

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

Métodos Numéricos

Trabajo Práctico N°3

¡El TP del Mundial!

Nombre	LU	Mail
Thomas Fischer	489/08	tfischer@dc.uba.ar
Kevin Allekotte	490/08	kevinalle@gmail.com
Pablo Herrero	332/07	pabloherrero@gmail.com

Abstract

Este trabajo pretende encontrar una estrategia para predecir el curso de una trayectoria, dados algunos puntos que pueden tener errores. Está basado en los métodos de Splines y Cuadrados mínimos.

Spline Cuadrados mínimos Trayectoria Extrapolación

Índice

Introducción teorica	2
Desarrollo	3
Resultados	8
Discusión	10
Conclusiones	11
Apéndices	11
Apéndice A: Enunciado	11
Apéndice B: Codigos Fuente	13
Referencias	16

Introducción teorica

El objetivo de este trabajo práctico es usar los métodos de interpolación y aproximación de funciones vistos en la materia para predecir el curso de una trayectoria. La trayectoria está dada por puntos en el plano en intervalos de tiempo discretos, que pueden tener errores de aproximación pero en general describen una curva suave.

Los métodos que vamos a usar en este trabajo están basados en Splines y/o cuadrados mínimos.

Splines es un método de interpolación. Genera una curva suave que pasa exactamente por los puntos dados. Esta curva está formada por muchos polinomios cúbicos que se tocan en los puntos interpolados y además los pedazos consecutivos coinciden en la derivada y en la derivada segunda. La variante de splines que implementamos es el Spline cúbico natural, que también cumple la propiedad de que la derivada segunda en los extremos es 0.

Cuadrados mínimos es una técnica para encontrar los coeficientes de una función que aproxime a los puntos dados. A diferencia de splines, genera una curva suave que no necesariamente pasa por los puntos, pero los aproxima lo mejor posible. Por esto es más apropiado para cuando los datos no son exactos, pero la desventaja es que hay que conocer o adivinar el tipo de función. En el trabajo sólo vamos a usar este método para aproximar las curvas con funciones polinómicas de distintos grados.

Observamos que los métodos están definidos para funciones y no para trayectorias. Podemos entonces separar nuestra trayectoria en $x(t)$ y $y(t)$ y trabajar con estas funciones.

Desarrollo

Comenzamos por implementar los métodos de splines y cuadrados mínimos para poder hacer pruebas de su comportamiento para después aplicarlo a nuestro objetivo. Desarrollamos interfaces gráficas para ver las curvas que se estaban generando a partir de los puntos de la trayectoria. En la figura 1 se observa como interpolan una secuencia de puntos los dos métodos.

Vemos que cuadrados mínimos es útil para filtrar errores en los datos de entrada porque no tiene la condición de tener que pasar exactamente por los puntos. Genera un polinomio que aproxima muy bien a la trayectoria, siempre que se le pide el grado correcto. Si le pidieramos que aproxime una función con un grado demasiado bajo, no se va a ajustar correctamente y el polinomio resultante no representaría a la función. En cambio si el grado es demasiado grande, aparecerán oscilaciones inesperadas.

