RESOLUCIÓN DE RELACIONES DE RECURRENCIA CON APOYO DE *MATHEMATICA*

Enrique Vílchez Quesada
Universidad Nacional de Costa Rica. (Costa Rica)
enrique.vilchez.quesada@una.cr

RESUMEN: El presente trabajo introduce algunos algoritmos para resolver relaciones de recurrencia lineales, homogéneas y no homogéneas, con coeficientes constantes y no constantes, utilizando software como recurso principal en los procesos de resolución. La aplicación comercial *Mathematica* ha brindado el sustento técnico necesario para la implementación de los métodos empleados. Se presentan además, distintos ejemplos de relaciones de recurrencia, mostrando la efectividad y limitaciones de los algoritmos creados por el autor y programados en el ambiente que provee *Mathematica*.

Palabras clave: relaciones de recurrencia, solución, software *mathematica*

ABSTRACT: This work introduces some algorithms to solve both homogeneous and non homogeneous lineal recursion relations, with constant and non constant coefficients, by using software as a main resource in solving processes. The commercial application “Mathematics” has given the necessary technical support to the implementation of the methods being used. Different examples of recursion relations are also given. They show the effectiveness and limitations of the algorithms created by the author and programmed in the environment that “Mathematics” software provides.

Key words: recursion relations, solution, “*mathematica*” software

Introducción

La resolución de relaciones de recurrencia es un tema fundamental en distintas áreas de conocimiento al proporcionar métodos de solución ante problemas complejos que muchas veces no pueden ser abordados de forma directa. Sus aplicaciones abarcan contenidos vinculados con matemática básica, estructuras de datos, análisis de algoritmos, entre otras.

Los procedimientos clásicos de resolución expuestos en la mayor parte de la literatura (Johnsonbaugh, 2005, Kolman, Busby y Ross, 1997) se fundamentan en la construcción de ecuaciones y sistemas de ecuaciones que los vuelven poco eficientes cuando el término a_n de la relación de recurrencia, depende de una cantidad significativa de expresiones anteriores a "*n*".

Con el presente artículo se muestran algunos algoritmos novedosos para resolver relaciones de recurrencia lineales, homogéneas y no homogéneas, con coeficientes constantes y no constantes, utilizando como apoyo el uso de software. En la actualidad el empleo de programas de computación para contribuir con la simplificación de procesos es una tarea muy común y necesaria, cuando los problemas abordados implican una cantidad de cálculos relativamente grandes. En este contexto, el software comercial *Mathematica* se ha convertido en el insumo esencial para resolver el álgebra relacionada con los métodos compartidos.

La notación que caracteriza a este documento, se sustenta en suponer una sucesión de números reales como una función a, $a : IN \cup \{0\} \to IR$ y donde a su elemento *n*-ésimo se le denota como a_n. Además, la sucesión de números reales se representa a través de la expresión: $(a_n)_{n \in IN \cup \{0\}}$.

Relaciones de recurrencia homogéneas lineales]

En esta sección se muestra un método para encontrar más rápidamente los elementos de una sucesión definida por una relación de recurrencia homogénea lineal. Las ideas se basan en expresar la relación de recurrencia mediante un sistema de ecuaciones lineales, aspecto que ya había sido expuesto en Vílchez (2004). El aporte del presente trabajo, reside en exhibir algunas funciones construidas mediante el uso del software *Mathematica*, para resolver relaciones de recurrencia de este tipo con o sin coeficientes constantes.

Aspectos generales

Una relación de recurrencia homogénea lineal es aquella de la forma:

$$a_{n+k} = \beta_{k-1}(n)a_{n+(k-1)} + \beta_{k-2}(n)a_{n+(k-2)} + \cdots + \beta_1(n)a_{n+1} + \beta_0(n)a_n \qquad (1)$$

junto con las *k* condiciones iniciales:

$$a_j = c_j, 0 \le j \le k-1$$

siendo los $\beta_j(n)$ funciones y los c_j números reales fijos $\forall j,\ j \in IN \cup \{0\}, 0 \le j \le k-1$.

