CME 213

SPRING 2019

Eric Darve

Syllabus

People

Instructors:

- Eric Darve, darve@stanford.edu
- Colfax engineer (openMP, multicore)
- NVIDIA engineers

Teaching assistants:

- William Jen
- Chenzhuo Zhu

Web sites

https://stanford-cme213.github.io/

- Lecture notes, reading material, computer code
- Homework, final project

https://piazza.com

- Forum discussion, Q&A
- You need to register

https://gradescope.com/

- Entry Code: **M42X7G** for registration
- Will be used for grading and regrades

Grading, homework, project

- 1 pre-req C++ homework + 4 homework assignments: 65% of grade
- One final project: 35% of grade

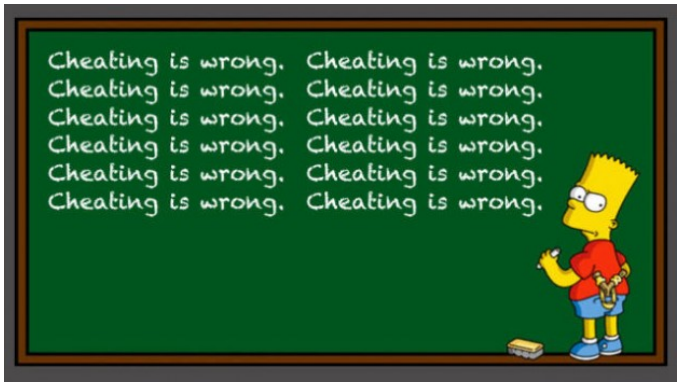
- The first homework is due Wednesday April 10.

Honor code

Honor code:

- “that they will not give or receive aid in examinations; that they will not give or receive unpermitted aid in class work, in the preparation of reports, or in any other work that is to be used by the instructor as the basis of grading”

- You cannot copy someone else’s computer code. **All answers must be your own.**
- Concerns are reported to the Office of Community Standards. The standard sanction for a first offense includes a one-quarter suspension from the University and 40 hours of community service.
- **The instructor will automatically issue a "No Pass" or "No Credit" grade.**



Computers

- For the first few homework assignments, you can use FarmShare or your own computer. You need a standard C/C++ compiler.
- For the GPU homework assignments and final project, we will use the Google Cloud Platform.
- We have some credits available from a teaching grant from Google to support the class.
- Instructions will be given later on how to use this resource.

The final project

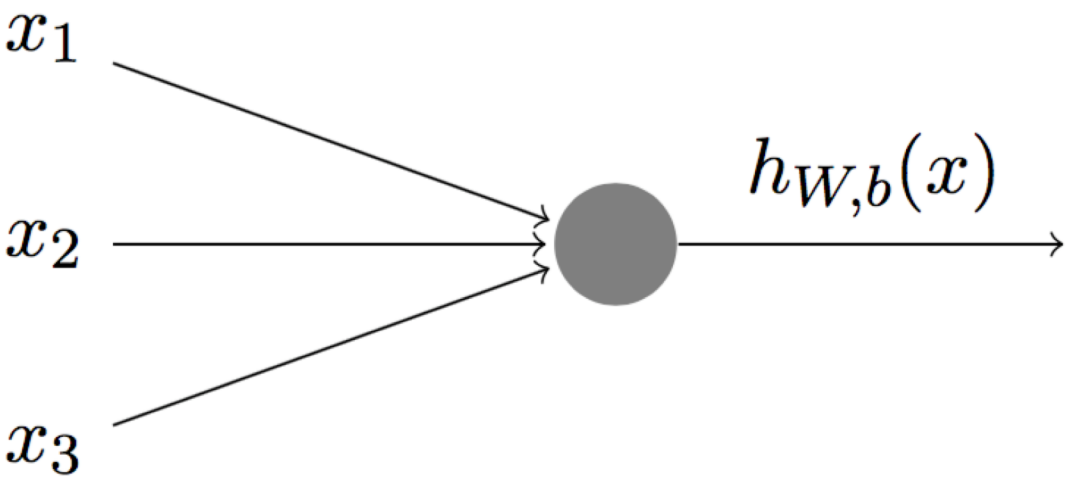
- You will be given a final project to work on towards the end of the quarter.
- Everyone will work on the same project.
- The final project will be on neural network for machine learning.
- The project will involve multi-GPU programming with CUDA and MPI.

