Trabajo Práctico Nº 4

Luigi sólo admira a las Ferraris

Universidad de Buenos Aires
Facultad de Ciencias Exactas y Naturales
Departamento de Computación
Métodos Numéricos

Primer Cuatrimestre - 2007

Guillermo Túnez, Matías Albanesi, Mercedes Bianchi

16 de julio de 2007

Nombre	LU	E-mail
Guillermo Túnez	790/00	gtunez2002@yahoo.com.ar
Matías Albanesi Cáceres	522/00	mac@sion.com
María Mercedes Bianchi	685/97	mbianchi9@arnet.com.ar

Extracto: Los Splines permiten representaciones matemáticas de superficies partiendo de información relativa a algunos de sus puntos. Su construcción consiste en obtener una función de interpolación que pase por esos puntos.

Palabras Clave: COMPLETAR ESTOOOOO.

Índice

1. Cosas por hacer

A medida que las vamos terminando las podemos ir borrando de aqui o comentando

1.1. TP 4

- Optimizción de trayectorias. Desarrollar y probar. Existen dos alternativas:

  A. Ir cambiando el valor de omega y ver en que puno muere y ajustar los puntos en función del error cometido en el derrape.

  B. Fijar un omega e ir moviendo los puntos.

  Se fijó un omega y se fue cambiando la ubicación de los puntos para mantener la forma de la trayectoria, pero haciendo que la distancia recorrida sea mayor o menor en cada punto.

- Completar la parte de desarrollo. Nuestro Splines y nuestro calculo del mínimo.

- Recomendaciones validas para analisis de instancias.

  A. Graficar la velocidad y la aceleracion en funcion del tiempo juntas.

- Analizar la instancia provista por al catedra.

1.2. Recuperatorio TP 4

- Criterio de selecion del epsilon que representa la longitud minima de los intervalos dentro de los cuales se analiza la existencia de raices del polinomio proveniente de g(t).

- Recomendaciones validas para analisis de instancias.

  B. Mostrar una animacion en pantalla de la trayectoria del auto, junto con su vector velocidad y fuerza de frenado, para ver como evoluciona a medida que recorre la pista.

2. Introducción

2.1. Interpolación

La interpolación consiste en obtener una función que corresponda a una serie de datos conocidos. Una de las clases de funciones más útiles y mejor conocidas es la de los polinomios algebraicos, es decir el conjunto de funciones de la forma:

$$P(x) = a_n x^n + a_{n-1} x^{n-1} + ... + a_1 x + a_0$$

donde n es un entero no negativo y $a_n, a_{n-1}, ..., a_1, a_0$ son constantes reales. Una de las razones importantes por la cual se debe considerar esta clase de polinomios en la interpolación de funciones, es que la derivada y la integral de un polinomio son fáciles de determinar y también son polinomios. Por estas razones resulta conveniente en muchos casos aproximar una función mediante un polinomio que coincida con la función en un conjunto finito de puntos x_0 ... x_n.

2.2. Interpolación Polinómica Segmentaria

Un procedimiento alterno consiste en dividir el intervalo en una serie de subintervalos y en cada uno construir un polinomio (generalmente) diferente. A esta forma se le conoce como aproximación polinómica segmentaria. La aproximación polinómica segmentaria más simple es la interpolación lineal segmentaria que consiste en unir la serie de puntos dados por medio de una poligonal.

2.3. Splines Cúbicos

La aproximación polinómica segmentaria más común utiliza polinomios de grado tres entre cada par de puntos consecutivos y recibe el nombre de interpolación de trazadores cúbicos o Splines cúbicos.

Definición: Dada una función f definida en $[a, b]$ y un conjunto de puntos $a = x_0 < x_1 < ... < x_n = b$ un interpolante de trazador cúbico S para f es una función que cumple con las siguientes condiciones:

1. $S(x)$ es un polinomio cúbico denotado por $S_j(x)$ en el intervalo $[x_j, x_{j+1}]$ para cada $j = 0, 1, 2, ..., n-1$
2. $S(x_j) = f(x_j)$ para cada $j = 0, 1, 2, ..., n$

3. $S_j(x_{j+1}) = S_{j+1}$ para cada $j = 0, 1, 2, ..., n-2$ (lo que asegura la continuidad)

4. $S'_j(x_{j+1}) = S'_{j+1}(x_{j+1})$ para cada $j = 0, 1, 2, ..., n-2$ (lo que asegura diferenciabilidad en los puntos)