El lector debe observar que si todos los $\beta_j(n)$ son números reales la relación de recurrencia formada es de coeficientes constantes. El método aquí propuesto se fundamenta en el siguiente sistema de ecuaciones:

$$\begin{cases} a_{n+k} = \beta_{k-1}(n)a_{n+(k-1)} + \beta_{k-2}(n)a_{n+(k-2)} + \cdots + \beta_1(n)a_{n+1} + \beta_0(n)a_n \\ a_{n+(k-1)} = a_{n+(k-1)} \\ a_{n+(k-2)} = a_{n+(k-2)} \\ \vdots \\ a_{n+1} = a_{n+1} \end{cases}$$

El cual, matricialmente puede expresarse así:

$$X_{n+1} = \mathbf{A}(n) \cdot X_n \qquad (2)$$

siendo,

$$\mathbf{A}(n) = \begin{pmatrix} \beta_{k-1}(n) & \beta_{k-2}(n) & \cdots & \beta_1(n) & \beta_0(n) \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{pmatrix} \text{ una matriz } k \times k$$

y,

$$X_n = \begin{pmatrix} a_{n+(k-1)} \\ a_{n+(k-2)} \\ \vdots \\ a_n \end{pmatrix} \text{ un vector en } IR^k$$

En (2) por el método iterativo, X_n puede ser dado en términos de:

$$X_0 = \begin{pmatrix} a_{k-1} \\ a_{k-2} \\ \vdots \\ a_0 \end{pmatrix} = \begin{pmatrix} c_{k-1} \\ c_{k-2} \\ \vdots \\ c_0 \end{pmatrix}$$

como se detalla a continuación:

$$\begin{cases} X_1 = \mathbf{A}(0)\cdot X_0 \\ X_2 = \mathbf{A}(1)\cdot X_1 = \mathbf{A}(1)\cdot \mathbf{A}(0)\cdot X_0 \\ X_3 = \mathbf{A}(2)\cdot X_2 = \mathbf{A}(2)\cdot \mathbf{A}(1)\cdot \mathbf{A}(0)\cdot X_0 \\ \vdots \\ X_n = \mathbf{A}(n-1)\cdot \ldots \cdot \mathbf{A}(0)\cdot X_0 = \prod_{j=0}^{n-1} \mathbf{A}(j)\cdot X_0 \end{cases}$$

Lo anterior permite concluir que:

$$X_n = \prod_{j=0}^{n-1} \mathbf{A}(j)\cdot X_0,\ \forall n \in IN \qquad (3)$$

Como:

$$X_n = \begin{pmatrix} a_{n+(k-1)} \\ a_{n+(k-2)} \\ \vdots \\ a_n \end{pmatrix}$$

entonces en (3), a_n corresponde a la última fila de la matriz resultante al calcular:

$$\prod_{j=0}^{n-1} \mathbf{A}(j)\cdot X_0$$

De hecho, la expresión (3) constituye un algoritmo que permite determinar los elementos de a_n más rápidamente en comparación con la relación de recurrencia original (ver 1) y es en este punto donde el empleo de software se convierte en una herramienta esencial.

Encontrando elementos de la sucesión y comparando velocidades

En *Mathematica* una implementación de (3) que retorna la última fila de la matriz es:

```
ElementoSucesionH[Coeficient_List, ConditionInitial_List, m_] :=
 Module[{CD = Reverse[ConditionInitial],
   h = Dimensions[ConditionInitial][[1]] - 1, Matriz},
  Matriz[Coeficientes_List, nn_] :=
   Module[{Identidad = IdentityMatrix[Dimensions[Coeficientes][[1]]],
     A = {Coeficientes}, i, CF},
    For[i = 1, i ≤ Dimensions[Coeficientes][[1]] - 1,
     A = Append[A, Identidad[[i]]]; i++];
    CF[L_List, a_] := Block[{n = a}, L]; Return[CF[A, nn]]];
  A[n_] := Matriz[Coeficient, n];
  If[m > h, Elemento = A[0].Transpose[{CD}];
   For[i = 1, i ≤ m - 1, Elemento = Dot[A[i], Elemento]; i++];
   Return[Elemento[[h + 1, 1]]], Return[ConditionInitial[[m + 1]]]]]
```