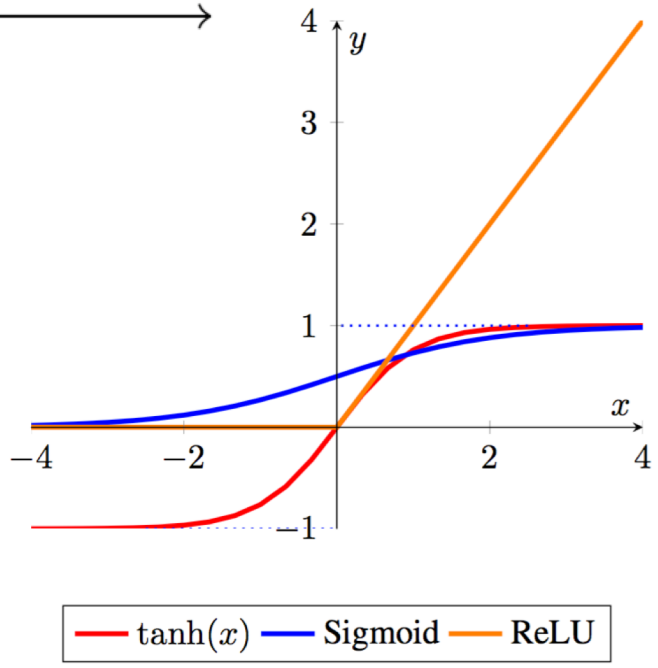
Neural networks

- Simple but powerful
- “Mimic” neurons in the brain: a network of nodes that receive inputs and generate outputs.

What's a neuron exactly?

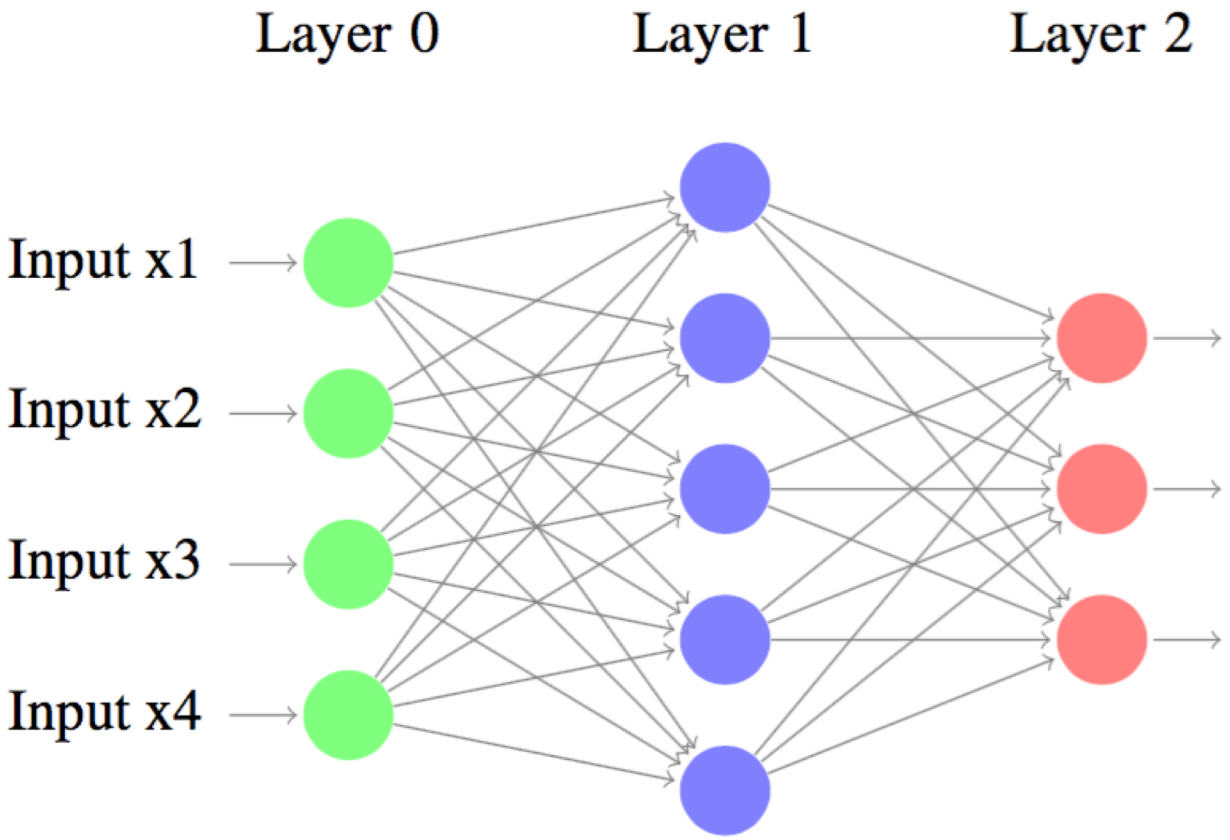
It's a highly simplified model of a biological neuron.