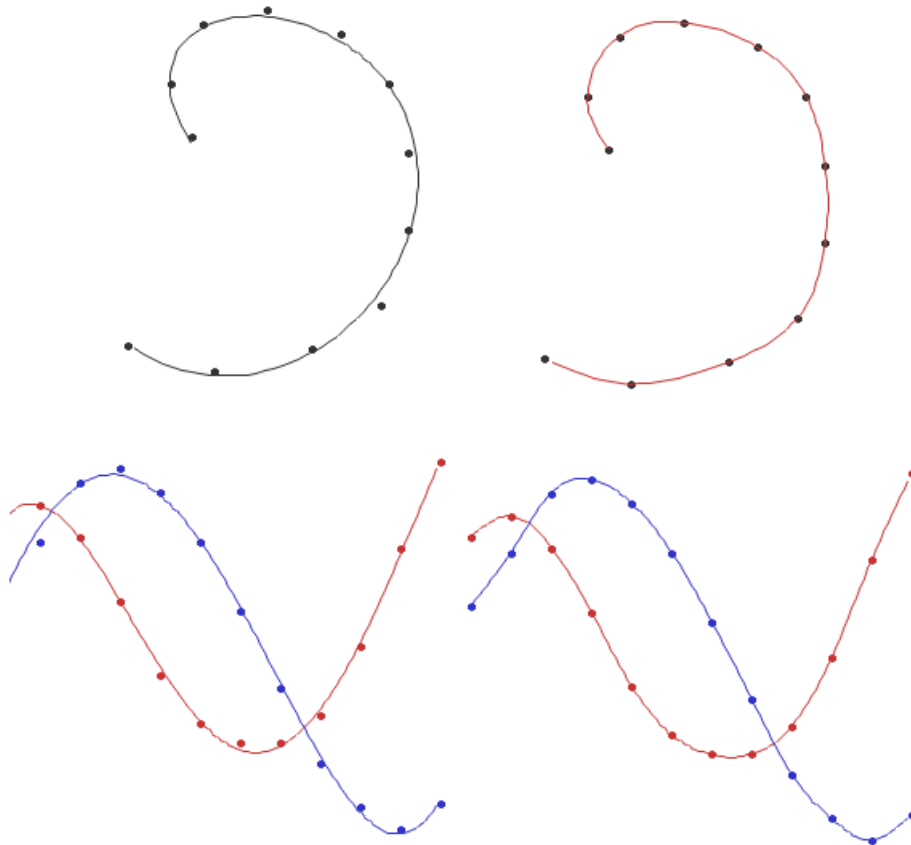


Figura 1: Arriba: Interpolación de puntos de una trayectoria con cuadrados mínimos (grado 4) y splines. Abajo: Interpolación con cuadrados mínimos y splines de $x(t)$ y $y(t)$ de la misma trayectoria.

El trabajo requería un método para predecir el curso de la trayectoria, y es por eso que nos interesa extrapolar las funciones pedidas. En el caso de cuadrados mínimos la extrapolación la tomamos simplemente como la continuación del polinomio para los valores más altos. Así, dados los puntos de una trayectoria, separamos ésta en dos funciones ($x(t)$ y $y(t)$), aplicamos cuadrados mínimos sobre cada función y obtenemos polinomios que aproximan a x e y . Para estimar la trayectoria en un punto futuro t , simplemente evaluamos los 2 polinomios en ese t y obtenemos una posición.

Con Splines podemos hacer algo parecido. La diferencia es que no devuelve una función global para toda la función, sino que genera una curva formada por muchos polinomios cúbicos. Para extrapolar decidimos usar el último o anteúltimo polinomio de la secuencia. Para un t futuro evaluamos SX_i y SY_i en t , siendo $i = n$ ó $n - 1$.

En la figura 3 vemos el comportamiento de las curvas obtenidas de los métodos para tiempos futuros con la técnica recién descrita. Entre otras cosas, vemos que el resultado de splines es muy sensible a variaciones en las posiciones de los puntos, pero a veces describe mejor lo que se esperaría del futuro de la trayectoria. Cuadrados mínimos funciona bien cuando el grado del polinomio es aproximadamente la cantidad de curvas de la trayectoria.