Los argumentos **Coeficient_List** y **ConditionInitial_List** definen la relación de recurrencia mediante un vector de coeficientes $\beta_j(n)$ y una lista de condiciones iniciales, respectivamente. En **ElementoSucesiónH** el parámetro **m_** simboliza un número natural sobre el cual, se desea obtener de la sucesión $(a_n)_{n\in IN\cup\{0\}}$, el término a_m. Pese a todos los cálculos que involucra **ElementoSucesiónH**, esta función resulta ser más rápida que la relación de recurrencia original. Lo anterior, se puede comprobar experimentalmente a través del siguiente programa elaborado con *Mathematica*:

```
PruebaH[m_, Con_List, Condici_List] :=
 Module[{k, alr = 0, mt = 0, alm = 0},
  For[k = 0, k ≤ m, Print["Se encontró el elemento de la sucesión: ", k];
   ClearSystemCache[];
   L1 = First[Timing[ElementoSucesionH[Con, Condici, k]]];
   ClearSystemCache[]; L2 = First[Timing[a[k]]];
   If[L1 > L2 && L1 ≠ L2, Print["Es mejor el algoritmo recursivo"];
    alr++, If[L1 == L2, Print["Tienen el mismo tiempo de ejecución"];
     mt++, Print["Es mejor el algoritmo matricial"]; alm++]]; k++];
  Print[
 "El algoritmo recursivo fue más efectivo la cantidad de
   ejecuciones igual a: ", alr];
Print["Tuvieron el mismo tiempo la cantidad de ejecuciones igual a: ",
 mt];
Print[
 "El algoritmo matricial fue más efectivo la cantidad de
   ejecuciones igual a: ", alm]]
```

El comando **Timing** en **PruebaH** determina el tiempo de ejecución en el cálculo de $m+1$ elementos de a_n, comparando dos formas de resolución por medio de las cuales se obtienen: **ElementoSucesiónH**

y la relación de recurrencia inicial. **PruebaH** compara los tiempos y determina cual método fue más eficiente en cada caso (la velocidad está en función del número de cálculos a realizar). Consideremos a continuación algunos ejemplos de recorrido.

Ejemplo 1. Sea la sucesión recursiva $a_{n+2} = 7a_{n+1} - 10a_n$ sujeta a las condiciones $a_0 = 3$ y $a_1 = 1$. Compare la velocidad de convergencia de la relación de recurrencia dada y la función **ElementoSucesiónH**.

Solución 1. Con la finalidad de crear a_n en el software *Mathematica*, se debe cambiar la notación empleada en el enunciado, sustituyendo *n* por *n-2*:

$$a_n = 7a_{n-1} - 10a_{n-2}$$

Luego:

In :=

```
ClearAll[a]
a[n_] := 7 * a[n - 1] - 10 * a[n - 2]
a[0] = 3;
a[1] = 1;

Table[ElementoSucesionH[{7, -10}, {3, 1}, k] == a[k], {k, 0, 30}]

PruebaH[30, {7, -10}, {3, 1}]
```

Out =

```
⋮

Se encontró el elemento de la sucesión: 28
Es mejor el algoritmo matricial
Se encontró el elemento de la sucesión: 29
Es mejor el algoritmo matricial
Se encontró el elemento de la sucesión: 30
Es mejor el algoritmo matricial
El algoritmo recursivo fue
   más efectivo la cantidad de ejecuciones igual a: 1
Tuvieron el mismo tiempo la cantidad de ejecuciones igual a: 15
El algoritmo matricial fue más
   efectivo la cantidad de ejecuciones igual a: 15
```

ClearAll limpia de la memoria la variable a, **Table** verifica que en las 31 comparaciones realizadas, a_n y **ElementoSucesiónH** den el mismo resultado y **PruebaH** proporciona la salida mostrada. Del Out se concluye que en la mayor parte de los casos el algoritmo expuesto en (3) es más eficiente.

Resolviendo relaciones de recurrencia

Mediante el uso de **ElementoSucesiónH** y el comando del software *Mathematica* **FindSequenceFunction** es posible generar una función, que intenta encontrar de manera explícita los elementos de una sucesión dada por una relación de recurrencia homogénea lineal. **FindSequenceFunction** recibe como parámetros una lista con algunos elementos de la sucesión y busca una fórmula explícita que la genere.