Neural network

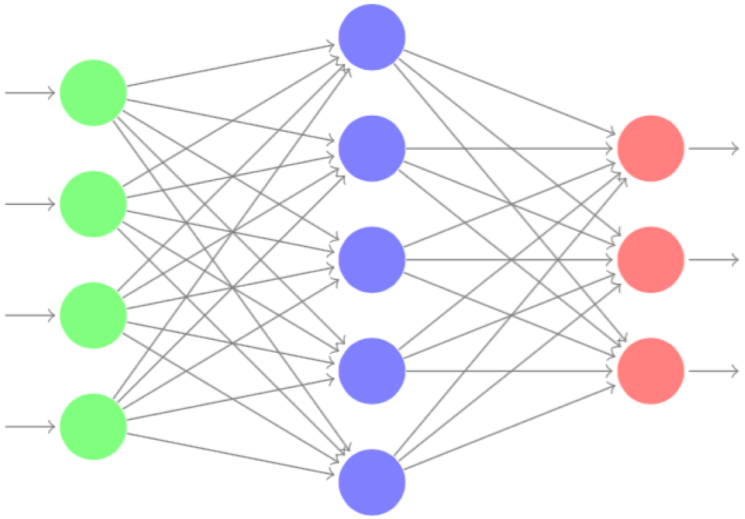
Assemble many neurons to create a neural network.



Input to neural network: handwritten digit

Output: probability of digit

- The network gives a probability vector of size 10
- Entry i is the probability that the image is showing the digit i.

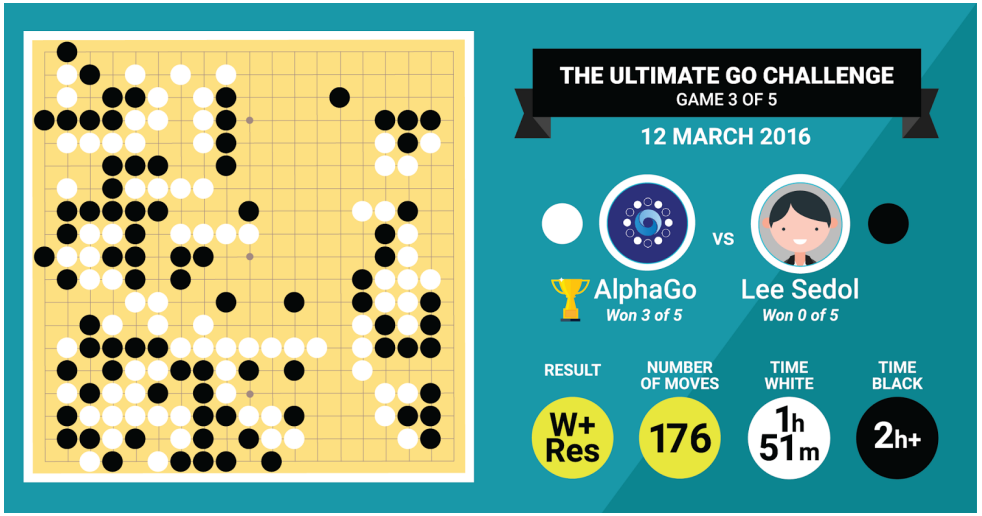


P(0)
P(1)
P(2)
P(3)
P(4)
P(5)
P(6)
P(7)
P(8)
P(9)

