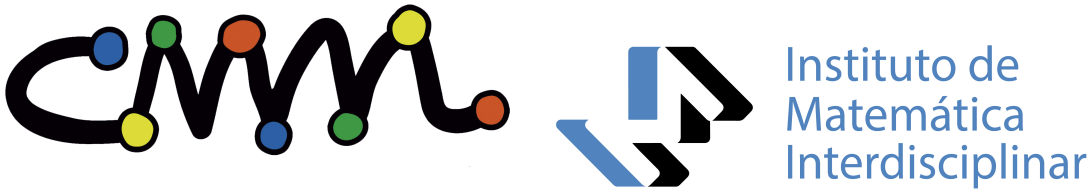
V Concurso de Modelización Matemática (CMM-IMI 2022)



SORTEO UEFA PARA LA FASE SEGUNDA DE LA CHAMPIONS

Análisis crítico y propuestas de mejora

Código del grupo: CMM2022-807797

Índice

1. Introducción

El sorteo de la UEFA para octavos de final en la Champions lleva asociada cierta polémica debido a que el método empleado no es completamente equitativo, esta polémica ha sido más marcada este año debido a la decisión de la organización de repetir el sorteo debido a ciertos errores que se cometieron durante el proceso.

En esta V edición del concurso de modelización matemática del IMI se nos presenta un interesante problema: optimizar este, de manera que sea más equitativo.

Este es un problema más complicado de lo que aparenta, para el cual se emplearán herramientas de diversas ramas de las matemáticas. Cabe destacar que, en el propio enunciado del concurso, se adjuntó un documento de los creadores de este en el que se presentaba un trabajo extenso sobre este asunto (demasiado quizás para ser un concurso). Este trabajo se puede encontrar en https://meteo.unican.es/temp/castie/informe7.pdf y se empleará para comparar resultados y tratar de mejorar los métodos.

1.1. Descripción del problema

Las condiciones específicas de este sorteo son las siguientes:

Se parte de los 8 campeones y los 8 subcampeones de la fase de grupos y se deben elegir de este conjunto las parejas que se enfrentarán a continuación, para ello se imponen ciertas restricciones:

- Se debe enfrentar un campeón con un subcampeón.
- No se pueden enfrentar dos equipos que hayan jugado en el mismo grupo de la fase anterior.
- No pueden enfrentarse dos equipos del mismo país.

El sorteo actual de la UEFA sigue el siguiente esquema:

1. Se elige al azar un campeón.
2. Se consideran los posibles contrincantes de este. Un programa de ordenador comprueba que no se produzca un bloqueo del sorteo si sale alguno de ellos, si ocurre para alguno, se elimina.
3. Se introducen bolas con el nombre de sus potenciales contrincantes y se elige uno al azar.
4. Se repite el ciclo teniendo en cuenta sólo los restantes participantes.

Este proceso, junto con las restricciones, hace que las probabilidades de que cierto equipo A se enfrente con B o con C no sean equitativas, de hecho, será imposible que lo sean. La complicación está en generar un método que reduzca las disparidades.

2. Modelización

Se plantea modelizar el resultado de un sorteo mediante una matriz 8×8. Para ello, se necesita dividir los equipos a sortear en dos grupos: los equipos que han quedado primeros en la fase de grupos (P) y los equipos que han quedado segundos en la fase de grupos (S). Enumeramos los equipos dentro de cada conjunto por el grupo en el que han jugado la fase de grupos para evitar cruces entre dos equipos del mismo.

Se marca con un 1 la casilla (i, j) si el sorteo determina que el equipo número i del grupo P juega contra el equipo número j del grupo S. Se marcará con un 0 en caso contrario.

A continuación, se presentan definiciones formales de herramientas que se emplearán en el trabajo

2.1. Formalismo matemático

En esta sección se aborda la modelización matemática del problema, para lo cual nos será necesarias algunas definiciones básicas.

Definición 1. Un espacio de probabilidad es una terna $(\Omega, \mathcal{F}, \mu)$ donde:

- Ω es el espacio muestral: Conjunto de todos los resultados posibles de un experimento.
- $\mathcal{F}$ es el conjunto de sucesos: conjunto de sub-conjuntos de Ω con estructura de σ-álgebra.
- $\mu : \mathcal{F} \to [0, 1]$ es la función de probabilidad.

Para el problema en cuestión, el espacio muestral es:

$$\Omega = \Big\{ A \in \{0,1\}^{8\times 8} | \sum_{i=1}^{8} (A)_{ij} = 1 \, \forall j \, \wedge \sum_{j=1}^{8} (A)_{ij} = 1 \, \forall i \, \wedge \, (A)_{ii} = 0 \, \forall i \, \wedge \\ (A)_{ii} = 0 \, \forall i \, \wedge \, (A)_{kl} = 0 \, \forall (k,l) \in R \Big\} \quad (1)$$

donde $R \subset \{1, 2, ..., 8\} \times \{1, 2, ..., 8\}$ es el conjunto posiciones prohibidas por las restricciones que se impondrán en el sorteo debido a que dos equipos de la misma liga nacional no pueden jugar

en contra.

Para futuros usos, se define el conjunto $\Omega_{ij} = \{A \in \Omega | (A)_{ij} = 1\}$.

Se define el conjunto de sucesos como $\mathcal{F} = \mathcal{P}(\Omega)$ donde $\mathcal{P}(\Omega)$ son las partes del espacio muestral [1].

Proposición 1. Dado el espacio de probabilidad $(\Omega, \mathcal{F}, \mu)$ con el espacio muestral Ω anterior y $\mathcal{F} = \mathcal{P}(\mathcal{F})$, μ viene caracterizado por una n-tupla $(m_1, ..., m_n) \in [0,1]^n$

Demostración. Tal y como se ha definido $\mathcal{F}$, la colección $A_i \in \mathcal{F} \quad i = 1, 2, ..., n$ $(n = |\Omega|)$ finita y son todos sucesos incompatibles, por tanto, se tiene que

$$P\left(\bigcup_{i=1}^{n} A_i\right) = \sum_{i=1}^{n} P(A_i)$$

Por tanto, la función de probabilidad μ puede caracterizarse a partir de las probabilidades de cada suceso incompatible $\mu(A_i)$, ya que, cualquier $\Omega' \in \mathcal{F}$ se puede escribir como $\Omega' = \bigcup_{i=1}^{t} A_{\sigma(t)}$ para alguna $\sigma : \mathbb{N} \to \mathbb{N}$. Es decir, se puede definir la función de probabilidad μ constructivamente de la siguiente manera:

$$\mu : \mathcal{F} \to [0,1]$$
$$\Omega' = \bigcup_{i=1}^{t} A_{\sigma(t)} \to \sum_{i=1} m_{\sigma(t)}$$

□

Esta función de probabilidad serán de suma importancia en el desarrollo del trabajo. De hecho, más adelante, se verá que las diferentes funciones de probabilidad que nos dan un sorteo igual de equitativo se encontrarán en un poliedro (más adelante se realiza la correspondencia con más cuidado). Para ello, definamos lo que es un poliedro.

Definición 2. Un conjunto $S \subset \mathbb{R}^n$ es un poliedro si es la intersección de un número finito de semiespacios cerrados, es decir, si:

$$S = \{\hat{x} : \hat{p_i}^T \hat{x} \leq \alpha_i, i = 1, ..., m\}$$

donde $\hat{p_i}$ son vectores no nules y α_i son escalares, $i = 1, ..., m$.

En particular, dada una matriz A $m \times n$ de rango m y $\hat{b}$ vector de m componentes, el siguiente conjunto es un poliedro:

$$S = \{\hat{x} : A\hat{x} = b, \hat{x} \geq 0\}$$

[1] Es inmediato comprobar que $\mathcal{P}(\Omega)$ tiene estructura de σ-álgebra

Definición 3. Se llama a P_{ij} a la probabilidad de que el equipo número i del grupo P se enfrente contra el equipo j del grupo S.

Proposición 2. Se cumple que:

- $\sum_{i=1}^{N} P_{ij} = 1 \quad \forall j$
- $\sum_{j=1}^{N} P_{ij} = 1 \quad \forall i$
- $P_{ii} = 0 \quad \forall i$
- $P_{kl} = 0 \quad \forall (k,l) \in R$

Demostración. Las primeras dos propiedades se demuestran viendo que $p_i := P_{ij} \quad \forall j \; p_j := P_{ij} \quad \forall i$ forman una función de probabilidad. Esto se ve del hecho de que el equipo número i del grupo P debe jugar con alguno del grupo S y viceversa, por tanto, la suma de las probabilidades p_i ha de ser 1. Las dos últimas propiedades se siguen del hecho de que no pueden jugar dos equipos que han quedado primeros en la fase de grupos o segundos. Además, equipos de la misma liga nacional tampoco pueden enfrentarse. □

Definición 4. Una *tabla* es una matriz $A \in [0,1]^{N \times N}$ tal que $(A)_{ij} = P_{ij}$.

Más adelante, se relacionará el espacio de probabilidad $(\Omega, \mathcal{F}, \mu)$ con diferentes *tablas*. El objetivo de este trabajo tratará de obtener la *tabla* más equitativa posible. En la siguiente sección se discuten diferentes nociones matemáticas de equidad.

