Uniciencia
E-ISSN: 2215-3470
revistauniciencia@una.cr
Universidad Nacional
Costa Rica

Vílchez-Quesada, Enrique
Resolución de relaciones de recurrencia con apoyo de Wolfram Mathematica
Uniciencia, vol. 29, núm. 1, enero, 2015, pp. 16-41
Universidad Nacional
Heredia, Costa Rica

Uniciencia. Vol. 29, No. 1, pp. 16-41. Enero, 2015. ISSN Electrónico: 2215-3470
URL: www.revistas.una.ac.cr/uniciencia DOI: http://dx.doi.org/10.15359/ru.29-1.2
Email: revistauniciencia@una.cr

Resolución de relaciones de recurrencia con apoyo de *Wolfram Mathematica*

Solving recurrence relations supported by Wolfram Mathematica

Enrique Vílchez-Quesada

enrique.vilchez.quesada@una.cr

Escuela de Informática

Universidad Nacional. Heredia, Costa Rica.

Fecha de recepción del artículo: ***22 de abril de 2014.***
Fecha de revisión del artículo: ***5 de junio de 2014.***
Fecha de aprobación del artículo: ***5 de julio de 2014.***

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 *Wolfram 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 claves: relaciones; recurrencia; solución; software; *Mathematica*.

Abstract

This paper introduces some algorithms for solving linear relationships, homogeneous and non-homogeneous recurrence with constant and non-constant coefficients, using software as the main resource in solving processes. The *Mathematica* commercial application has provided the technical support necessary for the implementation of the methods used. It also presents other examples of recurrence relations, showing the effectiveness and limitations of the algorithms created by the author and programmed in *Mathematica* environment that provides.

Keywords: relations; recurrence; solution; software; *Mathematica*.

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.

Uniciencia. Vol. 29, No. 1, pp. 16-41. Enero, 2015. ISSN Electrónico: 2215-3470
URL: www.revistas.una.ac.cr/uniciencia DOI: http://dx.doi.org/10.15359/ru.29-1.2
Email: revistauniciencia@una.cr

Los procedimientos clásicos de resolución expuestos en la mayor parte de la bibliografía (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 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 coeficientes constantes o sin estos.

Aspectos generales

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

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

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 $\forall j, \quad j \in IN \cup \{0\}, 0 \le j \le k-1$.

El lector debe observar que si todos los $\beta_j(n)$ son números reales, la relación de

Uniciencia. Vol. 29, No. 1, pp. 16-41. Enero, 2015. ISSN Electrónico: 2215-3470
URL: www.revistas.una.ac.cr/uniciencia DOI: http://dx.doi.org/10.15359/ru.29-1.2
Email: revistauniciencia@una.cr

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:

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

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 (1) 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:

Uniciencia. Vol. 29, No. 1, pp. 16-41. Enero, 2015. ISSN Electrónico: 2215-3470
URL: www.revistas.una.ac.cr/uniciencia DOI: http://dx.doi.org/10.15359/ru.29-1.2
Email: revistauniciencia@una.cr

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

Lo anterior permite concluir que:

$$X_n = \prod_{j=0}^{n-1} \mathbf{A}(j)\cdot X_0,\ \forall n \in IN \qquad (2)$$

Como:

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

entonces en (2), 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 (2) constituye un algoritmo que permite determinar los elementos de a_n más rápidamente en comparación con la relación de recurrencia original 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 (2) que retorna la última fila de la matriz es:

Uniciencia. Vol. 29, No. 1, pp. 16-41. Enero, 2015. ISSN Electrónico: 2215-3470
URL: www.revistas.una.ac.cr/uniciencia DOI: http://dx.doi.org/10.15359/ru.29-1.2
Email: revistauniciencia@una.cr

```
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]]
```

Uniciencia. Vol. 29, No. 1, pp. 16-41. Enero, 2015. ISSN Electrónico: 2215-3470
URL: www.revistas.una.ac.cr/uniciencia DOI: http://dx.doi.org/10.15359/ru.29-1.2
Email: revistauniciencia@una.cr

