SOFTWARE EDUCATIVO: CADENAS DE MARKOV EN TIEMPO DISCRETO

OLGA LUCIA GARCIA PINTO
FELIPE NIETO AVILA

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERIA INDUSTRIAL
BOGOTÁ
2004

SOFTWARE EDUCATIVO: CADENAS DE MARKOV EN TIEMPO DISCRETO

OLGA LUCIA GARCIA PINTO
FELIPE NIETO AVILA

TESIS

Director
RAFAEL GUILLERMO GARCIA CACERES
Ingeniero Industrial

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERIA INDUSTRIAL
BOGOTÁ
2004

DEDICATORIA

A mis padres y hermanos: Martha, Alba y Hugo.

Olga Lucía García Pinto

A mi familia y amigos, por darme todo su apoyo.

Felipe Nieto Avila

AGRADECIMIENTOS

A los Ingenieros: Rafael Guillermo García, Juan José Obagi, Leonardo Quintana, Cesar Bustacara y Juan Pablo Romero; a nuestros compañeros: Liliana Gómez, Gabriel Téllez, Francisco Guerrero y William Vasquez.

TABLA DE CONTENIDO

INTRODUCCIÓN

Los procesos estocásticos son una herramienta poderosa a la hora de modelar matemáticamente problemas prácticos de diferentes áreas de las organizaciones. Una herramienta que facilite y hagan aún más eficiente la labor de enseñanza de los docentes de esta asignatura es importante.

Dado el nivel de rigurosidad de la asignatura de Procesos Estocásticos; esto ha representado un alto nivel de dificultad para el estudiantado de la carrera de Ingeniería Industrial de la Universidad Javeriana, en lo que se refiere a la aprobación de la asignatura.

Gracias a la nueva reforma curricular y al nuevo sistema de créditos, se hace inminente la necesidad de utilizar herramientas pedagógicas como parte fundamental en la enseñanza de las asignaturas; fue así que se pensó en desarrollar un software educativo que apoyara la labor de los docentes y al mismo tiempo motivara el estudio de algunos de los temas de mayor relevancia de la asignatura antes mencionada, de forma didáctica.

Esta inquietud de mejorar de algún modo el proceso de enseñanza de la materia surgió a partir de la experiencia personal, como producto de una evaluación objetiva acerca de la metodología que se maneja en el desarrollo del curso y de los resultados obtenidos.

Particularmente el objeto de este trabajo es desarrollar de manera didáctica el tema de Cadenas de Markov en tiempo discreto, a través de un software educativo. La elección de este tema radica como primera medida en la relevancia

que tiene éste dentro del programa académico de la asignatura y en segunda medida en la falta de una herramienta computacional que lo apoye.

La utilización de un medio audiovisual para el desarrollo de los objetivos propuestos en este trabajo, fue consecuencia de una investigación de la que se puede llegar a afirmar que los métodos audiovisuales no solo son un apoyo material en la educación sino una parte fundamental del proceso educativo.

1 FORMULACIÓN DEL PROYECTO

1.1 DESCRIPCIÓN DEL PROBLEMA

Los estudiantes de la asignatura Procesos Estocásticos de la facultad de Ingeniería Industrial de la Pontificia Universidad Javeriana han expresado la dificultad en el aprendizaje y comprensión de los temas que contempla el programa académico de la asignatura, dentro de estos, el tema Cadenas de Markov en tiempo discreto; este hecho se manifiesta en una significativa mortalidad académica.

1.2 FORMULACIÓN DEL PROBLEMA

¿Cómo facilitar a los estudiantes de la Pontificia Universidad Javeriana los procesos de aprendizaje y comprensión del tema Cadenas de Markov en tiempo discreto, en la asignatura de Procesos Estocásticos?

1.3 JUSTIFICACIÓN

La progresiva necesidad de saber utilizar y aplicar cada día mejor, modelos probabilísticos para resolver problemas de ingeniería, respondiendo a requerimientos puntuales de las organizaciones, ha hecho que la industria informática contribuya cada vez más en la búsqueda de soluciones de software. Sin embargo, no todos los modelos cuentan con el apoyo computacional necesario para crear un ambiente favorable en donde los ingenieros puedan desarrollar el conocimiento teórico.

Analizando el programa académico de los estudiantes de Ingeniería Industrial de la Pontificia Universidad Javeriana, se ha establecido que las materias de métodos cuantitativos, tienen un porcentaje significativamente mayor de estudiantes reprobados. Además se ha establecido que la materia de procesos estocásticos no cuenta con soporte de software en un porcentaje alto de su temática, específicamente en lo que se refiere a Cadenas de Markov y Procesos Poisson.

En el estudio de los procesos industriales, se ha observado claramente que el comportamiento aleatorio es muy frecuente, así como también en otros escenarios de la vida cotidiana. Los Procesos Estocásticos tienen en su temática el modelaje matemático de una gran variedad de problemas prácticos convirtiéndose por esto en una asignatura fundamental de la formación que debe recibir un Ingeniero Industrial.

Es pertinente y justificable desarrollar un Software Educativo con aplicación en Cadenas de Markov en tiempo discreto que de manera didáctica y práctica ayude a los estudiantes a comprender el tema.

Se espera que el posible éxito de los resultados de este trabajo, motive a otros estudiantes y profesores de la Universidad Javeriana a promover estudios que generen valor agregado a las asignaturas mediante el desarrollo de herramientas computacionales de estudio propias.

1.4 OBJETIVOS

1.4.1 OBJETIVO GENERAL

Crear un software que facilite los procesos de aprendizaje y comprensión en la asignatura Procesos Estocásticos, específicamente en el tema Cadenas de Markov en Tiempo Discreto (CMTD).

1.4.2 OBJETIVOS ESPECÍFICOS

- Determinar la temática pertinente en el tema Cadenas de Markov en tiempo discreto, que debe incluir el software.
- Determinar las características pedagógicas pertinentes que debe incluir el software.
- Desarrollar el software educativo
- Desarrollar ejemplos académicos (Demos, ejercicios de aplicación) para ser incluidos en el software.
- Evaluar el software.

2 INVESTIGACIÓN

2.1. INVESTIGACIÓN DEL MERCADO

Con el fin de precisar si el objeto de este estudio representa una verdadera necesidad académica para el estudiantado de Ingeniería Industrial que cursa la asignatura Procesos Estocásticos, y en particular para determinar si se requiere un software educativo que apoye el tema: Cadenas de Markov en Tiempo Discreto, se realizó una investigación piloto en el medio estudiantil.

2.1.1 METODOLOGIA DEL ESTUDIO

Para la realización de la encuesta del estudio se consultó con un experto en el área de investigación de mercados[1], y se desarrolló la siguiente metodología:

- **Definición de los objetivos de la encuesta**
  Es primordial determinar los objetivos que persigue la investigación para realizar una encuesta que logre fundamentalmente descubrir la información deseada.