2.2. Criterios de equidad

Para elegir nuestra función de probabilidad óptima debemos definir lo que significa ser "óptimo". Aunque se pueden plantear diversas[2], nosotros trabajaremos esencialmente con tres definidas a partir de las probabilidades P_{ij}:

1. Desinformación máxima. En 1948, Claude Shannon introduce el concepto de entropía en el contexto de la Teoría de la Información [1]. Esta magnitud cuantifica el grado de incertidumbre que contienen los posibles resultados de una variable aleatoria. Partiendo de la definición de entropía, se construye una magnitud que se llamará desinformación asociada a una *tabla* , que se empleará como criterio de equidad. Dada una *tabla* con su $\hat{p}$ asociada, la función desinformación es:

$$S(\hat{p}) = -\sum_{i,j=1}^{N} P_{ij} log(P_{ij})$$

Véase la Sección A del Anexo para una formulación rigurosa de la desinformación S. La desinformación, tal y cómo se ha definido aquí, es máxima cuando todos los P_{ij} son iguales.

[2]Sin ir más lejos, combinaciones de las que daremos nosotros darían como resultado otro criterio de equidad.

2. Diferencia entre el máximo y el mínimo. Aquí comparamos las probabilidades extremas de las parejas y tratamos de minimizarla.

$$f_m(P_{ij}) = \text{máx}\{P_{ij}\} - \text{mín}\{P_{ij}\} \tag{2}$$

De igual manera, esta función será nula si y sólo si las parejas son todas equiprobables.

3. Varianza. En este método se minimiza la varianza, que también será nula si y sólo si son todas iguales y es una medida de dispersión estadística estándar.

$$Var(\{P_{ij}\}) = \frac{1}{N}\sum_{i,j}(P_{ij} - \bar{P})^2 \tag{3}$$

donde $\bar{P} = \frac{1}{N}\sum_{i,j} P_{ij}$ es la media de las probabilidades de pareja.

Se discute ahora la exixtencia y unicidad de las soluciones:

Proposición 3. Todos los criterios tienen solución.

Demostración. 1. S es una función continua en un dominio compacto, luego tiene máximo absoluto. 2. y 3. Son funciones acotadas inferiormente por 0 en un compacto, luego tienen mínimo. □

Proposición 4. Ninguna de las condiciones da una *tabla* solución única en general.

Demostración. Tomemos una tabla tal que los elementos (i,j), (i,k), (l,j) e (l,k) están permitidos y la solución óptima les asigna valores P_{ij}, P_{ik}, P_{lj}, $P_{lk} \in (0,1)$ Entonces basta con tomar $\delta < \text{mín}\{P_{ij}, P_{ik}, P_{lj}, P_{lk}, 1-P_{ij}, 1-P_{ik}, 1-P_{lj}, 1-P_{lk},\}$ y tomar una nueva *tabla* con $P'_{ij} = P_{ij} - \delta$, $P'_{lj} = P_{lj} + \delta$, $P'_{ik} = P_{ik} + \delta$, $P'_{lk} = P_{lk} - \delta$. Esta nueva configuración asignará igual varianza y entropía a las configuraciones.

Para el criterio 2, se debe pedir además que estos elementos no sean máximos o mínimos y que δ sea menor que las diferencias de los elementos con el máximo y el mínimo. Haciendo la transformación cambiará la tabla, pero los elementos máximo y mínimo se mantendrán, haciendo que el valor de nuestro principio de equidad se mantenga constante. □

Como comentario, decir que aunque en general no sea única, pueden darse casos concretos en los que sí lo sea.

2.2.1. Explicación para el público general

Cómo estas definiciones son formales, es necesario explicarlas a los miembros de la UEFA y aficionados al fútbol que no tengan formación matemática. Para ello se proponen las siguientes explicaciones.

1. Desinformación máxima Un sorteo equitativo es aquel que mayor incertidumbre conlleva, es decir, aquel cuyo resultado más sorprende. Estamos todos de acuerdo que si la probabilidad de que se de un enfrentamiento dado es 1, el resultado del sorteo no supone ninguna sorpresa para nadie; sin embargo, si la probabilidad de que un equipo se enfrente contra el resto de equipos es la misma, el resultado generará gran expectación en el público. En otras palabras, un sorteo es más equitativo que otro si el público puede pronosticar peor el resultado (gran interés por parte de las casas de apuestas). Este criterio trata de minimizar la información que podemos tener sobre el resultado del sorteo antes de realizarlo.

2. Diferencia entre el máximo y el mínimo En este caso tratamos de hacer que la diferencia entre la probabilidad de pareja más probable y la pareja menos probable se haga lo más pequeña posible. Minimizando esta diferencia, el máximo y el mínimo hacen de "sandwich" por lo que el resto de probabilidades se diferencian menos.

3. Varianza En este caso tratamos de hacer que las probabilidades de las parejas se alejen lo menos posible de su valor medio, intentando así que sean parecidos.

3. Obtención de soluciones

Se busca hallar la *tabla* más equitativa posible; como se ha visto, hay varias nociones de equidad posibles. Para ello, distinguimos dos *modus operandi*:

- Obtener una *tabla* equitativa manipulando las probabilidades P_{ij} directamente.
- Obtener una *tabla* equitativa a partir del espacio de probabilidad $(\Omega, \mathcal{F}, \mu)$

3.1. Multiplicadores de Lagrange

La primera manera de proceder consiste en tratar de hallar una *tabla* que sea lo más equitativa posible. Para ello, se emplea el método de los multiplicadores de Lagrange. Este procedimiento, permite hallar puntos extremales relativos de funciones de varias variables sujeto a restricciones.

Definimos la función $h(P_{ij}, \lambda)$:

$$h(P_{ij}, \lambda) = f(P_{ij}) + \lambda(\sum_{i \neq j}^{N} P_{ij} - 1)$$

donde $f(P_{ij})$, diferenciable, puede tomar varias expresiones en función del criterio de equidad que se elija:

- Desinformación: $f(P_{ij}) = \sum_{i \neq j \wedge (i,j) \notin R}^{N} P_{ij}\, log(P_{ij})$
- Diferencia máximo mínimo: Para este criterio no se puede usar este método porque la función $f(P_{ij}) = max_{i,j,i',j'}\{|P_{i'j'} - P_{ij}|\}$ no es diferenciable
- Varianza: $f(P_{ij}) = \sum_{i \neq j \wedge (i,j) \notin R}^{N} (P_{ij} - P_{media})^2$

El método cosiste en obtener las derivadas parciales respecto a las variables $P_{ij}, \lambda \quad i, j = 1, ..., N$ e igualarlas a 0. Los resultados son puntos extremos de la función elegida.

Este método se he aplicado usando el paquete de cálculo simbólico Sympy de Python (Véase el programa empleado para minimizar la desinformación en el Anexo).

Este método proporciona buenos resultados y un tiempo de cálculo razonable (al menos, para $N \leq 8$). Sin embargo, para ejecutar el algoritmo del sorteo se necesita describir la *tabla* obtenida mediante una función de probabilidad válida μ para el espacio de probabilidad $(\Omega, \mathcal{F}, \mu)$.

La obtención de una μ a partir de una *tabla* supone resolver la ecuación $A\hat{\mu} = \hat{p}$ donde $\hat{\mu} = (m_1, ..., m_n)^T$ (recordemos que $n = |\Omega|$), $\hat{p} = (P_{11}, P_{12}, ..., P_{NN})^T$ y

$$A = \begin{pmatrix} (A_1)_{11} & (A_2)_{11} & \cdots & (A_n)_{11} \\ (A_1)_{12} & (A_2)_{12} & \cdots & (A_n)_{12} \\ \vdots & \vdots & \ddots & \vdots \\ (A_1)_{NN} & (A_2)_{NN} & \cdots & (A_n)_{NN} \end{pmatrix} \tag{4}$$

con $A_i \in \mathcal{F}$. A la matriz A lo llamaremos *matriz representativa.*

Sin embargo, puede darse que se obtenga como resultado un sistema incompatible. Esto se debe a que, al resolver el problema de optimización sólo atendiendo a los valores de P_{ij} se puede llegar a una solución que no se corresponda con ninguna función de probabilidad real. En concreto, en el caso de este año, el rango de la matriz representativa es de $rg(A) = 35$, pero si consideramos el sistema de ecuaciones $A\hat{\mu} = \hat{p}$ dado por el vector $\hat{p}$ obtenido al optimizar con el criterio 2 de manera numérica se obtiene que $rg(A|\hat{p}) = 36$, por tanto no tiene solución. De esta manera este método es poco robusto y en general no nos podrá llevar a una solución completa.

3.2. Método Montecarlo de Variación de μ

La segunda forma de proceder consiste en hallar una *tabla* equitativa a partir de $\hat{\mu} = (m_1, ..., m_n)$ que, recordemos, caracteriza unívocamente la función de probabilidad del espacio de probabilidad definido anteriormente $(\Omega, \mathcal{F}, \mu)$.

Para enfrentar este problema hace falta hallar un vector $\hat{\mu}$ tal que al calcular la *tabla* , se cumpla el criterio de equidad elegido. Es decir, estamos en un problema de optimización con un miles de variables (Concretamente 4781 para el sorteo de este año). Su resolución se hace difícil mediante métodos de multiplicadores de Lagrange, o incluso con métodos numéricos de descenso de gradiente o basados en simplex[3]. De esta manera, se propone el uso de un método Montecarlo de optimización muy simple, pero que ha resultado ser muy eficiente, permitiendo resolver en problema en pocos minutos (entre 2 y 5).

