4.1 – Producto de Convolución

* *Código:*

Texto

Descripción generada automáticamente

Bloque 4

Bloque 3

Bloque 2

Bloque 1

* *Breve explicación:*
  + Bloque 1:

Se importan las librerías necesarias para la operación del programa desarrollado, y se definen aliases con el objetivo de simplificar las llamadas a dichas librerías

* + Bloque 2:

En primera instancia, se define una función que genera un pulso rectangular de amplitud 1, que abarca el rango de -0.5 a 0.5 en el eje t, para simplificar la llamada, ya que esta está compuesta por una resta de dos escalones de Heaviside, lo cual la convierte en algo muy aparatoso como para llamarlo constantemente en el código.

En segunda instancia, se definen las variables necesarias para invocar a las funciones utilizadas, utilizando la variable temporal t como parámetro.

La función *f* es, como se ha expuesto anteriormente, un pulso rectangular de amplitud 1 comprendido entre -0.5 y 0.5 en el eje t.

La función *g* es una función exponencial con argumento -t+1, multiplicada por un escalón, tal que su valor sea 0 siempre que t sea menor a 0.

* + Bloque 3:

Utilizando la función *linspace* de la librería numpy (función que devuelve un array de números en el intervalo especificado, con una distancia entre pasos calculada con la cantidad de pasos que debe haber en el rango especificada), se define para cada función (*t* para *f*, *t1* para *g*) una línea de tiempo sobre la cual se va a muestrear dicha función (dado que las computadoras no son capaces de procesar intervalos contínuos, dada su naturaleza física).

Luego, se llama a la función *convolve*, de la librería numpy (que devuelve una convolución lineal discreta de un array, que en este caso son las muestras de nuestras funciones), y se escalan los valores de esa convolución por un diferencial de tiempo *dt* (espacio entre la muestra actual y la anterior), a su vez multiplicados por un pequeño factor de corrección necesario para que los valores coincidan con los de la verificación matemática manual.

* + Bloque 4:

Este bloque utiliza el módulo *pyplot* de la librería *matplotlib* para graficar tanto la convolución resultante como las funciones utilizadas para realizarla. Por una cuestión estilística, se decidió integrar todo en un mismo gráfico, haciendo que el eje izquierdo represente el eje vertical de las funciones *f* y *g*, mientras que el derecho represente el eje vertical de la convolución *y*.

* *Resultado de la ejecución:*

Gráfico, Histograma

Descripción generada automáticamente

* *Análisis matemático manual:*

Para:

Con:

Debemos separar la integral en tres intervalos.

* Para t < -0.5:
* Para -0.5 < t < 0.5:
* Para t > 0.5

Entonces:

Gráfico:

Gráfico, Gráfico de líneas

Descripción generada automáticamente

4.2 – Suma de Convolución

* *Código:*
  + *Versión 1 (gran número de muestras):*

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

Bloque 5

Bloque 4

Bloque 3

Bloque 2

Bloque 1

* + *Versión 2 (pequeña cantidad de muestras):* Interfaz de usuario gráfica, Texto

    Descripción generada automáticamente con confianza media

Bloque 5

Bloque 4

Bloque 3

Bloque 2

Bloque 1

* *Breve explicación*
  + *Versión 1:*
  + Bloque 1:

Se importan las librerías necesarias para la operación del programa desarrollado, y se definen aliases con el objetivo de simplificar las llamadas a dichas librerías.

* + Bloque 2:

En primera instancia, se define una función que genera un pulso rectangular de amplitud 1, que abarca el rango de -0.5 a 0.5 en el eje n, para simplificar la llamada, ya que esta está compuesta por una resta de dos escalones de Heaviside, lo cual la convierte en algo muy aparatoso como para llamarlo constantemente en el código.

En segunda instancia, se definen las variables necesarias para invocar a las funciones utilizadas, utilizando la variable discreta n como parámetro.

La función *f* es, como se ha expuesto anteriormente, un pulso rectangular de amplitud 1 comprendido entre -0.5 y 0.5 en el eje n.

La función *g* es una función exponencial con argumento -n+1, multiplicada por un escalón, tal que su valor sea 0 siempre que n sea menor a 0.

* + Bloque 3:

Utilizando la función *arange* de la librería *numpy* (función que devuelve un array de números en el intervalo especificado, con una distancia especificada entre valores), se define una única línea de tiempo sobre la cual se va a muestrear ambas funciones, para obtener una lista de valores.

Luego, se llama a la función *convolve*, de la misma librería (que devuelve una convolución lineal discreta de un array, que en este caso es nuestra lista de valores) y se escalan los valores de esa convolución por un diferencial de tiempo *dn* (espacio entre el valor actual y el anterior).

* Bloque 4:

Código que imprime, uno por uno, los valores del array de Convolución devueltos por la función *convolve* de *numpy*, por cuestiones de depuración.

* + Bloque 5:

Este bloque utiliza el módulo *pyplot* de la librería *matplotlib* para graficar tanto la convolución resultante como las funciones utilizadas para realizarla. Por una cuestión estilística, se decidió integrar todo en un mismo gráfico, haciendo que el eje izquierdo represente el eje vertical de las funciones *f* y *g*, mientras que el derecho represente el eje vertical de la convolución *y*. Cada muestra se encuentra representada por una Delta de Dirac escalada por el valor del punto correspondiente.

