Chengyuan Ma

Phone: (857)-928-8339 (U.S.) / +86-133-9107-8300 (China)
Email: `MaCY404@mit.edu` / `chengyuanma@protonmail.com`
Blog: `https://danglingpointer.fun`
500 Memorial Drive, Cambridge, MA 02139-4306

Education

- **Massachusetts Institute of Technology** (B.S., expected graduation May 2025) 2021—Present
  Major: Computer Science and Engineering. GPA: 5.0
  Courses taken in major (†taking, $^{\forall}$not A, using EECS dept. old numbering scheme):
  - 6.905 Large-scale Symbolic System
  - 6.854† Advanced Algorithms
  - 6.034† Artificial Intelligence
  - 6.046† Design and Analysis of Algorithms
  - 6.041† Introduction to Probability
  - 6.004 Computation Structures
  - 6.009 Fundamentals of Programming
  - 6.147 Battlecode Programming Competition
- **Shanghai Foreign Language School Affiliated to SISU** (middle & high school) 2014—2021

Skills

- **Programming**: Rust, Python, Scheme, C++, Kotlin, Java, Julia, Go, JavaScript, RISC-V Assembly, Pascal. (ordered in approximately decreasing proficiency and recent usage)
- **Hardware**: Raspberry Pi, Arduino.
- **Software**: TEX, Vim, Jetbrains IDEs, Wolfram Mathematica, Git, Inkscape.
- **Languages**: Mandarin, English.

Research Experiences

- **Social Interactions in Robotics**, InfoLab, MIT Computer Science and Artificial Intelligence Lab. (CSAIL) Feb 2022—Present
  As *undergraduate researcher*,
  - Developed a GPU-accelerated recursive MDP solver with 1000x speedup with CUDA and C++.
  - Learned and experimented with RL algorithms such as DQN, VPG, and PPO.
  - Adapted a Unity environment (VirtualHome) for our lab to experiment multi-agent decisions.
  - Program Toyota HSR robot with ROS.
- **Order Book Construction and Generative Models**, InfoLab, MIT CSAIL, in collaboration with Nasdaq. Jun—Aug 2022
  As *undergraduate researcher*,
  - Researched and experimented realistic conditional order book generation with GANs and variational LSTMs.
  - Developed parallelized preprocessor in Rust to convert raw Nasdaq ITCH 5.0 feeds into analyst-friendly formats such as Apache Arrow 2 data frames and reconstruct order books.
  - Developed reactive web app with Plotly Dash to interactively visualize real-world and generated order books.
- **"Isomer Enumeration of Simple Hydrocarbons,"** published on *University Chemistry*. Jan—Mar 2021
  - Implemented in Julia algorithm to count hydrocarbon isomers and explored asymptotic properties of the problem.
  - Intrigued by the problem when isomerism was taught in high school; Encouraged by teacher to turn it into a paper.
  - Though the algorithm itself was not new, the derivation is the simplest in the Chinese literature.
  - Accepted and published on *University Chemistry*, a Chinese journal focusing on chemistry education, under support from high school chemistry teacher.
- **"Exploring PID Auto-tuning on Brushed DC Motors,"** May—Nov 2018
  - Designed a PSO-based heuristic algorithm to automatically tune PID controllers for brushed DC motors.
  - Implemented the algorithm and a GUI toolbox with C++.
  - Worked one-on-one with a researcher from Shanghai Institute of Technical Physics, Chinese Academy of Science to produce a research report. The research report in Chinese can found here.

Awards and Honors

Competitive Programming
- **3rd Place**, First-year Teams Tournament, MIT Battlecode Competition 2022
- **Bronze**, National Olympiad in Informatics (NOI), China 2020
- **First Prize and Top 3%**, NOI (Provincial), Shanghai 2016—2020
- **Silver**, Asia-Pacific Informatics Olympiad (China) 2020
- **Platinum Division**, USA Computing Olympiad 2018—2020

Mathematical Modelling
- **Finalist**, Team 2002090, The Mathematical Contest in Modelling (MCM) 2020
- **Meritorious Winner**, Team 1903350, MCM 2019

Robotics (VEX Robotics Team 8825A)
- **World Finalist**, VEX Robotics World Championship 2016, 2017
  Led team in 2017. Served as chief programmer in 2016.
- **Division Finalist**, VEX Robotics World Championship 2018
- **Think Award**, VEX Robotics World Championship 2017
  Awarded for excellent engineering log and innovative model-based overheat-prevention algorithm.