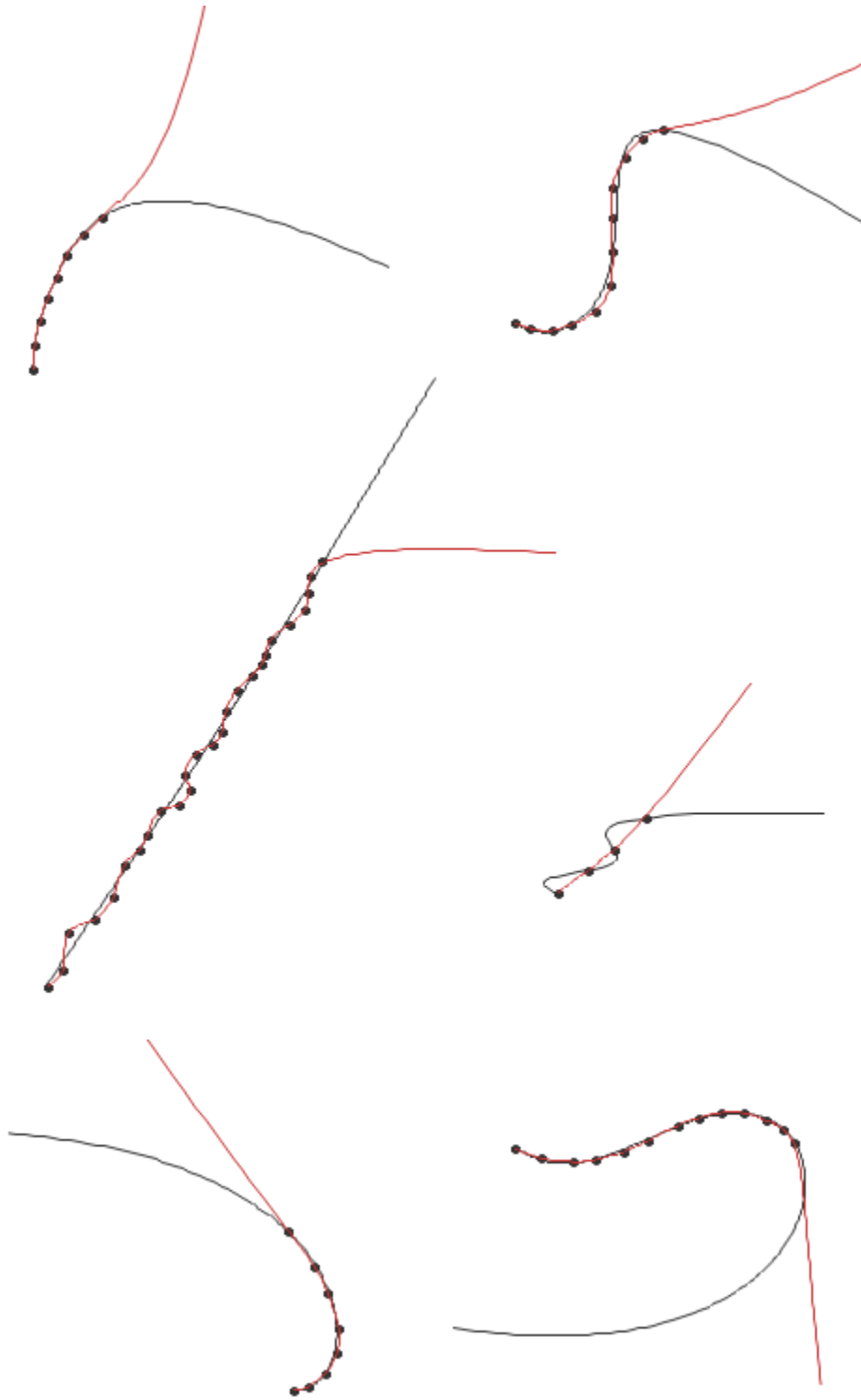


Figura 2: Extrapolación de trayectorias con cuadrados mínimos de grado 3 o 4 (negro) y con splines (rojo).

En la versión final, para adivinar el futuro de la trayectoria solamente utilizamos cuadrados mínimos. Además definimos estrategias alternativas para cuando no disponemos de suficientes datos para extrapolar con este método.

Si tenemos entre 2,3 o 4 puntos los aproximamos por un polinomio de grados 1,2,3 respectivamente y extrapolamos para calcular en que parte de la línea de gol va a llegar. Con más puntos primero los suavizamos. Para esto los aproximamos con un polinomio de grado a lo sumo 6 con cuadrados mínimos, y luego utilizamos para el resto del cálculo los puntos sobre esta curva con t entero. Decidimos utilizar solamente los 6 últimos puntos del paso anterior, porque consideramos que la trayectoria anterior que realizó no nos da información relevante. A estos 6 puntos los volvemos a someter a cuadrados mínimos, esta vez de grado 2, con el objetivo de encontrar los coeficientes de la curva que está realizando.