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 cuál método fue más eficiente en cada caso. 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
```

Uniciencia. Vol. 29, No. 1, pp. 16-41. Enero, 2015. ISSN Electrónico: 2215-3470
URL: www.revistas.una.ac.cr/uniciencia DOI: http://dx.doi.org/10.15359/ru.29-1.2
Email: revistauniciencia@una.cr

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 (2) es más eficiente.

Ejemplo 2. Considere la sucesión definida por la relación de recurrencia $a_{n+3} = 3a_{n+2} - 4a_n$ sujeta a las condiciones $a_0 = 7$, $a_1 = 5$ y $a_2 = 3$. Compare la velocidad de respuesta de la relación de recurrencia y la función **ElementoSucesiónH**.

Solución 2. Por las limitaciones sintácticas del software, en la relación de recurrencia inicial se debe sustituir *n* por *n-3*, de donde:

$$a_n = 3a_{n-1} - 4a_{n-3}$$

Por otra parte, en *Mathematica*:

In :=

```
ClearAll[a]
a[n_] := 3 * a[n - 1] - 4 * a[n - 3]
a[0] = 7;
a[1] = 5;
a[2] = 3;

Table[ElementoSucesionH[{3, 0, -4}, {7, 5, 3}, k] == a[k], {k, 0, 30}]

PruebaH[30, {3, 0, -4}, {7, 5, 3}]
```

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: 20

El algoritmo matricial fue más
   efectivo la cantidad de ejecuciones igual a: 10
```

En la salida arrojada por el programa, el algoritmo matricial fue más eficiente 10 veces. Pese a ello, las 10 ejecuciones donde fue más efectivo ocurrieron en los 10 cálculos finales, por lo que podríamos interpretar que a partir del elemento 21 este método se mostró más rápido.

Los ejercicios anteriores evidencian de forma empírica la efectividad de (2) sobre la relación de recurrencia preliminar.

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, 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]]]
```

Se abordan algunos ejemplos de uso.

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

ISSN Electrónico: 2215-3470
DOI: http://dx.doi.org/10.15359/ru.29-1.2

Solución 3. En *Mathematica*:

In :=

$\text{Re}\, solRRH[\{7,-10\},\{3,1\}]$

Out =

```
El valor mínimo para el cual se retorna una función es: 5
La lista de elementos usada es: {3, 1, -23, -171, -967, -5059}
La solución de la relación de recurrencia es: 1/3 (7×2^n - 5^n)
Una lista de elementos generada por la función anterior es:
 {3., 1., -23., -171., -967., -5059.}
```

Si se desea comparar la velocidad de respuesta de la solución de la relación de recurrencia con respecto a **ElementoSucesiónH**, un código de programación afín es el siguiente:

```
PruebaHS[m_, Con_List, Condici_List] :=
 Module[{k, alf = 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[f[k + 1]]];
   If[L1 > L2 && L1 ≠ L2, Print["Es mejor el algoritmo funcional"];
    alf++, If[L1 == L2, Print["Tienen el mismo tiempo de ejecución"];
     mt++, Print["Es mejor el algoritmo matricial"]; alm++]]; k++];
  Print[
   "El algoritmo funcional fue más efectivo la cantidad de
     ejecuciones igual a: ", alf];
  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]]
```

Al emplear **PruebaHS**, se obtiene:

In :=

```
f[n_] := 1/3 (7 × 2^n - 5^n)

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

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

Out =

⋮

```
Se encontró el elemento de la sucesión: 28
Tienen el mismo tiempo de ejecución
Se encontró el elemento de la sucesión: 29
Tienen el mismo tiempo de ejecución
Se encontró el elemento de la sucesión: 30
Tienen el mismo tiempo de ejecución
El algoritmo funcional fue
   más efectivo la cantidad de ejecuciones igual a: 1
Tuvieron el mismo tiempo la cantidad de ejecuciones igual a: 30
El algoritmo matricial fue
   más efectivo la cantidad de ejecuciones igual a: 0
```

Los resultados parecen mostrar pocas diferencias en tiempo de ejecución, entre la solución explícita y el algoritmo matricial.

