Simulación y Análisis de Modelos Poblacionales Matriciales

FAMAF, Universidad Nacional de Córdoba

R. J. Corin[1]

Director: Dr. Oscar H. Bustos

[1]Correo Electrónico: rcorin@hal.famaf.unc.edu.ar

Resumen

Los modelos poblacionales matriciales son herramientas poderosas y útiles para el estudio de dinámicas poblacionales, con aplicaciones en la ecología, epidemiología y demografía. En el presente trabajo se diseñará e implementará un software que permita el estudio de modelos poblacionales matriciales permitiendo su construcción, análisis e interpretación. Este trabajo está dividido en tres partes: primero se explican los modelos poblacionales matriciales en forma teórica; qué son, cómo se definen, qué tipo de análisis permiten. Luego se explica el diseño de un lenguaje de programación (llamado PML) desarrollado para permitir la simulación y el análisis de los modelos antes explicados; se dará su sintaxis y semántica. Finalmente, contando con una implementación funcional de este lenguaje, se analizarán diferentes casos poblacionales, utilizando simulación y rutinas numéricas. Esta implementación permitirá interpretar los resultados obtenidos por medio de gráficos de líneas, barras e imágenes de sensibilidad.

Contents

Chapter 1

Introducción

Los modelos poblacionales matriciales son herramientas poderosas y útiles para el estudio de dinámicas poblacionales, con aplicaciones en la ecología, epidemiología y demografía [1][2].

Consideramos poblaciones estructuradas (por edad, sexo, status reproductivo, espacio, etc.) cuyos parámetros son conocidos mediante trabajo de campo. Estas poblaciones pueden ser fácilmente analizadas utilizando modelos poblacionales matriciales.

En el presente trabajo se diseñará e implementará un software que permita el estudio de modelos poblacionales matriciales permitiendo su construcción, análisis e interpretación.

Existen paquetes diseñados para facilitar la tarea de simular modelos poblacionales matriciales. Una solución ampliamente utilizada es la familia de programas RAMAS[1]. Otro paquete es ULM ([3][5]). En ULM, el usuario describe el modelo a estudiar en un archivo que luego es interpretado y, si está bien formado, se presenta una serie de comandos disponibles para el análisis y la simulación del modelo.

Si bien estos paquetes facilitan la construcción y análisis de los modelos, esta simplicación lleva un precio. Muchos análisis poblacionales requieren tener en cuenta detalles únicos, que difícilmente pueden ser anticipados por un paquete que ofrece finitas opciones en su menú. En el otro extremo, se pueden encontrar ambientes de propósito general, como ser Matlab[2].

En ULM se opta por describir el modelo a estudiar utilizando un lenguaje específico reducido, pero solamente se utiliza este lenguaje para la descripción de este, quedando la simulación y análisis para una etapa posterior. En este trabajo, en cambio, se pretende desarrollar un lenguaje de programación (llamado PML) como motor primario, permitiendo al usuario un total control, desde la descripción y definición del modelo hasta la simulación y análisis de los resultados.

Teniendo una implementación de PML se podrán desarrollar librerías, que al ser *incluídas* proveerán al usuario con ejemplos de funciones que implementan modelos o facilitan en gran medida su construcción, además de incluir otras rutinas de uso frecuente. Esto otorga no sólo transparencia sino también la posibilidad de modificar o crear nuevas implementaciones desde el principio.

[1] Applied Biomathematics, 100 North Country Rd., Setauket, New York, 11733, U.S.A.; http://www.ramas.com
[2] The MathWorks, Inc., 3 Apple Hill Drive, Natick MA, 01760, U.S.A.; http://www.mathworks.com

El desarrollo de este software será una herramienta poderosa y novedosa, ya que se permitirá la modelación de dinámicas poblacionales de manera precisa, y gracias a la existencia del lenguaje de programación como medio para describir los modelos se provee máxima flexibilidad y potencia, aunque siempre manteniendo el foco en los modelos matriciales poblacionales.

1.1 Objetivos

- Implementar un programa que sirva para la modelación de dinámicas poblacionales, permitiendo su construcción, análisis (simulación y obtención de resultados tanto gráficos como numéricos) e interpretación. Para llegar finalmente a este programa, antes se deberá:
  - Diseñar un lenguaje de programación que sirva para la correcta descripción de modelos poblacionales matriciales. Definir su sintaxis y semántica.
  - Brindar una implementación del lenguaje mencionado, implementando un parser para la sintaxis propuesta y un intérprete correspondiente.
- Contando con una implementación funcional del lenguaje PML, construir, analizar e interpretar diferentes programas de modelos para diferentes casos poblacionales.

Chapter 2

Los modelos matriciales

2.1 Grafos de *ciclo de vida*

Los grafos de ciclo de vida proveen una forma simple y gráfica de representar el ciclo de vida de una población. La construcción de dicho grafo es sencilla. Se deben seguir los siguientes pasos:

1. Elegir un conjuntos de *etapas*. Estas etapas representarán el "camino" que recorrerá cada individuo a lo largo de su existencia.
2. Elegir un intervalo de tiempo, que será el paso de tiempo que ocurre entre $(t, t+1)$. (la variable t de tiempo será discreta en nuestros modelos)
3. Crear un *nodo* por cada *etapa*. Numerar los nodos de 1 a k. (nos referiremos al nodo i como N_i)
4. Crear una arista dirigida del nodo N_i a N_j si un individuo en la etapa i en el tiempo t puede contribuir con individuos (por desarrollo o reproducción) a la etapa j en el tiempo $t+1$. Cada arista recibe un coeficiente a_{ij} en el arco de N_j a N_i dando el número de individuos en la etapa i en el tiempo $t+1$ por cada individuo en la etapa j en el tiempo t. Entonces se tiene:

$$n_i(t+1) = \sum_{j=1}^{k} a_{ij} n_j(t) \tag{1}$$

Cuando se construye el grafo, se debe pensar también cuál será la variable que determinará las etapas. Por ejemplo, si consideramos la variable "edad", tendremos típicamente etapas como ser "recién nacido", "joven", "adulto", etc. Otras variables posibles pueden ser el tamaño (o desarrollo) o la ubicación.

Supongamos que consideramos k etapas, y pensaremos en etapas clasificadas por la variable edad. Un grafo posible es el dado por la figura 2.1.

Estos grafos pueden ser vistos también como matrices $k \times k$, simplemente tomando cada a_{ij} (los coeficientes de las aristas) como elementos de la matriz, y dejando las restantes entradas en 0.

La matriz asociada al grafo de la figura 2.1 es como sigue:

$$\begin{pmatrix} \lambda_1 & \cdots & \lambda_{k-1} & \lambda_k \\ p_1 & 0 & 0 & 0 \\ 0 & \ddots & 0 & 0 \\ 0 & 0 & p_{k-1} & 0 \end{pmatrix}$$

De esta manera arribamos al modelo definido por Leslie en 1945 [6].

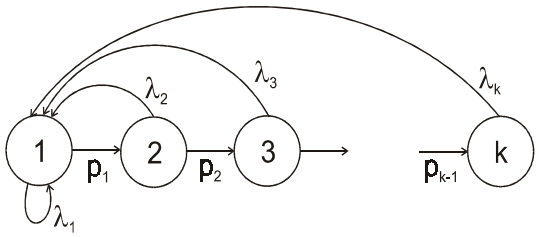

Figure 2.1: Grafo de Ciclo de Vida

2.2 Modelo de Leslie

Leslie definió en 1945 un modelo clasificado por edad, de la siguiente manera: Si tenemos k clases de edad, un individuo que comienza en la clase de edad i ($i = 1..k$), producirá λ_i hijos (esta cantidad es llamada $fertilidad$) durante el intervalo de tiempo considerado, y tendrá una probabilidad p_i de sobrevivir a la siguiente clase de edad ($0 \leq p_i \leq 1$).

La población en cuestión puede ser vista como un vector $\mathbf{n}(t)$, con $\mathbf{n}_i(t)$ la cantidad de individuos de edad i en el tiempo t.

De esta manera la dinámica de la población se puede determinar iterativamente por la siguiente ecuación matricial (llamada *ecuación de proyección*):

$$\mathbf{n}(t+1) = \mathbf{A}\mathbf{n}(t) \quad para\ t = 0, 1, ... \tag{2}$$

donde $\mathbf{A}$ es de la forma:

$$\mathbf{A} = \begin{pmatrix} \lambda_1 & \cdots & \lambda_{k-1} & \lambda_k \\ p_1 & 0 & 0 & 0 \\ 0 & \ddots & 0 & 0 \\ 0 & 0 & p_{k-1} & 0 \end{pmatrix} \tag{3}$$

($\mathbf{A}$ es llamada *matriz de proyección poblacional*).

La ecuación (2) puede ser vista como la versión matricial de (1).

En las siguientes modificaciones al modelo básico de Leslie, si bien varía la forma de los modelos, notar que siempre se mantiene (2).

2.3 Modificaciones al modelo de Leslie básico: Estocástico y Densodependiente

Consideramos casos en que la matriz $\mathbf{A}$ no es constante. Por ejemplo, las fertilidades λ_i pueden variar con respecto al tiempo. En este caso tendremos una nueva matriz $\mathbf{A}_t$ de la forma:

$$\mathbf{A}_t = \begin{pmatrix} h_1(t) & \cdots & h_{k-1}(t) & h_k(t) \\ p_1 & 0 & 0 & 0 \\ 0 & \ddots & 0 & 0 \\ 0 & 0 & p_{k-1} & 0 \end{pmatrix}$$

Luego tendremos una matriz $\mathbf{A}_t$ estocástica, que varía en relación al tiempo t. Las funciones $h_i(t)$ $(i = 1..k)$ serán procesos estocásticos.

Otro cambio posible sería que las fertilidades λ_i dependan de la densidad de población. Similarmente tendríamos una matriz $\mathbf{A}_\mathsf{n}$ de la forma:

$$\mathbf{A}_\mathsf{n} = \begin{pmatrix} g_1(n_1,...,n_k) & \cdots & g_{k-1}(n_1,...,n_k) & g_k(n_1,...,n_k) \\ p_1 & 0 & 0 & 0 \\ 0 & \ddots & 0 & 0 \\ 0 & 0 & p_{k-1} & 0 \end{pmatrix}$$

Donde la matriz $\mathbf{A}_\mathsf{n}$ es **dependiente de la densidad.** Si queremos que las fertilidades λ_i declinen exponencialmente con la densidad, podemos definir:

$$g_i(n_1,...,n_k) = e^{-bN}, \text{ para } i = 1..k$$

donde $N = \sum_{i=1}^{k} n_i$, y b determina la fuerza de la dependencia de las fertilidades con N.

2.4 Modelos matriciales clasificados por *etapas*

En algunos casos, es más apropiado tener en cuenta una variable que no sea la *edad*. En algunas especies, a veces el *tamaño* o el *desarrollo* conviene mas que la *edad*. Un modelo ampliamente usado de clasificación por *etapas* es el llamado *standard size-classified model*. Este modelo ha sido usado para modelar poblaciones de árboles, corales, tortugas desérticas y peces[1]. La matriz $\mathbf{A}_s$ es:

$$\mathbf{A}_s = \begin{pmatrix} g_1 & \lambda_2 & \cdots & \lambda_k \\ p_1 & g_2 & 0 & 0 \\ 0 & \ddots & \ddots & 0 \\ 0 & 0 & p_{k-1} & g_k \end{pmatrix}$$

Los elementos de la diagonal g_i son las probabilidades de sobrevivir y quedarse en la etapa i, mientras que los elementos de la subdiagonal p_i representan las probabilidades de sobrevivir y *crecer* a la etapa $i+1$.

A continuación damos el grafo de *ciclo de vida* correspondiente a un modelo clasificado por *etapas*:

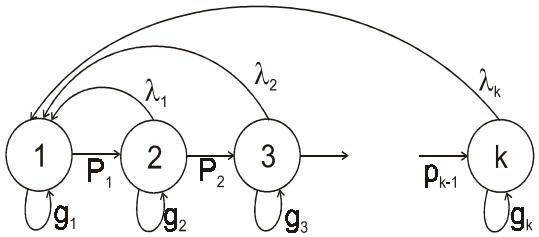

Figure 2.2: Grafo de Ciclo de Vida para modelos clasificados por etapas

2.5 Resultados Analíticos

En la sección anterior, se puede simular fácilmente el desarrollo de la población utilizando (1). En esta sección se darán métodos para analizar la matriz de proyección y obtener información directamente de ella.

2.5.1 Solución de la ecuación de proyección $\mathbf{n}(t+1) = \mathbf{A}\mathbf{n}(t)$

Sea $\mathbf{A}$ una matriz de proyección. Usaremos los autovalores λ_i, los autovectores derechos $\mathbf{w}_i$ y los izquierdos $\mathbf{v}_i$, que satisfacen:

$$\mathbf{A}\mathbf{w}_i = \lambda_i \mathbf{w}_i \tag{4}$$

$$\mathbf{v}_i^* \mathbf{A} = \lambda_i \mathbf{w}_i$$

donde $\mathbf{v}_i^*$ es la transpuesta conjugada *compleja* de $\mathbf{v}$ (en el caso de ser real, $\mathbf{v}^* = \mathbf{v}^T$). Supongamos $\mathbf{n}_0$ la población inicial. Gracias a propiedades de las matrices de proyección (que veremos más adelante), los autovalores de $\mathbf{A}$ son distintos entre sí, lo que nos dice que los autovectores $\mathbf{w}_i$ serán linealmente independientes. Luego, como los $\mathbf{w}_i$ son l.i., podemos escribir:

$$\mathbf{n}_0 = c_1\mathbf{w}_1 + c_2\mathbf{w}_2 + ... + c_k\mathbf{w}_k \tag{5}$$

para algunos coeficientes c_i.

Ahora premultiplicamos por $\mathbf{A}$ para obtener $\mathbf{n}(1)$, usando (2), (4) y (5):

$$\mathbf{n}(1) = \mathbf{A}\mathbf{n}_0 = \sum_i c_i \mathbf{A}\mathbf{w}_i = \sum_i c_i \lambda_i \mathbf{w}_i$$

Si multiplicamos nuevamente, obtenemos $\mathbf{n}(2)$:

$$\mathbf{n}(2) = \mathbf{A}\mathbf{n}(1) = \sum_i c_i \lambda_i \mathbf{A}\mathbf{w}_i = \sum_i c_i \lambda_i^2 \mathbf{w}_i$$

Continuando de esta manera, llegamos a la solución:

$$\mathbf{n}(t) = \sum_i c_i \lambda_i^t \mathbf{w}_i \tag{6}$$

De esta manera podemos ver que el comportamiento de $\mathbf{n}(t)$ depende de los autovalores λ_i, que a medida que t crece serán elevados a más altas potencias.

2.5.2 Propiedades de las matrices de proyección

Una matriz es *nonegativa* si todos sus elementos son mayor o igual que 0. Una matriz es *positiva* si todos sus elementos son mayor estricto que 0. Diremos que una matriz no negativa es *irreducible* sii su grafo de ciclo de vida contiene un camino de cada nodo a cada nodo (esto es, si es conexo). Diremos que una matriz no negativa irreducible es *primitiva* si se convierte *positiva* cuando es elevada a valores suficientemente altos.

El teorema perron-frobenius [2] asegura que toda matriz no negativa tiene un autovalor mayor o igual que el resto en magnitud. Este autovalor es llamado el *autovalor dominante*, λ_1. Además, si nuestra matriz es primitiva, λ_1 es real y positivo, y mayor estricto que el resto de los autovalores.

Dos propiedades importantes que usaremos son: (para desarrollos, ver [2]):

- La mayoría de la matrices de proyección son primitivas.
- Las matrices de proyección que consideramos en este trabajo (y, en general, en cualquier aplicación demográfica) tiene sus autovalores distintos entre sí.