Hasta acá tenemos una aproximación de la curvatura del final de la trayectoria. Decidimos mover al arquero en función de esto. Calculamos el punto en el que el polinomio de grado 2 cruza la línea de gol (si existe) y movemos al arquero en esa dirección.

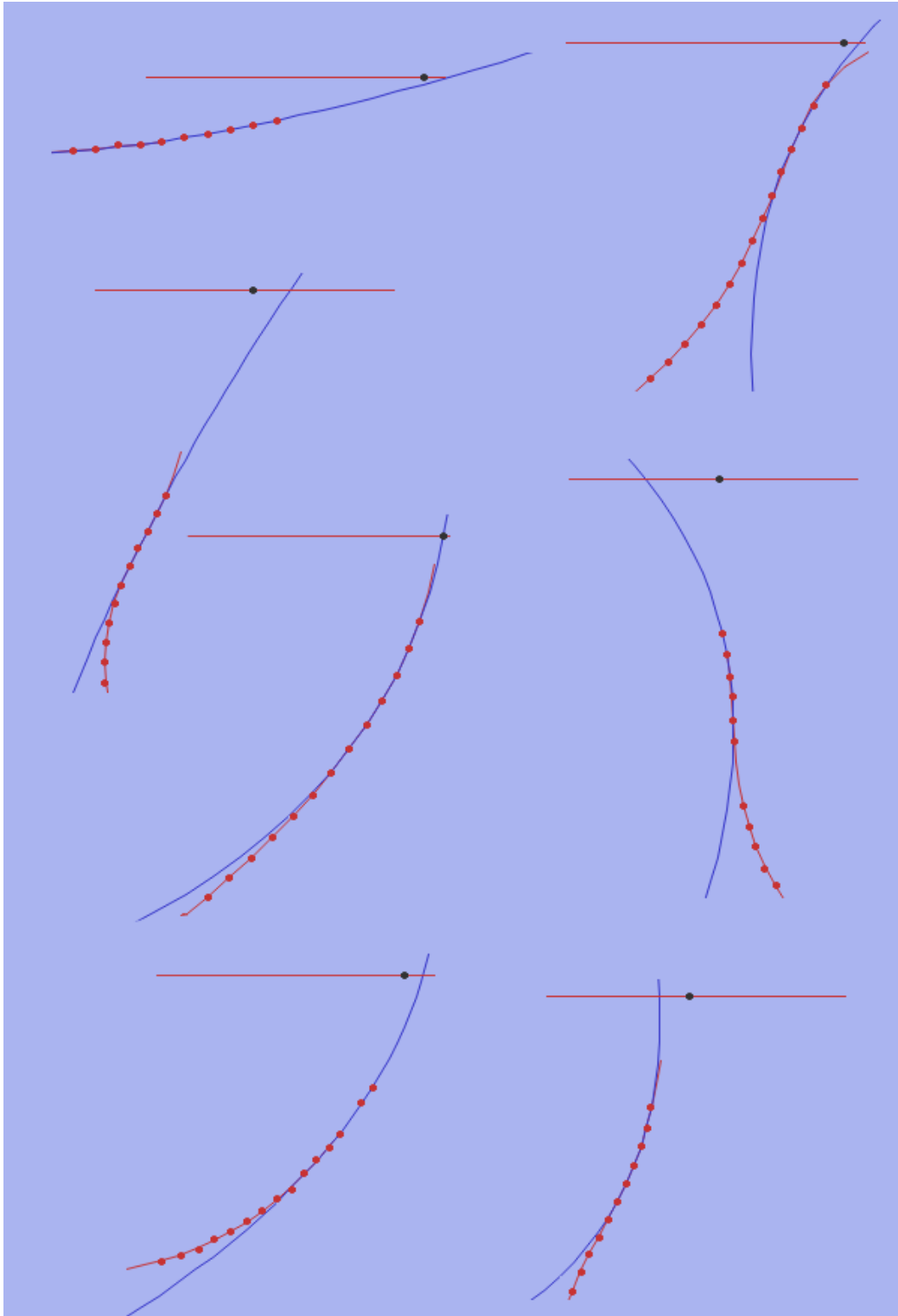



Figura 3: Acá se observan las curvas de grado 6 (rojo) y 2 (azul) que aproximan la trayectoria. El arco es la linea roja y el punto negro es el arquero.