Ejemplo 4. Sea dada la relación de recurrencia $a_{n+3} = 5n^3 a_{n+2} + n^2 a_{n+1} - n a_n$ sujeta a las condiciones $a_0 = 1$, $a_1 = 1$ y $a_2 = 2$. Resuelva a_n y compare la velocidad de respuesta con respecto a la función **ElementoSucesiónH**.

Solución 4. En *Mathematica*:

In :=

$$\text{Re}\, solRRH[\{5n^3, n^2, -n\}, \{1,1,2\}]$$

Out =

```
El valor mínimo para el cual se retorna una función es: 22
La lista de elementos usada es:
 {1, 1, 2, 0, 1, 36, 4869, 1 558 652, 974 279 045, 1 052 277 450 858,
  1 804 703 556 984 111, 4 620 108 443 841 946 712, 16 840 441 449 321 514 420 509,
  84 202 669 239 404 920 727 375 090, 560 370 801 430 833 922 461 664 691 707,
  4 841 615 849 344 690 266 985 509 505 315 332,
  53 185 244 806 622 229 693 656 453 967 765 944 333,
  729 702 507 695 718 271 736 227 202 683 148 492 369 934,
  12 313 741 783 972 703 087 810 344 763 975 490 913 500 381 195,
  252 185 618 538 751 965 425 327 469 565 283 437 531 017 598 467 376,
  6 194 943 278 063 412 656 153 534 456 441 308 569 420 707 783 590 967 917,
  180 644 627 696 247 872 256 962 353 900 348 110 830 390 040 182 771 521 028 034,
  6 195 209 743 212 552 633 192 256 511 095 296 014 318 879 603 630 569 855 624 963 923}

La solución de la relación de recurrencia es:
 DifferenceRoot[Function[{ẏ, ṅ},
    {(-1 + ṅ) ẏ[ṅ] - (-1 + ṅ)^2 ẏ[1 + ṅ] - 5 (-1 + ṅ)^3 ẏ[2 + ṅ] + ẏ[3 + ṅ] == 0,
     ẏ[1] == 1, ẏ[2] == 1, ẏ[3] == 2}]][n]
Una lista de elementos generada por la función anterior es:
 {1., 1., 2., 0., 1., 36., 4869., 1.55865×10^6, 9.74279×10^8,
  1.05228×10^12, 1.8047×10^15, 4.62011×10^18, 1.68404×10^22, 8.42027×10^25,
  5.60371×10^29, 4.84162×10^33, 5.31852×10^37, 7.29703×10^41,
  1.23137×10^46, 2.52186×10^50, 6.19494×10^54, 1.80645×10^59, 6.19521×10^63}
```

Es importante aclarar en este ejercicio, cómo el método **ResolRRH** toma 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 la relación de recurrencia, una ecuación diferencial que representa la función encontrada por el ordenador. La respuesta anterior, también se genera en el software de manera más rápida haciendo uso de **SolveRRHM:**

In :=

$$SolveRRHM[22, \{5n^3, n^2, -n\}, \{1,1,2\}]$$

Uniciencia. Vol. 29, No. 1, pp. 16-41. Enero, 2015. ISSN Electrónico: 2215-3470
URL: www.revistas.una.ac.cr/uniciencia DOI: http://dx.doi.org/10.15359/ru.29-1.2
Email: revistauniciencia@una.cr

Out =

```
DifferenceRoot[Function[{ẏ, ṅ},
   {(-1 + ṅ) ẏ[ṅ] - (-1 + ṅ)^2 ẏ[1 + ṅ] - 5 (-1 + ṅ)^3 ẏ[2 + ṅ] + ẏ[3 + ṅ] == 0,
    ẏ[1] == 1, ẏ[2] == 1, ẏ[3] == 2}]][n]
```