2.5.3 El teorema Strong-Ergodic

Una población será llamada *ergodica* si su eventual comportamiento es independiente de su estado inicial (Cohen 1979c). Veremos en esta sección que nuestros modelos tienen esta propiedad.

Sea $\mathbf{A}$ matriz de proyección primitiva.
Reescribiendo (6) con los autovalores decreciendo en magnitud:

$$\mathbf{n}(t) = c_1 \lambda_1^t \mathbf{w}_1 + c_2 \lambda_2^t \mathbf{w}_2 + c_3 \lambda_3^t \mathbf{w}_3 + \cdots \tag{7}$$

Si λ_1 es estrictamente mayor que los otros autovalores, entonces a medida que t crece el término $c_1 \lambda_1^t \mathbf{w}_1$ terminará dominando, sin importar la población inicial. La población crecerá a una tasa dada por λ_1, con una estructura proporcional a $\mathbf{w}_1$. Para ver esto, dividiendo en (7) por λ_1^t (λ_1 es positivo) obtenemos:

$$\frac{\mathbf{n}(t)}{\lambda_1^t} = c_1 \mathbf{w}_1 + c_2 \left(\frac{\lambda_2}{\lambda_1}\right)^t \mathbf{w}_2 + c_3 \left(\frac{\lambda_3}{\lambda_1}\right)^t \mathbf{w}_3 + \cdots$$

si $\lambda_1 > |\lambda_i|$ para $i \geq 2$, entonces tomando límite en $t \to \infty$, tenemos:

$$\lim_{t \to \infty} \frac{\mathbf{n}(t)}{\lambda_1^t} = c_1 \mathbf{w}_1$$

Este importante resultado es conocido como el teorema **strong-ergodic**. Muestra que, si $\mathbf{A}$ es primitiva, la dinámica a largo plazo de una población es descripta por la tasa de crecimiento λ_1 y con una distribución estructural estable $\mathbf{w}_1$. También es utilizado un valor llamado tasa *intrínseca* de crecimiento, definido como $r = ln(\lambda_1)$.

2.5.4 Valor Reproductivo

Imaginamos dos poblaciones iguales, pero que comienzan con estados iniciales diferentes. Una comienza con un individuo en la etapa i, y la otra en la etapa j ($i \neq j$). Ambas se desarrollarán de la misma manera, pero tendrán diferentes tamaños de población. Estos diferentes tamaños estarán dados por los diferentes valores reproductivos de cada etapa. Viendo en (5), podemos obtener:

$$\mathbf{c} = \mathbf{W}^{-1}\mathbf{n}_0 = \overline{\mathbf{V}}\mathbf{n}_0$$

Eventualmente la población converge a $\mathbf{n}(t) = c_1\lambda_1^t\mathbf{w_1}$, donde $c_1 = \mathbf{v_1^*}\mathbf{n_0}$ por lo visto recién. Luego el tamaño de la población es proporcional a una suma de los estados iniciales, pero estos estan directamente influenciados por los valores de $\mathbf{v_1}$.

2.5.5 Damping Ratio

Uno puede preguntarse cuán rápido una población converge a su distribución estable. Viendo (5), uno puede convencerse de que mientras mas grande es λ_1 con respecto a los otros autovalores, más rápido se convergerá. De aquí se desprende el *damping ratio* ρ, definido como:

$$\rho = \lambda_1/|\lambda_2|$$

2.5.6 Sensibilidad del autovalor λ_1

La sensibilidad del autovalor dominante λ_1 a cambios en los elementos a_{ij} de la matriz de proyección puede obtenerse por medio de la siguiente fórmula (Caswell 1978) [2]:

$$\frac{\partial\lambda}{\partial a_{ij}} = \frac{\overline{v}_i w_j}{< \mathbf{w}, \mathbf{v} >}$$

2.5.7 Elasticidad del autovalor λ_1

Las entradas p_i y λ_i difieren en "escala". Una forma de análisis diferente que soluciona el problema es el de considerar en cuanta proporción cambia el autovalor cuando se realizan cambios proporcionales en los elementos a_{ij}. Estos valores se llaman *elasticidades* (Caswell 1984, Kroon 1986) [2]) y son definidos como sigue:

$$e_{ij} = \frac{a_{ij}\partial\lambda}{\lambda\partial a_{ij}}$$

Chapter 3

El Lenguaje PML

El lenguaje desarrollado es imperativo, y su sintaxis es parecida a la del lenguaje PASCAL. PML permite la definición de funciones, procedimientos, estructuras de control (bucles for y $while$, y condicionales if). El lenguaje también permite el llamado recursivo entre funciones y procedimientos, así también como el pasaje de variables por valor y por referencia (el tipo de llamado es *call by value*). Los tipos primitivos son entero, real, boolean y string. Se pueden crear arreglos multidimensionales de los tipos primitivos. Además se proveen tipos específicos para manejo de entrada y salida de texto y gráficos: *textwindow*, *linegraphwindow*, *bargraphwindow* y *sengraphwindow*.

Estos tipos permiten al usuario definir variables que representan, en tiempo de ejecución, ventanas donde se puede visualizar información:

- *textwindow*: Ventana de texto. Esta ventana permite el manejo de texto; el contenido de esta ventana puede ser cargado desde un archivo, o salvado hacia un archivo.
- *linegraphwindow*: Ventana de gráfico de líneas. Permite la visualización de una o más trayectorias; este tipo de ventanas será usado para simular dinámicas poblacionales a medida que avanza el tiempo.
- *bargraphwindow*: Ventana de gráfico de barras. Permite la creación de gráficos de barras. Este tipo de ventanas será usado para graficar distribuciones estables y valores reproductivos de las poblaciones.
- *sengraphwindow*: Ventana de gráfico de sensibilidad. Permite la visualización de imágenes que representan valores por medio de colores. Este tipo de ventanas será usado para visualizar las sensibilidades del autovalor dominante.

3.1 Definición EBNF de PML

A continuación se lista la definición del lenguaje, en forma BNF extendida; esta definición se mantiene en estrecha relación con la definición estándar de PASCAL ([14]).

3.1.1 Definiciones básicas

⟨letra⟩ = "A"|"B"|"C"|"B"|"D"|"E"|"F"|"G"| "H" | "I" | "J" | "K" | "L" | "M" | "N" | "O" | "P" | "Q"| "R" | "S" | "T" | "U" | "V" | "W" | "X" | "Y" | "Z"| "a" | "b" | "c" | "d" | "e" | "f" | "g" | "h" | "i"| "j" | "k" | "l" | "m" | "n" | "o" | "p" | "q" | "r"| "s" | "t" | "u" | "v" | "w" | "x" | "y" | "z"

⟨digito⟩ = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"

⟨string⟩ = """" string-char { string-char } """"

⟨string-char⟩ = cualquier_char_excepto"""" | """""

⟨digito-sec⟩ = [signo] digito { digito }

⟨signo⟩ = "+"| "-"

⟨numero-entero⟩ = digito-sec

⟨numero-real⟩ = digito-sec"." [digito-sec][exp] | [digito-sec][exp]

⟨exp⟩ = ("e" | "E")[signo] digito-sec

⟨numero⟩ = numero-entero | numero-real

⟨constante⟩ = [signo] (identificador | numero)| string

3.1.2 Identificadores

⟨identificador⟩ = letra { letra | digito }

⟨lista-ident⟩ = ident { "," ident }

Expresiones

⟨expresion⟩ = expresion-simple [relacion expresion-simple]

⟨expresion-simple⟩ = [signo] termino { operador-suma termino }

⟨termino⟩ = factor { factor-mult factor }

⟨factor⟩ = identificador | numero | string | llamada-funcion | **not** factor | "(" expresion ")"

⟨relacion⟩ = "=" | "<>" | "<" | "<=" | ">" | ">="

⟨operador-suma⟩ = "+" | "-" | **or**

⟨factor-mult⟩ = "*" | "/" | **and**

⟨llamada-funcion⟩ = identificador [parlist]

⟨parlist⟩ = "(" expresion { "," expresion } ")"

Comandos

⟨seccom⟩ = [comm { ";" comm }]

⟨comm⟩ = comm-simple | comm-estr

⟨comm-simple⟩ = asig | llamada-proc

⟨asig⟩ = identificador ":=" expresion

⟨llamada-proc⟩ = identificador [parlist]

⟨comm-estr⟩ = comm-comp | comm-bucle | comm-cond

⟨comm-comp⟩ = **begin** seccomm **end**

⟨comm-bucle⟩ = comm-while | comm-for

⟨comm-while⟩ = **while** expresion **do** comm

⟨comm-for⟩ = **for** identificador ":=" expresion **(to**| **dto)** expresion **do** comm

⟨comm-cond⟩ = comm-if
⟨comm-if⟩ = **if** expresion **then** comm [**else** comm]

3.1.3 Funciones y Procedimientos

⟨proc-enc⟩ = **procedure** identificador [declist]
⟨func-enc⟩ = **function** identificador [declist]":" resultado
⟨resultado⟩ = tipo
⟨declist⟩ = "(" decl { ";" decl } ")"
⟨decl⟩ = vardecl | valdecl
⟨vardecl⟩ = **var** identificador ":" tipo
⟨valdecl⟩ = identificador ":" tipo
⟨tipo⟩ = integer | bool | real | string | textwindow | graphwindow

Programas y Bloques

⟨programa⟩ = encabezado bloque"."
⟨encabezado⟩ = **program** identificador ";"
⟨bloque⟩ = declaraciones prog
⟨declaraciones⟩ = [variables] [proc_y_func]
⟨variables⟩ = **global** valdecl ";" {valdecl}
⟨variables_s⟩ = **var** valdecl ";" {valdecl}
⟨proc_y_func⟩ = { (proc-decl | fun-decl) }";"
⟨proc-decl⟩ = proc-enc ";" bloque_simple
⟨fun-decl⟩ = fun-enc ";" bloque_simple
⟨bloque_simple⟩ = declaraciones_s prog
⟨declaraciones_s⟩ = [variables_s]
⟨prog⟩ = **begin** seccom **end**

3.2 Semántica de PML

En esta sección definiremos la semántica de PML. Hacemos un desarrollo análogo al realizado en [13] para el caso del lenguaje Algol; una diferencia es la existencia de *declaraciones* en nuestro lenguaje. Además se usarán *continuaciones* pues el lenguaje permite "saltos" utilizando los comandos **exit** y **break**. Se dará solamente la semántica *denotacional* de PML, dejando de lado la *operacional*.

A continuación listamos los tipos del lenguaje:

$$\tau ::= \textbf{bool} \mid \textbf{int} \mid \textbf{string} \mid \textbf{textwindow} \mid \textbf{linegraphwindow} \mid \textbf{bargraphwindow} \mid \textbf{sengraphwindow} \text{ (tipos de datos)}$$

$$\theta ::= \textbf{exp}[\tau] \mid \textbf{var}[\tau] \mid \textbf{comm} \mid \textbf{exp}[\tau] \to \theta \mid \textbf{var}[\tau] \to \theta$$

$$\kappa ::= \theta \mid \textbf{prog} \mid \textbf{decl} \text{ (frases tipo)}$$

Las últimas dos producciones de θ nos permitirán armar las funciones y procedimientos; la distinción entre **var** y **exp** se utilizará para el pasaje por valor y por referencia. Hay que tener en cuenta el hecho

de que en el cuerpo de una función en PASCAL (y, por lo tanto en PML) el identificador correspondiente a esa función puede ser visto tanto como una variable (que será el resultado de la función) o como una llamada recursiva. Esto introduce algunas complicaciones técnicas que preferimos evitar considerando sólo procedimientos, que permiten "emular" funciones. Por simplicidad y sin pérdida de generalidad pensaremos que los procedimientos toman sólo dos parámetros, el primero por valor y el segundo por referencia.

3.2.1 Sintaxis Abstracta

Daremos aquí una descripción abstracta de la sintaxis (utilizando el estilo de [13]), como paso previo a la definición de la semántica. En esta descripción, un contexto Γ es una función que mapea identificadores en frases tipo θ.

Declaraciones

$$\frac{}{\Gamma \vdash \iota : \tau \ \mathbf{:}\ \mathbf{decl}}$$

$$\frac{(\Gamma \mid p \mapsto \mathbf{exp}[\tau_1] \to \mathbf{var}[\tau_2] \to \mathbf{comm} \mid \iota_1 \mapsto \mathbf{var}[\tau_1] \mid \iota_2 \mapsto \mathbf{var}[\tau_2]) \vdash q : \mathbf{prog}}{\Gamma \vdash \ \mathbf{proc}\ p\ (\iota_1 : \tau_1, \mathbf{var}\ \iota_2 : \tau_2); q\ \mathbf{:}\ \mathbf{decl}}$$

Sea $\Gamma \vdash d : \mathbf{decl}$, definimos Γ, d contexto de la siguiente manera:

$$\Gamma,\ \iota : \tau = (\Gamma \mid \iota \mapsto \mathbf{var}[\tau])$$

$$\Gamma,\ \mathbf{proc}\ p\ (\iota_1 : \tau_1, \mathbf{var}\ \iota_2 : \tau_2); q = (\Gamma\ \mid p \mapsto \mathbf{exp}[\tau_1] \to \mathbf{var}[\tau_2] \to \mathbf{comm})$$

Programas

$$\frac{\Gamma \vdash d_1 : \mathbf{decl}\ \Gamma, d_1 \vdash d_2 : \mathbf{decl}\ ...\ \Gamma, d_1, ..., d_{m-1} \vdash d_m : \mathbf{decl}\ \ \Gamma, d_1, ..., d_m \vdash c_1, ..., c_n : \mathbf{comm}}{\Gamma \vdash d_1; ...; d_m;\ \mathbf{begin}\ c_1; ...; c_n\ \mathbf{end}\ \mathbf{:}\ \mathbf{prog}}$$

Comandos

$$\frac{}{\Gamma \vdash \textbf{skip, exit, break : comm}}$$

$$\frac{\Gamma \vdash c_1, ..., c_n : \textbf{ comm}}{\Gamma \vdash \textbf{begin } c_1; ...; c_n \textbf{ end : comm}}$$

$$\frac{\Gamma \vdash b : \textbf{exp}[bool]\ \Gamma \vdash c_1, c_2\textbf{: comm}}{\Gamma \vdash \textbf{if } b \textbf{ then } c_1 \textbf{ else } c_2 \textbf{ : comm}}$$

$$\frac{\Gamma \vdash b : \textbf{exp}[bool]\ \Gamma \vdash c\textbf{: comm}}{\Gamma \vdash \textbf{while } b \textbf{ do } c \textbf{ : comm}}$$

$$\frac{\Gamma \vdash e : \textbf{exp}[\tau]}{\Gamma \vdash \iota := e : \textbf{ comm}}\ \Gamma(\iota) = \textbf{var}[\tau]$$

$$\frac{\Gamma \vdash e_1, e_2 : \textbf{exp}[int]\ (\Gamma \mid \iota \mapsto \textbf{exp}[int]) \vdash c : \textbf{ comm}}{\Gamma \vdash \textbf{for } \iota := e_1 \textbf{ to } e_2 \textbf{ do } c : \textbf{ comm}}\ \Gamma(\iota) = \textbf{var}[int]$$

$$\frac{\Gamma \vdash e_1 : \textbf{exp}[\tau_1]}{\Gamma \vdash p(e_1, \iota_2) : \textbf{ comm}}\ \Gamma(\iota_2) = \textbf{var}[\tau_2]\ ,\ \Gamma(p) = \textbf{exp}[\tau_1] \to \textbf{var}[\tau_2] \to \textbf{comm}$$

3.2.2 Semántica

La semántica de los tipos de datos son los esperados, es decir:

$[\![bool]\!] = \{true, false\}$ (valores de verdad)

$[\![int]\!] = Z$ (números enteros)

$[\![real]\!] = R$ números reales

$[\![string]\!] =$ secuencias de caracteres

$[\![textwindow]\!] =$ ventanas de texto

$[\![linegraphwindow]\!] =$ ventanas gráficas

$[\![bargraphwindow]\!] =$ ventanas gráficas

$[\![sengraphwindow]\!] =$ ventanas gráficas

No describiremos aquí las "ventanas". Asumiremos que, al igual que los números enteros, los booleanos y las secuencias de caracteres, las ventanas pueden definirse matemáticamente. Dicha definición no tiene relevancia aquí.