El algoritmo es una simplificación propia del algoritmo Metrópolis [3] y consta de los siguientes puntos:

1. Se parte de una aproximación de la solución. En nuestro caso se usa la distribución uniforme.
2. Se eligen dos números aleatorios k, l diferentes entre 1 y el número de matrices de representación posibles (4781 este año).
3. Se propone una nueva solución tal que $\hat{\mu}'_k = \hat{\mu}_k + \Delta\hat{\mu}$ y $\hat{\mu}'_l = \hat{\mu}_l - \Delta\hat{\mu}$. Esta transformación mantiene la propiedad de normalización de la función de probabilidad.
4. Se comprueba si los nuevos valores son no negativos. Si alguno es negativo, se vuelve al paso 2.
5. Se calcula la *tabla* correspondiente a la nueva densidad de probabilidad $\hat{\mu}'$ mediante $\hat{p}' = A\hat{\mu}'$.
6. Se evalúa la función del criterio considerado: $f(\hat{p}')$.
7. Si $f(\hat{p}') \leq f(\hat{p})$ se acepta el cambio (para el caso de querer maximizar: $\geq$). Si no se cumple, el cambio se acepta con una probabilidad P_{change}.
8. Volver al punto 2.

Los valores usados para los parámetros del algoritmo ha sido: $\Delta\hat{\mu} = 0{,}000001$ y $P_{\text{change}} = 0{,}005$ o $P_{\text{change}} = 0{,}01$.

Merece la pena comentar que aceptar un cambio que, en principio, no es mejor que la situación anterior con una pequeña probabilidad P_{change} resulta en una aceleración considerable de la convergencia del algoritmo. De manera cualitativa, esto se debe a que la aceptación de estos

[3]Según el informe del enunciado con `MATLAB` se puede llegar a resolver, en este caso se probó con una función de `python` y el método no convergía. Igualmente, su solución tardaba horas y no disponemos de mucho tiempo en nuestro caso.

cambios puede ayudar a salir de mínimos locales y a producir cambios después que lleven a soluciones mejores.

Una vez alcanzada una solución, cabe preguntarse si esta $\hat{\mu}$ obtenida es única. Es decir, si no existe otra densidad de probabilidad $\hat{\mu}$ tal que su *tabla* asociada sea igual. La respuesta a esta pregunta es que no y se detalla en la siguiente proposición.

Proposición 5. Sea A la matriz $l \times n$ representativa de un sorteo, con rango m y sea $\hat{\mu}$ la n-tupla que caracteriza la densidad de probabilidad obtenida como solución en un proceso de optimización y m¡n. Sea $\hat{p}$ las probabilidades de la *tabla* calculadas a partir de $\hat{\mu}$. Entonces existe otra $\hat{\mu}'$ que da las mismas probabilidades $\hat{p}$. Es más, el conjunto de posibles densidades de probabilidad genera un poliedro.

Demostración. Por construcción, el sistema $Ax = \hat{p}$ es compatible, pues $x = \hat{\mu}$ es una solución. Así, por el Teorema de Rouche-Frobenius $rg(A) = rg(A|\hat{p})$. Sin embargo, $rg(A) < n$. De esta forma el sistema será incompatible determinado, por lo que habrá más soluciones. Si a partir de A definimos una matriz A' de rango m máximo obtenida de eliminar filas linealmente dependientes del sistema, entonces $S = \{\hat{x} : A\hat{x} = \hat{p}, \hat{x} \geq 0\}$ es un poliedro y sus elementos son soluciones. □

Se plantea por tanto otro problema que en este documento no se ha podido llegar a abordar: la elección de la densidad de probabilidad mejor dentro del poliedro. De manera experimental, a partir de las soluciones concretas obtenidas, hemos podido ver que no son demasiado diferentes de la uniforme, sin embargo podría darse algún caso particular, en el que alguna de las posibles soluciones asignara probabilidad cero a alguna combinación, lo cual no estaría acorde con la idea de equidad, siendo necesario ampliar la definición matemática de "equidad". Se podría explorar la opción de elegir una densidad de probabilidad óptima explotando las propiedades de un poliedro y sus puntos extremos.

4. Tablas y análisis

Cada uno de los principios de equidad y cada uno de los métodos de búsqueda de soluciones nos aporta diferentes soluciones. En esta sección se deja aparte la parte más formal de las matemáticas y se presentan algunos resultados.

4.1. Campeonato 2021-2022

Grupos	*Ganadores*	*Segundos*
A	Juventus (ITA)	PSG (FRA)
B	Lille (FRA)	Atlético (ESP)
C	Manchester United (ENG)	Sporting Lisboa (POR)
D	Bayern (GER)	Inter de Milán (ITA)
E	Real Madrid (ESP)	Benfica (POR)
F	Ajax (NED)	Villareal (ESP)
G	Liverpool (ENG)	Salzburgo (GER)
H	Manchester city (ENG)	Chelsea (ENG)

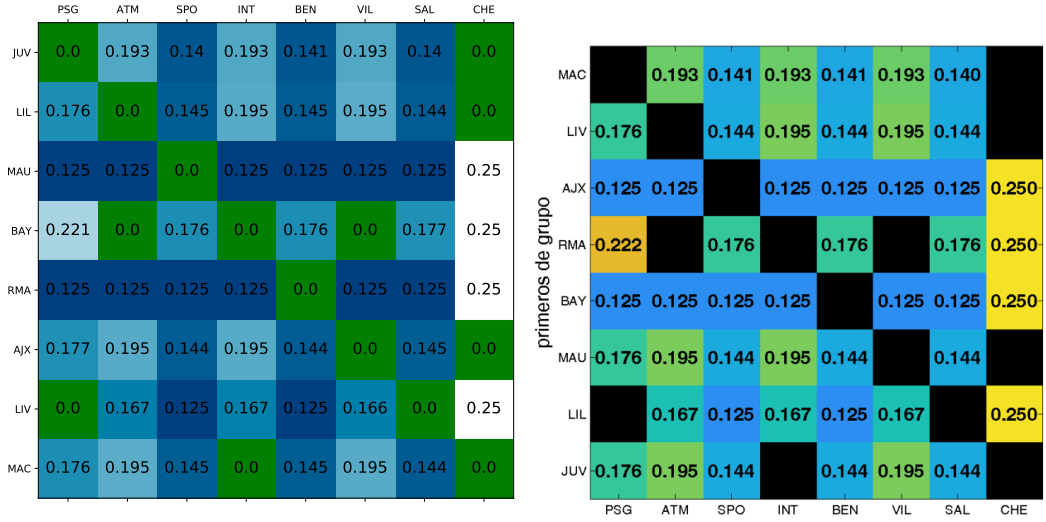
Tabla 1: Primeros y segundos clasificados en la fase de grupos de la UEFA 2021-2022.

La Tabla 1 recoge los primeros y segundos clasificados de la fase de grupos de la UEFA de la temporada 2021-2022. Se puede comprobar en las Figuras 1 y 3, que el método de la varianza y el de la diferencia entre el máximo y el mínimo arrojan los mismos resultados que en el informe de los creadores del concurso, salvo pequeñas variaciones en la tercera cifra decimal (notar que cómo se ha demostrado, las *tablas* solución no son únicas). Se obtiene $(f_m)_{\min} = 0{,}125$ y $Var_{\min} = 0{,}0012$.

En la figura 2 se puede ver el cálculo obtenido con el criterio de la desinformación con un método Montecarlo y con multiplicadores de Lagrange: ambos coinciden exactamente y dan un valor $S_{\min} = 14,32$.

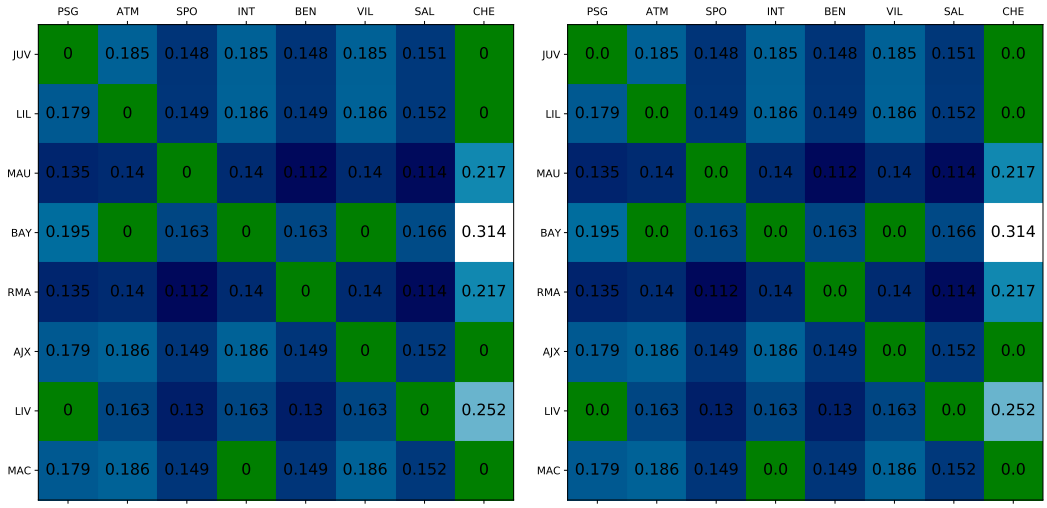
Además de las *tablas* de probabilidad condicionada, el método que varía el valor de las funciones de densidad de probabilidad nos da como solución esta función, para los tres criterios se pueden ver en la figura 4. Se puede ver que la función relacionada con el criterio de la desinformación da la función de probabilidad más parecida a la uniforme y la relacionada con el máximo y mínimo da la más irregular.