Desde luego, sin haber corrido el método **ResolRRH** es imposible poder adivinar que en el valor veintidós, **SolveRRHM** converge. Por otra parte, al emplear **PruebaHS** para comparar eficiencias, se obtiene:

In :=

```
f[n_] :=
 DifferenceRoot[
   Function[{ẏ, ṅ},
    {(-1 + ṅ) ẏ[ṅ] - (-1 + ṅ)^2 ẏ[1 + ṅ] - 5 (-1 + ṅ)^3 ẏ[2 + ṅ] + ẏ[3 + ṅ] == 0,
     ẏ[1] == 1, ẏ[2] == 1, ẏ[3] == 2}]][n]

Table[ElementoSucesionH[{5 n^3, n^2, -n}, {1, 1, 2}, k] == f[k + 1], {k, 0, 30}]

PruebaHS[30, {5 n^3, n^2, -n}, {1, 1, 2}]
```

Out =

⋮

```
Se encontró el elemento de la sucesión: 28
Tienen el mismo tiempo de ejecución
Se encontró el elemento de la sucesión: 29
Tienen el mismo tiempo de ejecución
Se encontró el elemento de la sucesión: 30
Tienen el mismo tiempo de ejecución
El algoritmo funcional fue
   más efectivo la cantidad de ejecuciones igual a: 4
Tuvieron el mismo tiempo la cantidad de ejecuciones igual a: 27
El algoritmo matricial fue
   más efectivo la cantidad de ejecuciones igual a: 0
```

Uniciencia. Vol. 29, No. 1, pp. 16-41. Enero, 2015. ISSN Electrónico: 2215-3470
URL: www.revistas.una.ac.cr/uniciencia DOI: http://dx.doi.org/10.15359/ru.29-1.2
Email: revistauniciencia@una.cr

De igual manera en este caso, tanto el algoritmo matricial como la función explícita presentan velocidades similares.

Un aspecto interesante de 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 (2) 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**.

La siguiente sección realiza un recorrido similar, para resolver relaciones de recurrencia lineales no homogéneas.

Relaciones de recurrencia lineales no homogéneas

Esta sección instaura un método para encontrar más rápidamente los elementos de una sucesión definida por una relación de recurrencia lineal no homogénea. Los fundamentos teóricos se sustentan en un conjunto de ideas demostradas formalmente en Vílchez (2009). El aporte principal de lo compartido reside en presentar ciertas funciones elaboradas con el software *Mathematica*, para resolver relaciones de recurrencia de este tipo con coeficientes constantes o sin estos.

Aspectos generales

Una relación de recurrencia lineal no homogénea es aquella de la forma:

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

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 $\forall j,\ j \in IN \cup \{0\}, 0 \le j \le k-1$. Si todos los $\beta_j(n)$ son números, el lector preverá que la relación de recurrencia comprendida

Uniciencia. Vol. 29, No. 1, pp. 16-41. Enero, 2015. ISSN Electrónico: 2215-3470
URL: www.revistas.una.ac.cr/uniciencia DOI: http://dx.doi.org/10.15359/ru.29-1.2
Email: revistauniciencia@una.cr

es de coeficientes constantes.