Vemos ahora la semántica de las frases tipo. Primero definimos Loc como el conjunto de "lugares" de memoria, que utilizaremos junto con la función $new : Env \to S \to Loc$, que dado un ambiente y un estado nos devuelve un nuevo lugar de memoria. Para esto, definiremos S: $S = Loc \to (V + \{used, unused\})$. Los valores $used$ y $unused$ nos servirán para "marcar" los lugares de memoria como reservados o libres en el estado (y de esta manera evitar conflictos en la asignación de lugares de memoria a identificadores).

$[\![\mathbf{exp}[\tau] \to \theta]\!] = [\![\tau]\!] \to [\![\theta]\!]$

$[\![\mathbf{var}[\tau] \to \theta]\!] = Loc \to [\![\theta]\!]$

$[\![\mathbf{var}[\tau]]\!] = Loc$

$[\![\mathbf{exp}[\tau]]\!] = S \to [\![\tau]\!]$

$[\![\mathbf{decl}]\!] = S \to (Env, S)$

$[\![\mathbf{prog}]\!] = (S \rightharpoonup O) \to (S \rightharpoonup O)$

Pensaremos en O como el conjunto de posibles "outputs".

$[\![\mathbf{comm}]\!] = (S \rightharpoonup O) \to (S \rightharpoonup O)$

$[\![\Gamma]\!] = \{\ \rho \mid \rho(exit)\ \epsilon\ S \rightharpoonup O,\ \rho(break)\ \epsilon\ (S \rightharpoonup O) + \{nobreak\},\ \forall \iota, \theta \text{ tq } \Gamma(\iota) = \theta \text{ y } \rho(\iota) \epsilon [\![\theta]\!]\ \}$

Estas definiciones nos permiten dar el tipo de la función semántica:

$$\text{si } \Gamma \vdash c : \kappa,\ [\![c]\!]\ :\ [\![\Gamma]\!] \to [\![\kappa]\!]$$

Daremos las definiciones de esta función a continuación para cada caso de κ y de la sintaxis abstracta.

Semántica de las declaraciones

$[\![\iota : \tau]\!]_{\mathsf{decl}}\ \rho\ s = (\ (\rho \mid \iota \mapsto new.\rho.s), (s \mid new.\rho.s \mapsto\ used))$

$[\![\textbf{proc } p(\iota_1 : \tau_1;\ \textbf{var } \iota_2 : \tau_2); q]\!]_{\mathsf{decl}}\ \rho\ s = (\rho', s)$ donde

$$\begin{aligned}
\rho' =& (\rho \mid p \mapsto \lambda v.x.k.s'.\ [\![q]\!]_{\mathsf{prog}}\rho''\ (release.l.k)\ s'') \\
\rho'' =& (\rho' \mid \iota_1 \mapsto l \mid \iota_2 \mapsto x) \\
s'' =& (s' \mid l \mapsto v) \\
l =& new.\rho'.s' \\
release\ l\ k =& \lambda s.\ k(s \mid l \mapsto unused)
\end{aligned}$$

El caso de las expresiones es similar al dado por Tennent [13], por lo tanto es dejado. Notamos que las semánticas de expresiones no toman continuaciones y siempre terminan, ya que en este desarrollo no son tenidas en cuenta las funciones.

Semántica de los comandos

$[\![skip]\!]_{\mathsf{comm}}\ \rho\ k = k$

$[\![\iota := e]\!]_{\mathsf{comm}}\ \rho\ k\ s = k(s \mid \rho(\iota) \mapsto [\![e]\!]\rho s)$

$[\![p(e_1, \iota_2)]\!]_{\mathsf{comm}}\ \rho\ k\ s = \rho(p)\ ([\![e_1]\!]\rho s)\ \rho(\iota_2)\ k\ s$

$$\begin{aligned}
[\![\textbf{for } \iota := e_1 \textbf{ to } e_2 \textbf{ do } c]\!]_{\mathsf{comm}}\ \rho\ k\ s = {} & let\ v_1 = [\![e_1]\!]\rho s\ v_2 = [\![e_2]\!]\rho s \\
& metafor\ n\ m\ =\ if\ n > m\ then\ k \\
& else\ [\![c]\!]_{\mathsf{comm}}(\rho \mid \iota \mapsto n \mid break \mapsto k)(metafor\ (n+1)\ m) \\
& \text{in} \\
& metafor\ v_1\ v_2\ s
\end{aligned}$$

$[\![\textbf{while } b \textbf{ do } c]\!]_{\mathsf{comm}}\ \rho\ k = k'$

donde $\quad k'(s) = \begin{cases} [\![c]\!]_{\mathsf{comm}}(\rho | break \mapsto k)k'\ s & \text{if } [\![b]\!]\rho\ s = true \\ k(s) & \text{if } [\![b]\!]\rho\ s = false \end{cases}$

$[\![\textbf{exit }]\!]_{\mathsf{comm}}\ \rho\ k = \rho(exit)$

$[\![\textbf{break }]\!]_{\mathsf{comm}}\ \rho\ k = \text{if } \rho(break) \neq nobreak \text{ then } \rho(break) \text{ else } k$

$[\![\textbf{if } b \textbf{ then } c_1 \textbf{ else } c_2]\!]_{\mathsf{comm}}\ \rho\ k\ s = \begin{cases} [\![c_1]\!]_{\mathsf{comm}}\rho\ k\ s & \textbf{if } [\![b]\!]\rho s = true \\ [\![c_2]\!]_{\mathsf{comm}}\rho\ k\ s & \textbf{if } [\![b]\!]\rho s = false \end{cases}$

Semántica de los programas

La siguiente definición sirve para programas que pueden ser tanto el programa principal como algún procedimiento. Para el caso del programa principal, asumiremos la existencia de ρ_0 y k_0, el ambiente

inicial y la continuación inicial. La continuación inicial k_0 genera output trivial y el ambiente inicial consta de un solo par, $\rho_0 = \{(break, nobreak)\}$. El programa principal será evaluado inicialmente en este ambiente y en esta continuación.

$$\begin{aligned}[\![d_1; ...; d_m;\ \textbf{begin}\ c_1; ...; c_n \textbf{end}]\!]_{\mathsf{prog}}\ \rho\ k\ s = &\text{let } (\rho_1, s_1) = [\![d_1]\!]_{\mathsf{decl}}\rho\ s,\ ...\ ,(\rho_m, s') = [\![d_m]\!]_{\mathsf{decl}}\rho_{m-1}\ s_{m-1} \\ &\rho' = (\rho_m \mid exit \mapsto k) \text{ in} \\ &[\![c_1]\!]_{\mathsf{comm}}\rho'([\![c_2]\!]_{\mathsf{comm}}\rho'(...([\![c_n]\!]_{\mathsf{comm}}\rho'\ k)...))\ s'\end{aligned}$$

3.3 Implementación de PML

3.3.1 Rol de la semántica

Si bien la semántica ha sido descripta usando herramientas propias de semántica denotacional, la implementación es más directa ya que se utiliza el lenguaje PASCAL que ofrece estructuras de control similares a las que se quieren implementar en PML. El rol de la semántica definida cumple entonces dos objetivos:

- Proveer una descripción matemática del lenguaje PML
- Establecer una base formal que permitiría demostrar la correctitud de la implementación.

3.3.2 Parser y Gramática

Teniendo la definición EBNF, se puede desarrollar un parser para PML, utilizando las herramientas *lex* & *yacc* ([9]). Primero definimos el analizador lexicográfico de PML. Las palabras reservadas de PML son (estas se pueden deducir de la definición EBNF):

$$\begin{aligned}\langle \textit{palabras-reservadas} \rangle = &\text{ "}+\text{" | "}-\text{" | "}*\text{" | "}/\text{" | "}=\text{" | "}<>\text{" | "}<\text{" | "}<=\text{" | "}>\text{" |} \\ &\text{ "}>=\text{" | "(" | ")" | "[" | "]" | "}:=\text{" | "}.\text{" | "," | "}:\text{" |} \\ &\text{";" | ".." | "}and\text{" | "}array\text{" | "}begin\text{" | "}div\text{" | "}do\text{" |} \\ &\text{"}dto\text{" | "}else\text{" | "}end\text{" | "}for\text{" | "}function\text{" |} \\ &\text{"}global\text{" | "}if\text{" | "}mod\text{" | "}not\text{" | "}of\text{" | "}or\text{" |} \\ &\text{ | "}procedure\text{" | "}program\text{" | "}then\text{" | "}to\text{" |} \\ &\text{ | "}var\text{" | "}while\text{"}\end{aligned}$$

Estas palabras serán reconocidas por el analizador como tokens. Además de estas, debemos agregar expresiones regulares para que el analizador reconozca los restantes tokens:

Definimos previamente: $L = [A - Za - z]$, $D = [0 - 9]$.

- Expresión regular para *identificadores*:

$$\{L\}(\{L\} \mid \{D\})^*$$

- Expresión regular para *números enteros*:

$$\{D\}^+$$

- Expresión regular para *números reales*:

$$\{D\}^+(.\{D\}^+)?([Ee][+-]?\{D\}^+)?$$

- Expresión regular para *strings*:

$$\backslash'[\hat{}\backslash'\backslash n^*\backslash'$$

- Expresión regular para *comentarios*:

$$\{[\hat{}\}\backslash n]^*\}$$

- Expresión regular para *secuencias de tabs y new-lines*:

$$[\ \backslash t\backslash n]^*$$

Una vez que tenemos definido el analizador lexicográfico, definimos la gramática. Esta gramática está definida directamente por las producciones (reglas) dadas en la definición EBNF del lenguaje.

3.3.3 Portabilidad y extensibilidad

Al proveer la directiva **extern** en la definición de funciones y procedimientos, PML soporta la llamada de funciones y procedimientos externos al lenguaje. Esto permite proveer librerías precompiladas con funciones y procedimientos, que pueden estar compiladas en cualquier otro lenguaje. En la actual implementación para plataforma Windows, se implementó las llamadas a librerías de formato DLL. Esta característica tiene las siguientes ventajas:

- Abstracción. Mantener librerías externas permite a PML abstraer las implementaciones de entrada/salida, gráficos, funciones de conversión, numéricos, que pueden ser mas tarde cambiadas.
- Portabilidad. El código del lenguaje puede ser mantenido libre de llamadas dependientes de plataforma, guardando todas las llamadas al sistema en las librerías.
- Performance. Se pueden proveer librerías precompiladas que favorecen en velocidad de ejecución (en la implementación actual, PML es un intérprete solamente, por lo cual es importante poder llamar a funciones y procedimientos precompilados).

Chapter 4

Implementaciones de los modelos en PML

4.1 Simulaciones

4.1.1 Modelo Constante

A continuación vemos la implementación de un modelo de leslie de matriz constante, para el caso de una matriz **A**:

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 5 \\ .3 & 0 & 0 \\ 0 & .7 & 0 \end{pmatrix} \tag{3}$$

y estado inicial **n**:

$$\mathbf{n} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$

Listado en PML

```
uses dialogs, sysutils, windows, arithmetic, matrixext;

global g:linegraphwindow;

var i,j,t:integer;
  a:matrix [3,3] of real;
n:vector [3] of real;
r:real;

begin
```

```
 scalelog(g, true);

 a^0 := [0.0, 1.0, 5.0];
 a^1 := [0.3, 0.0, 0.0];
 a^2 := [0.0, 0.5, 0.0];

 n[0]:=1.0;
 n[1]:=0.0;
 n[2]:=0.0;

 t:=strtoint(inputbox('Tiempo t','calcular hasta t=','10'));

 for i:=0 to t-1 do
  begin
 multiplicar(n, a);
 addrpoints(g, n[0], '', 10,0);
 addrpoints(g, n[1], '', 10,1);
 addrpoints(g, n[2], '', 10,2);
 repaint(g);
end;

end.
```

Resultado de la simulación

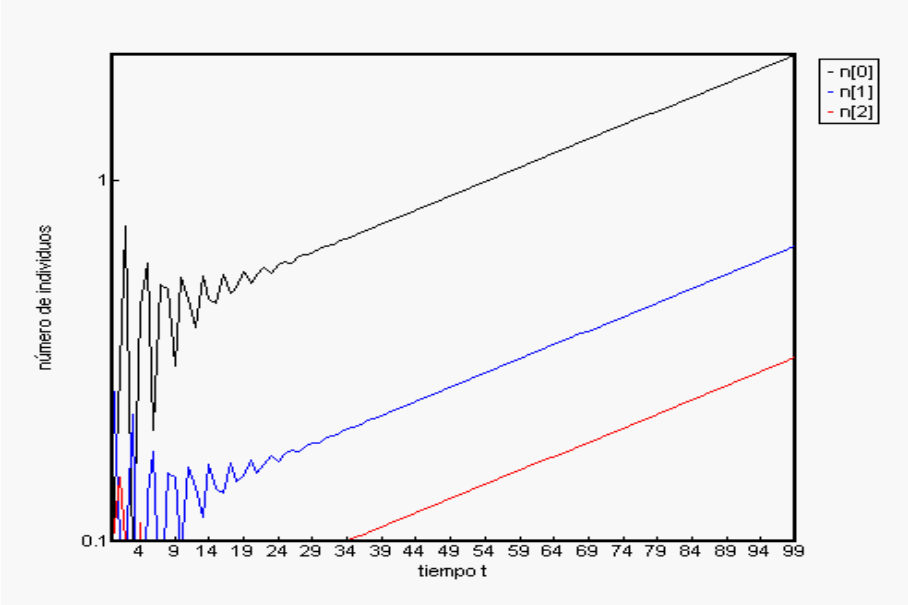

Figure 4.1: Leslie Constante

Podemos apreciar en el gráfico las evoluciones de cada clase de edad. Teniendo en cuenta la escala logarítmica, se puede ver que cada clase de edad luego de fluctuar irregularmente al principio, comienza a crecer exponencialmente con un crecimiento similar a medida que avanza el tiempo.

