



DEPARTAMENTO
DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Métodos Numéricos

Trabajo práctico 3

Hay que poner un poquito más de esfuerzo...

Resumen

El presente trabajo estudia metodologías de estimación de trayectorias en tiempo dinámico. El caso de estudio es el Fútbol de Robots, dónde nos enfocaremos en el trabajo que realiza arquero para defender el arco de distintas jugadas. Las metodoloías se basan en los algoritmos de Cuadrados Mínimos y algunas Heurísticas.'

Integrante	LU	Correo electrónico
Danós, Alejandro	381/10	adp007@gmail.com
Gandini, Luciano	207/10	gl.gandini@gmail.com
Russo, Christian Sebastián	679/10	christian.russo@gmail.com

Palabras claves:

Cuadrados Mínimos, Factorizacion QR, Heurística, Fútbol de Robots

Índice

1. Introduccion Teorica	4
1.1. Factorizacion QR	4
1.1.1. Given	4
1.1.2. Householder	4
2. Desarrollo	5
2.1. Archivo de entrada	5
2.1.1. Explicacion	5
2.2. Archivo de salida	5
2.2.1. Explicacion	5
2.3. Método de estimacion: Cuadrados Mínimos	5
2.3.1. Cuadrados Mínimos: General	5
2.3.2. Cuadrados Mínimos: Específico a nuestro trabajo	6
2.3.3. Resolver Cuadrados Mínimos usando QR	6
2.3.4. Pseudocodigo	7
2.4. Demostraciones	8
2.5. Eurísticas	8
2.5.1. Baseline	8
2.5.2. Cuadrados Mínimos	9
2.5.3. Supuesto	9
2.5.4. Muestras Acotadas	9
3. Experimentación	10
4. Resultados	10
4.1. Experimentación con Lineales	10
4.1.1. Conclusiones	12
4.2. Experimentacion con Curvas	12
4.2.1. Conclusiones	15
4.3. Experimentacion con Jugadores	15
4.3.1. Conclusiones	19
4.4. Experimentacion con Exoticos	19
4.4.1. Conclusiones	21
4.5. Conclusiones Finales	21
5. Apéndice	22
5.1. Generador de Tests	22
5.1.1. Generador desde cero	22

5.1.2. Agregar la posición específica con x=125	22
5.2. Método de compilación	22
5.2.1. Forma manual	22
5.2.2. Forma automatizada generando los archivos .arq	22
5.2.3. Forma automatizada sin generarlos archivos .arq	23
5.3. Generadores de estadísticas	23
5.4. Equipo de pruebas	24
5.5. Referencias bibliográficas	24

1. Introduccion Teorica

1.1. Factorizacion QR

Definicion: Se dice que una matriz tiene **factorizacion QR** si puede ser expresada de la forma

$$A = Q^*R$$

El algoritmo para llevar a una matriz a su forma QR tiene costo $O(n^3)$. Tiene la misma ventaja que la factorizacion LU de permitir resolver un sistema de ecuaciones en orden $O(n^2)$, pero con la ventaja que **toda matriz tiene factorizacion QR**

$$Ax = b$$

$$QR x = b$$

$$Q^t Q R x = Q^t b$$

$$R x = Q^t b$$

con Rx un **sistema triangular superior**

Para poder calcular la matriz R se pueden aplicar los metodos de **Givens o Householder**

1.1.1. Given

Para eliminar el elemento en la posicion (i,j) aplicamos la siguiente matriz:

$$G(i, j, \theta) = \begin{bmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & c & \cdots & -s & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & s & \cdots & c & \cdots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{bmatrix}$$

Figura 1: Matriz de Givens

con $c = \cos(\theta)$ y $s = \sin(\theta)$. Luego aplicando $G(i,j,\theta) * A$ queda en 0 el elemento (i,j). Entonces aplicamos sucesivamente este procedimiento para todos los elementos que queremos poner en 0 obteniendo así nuestra matriz R. Luego $Q^t = \prod_{i=n}^1 G_i$

1.1.2. Householder

Con este metodo vamos eliminando los 0 de abajo de la diagonal columna a columna. Sea $x = col_i(A)$, $y = (\|x\|_2, 0, \dots, 0)$ y sea $u = x - y$. Definimos $H_i = I - \frac{2uu^t}{u^t u}$. Luego aplicando $H_i * A$ queda triangulada la columna i de A. Aplicamos este procedimiento iterativamente sobre $A^{(i)}$ hasta dejar triangulada la matriz. Quedando $Q^t = \prod_{i=n}^1 H_i$

2. Desarrollo

En esta sección describiremos los métodos usados para resolver el problema, cada uno con sus ventajas y desventajas.

2.1. Archivo de entrada

2.1.1. Explicacion

El ejecutable toma cuatro parámetros por línea de comando, que serán el *path* del archivo de entrada, el *path* del archivo de jugadores, el *path* del archivo de salida y la estrategia que utilizaremos con el arquero.

El archivo de entrada seguirá el siguiente formato:

- La primera línea contendrá la posición inicial del arquero en y , luego las coordenadas que definen los límites del arco, también sobre el eje y . Se asume que la posición en x del arquero y de la línea de gol son las mismas: $x = 125$. Finalmente estará μ , la cota sobre el máximo desplazamiento que puede realizar el arquero en un instante de tiempo.
- Luego se muestra la secuencia de posiciones en \mathbb{R}^2 , una por línea, que toma la pelota para los instantes de tiempo $0, 1, \dots, T$, siendo T el tiempo final.

En un primer lugar, leeremos la primera línea del archivo de entrada para *setear* los valores correctos de la posición en y del arquero, las posiciones de los palos y el μ . Luego, dado que se asume que no podemos saber qué pasará más allá del tiempo actual, iremos leyendo la entrada a medida que hagamos hecho los cálculos para el tiempo anterior.

2.2. Archivo de salida

2.2.1. Explicacion

El archivo de salida especificado como parámetro será creado en caso de que no exista y reemplazado por uno nuevo en caso de que ya exista. Este nuevo archivo contendrá una instrucción por línea, correspondiente a la acción que realiza el arquero en el instante $0 \leq t \leq T$, siendo T el instante final.

Este archivo luego podrá ser usado junto con el archivo de entrada para analizar qué sucede con el visualizador proporcionado por la cátedra.

2.3. Método de estimacion: Cuadrados Mínimos

Nuestro enfoque fue mirar al problema como si fuera uno de analizar los datos obtenidos en un experimento y tratásemos de describir la distribución de estos mediante una función.

En esta perspectiva, nuestra entrada sería el tiempo y la salida la posición en la cancha de la pelota. Además, como las variaciones en las coordenadas x e y de la pelota son independientes podemos dividir al problema en una entrada y dos salidas. De esta forma, deberíamos resolver dos problemas de cuadrados mínimos.

2.3.1. Cuadrados Mínimos: General

El estudio de Cuadrados Mínimos nació al querer describir el comportamiento de datos con funciones polinómicas. Normalmente, las mediciones traen inherentemente una cuota de ruido y si

se sospecha que éstas siguen un crecimiento de un polinomio de grado como máximo n , es difícil encontrar los coeficientes de este polinomio dado que el ruido afecta a los puntos. Cuadrados Mínimos trata de solucionar este problema.

