Ampliando el alcance de lo aprendido en electrostática usando Python

Palazzo, Edgardo a; Bettachini, Víctor A. b; Real, Mariano A. c
a- UTN Facultad Regional Avellaneda, Buenos Aires, Argentina
b- UNLaM, Buenos Aires, Argentina; IGN, Buenos Aires, Argentina
c- INCALIN, UNSaM, Buenos Aires, Argentina; INTI, Buenos Aires, Argentina
epalazzo@fra.utn.edu.ar

Área de trabajo relacionada:

Innovación y tecnologías educativas.

Modalidad:

Virtual.

¿Desea que la exposición sea transmitida en vivo durante el congreso?

Sí.

Resumen:

Las actividades propuestas, que se están desarrollando en un curso de Física 2 de UTN FRA, apuntan a asistir a estudiantes de un primer curso de electromagnetismo en la adquisición de conceptos básicos, en particular en la resolución y análisis de problemas de electrostática. Tras las primeras clases desarrolladas en algún formato tradicional, dedicadas a la temática de campo y potencial electrostático, se pasa a una etapa en que se hace uso de código escrito en Python.
En las primeras de estas clases con código se realizan los mismos cálculos simbólicos que en papel correspondientes a ejercicios de guías anteriores, para así familiarizarse con la nueva metodología. En posteriores clases se encaran con esta herramienta problemas que por la complejidad de la matemática asociada resultaría muy tedioso y propenso a errores el resolverlos en papel. Delegar los cálculos en la computadora amplía la variedad de situaciones que el alumno puede estudiar y hace que destine su tiempo más en su análisis que en tareas repetitivas y mundanas.
Entre los tópicos que la herramienta pone al alcance del alumno, inabordables en pizarrón o papel con sus conocimientos al momento de cursar esta asignatura, están los cálculos de integrales necesarios para la resolución de problemas con distribuciones de cargas contínuas. La facilidad para realizar cálculos simbólicos en Python permite que los estudiantes se concentren en analizar los resultados y no en resolver integrales que aún no dominan. El material se presenta mediante cuadernos Jupyter que se ejecutan en Google Colaboratory y no se requieren conocimientos previos de programación ni de cálculo numérico por parte de los estudiantes. El código está diseñado para ser sencillo de leer, y que los estudiantes puedan resolver problemas nuevos reutilizando código con pequeñas modificaciones.

Palabras clave: electromagnetismo, electrostática, Python, Jupyter, colab

Introducción:

Los cuadernos de Jupyter son documentos que se dividen en celdas independdientes que contienen código de programación, siendo Python uno de los lenguajes más usados en este contexto (Kluyver et al., 2016). Permiten un trabajo interactivo con el código, y junto con la posibilidad de intercalar celdas escritas en lenguaje Markdown (Gruber, 2024), que presenten material didáctico o explicativo, los convierten en excelentes herramientas para educación. Su utilización se está extendiendo y podemos encontrar numerosos cursos basados en esta herramienta, la mayoría de los cuales buscan introducir el uso de Python para cálculos numéricos en temas específicos, como por ejemplo la resolución de las ecuaciones de Navier-Stokes (Barba y Forsyth, 2018) o un curso de química basado en datos (Cumby et al., 2023). Un ejemplo local es el curso sobre mecánica racional para estudiantes de ingenía mecánica de la Universidad Nacional de La Matanza. Los autores colaboran en un proyecto de investigación de esa unidad académica, que busca volcar lo aprendido en los tres años que lleva esa experiencia a otros cursos en ingeniería. Las actividades propuestas en este trabajo están fuertemente inspiradas en dicho curso, que está íntegramente basado en código Python (Bettachini y Palazzo, 2022).
Estos cursos, como los ejemplos mencionados, normalmente tienen como un objetivo principal la adquisición de conocimientos y técnicas de programación. En cambio, podemos encontrar también cursos donde la comprensión completa del código no es el objetivo principal, como es el caso de las actividades de esta experiencia. En estos casos, el código es sólo una herramienta que asiste a una mejor adquisición de los conceptos de la materia impartida (Laureline Duvillard, s.f.; Caligaris et al., 2018).