4.1.2 Estados Iniciales

El caso anterior era específico a la distribución de edades $\mathbf{n}(0)$. Consideramos en este caso varias poblaciones diferentes, cada una con un estado inicial aleatorio. La matriz es la misma que en el caso anterior (3). Los estados iniciales para este caso son:

```
n1 = [ 2.071531 0.8968879 2.17962  ]
n2 = [ 2.620035 1.033543 2.158412  ]
n3 = [ 2.274696 3.090863 0.9672987  ]
n4 = [ 1.322421 2.160626 3.134101  ]
n5 = [ 0.3748354 0.2227502 2.388807  ]
n6 = [ 3.029126 0.4063146 0.07311776  ]
n7 = [ 1.242075 1.898932 2.589447  ]
n8 = [ 0.5306102 1.123725 0.4605948  ]
n9 = [ 2.073476 0.4945276 0.07567254  ]
```

Listado en PML

```
uses dialogs, sysutils, windows, arithmetic, matrixext;

global w:textwindow;
g:linegraphwindow;

var i,j,t:integer;
  a:matrix [3,3] of real;
n1:vector [3] of real;
n2:vector [3] of real;
n3:vector [3] of real;
n4:vector [3] of real;
n5:vector [3] of real;
n6:vector [3] of real;
n7:vector [3] of real;
n8:vector [3] of real;
n9:vector [3] of real;

r:real;

begin

 scalelog(g, true);

 a^0 := [0.0, 1.0, 5.0];
```

```
 a^1 := [0.3, 0.0, 0.0];
 a^2 := [0.0, 0.5, 0.0];

 r:=3.3;

for i:=0 to 2 do n1[i]:=random*r;
for i:=0 to 2 do n2[i]:=random*r;
for i:=0 to 2 do n3[i]:=random*r;
for i:=0 to 2 do n4[i]:=random*r;
for i:=0 to 2 do n5[i]:=random*r;
for i:=0 to 2 do n6[i]:=random*r;
for i:=0 to 2 do n7[i]:=random*r;
for i:=0 to 2 do n8[i]:=random*r;
for i:=0 to 2 do n9[i]:=random*r;


 t:=strtoint(inputbox('Tiempo t','calcular hasta t=','10'));

 addstr(w, '');
 addstr(w, 'calculando hasta tiempo t = '+inttostr(t));

 for i:=0 to t-1 do
  begin
 multiplicar(n1, a);
 multiplicar(n2, a);
 multiplicar(n3, a);
 multiplicar(n4, a);
 multiplicar(n5, a);
 multiplicar(n6, a);
 multiplicar(n7, a);
 multiplicar(n8, a);
 multiplicar(n9, a);
 addrpoints(g, n1[0]+n1[1]+n1[2], '', 10,0);
 addrpoints(g, n2[0]+n2[1]+n2[2], '', 10,1);
 addrpoints(g, n3[0]+n3[1]+n3[2], '', 10,2);
 addrpoints(g, n4[0]+n4[1]+n4[2], '', 10,3);
 addrpoints(g, n5[0]+n5[1]+n5[2], '', 10,4);
 addrpoints(g, n6[0]+n6[1]+n6[2], '', 10,5);
 addrpoints(g, n7[0]+n7[1]+n7[2], '', 10,6);
 addrpoints(g, n8[0]+n8[1]+n8[2], '', 10,7);
 addrpoints(g, n9[0]+n9[1]+n9[2], '', 10,8);
 repaint(g);
end;
pause ('Programa Finalizado. Pulse una tecla para continuar.');
end.
```

Resultados

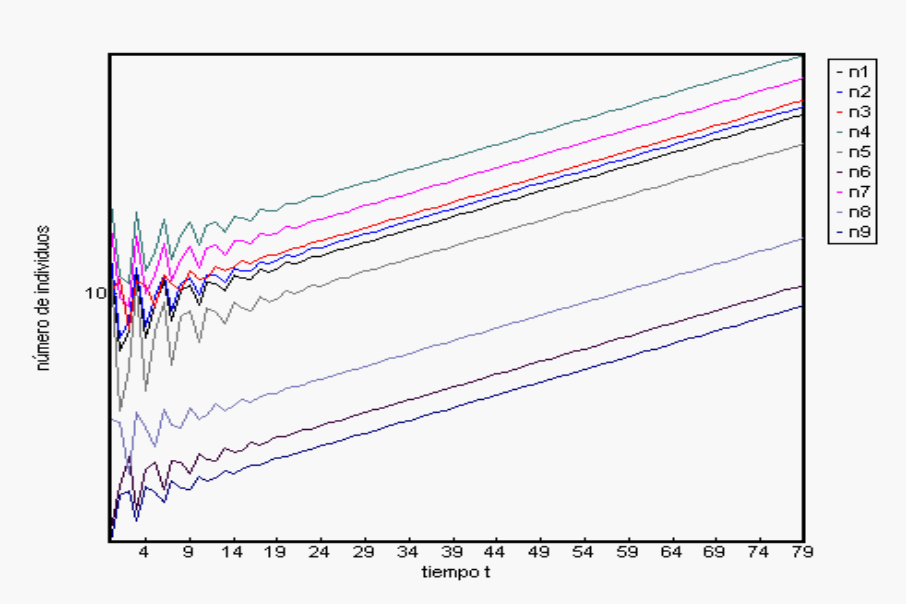

Figure 4.2: Estados Iniciales 1: mismo crecimiento, diferente tamaño

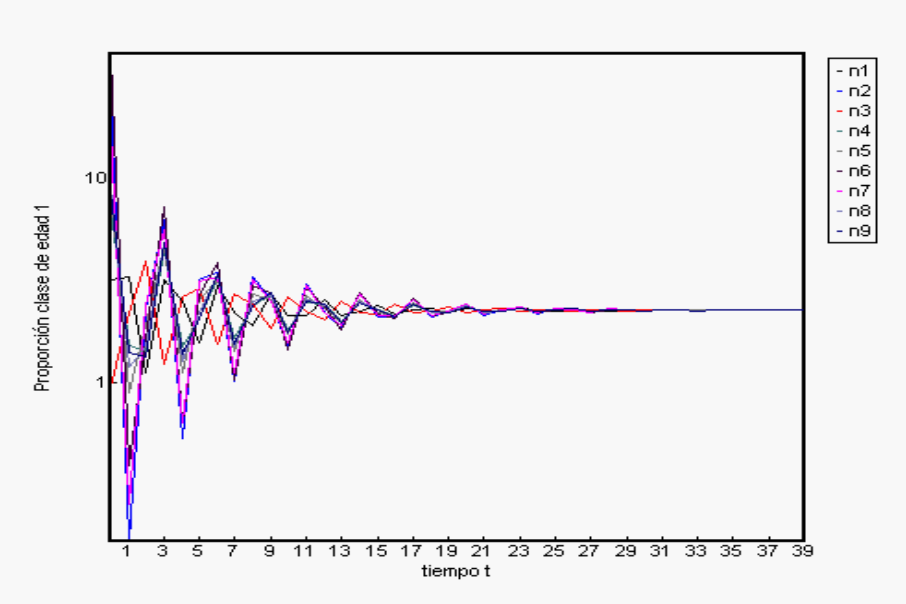

Figure 4.3: Estados Iniciales 2: misma distribución de clases de edad

Aquí podemos ver como cada una crece con la misma tasa (figura 4.2), y converge a una misma distribución de edades (figura 4.3). Se puede ver que en lo que difieren es en las cantidades de individuos (figura 4.2). Es claro que un estado inicial que favorezca a la clase de edad 3 será superior a uno que se torne mayor en la 1, ya que los individuos de la clase 3 comienzan a reproducirse inmediatamente, mientras que los de la clase 1 recién lo harán luego de 2 unidades de tiempo, y sujetos a las tasas de mortalidad.

4.1.3 Sensibilidad mediante proyección

Listado

```
uses dialogs, sysutils, windows, arithmetic, matrixext;

global g:linegraphwindow;

var i,j,t:integer;
  a1:matrix [3,3] of real;
  a2:matrix [3,3] of real;
  a3:matrix [3,3] of real;
  a4:matrix [3,3] of real;
  a5:matrix [3,3] of real;
n1:matrix [3] of real;
n2:matrix [3] of real;
n3:matrix [3] of real;
n4:matrix [3] of real;
n5:matrix [3] of real;

begin

 scalelog(g, true);

 titulo(g, 'Gráfico');

 a1^0 := [0.0, 1.0, 5.0];
 a1^1 := [0.3, 0.0, 0.0];
 a1^2 := [0.0, 0.5, 0.0];

 a2^0 := [0.0, 0.9, 5.0];
 a2^1 := [0.3, 0.0, 0.0];
 a2^2 := [0.0, 0.5, 0.0];

 a3^0 := [0.0, 1.0, 4.5];
 a3^1 := [0.3, 0.0, 0.0];
 a3^2 := [0.0, 0.5, 0.0];

 a4^0 := [0.0, 1.0, 5.0];
 a4^1 := [0.3, 0.0, 0.0];
 a4^2 := [0.0, 0.45, 0.0];
```

```
a5^0 := [0.0, 1.0, 5.0];
a5^1 := [0.27, 0.0, 0.0];
a5^2 := [0.0, 0.45, 0.0];

n1[0]:=1.0;
n1[1]:=0.0;
n1[2]:=0.0;

n2[0]:=1.0;
n2[1]:=0.0;
n2[2]:=0.0;

n3[0]:=1.0;
n3[1]:=0.0;
n3[2]:=0.0;

n4[0]:=1.0;
n4[1]:=0.0;
n4[2]:=0.0;

n5[0]:=1.0;
n5[1]:=0.0;
n5[2]:=0.0;

legend(g, true);
legends(g, 'original', 0);
legends(g, '10% menos l2', 1);
legends(g, '10% menos l3', 2);
legends(g, '10% menos p2', 3);
legends(g, '10% menos p1', 4);

xlabel(g, 'tiempo t');
ylabel(g, 'número de individuos');
label(g, 'Gráfico de Leslie');

t:=strtoint(inputbox('Tiempo t','calcular hasta t=','10'));


for i:=0 to t-1 do
 begin
multiplicar(n1, a1);
multiplicar(n2, a2);
multiplicar(n3, a3);
multiplicar(n4, a4);
multiplicar(n5, a5);
addrpoints(g, n1[0]+n1[1]+n1[2], '', 10,0);
```

```
 addrpoints(g, n2[0]+n2[1]+n2[2], '', 10,1);
 addrpoints(g, n3[0]+n3[1]+n3[2], '', 10,2);
 addrpoints(g, n4[0]+n4[1]+n4[2], '', 10,3);
 addrpoints(g, n5[0]+n5[1]+n5[2], '', 10,4);
 repaint(g);
end;

pause ('Simulación Finalizada. Pulse una tecla para continuar.');
end.
```

Resultados

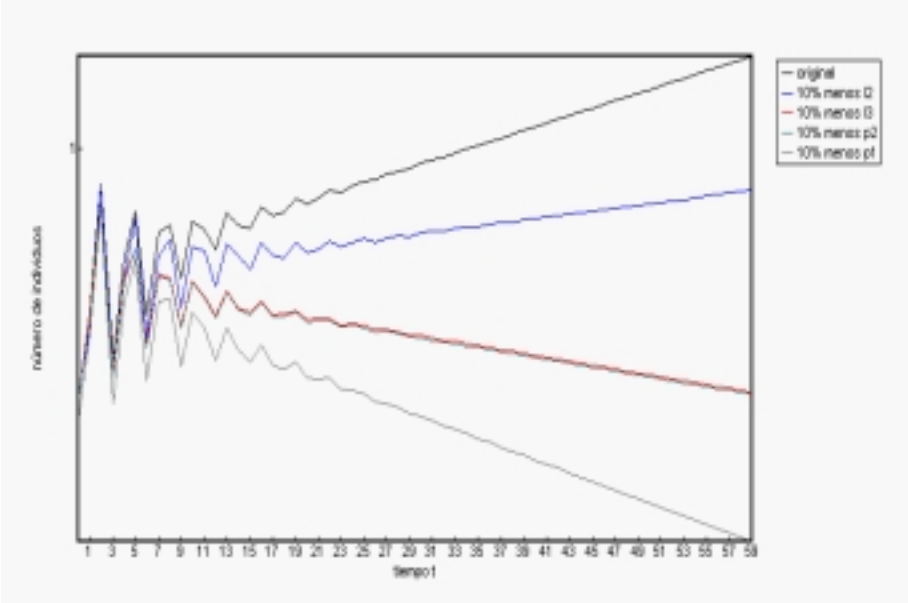

Figure 4.4: Sensibilidad

Podemos ver aquí cuán sensible es cada entrada de la matriz de proyección. Recordamos la matriz $\mathbf{A}$:

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 5 \\ .3 & 0 & 0 \\ 0 & .7 & 0 \end{pmatrix}$$

Si reducimos la entrada a_{01}, de 1 a 0.9, la población no se ve tan reducida en su desarrollo; incluso continúa creciendo. Pero si reducimos a_{10}, de .3 a .27, reducimos drásticamente la posibilidad de crecimiento de la población, quien se extingue rápidamente. Las otras reducciones tienden a extinguir la población, aunque más lentamente.