Comparando los resultados obtenidos, parece que el criterio 2 se ajusta más a la idea intuitiva de equidad, estamos de acuerdo por tanto, con esa parte del informe del enunciado.



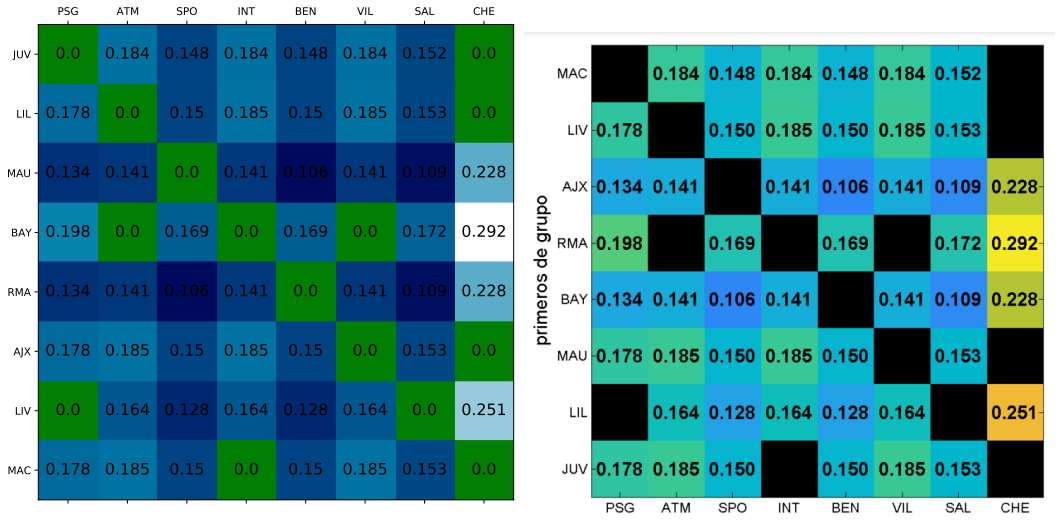
(a) Método Montecarlo. (b) Resultados informe.

Figura 1: Comparación de tablas de nuestra solución y la dada en el informe del enunciado para el criterio 2. Notar que por error al crear las figuras las etiquetas en el eje y están en orden inverso.



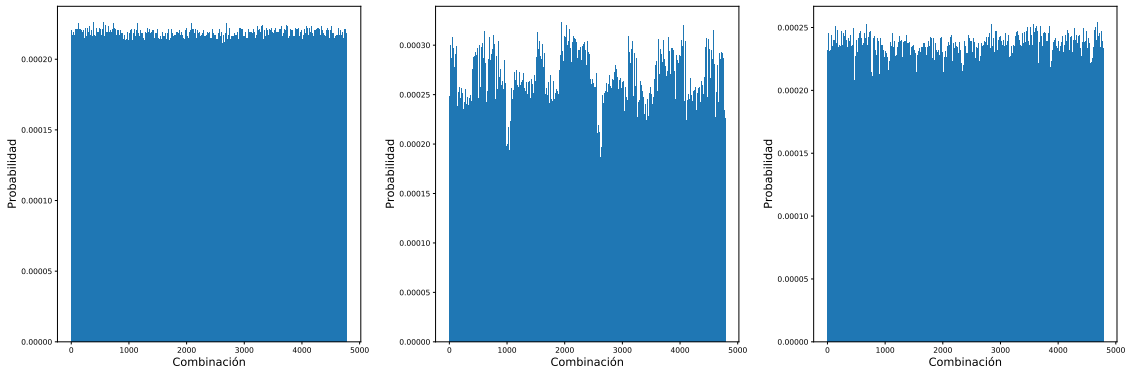
(a) Multiplicadores de Lagrange. (b) Método Montecarlo.

Figura 2: Comparación de dos métodos para el cálculo con el criterio 1.



(a) Método Montecarlo. (b) Resultados informe.

Figura 3: Comparación de dos métodos para el cálculo con el criterio 1.



(a) Criterio 1 (b) Criterio 2. (c) Criterio 3.

Figura 4: Funciones de probabilidad obtenidos por el método de variación de μ

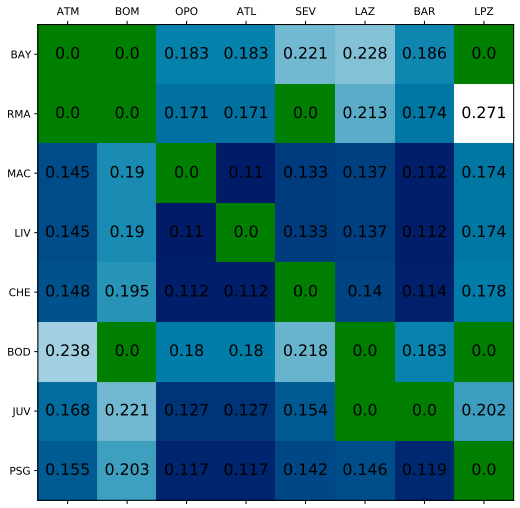
4.2. Campeonato 2020-2021

La Tabla 2 muestra los resultados de la fase de grupos de la UEFA del año 2020-2021.

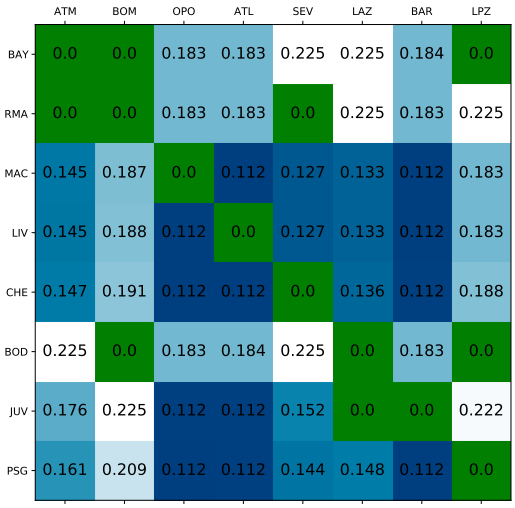
Grupos	*Primeros*	*Segundos*
A	Bayern (GER)	Atlético (ESP)
B	Real Madrid (ESP)	Borussia Mönchengladbach (GER)
C	Manchester City (ENG)	Oporto (POR)
D	Liverpool (ENG	Atalanta (ITA)
E	Chelsea (ENG)	Sevilla (ESP)
F	Borussia Dortmund (GER)	Lazio (ITA)
G	Juventus (ITA)	FC Barcelona (ESP)
H	Paris (FRA)	Leipzig (GER)

Tabla 2: Primeros y segundos clasificados en la fase de grupos de la UEFA 2020-2021.

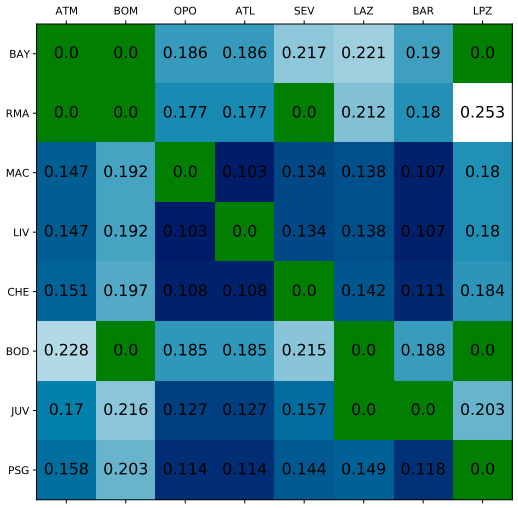
Se aplican en el caso del sorteo del año pasado los principios de equidad definidos, para resolver se emplea el método Montecarlo. En este caso la matriz de restricciones es diferente y el número de combinaciones posibles pasa de 4781 a 4572. Los resultados se pueden ver en la figura 5. Parece que en este caso los métodos arrojan resultados más parecidos entre ellas, sin probabilidades tan grandes, aún así, el criterio 2 parece adecuarse mejor a la idea intuitiva de equidad, disminuyendo las desigualdades,



(a) Criterio 1



(b) Criterio 2



(c) Criterio 3

Figura 5: Resultados del sorteo para el campeonato 2020-2021.

5. Sorteo

En esta sección se discutirán procedimientos y algoritmos para la realización del sorteo en la gala de la UEFA. Recordemos que el objetivo es diseñar un sorteo que reproduzca las probabilidades dadas por la *tabla* que considereremos más equitativa.

Por un lado, uno podría pensar que la *tabla* que se requiere reproducir es suficiente para diseñar el sorteo que lo reproduzca; sin embargo, para los métodos que se van a proponer, no es suficiente.

Por otro lado, se aprovecha que se ha obtenido la *tabla* [4] que se desea minimizar a partir de $\hat{\mu} = (m_1, ..., m_n)$ para usarlo. Tal y como se ha visto en la Sección 2.1, $S = \{\hat{\mu} : A\hat{\mu} = \hat{p}, \hat{\mu} \geq 0\}$ es un poliedro, donde A es la matriz representativa.

Una vez tomado un punto del poliedro, se podría intentar, mediante un método numérico, obtener otro punto de S (que daría la misma *tabla* por estar en el poliedro S) que presente mejores características como puede ser una baja desinformación definida como $D = \sum_{i,j=1}^{N} P_{ij}\, log(P_{ij})$. Sin embargo, esto se propone como futuras posibles mejoras, ya que, en este trabajo no se ha implementado.