- **Diseño de la encuesta Piloto (Formulación de preguntas)**
  Para facilitar la tabulación, el análisis de los resultados y para garantizar el cumplimiento de los objetivos propuestos, la encuesta debe tener las siguientes características:

[1] ARROYO, David. Director del CIM (Centro de Investigación de Mercados) - Casa Editorial EL TIEMPO.

  - Preguntas cerradas (para responder : si o no, ó de selección múltiple)
  - Preguntas claras, que sean de fácil comprensión
  - Preguntas que no den lugar a diversas interpretaciones
  - Encuesta corta

- **Aplicación de la encuesta Piloto**
  Aplicar la encuesta a una muestra piloto de 30 estudiantes que hayan cursado la asignatura Procesos Estocásticos.

- **Análisis de resultados de la encuesta piloto**
  Tabular y hacer el análisis de los resultados.

- **Determinación del tamaño de la muestra**
  Determinar el tamaño de la muestra del estudio, teniendo en cuenta el tamaño de la población, la confiabilidad, el error y los datos requeridos de la encuesta piloto.

- **Aplicación de la encuesta definitiva**
  Aplicar la encuesta al número de estudiantes que hayan cursado la asignatura Procesos Estocásticos que correspondan al tamaño de la muestra determinado.

- **Análisis de resultados de la encuesta definitiva**
  Tabular y hacer el análisis de los resultados de la encuesta definitiva.

2.1.2 ENCUESTA PILOTO

Para la realización de la investigación piloto, se diseñó una encuesta cuyos principales **objetivos** fueron:

- Determinar si los estudiantes creen que para el proceso educativo de la asignatura Procesos estocásticos es importante el apoyo de herramientas computacionales.
- Conocer en cuales de los temas que contempla la asignatura Procesos Estocásticos, los estudiantes conocen la existencia de softwares.
- Determinar si los estudiantes consideran importante contar con un software educativo que apoye el tema Cadenas de Markov en Tiempo Discreto.
- Determinar son los contenidos que tienen mayor relevancia para los estudiantes y que debe tener el software.

La encuesta se presenta en el **Anexo A**.

2.1.2.1 ANÁLISIS DE LOS RESULTADOS

Se realizaron 30 encuestas a estudiantes que ya hubieran cursado la materia Procesos Estocásticos. Teniendo en cuenta los objetivos de la encuesta, el análisis estadístico se mostrará a partir de la pregunta número 8.

- 27 de los 30 estudiantes encuestados (90%), creen que en el proceso educativo de la asignatura de Procesos Estocásticos es importante el apoyo de herramientas computacionales.
- El 10% de los encuestados conoce la existencia de un software en el “cálculo de esperanza”.
- El 6.67% de los estudiantes encuestados conoce la existencia de un software de “Proceso Poisson”.
- Ninguno de los estudiantes encuestados conoce la existencia de un software que trabaje el tema “Cadenas de Markov en Tiempo Discreto”.
- El 6.67% de los estudiantes encuestados conoce la existencia de un software que apoye el tema “Cadenas de Markov en tiempo continuo”.

- El 26.67% de los estudiantes encuestados conoce la existencia de un software que trabaje el tema “Teoría de Colas”.
- El 66.67% de los encuestados no conoce ningún software que apoye alguno de los temas enunciados.
- 25 de estos 30 estudiantes, es decir el 83.33% de la muestra tomada, estuvieron de acuerdo en que es importante desde el punto de vista académico contar con un software educativo que apoye el desarrollo de el tema Cadenas de Markov en Tiempo Discreto.

La información expresada a continuación se obtuvo teniendo en cuenta a los 25 estudiantes que consideraron importante contar con un software educativo que apoye el tema Cadenas de Markov en Tiempo Discreto.

- Ninguno de los estudiantes (0%) conoce la existencia de algún software que trabaje específicamente el tema anteriormente mencionado.
- El 68% de estos estudiantes consideran que el software debe contener: Análisis de estados (transientes, recurrentes y absorbentes).
- El 92% considera que el software debe contener: Probabilidades estacionarias.
- El 52% considera que el software debe contener: Análisis del tipo de cadena (transiente y ergódica).
- El 68% considera que el software debe contener: Cadenas de estados transientes.
- El 72% considera que el software debe contener: Tiempos esperados en estados transientes.
- El 80% considera que el software debe contener: Probabilidades transientes.
- El 56% considera que el software debe contener: Ecuaciones de Chapman – Kolmogorov.

- El 88% considera que el software debe contener: Probabilidades de estados futuros.

El análisis gráfico se presenta en el **ANEXO B**.

Se puede entonces concluir, que es relevante realizar el estudio y desarrollar una herramienta que dé solución a esta necesidad expresada.

2.1.3 ENCUESTA DEFINITIVA

El paso a seguir luego de realizar la investigación piloto, fue determinar la muestra mínima requerida para hacer la investigación formal. El primer paso fue determinar la población de la investigación; dentro de esta se contempló a todos los estudiantes que hayan cursado la asignatura de Procesos Estocásticos desde el año 2000.

Alrededor de 840 estudiantes han cursado la materia a lo largo de los últimos 3 años y medio. El estudio se realizó con una confiabilidad del 95%, un margen de error del 10% y con probabilidad de afirmación de la presente investigación estimada del 83.33% (dato arrojado de la investigación piloto descrita en el párrafo anterior). El tamaño de la muestra para poblaciones finitas se define bajo la siguiente formula de muestreo aleatorio simple:

$$n = \frac{NZ^2PQ}{E^2N + Z^2PQ}$$

Donde:
n= Tamaño de la muestra
N= Tamaño de la población

SA Distribución de la curva normal
P= Porcentaje de aciertos (respuesta afirmativa: Si)
Q= Porcentaje de no aciertos (respuesta negativa: No)
E= Índice de error

Se tienen los siguientes datos para determinar el tamaño de la muestra:
N= 840
Z= 1.96 (Confiabilidad del 95%, usando tabla de la distribución Normal)
P= 0.8333 (83.33%)
Q= 1-P (100%-83.33%)
E= 0.1 (10%)

Haciendo el cálculo se obtiene:
n= 51.16

Teniendo en cuenta que n representa el tamaño de la muestra, es decir, el número de personas que deben ser encuestadas, se debe aproximar a mayor esta cifra, a:
n= 52

2.1.3.1 ANÁLISIS DE LOS RESULTADOS ENCUESTA DEFINITIVA

Se realizaron 53 encuestas a estudiantes que ya hubieran cursado la materia Procesos Estocásticos. Teniendo en cuenta los objetivos de la encuesta, el análisis estadístico se mostrará a partir de la pregunta número 8.

- 49 de los 53 estudiantes encuestados (92.45%), creen que en el proceso educativo de la asignatura de Procesos Estocásticos es importante el apoyo de herramientas computacionales.

