Universidad de San Carlos de Guatemala
Escuela de Ciencias Físicas y Matemáticas
Programas de Licenciatura

Programa de Física Computacional

1. Descripción del Curso

Nombre: Física computacional
Código: F811
Prerrequisitos: F605 – F705
Créditos: 5
Profesor: Giovanni Ramírez García
Semestre: Primero, 2021

Las computadoras se han convertido en una parte fundamental de la física moderna. Tradicionalmente, la física se dividía en dos grandes ramas: la física experimental y la física teórica. Sin embargo una nueva rama, la física computacional, ha venido ganando mucha importancia ya que combina métodos de ambas partes. Además, la simulación computacional de sistemas físicos ayuda a desarrollar y validar modelos, y permite investigar sus propiedades haciendo uso de métodos numéricos. Por otro lado, con el aumento del poder computacional, la disponibilidad de sistemas de procesamiento en paralelo y de sistemas de alto rendimiento, resulta de utilidad conocer y manejar las distintas técnicas que ya se usan en la física computacional actualmente.

El curso inicia con una exposición de algunos de los lenguajes de programación y la programación de alto rendimiento. Posteriormente se divide en cuatro partes: interpolación y aproximación numéricas, integración numérica, diferenciación numérica y métodos Monte Carlo. Las cuatro partes se utilizarán para desarrollar ejemplos del análisis de las propiedades de sistemas físicos mediante métodos numéricos y computacionales. Si el tiempo y el grupo lo permite habrá una quinta parte donde se podrá estudiar el comportamiento de sistemas de autómatas celulares y de redes neuronales.

El lenguaje de programación que se utilizará es Fortran, un lenguaje de uso general pero especialmente dedicado a computación numérica y computación científica. Se usará la versión Fortran 95 porque ya incluye la especificación de *High Performance Fortran*. Se usará el compilador GNU puesto que es un *software* libre y de código abierto y se usará la versión que esté disponible en el clúster de la Escuela: euclides.

2. Competencias

2.1. Competencias generales

2.1.1 Plantear, analizar y resolver problemas físicos, tanto teóricos como experimentales, mediante la utilización de métodos analíticos, experimentales o numéricos.

2.1.2 Utilizar o elaborar programas o sistemas de computación para el procesamiento de información, cálculo numérico, simulación de procesos físicos o control de experimentos.

2.1.3 Demostrar una comprensión profunda de los conceptos y principios fundamentales, tanto de la física clásica como de la física moderna.

2.1.4 Demostrar hábitos de trabajo necesarios para el desarrollo de la profesión tales como el trabajo en equipo, el rigor científico, el auto- aprendizaje y la persistencia.

2.2. Competencias específicas

a) Identificar y resolver problemas que pueden resolverse mediante análisis numérico y simulaciones con computadoras.

b) Calcular la complejidad algorítmica de soluciones numéricas a problemas físicos.

c) Conocer los lenguajes de programación científica, especialmente FORTRAN.

d) Conocer y utilizar las soluciones de computación de alto rendimiento.

3. Unidades

3.1. Introducción

Descripción: Lenguajes de programación científica. Programación de Alto Rendimiento. Computabilidad y Complejidad. Clústers de computación

Duración: 10 períodos de 50 minutos

Metodología: Los períodos de clase son mayoritariamente magistrales, con tiempo dedicado a la solución de tareas cortas y tiempo para que el grupo de estudiantes pueda demostrar su aprendizaje y comprensión del tema mediante la exposición oral de los ejercicios y temas propuestos.

Evaluación: Se evaluará por medio de tareas cortas y una exposición.

3.2. Métodos numéricos de interpolación y aproximación

Descripción: Interpolación de Lagrange. Interpolación por Diferencias Divididas. Interpolación de Hermite. Interpolación Spline.

Duración: 10 períodos de 50 minutos

Metodología: Los períodos de clase son mayoritariamente magistrales, con tiempo dedicado a la solución de ejercicios guía y tiempo para que el grupo de estudiantes pueda demostrar su aprendizaje y comprensión del tema mediante la resolución y exposición oral de los ejercicios propuestos.

Evaluación: Se evaluará por medio de tareas cortas y un proyecto.