Más formalmente, si se tiene m entradas y para cada una de ellas una salida asociada, x_i e y_i respectivamente, y se los quiere describir con un polinomio $P_n(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0$ de grado máximo fijo n , entonces la técnica de Cuadrados Mínimos busca a los $n + 1$ coeficientes $a_i \forall i = 0 \dots n$ resolviendo el problema buscar el vector a tal que minimice a la norma de $A \times a - b$ al cuadrado, con $A \in \mathbb{R}^{m \times (n+1)}$, $a \in \mathbb{R}^m$ y $b \in \mathbb{R}^n$ los siguientes:

$$A = \begin{pmatrix} x_0^n & x_0^{n-1} & \dots & x_0 & 1 \\ x_1^n & x_1^{n-1} & \dots & x_1 & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_m^n & x_m^{n-1} & \dots & x_m & 1 \end{pmatrix}, \quad a = \begin{pmatrix} a_n \\ a_{n-1} \\ \vdots \\ a_1 \\ a_0 \end{pmatrix} \quad y \quad b = \begin{pmatrix} y_m \\ y_{m-1} \\ \vdots \\ y_1 \\ y_0 \end{pmatrix}$$

La diferencia entre resolver directamente el sistema $A \times a = b$ y minimizar a $\|A \times a - b\|$ consta en que el primero busca a los coeficientes tal que el polinomio pasa exactamente por los puntos y_i , es decir, $P(x_i) = y_i \forall i = 0..m$, mientras que el segundo trata de buscar los coeficientes que minimicen a $\sum_{i=0}^m (P(x_i) - y_i)^2$, o la suma de los errores al cuadrado.

2.3.2. Cuadrados Mínimos: Específico a nuestro trabajo

En nuestro caso, deberíamos resolver dos problemas de Cuadrados Mínimos dado que para cada tiempo t_i tenemos dos coordenadas independientes: x_i e y_i . Si seguimos la notación anterior, la matriz A no cambiaría entre una coordenada y otra aunque sí el vector b sí tendría dos casos a parte, que llamaremos b_x y b_y .

Para resolver esta minimización se pueden usar por lo menos 3 métodos distintos: resolver mediante una factorización QR, mediante funciones normales o también factorizando usando descomposición en valores singulares. En las siguientes secciones explicaremos los métodos de QR.

2.3.3. Resolver Cuadrados Mínimos usando QR

Sea $A = Q^*R$ la factorización QR de la matriz A mencionada en las secciones anteriores. Entonces,

$$\min_{x \in \mathbb{R}^{(n+1)}} \|Ax - b\|^2 = \min_{x \in \mathbb{R}^{(n+1)}} \|Q^t Ax - Q^t b\|^2 = \min_{x \in \mathbb{R}^{(n+1)}} \|Rx - Q^t b\|^2$$

Como A tiene columnas independientes¹, entonces $R_{i,i} \neq 0 \forall i = 1, \dots, n+1$ y además $R_{i,j} = 0 \forall i = 1, \dots, m; j = 1, \dots, i-1$. La multiplicación matriz-vector Rx entonces sería:

$$Rx = \begin{pmatrix} R_1 x \\ 0 \end{pmatrix} \text{ con } R_1 \in \mathbb{R}^{(n+1) \times (n+1)} \text{ la parte por arriba de la diagonal de } R.$$

Además, si reescribimos a $Q^t b$ como:

$$Q^t b = \begin{pmatrix} c \\ d \end{pmatrix} \text{ con } c \in \mathbb{R}^{(n+1) \times (n+1)} \text{ y } d \in \mathbb{R}^{m-(n+1)}, \text{ problema se reduce a:}$$

$$\begin{aligned} \min_{x \in \mathbb{R}^{(n+1)}} \|Rx - Q^t b\|^2 &= \|(R_1 x, 0)^t - (c, d)^t\|^2 = \min_{x \in \mathbb{R}^{(n+1)}} \|R_1 x - c\|^2 + \|d\|^2 \\ &\rightarrow \min_{x \in \mathbb{R}^{(n+1)}} \|R_1 x - c\|^2 \rightarrow x/R_1 x = c \end{aligned}$$

¹Para más información, referirse a la sección demostraciones

$$R = \begin{bmatrix} \hat{R} \\ 0 \end{bmatrix} = \begin{bmatrix} \hat{r}_{11} & \hat{r}_{12} & \hat{r}_{13} \\ 0 & \hat{r}_{22} & \hat{r}_{23} \\ 0 & 0 & \hat{r}_{33} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad c = \begin{bmatrix} \hat{c} \\ d \end{bmatrix} = \begin{bmatrix} \hat{c}_1 \\ \hat{c}_2 \\ \hat{c}_3 \\ d_1 \\ d_2 \\ d_3 \end{bmatrix}$$

Figura 7: Ejemplo de rango completo.

El sistema $R_1 x = b$ tiene solución dado que R_1 es triangular superior con elementos no nulos en la diagonal. Si encontramos el x que sea solución para ese sistema, será el mismo x solución para el problema de Cuadrados Mínimos.

2.3.4. Pseudocódigo

Algorithm 1 FactorizacionQR(Matrix $A \in \mathbb{R}^{n \times m}$)

```

Matriz  $R \leftarrow A$ 
Matriz  $Q \leftarrow$  Matriz Identidad  $\in \mathbb{R}^{n \times n}$ 
Matriz  $Qt \leftarrow$  Matriz Identidad  $\in \mathbb{R}^{n \times n}$ 
for  $i = 0$  hasta  $m$  do
  if  $(n - i) > 1$  then
    Matriz  $tmp \leftarrow$  Matriz Identidad  $\in \mathbb{R}^{n \times n}$ 
    Matriz  $subQt \leftarrow$  Matriz Identidad  $\in \mathbb{R}^{(n-i) \times (n-i)}$ 
    Matriz  $subR \leftarrow generarSubMatriz(R, i) \in \mathbb{R}^{(n-i) \times (m-i)}$ 
     $(subR, subQt) \leftarrow triangularColumna(subR, subQt)$ 
     $R \leftarrow agregarSubMatrix(subR, R, i)$ 
     $tmp \leftarrow agregarSubMatrix(subQt, tmp, i)$ 
  end if
   $Qt \leftarrow tmp * Qt$ 
end for
return  $(Qt, R)$ 

```

Algorithm 2 generarSubMatrix(Matrix $A \in \mathbb{R}^{n \times m}$, int i)

```

Matriz  $res \leftarrow$  Matriz  $\in \mathbb{R}^{(n-i) \times (m-i)}$ 
 $res_{k,l} \leftarrow A_{i+k,i+l} \quad \forall k = 0, \dots, (n-i) \text{ y } l = 0, \dots, (m-i)$ 
return  $res$ 

```

Algorithm 3 triangularColumna(Matrix $sub \in \mathbb{R}^{n \times m}$, Matrix $subQt \in \mathbb{R}^{n \times m}$)

```

Vector x ← Vector de Ceros ∈ ℝn
Vector y ← Vector de Ceros ∈ ℝn
Vector u ← Vector de Ceros ∈ ℝn
for  $i = 0$  hasta  $x.n$  do
     $x_i \leftarrow sub_i$ 
end for
 $y_0 \leftarrow \|x\|$ 
 $u \leftarrow x - y$ 
Vector  $uTranspuesto \leftarrow u^t \in \mathbb{R}^{1 \times n}$ 
Vector  $aux \leftarrow$  Vector  $uTranspuesto * sub \in \mathbb{R}^n$ 
Matriz  $aux2 \leftarrow$  Matriz  $u * aux \in \mathbb{R}^{n \times m}$ 
int  $coeficiente \leftarrow 2/\|u\|^2$ 
 $sub \leftarrow sub - (aux2 * coeficiente)$ 
 $aux \leftarrow uTranspuesto * subQt$ 
 $aux2 \leftarrow u * aux$ 
 $subQt \leftarrow subQt - (aux2 * coeficiente)$ 
return (sub, subQt)

```

Algorithm 4 agregarSubMatrix(Matrix $sub \in \mathbb{R}^{(n-i) \times (m-i)}$, Matrix $A \in \mathbb{R}^{n \times m}$, int i)

```

 $A_{i+k,i+l} \leftarrow sub_{k,l} \quad \forall k = 0, \dots, (n-i) \text{ y } l = 0, \dots, (m-i)$ 
return Matriz A modificada

```

2.4. Demostraciones

En esta sección daremos demostraciones de los supuestos considerados en los algoritmos usados en el trabajo.