4.1.4 Modelo Estocástico

Listado

```
uses dialogs, sysutils, windows, arithmetic, matrixext;

global w:textwindow;
g:linegraphwindow;

function h(t,i:integer):real;
var result:real;
begin

if random<0.5 then result:=0.5 else result:=2.0;

if i=2 then h:=5.0*result else h:=result;

end;

var i,j,t:integer;
  a1:matrix [3,3] of real;
  a2:matrix [3,3] of real;
  a3:matrix [3,3] of real;
  a4:matrix [3,3] of real;
  a5:matrix [3,3] of real;
  a6:matrix [3,3] of real;
  a7:matrix [3,3] of real;
  a8:matrix [3,3] of real;
  a9:matrix [3,3] of real;
  a10:matrix [3,3] of real;
  a11:matrix [3,3] of real;
  a12:matrix [3,3] of real;
  a13:matrix [3,3] of real;
  a14:matrix [3,3] of real;
  a15:matrix [3,3] of real;
n1:vector [3] of real;
n2:vector [3] of real;
n3:vector [3] of real;
n4:vector [3] of real;
n5:vector [3] of real;
n6:vector [3] of real;
n7:vector [3] of real;
n8:vector [3] of real;
n9:vector [3] of real;
n10:vector [3] of real;
n11:vector [3] of real;
n12:vector [3] of real;
n13:vector [3] of real;
```

```
n14:vector [3] of real;
n15:vector [3] of real;

begin

 scalelog(g, true);

 titulo(w, 'Información');
 titulo(g, 'Gráfico');

 width(w, 300);
 width(g, 500);
 left(g, 305);

 addstr(w, '');

 a1^0 := [0.0, 1.0, 5.0];
 a1^1 := [0.3, 0.0, 0.0];
 a1^2 := [0.0, 0.5, 0.0];

 a2^0 := [0.0, 1.0, 5.0];
 a2^1 := [0.3, 0.0, 0.0];
 a2^2 := [0.0, 0.5, 0.0];

 a3^0 := [0.0, 1.0, 5.0];
 a3^1 := [0.3, 0.0, 0.0];
 a3^2 := [0.0, 0.5, 0.0];

 a4^0 := [0.0, 1.0, 5.0];
 a4^1 := [0.3, 0.0, 0.0];
 a4^2 := [0.0, 0.5, 0.0];

 a5^0 := [0.0, 1.0, 5.0];
 a5^1 := [0.3, 0.0, 0.0];
 a5^2 := [0.0, 0.5, 0.0];

 a6^0 := [0.0, 1.0, 5.0];
 a6^1 := [0.3, 0.0, 0.0];
 a6^2 := [0.0, 0.5, 0.0];

 a7^0 := [0.0, 1.0, 5.0];
 a7^1 := [0.3, 0.0, 0.0];
 a7^2 := [0.0, 0.5, 0.0];

 a8^0 := [0.0, 1.0, 5.0];
 a8^1 := [0.3, 0.0, 0.0];
 a8^2 := [0.0, 0.5, 0.0];
```

```
a9^0 := [0.0, 1.0, 5.0];
a9^1 := [0.3, 0.0, 0.0];
a9^2 := [0.0, 0.5, 0.0];

a10^0 := [0.0, 1.0, 5.0];
a10^1 := [0.3, 0.0, 0.0];
a10^2 := [0.0, 0.5, 0.0];

a11^0 := [0.0, 1.0, 5.0];
a11^1 := [0.3, 0.0, 0.0];
a11^2 := [0.0, 0.5, 0.0];

a12^0 := [0.0, 1.0, 5.0];
a12^1 := [0.3, 0.0, 0.0];
a12^2 := [0.0, 0.5, 0.0];

a13^0 := [0.0, 1.0, 5.0];
a13^1 := [0.3, 0.0, 0.0];
a13^2 := [0.0, 0.5, 0.0];

a14^0 := [0.0, 1.0, 5.0];
a14^1 := [0.3, 0.0, 0.0];
a14^2 := [0.0, 0.5, 0.0];

a15^0 := [0.0, 1.0, 5.0];
a15^1 := [0.3, 0.0, 0.0];
a15^2 := [0.0, 0.5, 0.0];

n1[0]:=10.0;
n1[1]:=0.0;
n1[2]:=0.0;

n2[0]:=10.0;
n2[1]:=0.0;
n2[2]:=0.0;

n3[0]:=10.0;
n3[1]:=0.0;
n3[2]:=0.0;

n4[0]:=10.0;
n4[1]:=0.0;
n4[2]:=0.0;

n5[0]:=10.0;
n5[1]:=0.0;
```

```
n5[2]:=0.0;

n6[0]:=10.0;
n6[1]:=0.0;
n6[2]:=0.0;

n7[0]:=10.0;
n7[1]:=0.0;
n7[2]:=0.0;

n8[0]:=10.0;
n8[1]:=0.0;
n8[2]:=0.0;

n9[0]:=10.0;
n9[1]:=0.0;
n9[2]:=0.0;

n10[0]:=10.0;
n10[1]:=0.0;
n10[2]:=0.0;

n11[0]:=10.0;
n11[1]:=0.0;
n11[2]:=0.0;

n12[0]:=10.0;
n12[1]:=0.0;
n12[2]:=0.0;

n13[0]:=10.0;
n13[1]:=0.0;
n13[2]:=0.0;

n14[0]:=10.0;
n14[1]:=0.0;
n14[2]:=0.0;

n15[0]:=10.0;
n15[1]:=0.0;
n15[2]:=0.0;


legend(g, true);
legends(g, 'n1', 0);
legends(g, 'n2', 1);
legends(g, 'n3', 2);
```

```
legends(g, 'n4', 3);
legends(g, 'n5', 4);
legends(g, 'n6', 5);
legends(g, 'n7', 6);
legends(g, 'n8', 7);
legends(g, 'n9', 8);
legends(g, 'n10', 9);
legends(g, 'n11', 10);
legends(g, 'n12', 11);
legends(g, 'n13', 12);
legends(g, 'n14', 13);
legends(g, 'n15', 14);

xlabel(g, 'tiempo t');
ylabel(g, 'número de individuos');
label(g, 'Gráfico de Leslie');

t:=strtoint(inputbox('Tiempo t','calcular hasta t=','10'));

addstr(w, '');
addstr(w, 'calculando hasta tiempo t = '+inttostr(t));

for i:=0 to t-1 do
 begin
a1[0,1]:=h(i,1);
a2[0,1]:=h(i,1);
a3[0,1]:=h(i,1);
a4[0,1]:=h(i,1);
a5[0,1]:=h(i,1);
a6[0,1]:=h(i,1);
a7[0,1]:=h(i,1);
a8[0,1]:=h(i,1);
a9[0,1]:=h(i,1);
a10[0,1]:=h(i,1);
a11[0,1]:=h(i,1);
a12[0,1]:=h(i,1);
a13[0,1]:=h(i,1);
a14[0,1]:=h(i,1);
a15[0,1]:=h(i,1);

a1[0,2]:=h(i,2);
a2[0,2]:=h(i,2);
a3[0,2]:=h(i,2);
a4[0,2]:=h(i,2);
a5[0,2]:=h(i,2);
a6[0,2]:=h(i,2);
a7[0,2]:=h(i,2);
```

```
 a8[0,2]:=h(i,2);
 a9[0,2]:=h(i,2);
 a10[0,2]:=h(i,2);
 a11[0,2]:=h(i,2);
 a12[0,2]:=h(i,2);
 a13[0,2]:=h(i,2);
 a14[0,2]:=h(i,2);
 a15[0,2]:=h(i,2);

 multiplicar(n1, a1);
 multiplicar(n2, a2);
 multiplicar(n3, a3);
 multiplicar(n4, a4);
 multiplicar(n5, a5);
 multiplicar(n6, a6);
 multiplicar(n7, a7);
 multiplicar(n8, a8);
 multiplicar(n9, a9);
 multiplicar(n10, a10);
 multiplicar(n11, a11);
 multiplicar(n12, a12);
 multiplicar(n13, a13);
 multiplicar(n14, a14);
 multiplicar(n15, a15);

 addrpoints(g, n1[0]+n1[1]+n1[2], '', 10,0);
 addrpoints(g, n2[0]+n2[1]+n2[2], '', 10,1);
 addrpoints(g, n3[0]+n3[1]+n3[2], '', 10,2);
 addrpoints(g, n4[0]+n4[1]+n4[2], '', 10,3);
 addrpoints(g, n5[0]+n5[1]+n5[2], '', 10,4);
 addrpoints(g, n6[0]+n6[1]+n6[2], '', 10,5);
 addrpoints(g, n7[0]+n7[1]+n7[2], '', 10,6);
 addrpoints(g, n8[0]+n8[1]+n8[2], '', 10,7);
 addrpoints(g, n9[0]+n9[1]+n9[2], '', 10,8);
 addrpoints(g, n10[0]+n10[1]+n10[2], '', 10,9);
 addrpoints(g, n11[0]+n11[1]+n11[2], '', 10,10);
 addrpoints(g, n12[0]+n12[1]+n12[2], '', 10,11);
 addrpoints(g, n13[0]+n13[1]+n13[2], '', 10,12);
 addrpoints(g, n14[0]+n14[1]+n14[2], '', 10,13);
 addrpoints(g, n15[0]+n15[1]+n15[2], '', 10,14);

 repaint(g);

end;


pause ('Simulación Finalizada. Pulse una tecla para continuar.');
```

```
end.
```

Resultados

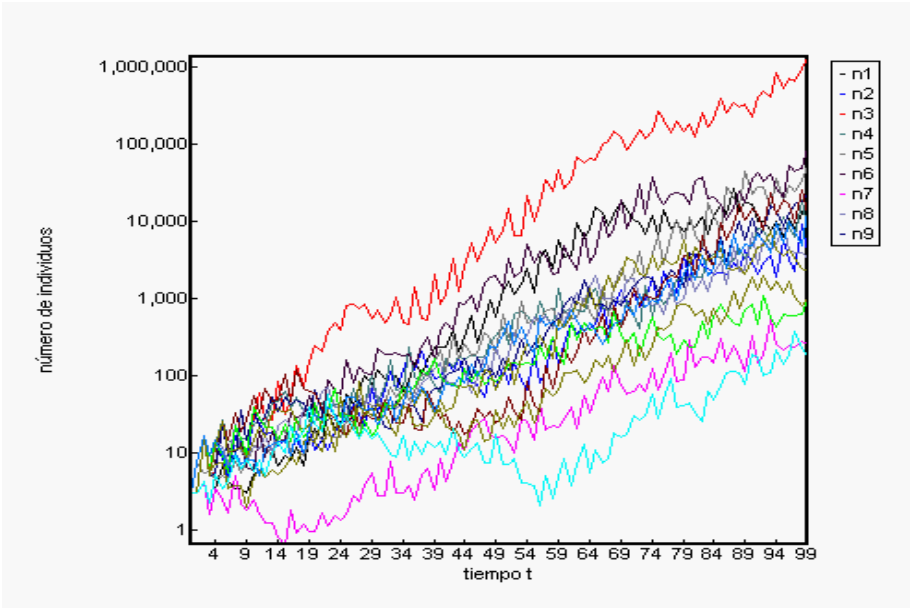

Figure 4.5: Leslie Estocástico

Podemos ver aquí el resultado de graficar simultáneamente 9 poblaciones estocásticas. Recordamos que en cada unidad de tiempo, las fertilidades son procesos estocásticos con probabilidad independiente 1/2 de valer 1/2 o 2. Podemos ver que las poblaciones fluctúan bastante, y resulta difícil predecir el comportamiento de la dinámica poblacional. Se puede ver que en realidad la varianza crece en forma lineal y mantenida a medida que avanza el tiempo, en la siguiente sección.

Listado

```
uses dialogs, sysutils, windows, arithmetic, matrixext;

global w:textwindow;
g:linegraphwindow;

function h(t,i:integer):real;
var result:real;
begin

if random<0.5 then result:=0.5 else result:=2.0;

if i=2 then h:=5.0*result else h:=result;
```

```
end;

var i,j,t:integer;
  a:matrix [3,3] of real;
n:vector [3] of real;
m:vector [300] of real;
v:vector [1000] of real;

begin

 scalelog(g, true);

 titulo(w, 'Información');
 titulo(g, 'Gráfico');

 legend(g, true);
 legends(g, 'media', 0);
 legends(g, 'varianza', 1);

 width(w, 300);
 width(g, 500);
 left(g, 305);

 addstr(w, '');

 a^0 := [0.0, 1.0, 5.0];
 a^1 := [0.3, 0.0, 0.0];
 a^2 := [0.0, 0.5, 0.0];

 n[0]:=10.0;
 n[1]:=0.0;
 n[2]:=0.0;

 xlabel(g, 'tiempo t');
 ylabel(g, 'número de individuos');
 label(g, 'Media y Varianza');

 t:=strtoint(inputbox('Tiempo t','calcular hasta t=','100'));

 addstr(w, '');
 addstr(w, 'calculando hasta tiempo t = '+inttostr(t));

 for j:=0 to 999 do
  begin
m[j]:=0.0;
v[j]:=0.0;
end;
```

```
for j:=0 to 999 do
 begin

  n[0]:=10.0;
  n[1]:=0.0;
  n[2]:=0.0;

  for i:=0 to t-1 do
  begin
 a[0,1]:=h(i,1);
 a[0,2]:=h(i,2);

 multiplicar(n, a);

 m[i]:=m[i]+((n[0]+n[1]+n[2]));
 v[i]:=v[i]+(sqr((n[0]+n[1]+n[2])));

end;

  addstr(w, 'm= '+inttostr(j));
  repaint(w);
 end;

 for i:=0 to t-1 do
  begin
 addrpoints(g, (m[i]/999.0), '', 10,0);
 addrpoints(g, ( (v[i]/999.0-sqr(m[i]/999.0) )), '', 10,1);
 repaint(g);
end;

savebmp(g, 'ej2-7.bmp');

pause ('Simulación Finalizada. Pulse una tecla para continuar.');
end.
```

Resultados

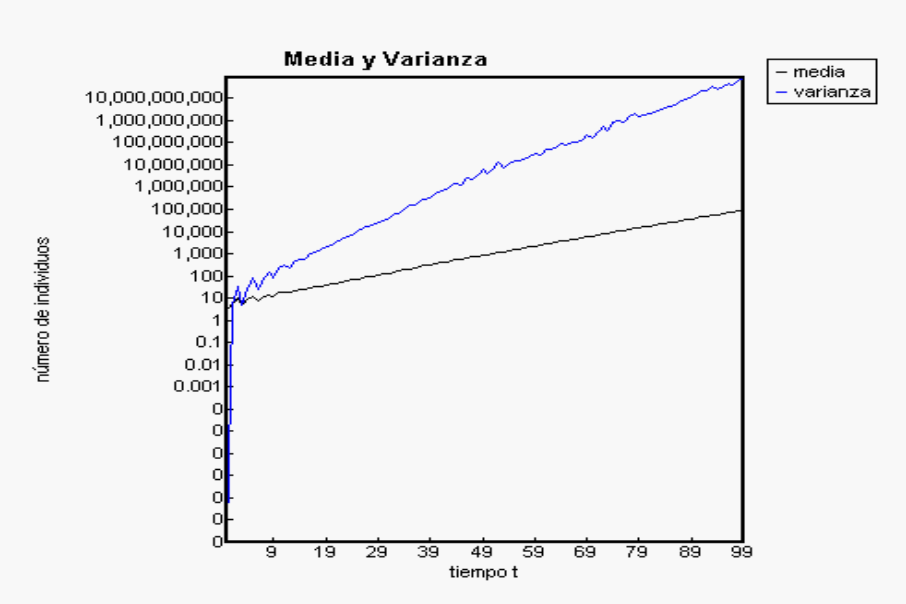

Figure 4.6: Media y Varianza

Graficamos la media y varianza de 1000 poblaciones estocásticas.

4.1.5 Modelo Densodependiente

Listado

```
uses dialogs, sysutils, windows, arithmetic, matrixext;

global w:textwindow;
g:linegraphwindow;

function f(n:real):real;
var b:real;
begin
b:=-0.005;
f:=exp(b*n);
end;

var i,j,t:integer;
  a:matrix [3,3] of real;
n:vector [3] of real;
r:real;
```

```
begin

 titulo(w, 'Información');
 titulo(g, 'Gráfico');

 width(w, 300);
 width(g, 500);
 left(g, 305);

 addstr(w, '');

 a^0 := [0.0, 1.0, 5.0];
 a^1 := [0.3, 0.0, 0.0];
 a^2 := [0.0, 0.5, 0.0];

 n[0]:=0.33;
 n[1]:=0.33;
 n[2]:=0.33;

 addstr(w, 'matriz A:');
 addstr(w, '');
 mostrar(a);
 addstr(w, '');
 addstr(w, 'Matriz de población inicial n:');
 addstr(w, '');
 mostrar_n(n);

 legend(g, true);
 legends(g, 'n[0]', 0);
 legends(g, 'n[1]', 1);
 legends(g, 'n[2]', 2);

 xlabel(g, 'tiempo t');
 ylabel(g, 'número de individuos');
 label(g, 'Gráfico de Leslie');

 t:=strtoint(inputbox('Tiempo t','calcular hasta t=','10'));

 addstr(w, '');
 addstr(w, 'calculando hasta tiempo t = '+inttostr(t));

 for i:=0 to t-1 do
  begin
 multiplicar(n, a);
 a[0,1]:=f(n[0]+n[1]+n[2]);
 a[0,2]:=5.0*f(n[0]+n[1]+n[2]);
 addrpoints(g, n[0], '', 10,0);
```

```
 addrpoints(g, n[1], '', 10,1);
 addrpoints(g, n[2], '', 10,2);
 repaint(g);
end;

pause ('Simulación Finalizada. Pulse una tecla para continuar.');
end.
```

Resultados

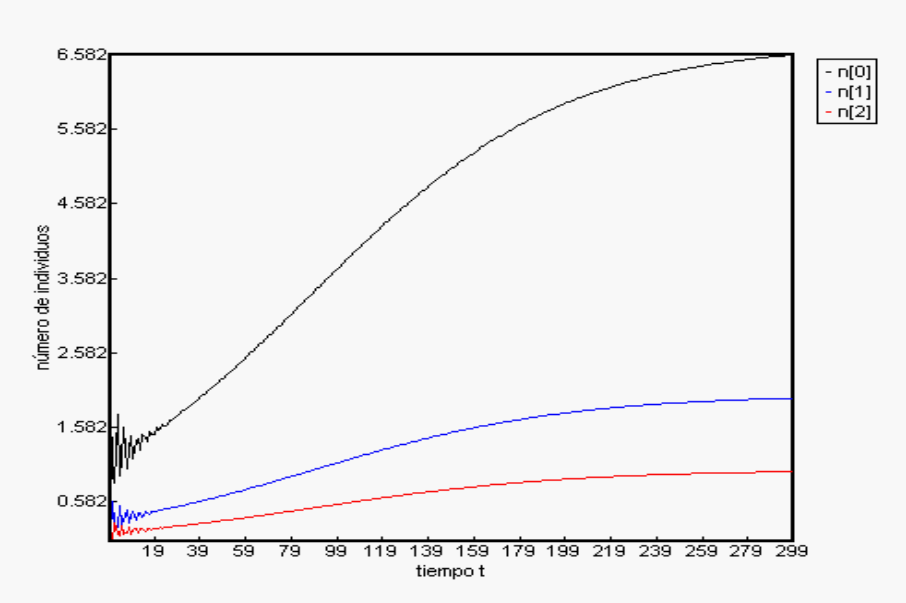

Figure 4.7: Leslie densodependiente

En este caso, podemos observar el desarrollo de una población densodependiente. Se puede notar como al principio la población fluctúa un poco y luego comienza a crecer exponencialmente, hasta que la densidad comienza a ejercer fuerza y entonces las fertilidades comienzan a decaer en las diferentes clases de edad.