Resultados



The screenshot shows a window titled "Resultado de las pruebas" with a table of test results. The table has three columns: "Arquero", "Test", and "Resultado". The first column contains the name "arquero1" for all rows. The second column lists various test scenarios, and the third column shows the results, which are either "El arqueeroo !!", "Gool !!", or "Gool !!". Below the main table, there is a summary table with columns "Arquero", "Atajo", and "de (tiros)", showing a score of 10 out of 18 for "arquero1". A "Cancel" button is located at the bottom of the window.

Arquero	Test	Resultado
arquero1	test_comba1	El arqueeroo !!
arquero1	test_comba2	El arqueeroo !!
arquero1	test_comba3	Gool !!
arquero1	test_comba4	El arqueeroo !!
arquero1	test_comba_ruido1	El arqueeroo !!
arquero1	test_comba_ruido2	Gool !!
arquero1	test_comba_ruido3	El arqueeroo !!
arquero1	test_comba_ruido4	Gool !!
arquero1	test_corner1	El arqueeroo !!
arquero1	test_corner2	El arqueeroo !!
arquero1	test_corner3	Gool !!
arquero1	test_recto1	El arqueeroo !!
arquero1	test_recto2	Gool !!
arquero1	test_recto3	Gool !!
arquero1	test_ruido1	El arqueeroo !!
arquero1	test_ruido2	El arqueeroo !!
arquero1	test_ruido3	Gool !!
arquero1	test_ruido4	Gool !!

Arquero	Atajo	de (tiros)
arquero1	10	18

Cancel

Figura 4: Ventana de resultados de la aplicación que prueba el arquero. 10/18 atajadas.

La versión final del arquero ataja unos cuantos tiros :). Con el método que usamos el arquero es bueno en tiros con muchos puntos de entrada aunque tengan ruido. Se pueden hacer optimizaciones para que ataje mejor los tiros rectos y los que tienen más velocidad, pero de todas las estrategias esta es la que mejor funcionó.



Figura 5: Resultados de algunos tiros satisfactorios e insatisfactorios al arco.

Discusión

Probamos muchos métodos y combinaciones para llegar a la estrategia final. Usando splines y cuadrados mínimos los combinamos y los probamos de distintas formas para estudiar técnicas alternativas.

Las primeras consistían por ejemplo en aproximar la trayectoria con cuadrados mínimos y encontrar la intersección con la línea de gol. Esta tenía el problema de que requería el grado del polinomio exacto, porque sino la extrapolación era inesperada.

También probamos interpolar con splines toda la trayectoria y extender el ultimo polinomio para encontrar la intersección. Esto era demasiado sensible al ruido y no siempre la extrapolación concordaba con la curvatura final.

Un método muy efectivo era suavizar la trayectoria aproximándola con cuadrados mínimos y luego usar splines con puntos sobre la curva, pero tenía algunos de los problemas de splines y a veces tenía comportamientos inesperados.

Finalmente nos decidimos por suavizar la curva con cuadrados mínimos y luego extrapolar con un polinomio cuadrático. Acordamos que los puntos iniciales de la trayectoria no brindaban información relevante porque podía haber cambios de curvatura a los que había que reaccionar. Dada la cantidad aproximada de puntos en la trayectoria y la cantidad de curvas que una trayectoria puede tener, vimos que cuadrados mínimos con grado menor o igual a 6 era el óptimo para suavizar los puntos. La razón por la cual la curva que extrapolamos es de grado 2 es porque define una trayectoria con una sola curvatura, que era lo que nos interesaba obtener. La idea era encontrar cómo estaba variando la dirección de la pelota para interceptarla. la información de la curvatura imaginamos encontrarla en aproximadamente los últimos 6 puntos de la trayectoria, también debido a la estimación de la cantidad de puntos de entrada.