5. $S''_j(x_{j+1}) = S''_{j+1}(x_{j+1})$ para cada $j = 0, 1, 2, ..., n-2$ (lo que asegura que no hay cambios de concavidad en los nodos o puntos)

También deben satisfacerse uno de las siguientes conjuntos de condiciones en la frontera:

1. $S''(x_0) = S''(x_n) = 0$ (frontera libre o natural)

2. $S'(x_0) = f'(x_0) y S'(x_n) = f'(x_n)$ (frontera sujeta)

En términos generales, con las condiciones de frontera sujeta se logran aproximaciones más exactas, ya que abarca mayor información acerca de la función. Pero para que se cumpla este tipo de condición, se requiere tener los valores de la derivada en los extremos o bien una aproximación precisa de ellos. Si se desea construir el conjunto de polinomios de la interpolación de trazador cúbico de una determinada función f, se van aplicando cada una de las condiciones de la definición a un polinomio cúbico general.

2.4. Raíces de una Función

Para una función continua sobre un intervalo $[a, b]$, si $f(a)$ y $f(b)$ tienen signos opuestos, sabemos por el Teorema de Valor Medio la función alcanza el valor 0 dentro del intervalo (a, b). Esta propiedad es utilizada por los algoritmos de búsqueda de raíces en funciones, como en el caso de estos algoritmos clásicos:

1. Método de la Bisección

2. Método de Newton

2.5. Método de la Bisección

Dado un intervalo cerrado $[a, b]$ y una función continua f que cambia de signo en los extremos del intervalo, este método acota la raíz en un intervalo pequeño (según se desee). El algoritmo comienza con el intervalo conocido; lo divide a la mitad y evalúa el valor central. Si no se encontró la raíz, el valor tiene el mismo signo que uno de los extremos; quedarse con el subintervalo donde existe el cambio de signo (por lo tanto, contiene la raíz) y repetir hasta que el subintervalo sea lo suficientemente pequeño.

2.6. Método de Newton

Requiere un punto x que se sabe cercano a una raíz de la función continua. Iniciar en el mismo punto x del eje x. Luego ajustar una línea tangente a $f(x)$ en x obtener la solución para el punto y en la intersección de la tangente con el eje x repetir el proceso hasta que $|x - y| < epsilon$, y devolver y.

3. Desarrollo

3.1. Análisis del problema

3.1.1. Modelo matemático

Como paso previo al diseño de una solución en computadora del problema, analicemos el problema y sus implicaciones.

El problema a resolver consiste en analizar si una trayectoria de un automóvil (puntos en el plano) puede completarse cumpliendo una restricción (inecuación) que determina si el auto derrapa debido a una velocidad o giro excesivos.

Este análisis debe hacerse no sólo en los puntos dados sino en todos los puntos de la trayectoria; para esto debe plantearse un spline paramétrico que interpole los puntos dados como dato para obtener una ecuación de la trayectoria que pueda utilizarse para controlar si se cumple la restricción de no pérdida de adherencia.

El intervalo de tiempo entre puntos es constante, y se lo denomina ω.

Se pide chequear si el auto derrapa en algún punto mediante un método de búsqueda de máximos o mínimos de una variable.

Por último debe aplicarse el programa para analizar la mejor manera de recorrer un tramo de pista, o sea encontrar una serie de puntos dentro de la pista y un valor para ω tal que el auto no derrape, cuyos datos pueden encontrarse en el Enunciado.

3.2. Análisis previo al desarrollo en computadora

3.2.1. Plataforma y compilador

La primer decisión a tomar consiste en plataforma y compilador a utilizar. Como el desarrollo debe cumplir con el requisito de poder compilarse y ejecutarse con el software disponible en los laboratorios del Departamento de Computación, optamos por computadoras con procesadores compatibles con la arquitectura Intel x86, y el compilador g++ de la Free Software Foundation (disponible para Windows y Linux).