Objetivos de la experiencia:

Objetivos para estudiantes:

- Repasar lo trabajado en clases previas sobre campo y potencial eléctrico con cargas puntuales, y al mismo tiempo validar que la computadora entrega los mismos resultados que los obtenidos en papel y en el pizarrón.
- Internalizar que las distribuciones contínuas de carga eléctrica se pueden analizar como formadas por numerosas cargas puntuales, al comprobar que se obtienen resultados muy aproximados a los resultados exactos que se obtienen usando fórmulas conocidas, como pueden ser la del campo eléctrico de esferas, líneas o planos cargados.
- Aprovechar las posibilidades de cálculo simbólico que provee la biblioteca SymPy (Meurer et al., 2017) de Python para obtener soluciones exactas a problemas que involucran integrales múltiples complicadas y herramientas de análisis que aún no manejan.
- Experimentar con problemas de creciente complejidad reutilizando el código que fueron adaptando a los ejercicios de aprendizaje propuestos.

Objetivos para docentes:

- Aprovechar que con el mismo código se pueden resolver problemas mucho más complejos, que antes no se tenían en cuenta para trabajar en pizarrón o para proponer como ejercitación. Esto permite generar distintos ejemplos y situaciones para indagar sobre la comprensión de los temas campo y potencial eléctrico.

- Delegar los cálculos de campos y potenciales eléctricos en presencia de distribuciones de cargas contínuas en la computadora, que involucran herramientas de análisis que los estudiantes de este nivel aún no manejan, y trabajar estos temas íntegramente en Python.

Desarrollo:

En la tabla 1 se presenta un resumen del contenido de los diferentes bloques o módulos que conforman esta experiencia. Cada módulo puede consistir en uno o más cuadernos Jupyter, y algún material extra como guías de ejercicios, y el tiempo de cada clase puede ser ocupado por el trabajo en uno o más módulos. Las actividades propuestas se insertan en un curso tradicional de introducción al electromagnetismo, luego de las primeras clases, tras un periodo en el cual se trabajó con problemas de campo y potencial eléctrico de cargas puntuales, con docentes que explicaron en pizarrón y estudiantes que resolvieron problemas en papel. ~~En ese momento del curso se pasa a una etapa de clases que~~ se desarrollan enteramente utilizando código escrito en lenguaje Python, que no requieren conocimientos previos de programación ni de cálculo numérico por parte de los estudiantes. Con código simple de utilizar, los estudiantes pueden resolver problemas nuevos realizando pequeñas modificaciones, que en la mayoría de los casos sólo implican cambiar valores de variables de la misma forma que proceden cuando aplican una fórmula conocida a un nuevo problema numérico en papel. El material se presenta mediante cuadernos Jupyter cargados en la plataforma Google Colaboratory, en la cual los estudiantes pueden trabajar desde cualquier dispositivo con un navegador de internet, con el único requisito de acceder a una conexión a internet, sin la necesidad de instalar programas específicos. Este material está disponible para ser utilizado, descargado y adaptado libremente en el repositorio https://github.com/frautn/F2-electromagnetismo.

Módulo	Contenido	Objetivos
1	Campo eléctrico de cargas puntuales	Repasar contenidos básicos y familiarizarse con la metodología de trabajo.
2	Potencial con cargas puntuales	Repasar contenidos básicos y familiarizarse con la metodología de trabajo.
3	Distribuciones de carga contínuas	Considerar a las distribuciones contínuas como formadas por numerosas cargas puntuales, y realizar los cálculos con las herramientas de los módulos 1 y 2.
4	Distribuciones de carga contínuas	Utilizar una librería de cálculo simbólico para calcular las integrales mútiples necesarias en estos casos.
5	Campos y equipotenciales en presencia de conductores	Observar en distintas geometrías y sistemas de cargas, como se ajustan las líneas de campo para mantener los potenciales en los conductores (condiciones de contorno) en situaciones de difícil resolución en papel.
6	Trabajo final	Resolver algún problema complejo nuevo.