El método aquí propuesto se fundamenta en el siguiente sistema de ecuaciones lineales:

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

El cual, matricialmente puede expresarse así:

$$X_{n+1} = \mathbf{A}(n) \cdot X_n + \mathbf{B}(n) \qquad (3)$$

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:

$$X_n = \begin{pmatrix} a_{n+(k-1)} \\ a_{n+(k-2)} \\ \vdots \\ a_n \end{pmatrix}, \mathbf{B}(n) = \begin{pmatrix} f(n) \\ 0 \\ \vdots \\ 0 \end{pmatrix} \text{ matrices } k \times 1$$

En (3) por el método iterativo, X_n puede ser hallado 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:

Uniciencia. Vol. 29, No. 1, pp. 16-41. Enero, 2015. ISSN Electrónico: 2215-3470
URL: www.revistas.una.ac.cr/uniciencia DOI: http://dx.doi.org/10.15359/ru.29-1.2
Email: revistauniciencia@una.cr

$$\begin{cases} X_1 = \mathbf{A}(0)\cdot X_0 + \mathbf{B}(0) \\ X_2 = \mathbf{A}(1)\cdot X_1 + \mathbf{B}(1) = \mathbf{A}(1)\cdot \mathbf{A}(0)\cdot X_0 + \mathbf{A}(1)\cdot \mathbf{B}(0) + \mathbf{B}(1) \\ X_3 = \mathbf{A}(2)\cdot X_2 + \mathbf{B}(2) = \mathbf{A}(2)\cdot \mathbf{A}(1)\cdot \mathbf{A}(0)\cdot X_0 + \mathbf{A}(2)\cdot \mathbf{A}(1)\cdot \mathbf{B}(0) + \mathbf{A}(2)\cdot \mathbf{B}(1) + \mathbf{B}(2) \\ \vdots \\ X_n = \prod_{j=0}^{n-1} \mathbf{A}(j)\cdot X_0 + \sum_{k=1}^{n-1}\prod_{j=k}^{n-1}\left[\mathbf{A}(j)\cdot \mathbf{B}(k-1)\right] + \mathbf{B}(n-1) \end{cases}$$

Lo anterior permite construir la ecuación:

$$X_n = \prod_{j=0}^{n-1} \mathbf{A}(j)\cdot X_0 + \sum_{k=1}^{n-1}\prod_{j=k}^{n-1}\left[\mathbf{A}(j)\cdot \mathbf{B}(k-1)\right] + \mathbf{B}(n-1),\ \forall n \in IN \qquad (4)$$

Como:

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

entonces en (4), a_n corresponde a la última fila de la matriz resultante. En la expresión (4), por lo tanto, es posible excluir a $\mathbf{B}(n-1)$, pues contribuye con un cero en la suma que deriva la última fila de la matriz. Luego, (4) conforma un algoritmo que calcula los elementos de a_n más rápidamente en comparación con la relación de recurrencia original. El software *Mathematica* se utilizará en este sentido, como una herramienta para el cómputo de (4).

Encontrando elementos de la sucesión y comparando velocidades

En *Mathematica* una función que retorna el resultado de la última fila de (4) es:

```
ElementoSucesionNH[Coefi_List, ConditInitial_List, F_List, w_] :=
 Module[{nc = Dimensions[Coefi][[1]], ElementoSucesionH, Sumatoria},
  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]]]]];

Sumatoria[Coeficient_List, Funct_List, ww_] :=
 Module[{Matriz, MatrizB, t = Dimensions[Coeficient][[1]],
   Productoria, s}, 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];
  MatrizB[Funcion_List, filas_, nn_] :=
   Module[{MB = ConstantArray[0, {filas, 1}], CF},
    MB[[1, 1]] = Funcion[[1]]; CF[L_List, a_] := Block[{n = a}, L];
    Return[CF[MB, nn]]]; B[n_] := MatrizB[Funct, t, n];
  Productoria[ini_, final_] :=
   Module[{i}, p = Dot[A[ini], B[ini - 1]];

     For[i = ini + 1, i ≤ final, p = Dot[A[i], p]; i++]; Return[p]];
    s = Sum[Productoria[k, ww - 1], {k, 1, ww - 1}]; Return[s[[t, 1]]]];
  If[w ≤ nc - 1, ElementoSucesionH[Coefi, ConditInitial, w],
   Return[ElementoSucesionH[Coefi, ConditInitial, w] +
     Sumatoria[Coefi, F, w]]]]
```

