Polinomios de Chebyshev y procesamiento de imágenes en Jupyter Notebooks

Objetivos

En esta actividad se van a realizar ajustes e interpolaciones en Python utilizando NumPy y los módulos interpolate y optimize de SciPy.

El ajuste de curvas (aproximación) es necesario para realizar el ajuste de los datos obtenidos de mediciones de acuerdo con el modelo.

La interpolación se utiliza para construir una curva que pase por un conjunto de puntos (ajuste exacto). Si los puntos son pocos se pueden encontrar polinomios interpoladores, y en caso contrario habrá que usar trazadores (*splines* en inglés).

Muchas de las conclusiones que se obtengan se podrán extraer de las visualizaciones que se obtengan de los datos, los polinomios o *splines* utilizados.

Descripción de la actividad.

Para el desarrollo de la actividad se pueden usar funciones de las bibliotecas mencionadas y, en el caso de no existir el método, se deberá implementar.

Tarea 1. Interpolación polinómica

La interpolación polinómica es uno de los métodos más usados para obtener las curvas que pasan por un conjunto de puntos. Sin embargo, existe un problema que se produce cuando el número de puntos crece bastante. En estos casos se produce lo que se conoce como el fenómeno de Runge, por el cual, el polinomio oscilará de manera considerable.

Para entender este efecto y ver una representación gráfica pueden consultar el [artículo](https://es.wikipedia.org/wiki/Fen%C3%B3meno_de_Runge) sobre el fenómeno de Runge en Wikipedia.

Esto hace que la elección de los nodos de interpolación, es decir, las x de los puntos, sea una tarea importante, porque el error del ajuste variará de acuerdo con esto.

Se trabajará con tres funciones distintas y, para ellas, se compararán tres métodos de interpolación polinómica y la interpolación con *splines* con nodos equiespaciados o con nodos obtenidos de los polinomios de Chebyshev.

Profundicen sobre la interpolación con la [presentación](https://personales.unican.es/segurajj/interp_p.pdf) de Javier Segura.

Funciones

Las funciones son:

* .

Consideraciones

* Número de nodos:
* 11 nodos y 21 nodos. Deben hacer los supuestos por separado.
* Obtención de los nodos:
* Equiespaciados.
* Raíces del polinomio de Chebyshev. Estos polinomios pertenecen a una familia de polinomios ortogonales. Se trata de obtener los polinomios hasta el grado correspondiente al número de puntos o nodos de interpolación que se quieren considerar, de forma que las raíces del polinomio de grado correspondiente serán los puntos de interpolación.

Para esto, se puede usar el módulo chebyshev de numpy.polinomial. Se deberá obtener el polinomio del grado correspondiente y sus raíces para obtener los nodos de interpolación.

Para obtener información sobre los polinomios de Chebyshev puede consultar este [enlace](http://www.sc.ehu.es/sbweb/fisica3/especial/chebyshev/chebyshev.html).

* Métodos de interpolación:
* Método de interpolación baricéntrica, mediante la función barycentric\_interpolate del módulo interpolated de scypy.

Esta función requiere la lista de las x e y de los puntos de interpolación y un *array* en el que evaluar el polinomio de interpolación.

* Método de interpolación de Lagrange mediante la función lagrange del módulo interpolated de ScyPy.
* Método de diferencias dividas de Newton. Se debe implementar en Python. La implementación debe ser explicada y justificada.

En este caso se podría usar la biblioteca sumpy para matemáticas simbólicas, que permite la definición de expresiones algebraicas, es decir, el polinomio de interpolación con la función lambdify. En este método se debe calcular una tabla que se construirá como se muestra en el ejemplo.

Dados tres puntos

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| i |  |  |  |  |
| 0 | 0 | -1 |  |  |
| 1 | 1 | -1 |  |  |
| 2 | 2 | 7 |  |  |

El polinomio interpolador es

Puede consultar más información en

<https://personales.unican.es/segurajj/tema3-2.pdf>

* Cálculo de los errores de los resultados obtenidos donde:

Siendo , la función que se va a aproximar y el polinomio obtenido.