- El 30.19% de los encuestados conoce la existencia de un software que apoye el “Cálculo de esperanza”.
- El 18.87% de los encuestados conoce la existencia de un software que apoye el “Proceso Poisson”.
- Ninguno de los estudiantes encuestados conoce la existencia de un software que trabaje el tema “Cadenas de Markov en Tiempo Discreto”.
- El 9.43% de los encuestados conoce la existencia de un software que apoye el tema “Cadenas de Markov en tiempo continuo”.
- El 37.74% de los encuestados conoce la existencia de un software de “Teoría de Colas”.
- El 41.51% no conoce la existencia de ningún software que apoye alguno de los temas enunciados.
- 45 de estos 53 estudiantes, es decir el 84.91% de la muestra tomada, estuvieron de acuerdo en que es importante desde el punto de vista académico contar con un software educativo que apoye el desarrollo de el tema Cadenas de Markov en Tiempo Discreto.

La información expresada a continuación se obtuvo teniendo en cuenta a los 45 estudiantes que consideraron importante contar con un software educativo que apoye el tema Cadenas de Markov en Tiempo Discreto.

- Ninguno de los estudiantes (0%) conoce la existencia de algún software que apoye específicamente el tema anteriormente mencionado.
- El 82.22% de estos estudiantes consideran que el software debe contener: Análisis de estados (transientes, recurrentes y absorbentes).
- El 53.33% considera que el software debe contener: Probabilidades estacionarias.
- El 48.89% considera que el software debe contener: Análisis del tipo de cadena (transiente y ergódica).

- El 48.89% considera que el software debe contener: Cadenas de estados transientes.
- El 62.22% considera que el software debe contener: Tiempos esperados en estados transientes.
- El 60% considera que el software debe contener: Probabilidades transientes.
- El 53.33% considera que el software debe contener: Ecuaciones de Chapman – Kolmogorov.
- El 66.67% considera que el software debe contener: Probabilidades de estados futuros.

El análisis gráfico se presenta en el **Anexo C**.

2.1.4 CONCLUSIONES DEL ANÁLISIS

- La muestra representativa de la población “Estudiantes de Ingeniería Industrial que han cursado la asignatura Procesos Estocásticos desde el año 2000” es de 52 estudiantes.
- La gran mayoría de los estudiantes encuestados (92.45%), están de acuerdo en que es importante el apoyo de herramientas computacionales en la asignatura Procesos Estocásticos.
- De acuerdo a los resultados obtenidos a través de la encuesta, se podría pensar en desarrollar los softwares que tienen menor porcentaje de conocimiento de existencia por parte de los estudiantes, es decir: proceso Poisson, Cadenas de Markov en Tiempo Continuo y Cadenas de Markov en Tiempo Discreto.
- El 84.91% de los estudiantes encuestados consideran importante contar con un software educativo que apoye el tema cadenas de Markov en tiempo discreto.

- Ninguno de los estudiantes encuestados conoce algún software que apoye el tema cadenas de Markov en tiempo discreto, lo cual robustece el propósito de este trabajo de grado.
- En términos generales, los estudiantes consideran que el software debe desarrollar los contenidos enumerados en la encuesta; el porcentaje más representativo (82.22%) que se obtuvo en esta parte de la encuesta es de los estudiantes que consideran que el software debe contener: análisis de estados (transientes, recurrentes y absorbentes).

3 SOFTWARE EDUCATIVO: CADENAS DE MARKOV EN TIEMPO DISCRETO

3.1 FUNDAMENTACIÓN TEORICA DEL SOFTWARE

El matemático polaco Henryk Greniewsky afirmó textualmente: “Siempre doy el siguiente ejemplo a mis estudiantes: tomen un mapa de África, el más sencillo, porque hay muy pocas inversiones en África, pocas líneas de ferrocarriles; y escríbanme en lenguaje corriente todas las informaciones que hay en el mapa. Eso da un gran volumen, pero no es eso todo: el volumen no servirá de nada, habrá que organizar un índice especial para buscar en él, mientras el mapa da inmediatamente informaciones, mensajes, si se prefiere. Probablemente es una cuestión de dimensiones, nuestra lengua escrita es cantoriana, unidimensional, como nuestra lengua acústica; es muy incomoda a decir verdad, mientras que las lenguas de dos dimensiones dan gran cantidad de mensajes en una superficie muy pequeña...”[2] y como concluye J.L.M. Arreguín de este ejemplo sobresalen dos afirmaciones:

1. Una sola imagen visual nos puede transmitir información que es muy difícil de describir verbalmente o por escrito.
2. La imagen visual muestra la información en dos dimensiones, mientras lo verbal-auditivo la desarrolla sucesivamente en el tiempo.

[2] GRENIEWSKY, Henrik. El concepto de información y la planificación, discusión, en el concepto de información en la ciencia contemporánea. 178 p.

A través de estas dos afirmaciones se puede visualizar el adecuado uso de los elementos visuales en la enseñanza. Puede Afirmarse que las imágenes se recuerdan en un grado muy elevado, no existe comparación posible con lo que se oye.

J.L.M. Arreguín realizó una síntesis de las características de las imágenes visuales citadas por diversos autores:

- Una imagen puede transmitir mucha información, que se capta instantáneamente, y que permita integrar una situación total con rapidez.
- Esta información tiene un carácter básico de algo actual, presente, que facilita la formación de vivencias, la adquisición de conocimientos sobre lo que se ve, pero limita en cuanto a la enunciación de ideas abstractas.
- Con la ayuda de colores y otras características formales, la imagen visual puede provocar con facilidad un impacto emotivo en el receptor.
- Las imágenes permiten visualizar muchas relaciones, integrándolas en una idea global.
- Las imágenes visuales son un recurso inigualable para representar objetos y situaciones reales.
- La limitación de las imágenes para enunciar con la exactitud y sutileza de matices que permite el lenguaje verbal, o su versión escrita, hace que para que haya comunicación precisa las imágenes visuales necesiten reforzarse mediante explicaciones verbales o escritas, ya que estas disminuyen las diversas interpretaciones (en caso de que solo se miran las imágenes).
- Las imágenes tienen la ventaja de que se pueden memorizar con gran facilidad, aunque las imágenes no se recuerdan por completo conciente.
- Cuando se refuerza lo que se dice y lo que se ve, la conciencia de lo que se recuerda de modo no consciente mejora de manera sensible.[3]

[3] Tomado de: ARREGUÍN, J.L.M. Tres Acercamientos a la Educación Audiovisual. 21 p.

