FACULTAD: Ciencias Básicas

PROGRAMA: Física

DEPARTAMENTO DE: Física y Geología

CURSO:	Física Computacional 1	**CÓDIGO:**	157218
ÁREA:	Profundización		
REQUISITOS:	R-167394	**CORREQUISITO:**	
CRÉDITOS:	3	**TIPO DE CURSO**:	Teórico - Práctico
FECHA ÚLTIMA ACTUALIZACIÓN	JULIO 2020		

JUSTIFICACIÓN

Las herramientas numéricas y computacionales se han convertido en elementos básicos para el quehacer y desarrollo de la Física actual. En este sentido es imperiosos que nuestras y nuestros egresados desarrollen competencias relacionadas al pensamiento computación, esto es que les permitan desarrollar códigos y simulaciones computacionales que den respuesta a problemas físicos.

Un aspecto clave en el desarrollo de las competencias computacionales es el dominio de un lenguaje de programación; sin este no es posible pasar del algoritmo a la resolución de problemas. Por su estilo sencillo y práctico, el lenguaje de programación Python se ha convertido en una herramienta útil, capaz y principalmente de rápido aprendizaje. Estas características permiten que las y los estudiantes dediquen más tiempo a entender la Física detrás de sus códigos y a analizar de manera critica los resultados que obtienen; competencias fundamentales en su desarrollo profesional.

En el contexto de la Física, una primera aproximación al pensamiento computacional consiste en resolver numéricamente problemas que involucren el cálculo de integrales. Este tipo de problemas puede implicar, resolver integrales muy complejas o sin solución analítica; o un número elevado de integrales, por ejemplo, en el caso de requerir estimar el campo eléctrico que produce una cierta distribución de carga en un cierto espacio.

La estimación de cantidades físicas a partir de datos medidos es parte fundamental del quehacer científico actual. Estimar, por ejemplo, razones de cambio, o dicho de forma general las derivadas a un volumen de datos permiten extraer información relevante para el análisis físico de un cierto fenómeno o sistema. Estimaciones que no serían posibles sin herramientas y algoritmos computacionales.

OBJETIVO GENERAL

Aplicar soluciones numéricas para resolver problemas físicos que involucren Integrales y derivadas, usando como lenguaje de programación Python.

OBJETIVOS ESPECÍFICOS

- ✓ Aprender los elementos básicos de la programación en Python.
- ✓ Aprender los conceptos básicos de Git como herramienta para el desarrollo y manejo de versiones de códigos computacionales; vía la plataforma Web GitHub.
- ✓ Aprender a calcular errores derivados de operaciones numéricas.
- ✓ Implementar algoritmos numéricos en Python que permitan realizar integrales funciones.
- ✓ Implementar algoritmos numéricos en Python que permitan realizar derivadas de funciones.

COMPETENCIAS

- ✓ Implementar algorítmos numéricos en el lenguaje de programación Python.
- ✓ Desarrollar códigos usando el control de versiones Git.
- ✓ Escribir códigos en Python que permitan solucionar problemas de la Física que involucren integración de funciones.
- ✓ Escribir códigos en Python que permitan solucionar problemas de la Física que involucren derivación de funciones y datos.
- ✓ Interpretar y analizar los resultados obtenidos a través de códigos computacionales.

Unidad 1: Introducción a Python

TEMA	HORAS DE CONTACTO DIRECTO	HORAS DE TRABAJO INDEPENDIENTE DEL ESTUDIANTE
Introducción y herramientas básicas ✓ ¿Por qué Física computacional? ✓ Notebooks (Google Colaboratory) ✓ Introducción a Git y GitHub ✓ Sincronización de Notebooks-Colab con GitHub	**7**	**4**
Elementos básicos de Python I: ✓ Variables y asignaciones ✓ Arítmetica ✓ Estamentos de entrada y salida	**7**	**4**
Elementos básicos de Python II: ✓ Estamentos de control ✓ Listas y arreglos ✓ Lazo *for* ✓ Funciones	**6**	**4**

Elementos básicos de Python II: ✓ Estamentos de control ✓ Listas y arreglos ✓ Lazo *for* ✓ *Funciones*	**6**	**4**
Gráficas y visualización en Python II: ✓ Graficas en 2D ✓ Graficas de dispersión ✓ Graficas de densidad ✓ Graficas en 3D	**6**	**4**