No en todos los casos se desempeña exitosamente (si las secuencias tienen números decimales) pero en muchos sí lo hace, convirtiéndose este mecanismo de razonamiento, en la base del algoritmo empleado para buscar la solución de una relación de recurrencia homogénea lineal cualesquiera. El método **ResolRRH** realiza el proceso descrito en este apartado.

```
ResolRRH[Co_List, Con_List] :=
 Module[{ElementoSucesionH, SolveR, SolveRR},
  ElementoSucesionH[Coeficient_List, ConditionInitial_List, m_] :=
   Module[{CD = Reverse[ConditionInitial],
     h = Dimensions[ConditionInitial][[1]] - 1, Matriz},
    Matriz[Coeficientes_List, nn_] :=
     Module[{Identidad = IdentityMatrix[Dimensions[Coeficientes][[1]]],
       A = {Coeficientes}, i, CF},
      For[i = 1, i ≤ Dimensions[Coeficientes][[1]] - 1,
       A = Append[A, Identidad[[i]]]; i++];
      CF[L_List, a_] := Block[{n = a}, L]; Return[CF[A, nn]]];
    A[n_] := Matriz[Coeficient, n];
    If[m > h, Elemento = A[0].Transpose[{CD}];
     For[i = 1, i ≤ m - 1, Elemento = Dot[A[i], Elemento]; i++];
     Return[Elemento[[h + 1, 1]]], Return[ConditionInitial[[m + 1]]]]];
```

```
SolveR[mm_, Coeficien_List, ConditionInitia_List, n_] :=
 Module[{L = {}},
  For[i = 0, i ≤ mm,
   L = Append[L, ElementoSucesionH[Coeficien, ConditionInitia, i]];
   i++]; Return[FunctionExpand[FindSequenceFunction[L, n]]]];
SolveRR[m_, Coeficient_List, ConditionInitial_List, n_] :=
 Module[{L = {}},
  For[i = 0, i ≤ m,
   L = Append[L, ElementoSucesionH[Coeficient, ConditionInitial, i]];
   i++]; Print["La lista de elementos usada es: ", L];
  Return[FunctionExpand[FindSequenceFunction[L, n]]]]; j = 1;
f[n_] := Evaluate@SolveR[j, Co, Con, n];
While[NumericQ[f[1]] == False, j++;
 f[n_] := Evaluate@SolveR[j, Co, Con, n];
 If[j > 1000, Print["Las ejecuciones superaron 1000 pruebas"];
  Break[]; j = 0]];
If[j ≠ 0,
 Print["El valor mínimo para el cual se retorna una función es: ",
  j]; f[n_] := Evaluate@SolveRR[j, Co, Con, n];
 Print["La solución de la relación de recurrencia es: ", f[n]];
 Print[
  "Una lista de elementos generada por la función anterior es: ",
  N[Table[f[n], {n, 1, j + 1}]]]]]
```

ResolRRH busca una lista mínima de elementos de la sucesión recursiva a_n para la cual **FindSequenceFunction** retorna una respuesta. Esto puede provocar en algunos ejemplos que **ResolRRH** se sobrecargue, ocasionando un tiempo de ejecución poco satisfactorio. En dichos casos, lo ideal es aislar del código de **ResolRRH**, la función que permite encontrar $m+1$ elementos de a_n, lo cual permitiría al usuario correr manualmente **FindSequenceFunction** sobre una cantidad de elementos **mm_** que se escogería directamente. Esta función corresponde a:

```
SolveRRHM[mm_, Coeficien_List, ConditionInitia_List] :=
 Module[{L = {}, ElementoSucesionH},
  ElementoSucesionH[Coeficient_List, ConditionInitial_List, m_] :=
   Module[{CD = Reverse[ConditionInitial],
     h = Dimensions[ConditionInitial][[1]] - 1, Matriz},
    Matriz[Coeficientes_List, nn_] :=
     Module[{Identidad = IdentityMatrix[Dimensions[Coeficientes][[1]]],
       A = {Coeficientes}, i, CF},
   For[i = 1, i ≤ Dimensions[Coeficientes][[1]] - 1,
    A = Append[A, Identidad[[i]]]; i++];
   CF[L_List, a_] := Block[{n = a}, L]; Return[CF[A, nn]]];
A[n_] := Matriz[Coeficient, n];
If[m > h, Elemento = A[0].Transpose[{CD}];
 For[i = 1, i ≤ m - 1, Elemento = Dot[A[i], Elemento]; i++];
 Return[Elemento[[h + 1, 1]]], Return[ConditionInitial[[m + 1]]]]];
For[i = 0, i ≤ mm,
 L = Append[L, ElementoSucesionH[Coeficien, ConditionInitia, i]]; i++];
Return[FunctionExpand[FindSequenceFunction[L, n]]]]
```

