Curriculum Vitae for Martin Sandve Alnæs

Personal information

Address:	Maridalsveien 192 0469 Oslo	E-mail:	martin@xal.no
		Phone:	+47 41 61 21 85
Born:	1981-08-06		

Summary

Expert software developer with a PhD in scientific computing and a broad background in programming from low level high performance and embedded software to high level frameworks and applications. In addition to software engineering and computer science my background includes mathematical modeling, visualization, numerical methods, and mechanical engineering. I love the search for elegant solutions to complex problems, working either alone, together with experienced peers, or while supervising less experienced coworkers.

Technical skills

Frameworks	FEniCS, Boost, STL, MPI, OpenMP, Swig, NumPy, SciPy, Ipywidgets, OpenGL, WebGL, Three.js, Backbone.js
Languages	C, C++ (C++11/14), Python, JavaScript (ES6), GLSL, HTML, CSS, Bash, Fortran, Matlab, Java
Tools	Git, Mercurial, Subversion, Jupyter Notebook, CMake, Make, pip, Virtualenv, Docker, Py.test, GTest, GCC, Valgrind, Linux, LaTeX

Education

2006 – 2009	PhD in Computer Science, The Faculty of Mathematics and Natural Sciences, University of Oslo.
2004 – 2006	Master in Applied Mathematics, Mechanics and Numerical Physics, Department of Mathematics, University of Oslo.

Professional experience

2017 –	Consultant at Expert Analytics.
2015 – 2017	Senior Research Engineer at Simula Research Laboratory.
2011 – 2015	Postdoctoral Fellow at Simula School of Research and Innovation.
2010 – 2011	Engineer III - Software Engineer at Cisco.
2009 – 2010	Senior Software Developer at TANDBERG ASA (acquired by Cisco).
2006 – 2009	PhD Candidate at Simula School of Research and Innovation.
2005 – 2005	Scientific Assistant at Simula Research Laboratory.
2005 – 2005	Teaching Assistant at Department of Mathematics, University of Oslo.
2004 – 2005	Teaching Assistant at Department of Informatics, University of Oslo.
2002 – 2004	Freelance C++ Developer for music production software "Renoise".

Languages

English	Fluent
Norwegian	Mother tongue

Personal skills

Communication	I'm experienced in structuring communication and work around bite sized chunks for efficient development teamwork. My research background provides experience writing and presenting technical material for documentation or education. Supervision and mentoring of smart people at Bachelor, Master, PhD and Post Doc level has taught me to adjust the message to the listener, and listen and learn myself where possible.
Development	I strive for a deep and detailed understanding of my primary programming languages, and I'm conscious about the design balance between rapid development and maintainable solutions. I have worked with both on-site and distributed development teams as well as solo.
Management	With experience from different team styles and some small project leadership I can be a productive coworker in both structured agile teams and navigate more chaotic situations such as distributed open source groups.
Mathematics	An analytic mindset and a solid mathematical background provides a good foundation for learning and modeling new problem areas.

Extended descriptions of selected projects

Activity	3D Visualization in Jupyter Notebooks with WebGL
Period	2017-05 – 2017-09
Role	Project leader and main developer
Staffing	3 developers
Volume	100%

Description	As one deliverable of the larger Horizon 2020 project "OpenDream-Kit" we worked to improve the state of 3D visualization in Jupyter Notebooks. I was project leader for this project locally at Simula, and developed a new package called unray for efficient volume rendering of unstructured tetrahedral meshes using WebGL and Javascript. As part of this work we interacted with a number of externally driven open source projects, including ipywidgets, pythreejs, k3d-jupyter, scivijs, and ipyvolume.
Tools	Javascript ES6, Python3, WebGL, Three.js, Ipywidgets, Backbone.js, Node.js, Webpack

Activity	Development of Jupyter Notebook diff and merge tools
Period	2015-09 – 2016-12
Role	Project leader and main developer
Staffing	3 developers
Volume	20% – 60%
Description	As one deliverable of the larger Horizon 2020 project "OpenDreamKit" we developed the software package nbdime to handle diff and merge operations on Jupyter Notebook documents. Jupyter Notebook is a framework for literate programming with source code interleaved with cells text, mathematics, and visual material, including material computed by running the code cells. The notebook document is represented in a JSON format with binary parts stored as base64 encoded strings. Regular diff tools produce unintuitive output and regular merge tools can produce a broken document for this type of file. Our tool produces an intuitive output and handles non-text output in intelligent ways. I was project leader for this project and developed most of the core diff and merge algorithms in Python.
Tools	Jupyter, git, Python, Javascript, Typescript, pip, Node.js

Activity	Reworking tools for just in time compilation of shared libraries from C++ code
Period	2015-09 – 2016-10
Role	Lead developer
Staffing	1
Description	Some times less is more. Running the test suite of DOLFIN (a C++/Python library in the FEniCS suite) took around 1-2 hours, and a large fraction of this was due to 1000 just in time compiled shared libraries being wrapped and compiled using SWIG via the Python module Instant. Identifying that most of the time this was doing unnecessary work, I developed a much simpler Python module named dijitso which still compiles shared libraries from C++ code (on Linux, OSX and Windows) but only imports a factory function with a pure C signature using the builtin Python module ctypes.
Tools	Python, ctypes, g++

Activity	Modernization of Python and C++ code in FEniCS

Period	2013 – 2016
Role	Main developer
Staffing	4 developers
Description	As one of the core developers of the FEniCS project, I took responsibility for optimizing Python and C++ code, improving test and build systems, and modernizing source code to use Python3 and C++11. I took a leading role on determining which parts of C++11 was most valuable to us as well as feasible to start using at the time. Most of the Python3 work was carried out by a summer intern under my supervision and myself. Maintaining performance of the symbolic computations under Python versions 2 and 3 was critical since it represents the main serial overhead of the otherwise highly scalable parallel high performance computing framework that FEniCS is.
Tools	Python2, Python3, six, py.test, C++11, CMake

