Centro de Investigación en Matemáticas, A.C.

Descripción tarea 10 - Métodos numéricos

Erick Salvador Alvarez Valencia

5 de Noviembre de 2017

1. Introducción

En el presente reporte se describirán los resultados de dos métodos empleados en la tarea 10 para la interpolación polinomial, el primero consiste en resolver un problema de minimización empleando elemento finito y el segundo consiste en realizar el desarrollo de un spline cúbico para la interpolación. Se presentarán los resultados de las ejecuciones así como algunas gráficas formadas a través de ellos.

2. Problema de minimización

2.1. Descripción

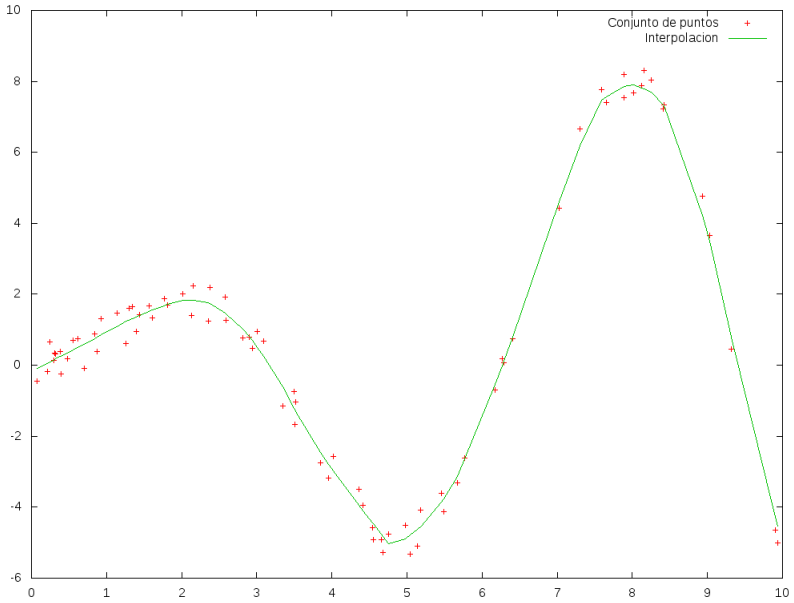
En este ejercicio se presentó un problema de minimización de la siguiente función:

$$min_{\phi_0,\phi_1,\ldots,\phi_n} \sum_{i=1}^{m}[\phi(x_i)-y_i]^2+\lambda \int_0^{10}(\phi')^2$$

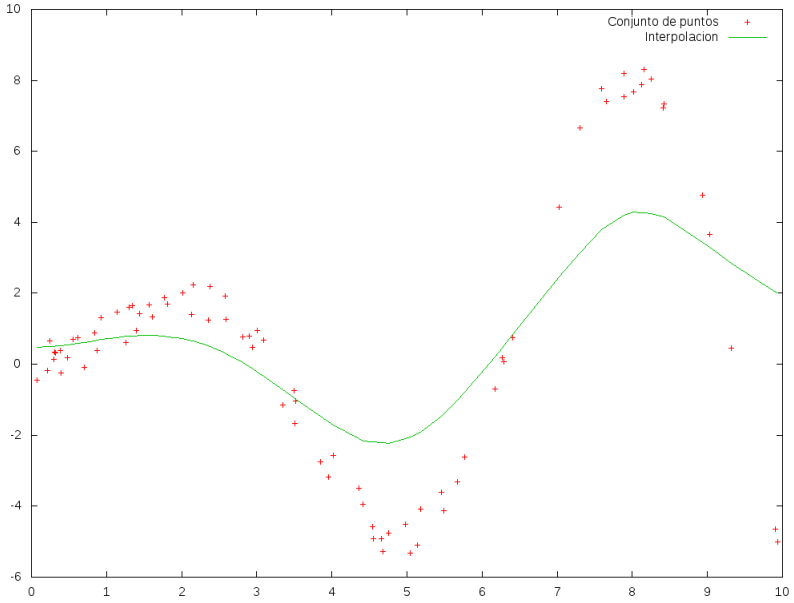
Para lo cual se utilizó la técnica de elemento finito, con ella se hizo una discretización para poder construir un sistema de ecuaciones tridiagonal y así encontrar el arreglo de ϕ_i. Al final el programa se probó con varios valores de λ los cuales generan una penalización en la parte de suavidad del polinomio interpolador.

2.2. Ejemplo de ejecución

A continuación se mostrará el resultado de la ejecución del programa usando dos conjuntos de datos distintos y a su vez valores de λ distintos.