3.2.2. Entrada y salida

El formato es simple: un archivo de texto donde se especifica con una palabra clave el valor se define (por ejemplo, **omega 0.75**), uno por línea, permitiendo comentarios estilo C++. Todos los valores son en punto flotante exceptuando **cantidad_muestras** que es entero. Un ejemplo de archivo de entrada es el siguiente:

```
Init

masa 512
coeficiente_carga 3.8851
coeficiente_rozamiento 0.9389
omega 1.1

cantidad_muestras 10

direccion_inicial 1 -0.25
x,y 10 3
x,y 11 1
x,y 12 -2
x,y 13 -5
x,y 14 1
x,y 15 3
x,y 17 7
x,y 18 9
x,y 19 11
x,y 25 13

End
```

3.2.3. Implementación del spline paramétrico

Se diseñaron funciones (sin llegar a ser una clase C++) para calcular los coeficientes de un spline en una variable, para utilizar posteriormente dos Splines, uno para la coordenada X de la trayectoria y otro para la coordenada Y.

La primera decisión a tomar fue el tipo de frontera a utilizar en el spline: frontera libre o frontera sujeta. Optamos por utilizar una frontera sujeta en el extremo inicial de la trayectoria, ya que era de suponer que el vehículo tomaría la curva con una velocidad conocida, o este dato también sería deseable manipularlo para analizar el óptimo (además esto se pide por enunciado). Sin embargo, para el extremo final de la trayectoria optamos por una frontera libre. Esto es factible ya que el sistema de ecuaciones resultante de plantear las condiciones de este Spline es diagonal dominante y por lo tanto existe solución.

Optamos por realizar un spline que reciba una tabla de pares $(t,\ f(t))$ en lugar de pedir que los valores de t sean equidistantes, por generalidad y porque no representaba un costo alto de desarrollo.

También se reciben los valores de $f'_x(t_0)$ y $f'_y(t_0)$ por parámetro. La función devuelve una tabla donde para cada valor de la variable t_i se tienen cuatro valores correspondientes a los coeficientes del polinomio cúbico válido para el segmento $[t_i, t_{i+1}]$.

Se crearon funciones que dada la tabla y un valor cualquiera de t se calcula el valor del polinomio correspondiente al segmento donde está contenido t, y también las derivadas primera, segunda y tercera.

3.3. Inecuación de no-derrape

En cada punto de la trayectoria, el auto debe verificar esta ecuación:

$$m\,a(t) \;\leq\; (mg + v(t)^2 s)\,\mu_0$$

Para chequear que se cumpla esta condición, se la transformó en una función H:

$$H(t) \;=\; (mg + v(t)^2 s)\,\mu_0 - m\,a(t)$$

Entonces el problema de verificar la restricción para que el auto no derrape es ahora verificar que H(t) se mantenga positiva en todos los puntos de la trayectoria del auto.

3.4. Chequeo de signo positivo en $H(t)$

Para chequear que H(t) se mantenga positiva contábamos con dos enfoque principales:

- Buscar sus valores mínimos. En cada mínimo, ver si la función se vuelve negativa.
- Buscar los puntos críticos, mediante la búsqueda de raíces de la derivada. Luego evaluar la función en estos puntos, y ver si se vuelve negativa.

En ambos casos tenemos el problema de que los algoritmos disponibles en la teoría consultada requieren hipótesis bastante fuertes para trabajar. Por ejemplo, en el caso de la búsqueda de raíces, debe tenerse un cambio de signo. Y tanto en la búsqueda de mínimos como en la búsqueda de raíces, los algoritmos disponibles encuentran sólo UN mínimo o raíz, dentro del intervalo de búsqueda.

A pesar de tener disponibles los algoritmos de búsqueda de mínimos de www.nr.com (Numerical Recipes in C), preferimos la búsqueda de puntos críticos ya que la teoría nos era mejor conocida. De hecho, desarrollamos un método para chequear si una función se mantiene con signo positivo en un intervalo $[a, b]$ inspirado en el método de Bisección. El pseudo-código del algoritmo es el siguiente:

- evaluar $f(a)$. Si es negativo, terminar
- evaluar $f(b)$. Si es negativo, terminar

- Si $f'(a) = 0$ o $f'(b) = 0$, obtener nuevos valores para a y b de forma que el signo de la derivada no sea 0, e iterar.
- Si el signo de las derivadas en a y b cambia, hay un punto crítico. Localizarlo con el método de Bisección, y evaluar la función en la raíz p obtenida. Si se hace negativa, terminar. En caso contrario, buscar recursivamente si la función se mantiene positiva en $[a, p)$ y $(p, b]$.
- Si el signo de las derivadas en a y b no cambia, dividir el intervalo $[a, b]$ a la mitad, y chequear recursivamente si la función se mantiene positiva en cada sub-intervalo.