3.3. Métodos numéricos de integración

Descripción: Integración de ecuaciones diferenciales ordinarias. Órbitas y trayectorias. Modelos matemáticos de epidemiología. Péndulo doble.

Duración: 10 períodos de 50 minutos

Metodología: Los períodos de clase son mayoritariamente magistrales, con tiempo dedicado a la solución de ejercicios guía y tiempo para que el grupo de estudiantes pueda demostrar su aprendizaje y comprensión del tema mediante la resolución y exposición oral de los ejercicios propuestos.

Evaluación: Se evaluará por medio de tareas cortas y un proyecto.

3.4. Métodos numéricos de diferenciación

Descripción: Ecuaciones diferenciales parciales. Ecuación de Poisson. Ecuación de Laplace. Condiciones de frontera de Dirichlet. Ecuación de Calor. Ecuación de onda.

Duración: 10 períodos de 50 minutos

Metodología: Los períodos de clase son mayoritariamente magistrales, con tiempo dedicado a la solución de ejercicios guía y tiempo para que el grupo de estudiantes pueda demostrar su aprendizaje y comprensión del tema mediante la resolución y exposición oral de los ejercicios propuestos.

Evaluación: Se evaluará por medio de tareas cortas y un proyecto.

3.5. Métodos Monte Carlo

Descripción: Métodos de Monte Carlo. Simulación de decaimiento radiactivo y de propiedades de transporte. El problema de Percolación. Caminatas aleatorias.

Duración: 10 períodos de 50 minutos

Metodología: Los períodos de clase son mayoritariamente magistrales, con tiempo dedicado a la solución de ejercicios guía y tiempo para que el grupo de estudiantes pueda demostrar su aprendizaje y comprensión del tema mediante la resolución y exposición oral de los ejercicios propuestos.

Evaluación: Se evaluará por medio del proyecto final.

3.6. Modelos para Sistemas Dinámicos y Sistemas Complejos

Descripción: Autómatas celulares. Redes neuronales.

Duración: 4 períodos de 50 minutos

Metodología: Los períodos de clase son mayoritariamente magistrales, con tiempo dedicado a la solución de ejercicios guía y tiempo para que el grupo de estudiantes pueda demostrar su aprendizaje y comprensión del tema mediante la resolución y exposición oral de los ejercicios propuestos.

Evaluación: Se evaluará por medio del proyecto final.

4. Evaluación del curso

Los porcentajes asignados a cada uno de los elementos de la evaluación están de acuerdo con el Reglamento General de Evaluación y Promoción del Estudiante de la Universidad de San Carlos de Guatemala. Es obligatorio cumplir con el 80 % de la asistencia al curso y la lista de asistencia se debe firmar antes de los 15 minutos de haber iniciado el periodo de clase.

El curso se evaluará mediante la presentación proyectos enfocados a las unidades 3.2, 3.3, 3.4 y 3.5. Durante el desarrollo de dichas unidades se definirán los detalles del proyecto y las fechas de entrega y exposición de los mismos. Las tareas y las exposiciones son de caracter obligatorio para tener derecho a presentar el Proyecto final. Se aceptan proyectos y tareas después de la fecha y hora convenida aplicando la tasa de decaimiento de nota dada por la función $100e^{-t/36}$, donde t es el tiempo de retraso en horas.

Tareas cortas y exposiciones	10	puntos
2 Proyectos	60	puntos
Proyecto final	30	puntos
Total	100	puntos

5. Bibliografía

Referencias comentadas

Para la introducción a los distintos lenguajes de programación se usarán los libros de Deitel y Deitel [1] para los lenguajes de C y C++, Langtangen [2] para Python, de Quarteroni, Saleri y Gervasio [3] para Octave. También se usarán las bibliotecas descritas en el libro *Numerical Recipes* de Press et al. [4]. Para aprender Fortran se pueden usar los libros de Brooks [5], de Hahn [6] y de Koelbel [7]. Para los detalles del compilador GNU para Fortran se puede consultar el manual [8].