Se proponen diferentes dinámicas de sorteo basados en $\hat{\mu}$:

5.1. Sorteo directo

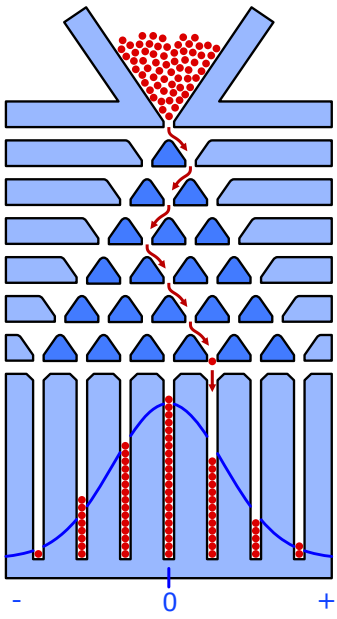
Se trata de la forma más de directa de hacer el sorteo basado en una densidad de probabilidad. Consiste en simplemente en dividir el intervalo $[0, 1]$ en tramos de longitud proporcional a la probabilidad de cada una de las 4781 combinaciones posibles y generar un número aleatorio. De esta forma, la combinación correspondiente al intervalo en el que cae el número aleatorio será la elegida y quedarán por tanto determinados todos los enfrentamientos. Este es el método más simple, directo y fiel a los resultados obtenidos, sin embargo es aburrido, poco atractivo para el público, además de poder generar desconfianza en el programa utilizado.

[4] La *tabla* está caracterizada por $\hat{p} = (P_{11}, P_{12}, ..., P_{NN})^T$

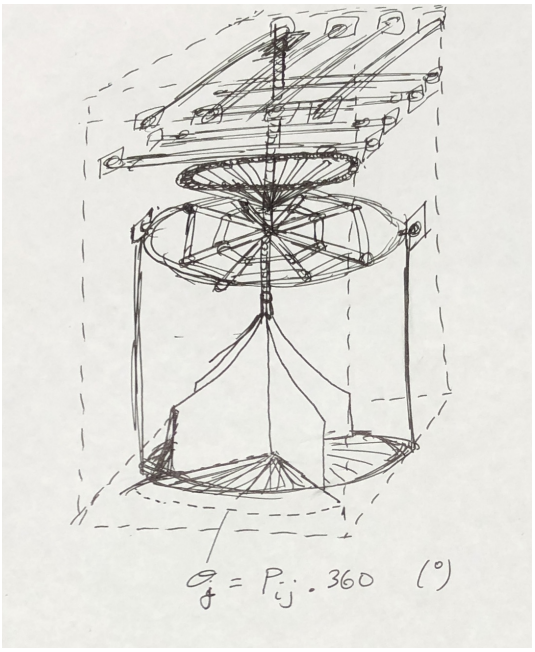
5.2. Sorteo de los Quesitos

Para este sorteo, se probone el método que hemos llamado la **Ruleta de los Quesitos** que tiene dos partes, una generalización de la máquina de Galton a tres dimensiones, aplicando una simetría de revolución a la máquina clásica (figura 6a) y los un recipiente cilíndrico con divisiones angulares, que llamamos "quesitos" debido a su semejanza con este producto (Véase Figura 6b).

La máquina de Galton 3D está hecho con cilindros cruzados de tal manera que una bola se suelta centrada y cae rebotando en la red de obstáculos, de esta forma se distribuye de manera uniforme angularmente. Esto se debe a la simetría de revolución que genera la disposición meticulosa de los cilindros, que debe ser estudiada con detalle a la hora de construir el aparato.



(a) Clásica 2d (2D. PhysikingerC, Public domain, via Wikimedia Commons.)



(b) Boceto de la Ruleta de Quesitos

Los "quesitos" consisten en N separaciones que rotan sobre el eje vertical y que permiten, con una precisión angular de aproximadamente 3^{o} (para poder representar probabilidades con dos cifras signficativas), crear una distribución que corresponde a una distribución de probabilidad concreta.

Este método de sorteo parte de unas $\hat{\mu}, \hat{p}$ dadas (previamente calculados por métodos ya explicados). En primer lugar, se sortea un equipo del grupo P de manera aleatoria (usando una distribución uniforme de probabilidad), por ejemplo, con las bolas que emplea la UEFA actualmente. A continuación se obtiene un equipo del grupo S usando la Ruleta de los Quesitos de la manera que se explicará más adelante.

Una vez se tiene el enfrentamiento (suponemos que se obtiene el enfrentamiento (k,l)), se recalculan las probabilidades de la *tabla*. Dadas $\hat{\mu}, \hat{p}$ se definen las nuevas probabilidades $\hat{p}' = (P'_{11}, ..., P'_{NN})$ como $P'_{ij} := P(\Omega_{ij}|\Omega_{kl}) = \sum_{A_t \in \Omega_{ij} \cap \Omega_{kl}} m_t \cdot A_t$, donde cada P'_{ij} define una distribución de probabilidad[5]. (Recuérdese la definición de $\Omega_{ij} \in \mathcal{F}$ hecha en la Sección 2.1). Las nuevas probabilidades $\hat{p}$ definirán una nueva *tabla*

Definamos $\hat{\mu}' = (m'_1, \cdots, m'_n)$ como

$$m'_i = \frac{\sum\limits_{t:A_t \in \Omega_{ij} \cap \Omega_{kl}} m_t}{\lambda}$$

donde $\lambda = \sum_{t:A_t \in \Omega_{kl}} m_t$ Se repite el proceso de nuevo, quitando el equipo i del grupo P del sorteo inicial y tomando la nueva tabla (el programa se encarga de que el equipo obtenido del grupo S no vuelva a aparecer al asignarle probabilidad 0). A partir de ahora se toma el espacio de probabilidad $(\Omega_{ij}, \mathcal{P}(\Omega_{ij}), \mu')$ y se reitera todo el proceso.

Este procedimiento requiere de un programa que vaya actualizando las probabilidades a medida que se avanza en la gala. Le toma menos de 1 segundo hacer el cálculo al ordenador, por lo que es factible dentro de la dinámica de una gala. Además, este método evita los bloqueos por construcción, ya que, asigna una probabilidad de 0 a aquellos enfrentamientos que generaran uno.

Algoritmo:

1. Elegir un equipo del grupo P de manera aleatoria con las bolas que emplea la UEFA actualmente.
2. Tomar la fila del equipo obtenido y distribuir las compuertas de la Ruleta de los Quesitos acorde a las probabilidades de la fila para cada equipo.
3. Dejar caer una bola por nuestra ruleta y anotar en qué región cae
4. Introducir en el programa la posición del nuevo enfrentamiento para que calcule las probabilidades nuevas .
5. Repetir el proceso, esta vez, quitando del sorteo el equipo del grupo P obtenido en el punto 1.

La pregunta natural llegados a este punto es si las probabilidades de cada enfrentamiento dadas por la *tabla* original corresponde a lo obtenido mediante este sistema de sorteo. El tratamiento analítico de este problema para un N general no se ha llegado a realizar. Sin embargo, se han realizado simulaciones de hasta 10500 sorteos con $N = 8$ y se ha comprobado que las probabilidades tienden a converger a las probabilidades buscadas (Véase Figura 6). Llegados a un número de sorteos, la diferencia entre las probabilidades llegan a ser del orden de 10^{-4} que es menor que el error numérico que se arrastra en el cálculo de probabilidades al emplear tres

[5]Nótese que estas probabilidades dependen de la elección $\hat{\mu}$

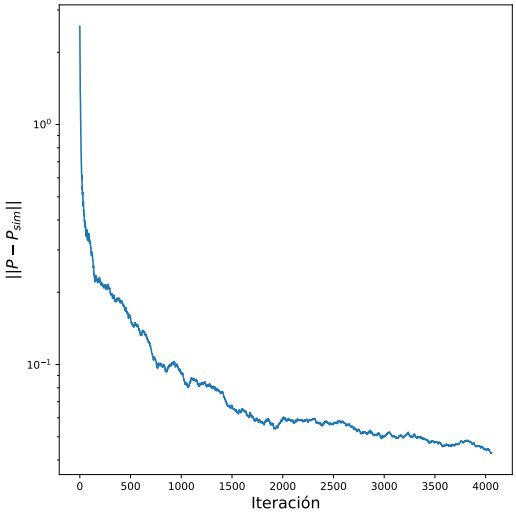


Figura 6: Convergencia de las probabilidades simuladas de los sorteos respecto a las probabilidades buscadas. Se emplea la norma matricial de Frobenius.

cifras significativas, Por tanto, se concluye que, aunque no se haya demostrado analíticamente que este método sea equivalente a obtener las probabilidades que se buscan, para el caso que se ocupa (con errores del orden de 10^{-3}), este sorteo reproduce las probabilidades de la *tabla* orginal, haciendo de este sorteo un método adecuado de sorteo, más equitativo que el establecido en la actualidad.

Para comprobar la simetría angular de la Máquina de Galton 3D, se ha simulado una partícula con dinámica Browniana bajo el potencial gravitatorio. Para ello se ha empleado la Ecuación diferencial estocástica de Langevin (1908):

$$m\frac{d^2\vec{r}}{dt^2} = -\eta\frac{d\vec{r}}{dt} + \xi(t)$$

donde $-\eta\frac{d\vec{r}}{dt}$ es una fuerza viscosa y $\xi(t)$ es una fuerza estocástica cuyas propiedades son:

- $< \xi(t) >= 0$
- $< \xi(t)\xi(t') >= \delta(t - t')$

No se pretende dar una formalización rigurosa de esta parte, ya que, se considera que no tiene relevancia en este trabajo (para el lector más curioso, véase [2]).

