, 2, 1). It has been demonstrated that extension of even the standard conjecture into negative numbers generates other infinite loops. Thus, it is critical to understand if a novel dynamical system is mapping to other infinite loops. These loops are captured as systems are generated, and the point or points at which all other numbers condense into these infinite loops is recorded.

The pipeline also allows for toggling between examining the behavior of the system in full within a given integer range, and condensing into a found structural irregularity in Collatz sequence construction hypothesized to be an artifact of a minimal attractor set. Subsequent analysis is informed of this option and customized to mode.

This phase is very nearly complete. Debugging is expected to be finished within the week (Apr 11).

Phase 2: Hardware-sensitive system analysis, compressive chunking into cold storage, abstraction of subsequence generation

CORAL examines very messy and unpredictable systems. This can outpace hardware quickly. However, the algorithm itself is designed with minimal overhead, and is intended to be built cross-platform and for whatever hardware available to a user.

The program will advantage dynamic and basic built-in Python system monitoring capabilities to ensure that resources are being used effectively by CORAL.

The basic use structure of CORAL is to generate all integers in a system up to a certain range once, in a small job, and then slowly expand that range. Resumption expects this, is seamless, and is carefully integrated into the data handling.

Conservative estimates of future memory usage and file size will be informed by previous iterations. If a job seems too big to store with existing resources uncompressed, then before initialization, the pipeline will be modified to batch chunks of the job in roughly equivalent size, compress them, and store to a desired location (personal Cloud, compressed location on disk, external drive, etc). This will subsequently limit analytical availability, but all data can be imported onto a larger system to perform these analyses, and some periodic metadata from each chunk can still be obtained. Crucially, any higher-order infinite loops could still be detected, and all higher-order convergences with smaller numbers will be preserved with the constructed chunking format.

This aspect of the project is also fully fleshed-out in terms of conceptualization, but implement is underway. Once implemented, testing flexibility of the program on different systems will be needed.

A version that is prototyped for these functions, and cross-platform compatibility, will be completed by June 2nd at the latest. The repository will be shared privately with SageMath on May 8th, and pull requests and feedback are very much welcomed from anyone interested but not expected.

Phase 3: Testing backend compatibility cross-platform, implementing bug monitoring, and prototyping GUI

CORAL has been developed fully on a MacBook Air 2020 with M1 chip, 8GB RAM, 256GB SSD. A 2TB external SSD was recently purchased for testing the system chunking generation. As this is an independent project, examining implement on various systems for effectiveness and logic gaps will be more challenging. However, the project has been assigned a private GitHub, and downloads could be made available privately to interested SageMath contributors to suggest refinements of all aspects of the backend for compatibility, safety, and usability with any hardware.

Likewise, bug monitoring and fixing can be addressed during this phase.

Ideally, testing system compatibility with a cluster would also take place during this phase, if possible.

Outside of these tests, work will begin on the JupyterLabs GUI. A perfunctory GUI has been already created with tkinter, which requires some minimal prior understanding of the variables involved in system generation to use but is sufficient for the most basic use case. If CLI implement would be preferred before adding the GUI, that can happen.

A large quantity of analytical tools for these systems have already been generated, but they must be generalized and interface with the pipeline upon full population of a system within a range, or at least be aware of the unique parameters set for system generation before performing analysis.

This phase of the project will be addressed from June 2nd to September 1st, and the base version of the app will be released publicly along with an arXiv paper as the coding period for GSoC ends. While the base version of the GUI will be fairly simple to create, this creates a great deal of breathing room for developing further analysis and visualization tools, integrating with SageMath, and ensuring that the first public release is clean and functional.

Proposal Timeline

April 8 – April 11:

- Phase 1 of CORAL will be completed independently. Slight tweaks and bug fixes are needed.

April 11 – May 8:

- Dedicated time for examining the SageMath codebase for possible areas of integration and enhancements, and implementing them.
- Deeper exploration of statistical analyses and mathematical properties that could be applied to CORAL for rigorous testing and evaluation of system dynamics- some measure of normalization without sacrificing data granularity is most valuable.
- Allotted private brainstorming time for improvements to code, prototyping further extensions, and whatever else may arise.

May 8 - June 2:

- A private GitHub repository link will be shared with SageMath for interested parties to review if the project is accepted. This will be read access. You are welcome to explore all aspects of the existing codebase. If you would like supplemental information before accepting mentorship of this project, reach out at any time with specific inquiries by email, and I will be happy to supply more detailed information regarding the code structure or mathematical framework.
- If desired, simple presentations of the logic involved in CORAL have already been prepared, to provide a deeper contextual understanding. This will ultimately be refactored into a complete and robust readme before sharing repository access, but often the human element in explaining complex topics is helpful. I am more than happy to explain anything about this software in depth at any time, over Zoom or through text publications.