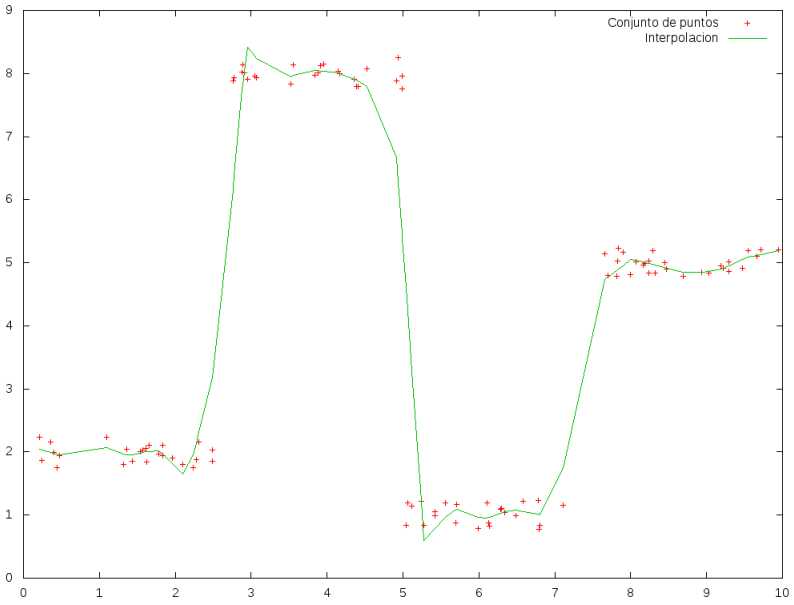


(a) Figura 1. Conjunto de datos 1, lambda = 0.1.

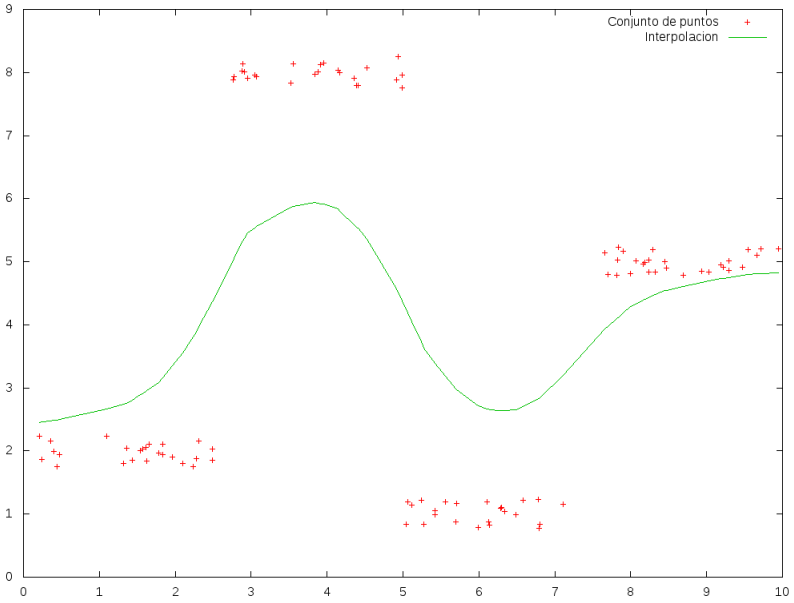


(b) Figura 2. Conjunto de datos 1, lambda = 10.

En las Figuras 1 y 2. podemos observar la ejecución del mismo algoritmo pero con una variación en el parámetro lambda que genera una gran penalización en el polinomio interpolador, ya que para el promer lambda dado se genera una muy buena interpolación y para el otro hay muchas diferencias en la nube de puntos con respecto a la curva.



(c) Figura 3. Conjunto de datos 2, lambda = 0.1.



(d) Figura 4. Conjunto de datos 2, lambda = 10.

Al igual que en el caso anterior, se ejecutó el programa con dos diferentes lambdas para ver la penalización provocada por las mismas. En las Figuras 3 y 4. se puede apreciar la diferencia de la interpolación provocada por un lambda de 0.1 y otro de 10.

2.3. Observaciones

Cabe destacar que en el código se definió la función $phi(x)$ como se describe en la presentación de la clase usando las funciones N_k y N_{k+1} y no usando la delta de Kronecker.