Para resolver la ecuación diferencial, se ha empleado Box-Muller para la generación de números aleatorios y el método de Euler-Maruyama [4] para hacer la integración numérica estocástica (véase código del programa en Anexo). Para la simulación suponemos como condiciones iniciales $(\vec{x}, \vec{p}) = (0, h, 0, \vec{0})$ donde h es la altura desde donde se deja caer la bola, y añadimos un potencial

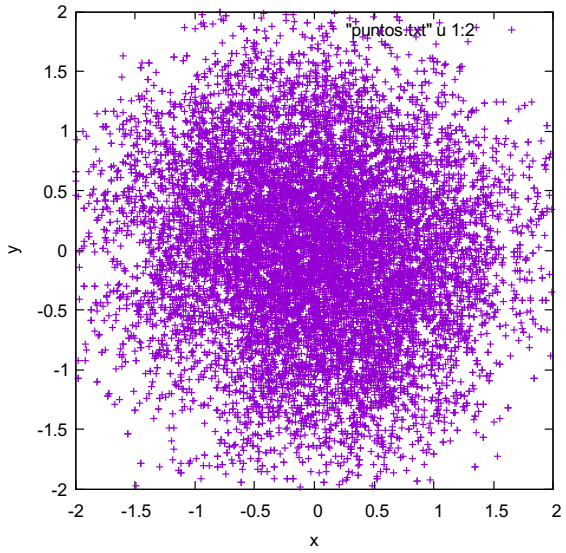


Figura 7: Simulaciones de máquina de galton 3d

$V(\vec{r}) = m \cdot g \cdot y$ donde m masa de la bola y $g = 9{,}8m/s^2$; las unidades y los valores no van a tener relevancia en este propósito, ya que se busca analizar el comportamiento cualitativo.

Se aprecia que las simulaciones muestran una distribución angular uniforme para el movimiento browniano de una partícula. En el caso de un diseño cuidadoso de la posición de los cilindros (una forma es que las geometrías que generan estos cilindros vistos desde encima tengan también simetría angular), el movimiento de una bola bajo esta máquina de galton 3d se asemeja mucho al movimiento browniano. Por tanto, esta simulación justifica cualitativamente el procedimiento empleado en este sorteo para emular una distribución de probabilidad determinada.

En cuanto a la practicidad del sorteo, este método resulta vistoso ya que las bolas y la Ruleta de los Quesitos hacen de la gala algo dinámico y ameno. En primer lugar, permite que la UEFA invite a personajes de renombre en el ámbito del fútbol profesional a que sean ellos los que cojan una bola al azar o que lanzen una pelota a la Ruleta de los Quesitos. Además, los espectadores ven con sus propios ojos el movimiento de la bola, y no dudarán de que el movimiento de la bola da un resultado aleatorio. Lo único que puede no ser entendido del todo por el público puede ser las diferentes distribuciones de probabilidades que se colocan en la Ruleta; se podría añadir este informe a la página web de la UEFA para que los escépticos entendieran el motivo de este detalle.

Esta ruleta se adapta al *timing* que se desea imponer en la gala. Ya que, no requiere prácticamente de tiempo el recolocar los separadores como se desee. Además, la ruleta podría automatizarse con la precisión necesaria para que durante la gala, se pudieran mover los separadores de manera automática rápida y precisamente, mientras se distrae al público con discursos de personas reconocidas o con la labia de el/la presentador/a. Además, se podría hacer todo lo estético que se quiera añadiendo pantallas con los equipos que corresponde a cada intervalo angular, o el uso de LEDs para iluminar cada hueco con los colores característicos de cada uno.

5.3. Máquina de Bingo con muchas bolas.

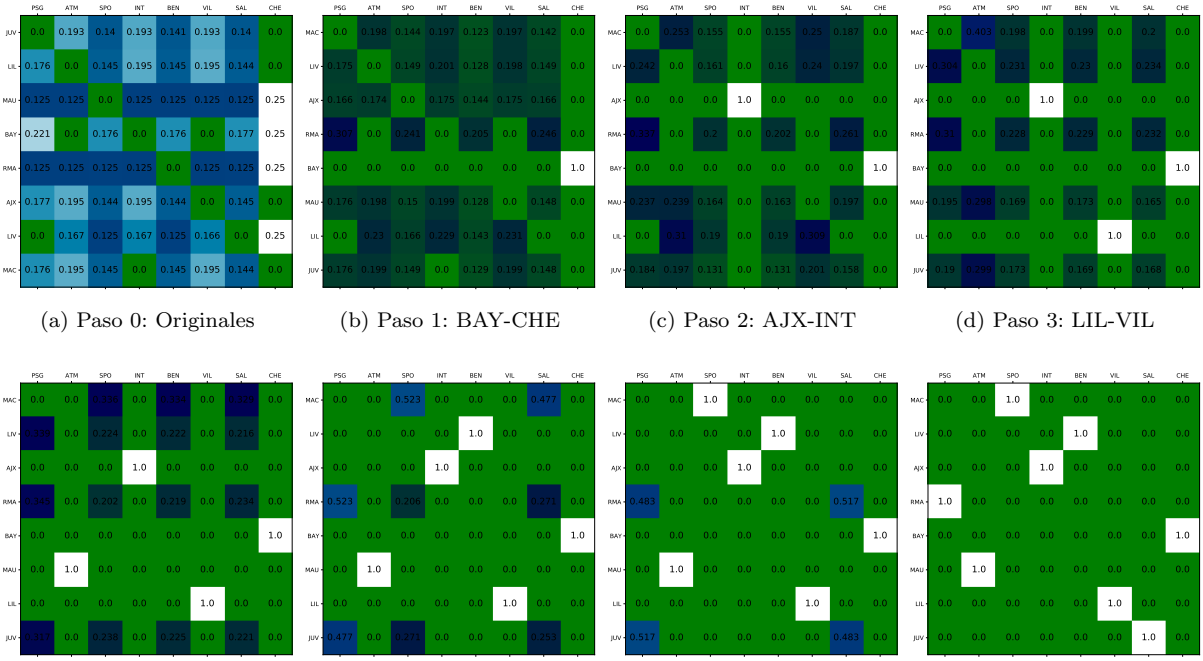
Una variación del método anterior que sólo cambia en el mecanismo físico de hacer el sorteo el el siguiente: sustituir la "Ruleta de los Quesitos" por una clásica máquina de bingo con 100 bolas. En cada iteración se introduce un número de bolas proporcional a la probabilidad condicionada correspondiente. Por ejemplo, supongamos que ha salido en el sorteo del campeón el Real Madrid y nuestra densidad de probabilidad dicta que la probabilidad de que se enfrente con el PSG es de 0.135, entonces introduciremos 14 bolas correspondientes a este equipo en la máquina. Si se desea mayor precisión se puede tomar el numero total como 1000 y obtener así 3 cifras significativas de precisión (introduciendo 135 bolas).

Este método tiene la ventaja de ser más fácil de implementar técnicamente, pero también es menos original.

5.4. Ejemplo de sorteo.

Se va a realizar un simulación del método de sorteo que se usa en el método de "La Ruleta de los Quesitos" y "EL Bingo con Muchas Bolas". Para ello comenzamos con los equipos de este año y las probabilidades dadas por el criterio 2 (máximo menos mínimo).

1. En el sorteo sale el Bayern Munich, tras hacer el sorteo mediante la "Ruleta" o el "Bingo" en base a las probabilidades de pareja, se obtiene como contrincante al Chelsea.
2. En el sorteo se obtiene el Ajax de Ámsterdam y ahora se hace el sorteo con las probabilidades de los contrincantes condicionadas a que ya haya salido el enfrentamiento BAY-CHE (fila AJX de la primera matriz de la figura 8). Se obtiene el enfrentamiento contra el Inter de Milán.
3. En el sorteo sale el Lille y se realiza el sorteo condicionado a que ya hayan salido BAY-CHE y AJX-INT. Se obtiene el Villarreal.
4. Análogamente se llega a Manchester United-Atlético de Madrid.
5. Sale Liverpool-Benfica.
6. En el sorteo sale el Manchester City-Sporting de Lisboa.
7. Ahora se obtiene Real Madrid-PSG y sólo queda el enfrentamiento Juventus-Red Bull Salzburgo.



(a) Paso 0: Originales (b) Paso 1: BAY-CHE (c) Paso 2: AJX-INT (d) Paso 3: LIL-VIL

(e) Paso 4: MAU-ATM (f) Paso 5: LIV-BEN (g) Paso 6: MAC-SPO (h) Paso 7: RMA-PSG, JUV-SAL

Figura 8: Ejemplo de Sorteo.

6. Conclusiones

Se concluye que el criterio de equidad matemático que más se aproxima a nuestra concepción intuitiva es la de minimizar la diferencia entre el máximo y el mínimo valor. Por tanto, se corrobora el resultado del informe del enunciado.

En este trabajo se han planteado 3 formas de sorteo. La elección de una vendrá dada por las preferencias de la UEFA de cara a las características que debe tener la gala, es decir, si prioriza la precisión de los sorteos o el dinamismo y la componente visual de la gala. Se considera la Ruleta de los Quesitos como una opción intermedia entre ambas características.