Sean $A \in \mathbb{R}^{m \times (n+1)}$ con:

$$A = \begin{pmatrix} x_0^n & x_0^{n-1} & \dots & x_0 & 1 \\ x_1^n & x_1^{n-1} & \dots & x_1 & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_m^n & x_m^{n-1} & \dots & x_m^1 & 1 \end{pmatrix}, \quad a = \begin{pmatrix} a_n \\ a_{n-1} \\ \vdots \\ a_1 \\ a_0 \end{pmatrix} \quad \text{y} \quad b = \begin{pmatrix} y_m \\ y_{m-1} \\ \vdots \\ y_1 \\ y_0 \end{pmatrix}$$

Lema Si $m > n+1$, entonces A tiene rango de columnas máximo.

Prueba: $A = (C_1, C_2, \dots, C_n, C_{n+1})$ si la miramos como columnas. Asumamos que no tiene rango máximo. Eso es equivalente a que:

$\alpha_1 C_1 + \alpha_2 C_2 + \dots + \alpha_n C_n + \alpha_{n+1} C_{n+1} = 0$ con $\alpha_i \in \mathbb{R}$ y $\alpha_i \neq 0$ para algún i , que es equivalente a que $A * \alpha = 0$. O sea, que el polinomio $P(x)$ de grado n tendría $m > n+1$ raíces dado que cada fila sería una evaluación en un punto distinta del polinomio dado que los x_i son distintos. Absurdo.

2.5. Eurísticas

2.5.1. Baseline

Como base para las comparaciones se definió el **método 0** que se compone de la siguiente inteligencia. Observa la posición actual de la pelota y se mueve sobre la línea del arco a la posición

de mas cercana a este posible respetando la velocidad del arquero.

2.5.2. Cuadrados Mínimos

Aquí solo utilizamos la estimacion ya mencionada y nos movemos hacia esa posicion respetando los límites de velocidad necesarios en adisión del chequeo de la distancia a los palos, para no realizar movimientos innecesarios el arquero se acercara a lo sumo hasta a lo sumo 6 puntos de distancia de ambos palos ya que podrá igualmente atajar y conserva una buena posición ante algun cambio brusco.

En caso de haber una sola muestra el arquero siempre se movera al centro del arco para posicionarse mejor ante el desconcierto del futuro.

De haber jugadores en el campo de juego, se borrará siempre el buffer de muestras si una pelota pasa cerca de un jugador, ya que en la mayoría de los casos este hará cambiar de posición.

Para la estimación se tomará como parámetro el grado máximo de cada eje permitiendo así lograr varias variaciones de estimación. Esto se observa en los métodos 1 a 10.

Con los polinomios ya calculados, se busca linealmente en que punto la pelota cruzará la linea de gol, de no encontrarse este punto se utiliza la metodología de moverse hacia el centro del arco, en caso contrario se busca mediante biyección el punto exacto entre el instante donde ya superó la linea y el instante anterior, encontrado este valor se calcula con el otro polinomio la posicion en ese punto, hacia alli se movera el arquero.

2.5.3. Supuesto

Basado en la heurística de Cuadrados Mínimos, agregamos al buffer de muestras una suposición, dado que solo nos interesa si la pelota va hacia el arco, agregaremos un dato inventado indicando que en un instante futuro la pelota se encontrará al fondo del centro del arco, esto creemos proporcionará al método la habilidad de predecir mejor ciertos movimientos de la pelota y adelantarse a cambios bruscos relacionados con toques de de jugadores o curvas pronunciadas que serán mal estimadas con pocos datos, creyendo que la pelota no logrará entrar al arco.

Esto se agrega a Cuadrados Mínimos sin incluir ningún parámetro más al método (sólo se pide grado máximo de los polinomios), al ejecutar el binario se pueden probar con los métodos 11 a 21

2.5.4. Muestras Acotadas

También basado en la heurística de Cuadrados Mínimos decidimos limitar la cantidad de muestras que utiliza el método, esto se agrega como parámetro del huerística e indica que se utilizarán solamente las últimas n muestras tomadas, claro esta que de haber sido tocada la pelota por un jugador, el conteo se reinicia.

De esta manera decidimos agregar los métodos 22 a 39 con los mismos parametros de entrada que las heurísticas anteriores pero variando la cantidad de muestras máxima en 3, 6 y 9.

3. Experimentación

El objetivo del trabajo era encontrar el *mejor* arquero usando técnicas de métodos numéricos. Qué dice qué arquero es mejor que otro es difícil de decidir cuando existen arqueros que atajan tiros distintos pero en total la misma cantidad. En esta sección trataremos de decidir utilizando los siguientes criterios:

4. Resultados

4.1. Experimentación con Lineales

Utilizando lineal6.tiro

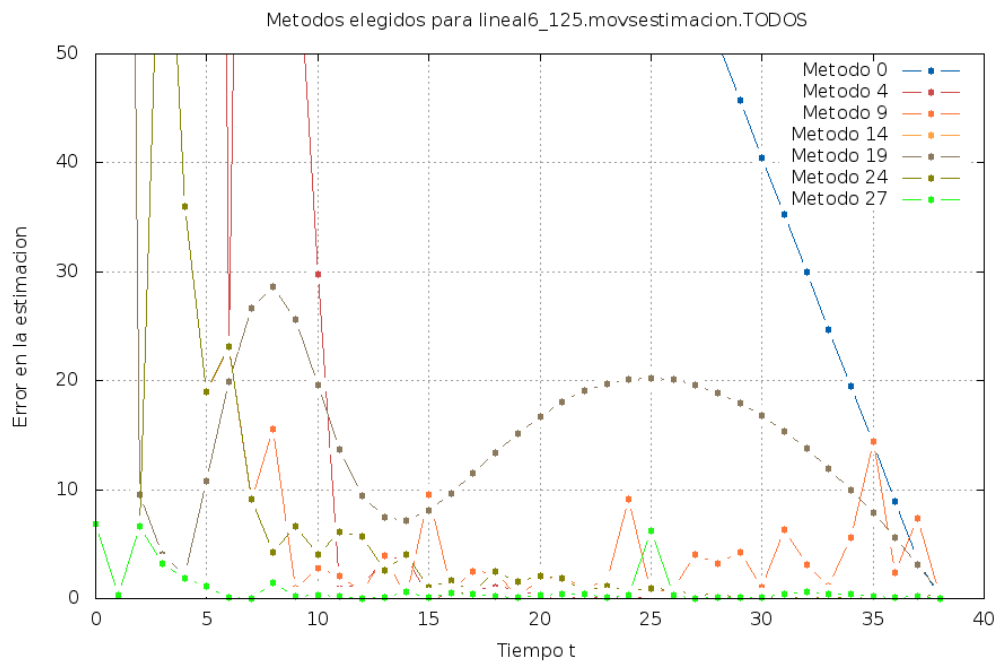


Figura 2: ESTIMACION de tiro lineal6

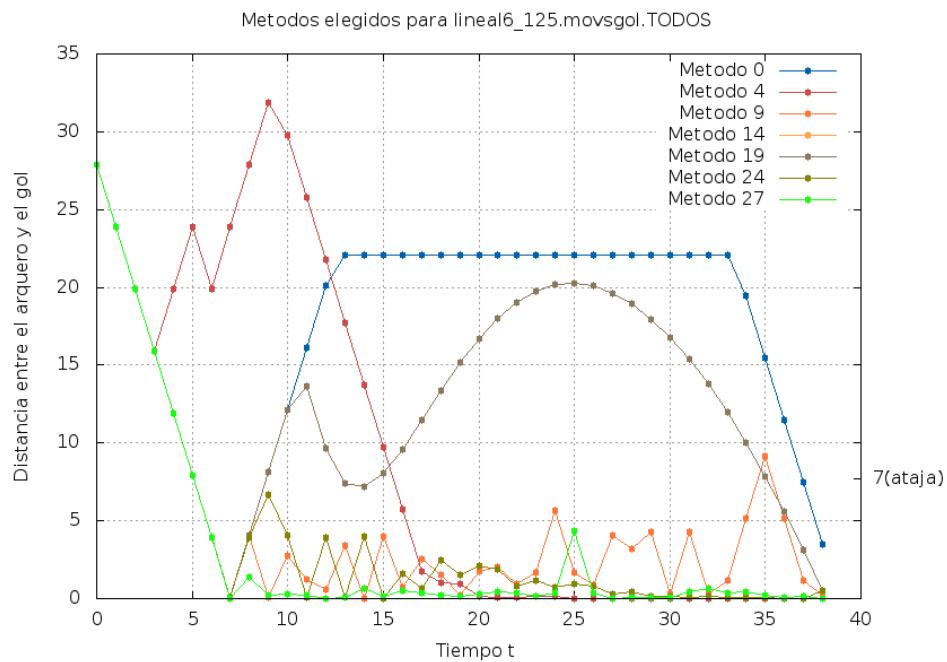


Figura 3: MOVSGOL de tiro lineal6

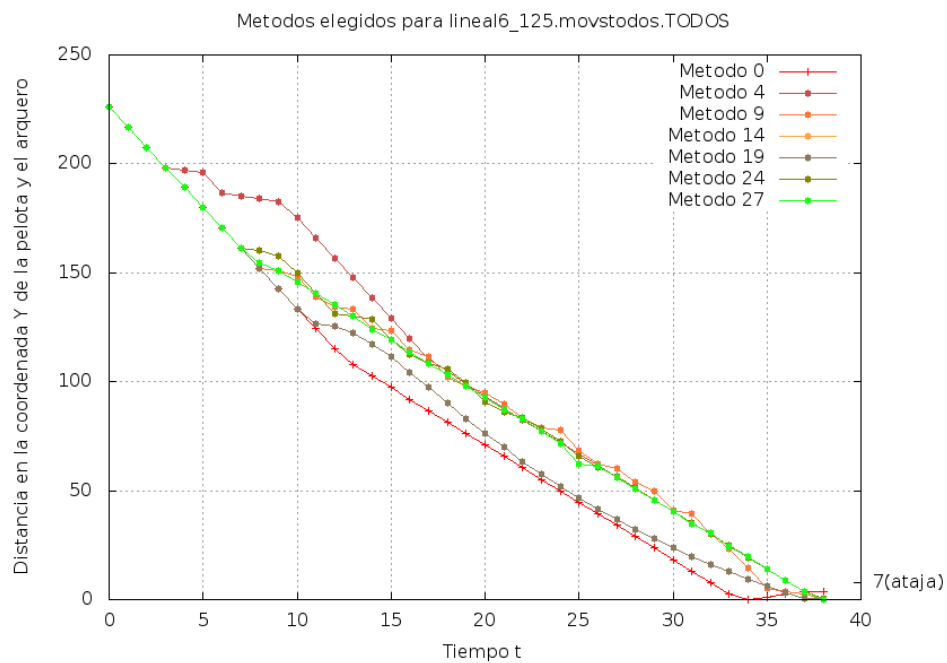


Figura 4: MOVSTODOS de tiro lineal6

4.1.1. Conclusiones

4.2. Experimentacion con Curvas

Utilizando cuadr3.tiro

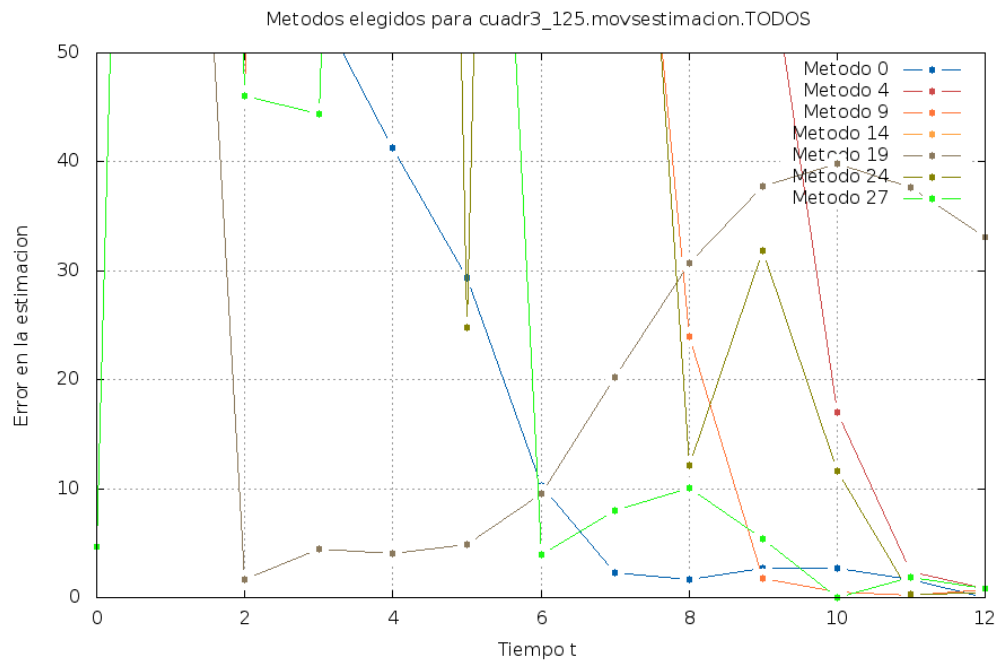


Figura 5: Estimacion de tiro cuadr3

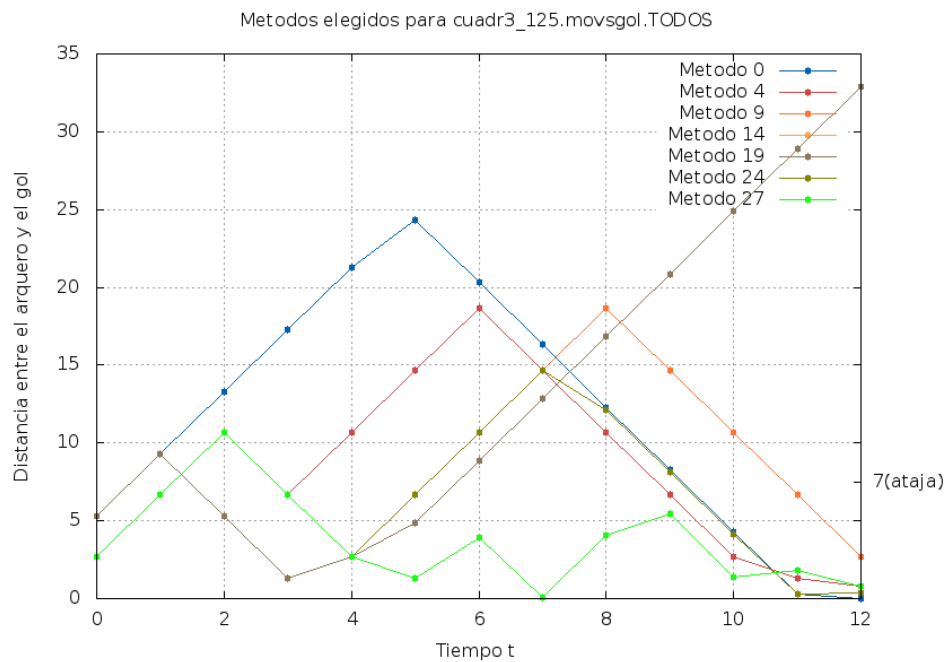


Figura 6: MOVSGOL de tiro cuadr3

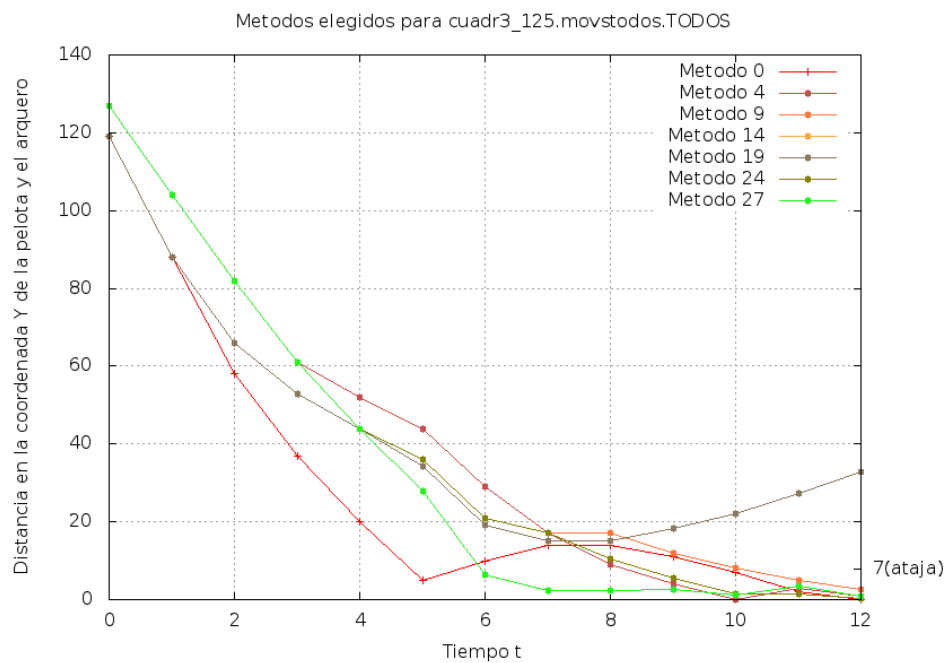


Figura 7: MOVSTODOS de tiro cuadr3

Utilizando cuadr6.tiro

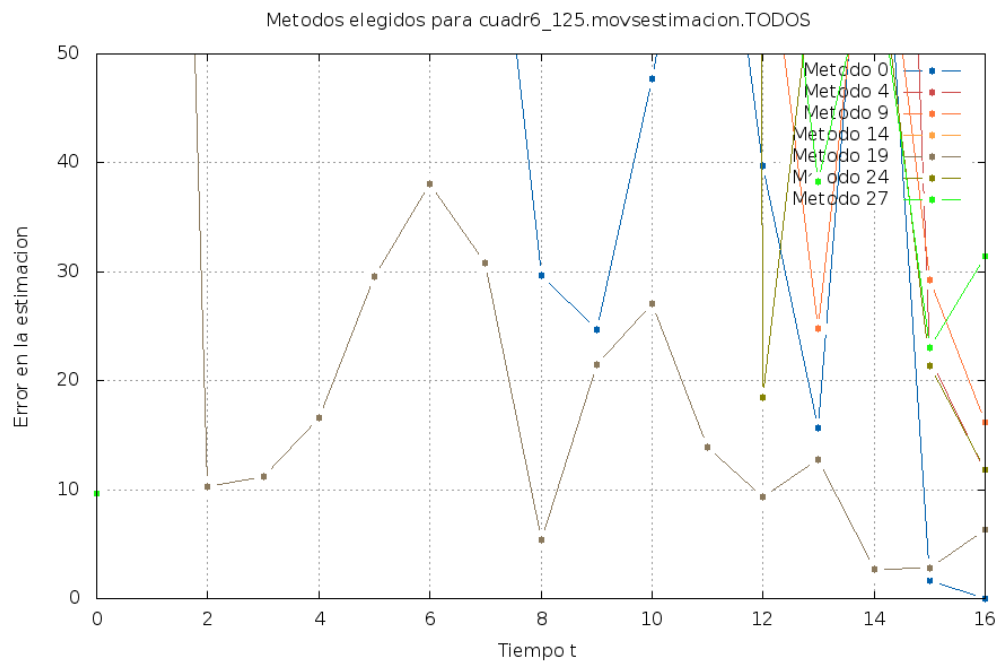


Figura 8: Estimacion de tiro cuadr6

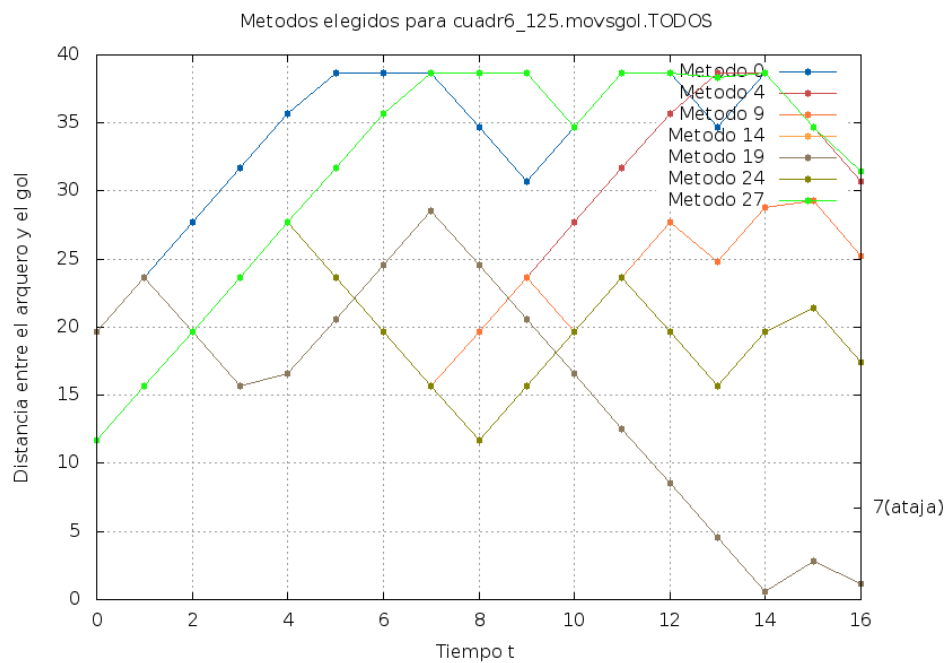


Figura 9: MOVSGOL de tiro cuadr6

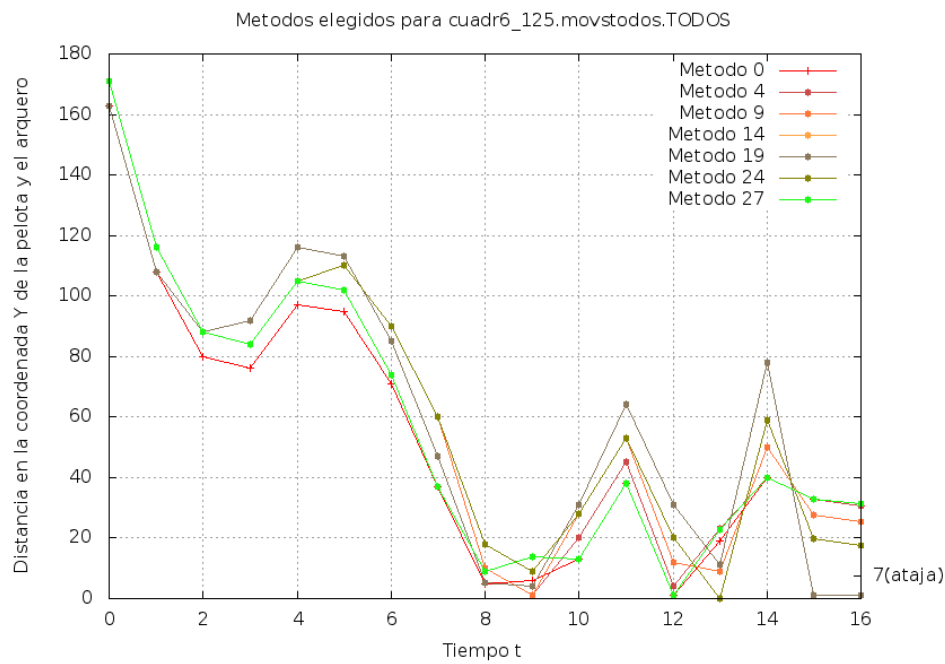


Figura 10: MOVSTODOS de tiro cuadr6

4.2.1. Conclusiones

4.3. Experimentacion con Jugadores

Utilizando conj4.tiro

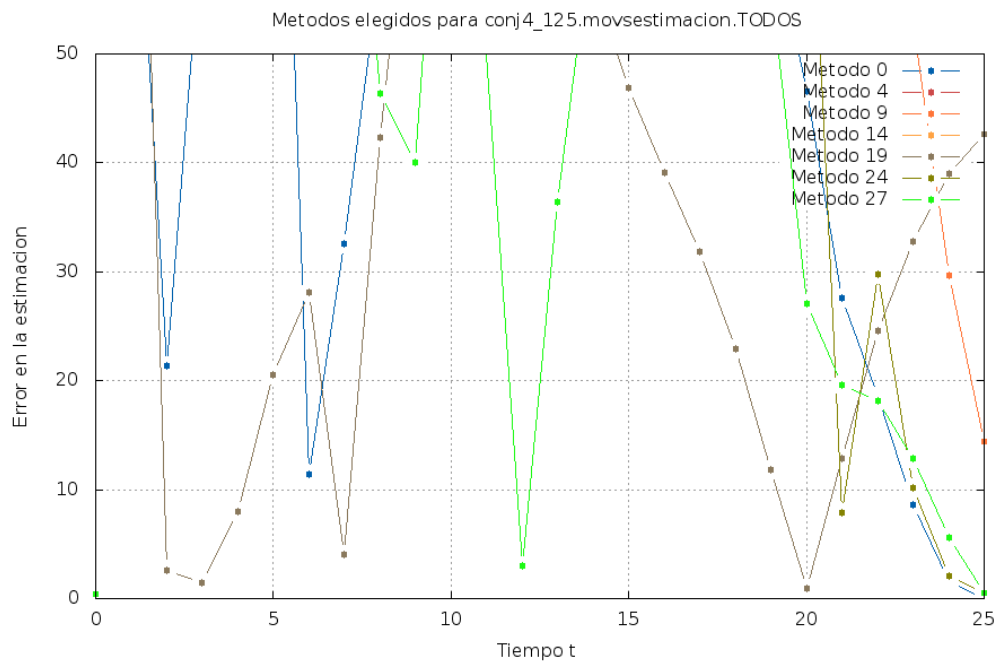


Figura 11: Estimacion de tiro conj4

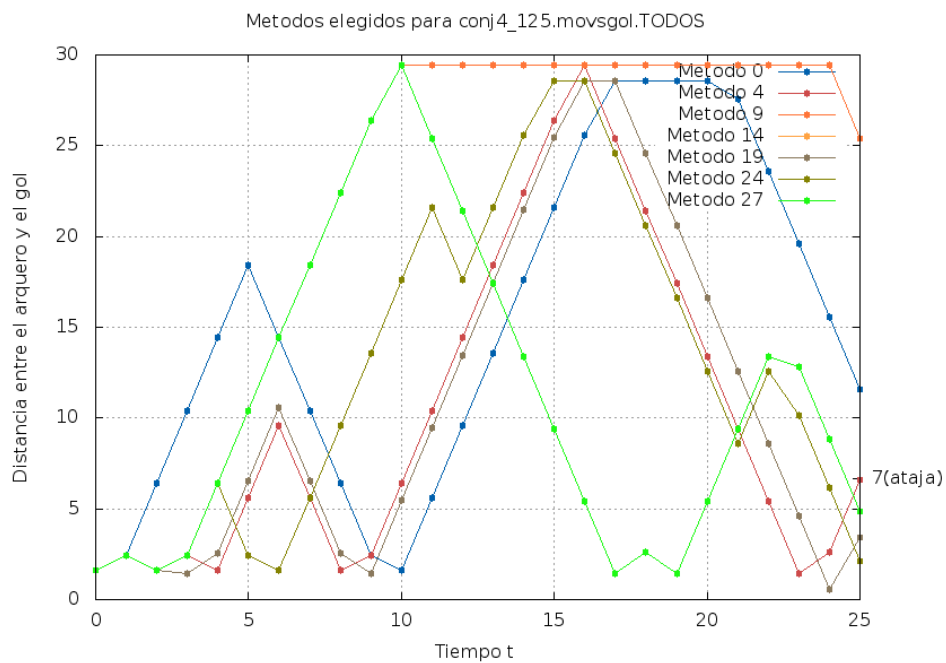


Figura 12: MOVSGOL de tiro conj4

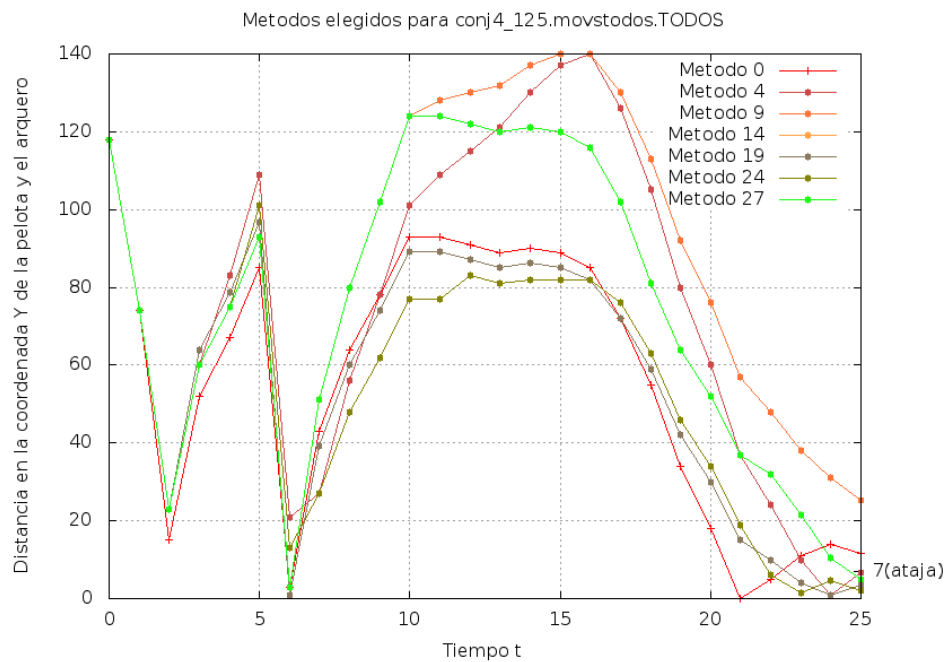