Unidad 2: Errores e integración numérica

TEMA	HORAS DE CONTACTO DIRECTO	HORAS DE TRABAJO INDEPENDIENTE DEL ESTUDIANTE
Errores numéricos ✓ Errores por redondeo ✓ Errores de truncamiento ✓ Series de Tylor	6	5
Integración numérica I ✓ Regla del trapecio ✓ Regla de Simpson ✓ Errores en la integración numérica	6	5
Integración numérica II ✓ Selección del número de pasos ✓ Integración de Romberg	6	5
Integración numérica III ✓ Cuadratura de Gauss	6	5
Integración numérica IV ✓ Integrales sobre intervalos infinitos ✓ Integrales múltiples	6	6

Unidad 3: Derivación numérica

TEMA	HORAS DE CONTACTO DIRECTO	HORAS DE TRABAJO INDEPENDIENTE DEL ESTUDIANTE
Derivación numérica I ✓ Diferencias finitas ✓ Diferencias centrales ✓ Erros en la derivación numérica	6	6
Derivación numérica II ✓ Segundas derivadas ✓ Derivadas parciales	6	6
Interpolación ✓ Interpolación lineal ✓ *Spline* cúbico	6	6

METODOLOGÍA

Se implementarán herramientas basadas en tecnologías de la información y la comunicación para el desarrollo de las clases. Para esto se habilitará todo el material del curso (lecturas, videos, etc) en las plataformas tipo MOOC dispuestas por la universidad para este fin. Como herramienta adicional, se hará uso de las plataformas web y *open-access* Github y Github-classroom.

Durante la dinámica del curso se destinará una hora semanal al desarrollo de un entorno SOLE (Self Organized Learning Environment), donde las y los estudiantes, divididos en grupos, desarrollarán actividades propuestas por ellos mismos o por el docente, relacionada con los temas vistos en la semana:

- 5 minutos para el planteo de la pregunta o actividad a realizar (docente o estudiante).
- 35 minutos para el desarrollo del tema planteado (grupos de estudiantes).
- 15 minutos para la exposición de las conclusiones de cada grupo (representante de cada grupo).
- 5 minutos para desarrollar las conclusiones y el sumario (docente).

Se invitarán expertos para que realicen una charla donde expongan los últimos desarrollos relacionados las temáticas del curso. De esta manera, los y las estudiantes tendrán una visión más amplia actualizada del estado del arte de los contenidos aquí expuestos.

SISTEMA DE EVALUACIÓN

Se realizarán 3 evaluaciones, según el calendario académico, las cuales corresponden al 60% de la nota definitiva, más las actividades propuestas por el profesor (quices, trabajos, etc) correspondiente al 40% de la nota definitiva restante.

- ✓ Primera evaluación del 35%: 20% corresponderá a un proyecto a entregar durante la respectiva semana de examen; y 15% otras actividades tales como talleres y quices.
- ✓ Segunda evaluación del 35%: 20% corresponderá a un proyecto a entregar durante la respectiva semana de examen; y 15% otras actividades tales como talleres y quices.
- ✓ Tercera evaluación del 30%: 20% corresponderá a un proyecto a entregar durante la respectiva semana de examen; y 10% otras actividades tales como talleres y quices.

BIBLIOGRAFÍA DISPONIBLE EN UNIDAD DE RECURSOS BIBLIOGRÁFICOS DE LA UNIVERSIDAD DE PAMPLONA

Chapra, Steven, C. Métodos numéricos para ingenieros. Mc Graw-Hill, 2007.

BIBLIOGRAFÍA COMPLEMENTARIA

- ✓ Newman, Mark. Computational Physics. Univeristy of Michigan, 2013.
- ✓ Landau, Rubin H. Computational Problems for Physics. CRC Press, 2018.
- ✓ Sherer, Philipp O. J., Computational Physics, Simulation of Classical and Quantum Systems. Springer, 2010.

DIRECCIONES ELECTRÓNICAS DE APOYO AL CURSO

- ✓ http://www.umich.edu/~mejn/cp
- ✓ http://sites.science.oregonstate.edu/~landaur/Books/CPbook/eBook/Lectures/
- ✓ https://www.python.org/
- ✓ https://git-scm.com/book/es/v2

NOTA: EN CADA UNA DE LAS UNIDADES EL DOCENTE DEBERA PROPONER MÍNIMO UNA LECTURA EN LENGUA INGLESA Y SU MECANISMO DE CONTROL