Para la parte del curso relacionada a computación de alto rendimiento y computación en paralelo se usarán los libros de Fountain [9], de Sipser [10], de Golub y Ortega [11] y de Dowd y Severance [12]. Además, vamos a tomar varios de los trabajos compilados en los libros de Fountain [9] y de Goulb y Ortega [11] para construirnos una idea de las aplicaciones y de los algoritmos usados en computación en paralelo.

El tercer capítulo del libro de Atkinson y Han [13] será usado para la parte de análisis y teoría de aproximaciones, el libro cuenta con definiciones muy formales pero en las clases magistrales se discutirá lo necesario para entender los métodos presentados. Sin embargo el libro de Cohen [14] puede usarse como una guía más didáctica.

Una introducción al Método de Monte Carlo aparece en el primer capítulo del libro de Landau y Binder [15] de donde también se tomará una guía para las aplicaciones en mecánica estadística. Por otro lado, el tercer capítulo del libro de Binder y Heermann [16] será usado como guía práctica en la aplicación del Método Monte Carlo. En caso de ser necesario un refuerzo en las áreas de termodinámica y mecánica estadística puede usarse el segundo capítulo del libro de Landau y Binder [15].

Para la última unidad, en el caso de que podamos cubrir todos los contenidos obligatorios, vamos a usar dos libros introductorios: el de Rojas [17] y el de Gurney [18]. Hay dos libros más avanzados para estos temas: el de Bishop [19] y el de Hastie et al. [20].

Referencias

[1] H. Deitel and P. Deitel, *C++ How to Program*. Prentice Hall, 2003.

[2] H. Langtangen, *A Primer on Scientific Programming with Python*. Texts in Computational Science and Engineering, Springer Berlin Heidelberg, 2014.

[3] A. Quarteroni, F. Saleri, and P. Gervasio, *Scientific Computing with MATLAB and Octave*. Texts in Computational Science and Engineering, Springer Berlin Heidelberg, 2014.

[4] W. Press, S. Teukolsky, W. Vetterling, and B. P. Flannery, *Numerical Recipes in Fortran 90: The Art of Parallel Scientific Computing*. No. v. 2 in Fortran numerical recipes, Cambridge University Press, 1996.

[5] D. Brooks, *Problem Solving with Fortran 90: For Scientists and Engineers*. Undergraduate Texts in Computer Science, Springer New York, 1997.

[6] B. Hahn, *FORTRAN 90 for Scientists and Engineers*. Elsevier Science, 1993.

[7] C. Koelbel, D. Loveman, R. Schreiber, G. Steele-Jr, and M. Zosel, *The High Performance Fortran Handbook*. Scientific and engineering computation, MIT Press, 1997.

[8] The GNU Project (Free Software Foundation), *GNU Fortran*, 2017.

[9] T. Fountain, *Parallel Computing: Principles and Practice*. Cambridge University Press, 2006.

[10] M. Sipser, *Introduction to the Theory of Computation*. Thomson Course Technology, 2006.

[11] G. Golub and J. Ortega, *Scientific Computing: An Introduction with Parallel Computing*. Academic Press, Inc., 1993.

[12] K. Dowd and C. Severance, *High Performance Computing*. A Nutshell handbook, O'Reilly, 1998.

[13] K. Atkinson and W. Han, *Theoretical Numerical Analysis: A Functional Analysis Framework*. Texts in Applied Mathematics, Springer New York, 2001.

[14] H. Cohen, *Numerical Approximation Methods:* $\Pi \approx 355/113$. Springer, 2011.

[15] D. Landau and K. Binder, *A Guide to Monte Carlo Simulations in Statistical Physics*. Cambridge University Press, 2005.

[16] K. Binder and D. Heermann, *Monte Carlo Simulation in Statistical Physics: An Introduction*. Graduate Texts in Physics, Springer Berlin Heidelberg, 2010.

[17] R. Rojas, *Neural Networks: A Systematic Introduction*. Springer Berlin Heidelberg, 2013.

[18] K. Gurney, *An Introduction to Neural Networks*. CRC Press, 2018.

[19] C. Bishop, *Pattern Recognition and Machine Learning*. Information Science and Statistics, Springer New York, 2016.

[20] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Series in Statistics, Springer New York, 2013.

http://www.ecfm.usac.edu.gt/programas