Tenemos como condición de parada para el algoritmo que el intervalo donde debe chequear se vuelve muy pequeño.

A pesar de la evidente ineficiencia del algoritmo, funciona correctamente y su implementación fue relativamente rápida, una vez resueltos *bugs* que hacían que no termine.

3.5. Método de Newton

Recibimos de parte de los docentes la recomendación de utilizar el método de Newton-Raphson para encontrar la raíz de una función luego de tenerla acotada en un intervalo con el método de Bisección. Inicialmente habíamos decidido hacerlo de esta manera, pero no logramos que funcione correctamente. De todas formas la razón principal fue otra: la ganancia utilizando ese algoritmo sería mínima, si de todas formas la función *chequearPositivoBisec* "barrería" cada intervalo de tamaño mínimo haciendo miles de evaluaciones de la función $H(t)$ y su derivada.

3.6. Problemas encontrados

Implementamos una variante del método de Bisección, que devuelve además del valor más aproximado encontrado de la raíz, el intervalo donde se "encerró" a la raíz. Esto fue necesario ya que este intervalo se utiliza en *chequearPositivoBisec* para seguir buscando recursivamente luego de encontrar un punto crítico.

3.7. Graficación

Utilizando la librería gráfica CImg (cimg.sourceforge.net) nuestra implementación dibuja los siguientes gráficos para una instancia del problema:

- Gráfico de la trayectoria en la pista, mostrando en los puntos dados como entrada los vectores velocidad y aceleración.
- Gráfico de la función $H(t)$ y su derivada $H'(t)$, para observar el punto donde $H(t)$ se vuelve negativa.
- Gráfico del módulo del vector velocidad
- Gráfico del módulo del vector aceleración, dandole signo según apunte "hacia adelante" o "hacia atrás" respecto del vector velocidad de la trayectoria.

4. Resultados

Ver los gráficos en la carpeta informe graficos del medio electrónico.

5. Discusión

No era necesario hacer un Spline general, hubiera sido más eficiente en el contexto del problema que los valores de la variable independiente sean equidistantes.

Debimos hacer un Spline de frontera sujeta en ambos extremos para poder acelerar hasta el final de la pista, en lugar de ir desacelerando hasta llegar a 0 en ese punto.

6. Conclusiones

- Pudo optimizarse la matriz utilizada para calcular los coeficientes del spline, ya que sabemos que es tridiagonal.
- Pudo perfeccionarse el cálculo del mínimo. Por ejemplo, utilizar los algoritmos que buscan mínimos locales para que una vez encontrado el mínimo buscar un máximo, para luego buscar el próximo mínimo.
-

7. Apéndice A

7.1. Enunciado

Laboratorio de Métodos Numéricos - Primer Cuatrimestre 2007
Trabajo Práctico Número 4: Luigi sólo admira a las Ferraris

Introducción

Nos encontramos en el grupo de análisis numérico de uno de los equipos de Fórmula 1, y estamos a pocas semanas de un gran premio que se realizará en un autódromo aún no visitado por la categoría. Nos enfrentamos al serio problema de decidir la mejor forma de tomar las curvas del nuevo circuito, y para esto solamente podemos realizar simulaciones, dado que el fin de semana de la carrera será la primera vez que nuestros autos tomen contacto con esta pista.

Consideremos una curva del circuito, como la que muestra la Figura 1. Queremos que nuestro auto comience en algún punto del segmento A con una cierta velocidad inicial, y que llegue a algún punto del segmento B en buenas condiciones (es decir, a una buena velocidad final, con las cuatro ruedas sobre el pavimento y con la carrocería en su lugar). Nuestro objetivo principal es que el auto pueda sortear la curva sin derrapar, dado que a la velocidad a la que circula se pierde el control muy rápidamente si el auto pierde adherencia.

Figura 1: Ejemplo de una curva y los puntos que determinan la trayectoria.

Para esto, trazamos una serie de puntos por los que esperamos que pase el auto a intervalos regulares de $\omega = 0{,}1$ sg. Es decir, en el instante $t = 0$ sg. el auto estará en p_1, en el instante $t = 0{,}1$ sg. estará en p_2, etc. Como los puntos corresponden a instantes equiespaciados en el tiempo, entonces estarán más separados en los tramos en los que el auto circule a mayor velocidad, y estarán más juntos en los tramos en los que el auto disminuya su velocidad.