Para probar las estrategias y visualizar las curvas escribimos scripts en `Python` con `pygame`. Luego volcábamos la estrategia a código `C`, pero los resultados no siempre eran los mismos. Posiblemente se deba a pequeñas diferencias en la implementación de los algoritmos que cambian el resultado.

Conclusiones

Durante el desarrollo del trabajo práctico implementamos algoritmos vistos en la clase teórica como Splines y Cuadrados mínimos y estudiamos su comportamiento para distintos sets de entrada. Pudimos aplicar estos métodos que eran para interpolación y aproximación de funciones, para extrapolar una función y predecir el curso de una trayectoria.

Es un problema difícil el de predecir trayectorias, pero con métodos de interpolación pudimos obtener algunos resultados satisfactorios. Sin embargo, las estrategias presentadas hacen suposiciones sobre la entrada que no siempre son ciertas. Es por esto que no son apropiadas para un caso mas general, y tienen que ser modificadas segun el contexto.

Apéndices

Apéndice A: Enunciado

Laboratorio de Métodos Numéricos - Primer cuatrimestre 2010 **Trabajo Práctico Número 3: ¡El TP del Mundial!**

Introducción

Nos encontramos en la máxima cita del fútbol mundial... de robots. Uno de los problemas más importantes a resolver en este contexto es predecir con la mayor anticipación posible la posición futura de la pelota en función de su posición en el pasado reciente. Sobre la base de estas predicciones se coordinan los movimientos de los jugadores de campo y, en el caso que nos ocupa ahora, la posición del arquero cuando existe peligro de gol.

Cuando la pelota se dirige hacia nuestro arco, es muy importante que ubiquemos el arquero en la posición exacta en la que la pelota cruzará la línea de gol, de manera que pueda interceptarla y evitar la caída de nuestra valla. El sistema de control de cada equipo suministra información en pasos discretos. En cada paso nuestras cámaras de video determinan la posición de la pelota, y debemos indicarle la acción a seguir al arquero: quedarse quieto, moverse hacia la izquierda o moverse hacia la derecha.

Los postes del arco están ubicados en las coordenadas $(-1,0)$ y $(1,0)$, y la línea de gol es el segmento entre estos dos puntos. Se marca un gol cuando la pelota cruza este segmento. Vistos desde arriba, la pelota es un círculo de 0.10 de radio y el arquero se representa mediante un segmento paralelo a la línea de gol de 0.10 de longitud y ubicado sobre la misma. Inicialmente el punto central del arquero se encuentra en la posición $(0,0)$, y en

cada paso se le indica al arquero qué acción debe tomar. Si se le indica un movimiento hacia alguno de los lados (izquierda o derecha) y en el paso anterior estaba quieto o se estaba moviendo hacia ese mismo lado, entonces el arquero se mueve 0.05 en la dirección indicada. Por el contrario, si en el paso anterior se había indicado un movimiento en la dirección opuesta, entonces el arquero se queda quieto durante el paso actual.

Un problema fundamental que debe enfrentarse es la presencia de ruido en las mediciones de la posición de la pelota. El sistema de visión está sujeto a vibraciones, golpes y errores de captura de datos, que hacen que las mediciones de la pelota sufran errores, e incluso registren posiciones irreales (es un efecto muy común que la pelota “desaparezca” en un cuadro y vuelva a aparecer en el cuadro siguiente). Por otra parte, la pelota no siempre viaja hacia el arco en línea recta sino que puede describir curvas más o menos complicadas, dependiendo del “efecto” dado por el jugador al momento de impactar la pelota y de posibles curvaturas en la superficie del campo de juego.

Enunciado

El objetivo del trabajo práctico es implementar un programa que tome como datos las posiciones sucesivas de la pelota y que determine en cada paso qué debe hacer el arquero para evitar el gol. Se deben tomar las mediciones con la posición de la pelota de un archivo de entrada, que tiene en cada línea el número de medición, la posición x y la posición y de la pelota, separados por espacios. La última línea del archivo tiene -1 como primer dato, indicando el fin de las mediciones.

