Herramientas Computacionales

William Oquendo

1 Descripción

La asignatura herramientas computacionales busca formar al estudiante en los conceptos básicos de la física computacional como complemento y herramienta fundamental a la formación que debe tener el físico independiente de si su orientación es más teórica o experimental, ya que la herramienta computacional subyace y permea el estudio moderno de la física. Los métodos de la física computacional brindan solución donde la aproximación analítica se torna muy compleja, o complementan y refuerzan el resultado experimental, donde este existe. En esta asignatura el estudiante recibirá formación en los métodos básicos de la física computacional usando un lenguaje de alto nivel, aplicando librerias numéricas compactas y eficientes que permiten abordar los diferentes métodos numéricos y técnicas computacionales. Con estas herramientas, el estudiante será capaz de proponer soluciones a problemas básicos de tipo académico, investigativo y aplicado. Así mismo podrá decidir qué método podría ser el más apropiado para resolver un problema específico de la física. Finalmente, será capaz de crear visualizaciones que presenten los datos de forma compacta, eficiente y clara, y que permitan extraer de allí informaición

2 Objetivos

Al finalizar este curso, el estudiante será capaz de:

- Seleccionar el método computacional apropiado para resolver problemas aplicados de la física computacional en contextos como las ecuaciones diferenciales, matrices y vectores, dinámica molecular, etc.
- Reconocer los límites de aplicabilidad y de desempeño de los diversos métodos computacionales.
- Usar librerías propias o públicas que le permitan resolver problemas de la física computacional de forma rápida y correcta, de acuerdo a la escala del problema.
- Resolver problemas matriciales aprovechando al máximo las capacidades computacionales del hardware donde se ejecutan estas simulaciones.
- Analizar datos obtenidos de las simulacions de forma gráfica usando tanto la representación de los datos para buscar relaciones entre sí, como a partir de visualizaciones que permitan comprender mejor los sistemas bajo análisis, usando herramientas como Paraview, Python, Gnuplot, etc.

3 Contenidos Generales

1. Introducción:
   - Lenguajes de programación. Uso de la consola (shell). Comandos básicos de Linux y de git.
   - Entornos de desarollo en Python: ipython, notebook, online notebooks. Instalación de programas usando Anaconda.
   - Introducción/repaso de Python
2. Aritmética de punto flotante. Errores y estabilidad.
3. Cálculo numérico básico con scipy: Integración y derivación numéricas. Ceros de funciones.
4. Matrices y vectores:
   - Numpy. Arreglos básicos.
   - Sistemas matriciales $Ax = b$ con scipy
   - Problemas de valores propios con scipy
5. Ecuaciones diferenciales ordinarias de valor inicial (IVP) y de frontera (BVP)
   - Algoritmos básicos
   - Soluciones a problemas dinámicos de valor inicial. Animaciones con vpython.
   - Problemas de valor de frontera por diferencias finitas y por shooting.
6. Ecuaciones diferenciales parciales
   - Diferencias finitas.
   - Solución a la ecuación de Laplace, método de relajación.
   - Uso de librerias de elementos finitos como Fenics.
7. Procesos aleatorios
   - Números aleatorios y transformación entre distribuciones. Generación de números con distribuciones no estándar.
   - Integración de MonteCarlo.

- Método de Montecarlo, Markov-Chain. Ising Model

8. Introducción a la Dinámica Molecular
   - Algoritmos básicos de integración: Euler, Verlet, LeapFrog.
   - Simulaciones directas y con paquetes como lammps/liggghts.
   - Métodos de optimización como listas de celdas y de verlet.
   - Visualización con Paraview.
9. Análisis de Fourier
   - Transformadas de Fourier aplicada a señales como series de tiempo.
   - Análisis de imágenes y filtros.
10. Opcional: Introducción a pandas para el análisis de datos.

4 Metodología de evaluación

La nota se dividirá en cuatro ítems :

Trabajo independiente en clase o fuera de clase Se usan tareas de envío de código por teams o github, workshops de moodle (Martes a Viernes), etc.

Un examen tipo parcial Se evalúa lo aprendido en las primeras semanas.

Un proyecto intermedio Se trabaja en grupo y se entrega un mes después del examen parcial. Se entrega código e informe tipo artículo.