Es de vital importancia que los docentes y todos aquellos que tengan relación con la educación, dejen de lado la apatía, el temor y el desconocimiento con respecto a la tecnología, de este modo no se corre el riesgo de dejarle esta responsabilidad educativa a quienes no posean el criterio para vincular y relacionar los aspectos de enseñanza-aprendizaje con las nuevas tecnologías. Los docentes tienen el deber de conocer y utilizar de la mejor manera posible diferentes medios tecnológicos de información en su docencia. "La sabiduría del maestro no puede seguir siendo la única fuente de información, el sabio maestro es el que hace una labor eficiente en la transmisión de la información."[4]

A pesar de utilizar diversos elementos pedagógicos fundamentados en el uso de imágenes, como los software educativos, los libros seguirán ocupando un lugar preponderante en los procesos de enseñanza – aprendizaje, por la capacidad explicativa y rigor académico asociado al formalismo literario.

El conocimiento en los medios tecnológicos se construye, de esta manera se induce a una relación equitativa y compartida entre el estudiante y la tecnología, que desarrolle el conocimiento y la capacidad del estudiante para interactuar con la máquina. El uso de elementos tecnológicos ayuda a llevar a cabo una nueva forma de conocimiento que no se soporta en la memoria, sino en la asociación.

Entre todos los sentidos, el de la vista es el más utilizado y desarrollado por el hombre, este se considera como el medio principal para recoger información, por lo tanto la imagen estimula la sensibilidad y hace participativo a su destinatario.

---

[4] HURTADO BONILLA, Jaime Iván. Educación, Tecnología y Conocimiento. Ediciones Unisalle. 1998. 28 p.

Según Edward T. Hall, “son muchos más los datos que llegan al sistema nervioso por los ojos, y a un ritmo mucho mayor, que por el tacto o el oído”[5]

En un método de enseñanza tradicional, lo audiovisual ayuda a garantizar que este proceso cumpla con los objetivos propuestos.

Características En El Proceso De Enseñanza[6]

En todo proceso de enseñanza el docente debe:

1. Definir las instrucciones que son necesarias para el buen desempeño del trabajo.
2. Definir un tipo de evaluación para la actividad.
3. Describir las distintas etapas de proceso, imaginar algunas dificultades posibles y buscar soluciones a éstas.
4. Preparar el instrumento educativo y/o materiales necesarios para la actividad.
5. Preparar actividades alternativas que tengan exigencia media para el estudiante.

Factores Que Favorecen El Uso De Computadores En La Educación[7]

Existen diversos factores que favorecen el uso de herramientas computacionales en la educación, dentro de los más concernientes para el propósito de este trabajo, están:

---

[5] HALL, Edward t. La dimensión Oculta. 84 p.
[6] Tomado de: HURTADO BONILLA, Jaime Iván. Educación, Tecnología y Conocimiento. Ediciones Unisalle. 1998. 57 p.
[7] Tomado de: GALVIS PANQUEVA, Álvaro H. Ingeniería de Software Educativo. 4 p.

1. **Interacción y Control sobre la Maquina:** Anteriormente la complejidad de algunos programas de computación puso barreras entre la maquina y la mayoría de sus usuarios potenciales, hoy en día es posible llevar a cabo una perfecta interacción y comunicación hombre-máquina valiéndose de lenguajes cercanos al idioma natural.
2. **Papel del público en la informática:** La creciente apertura hacia el uso del computador en la educación parece estar relacionada con el impacto de la revolución de la informática.

La decisión de desarrollar un software como medio pedagógico para dar solución al problema identificado que es objeto de este trabajo de grado, esta sustentada entonces, en la información investigada expresada anteriormente; de la cual se puede concluir que utilizar un medio audiovisual, específicamente, un software educativo para ayudar a mejorar los procesos de aprendizaje y comprensión del tema Cadenas de Markov en Tiempo Discreto es una excelente alternativa. Esta herramienta audiovisual desempeñará una importante labor complementaría en la medida en que sea utilizado como herramienta de estudio por los estudiantes de la asignatura Procesos Estocásticos.

3.2 METODOLOGIA PARA EL DESARROLLO DE UN SOFTWARE EDUCATIVO[8]

3.2.1 ANÁLISIS DE NECESIDADES EDUCATIVAS

Consulta a fuentes de información apropiadas e identificación de problemas

Se deben identificar en primera instancia los problemas existentes, sus causas y las posibles soluciones, para posteriormente determinar cuales de estas soluciones son aplicables para generar los mejores resultados.

Los alumnos y profesores son fuentes de información primaria para detectar las situaciones problemáticas, ellos saben mejor que nadie en que aspectos están fallando el contenido, modo o medios de enseñanza.

Análisis de posibles causas de los problemas detectados

Lo primordial es resolver los problemas relacionados con el aprendizaje, en los que un software educativo podría ser de gran utilidad.

- Los alumnos pueden tener carencias con respecto a bases teóricas o la motivación para estudiar los temas.
- El material de estudio puede ser defectuoso cuando la teoría es muy compleja, y/o carece de buenos ejemplos ilustrativos que sean prácticos.
- El profesor puede ser también una posible causa de fracaso, cuando su preparación es inadecuada o insuficiente para orientar la asignatura que tiene a cargo o cuando su motivación para transmitir es mínima.

[8] Tomado de: GALVIS PANQUEVA, Álvaro H. Ingeniería de Software Educativo. 63 p.

- El tiempo dedicado al estudio de un tema, la cantidad y variedad de ejercicios también pueden ser insuficientes.
- La metodología o los medios que se utilizan para apoyar el proceso de enseñanza – aprendizaje pueden ser inadecuados.

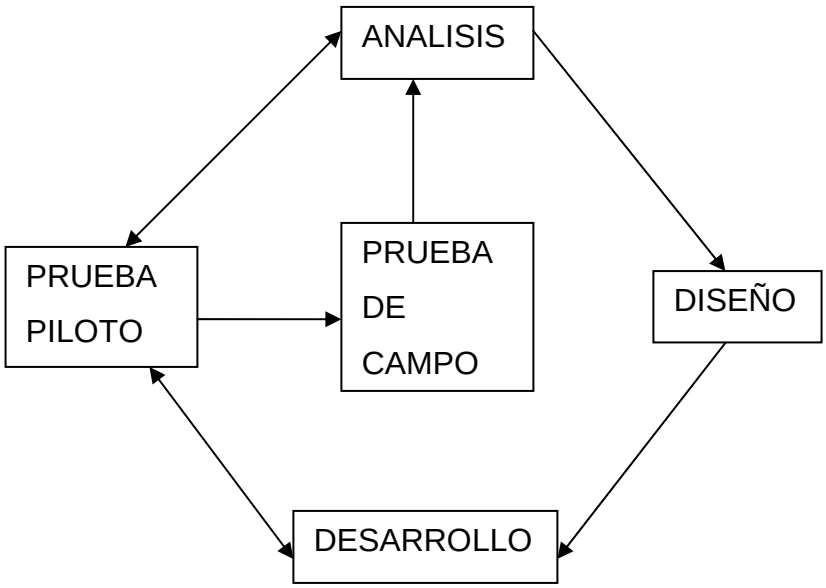
Análisis de Alternativas de solución

Dependiendo de las causas, los problemas se pueden resolver tomando decisiones administrativas como dedicar más tiempo al estudio de ciertas temáticas, conseguir los medios y materiales que faciliten los ambientes de aprendizaje apropiados y capacitar a los profesores en el uso de estos nuevos medios. En caso de que la causa del problema radique en la falta de conocimientos de base, pueden tomarse medidas como impedir que se avance en el currículo hasta no alcanzar el nivel requerido, u ofrecer instrucción remedial. Este camino administrativo es una primera alternativa que es importante considerar.