June 2 – July 14:

- Begin immediately with nitpicking code structure. If you think I could be handling any data or standardizing functions more effectively, or that I should tweak my analytical methods, or just don't find my syntax or code approachable enough, let me know. This is my first software, and I would really appreciate a critical eye.
- A more macroscopic priority will be nailing down compatibility of the platform to other systems. Once it has been ensured that CORAL will run well with other hardware, and can perform all desired data management functions, then extension into building out the GUI will begin. A complete plan for the GUI interface has been

laid out. A comprehensive development log of the project with detailed information on every future feature can also be given if desired.

July 14 – July 18:

- Midterm evaluations. By this point, the backend should be as refined and close to publishable for a first version as it can get. Expansion of analytical tools and visualizations can be rolled out with future updates.

July 14 – August 25:

- This period will, as a priority, involve construction of the GUI with all basic features and result in a downloadable app package. CORAL is highly customizable but will be shipped as a research tool with some prebuilt options for analysis, expanding the user base to those who do not have a robust background in math or computer science but are curious about dynamical systems. Support for community packages could be enabled at this time.
- Porting in the many existing tools as available and generalizable post-pipeline functions for the GUI.
- Two novel and lightweight visualization tools for CORAL systems have been drawn out, but implementation will be more complex. One in particular will require a modular launch and use of some animation software to monitor real-time transformations of integers under different system dynamics. This may extend out past the GSoC period, and that is perfectly okay. The focus is just on prepping the base software for release. Further plans for the visualizations can be shared upon request.

Related Work

- CORAL and Collatz
  - As mentioned before, study of the conjecture has largely originated from a probabilistic approach. Open generalization and extension of its dynamics to larger systems could result in deeper insights.
  - This software doesn't claim to have found complete structural decomposition of the 3n+1 system. The aim is to provide a window into the unknown.
- CORAL and Open Science
  - Accessibility of scientific software to the public can increase enthusiasm towards open problems and potentially lead to important insights. One instance of this in biology was [look up the protein sequencing thing that was crowdsourced]
- CORAL and SageMath
  - As experimental mathematical software, CORAL overlaps naturally and elegantly with the purpose of SageMath, and its purpose and structure is

quite similar. The software itself is just condensed to analysis of these unusual dynamical systems. Mentorship from and integration with SageMath could enormously increase the analytical tools available to CORAL.

Biographical Information

I am a graduating senior at the University of Kansas, with a major in biochemistry. I began creating this project after I was introduced to the conjecture on March 4th of this year purely out of a deep curiosity, and it evolved far past my expectations.

I have no prior background in coding besides a basic semester-long class I took in high school in 2018 for Python. I have leveraged advice from internet forums and archives, textbooks, LLMs, and people kind enough to provide it to refine my software. (I have maintained complete chatlogs with LLMs through this process and am happy to share them.) I currently only know Python and minimal R but am happy to expand to other languages if the project requires it.

While I haven't branched out into this specific field prior to this project, I work as a teaching assistant for cadaver dissection, and I have conducted and documented original research before while working there (developing novel approaches for removing delicate tissues that have already been fixed with formalin). I will be leaving this job May 17th as I am graduating. My experiences as an instructor inform my philosophy towards work, and I am excited to develop new skills if accepted for GSoC.

I am comfortable with working independently, as my current role is largely self-directed, but I constantly question myself and ask for feedback. While I'm not strictly from a computer science background, I have a strong grasp of the scientific method and formal inquiry and understand my own personal limitations and biases.

As my project has evolved, I have felt more of a need to seek out constructive criticism, which led me to find GSoC on April 5th. I'm aware that I have not yet met all the SageMath guidelines for accepting contributors, namely in submitting a pull request, and I do understand if you are not able to accept the project at this time. I have just simply not had the time to put that together yet, but I would be happy to after submitting today. I would just like to submit something of quality when doing so.

However, even if you do not have any space for me this year, I would still be happy to share read access to my repository if interested. Any feedback for further development would be invaluable, and no matter what, I would be immensely grateful for some direction towards publishing this as a free, accessible tool.

I have been exhaustive in my documentation of my process and have plenty of other materials that I am happy to share. I will be monitoring the email associated with this proposal for any feedback or questions you may have.

Final Considerations

This scope is flexible and already mapped out in significant detail. I have no commitments for the summer and anticipate having the app completed and publishable well before the deadline. This allows for plenty of time to consider more fun things, like analysis inter- and intra-system.

Thank you for your time and consideration!