Es importante aclarar, cómo el método **ResolRRH** puede tomar un tiempo de ejecución significativo dada la complejidad que implica para el programa *Mathematica*, encontrar la solución explícita. El lector puede observar inclusive, en la resolución de algunas relaciones de recurrencia, una ecuación diferencial que representa la función encontrada por el ordenador.

Un aspecto interesante a destacar en los ejemplos expuestos previamente, lo constituye la comparación del método matricial y la función explícita que resuelve cada relación de recurrencia. En los ejercicios se concluyó que su velocidad de respuesta es muy similar. Ciertamente, el desenlace es curioso dado que se esperaría un mejor rendimiento en la solución explícita. Pese a ello, los experimentos verifican que el algoritmo establecido en (3) es bastante competente para hallar los elementos de una sucesión, definida por una relación de recurrencia homogénea lineal.

También para finalizar esta sección, es fundamental indicar que los métodos **ResolRRH** y **SolveRRHM** brindan buenos resultados en la mayor parte de relaciones de recurrencia homogéneas lineales, donde no aparecen funciones trascendentales. Esta advertencia es importante para el lector, pues si la relación de recurrencia contiene un logaritmo, o bien, una función trigonométrica, el comando **FindSequenceFunction** no suele proporcionar un resultado y como consecuencia de ello, tampoco **ResolRRH** y **SolveRRHM**.

A través de las ideas expuestas con anterioridad, es posible realizar un recorrido similar, para resolver relaciones de recurrencia lineales no homogéneas, lo cual constituye una futura segunda parte del artículo.

Conclusiones

Los métodos de trabajo expuestos en el presente artículo, representan un esfuerzo por buscar algoritmos novedosos que permitan resolver computacionalmente relaciones de recurrencia de cualquier tipo. Lo anterior, resulta una búsqueda muy ambiciosa pero necesaria, ante los diversos problemas que demandan el planteamiento y la exigencia de resolución, de una relación de recurrencia.

A pesar de las limitaciones de los algoritmos empleados, éstos ofrecen una buena alternativa específicamente en relaciones de recurrencia lineales homogéneas, no homogéneas, con o sin coeficientes constantes. Además, en casos particulares, las funciones compartidas pueden resultar caminos viables hacia la exploración de relaciones de recurrencia no lineales.

Referencias bibliográficas

Johnsonbaugh, R. (2005). *Matemáticas discretas*. México, DF: Pearson Prentice Hall.

Kolman, B., Busby, R. y Ross, S. (1997). *Estructuras de matemáticas discretas para computación*. México, DF: Prentice-Hall Hispanoamericana.

Monge, J. y Vílchez, E. (2001). Valores propios y las sucesiones definidas de forma recursiva. *Revista Virtual Matemática, Educación e Internet*, *2*(2).

Rosen, K. (2007). Discrete mathematics and its applications. USA: Mc. Graw-Hill.

Vílchez, E. (2004). Resolución de sucesiones definidas por una relación de recurrencia homogénea lineal con valores propios de multiplicidad algebraica mayor estricta que uno. *Revista Virtual Matemática, Educación e Internet*, *5*(2).

Vílchez, E. (2009). Resolución de relaciones de recurrencia lineales no homogéneas con coeficientes constantes a través de valores y vectores propios. *Revista Virtual Matemática, Educación e Internet*, *10*(1).