Hay causas que en cambio exigen tomar decisiones académicas. Algunas soluciones pueden ser desarrolladas por el profesor, tratando de promover un mayor trabajo sobre los materiales para aprendizaje o tratando de preparar nuevas ayudas educativas. Otras requieren mejorar los medios y materiales de enseñanza convencionales.

La herramienta computarizada debe ser un complemento más que el sustituto de una práctica. Las ayudas informáticas se considerarán siempre que no haya un mejor medio que pueda ayudar a resolver el problema.

Álvaro Galvis Panqueva plantea como metodología básica de trabajo un “Modelo sistemático para selección y desarrollo de **MECs**” *(Materiales Educativos Computarizados)*, ilustrado a continuación:



Los dos ciclos (selección o desarrollo de MECs), parten de la identificación de las necesidades educativas reales que sería conveniente satisfacer con material educativo computarizado. Cuando se tata de seleccionar un MEC se procede en el sentido contrario a las manecillas del reloj, pero cuando conviene desarrollarlo se procede en el sentido de las manecillas.

3.2.2 DISEÑO DE MECs

El contenido de un material educativo computarizado depende de las necesidades o problemas educativos identificados.

Entorno para el diseño del MEC

Es importante especificar los datos que caracterizan el entorno del MEC que se va a diseñar:

¿A quiénes se dirige? ¿Qué características tienen sus destinatarios?
¿Que área de contenido se beneficia con el estudio?
¿Qué problemas se pretende resolver?
¿Bajo que condiciones se espera que los destinatarios usen la herramienta?

Diseño educativo del MEC

Este diseño debe resolver las dudas con respecto al alcance, contenido y tratamiento que debe apoyar el material educativo computarizado.

¿Qué aprender con apoyo del MEC?
¿En qué ambiente aprenderlo?
¿Cómo motivar y mantener motivados a los usuarios?
¿Cómo saber que el aprendizaje se está logrando?

Diseño de Comunicación

La forma de comunicación entre el usuario y el programa se llama interfaz. Es importante determinar como se comunicará el usuario con el programa (interfaz de entrada), y como se comunicará el programa con el usuario (interfaz de salida).

¿Qué dispositivos de entrada y salida conviene poner a disposición del usuario para que se intercomunique con el material educativo computarizado?
¿Cómo verificar que la interfaz satisface los requisitos mínimos deseables?

Diseño Computacional

De acuerdo a las necesidades se determina qué funciones debe cumplir el Material Educativo Computarizado, como apoyo de sus usuarios, profesores y estudiantes.

El Material Educativo computarizado le puede brindar al alumno la posibilidad de controlar la secuencia, la cantidad de ejercicios, de abandonar y de reiniciar. El profesor puede tener la posibilidad de editar los ejercicios o las explicaciones.

¿Qué funciones se requiere que cumpla el Material Educativo Computarizado para cada tipo de usuario?
¿Qué estructuras de datos, en memoria principal, y en memoria secundaria, se necesitan para que funcione el Material Educativo Computarizado?

Preparación y revisión de un prototipo del MEC

La fase final del diseño es la creación del prototipo y la verificación de la utilidad frente a las necesidades para las cuales fue desarrollado el Material Educativo computarizado.

La forma más elemental de elaborar el prototipo es hacer bocetos en papel, definiendo los pantallazos. Otra forma complementaria de crear un prototipo es hacer lo mismo en el computador, con el que se define la red de pantallazos.

3.2.3 DESARROLLO DE MECS

Una vez que se tiene el diseño se desarrolla el material educativo computarizado usando las herramientas que permitan cumplir con las metas planteadas.

Desarrollo y documentación del MEC

Teniendo en cuenta la posterior necesidad de mantenimiento del material es importante desarrollar estándares de la forma como se van a denominar los procedimientos, los archivos, las constantes, las variables globales y locales, y la forma como se va a documentar cada procedimiento de que consta el programa.

La documentación a realizar es de diversa índole:

1. La documentación en un manual de usuario, debe ser fácil conocer la forma de instalación y uso del material educativo computarizado.
2. La documentación del algoritmo.

Evaluación del MEC mediante juicio de expertos

Es importante verificar si lo previsto se llevó a la práctica. Para esto se recurre a especialistas; preferiblemente a personas ajenas al proyecto para garantizar objetividad.

Estos expertos determinarán si los objetivos, contenidos y tratamientos satisfacen las necesidades que pretende solucionar el MEC.

Se selecciona la muestra, el diseño, la prueba de los instrumentos de recolección de la información y el entrenamiento de quienes van a administrar la prueba del material.

Selección de muestra y de condiciones de realización

Esta prueba se debe realizar con una muestra de expertos que cumplan con unos requisitos básicos. Estos requisitos dependen de la clase de proyecto que se esté desarrollando.

Diseño y prueba de instrumentos para recolectar información

Es importante saber la opinión y sugerencias de los expertos sobre los componentes del MEC. Para esto se pueden hacer formatos de información de retorno que sirvan para documentar las ideas que surgieron en la evaluación del material.

Toma de decisiones acerca del MEC

Dependiendo de los resultados obtenidos se pueden tomar las siguientes decisiones:

1. Desechar el MEC, si es que no resuelve los problemas que motivaron su desarrollo.
2. Ajustar detalles, adoptarlo para usarlo y evaluarlo con todos los destinatarios.
3. Hacer ajustes mayores al MEC, volviendo tan atrás como sea necesario: análisis, diseño, o nuevo desarrollo del mismo.

3.2.4 DESARROLLO DE LA METODOLOGIA "SOFTWARE EDUCATIVO: CADENAS DE MARKOV EN TIEMPO DISCRETO"

3.2.4.1 ANÁLISIS DE NECESIDADES EDUCATIVAS

Problema

Los estudiantes de la asignatura Procesos Estocásticos manifiestan dificultad en los procesos de aprendizaje y comprensión de los diferentes temas que contempla el programa académico. En este estudio en particular se pretende facilitar los anteriores procesos concernientes al tema Cadenas de Markov en Tiempo Discreto, específicamente.

Causas

Las posibles causas de la problemática afrontada por los estudiantes son:

- Tiempo insuficiente dentro del programa académico para tratar el tema.
- Falta de motivación e interés de los estudiantes.
- Falta de ejemplos demostrativos prácticos.
- Falta de ayudas educativas que apoyen este tema en particular.