En cada paso, el programa debe determinar qué acción debe tomar el arquero, escribiendo la decisión correspondiente en un archivo de salida. Cada línea de este archivo debe tener el número de medición y la acción del arquero (0: quedarse quieto, 1: izquierda, 2: derecha), separados por espacio. El programa debe tomar por línea de comandos el nombre del archivo de entrada y el nombre del archivo de salida. Dado que estamos simulando la decisión en tiempo real, para generar la acción correspondiente a una medición el programa solamente puede usar la información de esa medición y las anteriores (es decir, no se puede consultar lo que sucederá en el futuro para tomar las decisiones).

Las instrucciones al arquero deben estar basadas en un mecanismo de predicción de la posición futura de la pelota. Esta predicción se debe realizar sobre la base de algún método numérico visto en la materia, o alguna variación de los temas vistos en clase. Suggerimos consultar con los docentes del laboratorio para validar los enfoques que propongan implementar.

Se adjunta a este enunciado un programa simulador de los tiros al arco, junto con algunos archivos de prueba para testear el formato de los archivos de entrada y salida. Todos los programas participarán de un campeonato mundial de arqueros, y el grupo cuyo arquero logre atajar la mayor cantidad de tiros al arco se hará acreedor a la copa “Laboratorio de Métodos Numéricos” al mejor guardavallas del mundial.

Preguntas adicionales

1. ¿En qué minuto del partido Argentina-Unión Soviética del mundial Italia '90 se lesionó Nery Pumpido, arquero de la selección argentina?
2. ¿Cómo se llamaba el árbitro del partido Argentina-Francia en el mundial Argentina '78?
3. ¿Cuántos jugadores fueron expulsados en el mundial Italia '90 por derribar a Claudio Paul Caniggia?
4. ¿Cuántos pases hizo la selección argentina antes del gol de Maradona ante Grecia en el mundial EEUU 94?
5. ¿Cuántos minutos jugó Ricardo Bochini en la semifinal del mundial México '86?
6. ¿Cómo se llamaba el arquero suplente de la selección argentina en la final del mundial Uruguay '30?

Fecha de entrega: Viernes 25 de Junio

Apéndice B: Codigos Fuente

Función que calcula splines (Referencia ??)

```
Spline* GenerarSpline(const Punto* datos, const int m){
    int n=m-1;
    double a[n+1]; forn(i,n+1){ a[i] = datos[i].y; cout << a[i]<<~
        ~endl; }
    double h[n];
    forn(i,n) h[i] = datos[i+1].x - datos[i].x;
    double alpha[n]; alpha[0]=0; forsn(i,1,n) alpha[i] = (3./h[i~
        ~])*(a[i+1]-a[i])-(3./h[i-1])*(a[i]-a[i-1]);
    double l[n+1]; double mu[n+1]; double z[n+1]; l[0] = 1; mu[0]~
        ~ = z[0] = 0;
    forsn(i,1,n){
        l[i] = 2.*(datos[i+1].x-datos[i-1].x)-h[i-1]*mu[i-1];
        mu[i] = h[i]/l[i];
        z[i] = (alpha[i]-h[i-1]*z[i-1])/l[i];
    }
    double c[n+1]; double b[n]; double d[n];
```

```

l[n] = 1; z[n] = c[n] = 0;
forrn(j,n){
    c[j] = z[j]-mu[j]*c[j+1];
    b[j] = (a[j+1]-a[j])/h[j] - h[j]*(c[j+1]+2.*c[j])/3.;
    d[j] = (c[j+1]-c[j])/(3.*h[j]);
}
Spline* P = new Spline[n];
forrn(i,n){
    P[i].a = a[i];
    P[i].b = b[i];
    P[i].c = c[i];
    P[i].d = d[i];
}
return P;
}