Un proyecto final se sustenta al final del semestre

El tema del proyecto final vendrá de una propuesta pre-aprobada, ya sea enviada por el profesor o enviada por el grupo con el debido visto bueno del profesor. El código fuente debe ejecutarse exitosamente y producir la solución correcta, de otra manera no será válida la entrega. Así mismo, se revisará que el repositorio tenga solamente los archivos apropiados. En los trabajos en grupo se evaluará la contribución de todos al trabajo final.

Item	Porcentaje	Fecha
Examen Parcial 1	20	Semana 4-5
Exámenes intermedios - tareas	35	-
Proyecto Intermedio	20	Semana 8-9
Proyecto Final	25	Última semana

- Horario de atención : Miércoles 9-11, cita previa por email para confirmar disponibilidad. También pueden escribir las dudas por email en cualquier momento o pedir sesión extra en otro horario.

5 Complementary tools

5.1 Herramientas complementarias de aprendizaje de c++

5.1.1 Cursos virtuales de C++

En la web encuentra muchos cursos, pero asegurese de usar un curso que utilice la versión moderna del estándar (C++11 o C++14). Puede seguir, por ejemplo,

- `https://learnxinyminutes.com/docs/c++/`
- `http://www.learncpp.com/`
- `https://www.codesdope.com/cpp-introduction/`
- `http://en.cppreference.com/w/`
- `https://www.edx.org/course/programming-basics-iitbo`
- `https://www.udemy.com/curso-de-cpp-basico-a-avanzad`
- `https://developers.google.com/edu/c++/`
- Channel 9
- `https://cestlaz.github.io/stories/emacs/`
- Curso Microsoft C++
- Jamie King
- Modern C++ - Bo Quian

En cuanto a libros, revise la bibliografía más abajo.

5.1.2 Ejercicios de programación

- `http://codeforces.com/problemset`
- `https://projecteuler.net/`
- `https://projectlovelace.net/`
- `https://exercism.io`
- `http://www.codeabbey.com/`

5.1.3 Shell/bash:

- `https://ubuntu.com/tutorials/command-line-for-begin 1-overview`
- `https://www.youtube.com/watch?v=Z56Jmr9Z34Q& list=PLyzOVJj3bHQuloKGG59rS43e29ro7I57J&index= 1`
- `https://www.youtube.com/watch?v=kgII-YWo3Zw& list=PLyzOVJj3bHQuloKGG59rS43e29ro7I57J&index= 2`
- `https://linuxcommand.org/lc3_learning_the_ shell.php`

5.1.4 Git practice

- https://gitexercises.fracz.com/
- https://www.youtube.com/watch?v=2sjqTHE0zok&list=PLyzOVJj3bHQuloKGG59rS43e29ro7I57J&index=6
- https://learngitbranching.js.org/

5.1.5 Python tutor : visualization of c++ code

- http://www.pythontutor.com/cpp.html#mode=edit

5.1.6 Online c++ compilers

- https://www.jdoodle.com/online-compiler-c++
- http://cpp.sh/
- https://www.codechef.com/ide

5.1.7 Data structure visualizations

https://www.cs.usfca.edu/~galles/visualization/Algorithms.html

5.2 Emacs Resources

- https://www.youtube.com/playlist?list=PL9KxKa8NpFxIcNQa9js7dQQIHc81b0-Xg
- https://mickael.kerjean.me/2017/03/18/emacs-tutorial-using-c-episode-0/
- Configuration builder : http://emacs-bootstrap.com/
- http://cestlaz.github.io/stories/emacs/
- http://tuhdo.github.io/c-ide.html
- http://emacsrocks.com
- http://emacs.sexy/
- https://github.com/emacs-tw/awesome-emacs
- http://spacemacs.org

6 Bibliografía

References

[1] Cristian C. Bordeianu Rubin H. Landau, Manuel J Paez. *Computational Physics: Problem Solving with Python*. Wiley-VCH, 3 edition, 2015.

[2] Rubin H Landau, Manuel J Páez Mejía, and William H Press. Computational physics: Problem solving with computers. *Physics Today*, 51:71, 1998.

[3] H Rubin and Author Landau. *Computational Physics: Problem Solving with Computers*. J. Wiley & sons, 1997.