Soluciones

Teniendo en cuenta las posibles causas de la problemática que se trata en este estudio, podrían identificarse las potenciales soluciones:

- Asignar al tema Cadenas de Markov en Tiempo Discreto, más tiempo dentro del programa académico para afianzar los conocimientos.

- Crear estrategias de motivación para promover el interés en los estudiantes, haciendo más práctica y manipulable la materia.
- Poner a disposición de los estudiantes una herramienta pedagógica que apoye el tema CMTD que contenga ejemplos ilustrativos y promover el uso de ésta mediante talleres. De esta manera la solución contemplaría la creación de un software educativo que facilite el aprendizaje y comprensión del tema.

Es evidente que cualquier tipo de solución para tener éxito requiere del interés de los estudiantes. El uso de un software educativo incentivará el estudio del tema; de manera didáctica ayudará al estudiante a estudiar y por ende, facilitará los procesos de aprendizaje y comprensión del tema Cadenas de Markov en Tiempo Discreto.

Para el caso particular de este trabajo de grado donde la solución indicada involucra el desarrollo del material educativo, se seguirá el modelo planteado por Galvis Panqueva en el sentido de las manecillas del reloj.

3.2.4.2 DISEÑO DEL SOFTWARE EDUCATIVO

Entorno para el diseño del Software Educativo

Especificación de los datos que caracterizan el entorno del Software Educativo que se va a diseñar:

El software educativo esta dirigido a estudiantes que cursen la asignatura procesos estocásticos, y específicamente, que estudien el tema: Cadenas de Markov en tiempo Discreto.
El área de contenido que se beneficia con el estudio es Cadenas de Markov en

tiempo Discreto.
Con este software educativo se pretende facilitar los procesos de aprendizaje y comprensión del tema Cadenas de Markov en Tiempo Discreto.
Se espera que los estudiantes usen esta herramienta durante el transcurso del tema cadenas de Markov en Tiempo Discreto, de acuerdo a lo estipulado en el programa académico (tiempo extra clase y/o en laboratorio dirigido por el profesor).

Diseño educativo del Software Educativo

Se pretende que los estudiantes aprendan el tema cadenas de Markov en Tiempo Discreto con apoyo del Software Educativo.
El ambiente propicio para aprender el tema es aquel en el cual el estudiante se sienta más cómodo, puede ser en la casa, en las instalaciones de la universidad (salas de computación o biblioteca), etc.
Con el ánimo de motivar a los estudiantes con respecto al uso del software, el profesor deberá promover las bondades de esta herramienta y en lo posible involucrarlo como parte de su explicación en la clase.
Para saber si el aprendizaje se está logrando, será indispensable analizar los resultados de las evaluaciones y de los talleres programados para la utilización del software.

Diseño de Comunicación

Conviene poner a disposición del estudiante dispositivos de entrada (ingresar datos del problema a resolver, iniciación de demostración de ejemplos prácticos de acuerdo a su elección) y dispositivos de salida (avisos de ayuda, apoyo teórico explícito a disposición del usuario, resultados).
Para verificar que la interfaz satisface los requisitos mínimos deseables, se debe

someter a la evaluación de los expertos que valorarán el software.

Diseño Computacional

Se requiere que el software cumpla funciones que desarrollen los siguientes contenidos: • Análisis de estados (transientes, recurrentes y absorbentes) • Probabilidades estacionarias • Análisis del tipo de cadena (transiente y ergódica) • Cadenas de estados transientes • Tiempos esperados en estados transientes • Probabilidades transientes • Ecuaciones de Chapman - Kolmogorov • Probabilidades de estados futuros
Las estructuras de datos, en memoria principal que se necesitan para que funcione el software educativo son ciclos (FOR, IF).

Preparación y revisión de un prototipo del Software Educativo

Se creó un prototipo en el computador en el que se define la red de pantallazos.

3.2.4.3 DESARROLLO DEL SOFTWARE EDUCATIVO

Desarrollo y documentación del Software Educativo

Se realizó el software computacional en el programa orientado a objetos: Builder C++, y que para garantizar el correcto uso de esta herramienta educativa se

realizó la documentación necesaria para soportar el uso y mantenimiento del software.

- Manual de usuario
- Algoritmo documentado

Evaluación del Software Educativo mediante juicio de expertos

Para llevar a cabo la evaluación del software educativo, se recurrió a la selección de dos clases de expertos que es importante contemplar en este trabajo de grado: ingenieros de sistemas y profesores de la facultad de ingeniería, que estén en capacidad de emitir un juicio objetivo al respecto.

Se creó un formato para recolectar la información, del tipo de una lista de chequeo, de forma que los evaluadores pudieran aprobar o desaprobar los puntos concernientes a la evaluación, fácilmente.

El formato de evaluación del software se presenta en el **ANEXO G**.

3.3 PROGRAMA DESARROLLADO EN BUILDER C++

Se eligió este programa para el desarrollo del prototipo por varias razones, entre ellas: por que es un programa orientado a objetos y por tal motivo es de fácil uso, por que es uno de los programas que un estudiante de ingeniería industrial de la universidad Javeriana esta en capacidad de entender y manejar debido a que el programa académico de las asignaturas de programación es orientado a objetos; por otro lado, el programa tiene una interfaz amigable que permite elaborar los pantallazos de un prototipo en poco tiempo.

Como ya se mencionó anteriormente Builder maneja como lenguaje base a “C” , lo cual representa un beneficio, debido a que si en algún momento se desea hacer un cambio de programa, el algoritmo sirve en otros programas que tengan como base “C”.

El código del programa desarrollado en Builder C++ se presenta en el **ANEXO D**.

3.4 DEMOS (EJERCICIOS DE APLICACIÓN)

Los demos que contiene el software educativo, pretenden mostrar de forma sencilla al usuario, el manejo del programa y a su vez el tipo de ejercicios que se pueden desarrollar con la ayuda del software. Estos elementos son de fácil manejo. El usuario solo debe elegir esta opción desde la página de bienvenida.

Los demos contenidos en el software educativo se presentan en el **ANEXO E**.

3.5 MANUAL INSTRUCTIVO DE USO

El manual instructivo de uso es un elemento fundamental del software; sirve de puente entre el programador y el usuario final, informando a este último la función de cada uno de los elementos del software, como se relacionan y el resultado que se espera obtener con cada uno de ellos.

El manual instructivo de uso se presenta en el **ANEXO F**.

3.6 EVALUACIÓN DEL SOFTWARE

Para la realización de la evaluación del software educativo por los ingenieros, se diseñó un formato cuyo principal **objetivo** es garantizar que el software tenga varias características con las cuales debe contar un material educativo computarizado.

El formato de evaluación del software se presenta en el **ANEXO G**

3.6.1 ANÁLISIS DE LOS RESULTADOS

La evaluación del software fue realizada por 3 ingenieros de sistemas y 2 profesores de la facultad de ingeniería. Teniendo en cuenta los objetivos de la evaluación, el análisis estadístico se mostrará a partir de la pregunta número 5.

