Neil G. Dickson

neil.g.dickson@gmail.com www.neildickson.com

I live for difficult challenges that have the potential to change the world. I specialize in software performance optimization and high-performance simulations. I love to explore and advance new realms.

Work Experience

Software Developer
Side Effects Software, Toronto, Ontario, Feb. 2012-Present

- Developing and optimizing Houdini animation and effects software used in major motion pictures
- Multi-threaded and otherwise optimized a wide variety of geometry processing and rendering operations, often guided by requests and reports from customers
- Restructured and extended key geometry data structures, as well as adding new functionality, and propagating use of new interfaces to a large existing codebase (a few million lines), generally ravaging the code's status quo

Software Developer and Quantum Algorithms Researcher
D-Wave Systems Inc., Burnaby, British Columbia, Jun. 2009-Jan. 2012

- Optimized AQUA@home distributed, multi-threaded quantum physics and classical physics simulation software: 10x speedups on several different applications in addition to multi-threading
- Managed simulations deployed on 15,000+ volunteer computers, as well as volunteer relations
- Designed and analysed 4 new quantum optimization algorithms
- Designed 3 major experiments to examine and characterize quantum computers

Education

Bachelor of Computer Science, Minor in Mathematics, Co-op Option
Carleton University, 2004-2009

- Completed, 11.9 of 12.0 CGPA, Corresponding letter grade: A+
- Carleton University Medal in Computer Science

Other Activities

Educational YouTube Series: You Can Solve the Schrödinger Equation
http://www.youtube.com/neilgdickson/

- I teach, with ample doses of awkward humour, how to solve the Schrödinger equation of quantum physics without needing to know any complicated math in advance; just plus, minus, times, divide.

Integrated Development Environment Programming (Inventor IDE)
http://www.codecortex.com/ide/

- Fully-integrated documentation
- Designed for the flexibility to support multiple languages together easily
- Supports auto-completion, refactoring, and error-detection

Selected Publications

Entanglement in a Quantum Annealing Processor
Published in Physical Review X, May 2014

- Experimentally demonstrates persistent quantum entanglement, even at thermal equilibrium, in an 8-qubit subset of a quantum computer
- I designed a mathematical construct used experimentally to help demonstrate entanglement.

Thermally Assisted Quantum Annealing of a 16-Qubit Problem
Published in Nature Communications, May. 2013

- Experimentally demonstrates that thermal noise does not break quantum annealing, and can even be used to gain a significant performance advantage
- I designed and ran the experiment.

Tunnelling Spectroscopy using a Probe Qubit
Published in Physical Review B, Jan. 2013

- Demonstrates experimental method to probe the spectrum of a quantum annealing computer
- I designed the mathematical construct on which the experimental method was based.

Elimination of Perturbative Crossings in Adiabatic Quantum Optimization
Published in New Journal of Physics, July 2011

- Gives simple construction to eliminate effect causing quantum optimization to take exponential time
- (single-author paper)

Quantum Annealing with Manufactured Spins
Published in Nature, May 2011

- Demonstrates that an 8-qubit quantum computer uses quantum physics, not classical, to compute
- I optimized and managed the distributed, multi-threaded simulations of the quantum processor.

Does Adiabatic Quantum Optimization Fail for NP-complete Problems?
Published in Physical Review Letters, Feb. 2011

- Invalidates claims adiabatic quantum optimization of NP-complete problems takes exponential time
- I worked closely with the co-author in developing the analyses and examples.

Importance of Explicit Vectorization for CPU and GPU Software Performance
Published in Journal of Computational Physics, June 2011

- Heavily optimizing both CPU and GPU Metropolis Monte Carlo simulations casts doubt on assumption that compilers optimize adequately and fairness of CPU vs. GPU performance comparisons
- I performed the optimizations and performance experiments.