Tabla 1: Desarrollo de los contenidos.

A continuación se presentan algunos detalles sobre los módulos mencionados en la tabla 1.

Módulos 1 y 2 - Campo y potencial eléctrico de cargas puntuales: Utilizando el código Python provisto por los docentes, los estudiantes comienzan estas actividades reproduciendo algunos

de los cálculos de los ejercicios trabajados en las clases anteriores, con el objetivo de repasar los conceptos básicos y familiarizarse con el método de trabajo. Los estudiantes avanzan con la ejecución del código y con los ejercicios propuestos, asistidos por los docentes y también por las celdas escritas en Markdown que contienen breves explicaciones, imágenes y comentarios, intercaladas entre bloques de código Python.

En poco tiempo se pasa a trabajar en ejercicios cuya resolución en papel está descartada, no por su dificultad, sino por el volumen de cálculos repetitivos que deberían realizar los estudiantes. ~~Pero también~~ se analizan las líneas de campo y las equipotenciales de distribuciones de cargas arbitrarias, como la mostrada en la figura 1, o incluso visualizaciones en 3 dimensiones, tareas imposibles de implementar si se trabajara en papel. En la misma figura se destaca ~~la simplicidad con la~~ que los estudiantes pueden generar dichos gráficos, mediante el llamado a una única función diseñada específicamente para este curso. A lo largo de toda la experiencia, los estudiantes solo necesitan modificar códigos ~~sencillos~~, como la definición de un campo eléctrico de cargas puntuales, muy similar a un cálculo realizado en papel. ~~En cambio, cualquier~~ código avanzado, como el necesario para generar líneas de campo, se provee en una librería propia del curso, para poner el foco en el análisis de los resultados y no en ~~la dificultad del código para~~ generarlos.

```python
# Lista de cargas:
#  5 nC en ( 0; 0; 0) m
# -3 nC en (-2; 0; 0) m
#  7 nC en ( 2; 3; 0) m

Q = [
    [5E-9,0,0,0],
    [-3E-9,-2,0,0],
    [7E-9,2,3,0],
]

plotEf(Q)
```

✓ 0.2s Python

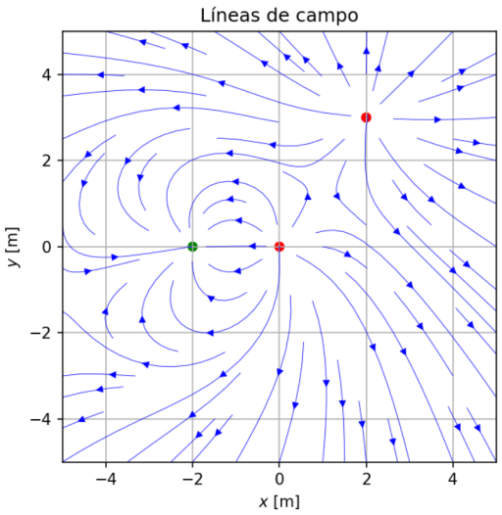

Figura 1: Código ~~simple~~ que genera líneas de campo.

Módulo 3 - Distribuciones de carga contínuas (parte 1): Se continúa replicando el mismo código de los módulos 1 y 2, para calcular campos y potenciales de distribuciones contínuas generadas como la suma de muchas cargas puntuales.

Módulo 4 - Distribuciones de carga contínuas (parte 2): Se obtienen soluciones exactas de configuraciones con distribuciones contínuas, utilizando la librería Sympy para el cálculo de las integrales y los gradientes. En la figura 2 se muestra a modo de ejemplo el cálculo del campo generado por un segmento de carga. Notar que se obtiene el resultado del campo en cualquier posición, no solo en los ejes de simetría como usualmente se busca cuando se trabaja en papel.