- El 100% de los evaluadores afirman que el software educativo le brinda la posibilidad al estudiante de interactuar con una interfaz amigable.
- El 100% de los evaluadores afirman que el software educativo le brinda la posibilidad al estudiante de reforzar imágenes con explicaciones escritas.
- El 80% de los evaluadores afirman que el software educativo le brinda la posibilidad al estudiante de controlar la cantidad de ejercicios.
- El 100% de los evaluadores afirman que el software educativo le brinda la posibilidad al estudiante de controlar cuando abandonar.
- El 100% de los evaluadores afirman que el software educativo le brinda la posibilidad al estudiante de controlar cuando reiniciar.
- El 80% de los evaluadores afirman que el software educativo le brinda la posibilidad al profesor de editar los ejercicios.
- El 100% de los evaluadores afirman que el software educativo desarrolla el contenido: análisis de estados (transientes, recurrentes y absorbentes).

- El 100% de los evaluadores afirman que el software educativo desarrolla el contenido: probabilidades estacionarias.
- El 100% de los evaluadores afirman que el software educativo desarrolla el contenido: análisis del tipo de cadena (transiente y ergódica).
- El 100% de los evaluadores afirman que el software educativo desarrolla el contenido: cadenas de estados transientes.
- El 100% de los evaluadores afirman que el software educativo desarrolla el contenido: tiempos esperados en estados transientes.
- El 100% de los evaluadores afirman que el software educativo desarrolla el contenido: probabilidades transientes.
- El 100% de los evaluadores afirman que el software educativo desarrolla el contenido: ecuaciones de Chapman – Kolmogorov.
- El 100% de los evaluadores afirman que el software educativo desarrolla el contenido: probabilidades de estados futuros.
- El 40% de los evaluadores afirman que el manual de usuario facilita la forma de instalación.
- El 100% de los evaluadores afirman que el manual de usuario facilita el uso del software educativo.

3.6.2 CONCLUSION

En terminos generales, los evaluadores calificaron positivamente los puntos que contempla la evaluación del software educativo. Sin embargo, en lo relacionado con la forma de instalación, tan solo el 40% de los evaluadores consideró que el manual instructivo de uso facilita esta tarea, por lo cual, se realizaron las modificaciones necesarias al respecto.

4 CONCLUSIONES Y RECOMENDACIONES

La gran mayoría de los estudiantes encuestados (92.45%), estuvieron de acuerdo en que es importante el apoyo de herramientas computacionales en la asignatura Procesos Estocásticos. Del mismo modo, el 84.91% de los estudiantes encuestados consideraron importante contar con un software educativo que apoye el tema cadenas de Markov en tiempo discreto, y por otra parte, ninguno de los estudiantes encuestados conoce algún software que apoye el tema cadenas de Markov en tiempo discreto. Estos datos e información obtenidos de la investigación realizada, sustentan el desarrollo del software educativo como herramienta complementaria de estudio, dentro del programa académico de la asignatura Procesos Estocásticos.

Conforme a los buenos resultados que se obtuvieron en la evaluación del software realizada por los expertos, se decidió que este trabajo está listo para ser adoptado por los profesores y estudiantes de la asignatura procesos estocásticos en el momento en que lo prefieran.

El éxito que este proyecto pueda tener, depende de la motivación de los estudiantes con respecto al uso y al tiempo extra clase que deben dedicar para interactuar con el software. Por obvias razones, no es posible asegurar que a partir de la aparición de esta herramienta como complemento del programa académico de la asignatura procesos estocásticos, se reflejarán sus bondades en los resultados académicos del estudiantado, esto es una consecuencia que solo se conseguirá a partir de su uso concienzudo.

La versión de Builder C++ que se manejó para desarrollar el software fue la 5, sin embargo, si en algún momento se desea hacer un cambio de interfaz se recomienda que utilicen versiones más actualizadas de Builder C++ que faciliten la tarea, ya que estas tienen comandos que facilitan la actualización o cambios al prototipo.

La evaluación del software educativo, a cargo de los ingenieros, fue muy positiva. Sin embargo, en lo relativo a la forma de instalación, se realizaron modificaciones, que responden a las sugerencias manifestadas por algunos de los evaluadores. En terminos generales, los resultados reflejan la calidad del software y el trabajo invertido en su desarrollo.

El software contiene un menú de ayuda, que permite aclarar las diversas inquietudes a los estudiantes en el momento mismo de realizar la práctica, lo cual garantiza mayor aprovechamiento de este recurso.

Es recomendable que el software se empiece a utilizar inmediatamente después de ver la teoría básica del tema Cadenas de Markov en Tiempo Discreto; esto le permite al estudiante afianzar sus conocimientos al respecto, facilita los procesos de aprendizaje y comprensión y ayuda a generar inquietudes que contribuyan en la profundización del tema en mención.

5 BIBLIOGRAFÍA

ARREGUÍN, J.L.M. Tres Acercamientos a la Educación Audiovisual.

DEITEL, Harvey. C++ Cómo programar.

GALVIS PANQUEVA, Álvaro H. Ingeniería de Software Educativo. Ediciones Uniandes. 1992.

GARCIA, Alberto. Sistemas de Información, Planeamiento Estratégico y análisis. Una Guía Práctica.

GRENIEWSKY, Henrik. El concepto de información y la planificación, discusión, en el concepto de información en la ciencia contemporánea.

HALL, Edward t. La dimensión Oculta.

HURTADO Bonilla, Jaime Iván. Educación, Tecnología y Conocimiento. Ediciones Unisalle. 1998.

INSTITUTO COLOMBIANO DE NORMAS TECNICAS. Normas Técnicas Colombianas sobre documentación de Trabajos de grado. Bogotá Colombia. Quinta actualización. 2003.

JOYANES, Luis. Programación Orientada a Objetos.

KINNEAR, Thomas; Taylor, James. Investigación de Mercados.

LOHR, Sharon. Muestreo: Diseño y Análisis.

MANZANO, Vicente. Inferencia Estadística, Aplicaciones con SPSS.

RODRIGUEZ, Jacinto. Inferencia Estadística, Niveles de Precisión y Diseño Muestral.

ROSS, Sheldon. Sthochastic Processes.

6 ANEXOS

ANEXO A ENCUESTA DE INVESTIGACIÓN

SOFTWARE EDUCATIVO: CADENAS DE MARKOV EN TIEMPO DISCRETO

PROCESOS ESTOCASTICOS

Es Usted:

1. EX ALUMNO MATERIA: Si_____ No_____

Si la respuesta fue afirmativa,

2. NOMBRE:_____
3. TELEFONO:_____
4. SEXO: F_____ M_____
5. EDAD:_____
6. FECHA:_____
7. UNIVERSIDAD:_____

8. ¿Cree que en el proceso educativo de la asignatura de Procesos Estocásticos es importante el apoyo de herramientas computacionales?