4.1.6 Estados Iniciales y Ergodicidad

Listado

```
uses dialogs, sysutils, windows, arithmetic, matrixext;

global w:textwindow;
g:linegraphwindow;
b:real;
```

```
function f(n:real):real;
begin
f:=exp(-b*n);
end;


var i,j,t:integer;
  a1:matrix [3,3] of real;
  a2:matrix [3,3] of real;
  a3:matrix [3,3] of real;
  a4:matrix [3,3] of real;
  a5:matrix [3,3] of real;
  a6:matrix [3,3] of real;
  a7:matrix [3,3] of real;
  a8:matrix [3,3] of real;
  a9:matrix [3,3] of real;
n1:vector [3] of real;
n2:vector [3] of real;
n3:vector [3] of real;
n4:vector [3] of real;
n5:vector [3] of real;
n6:vector [3] of real;
n7:vector [3] of real;
n8:vector [3] of real;
n9:vector [3] of real;
r,t1,t2,t3,t4,t5,t6,t7,t8,t9:real;

begin

 titulo(w, 'Información');
 titulo(g, 'Gráfico');

 width(w, 300);
 width(g, 500);
 left(g, 305);

 addstr(w, '');

 a1^0 := [0.0, 1.0, 5.0];
 a1^1 := [0.3, 0.0, 0.0];
 a1^2 := [0.0, 0.5, 0.0];

 a2^0 := [0.0, 1.0, 5.0];
 a2^1 := [0.3, 0.0, 0.0];
 a2^2 := [0.0, 0.5, 0.0];
```

```
 a3^0 := [0.0, 1.0, 5.0];
 a3^1 := [0.3, 0.0, 0.0];
 a3^2 := [0.0, 0.5, 0.0];

 a4^0 := [0.0, 1.0, 5.0];
 a4^1 := [0.3, 0.0, 0.0];
 a4^2 := [0.0, 0.5, 0.0];

 a5^0 := [0.0, 1.0, 5.0];
 a5^1 := [0.3, 0.0, 0.0];
 a5^2 := [0.0, 0.5, 0.0];

 a6^0 := [0.0, 1.0, 5.0];
 a6^1 := [0.3, 0.0, 0.0];
 a6^2 := [0.0, 0.5, 0.0];

 a7^0 := [0.0, 1.0, 5.0];
 a7^1 := [0.3, 0.0, 0.0];
 a7^2 := [0.0, 0.5, 0.0];

 a8^0 := [0.0, 1.0, 5.0];
 a8^1 := [0.3, 0.0, 0.0];
 a8^2 := [0.0, 0.5, 0.0];

 a9^0 := [0.0, 1.0, 5.0];
 a9^1 := [0.3, 0.0, 0.0];
 a9^2 := [0.0, 0.5, 0.0];

 r:=4.0;

for i:=0 to 2 do n1[i]:=random*r;
for i:=0 to 2 do n2[i]:=random*r;
for i:=0 to 2 do n3[i]:=random*r;
for i:=0 to 2 do n4[i]:=random*r;
for i:=0 to 2 do n5[i]:=random*r;
for i:=0 to 2 do n6[i]:=random*r;
for i:=0 to 2 do n7[i]:=random*r;
for i:=0 to 2 do n8[i]:=random*r;
for i:=0 to 2 do n9[i]:=random*r;

 legend(g, true);
 legends(g, 'n1', 0);
 legends(g, 'n2', 1);
 legends(g, 'n3', 2);
 legends(g, 'n4', 3);
 legends(g, 'n5', 4);
 legends(g, 'n6', 5);
```

```
legends(g, 'n7', 6);
legends(g, 'n8', 7);
legends(g, 'n9', 8);

xlabel(g, 'tiempo t');
ylabel(g, 'número de individuos');
label(g, 'Gráfico de Leslie');

t:=strtoint(inputbox('Tiempo t','calcular hasta t=','10'));
b:=strtofloat(inputbox('Parámetro b (fuerza de dependencia)','b =','0.005'));

addstr(w, '');
addstr(w, 'calculando hasta tiempo t = '+inttostr(t));
addstr(w, '');
addstr(w, 'b = '+floattostr(b));

for i:=0 to t-1 do
 begin

t1:=f(n1[0]+n1[1]+n1[2]);
t2:=f(n2[0]+n2[1]+n2[2]);
t3:=f(n3[0]+n3[1]+n3[2]);
t4:=f(n4[0]+n4[1]+n4[2]);
t5:=f(n5[0]+n5[1]+n5[2]);
t6:=f(n6[0]+n6[1]+n6[2]);
t7:=f(n7[0]+n7[1]+n7[2]);
t8:=f(n8[0]+n8[1]+n8[2]);
t9:=f(n9[0]+n9[1]+n9[2]);

a1[0,1]:=t1;
a2[0,1]:=t2;
a3[0,1]:=t3;
a4[0,1]:=t4;
a5[0,1]:=t5;
a6[0,1]:=t6;
a7[0,1]:=t7;
a8[0,1]:=t8;
a9[0,1]:=t9;

a1[0,2]:=5.0*t1;
a2[0,2]:=5.0*t2;
a3[0,2]:=5.0*t3;
a4[0,2]:=5.0*t4;
a5[0,2]:=5.0*t5;
a6[0,2]:=5.0*t6;
a7[0,2]:=5.0*t7;
a8[0,2]:=5.0*t8;
```

```
 a9[0,2]:=5.0*t9;

 multiplicar(n1, a1);
 multiplicar(n2, a2);
 multiplicar(n3, a3);
 multiplicar(n4, a4);
 multiplicar(n5, a5);
 multiplicar(n6, a6);
 multiplicar(n7, a7);
 multiplicar(n8, a8);
 multiplicar(n9, a9);

 addrpoints(g, n1[0]+n1[1]+n1[2], '', 10,0);
 addrpoints(g, n2[0]+n2[1]+n2[2], '', 10,1);
 addrpoints(g, n3[0]+n3[1]+n3[2], '', 10,2);
 addrpoints(g, n4[0]+n4[1]+n4[2], '', 10,3);
 addrpoints(g, n5[0]+n5[1]+n5[2], '', 10,4);
 addrpoints(g, n6[0]+n6[1]+n6[2], '', 10,5);
 addrpoints(g, n7[0]+n7[1]+n7[2], '', 10,6);
 addrpoints(g, n8[0]+n8[1]+n8[2], '', 10,7);
 addrpoints(g, n9[0]+n9[1]+n9[2], '', 10,8);

 repaint(g);

end;

pause ('Simulación Finalizada. Pulse una tecla para continuar.');
end.
```

Resultados

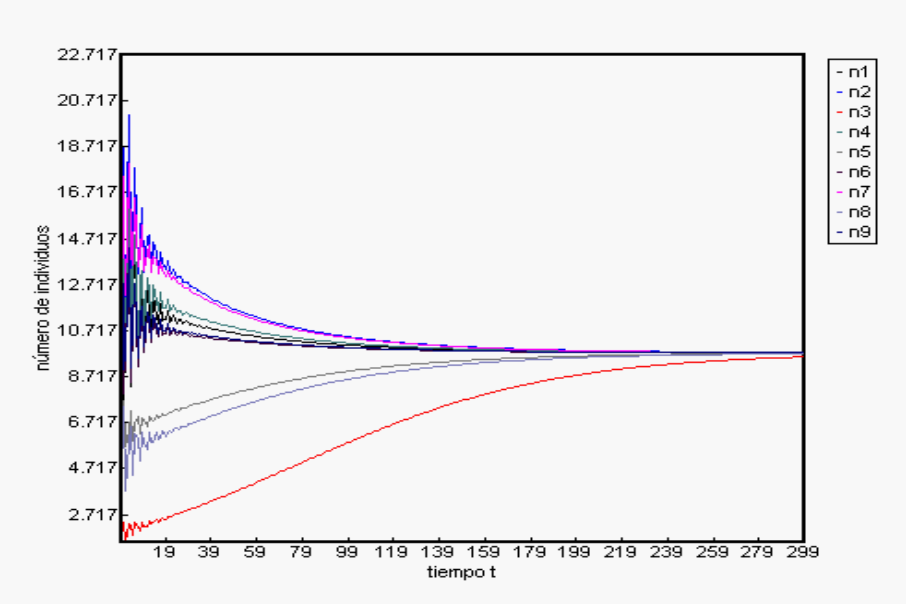

Figure 4.8: Ergodicidad

En este gráfico podemos ver 9 poblaciones graficadas simultáneamente. Cada población comienza con un estado inicial aleatorio, pero podemos ver que convergen a una misma cantidad de individuos a medida que transcurre el tiempo. Esto se debe a la característica de ergodicidad de los modelos densodependientes.

4.1.7 Sensibilidad del parámetro R, en $g(N) = Re^{-bN}$

Listado

```
uses dialogs, sysutils, windows, arithmetic, matrixext;

global w:textwindow;
g:linegraphwindow;
fac:real;

function f(n:real):real;
var b:real;
begin
b:=-0.005;
f:=fac*exp(b*n);
end;

var i,j,t:integer;
```

```
  a:matrix [3,3] of real;
n:vector [3] of real;
r:real;

begin

 titulo(w, 'Información');
 titulo(g, 'Gráfico');

 width(w, 300);
 width(g, 500);
 left(g, 305);

 addstr(w, '');

 a^0 := [0.0, 1.0, 5.0];
 a^1 := [0.3, 0.0, 0.0];
 a^2 := [0.0, 0.5, 0.0];


 n[0]:=0.33;
 n[1]:=0.33;
 n[2]:=0.33;

 addstr(w, 'matriz A:');
 addstr(w, '');
 mostrar(a);
 addstr(w, '');
 addstr(w, 'Matriz de población inicial n:');
 addstr(w, '');
 mostrar_n(n);


 t:=strtoint(inputbox('Tiempo t','calcular hasta t=','10'));

 fac:=strtofloat(inputbox('factor R','R=','2.0'));

 addstr(w, '');
 addstr(w, 'calculando hasta tiempo t = '+inttostr(t));
 addstr(w, '');
 addstr(w, 'Rojo: n[0]+n[1]+n[2].');


 for i:=0 to t-1 do
  begin
 multiplicar(n, a);
 a[0,1]:=f(n[0]+n[1]+n[2]);
```

```
 a[0,2]:=5.0*f(n[0]+n[1]+n[2]);
 addrpoints(g, n[0]+n[1]+n[2], '', 10,0);
 repaint(g);
end;

pause ('Simulación Finalizada. Pulse una tecla para continuar.');
end.
```

Resultados

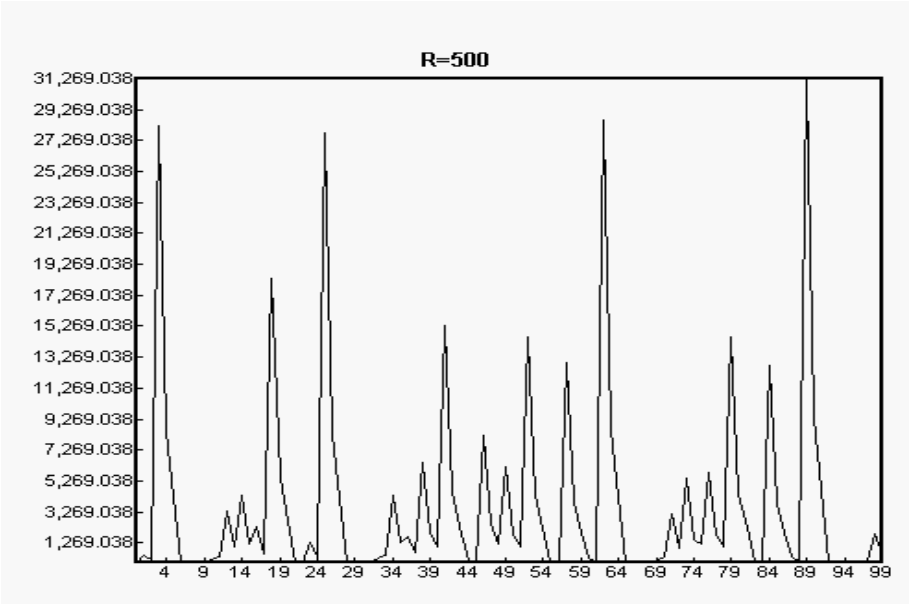

Figure 4.9: Párametro R=500

Aquí podemos observar que si hacemos $R = 500$, las fertilidades se vuelven incontrolables, en cuyo caso la población muestra un desarrollo caótico, en el cual no podemos obtener conclusiones.

4.2 Cálculos Analíticos

Se estudiarán tres casos diferentes:

1. Población humana estadounidense de 1966. Caso de modelo clasificado por edad, 10 clases de edad de 5 años cada una y paso de tiempo de 5 años.

2. Población de tortuga desértica (*Gopherus agassizii*). Caso de modelo clasificado por tamaño, 8 etapas.

3. Población de plantas cardenchas (*Dipsacus sylvestris*). Caso patológico, con una sola etapa reproductiva, 6 etapas, 1 año de paso de tiempo.

4.2.1 Caso de estudio: Población humana estadounidense del año 1966 - USA66

Matriz de proyección poblacional

$$\begin{pmatrix}
0.00000 & \mathbf{0.00102} & \mathbf{0.08515} & \mathbf{0.30574} & \mathbf{0.40002} & \mathbf{0.28061} & \mathbf{0.15260} & \mathbf{0.06420} & \mathbf{0.01483} & \mathbf{0.00089} \\
\mathbf{0.99670} & 0.00000 & 0.00000 & 0.00000 & 0.00000 & 0.00000 & 0.00000 & 0.00000 & 0.00000 & 0.00000 \\
0.00000 & \mathbf{0.99837} & 0.00000 & 0.00000 & 0.00000 & 0.00000 & 0.00000 & 0.00000 & 0.00000 & 0.00000 \\
0.00000 & 0.00000 & \mathbf{0.99780} & 0.00000 & 0.00000 & 0.00000 & 0.00000 & 0.00000 & 0.00000 & 0.00000 \\
0.00000 & 0.00000 & 0.00000 & \mathbf{0.99672} & 0.00000 & 0.00000 & 0.00000 & 0.00000 & 0.00000 & 0.00000 \\
0.00000 & 0.00000 & 0.00000 & 0.00000 & \mathbf{0.99607} & 0.00000 & 0.00000 & 0.00000 & 0.00000 & 0.00000 \\
0.00000 & 0.00000 & 0.00000 & 0.00000 & 0.00000 & \mathbf{0.99472} & 0.00000 & 0.00000 & 0.00000 & 0.00000 \\
0.00000 & 0.00000 & 0.00000 & 0.00000 & 0.00000 & 0.00000 & \mathbf{0.99240} & 0.00000 & 0.00000 & 0.00000 \\
0.00000 & 0.00000 & 0.00000 & 0.00000 & 0.00000 & 0.00000 & 0.00000 & \mathbf{0.98867} & 0.00000 & 0.00000 \\
0.00000 & 0.00000 & 0.00000 & 0.00000 & 0.00000 & 0.00000 & 0.00000 & 0.00000 & \mathbf{0.98274} & 0.00000
\end{pmatrix}$$

Distribución Estable de clases de edad

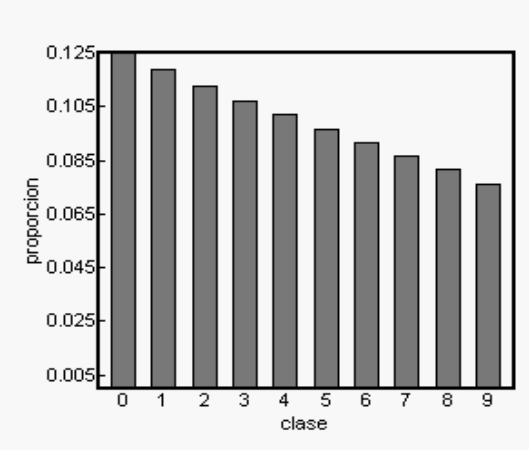

Figure 4.10: Distribución estable

El patrón mostrado por $\mathbf{w}_1$ (a medida que avanzan las clases de edad, decrecen las propociones) es típico de los modelos clasificados por edad.