Example of NN in action

Double spiral problem on TensorFlow Playground

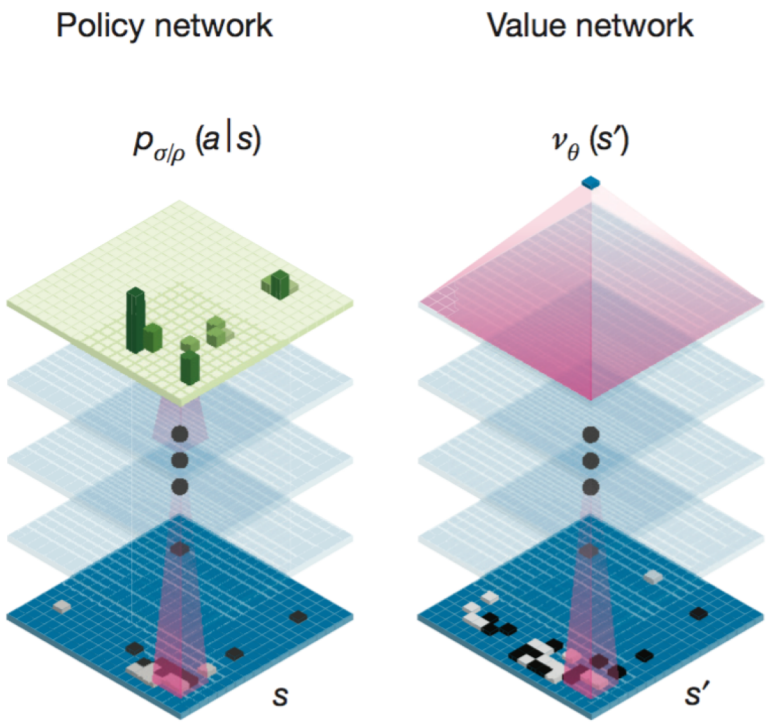
AlphaGo



- AlphaGo (Google DeepMind) defeated legendary Go player, Lee Sedol (9-dan professional with 18 world titles).
- The final score was 4 to 1.
- AlphaGo played a handful of highly inventive winning moves, several of which were so surprising they overturned hundreds of years of received wisdom.
- Newest kid on the block: AlphaGo Zero.

AlphaGo

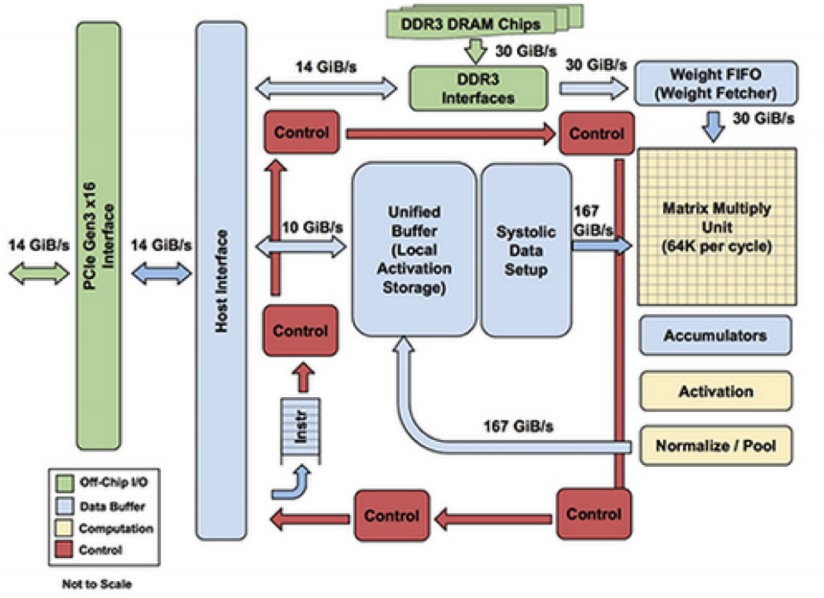
- We will talk more about this later in the quarter.
- DeepMind used deep learning to create a program to play Go.
- It uses parallel computing and GPUs to use neural networks to play Go.