Para el cálculo del error se pueden usar funciones de los módulos de algebra lineal de NumPy o SciPy.

* Cálculo de los tiempos de obtención del polinomio por cada método. Puede usar time de la biblioteca time.
* Para la obtención de las gráficas debe tener en cuanto los valores de las x para los que se hace la representación.

Se pide que para cada una de las funciones:

* Obtengan los puntos de la función en los nodos (11 y 21) de interpolación seleccionados. Expliquen cómo los construye para cada una de las funciones.
* Usen esos puntos para obtener el polinomio de interpolación según el método elegido. Expliquen los resultados obtenidos (polinomio) para cada función, método y tipo y número de nodos.
* Calculen el error de interpolación en cada caso.
* Calculen los tiempos de cálculo de los polinomios.
* Dibujen las gráficas de la función real y del polinomio obtenido. Pueden usar una misma figura para cada método o pueden incorporarlas todas en la misma. Indiquen las conclusiones que se obtienen del análisis de las gráficas.
* Muestren las conclusiones obtenidas indicando las diferencias encontradas, contrastando los valores obtenidos en los errores, en el tiempo de cálculo y el análisis de las gráficas.

Tarea 2. Interpolación con *splines*

Esta interpolación utiliza *splines*, que es una curva diferenciable definida en trozos mediante polinomios.

La ventaja de este método estriba en que se elimina el fenómeno de Runge o, los que es lo mismo, se eliminan las oscilaciones, ya que los polinomios que se utilizan son de un grado bajo.

Realicen la interpolación de las funciones anteriores usando *splines*. Para ellos pueden usar InterpolatedUnivariateSpline del módulo interpolated de ScyPy.

Debe repetir los mismos pasos realizados en la tarea anterior:

* Obtengan los puntos de la función en los nodos de interpolación seleccionados. Expliquen cómo los construye para cada una de las funciones.
* Usen esos puntos para obtener el polinomio de interpolación con *splines* cúbicos. Expliquen el resultado obtenido para cada función, tipo y número de nodos.
* Calculen el error de interpolación en cada caso.
* Calculen los tiempos de cálculo de los polinomios.
* Dibujen las gráficas de la función real y del polinomio obtenido.
* Muestren las conclusiones obtenidas indicando las diferencias encontradas, contrastando los valores obtenidos en los errores, en el tiempo de cálculo y el análisis de las gráficas.

Rúbrica

|  |  |  |  |
| --- | --- | --- | --- |
| Polinomios de Chebyshev y procesamiento de imágenes en Jupyter Notebooks | Descripción | Puntuación máxima  (10) | Peso  100% |
| Nodos de interpolación | * Se definen los nodos (11 y 21 nodos). * Se obtienen los valores de las funciones en esos nodos. * Se explica el procedimiento. | 2 | 20% |
| Cálculo de los polinomios de interpolación | * Método de interpolación baricéntrica. * Método de interpolación de Lagrange. * Método de diferencias dividas. * Interpolación por *splines.* * Se explica la aplicación de cada uno de los métodos. * Se muestra el polinomio de interpolación. | 3  (0,5 puntos cada método y 1 para diferencias divididas, resto explicación y visualización polinomio). | 30% |
| Error de los métodos y cálculo de los tiempos | * Se calcula el error absoluto en cada caso. * Se calcula el tiempo de cálculo del polinomio en cada caso. | 2 | 20% |
| Gráficas de la función y de los polinomios | * Se dibujan las gráficas de las funciones y sus polinomios interpoladores. * Se analizan las gráficas. | 2 | 20% |
| Conclusiones | * Se extraen las conclusiones a la vista de los datos de error, tiempo y análisis de las gráficas. | 1 | 10% |
|  |  | **10** | **100 %** |

Forma de entrega

Este trabajo deberá entregarse en formato editable (.doc, .docx, .odf, .rtf) o un notebook.

Extensión máxima de la actividad

La extensión máxima de la actividad es de diez páginas con fuente Georgia 11 e interlineado 1,5.

Se debe adjuntar el código de las implementaciones si no se entrega con un notebook.