* + *Versión 2:*
  + Bloque 1:

Se importan las librerías necesarias para la operación del programa desarrollado, y se definen aliases con el objetivo de simplificar las llamadas a dichas librerías

* + Bloque 2:

En primera instancia, se define una función que genera un pulso rectangular de amplitud 1, que abarca el rango de -0.5 a 0.5 en el eje n, para simplificar la llamada, ya que esta está compuesta por una resta de dos escalones de Heaviside, lo cual la convierte en algo muy aparatoso como para llamarlo constantemente en el código.

En segunda instancia, se definen las variables necesarias para invocar a las funciones utilizadas, utilizando la variable discreta n como parámetro.

La función *f* es, como se ha expuesto anteriormente, un pulso rectangular de amplitud 1 comprendido entre -0.5 y 0.5 en el eje n.

La función *g* es una función exponencial con argumento -n+1, multiplicada por un escalón, tal que su valor sea 0 siempre que n sea menor a 0.

* + Bloque 3:

Utilizando la función *linspace* de la librería numpy (función que devuelve un array de números en el intervalo especificado, con una distancia especificada entre valores), se define para cada función (n para *f*, *n1* para *g*) una línea de tiempo sobre la cual se va a muestrear dicha función, con el fin de obtener un array de valores.

Luego, se llama a la función *convolve*, de la librería numpy (que devuelve una convolución lineal discreta de un array, que en este caso es nuestro array de valores.

* Bloque 4:

Código que imprime, uno por uno, los valores del array de Convolución devueltos por la función *convolve* de *numpy*, por cuestiones de depuración.

* + Bloque 5:

Este bloque utiliza el módulo *pyplot* de la librería *matplotlib* para graficar tanto la convolución resultante como las funciones utilizadas para realizarla. Por una cuestión estilística, se decidió integrar todo en un mismo gráfico, haciendo que el eje izquierdo represente el eje vertical de las funciones *f* y *g*, mientras que el derecho represente el eje vertical de la convolución *y*. Cada muestra se encuentra representada por una Delta de Dirac escalada por el valor del punto correspondiente.

* *Resultado de la ejecución*
  + *Versión 1:*

*Gráfico

Descripción generada automáticamente*

*Versión 2:*

*Gráfico, Histograma

Descripción generada automáticamente*

* *Análisis matemático manual:*

Para:

Con valores de h[n]:

|  |  |
| --- | --- |
| n | h[n] |
| 0 | 2,71828183 |
| 0,25 | 2,11700002 |
| 0,5 | 1,64872127 |
| 0,75 | 1,28402542 |
| 1 | 1 |
| 1,25 | 0,77880078 |
| 1,5 | 0,60653066 |
| 1,75 | 0,47236655 |
| 2 | 0,36787944 |

Con:

Realizamos el deslizamiento de la función h[n]:

Gráfico, Histograma

Descripción generada automáticamente

Sumando algorítmicamente, tenemos que:

Diagrama

Descripción generada automáticamente con confianza baja

Por lo que la convolución resultante será:

Gráfico, Histograma

Descripción generada automáticamente

Conclusiones y aclaraciones:

* Se ha limitado el análisis matemático manual a la Versión 2 (la original, en realidad) del código, debido a que, en el otro caso, las muestras son tan abundantes y con intervalos intermedios tan pequeños que es imposible calcularlo manualmente. La Versión 1 fue realizada como una prueba de concepto, para medir los efector del número de muestras sobre los resultados.
* La diferencia entre las versiones 1 y 2 radica en dos nociones principales:
  + En la versión 1, los resultados obtenidos poseen un alto grado de dependencia en la cantidad de muestras tomadas de la función original, y se van acercando progresivamente a los resultados reales a medida que se incrementa el número de muestras por división de escala. 25 muestras parece ser el mejor balance entre tiempo de computación, legibilidad y precisión de los resultados.
    - 5 muestras:

Gráfico, Histograma

Descripción generada automáticamente

* + - 15 muestras:

Gráfico, Histograma

Descripción generada automáticamente

* + - 25 muestras

Gráfico

Descripción generada automáticamente con confianza media

* + - 50 muestras

Tabla

Descripción generada automáticamenteTabla

Descripción generada automáticamente

* + En la Versión 1, el array de valores que describen el resultado de la convolución se encuentra, al igual que en el programa de Producto de Convolución, escalados por un diferencial que corresponde a la distancia entre cada valor de la lista, lo cual hace que coincidan con la integral realizada en el análisis manual; mientras que el código de la Versión 2 realiza los cálculos estrictamente como están definidos en la fórmula de la suma de convolución.

Se observa que, mientras que los resultados de la Versión 1 del código coinciden con los del producto de convolución (más allá de algunos errores eventuales de precisión), los resultados de la Version 2 no lo hacen, limitándose simplemente a adoptar el valor de la suma recursiva de los valores de la función *g* al pasar por cada posición discreta