Xibelly Eliseth Mosquera Escobar
CC 1035428318

Solución Tarea 4

Diseño y análisis de requisitos

De manera general se abordará el diseño de sofware necesario para llevar a cabo la solución de los problemas planteados. Para cada punto podemos establecer las necesidades y los requisitos.

Punto 1)

-En primer lugar necesitamos muestrear la función -f(x) = x*sin(x)- para luego interpolarla en puntos igualmente espaciados con frecuencia de 0.01. Para la interpolación haremos uso de los diferentes tipos de interpolación establecidos en GSL, creando una función para cada caso e imprimiendo en disco los resultados obtenidos.

-Para calcular la diferencia cuadrática media (**dms**), necesitamos comparar los datos que corresponden al muestreo de la función original con los datos interpolados. Para ello cargamos los datos interpolados y generamos un nuevo muestreo de la función con igual número de puntos que tienen los datos interpolados, talque por medio de una función diseñada para calcular la diferencia cuadrática se pasen dicho conjunto de datos y se retorne el valor que corresponde a la diferencia cuadrática entre estos. Dicho valor de la dms será impreso en pantalla talque el usuario pueda observar directamente que método entregar el menor dms y concluya cual es el mejor.

-A modo de prueba se realizan gráficas que ilustran la diferencia entre el primer muestreo y los diferentes métodos de interpolación, además se realizan gráficas individuales del segundo muestreo con cada uno de los métodos usados para interpolar.

Punto 2)

-Necesitamos generar 1000 números aleatorios uniformemente distribuidos entre el 0 y 2*pi, por lo que hacemos uso de los diferentes tipos de generación de números aleatorios disponibles en GSL, creando una función para cada caso, a partir de la generación de dichos números, obtenemos un muestreo para x de tal modo que podemos a su vez muestrear la función f(x). Para interpolar dicha función con el método que presenta el menor *dms*, debemos primero ordenar los datos en forma creciente y luego llamando una función que utiliza dicho método de interpolación podemos interpolar los datos generados.

Punto 3)

-Necesitamos integrar la función original entre (0,2 *M_PI) por medio de cuadraturas adaptativas usando Gauss-Konrod. Para ello cargamos de nuevo los datos interpolados que presenten el menor dms, esto se hace a través de una estructura de datos y luego usamos la librería de integración disponible en GSL "*gsl_integration*" donde se usa especificamente -*gsl_integration_qags*-. A manera de prueba se compara el valor análitico de la integración con el valor numérico obtenido y se calcula el error entre estos.

Punto 4)

-Necesitamos cargar de nuevo los datos interpolados con el menor valor de dms y se les suma un ruido gaussiano. Luego debemos ajustar este nuevo conjunto de datos con la siguiente función g(x) = a*x+sin(b*x). Para llevar a cabo este punto se diseña una programa que carga los datos interpolados y

se les suma un ruido gaussiano, haciendo uso de la siguiente función definida en GSL “gsl_randist” más precisamente -*gsl_ran_poisson*-, luego se imprime en disco. Posteriormente para el ajuste se crea un nuevo programa que leerá dicho conjunto de datos y usará “*gsl_multifit_nlin*” para efetuar el ajuste no lineal y estimar los valores de *a* y *b* que mejor describen el modelo dado.

Punto 5)

-Para calcular la transformada de fourier de la función definida en el punto 1) necesitamos hacer un muestreo uniforme de la función en el dominio (0,2 *M_PI) con frecuencia 1, 0.5 y 0.1 y hacer uso de la libre fftw3. Para tal fin, se efectúa el muestreo en cada caso y los datos son almacenados en un estructura talque estos son luego cargados a un vector tipo fftw_complex que es pasado directamente a la función que calcula la DFT -Discrete Fourier Transform- y nos retorna los datos que corresponden al calculo de la transformada de fourier.

Nota: en ambos caso se imprime en disco, tanto el muestreo como el calculo de la DFT.

A manera de prueba se calcula además de IDFT -Inverse Discrete Fourier Transform- de los datos obtenidos de hacer la DFT talque se pueda comprobar que en efecto el cálculo de la la transformada este bien realizado. Para ello se realizan gráficas comparativas entre el muestreo y la IDFT y gráficas correspondientes al calculo de la DFT.

Respuesta

Punto 1)

THE DIFERENCE MEAN SQUARE OF inter_lineal.dat IS: 0.320424

THE DIFERENCE MEAN SQUARE OF inter_poly.dat IS: 0.334178

THE DIFERENCE MEAN SQUARE OF inter_cspline.dat IS: 0.332490

THE DIFERENCE MEAN SQUARE OF inter_cspline_per.dat IS: 0.321951

THE DIFERENCE MEAN SQUARE OF inter_akima.dat IS: 0.334841

THE DIFERENCE MEAN SQUARE OF inter_akima_per.dat IS: **0.317622**

El mejor método de interpolación es el **akima periódico**

Punto 3)

result = -11.685656529851810248
exact result = -6.283199999999999896
estimated error = 0.000000000000129737
actual error = -5.402456529851810352
intervals = 1

Punto 4)

chisq/dof = 77.6223

A = 0.91364 +/- 0.01465
b = 0.99652 +/- 0.00396
status = success

Diseño del sistema

La solución a los problemas enunciados anteriormente se realiza en el lenguaje de programación C con un código modular el cual hace uso de librerías de GSL.

Punto 1)
El programa principal se llama **Interpolacion.c** el cual lhace llamados a **gsl_spline.**

Punto 2)
El programa principal se llama **numeros_aleatorios.c,** dicho programa llama a **gsl_rng,** para generar los números aleatorios y a **gsl_sort,** para ordenar los datos cargados de forma creciente y a su vez a hace llamdos a **gsl_spline** para interpolar con akima periódico los datos muestreados.

Punto 3)
El programa principal se llama **Integracion.c** dicho programa llama a **input.c** y a su vez hace llamados a **gsl_integration,** que se usa para cargar **gsl_integration_qags** y poder integrar la función interpolada.

Punto 4)
El programa principal se llama **ajust.c** dicho programa lee los datos que arroja el subprograma **random_noise.c** -diseñado para cargar los datos de la interpolación con el menor dms y les suma un ruido gaussiano con varianza 0.5- y hace llamado a **gsl_multifit_nlin**, que se usa para ajustar los datos al modelo dado.

Punto 5)
Hay tres programas **Fourier_transforms.c, Fourier_transforms2.c** y **Fourier_transforms3.c** cada uno de los cuales resuelve los casos a), b) y c) respectivamente.

Programas Gráficas

Para realizar cada gráfica se hace un script de gnuplot, esto son:

script_grafica_fourier_comparacion.sh → grafica el sampling de f(x) y la IDFT
script_grafica_fourier.sh → grafica la DFT
script_grafica_interpolacion_comparacion.sh → grafica el sampling de f(x) y cada uno de los métodos de interpolción
script_grafica_interpolacion.sh → grafica el sampling de f(x) y uno de los métodos de interpolación
script_grafica_gaussina_noise.sh → graficas los datos interpolados mas un ruido gaussiano

grafica_ajuste.gp → *grafica los datos con ruido gaussiano y su respectivo ajuste*

script_grafica_random_noise.sh → grafica el samplig de f(x) y lainterpolación obtenida con los

diferetes métodos para generar 1000 números aleatorios entre 0 y 2*pi