Valor Reproductivo

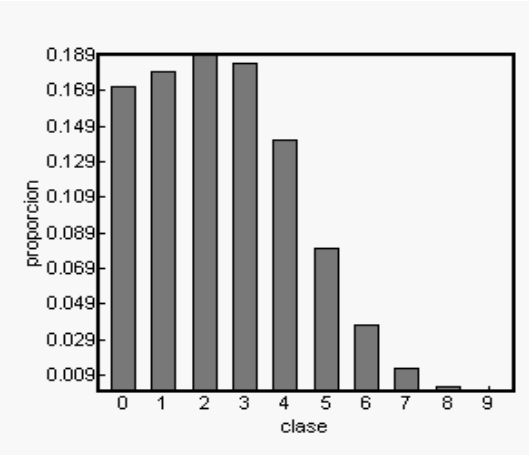

Figure 4.11: Valor Reproductivo

Este patrón también es típico de los modelos clasificados por edad; se puede observar un máximo en la clase de edad 3 (correspondiente a la edad 10-15 años).

Sensibilidad de λ_1

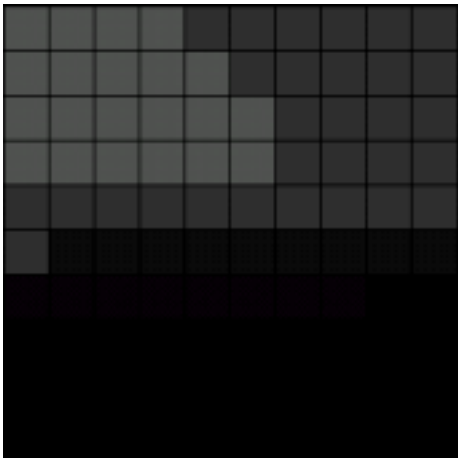

Figure 4.12: Sensibilidad - Oscuro significa *menos* sensible

Podemos ver aquí que las fertilidades λ_i decrecen a medida que avanzan las clases de edad, al igual que las probabilidades de supervivencia p_i, aunque estas últimas decrecen mucho más (en realidad, se puede ver[2] que esta caída es exponencial si $\lambda_1 > 1$ y los p_i son constantes.

Elasticidad de λ_1

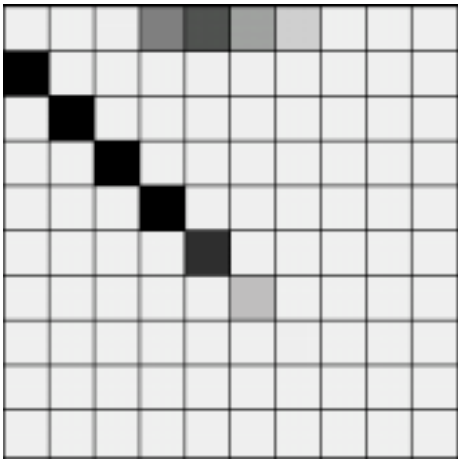

Figure 4.13: Elasticidad - Oscuro significa *mas* sensible

Estas elasticidades son típicas de los modelos clasificados por edad. Muestra que las elasticidades de las probabilidades de supervivencia decaen consistentemente con la edad, mientras que las fertilidades primero crece y luego declina.

Listado

```
uses dialogs, sysutils, windows, matrixext, arithmetic;

global w:textwindow;
g1,g2:bargraphwindow;
s1,s2:sengraphwindow;

var i,j,t:integer;
  a:matrix [10,10] of real;
temp:matrix [10,10] of real;
autovalores:vector [10] of real;
autovalores_im:vector [10] of real;
autovector:vector [10] of real;
lautovector:vector [10] of real;
fi:vector [10] of real;
r,r2,dr, f,s:real;
b:boolean;
result:integer;

begin

 a^0 := [ 0.00000, 0.00102, 0.08515, 0.30574, 0.40002, 0.28061, 0.15260, 0.06420, 0.01483, 0.00089 ];
 a^1 := [ 0.99670, 0.00000, 0.00000, 0.00000, 0.00000, 0.00000, 0.00000, 0.00000, 0.00000, 0.00000 ];
```

```
a^2 := [ 0.00000, 0.99837, 0.00000, 0.00000, 0.00000, 0.00000, 0.00000, 0.00000, 0.00000, 0.00000 ];
a^3 := [ 0.00000, 0.00000, 0.99780, 0.00000, 0.00000, 0.00000, 0.00000, 0.00000, 0.00000, 0.00000 ];
a^4 := [ 0.00000, 0.00000, 0.00000, 0.99672, 0.00000, 0.00000, 0.00000, 0.00000, 0.00000, 0.00000 ];
a^5 := [ 0.00000, 0.00000, 0.00000, 0.00000, 0.99607, 0.00000, 0.00000, 0.00000, 0.00000, 0.00000 ];
a^6 := [ 0.00000, 0.00000, 0.00000, 0.00000, 0.00000, 0.99472, 0.00000, 0.00000, 0.00000, 0.00000 ];
a^7 := [ 0.00000, 0.00000, 0.00000, 0.00000, 0.00000, 0.00000, 0.99240, 0.00000, 0.00000, 0.00000 ];
a^8 := [ 0.00000, 0.00000, 0.00000, 0.00000, 0.00000, 0.00000, 0.00000, 0.98867, 0.00000, 0.00000 ];
a^9 := [ 0.00000, 0.00000, 0.00000, 0.00000, 0.00000, 0.00000, 0.00000, 0.00000, 0.98274, 0.00000 ];

addstr(w, 'Matriz A correspondiente a la población USA 1966');
mostrar(a);
copiarmatriz(a, temp);

result:=eig(0,10,a, autovalores, autovalores_im);

r:=absmay(autovalores);
r2:=absmay(autovalores_im);

if r2>r then r:=r2;

r2:=ln(r);

addstr(w, 'net fertility functions:');
copiarmatriz(temp, a);

netf(0,10, a, fi);
mostrar_n(fi);

copiarmatriz(temp, a);

addstr(w, '');

s:=sumavec(fi);

addstr(w, 'net rep rate: '+floattostr(s)) ;

addstr(w, '');

addstr(w, 'gen time: '+floattostr(ln(s)/ln(r)));

addstr(w, 'Autovalores:');

mostrar_n(autovalores);
addstr(w, '');
mostrar_n(autovalores_im);

copiarmatriz(temp, a);
```

```
 addstr(w, 'Rate of increase:');

 addstr(w, 'r='+floattostr(r2));

 result:=eigvect(0,10,r,a,autovector);

 addstr(w, '');
 addstr(w, 'autovector de '+floattostr(r));
 addstr(w, '');
 mostrar_n(autovector);

 transp(temp, a);

 result:=eigvect(0,10,r,a,lautovector);

 addstr(w, '');
 addstr(w, 'autovector izq. de '+floattostr(r));
 addstr(w, '');
 mostrar_n(lautovector);

 copiarmatriz(temp, a);

 addstr(w, '');
 addstr(w, 'escalados');
 addstr(w, '');
 escalar(autovector);
 mostrar_n(autovector);
 addstr(w, '');
 escalar(lautovector);
 mostrar_n(lautovector);

 r2 := abssegmay(autovalores);
 dr := absmay(autovalores_im);

 if dr>r2 then r2:=dr;

 dr := r / r2;

 addstr(w, '');
 addstr(w, 'Damping ratio:= '+floattostr(dr));

 for i:=0 to 9 do addrpoint(g1, autovector[i], inttostr(i), 10);
repaint(g1);

 for i:=0 to 9 do addrpoint(g2, lautovector[i], inttostr(i), 10);
repaint(g2);
```

```
inisen(s1, 10, 10);
inisen(s2, 10, 10);

 addstr(w, 'sensitivity:');

 for i:=0 to 9 do
  for j:=0 to 9 do
   begin
   s := log10(sensitivity(i,j,lautovector,autovector));
   addstr(w, floattostr(s));
   brush(s1, trunc(-2.0*s));
   pintar(s1, i, j);
   end;

 addstr(w, 'elasticity:'+floattostr(a[i,j]));

 for i:=0 to 9 do
  for j:=0 to 9 do
   begin
   s:=elasticity(i,j,r,a,lautovector,autovector);
   addstr(w, floattostr(s));
   brush(s2, trunc(40000.0*s));
   pintar(s2,  i, j);
   end;

pause ('Programa Finalizado. Pulse una tecla para continuar.');
end.
```

4.2.2 Caso de estudio: Población tortuga desértica

Matriz de proyección poblacional

$$\begin{pmatrix} 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & \mathbf{1.300} & \mathbf{1.980} & \mathbf{2.570} \\ \mathbf{0.716} & \mathbf{0.567} & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & \mathbf{0.149} & \mathbf{0.567} & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & \mathbf{0.149} & \mathbf{0.604} & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & \mathbf{0.235} & \mathbf{0.560} & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.000 & \mathbf{0.225} & \mathbf{0.678} & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & \mathbf{0.249} & \mathbf{0.851} & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & \mathbf{0.016} & \mathbf{0.860} \end{pmatrix}$$

Distribución Estructural Estable

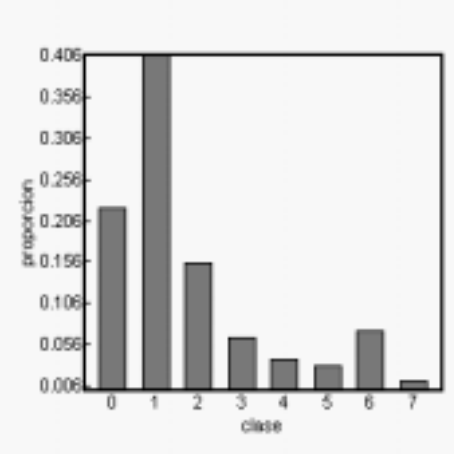

Figure 4.14: Distribución estable

Vemos aquí que gran parte de la población se mantiene en la etapa pre-reproductiva. Solamente en la etapa 6 se ve un cierto incremento.

Valor Reproductivo

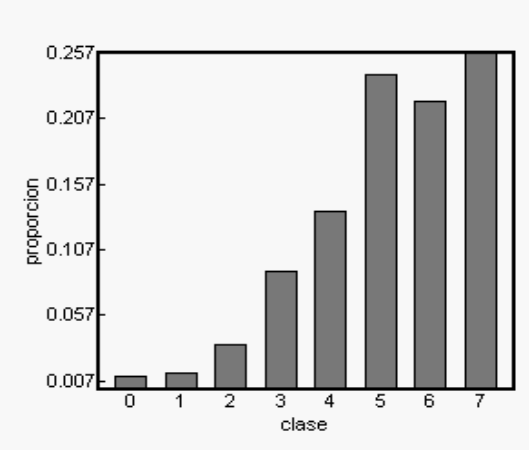

Figure 4.15: Valor Reproductivo

Podemos ver que los valores reproductivos van creciendo a medida que se acercan a las etapas reproductivas.

Sensibilidad de λ_1

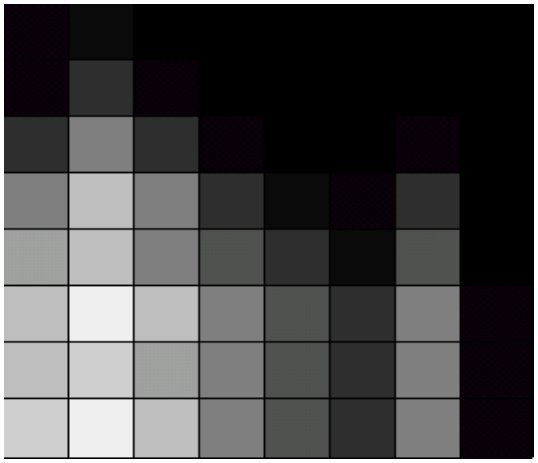

Figure 4.16: Sensibilidad - Oscuro significa *menos* sensible

En este gráfico de sensibilidad podemos observar que λ_1 es más sensible en los las probabilidades de supervivencia y paso a siguiente etapa, y de supervivencia y quedarse en la misma etapa, principalmente en las etapas 2, 3 y 7.

Elasticidad de λ_1

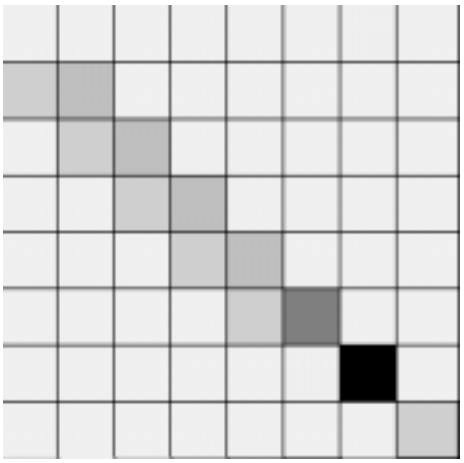

Figure 4.17: Elasticidad - Oscuro significa *mas* sensible

Aquí podemos ver que la probabilidad de sobrevivir y quedarse en la etapa 7 es la elasticidad más grande.

