UCSD CSE 291 (Differentiable Programming) Assignment 4:

Final project

The idea of the final project is to do something interesting with loma. You can extend loma with some compiler features (like supporting recursion or have some object oriented features), or use loma to do some differentiable programming (like implementing a control algorithm, or an inverse renderer). Or you can do both. In fact, if you choose to do the latter, you will likely need to extend loma in one way or another.

Grading. The project will mostly be graded by the technical sophistication, and the completeness of the report. To encourage people to pursue ambitious projects, the more ambitious the project is, the less requirement that is to the completness of the project. For example, if you aim to work on a research idea that is novel enough to be published at a top venue, we will not judge you using the standard of a technical paper. It is sufficient to show your work and the initial prototypes and experiments you develop, even if the initial result does not show practical benefit yet.

Teaming up. For the final project, you can either do it alone or form a group of two people. We do expect that you'll do more difficult projects with two people, but only slightly so. As every software developer know, more people doesn't immediately lead to faster progress, so be careful. What teamwork buys you is probably more capacity to brainstorm ideas and the friendship.

Logistics. Before 5/20, please let us know what you plan to do for the final project by submitting a 1-2 pages brief report to Canvas. Schedule a chat with us if you have any question. We will have a checkpoint at 5/29: send us a brief progress report in a few pages on Canvas describing what you did and what you plan to do next. The final report is due on 6/13. Please submit a report (no format restriction) and your code. Describe what

1 Random ideas

Below we provide some ideas for the final project. They are not sorted by the difficulty. Feel free to come up with your own ones.

Checkpointing. Implement checkpointing in loma, either automatically or by user instruction. How should a user specify the checkpointing scheme? What is the optimal way to do it automatically?

Dependency analysis to reduce stack size. In Homework 2 and 3, we push the variable overwrite into a stack for every assign statement. Can we reduce the stack size, especially for long loops? This requires some analysis of the dependencies between variables. A paper to read is "Linearity Analysis for Automatic Differentiation" from Strout and Hovland.

Distributed AD. Add OpenMPI support to loma so that it can be distributed to multiple machines.

Remove atomic add in loma. Do as much dependency analysis as possible, and remove atomic add when it's possible to do so safely. A paper to read is "Automatic Differentiation of Parallel Loops with Formal Methods" from Huckelheim et al.

Cross country mode. Implement cross country mode (e.g., vertex elimination using greedy Markowitz deegree) in loma. You don't have to support loops. You can try out your own heuristics! (Even if it doesn't work better, it's fine.)

Sparse Jacobian. Implement graph-coloring-based sparse Jacobian computation in loma. Again, you can try out your own heuristics for solving the graph coloring problem!

Randomized automatic differentiation. Implement the randomized automatic differentiation from Oktay et al.

Functional automatic differentiation. Implement a continuation-based reverse mode in loma. Try to support recursion.

Implicit differentiation. Implement implicit differentiation in loma and use it to solve some interesting problems. Think hard about the interface: how should user specify the implicit function? A paper to read is "Efficient and Modular Implicit Differentiation" from Blondel et al.

Differentiable sorting. Implement a differentiable sorting algorithm in loma. Think about some fun applications.

Stochastic differentiable programs. Add support of random number generation to loma, and make it possible to switch between score and pathwise estimators.

Parametric discontinuities. Make it possible for loma to handle the Dirac delta distribution coming from differentiating the if/else conditions under integration. A paper to read is "Systematically Differentiating Parametric Discontinuities" from Bangaru and Michel et al.

Relaxation. Make if/else statements in loma differentiable by automatically turning them into soft operations.

Optimal control/differentiable physics. Implement a physical simulator in loma, solve for the initial/boundary conditions, forces, or other parameters, by differentiating the simulation.

Hamiltonian Monte Carlo. Implement Hamiltonian Monte Carlo in loma and use it for some interesting problems (e.g., uncertainty quantification).

Implementing Komogorov-Arnold Networks or Capsule Networks. There are innovative neural networks that do not fit well into existing deep learning frameworks in terms of efficiency. Implement them in loma and hopefully show that you can achieve faster performance.

Bundle adjustment. Implement bundle adjustment for recovering camera poses from images using loma.

Differentiable shadertoy. Take a random program from shader toy, make it differentiable, and learn for the parameters over some loss function (e.g., match a drawing, or have more high frequency details, etc).

Equivarient neural networks. Implement a equivariant neural network in loma. A paper to read is "Tensor field networks: Rotation- and translation-equivariant neural networks for 3D point clouds" from Thomas et al.

Fluids control. Implement "Fluid Control Using the Adjoint Method" from McNamara et al. in loma.

Differential manipulation. Implement "Differential Manipulation" from Gleicher and Witkin in loma.

Space-time constraints. Implement "Space-time Constraints" from Witkin and Kass in loma.

ReLU fields. Implement "ReLU Fields: The Little Non-linearity That Could" from Karnewar et al. in loma.

Differentiable program simplification. Given a program, find a cheaper program to approximate it. Optimize the parameters of the cheaper program to increase the quality.

Physics engine using loma. Write a physics engine in loma in which the user supply the potential energy and variables, and your engine would automatically synthesize the physical trajectory. See this blog post for details.