Módulo 5 - Campos y equipotenciales en presencia de conductores: La teoría involucrada para obtener equipotenciales cuando se tienen conductores con carga superficial no uniformes no forma parte de un curso de este nivel. Sin embargo, el análisis de los resultados sí

puede incluirse como tarea. En particular, en este curso en el cual se está llevando a cabo la experiencia, los estudiantes realizan una práctica de laboratorio donde mapean el voltaje en una cuba que contiene electrodos extensos mantenidos a potencial constante, para luego estimar las curvas equipotenciales y las líneas de campo correspondientes. Por lo tanto, los análisis que los estudiantes realicen en este módulo son muy similares a dicha tarea de laboratorio. El código necesario para generar los gráficos se desplaza del centro de atención, incluyendo el máximo posible en librerías, y los estudiantes sólo deben familiarizarse con la forma de presentar la geometría de los conductores.

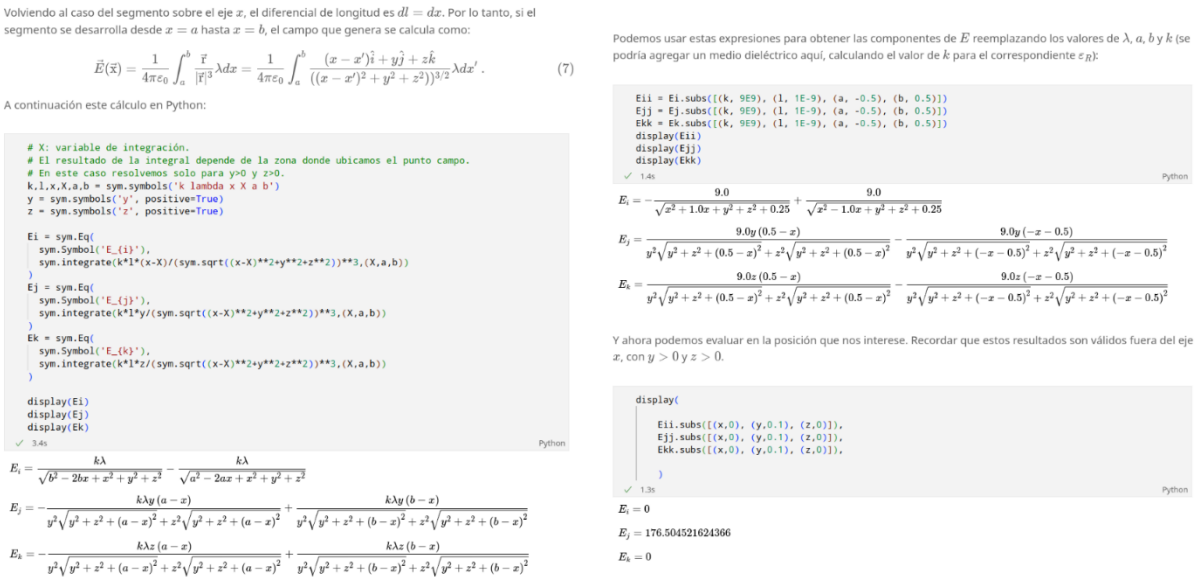

Figura 2: Cálculos simbólicos.

Módulo 6 - Trabajo final: Se pide la resolución de un problema que involucre más de una de las herramientas usadas anteriormente, y se provee un cuaderno Jupyter en forma de plantilla para facilitar el inicio del trabajo.

Evaluación de los resultados

Según las posibilidades que otorgue el cronograma y el desarrollo de las clases del curso, se preveen evaluaciones escritas para medir logros cognitivos y la transferencia exitosa de los conocimientos adquiridos a situaciones novedosas, no practicadas en el curso. Estas evaluaciones se realizarán en un formato tradicional para este nivel de estudiantes, es decir, en papel y sin la asistencia de una computadora, permitiendo la comparación de los logros con otros cursos en los cuales no se adopten las actividades propuestas por este trabajo.