[4] Simon Sirca and Martin Horvat. *Computational Methods for Physicists*. Springer Berlin Heidelberg, 2012.

[5] Simon Širca and Martin Horvat. *Computational Methods for Physicists: Compendium for Students*. Springer, 2012.

[6] Georg Hager and Gerhard Wellein. *Introduction to high performance computing for scientists and engineers*. CRC Press, 2010.

[7] Joe Pitt-Francis and Jonathan Whiteley. *Guide to scientific computing in C++*. Springer, 2017.

[8] Einar Smith. *Introduction to the Tools of Scientific Computing*, volume 25. Springer International Publishing, 2020.

[9] Anthony Scopatz and Kathryn D Huff. *Effective computation in physics: Field guide to research with python*. " O'Reilly Media, Inc.", 2015.

[10] Koenig Andrew. *Accelerated C++: practical programming by example*. Pearson Education India, 2000.

[11] B. Stroustrup. *Programming: Principles and Practice using C++*. Pearson Education, 2014.

[12] CH Swaroop. A byte of python. *Enllaç web*, 2003.

[13] John R Hubbard. *Schaum's Outline of Programming with C++*. McGraw-Hill Professional, 2002.

[14] William H Press. *Numerical recipes 3rd edition: The art of scientific computing*. Cambridge university press, 2007.

[15] Richard Fitzpatrick. Computational physics. *Lecture notes, University of Texas at Austin*, 2006.

[16] Tao Pang. *An introduction to computational physics*. Cambridge university press, 2006.

[17] Harvey Gould, Jan Tobochnik, and Wolfgang Christian. *An introduction to computer simulation methods*, volume 1. Addison-Wesley New York, 1988.

[18] Nicholas J Giordano. *Computational physics*. Prentice Hall New Jersey, 1997.

[19] Joe Pitt-Francis and Jonathan Whiteley. *Guide to scientific computing in C++*. Springer Science & Business Media, 2012.

[20] Hans Petter Langtangen. *A Primer on Scientific Programming with Python*. Springer Berlin Heidelberg, 2012.

[21] Suely Oliveira and David E Stewart. *Writing Scientific Software: a guide to good style*. Cambridge University Press, 2006.

[22] Cyrille Rossant. *Learning IPython for interactive computing and data visualization*. Packt Publishing Ltd, 2015.

[23] Gaël Varoquaux, Emmanuelle Gouillart, Olav Vahtras, Valentin Haenel, Nicolas P Rougier, Ralf Gommers, Fabian Pedregosa, Zbigniew Jędrzejewski-Szmek, Pauli Virtanen, Christophe Combelles, et al. Scipy lecture notes. 2015.

[24] Herman J. C. Berendsen. *Simulating the Physical World: Hierarchical Modeling from Quantum Mechanics to Fluid Dynamics*. Cambridge University Press, 1 edition, 2007.

[25] N.J. Giordano and H. Nakanishi. *Computational Physics*. Pearson/Prentice Hall, 2006.

[26] H.P. Langtangen. *A Primer on Scientific Programming with Python*. Texts in Computational Science and Engineering. Springer Berlin Heidelberg, 2011.

[27] Jaan Kiusalaas. *Numerical methods in engineering with Python 3*. Cambridge university press, 2013.

[28] A.L. Garcia. *Numerical Methods for Physics*. Prentice Hall, 2000.

[29] Philip Nelson Jesse M. Kinder. *A Student's Guide to Python for Physical Modeling*. Princeton University Press, 2015.

[30] B. Einarsson. *Accuracy and Reliability in Scientific Computing*. SIAM e-books. Society for Industrial and Applied Mathematics (SIAM, 3600 Market Street, Floor 6, Philadelphia, PA 19104), 2005.

[31] F.J.B. Silva. *Learning SciPy for Numerical and Scientific Computing*. Community experience distilled. Packt Publishing, Limited, 2013.

[32] Robert Johansson. *Numerical Python. A Practical Techniques Approach for Industry*. Apress, 2015.

[33] John V. Guttag. *Introduction to Computation and Programming Using Python, Revised & Expanded*. The MIT Press, revised and expanded edition edition, 2013.