```

Algoritmo que calcula cuadrados mínimos

Usa los módulos de Matriz y resolución de sistemas lineales implementados para el TP2

```

Matriz T_CM(Matriz x, int grado)
{
    Matriz res(x.Filas(),grado);
    forrn(i,res.Filas()) forrn(j,res.Columnas())
    {
        res(i,j) = pow(x(i,0),res.Columnas()-j-1);
    }
    return res;
}

Matriz CM(const double* x, const double* y, const int n, const int
grado)
{
    Matriz T(T_CM(Matriz(x,n,1),grado+1));
    Matriz T_t(T.T());
    Matriz A(T_t*T);
    Matriz b(T_t*Matriz(y,n,1));
    return A.resolver(b);
}

```

Algoritmo del arquero que decide cómo moverse

```

int ArqueroLoco::ComputarRespuesta(int i)

```

```

{
    int n = datos_i.size();
    int res = 0;
    if(n<2){
        // Me quedo quieto, no puedo deducir mucho
        Mover(0);
        Print(Matriz(0,0),Matriz(0,0));
        res = 0;
    }else if(n==2){
        // an ... a0
        // Extrapolo la recta que pasa por ambos puntos
        Matriz PX = Recta(datos_i.front(),datos_x.front(),datos_i.~
            ~back(),datos_x.back());
        Matriz PY = Recta(datos_i.front(),datos_y.front(),datos_i.~
            ~back(),datos_y.back());

        Print(PX,PY);
        res=Extrapolar(PX,PY);
    }else if(2<n && n<GRADO){
        // an ... a0
        // Extrapolo por CM comun de grado i-1
        Matriz PX = CM(ToArray(datos_i),ToArray(datos_x),datos_i.~
            ~size(),i-1);
        Matriz PY = CM(ToArray(datos_i),ToArray(datos_y),datos_i.~
            ~size(),i-1);

        Print(PX,PY);
        res=Extrapolar(PX,PY);
    }else{
        // Aproximo las funciones por cuadrados minimos de grado ~
        ~GRADO
        Matriz PX = CM(ToArray(datos_i),ToArray(datos_x),datos_i.~
            ~size(),GRADO);
        Matriz PY = CM(ToArray(datos_i),ToArray(datos_y),datos_i.~
            ~size(),GRADO);

        // Calculo los 6 ultimos puntos en los polinomios ~
        ~aproximados
        double i6[6] = { i-5 , i-4 , i-3 , i-2 , i-1 , i };
        double x6[6]; forn(k,6) x6[k] = Pn(PX,i6[k]);
        double y6[6]; forn(k,6) y6[k] = Pn(PY,i6[k]);

        // Aproximo los 5 ultimos datos con una cuadratica con CM
        Matriz PX2 = CM(i6,x6,6,2);
        Matriz PY2 = CM(i6,y6,6,2);
    }
}

```



```

// Calculo los 6 ultimos puntos en el polinomio aproximado
double xx6[6]; for(k,6) xx6[k] = Pn(PX2,i6[k]);
double yy6[6]; for(k,6) yy6[k] = Pn(PY2,i6[k]);

// Calculo los 6-1 splines interpolantes
//Spline* Sx = GenerarSpline(i6,xx6,6);
//Spline* Sy = GenerarSpline(i6,yy6,6);
Spline* Sx = GenerarSpline(APuntos(i6,xx6,6),6);
Spline* Sy = GenerarSpline(APuntos(i6,yy6,6),6);

// Extrapolo el anteultimo
Matriz SPx = SplineAMatriz(Sx[0]);
Matriz SPy = SplineAMatriz(Sy[0]);

Print(SPx,SPy);
res=Extrapolar(SPx,SPy);
}
return res;
}

```

Referencias

1. [http://en.wikipedia.org/wiki/Spline_\(mathematics\)](http://en.wikipedia.org/wiki/Spline_(mathematics))
Artículo de wikipedia sobre splines con un pseudocódigo de la implementación de Spline cúbico natural.
2. <http://www.youtube.com/watch?v=pRpeEdMmmQ0>
Video del Waka Waka eh eh