Project Course in Applied Physics, CDIO

Computational Physics Project

Implementation and execution of computational physics computer simulations using industry-relevant project models and software engineering metodologies.

The project at a glance

• Collaboratively engineer software for molecular dynamics simulations, run it in high-throughput on the NSC supercomputers to generate big data, analyze it, visualize it, and make the results available to the world in a database accessible by an open API.

Skills

- Evolve your skills from programming→ software development→ ***software engineering.***
- Get experience working in an industry-relevant ***agile project model.***
- Work on your software ***portfolio*** for showing prospective employers.
- Lectures will cover the ***relevant physics***, background theory, analysis, ***visualization,*** + a wide range of ***software-related topics:***
  (e.g., version control, databases, visualization, test-driven development / continuous integration, optimization / profiling, security aspects, and advanced programming topics: patterns / paradigms / concurrency.)
- Train your ***presentation skills*** in written and oral presentations.

Four hands-on-sessions

- Version control with git, collaborative development with GitHub or GitLab.
- Visualization and exploratory data analysis; and source code documentation.
- Molecular dynamics with ASE and ASAP; and software testing.
- Running high-throughput computations on NSC supercomputers.

Contact & Info: **Rickard Armiento**, rickard.armiento@liu.se, Room G307

Project Course in Applied Physics, CDIO

Computational Physics Project

Course Outline

Lecture 1: *Course Introduction and Project Models*
- Overview, background, course plan, CDIO, waterfall vs. agile project models, LIPs, Scrum.

Lecture 2: *Software Versioning and Collaborative Development*
- Version control systems (git, svn), commits/branching/merging, collaborative workflows with pull requests and reviews (GitHub).

Hands-on exercise 1: *Git and GitHub*
- Working with a local repository: commits, branches, merging.
- Collaborative software development with pull-requests, review, and approvals.
- Creating the shared online repository for your project.

Lecture 3: *Exploratory Data Analysis by Visualization and Introduction to Computer Simulations*
- Single and multi-property exploration, identifying outliers, descriptors, heatmaps, PCA.
- Introduction to materials simulations in computational physics.
- Implementation considerations: representations of periodic structures, boundary conditions.

Lecture 4: *Software Documentation and Licensing*
- Documentation: UML, Source code comments, Embedded documentation, Sphinx.
- Software licensing: Open and closed source licenses (GPL, MIT, BSD, CC, etc.), CLAs.

Lecture 5: *Software Engineering in Industry (guest lecture)*

Hands-on exercise 2: *Exploratory Data Analysis and Documentation*
- Visualization and data exploration in Python (matplotlib, and more.)
- Extracting inline software documentation with Sphinx.

Lecture 6: *Introduction to Computational Physics and Molecular dynamics*
- Theoretical modeling of solid-state properties.
- The anatomy of a molecular dynamics program: interaction potentials, integration of equations of motion.

Lecture 7: *Software Testing, Debugging, and Profiling*
- Unit/integration/system/acceptance tests, black/white box, (non-)functional, test-driven development, coverage, CI/CD.
- Debuggers, profiling tools, algorithmic complexity.

Lecture 8: *Molecular Dynamics (cont.)*
- Calculating instantaneous properties, timesteps, thermalization.
- More advanced interaction potentials.
- Time and ensemble averages; pressure, heat capacity, MSD, Lindemann criterion, self-diffusion coefficient.
- Finding the equilibrium structure.

Hands-on exercise 3: *Molecular dynamics and software testing*
- Molecular dynamics with ASE and ASAP.
- Unit tests and continuous integration with GitHub actions.

Lecture 9: *Concurrency and Parallelism*
- Concurrency with coroutines; parallel threads (OpenMP), processes (MPI)
- Supercomputers.

Hands-on excerise 4: *Supercomputing*
- Supercomputer usage, queue scripts, high-throughput computations, etc.
- Running ASAP-simulations on supercomputers.

Lecture 10: *Databases, Wrap-up*
- Relational databases, normalization, transactions/ACID, SQL; noSQL, mongoDB.
- Making data available via open APIs.
- Final remarks about the project execution and final phases.

Project work: final presentation at end of december, final report deadline at the end of term (January).

Contact & Info: **Rickard Armiento**, rickard.armiento@liu.se, Room G307