Los argumentos **Coefi_List**, **ConditInitial_List** y **F_List** definen la relación de recurrencia mediante un vector de coeficientes $\beta_j(n)$, una lista de condiciones iniciales y la

función $f(n)$, respectivamente. En **ElementoSucesiónNH** el parámetro **w_** simboliza un número natural sobre el cual se desea obtener de la sucesión $(a_n)_{n\in IN\cup\{0\}}$, el término a_w. A pesar de todos los cálculos que involucra **ElementoSucesiónNH**, esta función resulta ser más rápida que la relación de recurrencia original. Lo anterior, se puede comprobar experimentalmente a través del siguiente programa creado en *Mathematica*:

```
PruebaNH[m_, Con_List, Condici_List, Fun_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[ElementoSucesionNH[Con, Condici, Fun, 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]]
```

La instrucción **Timing** en **PruebaNH** encuentra el tiempo de ejecución en el cálculo de $m+1$ elementos de a_n, comparando la eficiencia de **ElementoSucesiónNH** con respecto a la relación de recurrencia inicial. **PruebaNH** caso por caso, determina cuál algoritmo muestra un tiempo de ejecución menor.

Se puede verificar experimentalmente la efectividad de (4) sobre la ecuación recursiva de una sucesión dada.

Resolviendo relaciones de recurrencia

Por medio del uso de la función **ElementoSucesiónNH** y el comando del software *Mathematica* **FindSequenceFunction,** es posible construir un método que busca encontrar de manera explícita, los elementos de una sucesión dada por una relación de recurrencia lineal no homogénea. Esta forma de razonamiento es similar a lo propuesto en **ResolRRH**, con la excepción de recibir una relación de recurrencia lineal no homogénea. La función **ResolRRNH** realiza lo anteriormente descrito.

```
ResolRRNH[Co_List, Con_List, Func_List] :=
 Module[{ElementoSucesionNH, SolveR, SolveRR},
  ElementoSucesionNH[Coefi_List, ConditInitial_List, F_List, w_] :=
   Module[{nc = Dimensions[Coefi][[1]], ElementoSucesionH, Sumatoria},
    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]]]]];
   Sumatoria[Coeficient_List, Funct_List, ww_] :=
    Module[{Matriz, MatrizB, t = Dimensions[Coeficient][[1]],
      Productoria, s}, 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];
     MatrizB[Funcion_List, filas_, nn_] :=
      Module[{MB = ConstantArray[0, {filas, 1}], CF},
```

```
      MB[[1, 1]] = Funcion[[1]]; CF[L_List, a_] := Block[{n = a}, L];
      Return[CF[MB, nn]]]; B[n_] := MatrizB[Funct, t, n];
    Productoria[ini_, final_] :=
     Module[{i}, p = Dot[A[ini], B[ini - 1]];
      For[i = ini + 1, i ≤ final, p = Dot[A[i], p]; i++]; Return[p]];
    s = Sum[Productoria[k, ww - 1], {k, 1, ww - 1}]; Return[s[[t, 1]]]];
  If[w ≤ nc - 1, ElementoSucesionH[Coefi, ConditInitial, w],
   Return[ElementoSucesionH[Coefi, ConditInitial, w] +
     Sumatoria[Coefi, F, w]]]];
SolveR[mm_, Coeficien_List, ConditionInitia_List, Fun_List, n_] :=
 Module[{L = {}},
  For[i = 0, i ≤ mm,
   L = Append[L, ElementoSucesionNH[Coeficien, ConditionInitia,
       Fun, i]]; i++];
  Return[FunctionExpand[FindSequenceFunction[L, n]]]];

SolveRR[m_, Coeficient_List, ConditionInitial_List, Fun_List, n_] :=
 Module[{L = {}},
  For[i = 0, i ≤ m,
   L = Append[L, ElementoSucesionNH[Coeficient, ConditionInitial,
       Fun, i]]; i++]; Print["La lista de elementos usada es: ", L];
  Return[FunctionExpand[FindSequenceFunction[L, n]]]]; j = 1;
f[n_] := Evaluate@SolveR[j, Co, Con, Func, n];
While[NumericQ[f[1]] == False, j++;
 f[n_] := Evaluate@SolveR[j, Co, Con, Func, 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, Func, 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}]]]]]
```