Figura 13: MOVSTODOS de tiro conj4

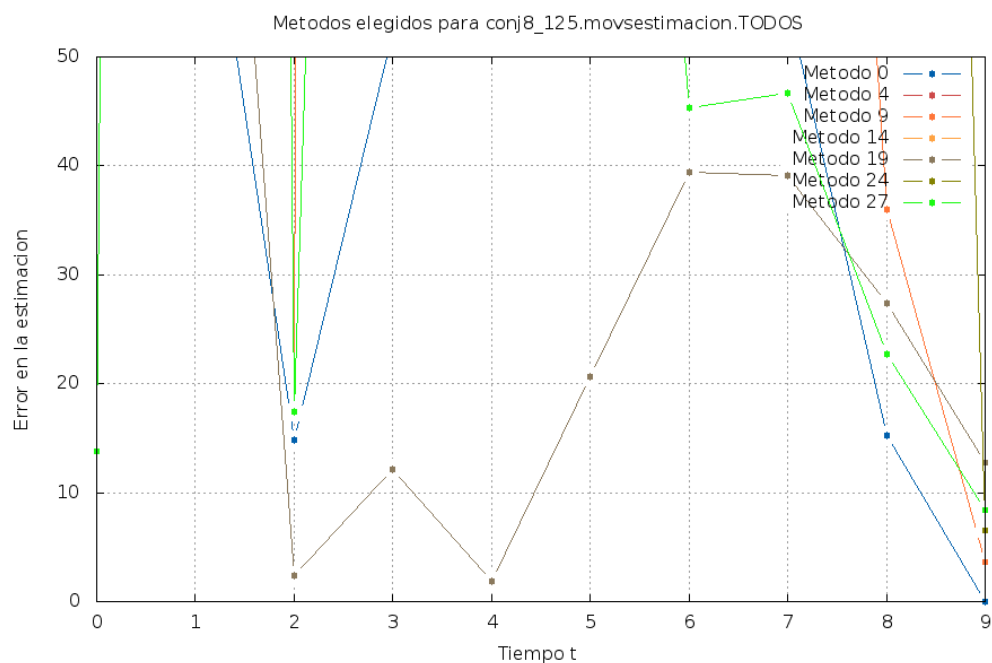
Utilizando conj8.tiro

Figura 14: Estimacion de tiro conj8

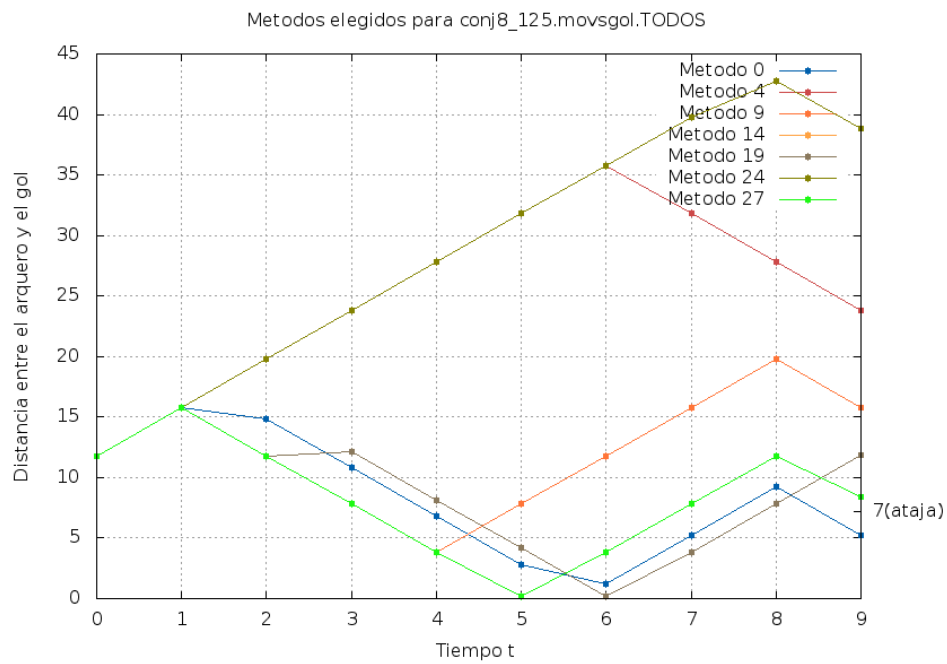


Figura 15: MOVSGOL de tiro conj8

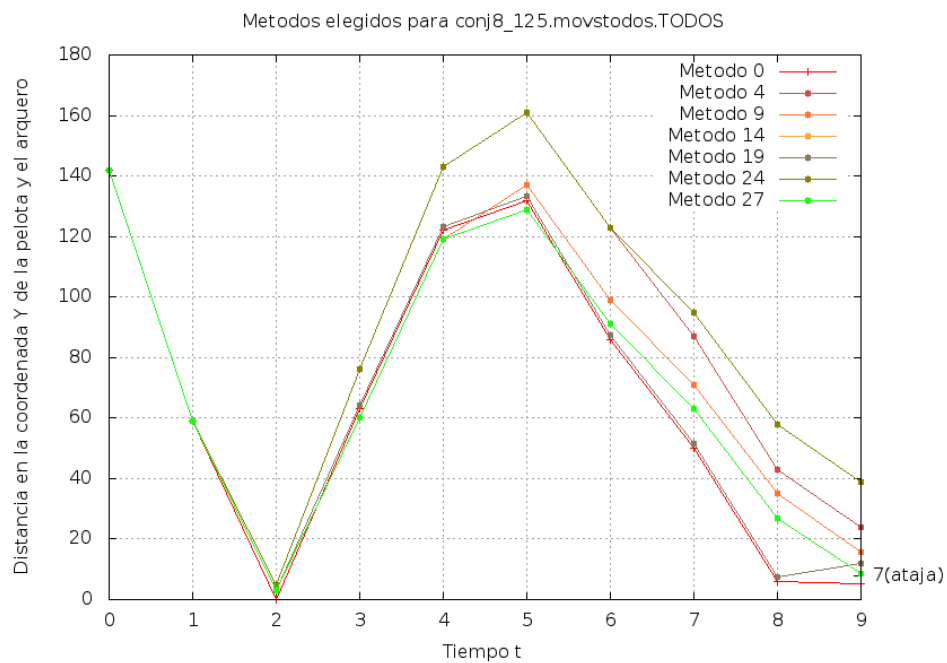


Figura 16: MOVSTODOS de tiro conj8

4.3.1. Conclusiones

4.4. Experimentacion con Exoticos

Utilizando ex5.tiro

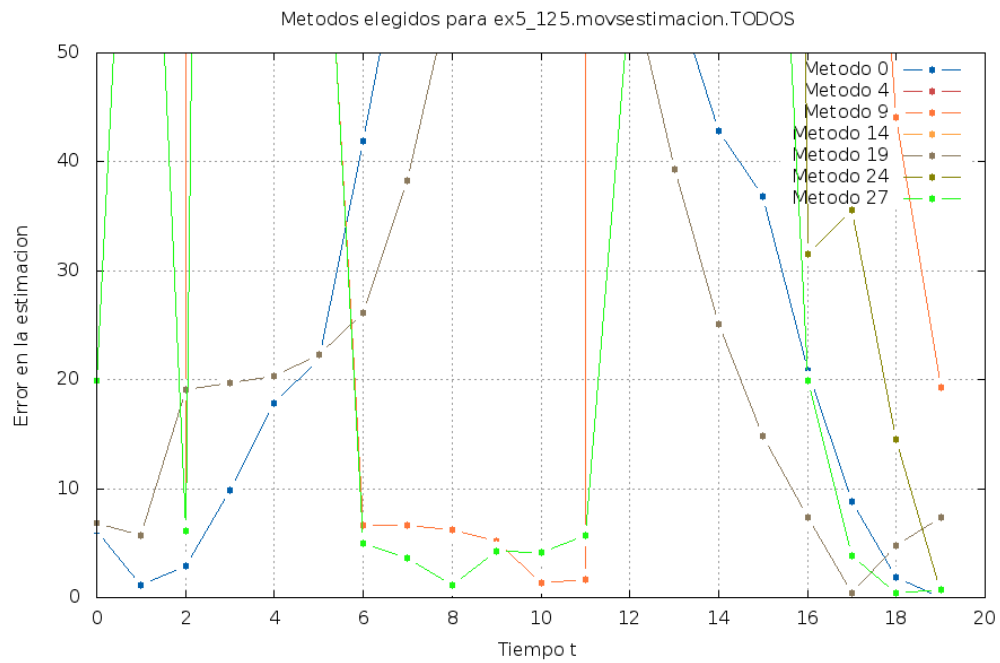


Figura 17: Estimacion de tiro ex5

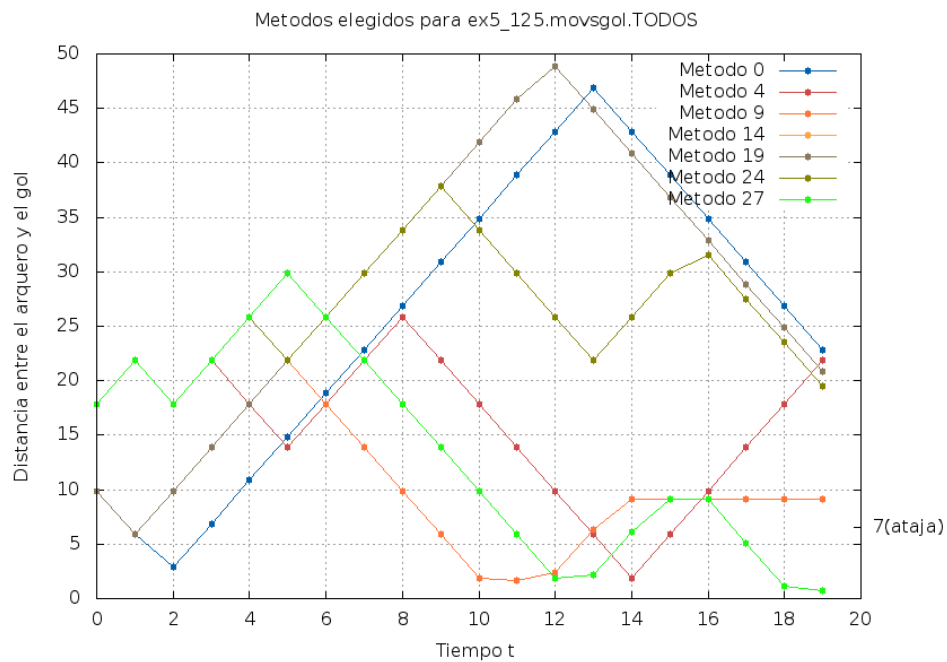


Figura 18: MOVSGOL de tiro ex5

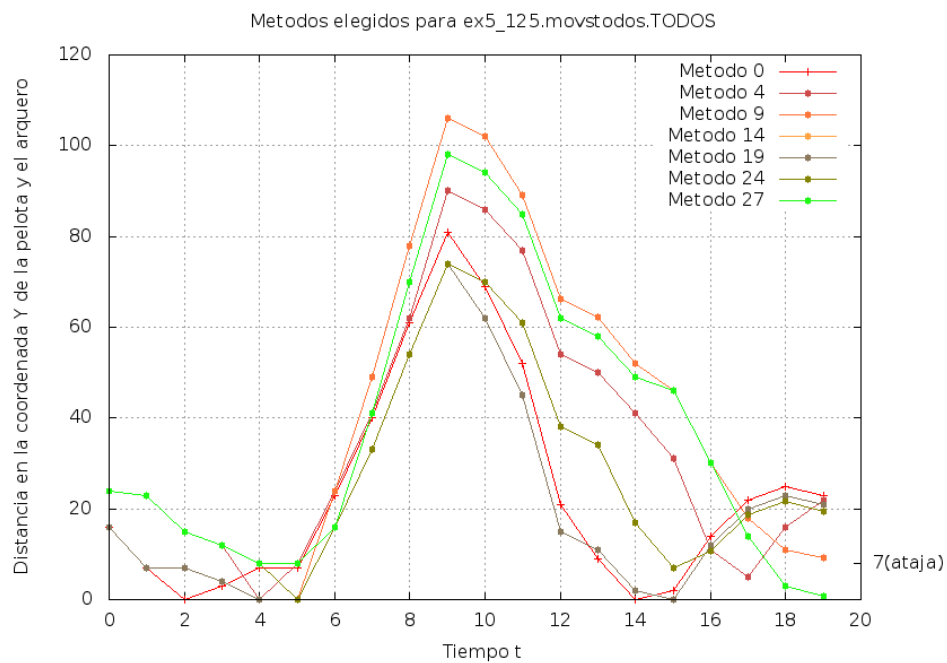


Figura 19: MOVSTODOS de tiro ex5

4.4.1. Conclusiones

4.5. Conclusiones Finales

5. Apéndice

5.1. Generador de Tests

5.1.1. Generador desde cero

Para generar Tests realizamos un algoritmo en Python en el cual generamos instancias lineales tomando como parametros el μ , la posición del arquero y la ubicación de los arcos. De la misma forma generamos instancias polinómicas. Para ambos casos tuvimos en cuenta el punto inicial, es decir donde empieza la trayectoria de la pelota y el punto final, es decir la posición de la pelota dentro del arco.