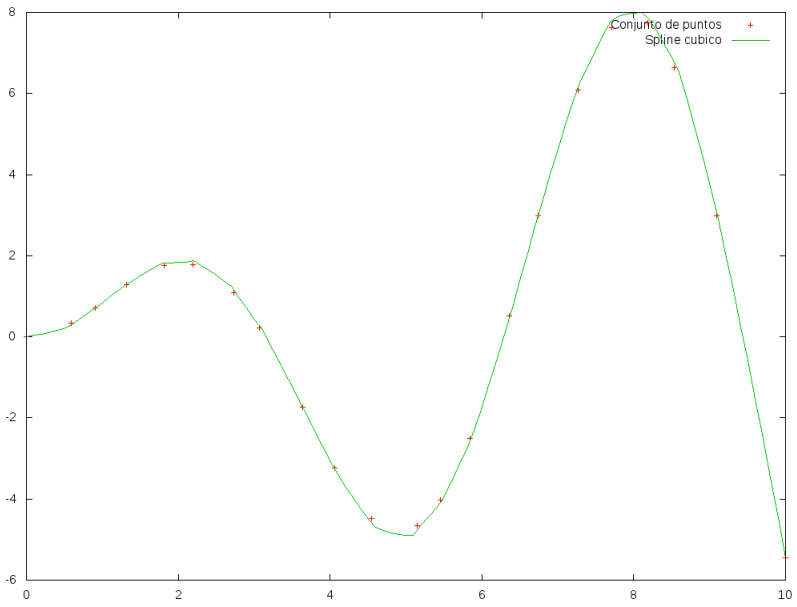
3. Splines

3.1. Descripción

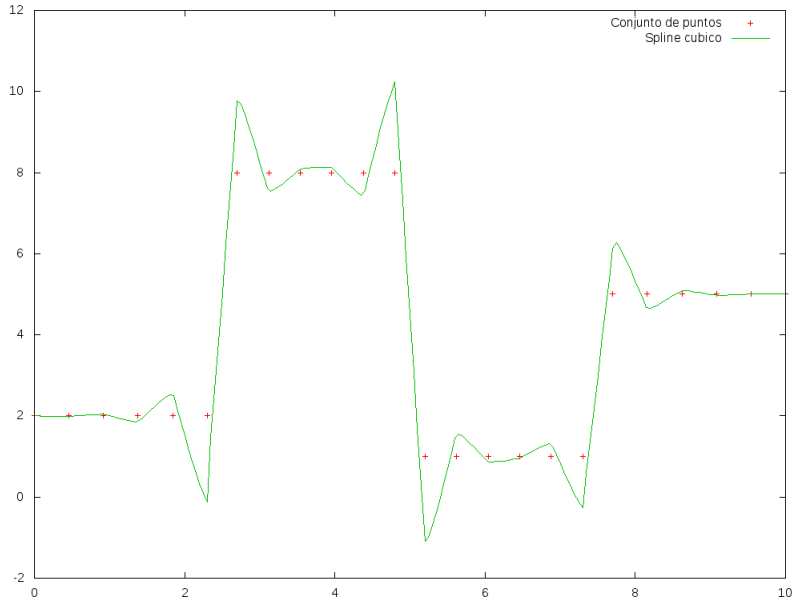
Para este ejercicio se dieron un conjunto de puntos y se propuso un spline cúbico que los interpolara, para ello se analizó el desgloce propuesto en el paper de la clase de Splines y se logró definir un sistema tridiagonal que al resolverlo nos daría el arreglo de M_i el cual serviría para generar los splines $S(x_i)$.

3.2. Ejemplo de ejecución

A continuación se mostrarán los resultados de la ejecución del programa con dos conjuntos distintos de puntos.



(e) Figura 4. Conjunto de datos 1.



(f) Figura 5. Conjunto de datos 2.

Se puede ver en las gráficas anteriores que los conjuntos de puntos fueron interpolados por los splines definidos a pares. Aunque en el conjunto dos se pueden notar unos picos y valles tal vez no deseados, esto puede presentarse porque la función está definida a trozos y se genera el genómeno de Gibbs.

4. Compilación y ejecución

Para esta tarea, los programas se dividieron en tres carpetas, mediante las cuales se encuentran los archivos del código y los de prueba, hay que mencionar que para cada programa se usará la misma forma de compilar, por lo que solo es necesario describir los pasos una vez.

Para compilar: En la carpeta encontraremos los archivos *.c* y *.h* con los que se podrá compilar el ejecutable. De la misma forma, en conjunto con los archivos anteriores, también podremos encontrar un Makefile para, en caso de encontrarse en linux, compilar de manera sencilla.

1. **Compilar usando Makefile:** En la terminal, nos colocamos en el directorio donde se encuentre el programa, y ejecutamos el comando *make*, automáticamente se realizará la compilación y se generará el ejecutable. El Makefile también contiene el comando *make clean* el cual limpiará los archivos generados por la compilación, incluyendo el ejecutable.

2. **Compilar directamente:** De la misma forma, podemos compilar directamente usando los siguientes comandos (en terminal):

   - gcc -c main.c -o obj/main.o
   - gcc -c memo.c -o obj/memo.o

- gcc -c reader.c -o obj/reader.o
- gcc -c matriz_vector.c -o obj/matriz_vector.o
- gcc -c met_num.c -o obj/met_num.o
- gcc -o main obj/main.o obj/memo.o obj/reader.o obj/matriz_vector.o obj/met_num.o -lm

Para ejecutar (programa 1): Únicamente debemos de usar el comando *./main* para ejecutar el programa en consola, este recibe los siguientes argumentos:

- **Un string:** El nombre del archivo binario donde se encuentra la matriz de puntos.
- **Un entero**: El tamaño de la discretización que se generará.
- **Un doble**: El valor λ que penalizará la interpolación.

El programa ejecutará generará un archivo de texto el cual contendrá los valores de los puntos del polinomio interpolador.

Para ejecutar (programa 2): Únicamente debemos de usar el comando *./main* para ejecutar el programa en consola, este recibe los siguientes argumentos:

- **Un string:** El nombre del archivo binario donde se encuentra la matriz de puntos.
- **Un entero**: El tamaño de puntos que contendrá el spline.

El programa ejecutará generará un archivo de texto el cual contendrá los valores de los puntos del spline.