Conclusiones:

Los estudiantes fueron capaces de resolver y analizar problemas con distribuciones de carga contínuas complejas que serían imposibles de abordar utilizando únicamente papel y lápiz. Gracias al uso de cuadernos Jupyter y código Python, lograron aproximar estas distribuciones utilizando cargas puntuales, facilitando así la comprensión de conceptos avanzados sin la necesidad de técnicas matemáticas más complejas que aún no dominan. Los docentes pudieron explicar las distribuciones contínuas de carga sin tener que recurrir a integrales múltiples, que

son difíciles de manejar para los estudiantes en esta etapa de su formación. Este enfoque permitió a los estudiantes visualizar y entender cómo se comportan las distribuciones de carga en situaciones más realistas, haciendo uso de herramientas computacionales que simplifican el proceso de aprendizaje. Se ha planificado la realización de mediciones de resultados de aprendizaje cuantitativas una vez que los estudiantes hayan completado la experiencia con los cuadernos Jupyter, y estas mediciones permitirán evaluar de manera objetiva el impacto de este enfoque en su comprensión y habilidades para resolver problemas de electrostática. Los resultados de estas mediciones serán fundamentales para validar la eficacia del método y considerar su implementación a largo plazo en el currículo del curso. En resumen, la implementación de cuadernos Jupyter en el curso ha permitido a los estudiantes abordar problemas complejos de manera efectiva, facilitando la enseñanza de conceptos difíciles sin necesidad de realizar cálculos matemáticos avanzados.

Bibliografía:

Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J., Grout, J., Corlay, S., Ivanov, P., Avila, D., Abdalla, S., & Willing, C. Jupyter-Notebooks – a publishing format for reproducible computational workflows (F. Loizides & B. Schmidt, Eds.). En: Positioning and Power in Academic Publishing: Players, Agents and Agendas (F. Loizides & B. Schmidt, Eds.). Ed. por Loizides, F., & Schmidt, B. IOS Press. 2016, 87-90.

Gruber, J. (2024, 8 de junio). Daring Fireball. https://daringfireball.net/projects/markdown/

Barba, L. A., & Forsyth, G. F. (2018). CFD python: The 12 steps to navier-stokes equations. Journal of Open Source Education, 2 (16), 21.

Cumby, J., Degiacomi, M., Erastova, V., Güven, J., Hobday, C., Mey, A., Pollak, H., & Szabla, R. (2023). Course Materials for an Introduction to Data-Driven Chemistry. Journal of Open Source Education, 6 (63), 192.

Bettachini, V. A., & Palazzo, E. (2022). Experiencia de un curso de mecánica racional basado en código. En L. F. L. et al (Ed.), Memorias del Encuentro Argentino y Latinoamericano de Ingeniería – 2021 (pp. 1039-1049, Vol. 3).

Laureline Duvillard. (s.f.). Using code to better understand the physics behind equations [EPFL]. Consultado el 8 de junio de 2024, desde https://www.epfl.ch/education/educational-initiatives/jupyter-notebooks-for-education/teachers-experience-with-jupyter-notebooks/using-code-to-better-understand-the-physics-behind-equations/

Caligaris, M., Rodríguez, G., Favieri, A., & Laugero, L. (2018). Desarrollo de habilidades matemáticas durante la resolución numérica de problemas de valor inicial usando recursos tecnológicos. Revista Educación en Ingeniería, 14, 30-40.

Meurer, A., Smith, C. P., Paprocki, M., Čertík, O., Kirpichev, S. B., Rocklin, M., Kumar, A., Ivanov, S., Moore, J. K., Singh, S., Rathnayake, T., Vig, S., Granger, B. E., Muller, R. P., Bonazzi, F., Gupta, H., Vats, S., Johansson, F., Pedregosa, F., . . . Scopatz, A. (2017). SymPy: symbolic computing in Python. PeerJ Computer Science, 3, e103.

Requerimientos técnicos e insumos:

No hay requerimientos.