De cara a futuros desarrollos, se plantean varias líneas de trabajo:

- Se propone, al equipo de Diseño de la UEFA, cualquier mejora estética de la Ruleta de los Quesitos, siempre y cuando, se mantengan las características y el formato del mismo.
- En cuanto a la Ruleta de los Quesitos, dada una *tabla* con sus probabilidades $\hat{p}$ se plantea tratar de hallar un método numérico que evalúe de manera aleatoria diferentes puntos del poliedro dado por $S = \{\hat{x} : A\hat{x} = \hat{p}, \hat{x} \geq 0\}$ para obtener $\hat{\mu}'$ con mejores propiedades

(por construcción, todos los puntos del poliedro S generan la misma *tabla*). Con mejores propiedades se entiende que, empleando la nueva $\hat{\mu}'$, las probabilidades obtenidas mediante simulaciones de los sorteos converjan más rápidamente a las probabilidades de la *tabla* en cuestión. Se podrían estudiar diferentes criterios para ello, como pueden ser los criterios de equidad planteados en este trabajo[6]. La evaluación aleatoria de los puntos del poliedro, requieren del conocimiento de los puntos extremos del mismo para, mediante el Teorema de Representación, hallar cualquier punto como combinación convexa de las mismas; sin embargo, en este caso, hallar los puntos extremos se vuelve un problema arduo y se requeriría del uso de la técnica de programación lineal de Generación de columnas[7]. Por eso, la idea que se plantea es realizar la evaluación aleatoria mediante el método Montecarlo; sin embargo, la probabilidad de salirse fuera del poliedro en estos movimientos aleatorios es alta. Se propone, minimzar también la distancia desde el punto nuevo al poliedro, para que los puntos aleatorios se acerquen lo máximo al mismo.

Referencias

[1] C. E. SHANNON, A Mathematical Theory of Communication, 1948, from *The Mathematical Theory of Communication*.

[2] Stochastic dynamical systems, from http://www.scholarpedia.org/article/Stochastic_dynamical_systems

[3] G. BHANOT, The Metropolis algorithm (1988), Rep. *Prog. Phys.* 51 429.

[4] Euler-Maruyama method for approximation of stochastic differential equations, from https://appliedprobability.blog/2020/10/11/euler-maruyama/.

[6] Maximizar la desinformación, minimizar la diferencia entre máximo y mínimo o minimizar la varianza

[7] Cuando se trata de un problema de grandes dimensiones, se aproxima el problema a pequeños problemas aseguibles que se resuelven mediante el método de Símplex, y, a continuación, se evalúa si el resultado es suficientemente bueno como para aceptarlo

A. Entropía

En este apartado se pretende definir formalmente el concepto de desinformación de una *tabla* a partir de la entropía de Shannon.

Definición 5. Una variable aleatoria X es una función real definida sobre un espacio de probabilidad $(\Omega, \mathcal{F}, \mu)$, asociado a un experimento aleatorio

$$X : \Omega \to \mathbb{R}$$

Definición 6. Se define como entropía H de una variable aleatoria discreta X, que toma valores en $\varkappa$ y está distribuido de acuerdo a $p : \varkappa \to [0, 1]$ tal que $p(x) := \mathbb{P}[X = x]$ como:

$$H(X) = \mathbb{E}[-log_b p(X)]$$

La entropía se puede escribir explícitamente como

$$H(X) = -\sum_{x \in \varkappa} p(x)\, log_b p(x)$$

aquí se tomará $b = e$

Definición 7. Dada una *tabla* caracterizada por $\hat{p} = (P_{11}, \cdots, P_{NN})$, se define como probabilidad condicionada a una fila como:

$$p_i(A \in \Omega_{ij}) = P_{ij}$$

Definición 8. Se define la variable aleatoria discreta X_i a aquella asociada al espacio de probabilidad $(\Omega_i := \bigcup_j \Omega_{ij}, \mathcal{F}(\Omega_i), p_i)$ tal que

$$X : \Omega_i \to \mathbb{N}$$

$$A \in \Omega_{ij} \to j$$

Por construcción, $\sum_{A \in \Omega_i} p_i(A) = 1 \ \forall i$.

Partiendo de la definición de entropía presentada, se construye una magnitud que se llamará desinformación asociada a una *tabla* , que se empleará como criterio de equidad.

Definición 9. Dada una *tabla* caracterizada por $\hat{p} = (P_{11}, \cdots, P_{NN})$, se define como desinformación asociada a la *tabla* como

$$S(\hat{p}) = \sum_{i=1}^{N} H(X_i)$$

Se puede escribir explícitamente como

$$S(\hat{p}) = -\sum_{i,j=1}^{N} P_{ij} log(P_{ij})$$

B. Código

```
from sympy import *
import numpy as np
from scipy.optimize import fsolve


N=8
posiciones_condiciones=[[0,7],[1,7],[3,1],[3,5],[5,7],[6,0],[7,3]]
num_condiciones=len(posiciones_condiciones)
num_condiciones_total=2*N
NN=N**2-N-num_condiciones

for i in range(N):
    for j in range(N):
        exec(f"a{i}{j}=symbols('a{i}{j}')")


for i in range(num_condiciones_total):
    exec(f"l{i}=symbols('l{i}')")

def orden(i,j):
    s=0
    for k in range(N):
        for t in range(N):
            if i==k and t==j:
                return s
            s+=1
def comprobar(i,j):
    for t in range(num_condiciones):
        if posiciones_condiciones[t][0]==i and posiciones_condiciones[t][1]==j:
            return 0

    return 1


S=symbols('S')

S=0
media=0
t=0
for i in range(N):
    for j in range(N):
        if i!=j and comprobar(i,j):
            exec(f"media+=a{i}{j}")
            t+=1
media=media/t

for i in range(N):
    for j in range(N):
        if i!=j and comprobar(i,j):
            #exec(f"S=S+1/(NN-1)*(a{i}{j}-media)**2")
            exec(f"S=S+a{i}{j}*log(a{i}{j})")
    #xec(f"S=S+l{i}*(a{i}{i})")

#for i in range(num_condiciones):
    #exec(f"S=S+l{i+2*N}*(a{posiciones_condiciones[i][0]}{posiciones_condiciones[i
    ][1]})")


for i in range(N):
```

```
    for j in range(N):
        if i!=j and comprobar(i,j) :
            exec(f"S=S+l{i}*(a{i}{j})")
        if i!=j and comprobar(j,i):
            exec(f"S=S+l{i+N}*(a{j}{i})")
    exec(f"S=S-l{i}-l{i+N}")
#for i in range(num_condiciones):
    #exec(f"S=S-l{posiciones_condiciones[i][0]}*a{posiciones_condiciones[i][0]}{
    posiciones_condiciones[i][1]}-l{posiciones_condiciones[i][0]+N}*a{
    posiciones_condiciones[i][0]}{posiciones_condiciones[i][1]}")

for i in range(N):
    for j in range(N):
        if i!=j and comprobar(i,j):
            exec(f"A{i}{j}=diff(S,a{i}{j})")
for i in range(num_condiciones_total):
    exec(f"B{i}=diff(S,l{i})")

def func(x):
    sol=[]
    for i in range(N):
        for j in range(N):
            if i!=j and comprobar(i,j):
                exec(f"sol.append(A{i}{j})")
    for i in range(num_condiciones_total):
        exec(f"sol.append(B{i})")
    for t in range(len(sol)):
        m=0
        for i in range(N):
            for j in range(N):
                if i!=j and comprobar(i,j):
                    exec(f"sol[t]=sol[t].subs(a{i}{j},"+"x[m]"+")")
                    m+=1

        for i in range(num_condiciones_total):
            exec(f"sol[t]=sol[t].subs(l{i},x[NN+i])")
    return sol

cond_ini=[]
variables=[]
for i in range(N):
    for j in range(N):
        if i!=j and comprobar(i,j):
            exec(f"variables.append(a{i}{j})")
for i in range(NN):
    cond_ini.append(1/N)
for i in range(num_condiciones_total):
    cond_ini.append(0.1)
    exec(f"variables.append(l{i})")

root=fsolve(func,cond_ini)


#root = nsolve(sol,variables, cond_ini)
root

m=0
resultado=[]
for i in range(N):
    resultado2=[]
    for j in range(N):
        if i!=j and comprobar(i,j):
```

```
            print(root[m])
            resultado2.append(round(root[m],3))
            m+=1
        else:
            resultado2.append(0)
            print(0)
    resultado.append(resultado2)
    print("")

import numpy as np
import matplotlib.pyplot as plt

plt.rcParams["figure.figsize"] = [15.50, 8.50]
plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()
ax.matshow(resultado, cmap='summer')

for i in range(N):
    for j in range(N):
        c = resultado[j][i]
        ax.text(i, j, str(c), va='center', ha='center')

plt.show()
```