Para tests mas complejos utilizamos un script en C++ en donde para generar las curvas utilizamos la función spline de la librería boots con la cual le agregamos los puntos por donde queríamos que pase la pelota e interpolábamos para conseguir una curva que pase por ese lugar tomando esa curva como el tests.

Ambos generadores pueden encontrarse en `/visualizador/`.

5.1.2. Agregar la posición específica con $x=125$

Parados en la carpeta donde se encuentra el tp (donde se encuentra el archivo `run.py`)

```
$ python generar_125.py < input tiro >
```

Donde INPUT TIRO puede ser

- Path a una carpeta: modifica los **.tiro** de esa carpeta y los guarda en `/visualizador/todos_125`
- Path a un archivo: ídem pero sólo para un archivo.

5.2. Método de compilación

5.2.1. Forma manual

Parados en la carpeta `/src` del proyecto ejecutar

```
$ make
```

De esta forma se limpia y compila. Para compilar por separado se puede hacer: **make data.o**, **make functions.o**, **make Matrix.o**, **make main.o**. O también se puede borrar haciendo **make clean**. Por defecto al ejecutar **make** el nombre del ejecutable es **yoAtajo**

5.2.2. Forma automatizada generando los archivos `.arq`

Parados en la carpeta donde se encuentra el tp (donde se encuentra el archivo `run.py`)