ResolRRNH determina una lista mínima de elementos de la sucesión recursiva a_n para la cual **FindSequenceFunction** busca una función explícita. Esto puede provocar una sobrecarga, al igual que en la instrucción **ResolRRH**, suscitando un tiempo de ejecución poco satisfactorio. Lo ideal, algunas veces, para solucionar este conflicto, reside en aislar del código de **ResolRRNH**, la función que permite encontrar $m+1$ elementos de a_n, facilitando correr manualmente **FindSequenceFunction** sobre una cantidad **mm_** de elementos que el usuario elegiría. Esta función corresponde a:

```
SolveRRNHM[mm_, Coef_List, ConIn_List, Fun_List] :=
 Module[{L = {}, ElementoSucesionNH},
  ElementoSucesionNH[Coefi_List, ConditInitial_List, F_List, w_] :=
   Module[{nc = Dimensions[Coefi][[1]], ElementoSucesionH, Sumatoria},
    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]]]]];
    Sumatoria[Coeficient_List, Funct_List, ww_] :=
     Module[{Matriz, MatrizB, t = Dimensions[Coeficient][[1]],
       Productoria, s}, 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];
       MatrizB[Funcion_List, filas_, nn_] :=
        Module[{MB = ConstantArray[0, {filas, 1}], CF},

          MB[[1, 1]] = Funcion[[1]]; CF[L_List, a_] := Block[{n = a}, L];
          Return[CF[MB, nn]]]; B[n_] := MatrizB[Funct, t, n];
        Productoria[ini_, final_] :=
         Module[{i}, p = Dot[A[ini], B[ini - 1]];
          For[i = ini + 1, i ≤ final, p = Dot[A[i], p]; i++]; Return[p]];
        s = Sum[Productoria[k, ww - 1], {k, 1, ww - 1}]; Return[s[[t, 1]]]];
      If[w ≤ nc - 1, ElementoSucesionH[Coefi, ConditInitial, w],
       Return[ElementoSucesionH[Coefi, ConditInitial, w] +
         Sumatoria[Coefi, F, w]]]];
   For[i = 0, i ≤ mm, L = Append[L, ElementoSucesionNH[Coef, ConIn, Fun, i]];
    i++]; Return[FunctionExpand[FindSequenceFunction[L, n]]]]
```

Prosigue un ejemplo relacionado con el uso de estas funciones.

Ejemplo 5. Resuelva la sucesión recursiva $a_{n+2} = 7a_{n+1} - 10a_n + n$ sujeta a las condiciones $a_0 = 3$ y $a_1 = 1$. Compare la velocidad de convergencia de la solución explícita y la función **ElementoSucesiónNH**.

Solución 5. En el software:

In :=

$$\text{Re}\, solRRNH[\{7,-10\},\{3,1\},\{n\}]$$

Out =

```
El valor mínimo para el cual se retorna una función es: 7
La lista de elementos usada es:
 {3, 1, -23, -170, -958, -5003, -25 437, -128 024}
La solución de la relación de recurrencia es: 1/240 (15 + 65 × 2^(3+n) - 79 × 5^n + 60 n)
Una lista de elementos generada por la función anterior es:
 {3., 1., -23., -170., -958., -5003., -25 437., -128 024.}
```