Listing 1: Metodo de Lagrange para minimizar la entropia

```
import numpy as np
import scipy.optimize
import matplotlib.pyplot as plt
import random as rd
from sympy.utilities.iterables import multiset_permutations
import sympy

#ELEGIR UNA DE LAS 3 FUNCIONES:

def f(x):
  aux=B.dot(x)
  return np.max(aux)-np.min(aux[aux!=0])

def f(x):
  aux=B.dot(x)
  val=0
  for i in range(len(aux)):
    if aux[i]!=0:
      val=val-aux[i]*np.log(aux[i])
  return val

def f(x):
  aux=B.dot(x)
  return np.var(aux[aux!=0])

N=8
val=np.array([[0,1,1,1,1,1,1,0],[1,0,1,1,1,1,1,0],[1,1,0,1,1,1,1,1],
[1,0,1,0,1,0,1,1],[1,1,1,1,0,1,1,1],[1,1,1,1,1,0,1,0],[0,1,1,1,1,1,0,1],
[1,1,1,0,1,1,1,0]])

A_aux=np.zeros(N)
B=np.zeros((N*N,4781))
N_posib=0
a = np.array([1,2,3,4,5,6,7,8])
#a = np.array([1,2,3])
```

```
for p in multiset_permutations(a):
  sirve=1
  for i in range(N):
    if val[p[i]-1,i]!=1:
      sirve=0
  if sirve:
    for i in range(N):
      B[(p[i]-1)*N+i,N_posib]=1
    N_posib=N_posib+1

tam_mu=4781
sol=np.ones(tam_mu)/tam_mu
it_max=100000000
tol=0.1
eps=1
it=0
change=0.000001
P_acep=0.005

while it<it_max:
  it=it+1
  x=rd.randint(0,tam_mu-1)
  y=rd.randint(0,tam_mu-1)
  if x!=y:
    posible_sol=np.copy(sol)
    posible_sol[x]=posible_sol[x]+change
    posible_sol[y]=posible_sol[y]-change
    if posible_sol[x]>0 and posible_sol[y]>0:
      if f(posible_sol)<=f(sol):
        sol[x]=posible_sol[x]
        sol[y]=posible_sol[y]
      elif rd.random()<P_acep:
        sol[x]=posible_sol[x]
        sol[y]=posible_sol[y]

vec_P=np.matmul(B,sol)
count=0
P=np.zeros((8,8))
for i in range(N):
  for j in range(N):
      P[i,j]=vec_P[count]
      count=count+1
np.size(P[P==0])

plt.rcParams["figure.figsize"] = [7, 7]
plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()
ax.matshow(P, cmap='ocean')

for i in range(N):
   for j in range(N):
      c = round(P[j, i],3)
      ax.text(i, j, str(c), va='center', ha='center',size=15)

ax.xaxis.set_ticklabels(['','PSG','ATM','SPO','INT','BEN','VIL','SAL','CHE']); ax.
    yaxis.set_ticklabels(['','JUV','LIL','MAU','BAY','RMA','AJX','LIV','MAC']);
plt.show()
fig.savefig("Estocast_var.pdf")

fig, ax = plt.subplots()
plt.bar(np.linspace(0,4781,4781),sol)
```

```
ax.set_xlabel('Combinaci n',size=15);ax.set_ylabel('Probabilidad',size=15);
fig.savefig("Densidad_prob_var.pdf")
plt.show()

```

Listing 2: Metodo Montecarlo de variacin de la funcion de probabilidad.

```
#AQUI SE SUPONE QUE SE TIENE DEFINIDO UN VECTOR SOL CON LAS SOLUCIONES DE LA
    OPTIMIZACION
P_new=P
it_max=15000
mu=sol
final=np.zeros((N,N))
primeros=np.arange(N)
v_norma=[]
for it in range(it_max):
  B_new=B
  mu=sol
  v_elegidos=np.random.permutation(primeros)
  contador=0
  for elegido in v_elegidos:
    contador+=1
    aleatorio=rd.random()
    vect_P_cond=P_new[elegido,:]
    #vect_P_cond.remove(0)
    f_distr_cond=np.cumsum(vect_P_cond)
    pos_contr=0
    fin=0
    for ind_dis in range(len(f_distr_cond)):
      if f_distr_cond[ind_dis]>=aleatorio or fin==1:
        fin=1
      else:
        pos_contr+=1
    contrincante=primeros[pos_contr]
    #print(elegido,contrincante)
    final[elegido,contrincante]+=1
    mu_aux,B_aux=condicionada(elegido,contrincante,mu,B_new)
    mu=np.copy(mu_aux)
    B_new=np.copy(B_aux)
    vec_P=B_new.dot(mu)
```

Listing 3: Simulacion del algoritmo de sorteo.

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>
#define NormRANu (2.3283063671E-10F)
#define PUNTOSMAX  250000
#define BLOQUESMAX 500
#define MEDIDASMAX 100000

#define NormRANu (2.3283063671E-10F)
#define PUNTOSMAX  250000
#define BLOQUESMAX 500
#define MEDIDASMAX 100000
#define pi 3.141592
#define m   1
#define k   1
#define const_estocastica  2*mu*T

double parisi_rapuano ()
```

```
{
    unsigned int irr [256];
    unsigned char ind_ran=0, ig1=0, ig2=0, ig3=0;
    double numero_aleatorio;
    int i;

    for (i=0; i<256; i++)
        irr [i] = (rand () << 16) + rand ();

    ig1 = ind_ran - 24;
    ig2 = ind_ran - 55;
    ig3 = ind_ran - 61;
    irr [ind_ran] = irr [ig1] + irr [ig2];

    numero_aleatorio = (irr[ind_ran]^irr [ig3]);
    ind_ran++;

    return numero_aleatorio*NormRANu;
}

void box_muller (double *g1, double *g2, double varianza)
{
    double w1, w2;
    w1 = parisi_rapuano ();
    w2 = parisi_rapuano ();

    *g1 = -sqrt(-2*log(w1))*cos(2*pi*w2)*sqrt(varianza);
    *g2 = -sqrt(-2*log(w1))*sin(2*pi*w2)*sqrt(varianza);
}

void eulerMaruyama(double x[6],double y[6],double h,double mu, double T, double
    num_aleatorio){
    double random1, random2,random3, random4;

    box_muller (&random1, &random4, const_estocastica);
    box_muller (&random3, &random4, const_estocastica);
    box_muller (&random2, &random4, const_estocastica);

    y[0]=x[0]+h*x[1]/m;
    y[2]=x[2]+h*x[3]/m;
    y[4]=x[4]+h*x[5]/m;

    y[1]=x[1]+(-mu*x[1]/m)*h+sqrt(2*mu*T*h)*random1;
    y[3]=x[3]+(-mu*x[3]/m-m*9.8)*h+sqrt(2*mu*T*h)*random3;
    y[5]=x[5]+(-mu*x[5]/m)*h+sqrt(2*mu*T*h)*random2;

}
void min_max (int npuntos, double *puntos, double *minimo, double *maximo)
{
    int i;

    *minimo = *maximo = puntos [0];
    for (i=1; i<npuntos; i++)
    {
        if (puntos[i]<*minimo) *minimo=puntos[i];
        if (puntos[i]>*maximo) *maximo=puntos[i];
    }
}
void construye_histograma (int nbloques, int npuntos, double *puntos, double *hist
    , double *x)
{
    double minimo, maximo, delta;
```

```
80      int i, aux;
81
82      min_max (npuntos, puntos, &minimo, &maximo);
83      delta = (maximo-minimo)/((double)(nbloques));
84      printf ("minimo=%lf\tmaximo=%lf\tdelta=%lf\n", minimo, maximo, delta);
85
86      for (i=0; i<npuntos; i++)
87      {
88          aux = (int)((puntos[i]-minimo)/delta);
89          if (aux==nbloques) aux--;
90          hist[aux] ++;
91      }
92
93      for (i=0; i<nbloques; i++)
94      {
95          hist[i]/=npuntos;
96          x[i]=minimo+i*delta;
97      }
98
99
100 }
101
102
103 void exporta_histograma (int nbloques, double *x, double *hist, char *nombre)
104 {
105     int i;
106     FILE *f = fopen (nombre, "w");
107
108     for (i=0; i<nbloques; i++)
109     {
110         fprintf (f, "%lf\t%lf\n", x[i], hist[i]);
111     }
112     fclose (f);
113 }
114
115
116 int main(){
117
118 double t_total=50;
119 double h=0.01;
120 double mu=1.5;
121 double T=0.1;
122 double hist [BLOQUESMAX], hist_ejex[BLOQUESMAX];
123 double xaux, x[MEDIDASMAX], histx [BLOQUESMAX], x_ejex[BLOQUESMAX];
124 FILE *f,*p,*d;
125 double nbloques=10;
126
127 int num_pasos=(int)(t_total/h);
128 double altura=100;
129
130 f=fopen("estados.txt","w");
131 p=fopen("puntos.txt","a");
132 d=fopen("distancias.txt","w");
133
134 double estados[num_pasos+1][6];
135
136 // Condiciones iniciales
137 estados[0][0]=0;
138 estados[0][1]=0;
139 estados[0][2]=altura;
140 estados[0][3]=0;
141 estados[0][4]=0;
```

```
142 estados[0][5]=0;
143
144 double itmax=10000;
145 double x_aux,y_aux;
146
147 int seguir=1;
148 for(int j=0;j<=itmax;j++){
149     seguir=1;
150     for(int i=1;i<=num_pasos && seguir;i++){
151         eulerMaruyama(estados[i-1],estados[i],h,mu,T,0.1);
152         if (estados[i][2]<=0){
153             x_aux=estados[i][0];
154             y_aux=estados[i][4];
155             fprintf(p,"%f %f\n",x_aux,y_aux);
156             //dd[i]=sqrt(x_aux*x_aux+y_aux*y_aux);
157             fprintf(d,"%f\n",sqrt(x_aux*x_aux+y_aux*y_aux));
158             seguir=0;
159         }
160         fprintf(f,"%f %f %f \n",estados[i][0],estados[i][4],estados[i][2]);
161     }
162 }
163
164
165
166 fclose(p);
167 fclose(d);
168 fclose(f);
169 system("distancias.plt");
170 system("distancias2.plot");
171
172
173 }
```

Listing 4: Simulacion estocastica Euler-Maruyama