AlphaGo: hardware

Program	Hardware
AlphaGo Fan	176 GPUs
AlphaGo Lee	48 TPUs
AlphaGo Master	Single machine with 4 TPUs
AlphaGo Zero	Single machine with 4 TPUs

TPU



A TPU includes the following computational resources:

- Matrix Multiplier Unit (MXU): 65,536 8-bit multiply-and-add units for matrix operations
- Unified Buffer (UB): 24MB of SRAM that work as registers
- Activation Unit (AU): Hardwired activation functions

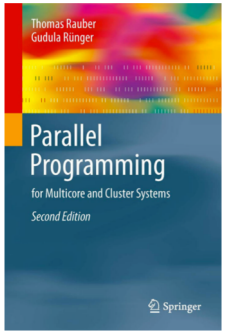
Books

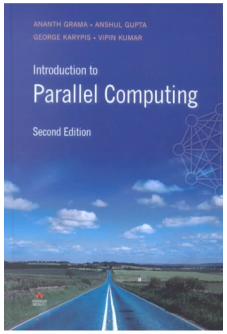
Good news: all books are available electronically from the Stanford Library. Just go to:

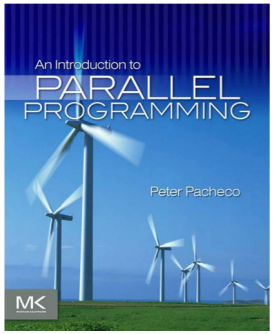
http://searchworks.stanford.edu/

OpenMP, MPI, Parallel programming

- Parallel Programming for Multicore and Cluster Systems, Rauber and Rünger. Applications focus mostly on linear algebra.
- Introduction to Parallel Computing, Grama, Gupta, Karypis, Kumar. Wide range of applications from sort to FFT, linear algebra and tree search.
- An introduction to parallel programming, Pacheco. More examples and less theoretical. Applications include n-body codes and tree search.

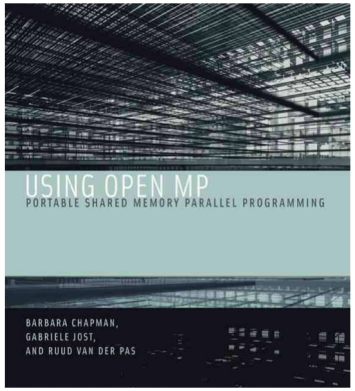


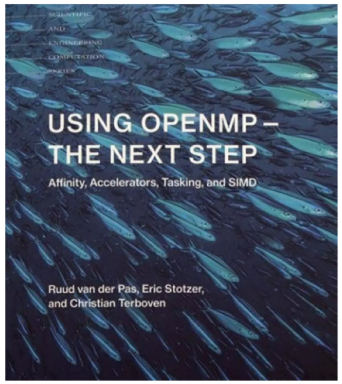


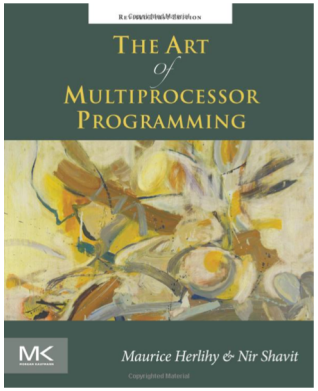


OpenMP and Multicore Books

- Using OpenMP: portable shared memory parallel programming, Chapman, Jost, van der Pas. Advanced coverage of OpenMP.
- Using OpenMP—The Next Step: Affinity, Accelerators, Tasking, and SIMD, van der Pas, Stotzer, Terbo. Covers recent extensions to OpenMP and some advanced usage
- The Art of Multiprocessor Programming, Herlihy, Shavit. Specializes on advanced multicore programming.