Listado

```
uses dialogs, sysutils, windows, matrixext, arithmetic;

global w:textwindow;
g1,g2:bargraphwindow;
s1,s2:sengraphwindow;

var i,j,t:integer;
  a:matrix [8,8] of real;
temp:matrix [8,8] of real;
autovalores:vector [8] of real;
autovalores_im:vector [8] of real;
autovector:vector [8] of real;
lautovector:vector [8] of real;
fi:vector [8] of real;
r,r2,dr, f,s:real;
b:boolean;
result:integer;

begin
 titulo(w, 'Cálculo de autovalores y autovectores');

 a^0 := [0.000, 0.000, 0.000, 0.000, 0.000, 1.300, 1.980, 2.570 ];
 a^1 := [0.716, 0.567, 0.000, 0.000, 0.000, 0.000, 0.000, 0.000 ];
 a^2 := [0.000, 0.149, 0.567, 0.000, 0.000, 0.000, 0.000, 0.000 ];
 a^3 := [0.000, 0.000, 0.149, 0.604, 0.000, 0.000, 0.000, 0.000 ];
 a^4 := [0.000, 0.000, 0.000, 0.235, 0.560, 0.000, 0.000, 0.000 ];
 a^5 := [0.000, 0.000, 0.000, 0.000, 0.225, 0.678, 0.000, 0.000 ];
 a^6 := [0.000, 0.000, 0.000, 0.000, 0.000, 0.249, 0.851, 0.000 ];
 a^7 := [0.000, 0.000, 0.000, 0.000, 0.000, 0.000, 0.016, 0.860 ];

 addstr(w, 'Matriz A correspondiente a una población de desert tortoise');
 mostrar(a);
 copiarmatriz(a, temp);

 result:=eig(0,8,a, autovalores, autovalores_im);

 r:=absmay(autovalores);
 r2:=absmay(autovalores_im);

 if r2>r then r:=r2;

 r2:=ln(r);

 addstr(w, 'net fertility functions:');
 copiarmatriz(temp, a);
```

```
netf(0,8, a, fi);
mostrar_n(fi);

copiarmatriz(temp, a);

addstr(w, '');

s:=sumavec(fi);

addstr(w, 'net rep rate: '+floattostr(s)) ;

addstr(w, '');

addstr(w, 'gen time: '+floattostr(ln(s)/ln(r)));

addstr(w, 'Autovalores:');

mostrar_n(autovalores);
addstr(w, '');
mostrar_n(autovalores_im);

copiarmatriz(temp, a);

addstr(w, 'Rate of increase:');

addstr(w, 'r='+floattostr(r2));

result:=eigvect(0,8,r,a,autovector);

addstr(w, '');
addstr(w, 'autovector de '+floattostr(r));
addstr(w, '');
mostrar_n(autovector);

transp(temp, a);

result:=eigvect(0,8,r,a,lautovector);

addstr(w, '');
addstr(w, 'autovector izq. de '+floattostr(r));
addstr(w, '');
mostrar_n(lautovector);

copiarmatriz(temp, a);

addstr(w, '');
addstr(w, 'escalados');
```

```
 addstr(w, '');
 escalar(autovector);
 mostrar_n(autovector);
 addstr(w, '');
 escalar(lautovector);
 mostrar_n(lautovector);

 r2 := abssegmay(autovalores);
 dr := absmay(autovalores_im);

 if dr>r2 then r2:=dr;

 dr := r / r2;

 addstr(w, '');
 addstr(w, 'Damping ratio:= '+floattostr(dr));

 for i:=0 to 7 do addrpoint(g1, autovector[i], inttostr(i), 6);
repaint(g1);

 for i:=0 to 7 do addrpoint(g2, lautovector[i], inttostr(i), 6);
repaint(g2);

inisen(s1, 8, 8);
inisen(s2, 8, 8);

 addstr(w, 'sensitivity:');

 for i:=0 to 7 do
  for j:=0 to 7 do
   begin
   s := log10(sensitivity(i,j,lautovector,autovector));
   addstr(w, floattostr(s));
   brush(s1, trunc(-2.0*s)-3);
   pintar(s1, i, j);
   end;

 addstr(w, 'elasticity:'+floattostr(a[i,j]));

 for i:=0 to 7 do
  for j:=0 to 7 do
   begin
   s:=elasticity(i,j,r,a,lautovector,autovector);
   addstr(w, floattostr(s));
   brush(s2, trunc(40000.0*s));
   pintar(s2,  i, j);
   end;
```

```
end.
```

4.2.3 Caso de estudio: Población cardencha

Matriz de proyección poblacional

$$\begin{pmatrix} 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & \mathbf{322.38} \\ \mathbf{0.966} & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \\ \mathbf{0.013} & \mathbf{0.010} & \mathbf{0.125} & 0.000 & 0.000 & \mathbf{3.448} \\ \mathbf{0.007} & 0.000 & \mathbf{0.125} & \mathbf{0.238} & 0.000 & \mathbf{30.170} \\ \mathbf{0.001} & 0.000 & 0.000 & \mathbf{0.245} & \mathbf{0.167} & \mathbf{0.862} \\ 0.000 & 0.000 & 0.000 & \mathbf{0.023} & \mathbf{0.750} & 0.000 \end{pmatrix}$$

Grafo de ciclo de vida

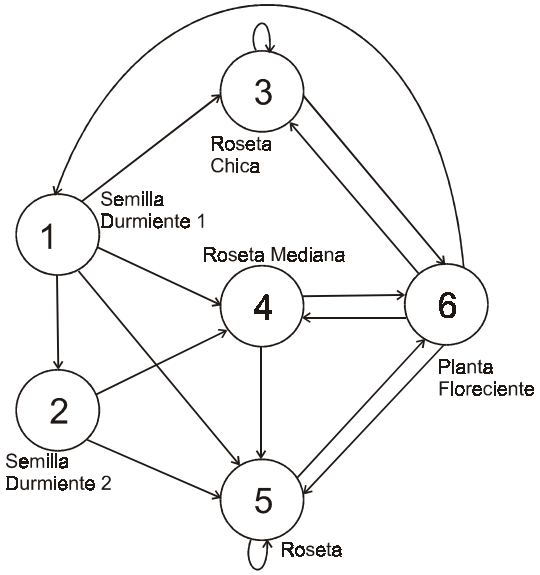

Figure 4.18: Grafo de ciclo de vida

Distribución estable de clases

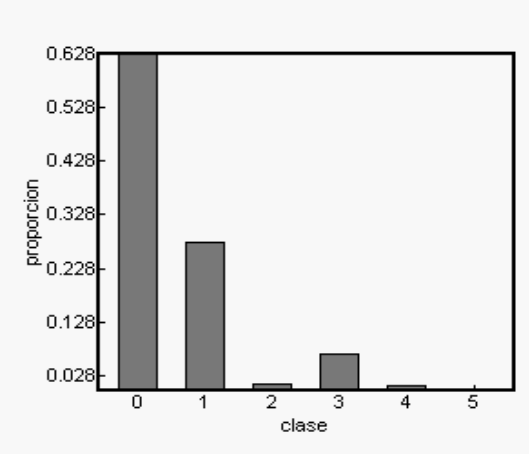

Figure 4.19: Distribución estable

En esta distribución de clases podemos ver que las mayores clases son las 1 y 2, que representan los estados de "semilla durmiente-año 1" y "semilla durmiente-año 2".

Valor Reproductivo

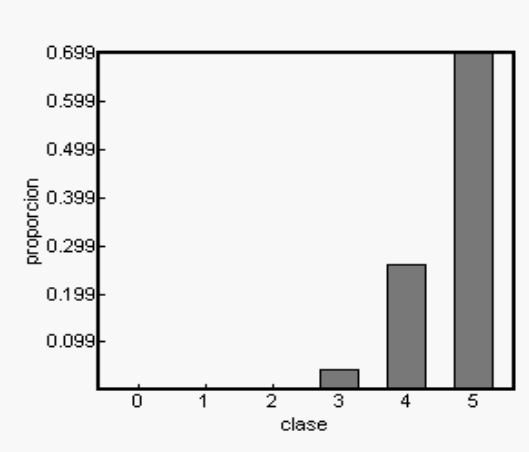

Figure 4.20: Valor Reproductivo

Se puede ver en el vector de valor reproductivo, como comienza a crecer a medida que se acerca a la etapa correspondiente a planta floreciente.

Sensibilidad de λ_1

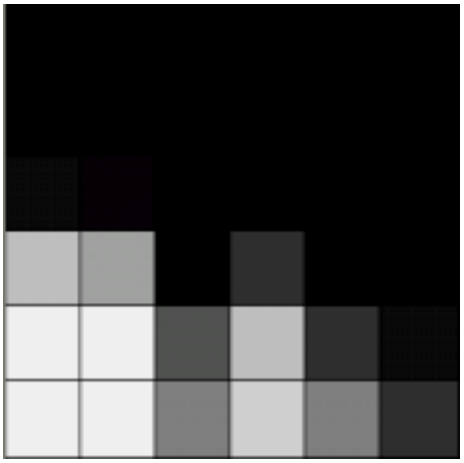

Figure 4.21: Sensibilidad - Oscuro significa *menos* sensible

En este gráfico podemos ver que las mayores sensibilidades ocurren en la esquina inferior izquierda de la matriz. Esto refleja que la mayor sensibilidad ocurre cuando se colocan "aristas" desde las etapas de semilla durmiente a planta floreciente.

Elasticidad de λ_1

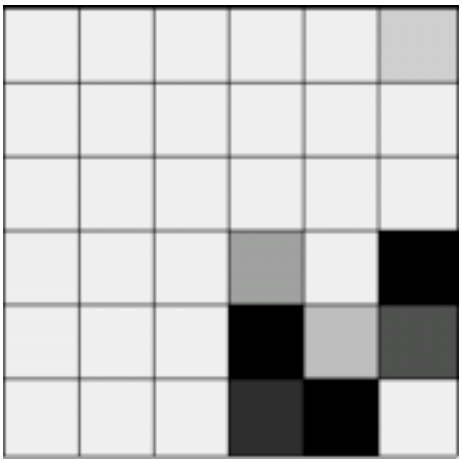

Figure 4.22: Elasticidad - Oscuro significa *mas* sensible

Las elasticidades nos reflejan que los mayores aportes a λ_1 están dados por los lugares que representan las siguientes aristas:

- a_{54}: de roseta mediana a roseta (19%)

- a_{65}: de roseta a planta floreciente (29%)
- a_{64}: de planta floreciente a roseta mediana (23%)

Listado

```
uses dialogs, sysutils, windows, matrixext, arithmetic;

global w:textwindow;
g1,g2:bargraphwindow;
s1,s2:sengraphwindow;

var i,j,t:integer;
  a:matrix [6,6] of real;
temp:matrix [6,6] of real;
autovalores:vector [6] of real;
autovalores_im:vector [6] of real;
autovector:vector [6] of real;
lautovector:vector [6] of real;
fi:vector [6] of real;
r,r2,dr, f,s:real;
b:boolean;
result:integer;

begin

 titulo(w, 'Cálculo de autovalores y autovectores');

 a^0 := [ 0.000, 0.000, 0.000, 0.000, 0.000, 322.38 ];
 a^1 := [ 0.966, 0.000, 0.000, 0.000, 0.000, 0.000  ];
 a^2 := [ 0.013, 0.010, 0.125, 0.000, 0.000, 3.448  ];
 a^3 := [ 0.007, 0.000, 0.125, 0.238, 0.000, 30.170 ];
 a^4 := [ 0.001, 0.000, 0.000, 0.245, 0.167, 0.862  ];
 a^5 := [ 0.000, 0.000, 0.000, 0.000, 0.023, 0.750  ];

 addstr(w, 'Matriz A correspondiente a una población de "teasel"??');
 mostrar(a);
 copiarmatriz(a, temp);

 result:=eig(0,6,a, autovalores, autovalores_im);

 r:=absmay(autovalores);
 r2:=absmay(autovalores_im);

 if r2>r then r:=r2;

 r2:=ln(r);
```

```
 addstr(w, 'net fertility functions:');
 copiarmatriz(temp, a);

 netf(0,6, a, fi);
 mostrar_n(fi);

 copiarmatriz(temp, a);

 addstr(w, '');

 s:=sumavec(fi);

 addstr(w, 'net rep rate: '+floattostr(s)) ;

 addstr(w, '');

 addstr(w, 'gen time: '+floattostr(ln(s)/ln(r)));

 addstr(w, 'Autovalores:');

 mostrar_n(autovalores);
 addstr(w, '');
 mostrar_n(autovalores_im);

 copiarmatriz(temp, a);
 addstr(w, 'Rate of increase:');
 addstr(w, 'r='+floattostr(r2));

result:=eigvect(0,6,r,a,autovector);

 addstr(w, '');
 addstr(w, 'autovector de '+floattostr(r));
 addstr(w, '');
 mostrar_n(autovector);

 transp(temp, a);

 result:=eigvect(0,6,r,a,lautovector);

 addstr(w, '');
 addstr(w, 'autovector izq. de '+floattostr(r));
 addstr(w, '');
 mostrar_n(lautovector);

 copiarmatriz(temp, a);

 addstr(w, '');
```

```
 addstr(w, 'escalados');
 addstr(w, '');
 escalar(autovector);
 mostrar_n(autovector);
 addstr(w, '');
 escalar(lautovector);
 mostrar_n(lautovector);

 r2 := abssegmay(autovalores);
 dr := absmay(autovalores_im);

 if dr>r2 then r2:=dr;

 dr := r / r2;

 addstr(w, '');
 addstr(w, 'Damping ratio:= '+floattostr(dr));

 for i:=0 to 5 do addrpoint(g1, autovector[i], inttostr(i), 6);
repaint(g1);

 for i:=0 to 5 do addrpoint(g2, lautovector[i], inttostr(i), 6);
repaint(g2);

inisen(s1, 6, 6);
inisen(s2, 6, 6);

 addstr(w, 'sensitivity:');

 for i:=0 to 5 do
  for j:=0 to 5 do
   begin
   s := log10(sensitivity(i,j,lautovector,autovector));
   addstr(w, floattostr(s));
   brush(s1, trunc(-2.0*s)-3);
   pintar(s1, i, j);
   end;

 addstr(w, 'elasticity:'+floattostr(a[i,j]));

 for i:=0 to 5 do
  for j:=0 to 5 do
   begin
   s:=elasticity(i,j,r,a,lautovector,autovector);
   addstr(w, floattostr(s));
   brush(s2, trunc(40000.0*s));
   pintar(s2,  i, j);
```

```
   end;
end.
```

Chapter 5

Conclusiones

El intérprete actual puede ser utilizado como una herramienta útil, aunque aún le falta mucho desarrollo por delante para considerarse estable y completo.

El presente trabajo se encuentra incompleto en varios aspectos:

- Semántica: La semántica puede ser reescrita contemplando los casos de funciones y arreglos.
- Intérprete actual: El intérprete actual es lento e ineficiente; falta trabajo de optimización. Falta un depurador (debugger) apropiado.
- Compilador: Un compilador a código ensamblador o PASCAL sería una solución alternativa a desarrollar.
- Portabilidad: un intérprete/compilador para otra plataforma, por ejemplo Linux.
- Lenguaje PML: el lenguaje puede ser extendido de muchas maneras, por ejemplo PML carece de la definición de tipos propios (*record*).
- Implementaciones de los modelos: Actualmente los análisis llevados a cabo son básicos; en [2] se pueden consultar análisis mucho más complicados para implementar en PML.

Afortunadamente gran parte de este trabajo seguirá en desarrollo; el lenguaje aquí desarrollado posiblemente se integre al paquete estadístico **InfoStat**. Por otra parte, librerías pueden ser desarrolladas independientemente del intérprete gracias a la posibilidad de llamadas a librerías externas, permitiendo continuar el desarrollo de manera paralela.

Bibliography

[1] CASWELL, H., Matrix Population Models, Sinauer, Sunderland, MA, 1989.

[2] CASWELL, H., Matrix Population Models, 2nd ed., Sinauer, Sunderland, MA, 2000.

[3] FERRIERE R., SARRAZIN F., LEGENDRE S., BARON J.P. *Matrix population models applied to viability analysis and conservation: Theory and practice with ULM software.*, Acta OEcologica , no. 17 (1996), pp. 629-656.

[4] LEAH EDELSTEIN-KESHET Mathematical Models In Biology, Birkhauser Mathematic Series, 1988.

[5] LEGENDRE, S., CLOBERT J. *ULM, a software for conservation and evolutionary biologists.*, Journal of Applied Statistics , no. 22 (1995), pp. 817-834.

[6] LESLIE P.H. *On the Use of Matrices in Certain Population Mathematics*, Biometrika , no. 33 (1945), pp. 183-212.

[7] LESLIE P.H. *Some Further Notes on the Use of Matrices in Population Mathematics*, Biometrika , no. 35 (1948), pp. 213-245.

[8] LESLIE P.H. *A Stochastic Model for Studying the Properties of Certain Biological Systems by Numerical Methods*, Biometrika , no. 45 (1958), pp. 16-31.

[9] LEVINE, J.R, MASON, T., BROWN, D. lex & yacc , O'Reilly & Associates, Inc., 1992.

[10] POLLARD J.H., Mathematical Models for the Growth of Human Populations, Cambridge Univ. Press, Cambridge, 1973.

[11] POOLE R.W., An Introduction to Quantitative Ecology, McGraw-Hill, 1974.

[12] PRATT T.W., ZELKOWITZ, M.V. Programming Languages, Design and Implementation, 3rd edition, Prentice Hall, 1999.

[13] TENNENT R.D., Semantics of Programming Languages, Prentice Hall, 1991.

[14] WELSH, J., ELDER, J. Introduction to Pascal, 3rd edition, Prentice Hall, 1988.

[15] WIRTH, N., Compiler Construction, Addison-Wesley, 1996.