Si el objetivo ahora es comparar la velocidad de respuesta de la solución de la relación de recurrencia con respecto a **ElementoSucesiónNH**, un código de análisis es:

```
PruebaNHS[m_, Con_List, Condici_List, Fun_List] :=
 Module[{k, alf = 0, mt = 0, alm = 0},
  For[k = 0, k ≤ m, Print["Se encontró el elemento de la sucesión: ", k];
   ClearSystemCache[];
   L1 = First[Timing[ElementoSucesionNH[Con, Condici, Fun, k]]];
   ClearSystemCache[]; L2 = First[Timing[f[k + 1]]];
   If[L1 > L2 && L1 ≠ L2, Print["Es mejor el algoritmo funcional"];
    alf++, If[L1 == L2, Print["Tienen el mismo tiempo de ejecución"];
     mt++, Print["Es mejor el algoritmo matricial"]; alm++]]; k++];
  Print[
   "El algoritmo funcional fue más efectivo la cantidad de
     ejecuciones igual a: ", alf];
  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]]
```

ISSN Electrónico: 2215-3470
DOI: http://dx.doi.org/10.15359/ru.29-1.2

Al utilizar **PruebaNHS** se obtiene:

In :=

$$f[n\_] := \frac{1}{240}\left(15 + 65 \times 2^{3+n} - 79 \times 5^{n} + 60\, n\right)$$

```
Table[ElementoSucesionNH[{7, -10}, {3, 1}, {n}, k] == f[k + 1], {k, 0, 30}]
```

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

Out =

⋮

```
Se encontró el elemento de la sucesión: 28
Es mejor el algoritmo funcional
Se encontró el elemento de la sucesión: 29
Es mejor el algoritmo funcional
Se encontró el elemento de la sucesión: 30
Es mejor el algoritmo funcional
El algoritmo funcional fue más
   efectivo la cantidad de ejecuciones igual a: 14
Tuvieron el mismo tiempo la cantidad de ejecuciones igual a: 17
El algoritmo matricial fue
   más efectivo la cantidad de ejecuciones igual a: 0
```

El Out muestra un tiempo de ejecución más adecuado en la solución explícita.

Un aspecto importante de destacar en el ejemplo anterior reside en la comparación del método matricial y la función explícita que resuelve cada relación de recurrencia. En el ejercicio se concluyó que la velocidad de convergencia es mejor en la función explícita. El experimento verifica que el algoritmo expuesto en (4) es menos competente para hallar los elementos de una sucesión, definida por una relación de recurrencia lineal no homogénea.

Finalizando esta sección, es fundamental indicar que los métodos **ResolRRNH** y **SolveRRNHM** brindan resultados muy aceptables en la mayor parte de relaciones de recurrencia lineales no homogéneas, donde no aparecen funciones trascendentales. Esta advertencia es esencial, pues si la relación de recurrencia contiene un logaritmo, o bien, una función trigonométrica, el comando **FindSequenceFunction,** como ya se había señalado, con frecuencia no proporciona un resultado y, por lo tanto, tampoco lo hacen **ResolRRNH** y **SolveRRNHM**.

ISSN Electrónico: 2215-3470
DOI: http://dx.doi.org/10.15359/ru.29-1.2

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

Referencias

Calderón, S. y Morales, M. (2000). *Relaciones de recurrencia*. Costa Rica: Taller de publicaciones del Instituto Tecnológico de Costa Rica.

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

Kolman, B., Busby, R. y Ross, S. (1997). *Estructuras de matemáticas discretas para computación.* México: 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), 1-16. Descargado de http://www.tec-digital.itcr.ac.cr/revistamatematica/ContribucionesN22001/Monge/final/index.html

Rosen, K. (2007). *Discrete Mathematics and its applications* [Matematica discrete y sus aplicaciones.]. 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). 1-16. Descargado de http://www.tec-digital.itcr.ac.cr/revistamatematica/contribuccionesV5n2dic004/Vilchez-recurrencia/ARecurrencia2WEB/index.html

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). 1-29. Descargado de http://www.tec-digital.itcr.ac.cr/revistamatematica/ARTICULOS_V10_N1_2009/RESOLUCION_RELACIONES_RECURRENCIA/index.htm