Activity	Replacing the code generation tools in FEniCS with scalable algorithms
Period	2012-03 – 2017-03
Role	Lead developer
Staffing	1 developer
Description	The FEniCS Form Compiler (FFC) had performance problems when compiling equations with large amounts of tensor terms, sometimes taking hours to compile. I developed a software package called uflacs which employs O(N) value numbering and factorization algorithms to make the code generation process scalable. This project was highly successful, enabling more complex scientific applications of FEniCS. Uflacs is now merged into FFC and has replaced the legacy code generation backends.
Tools	Python, C++

Activity	C++ development on a high performance parallel finite element library
Period	2008-03 – 2017-04
Role	Main developer
Staffing	5-15 developers
Description	DOLFIN is the main C++ library of the FEniCS suite of tools for implementing solvers of partial differential equations. I've had a central role in the development and maintenance of the package DOLFIN, in particular in questions relating to co-designing the algorithms, datastructures, and interfaces in C++ with the higher level Python interface and the corresponding low level just-in-time generated C++ kernels.
Tools	C++, CMake, SWIG, Python

Activity	A postprocessing framework for timestepping simulations using workflow graphs
Period	2014-09 – 2016-08
Role	Main developer and supervisor

Staffing	2 developers
Description	Simulations that output a sequence of timesteps with high temporal resolution may only need to or be able to store results for a subset of the timesteps. Sometimes the data a scientist wants to view and analyse is not the direct output of the simulation but some quantity derived from the output. Sometimes the nature of the interesting derived quantities is know a priori, other times new ideas occur during analysis and data must be read back for recomputation. In the cbcpost Python module we developed a framework to steer such postprocessing tasks based on declarative workflow graphs with timestepping awareness.
Tools	Python, C++, FEniCS

Activity	Implementing PDE constrained optimization software
Period	2011-04 – 2013-07
Role	Main developer
Staffing	3 developers
Description	Developed Python scripts for solving PDE constrained optimization problems to perform data assimilation for blood flow simulations. My work focused on the Stokes problem, ironing out issues and bottlenecks in the simulation software to lay the groundwork for later extensions to Navier-Stokes with coworkers.
Tools	Python, FEniCS, dolfin-adjoint

Activity	Portable embedded C development of audio processing software for video conference tools
Period	2009-08 – 2011-03
Role	Senior Developer
Staffing	10 developers in team
Volume	100%
Description	In TANDBERG I worked with low level C programming for audio processing on multiple platforms ranging from the primary low power TI DSP platform to PC platforms. One of my primary tasks was porting memory systems to new platforms, involving low level asynchrous RTOS programming and explicit control of memory hierarchies. A related task was the general preparation of the software for improved cross platform compatibility, requiring an understanding of the low level nuances that differ between hardware platforms.
Tools	C (C89), TI C674x DSP, Texas Instruments RTOS

Activity	Design of a domain specific language for tensor algebra and partial differential equations
Period	2007-10 – 2009-06
Role	Lead developer
Staffing	2 developers

Description	As the most significant contribution during my PhD work I designed and implemented a software package UFL - the Unified Form Language. This module implements a domain specific language embedded in the Python language in the form of a symbolic computing engine with domain specific abstractions. In particular the language abstractions and surrounding toolchain allows expressing and analysing variational formulations of partial differential equations using tensor algebra, Einstein notation, and differential operators. UFL has since been used outside of its original FEniCS project context by several independent research groups. The period and staffing above is for the initial development. About 25 developers have since contributed to the project under my supervision, and I have later extended and greatly optimized much of the implementation myself.
Tools	Python

Activity	Development of a low level C++ API interface for code generators
Period	2006-10 – 2007-06
Role	Lead developer
Staffing	5 developers
Description	As a young PhD student I took a leading role in development of low level C++ APIs for interfacing several code generator projects with library code in a unified way, allowing us to refactor separate projects to be interoperable. The interface was called UFC and is still at the core of the FEniCS libraries a decade later with core design principles intact.
Tools	C++

Activity	Development of symbolic computing tools for finite element methods
Period	2006-09 – 2009-06
Role	Main developer
Staffing	2 developers
Description	At Simula we developed tools for computing basis functions of finite elements symbolically, and generating C++ code from a symbolically integrated expression to compute the dense element matrix kernel efficiently. This was published as two software packages SyFi and SFC. I was the main developer of the code generation tools in particular.
Tools	C++, Python, SWIG, GiNaC

Activity	Multigrid mesh generation software for pipe flow problems
Period	2005-06 – 2005-08
Role	Main developer
Staffing	2 developers
Volume	100%
Description	As a pair of summer interns we developed 3D mesh generation software for pipe flow problems using Fortran95. The software took as input a splines representation of pipe centerlines together with radii and a graph of pipe connections, and produced a nested multigrid hexahedral mesh in the custom format of the Fortran77 library FEATFLOW.

Tools	Fortran77, Fortran95, FEATFLOW

Activity	Freelance C++ developer on music production software
Period	2002 – 2004
Role	Developer
Staffing	3 developers
Volume	20%
Description	For the shareware music production software Renoise, I worked on C++ development with a small team of part time developers, as a side job during university studies. The work involved audio, GUI, optimization, general C++ development, as well as engaging with the user community on feature discussions.
Tools	C++, Visual Studio, DirectX, VST Audio Plugins