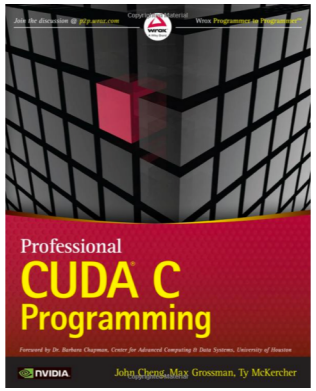


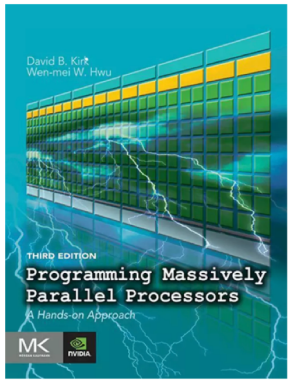


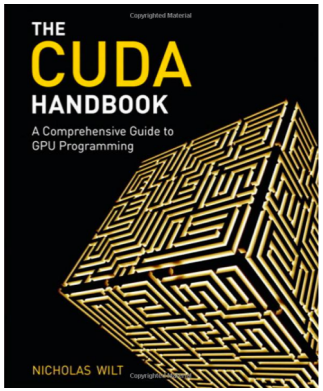


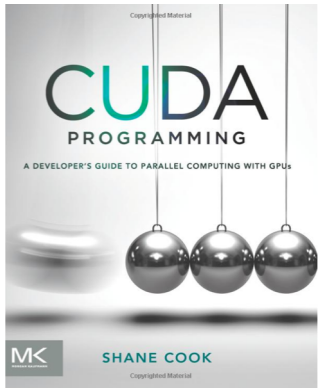
CUDA books

- Professional CUDA C Programming, Cheng, Grossman, McKercher; recent book, recommended for this class
- Programming Massively Parallel Processors: A Hands-on Approach, Kirk, Hwu; in its 3rd edition now; covers a wide range of topics including numerical linear algebra, applications, parallel programming patterns
- CUDA Handbook: A Comprehensive Guide to GPU Programming, Wilt
- CUDA Programming: A Developer's Guide to Parallel Computing with GPUs, Cook



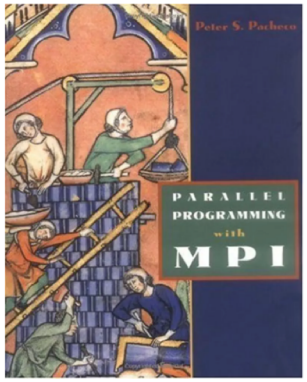


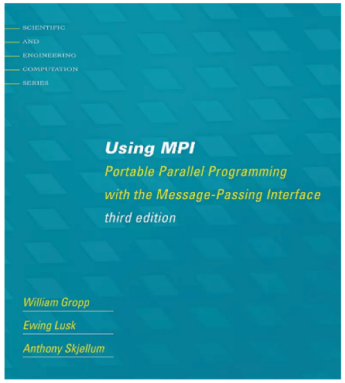


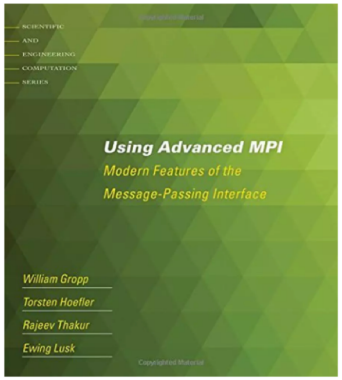


MPI books

- Parallel Programming with MPI, Pacheco. Classic reference; somewhat dated at this point
- Using MPI: Portable Parallel Programming with the Message-Passing Interface, Gropp, Lusk, Skjellum. Very complete reference
- Using Advanced MPI: Modern Features of the Message-Passing Interface, Gropp, Hoefler, Thakur, Lusk. Same authors as previous entry; discusses recent and more advanced features of MPI







CUDA books: preferred references

CUDA online documentation:

- Programming guides and API references
  http://docs.nvidia.com/cuda/index.html
- Teaching and learning resources from NVIDIA
  https://developer.nvidia.com/cuda-education-training

Uploaded reading material on class page:

- CUDA_C_Best_Practices_Guide.pdf
- CUDA_C_Programming_Guide.pdf

What this class is about

We will focus on how to program:

- Multicore processors, e.g., desktop processors: Pthreads, C++ threads, openMP
- NVIDIA graphics processors using CUDA
- Computer clusters using MPI
- We will cover various numerical algorithms for illustration: sort, linear algebra, basic parallel primitives.

What this class is not about

- Parallel computer architecture
- Parallel design patterns and programming models
- Parallel numerical algorithms.
- *CME 342: Parallel Methods in Numerical Analysis;* parallel algorithms
- *CS 149: Parallel Computing;* hardware, synchronization mechanisms, parallel programming models
- *EE 382A: Parallel Processors Beyond Multicore Processing*; SIMD programming, parallel sorting with sorting networks, string comparison with dynamic programming, arbitrary-precision operations with fixed-point numbers

What this class requires

- Some basic knowledge of UNIX (ssh, makefile, git, etc.)
- Good knowledge of C and C++ (including pointers, templates, standard library)
- Proficiency in scientific programming, including testing and debugging