Para determinar la trayectoria completa del auto, utilizamos un spline paramétrico que interpole estos puntos, con lo cual tendremos una función $T : \mathbb{R} \to \mathbb{R}^2$

que nos indicará explícitamente la posición del auto en función del tiempo. Es decir, en el instante t, el auto se encuentra en el punto $T(t)$. Notar que en función de este spline también podemos calcular el vector velocidad y el vector aceleración en cada instante de la trayectoria.

Cómo decidir si el auto se sale de la trayectoria

Sea t un instante del recorrido, en el cual el auto se encuentra en el punto $T(t) \in \mathbb{R}^2$. Llamemos $v(t)$ y $a(t)$ al módulo del vector velocidad y al módulo del vector aceleración en el instante t, respectivamente. En este instante, la fuerza de frenado que ejerce el auto sobre el piso es $F(t) = m\, a(t)$, siendo m la masa del auto. El auto no patina en el instante t si

$$F(t) \;\leq\; (mg + v(t)^2 s)\, \mu_0. \tag{1}$$

En esta fórmula, $g = 9{,}81$ m/sg^2 es la aceleración gravitatoria, s es el coeficiente de carga dinámica (de modo tal que $v(t)^2 s\mu_0$ es la fuerza que ejerce el aire sobre la superficie del vehículo en dirección normal a la superficie de la pista, representando la acción de los alerones), y μ_0 es el coeficiente de rozamiento estático (que depende del compuesto de los neumáticos, de la composición del pavimento y de las temperaturas del piso y de los neumáticos). El auto se mantiene dentro de la trayectoria sin derrapar si la condición (1) se cumple en todos los puntos de la misma.

Enunciado

El objetivo del trabajo práctico es implementar un programa que tome los datos del auto y de la trayectoria desde un archivo, que calcule un spline paramétrico que interpole los puntos de la trayectoria y que determine si en algún punto del recorrido el auto pierde adherencia. En caso positivo, el programa deberá informar el punto en el cual esto sucede, y la velocidad y dirección del auto en ese momento.

El programa deberá tomar los datos de entrada desde un archivo de texto cuyo formato queda a criterio del grupo. Debe notarse que los datos de entrada incluyen, además de los puntos de la trayectoria, los datos del auto (masa, coeficiente de rozamiento, coeficiente de carga dinámica, etc.) y el intervalo ω entre puntos consecutivos de la trayectoria.

Se deberá utilizar un algoritmo eficiente para el cálculo del spline paramétrico. Se debe mencionar en el informe el tipo de spline que se usa para la interpolación, comentando las diferentes opciones consideradas y las razones que llevaron a seleccionar esta opción.

El cálculo para determinar si el auto derrapa en algún punto de la trayectoria deberá estar basado en un método numérico a elección del grupo para buscar máximos o mínimos de funciones de una variable. Sugerimos consultar con

los docentes antes de utilizar algún método numérico que no se encuentre dentro de esta categoría.

Además de estos puntos obligatorios, los invitamos a implementar los siguientes ítems optativos que pueden ser de interés para el análisis:

- Realizar gráficos de la velocidad y la aceleración en función del tiempo. También puede ser interesante graficar la fuerza $F(t)$ en función del tiempo, para ver en qué momento el giro se torna peligroso.
- Mostrar una animación en pantalla de la trayectoria del auto, junto con su vector velocidad y la fuerza de frenado, para ver cómo evolucionan a medida que el auto recorre la pista.

Experimentos obligatorios

Utilizando el programa implementado, analizar la mejor forma de tomar la curva mostrada en la Figura 2, sabiendo que $\mu_0 = 0{,}9389$, $s = 3{,}8851$ kg/m y $m = 512$ kg. Buscar la velocidad inicial que permite tomar las curvas lo más rápidamente posible. Analizar si conviene tomar cada curva con un radio de giro pequeño pero a baja velocidad, o si es mejor describir una trayectoria más abierta pero a mayor velocidad. Buscar en qué momento conviene frenar, y desde qué punto ya es seguro acelerar. El grupo que logre la trayectoria más rápida obtendrá una botella de champagne y el derecho a mojar a sus compañeros a modo de festejo.

Figura 2: Curva para los experimentos.

Fecha de entrega: Lunes 25 de Junio.