Work Experiences

Student Advisor, Unity Technologies (Shanghai) Jul. 2019

- A member of a 12-member development team focused on producing gamified programming curriculum for ages K-12.
- Assess and provide advice on quiz and example problems, thus improving curriculum engagement.
- Design course material of several units involving game engine elements such as Rigidbody, Transforms and Material Interfaces.

Extracurricular Activities

- **Software and Hardware Engineer**, "Project λ", Tsinghua University GIX Innovation Camp Jul. — Aug. 2019
  - My team brainstormed, built, and presented to industry experts a prototype to address accessibility concerns for the elderly, "Project λ" no-slip shoes, within a 72 hour window.
  - Designed and programmed prototype with Arduino and Raspberry Pi.
  - Awarded Outstanding Student (6/36); Project awarded Outstanding Project (2/9) for idea, prototype functionality, and presentation.

Personal Programming Projects

I enjoy learning through various personal programming projects. Project names contain links to repositories.

1. **Web-Scheme– A Scheme compiler with a WebAssembly based runtime** (Scheme + C++, 2022): Final project for 6.905. The compiler performs hygienic syntactic expansion, optimized CPS transform, and tail-call-optimized code generation. Also implemented and tuned the GC in the runtime. The compiler compiles itself.
2. **Just-LaTeX– Painless true LaTeX-based equation rendering for Markdown $\rightarrow$ HTML** (Rust, 2022): A Pandoc filter automatically extracting LaTeX fragments from Markdown and replacing them with SVGs rendered by true LaTeX engine. It is more powerful than MathJax and is very easy to integrate in various workflows, now powering my blog. Includes an SVG optimizer that uses a B-tree-in-Trie data structure to efficiently identify and deduplicate similar paths, leading to 80% reduction in file size.
3. **NTTP – An obfuscated SOCKS5 client and server** (Go, early 2019): An attempt to apply my knowledge in competitive programming to wrestle with the national firewall(GFW) of China. Used mod-257 NTT (a variant of FFT on finite fields) to obfuscate SOCKS5 traffic so that the GFW does not recognize it as a proxy. Peak throughput around 30MiB/s. The proxy became ineffective around early 2020 as GFW upgrades to more sophisticated probing strategies.
4. **Ekho – An ICMP-based SOCKS5 proxy** (Rust, early 2021): A sequel to NTTP. Uses KCP (an ARQ protocol) over ICMP to ensure reliable transmission and circumvent GFW probing on TCP and UDP. Re-implemented KCP from scratch, as well as multiple congestion control methods such as Tahoe, CUBIC, BBR, and PCC. While never blocked, it is found to suffer from other limitations: NAT compatibility, low routing priority, lack of system-level packet pacing, etc. Nevertheless, the project was primarily intended as an opportunity for me to learn the principles of transport protocols as well as practice asynchronous programming.
5. **Particle – A lexer generator** (Rust, early 2019): Generates efficient lexers following textbook algorithms of constructing NFA from regular expression and reduction to DFA. Flexible and declarative grammar to define lexers on user-defined token types. Location information is preserved in tokenization. Supports Unicode; Allows dumping DFA in Graphviz format for visualization and debugging.
6. **A web app dewarping book pages** (JavaScript & WASM, early 2020): Similar to CamScanner and Office Lens, but can handle curved edges of pages. Users drag control points to mark page boundaries. The app then automatically calculates perspective parameters, fits cubic splines to edges, and produces de-warped rectangular images of the page. All the computation takes place on the browser side. The core algorithm is written in C to be compiled into WebAssembly for better performance. The derivation of the algorithm is documented on my blog.
7. **An automatic differentiation framework** (C++, late 2019): An effort to create a toy version of TensorFlow. It constructs static computation graphs and uses reverse accumulation to calculate gradients. Supports matrix arithmetic, softmax, dropout, and common activation functions; Defining new operators and optimizers is straightforward. Performs well on MNIST and simple networks. However, a mistaken early design decision limits tensor dimension to 3, so CNNs are not attempted. GPU acceleration is never planned.
8. **An interpreter for a homemade scripting language** (C++, mid 2018): The language is functional and minimal, supporting basic control flows, lexical-scoped closures, and arbitrary-precision arithmetic. The interpreter has a hand-written recursive-descent parser with robust error recovery and decent error messaging. Code is executed by recursively walking the AST directly. It also has a GUI to visualize AST for debugging purposes.

More on `https://github.com/ma-chengyuan`.