```
$ python run.py < input tiro > < metodo > < velocidad >
```

Donde METODO puede ser cualquiera de los 40 especificados en secciones anteriores

Donde INPUT TIRO puede ser

- Path a una carpeta: ejecuta todos los tests que contiene dicha carpeta.
- Path a un archivo: ejecuta el tests que corresponde a esta ruta.

Donde VELOCIDAD puede ser

- 0: Para correr el visualizador rapido (no se muestra el tiro)
- 1: Para correr el visualizador en modo lento (se muestra el tiro)

5.2.3. Forma automatizada sin generarlos archivos .arq

Parados en la carpeta donde se encuentra el tp (donde se encuentra el archivo run.py)

```
$ python runStatistics.py < input tiro >
```

Donde INPUT TIRO puede ser

- Path a una carpeta: ejecuta todos los tests que contiene dicha carpeta.
- Path a un archivo: ejecuta el tests que corresponde a esta ruta.

Es muy parecido a run.py sólo que no genera los **.arq** resultantes a cada **.tiro**. Como cada vez que se ejecuta el código resolvente del problema se generan las estadísticas, al correr este script se generan las estadísticas sin ensuciar la carpeta que en la que se encuentran los **.tiro** con los **.arq**.

5.3. Generadores de estadísticas

En la carpeta /src/estadisticas se pueden encontrar muchos scripts para generar analizar los gráficos. Los más automáticos son los siguientes:

- juntar_todo.py: une todos los métodos para un tiro especificado en el comando en un archivo *.TODO guardado en /src/estadisticas/todos con formato csv.
- juntar_y_graficar_todos_los_archivos.py: llama a juntar_todo.py para cada método y los grafica.
- juntar_y_graficar_todos_archivos_elegidos.py: parecido al anterior pero sólo para los métodos elegidos en la sección Tests.

Además, tenemos una carpeta para cada método que incluye la información de este para los 3 criterios explicados en la sección Tests (**.movsgol**, **.movstodos**, **.movsestimación**), una carpeta /todos que incluye el **.TODO** para cada tiro, una carpeta /graficos que contiene a todos los gráficos, una carpeta /tabla que contiene la tabla también mencionada en la sección Tests y una carpeta /tabla/graficos que contiene a los gráficos sólo de los métodos elegidos finales.

5.4. Equipo de pruebas

5.5. Referencias bibliográficas

Referencias

- [1] Richard L. Burden and J. Douglas Faires *Numerical Analysis*. 2005.