   Si_____ No_____

9. ¿En cuales de los siguientes temas conoce la existencia de un software?:

   a. _____Cálculo de esperanza
   b. _____Proceso Poisson
   c. _____Cadenas de Markov en tiempo discreto
   d. _____Cadenas de Markov en tiempo continuo
   e. _____Teoría de Colas (Líneas de espera)
   f. _____Ninguno de los anteriores

10. ¿Considera usted importante desde el punto de vista académico contar con un software educativo que apoye el desarrollo del tema "Cadenas de Markov en Tiempo Discreto"?

Si_____ No_____

11. ¿Conoce algún software que apoye este tema específicamente?

Si_____ No_____

Si la respuesta fue negativa, pase a las pregunta 13

¿Cuáles?_____ Idioma_____
_____ Idioma_____

12. ¿El lenguaje de estos programas es de fácil comprensión?

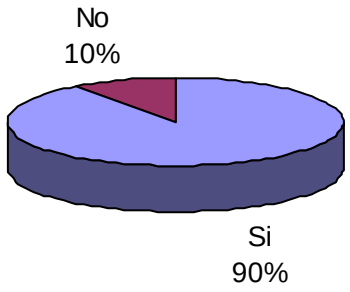
Si_____ No_____

13. En el caso del tema Cadenas de Markov en tiempo discreto, ¿qué contenidos debe tener el software?:

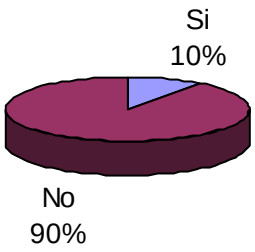
a. _____Análisis de estados (transientes, recurrentes y absorbentes)
b. _____Probabilidades estacionarias
c. _____Análisis del tipo de cadena (transiente y ergódica)
d. _____Cadenas de estados transientes
e. _____Tiempos esperados en estados transientes
f. _____Probabilidades transientes
g. _____Ecuaciones de Chapman - Kolmogorov
h. _____Probabilidades de estados futuros
i. _____Otros Cuales_____

ANEXO B ANÁLISIS GRÁFICO DE LA INVESTIGACIÓN PILOTO

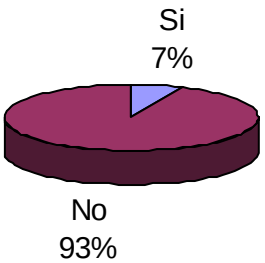
Pregunta 8. ¿Cree que en el proceso educativo de la asignatura Procesos Estocásticos es importante el apoyo de herramientas computacionales?



Pregunta 9. a) ¿Conoce la existencia de un software de "Cálculo de esperanza"?

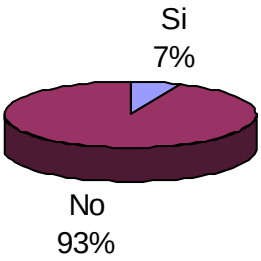


Pregunta 9. b) ¿Conoce la existencia de un software de "Proceso Poisson"?

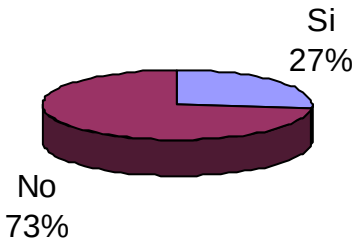


Pregunta 9. c) ¿Conoce la existencia de un software de "Cadenas de Markov en Tiempo Discreto"?

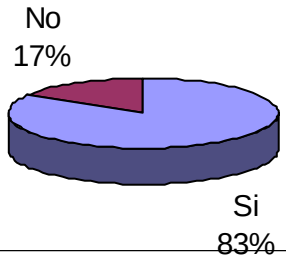
Pregunta 9. d) ¿Conoce la existencia de un software de "Cadenas de Markov en tiempo continuo"?



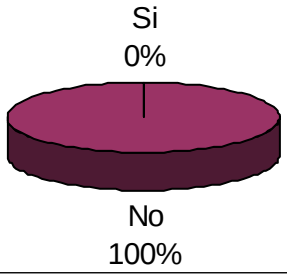
Pregunta 9. e) ¿Conoce la existencia de un software de "Teoría de Colas"?



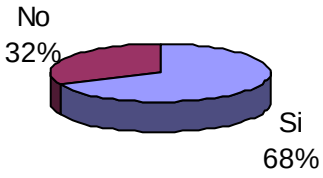
Pregunta 10. ¿Considera usted importante desde el punto de vista académico contar con un software educativo que apoye el desarrollo del tema "Cadenas de Markov en Tiempo Discreto"?



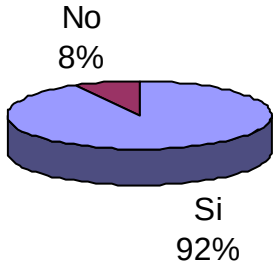
Pregunta 11. ¿Conoce la existencia de algún software que apoye específicamente el tema "Cadenas de Markov en Tiempo Discreto"?



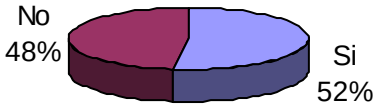
Pregunta 13. a) ¿El software educativo "Cadenas de Markov en Tiempo Discreto" debe contener "análisis de estados (transientes, recurrentes y absorbentes)"?



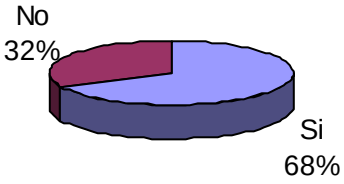
Pregunta 13. b) ¿El software educativo "Cadenas de Markov en Tiempo Discreto" debe contener "probabilidades estacionarias"?



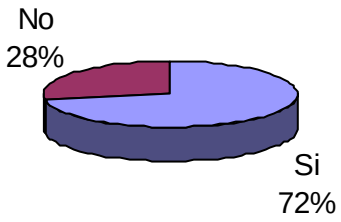
Pregunta 13. c) ¿El software educativo "Cadenas de Markov en Tiempo Discreto" debe contener "análisis del tipo de cadena (transiente y ergódica)"?



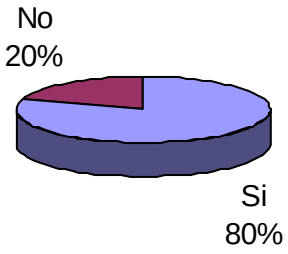
Pregunta 13. d) ¿El software educativo "Cadenas de Markov en Tiempo Discreto" debe contener "cadenas de estados trasientes"?



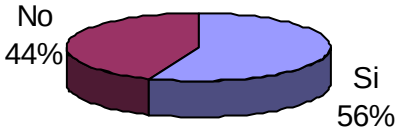
Pregunta 13. e) ¿El software educativo "cadenas de Markov en Tiempo Discreto" debe contener "tiempos esperados en estados transientes"?

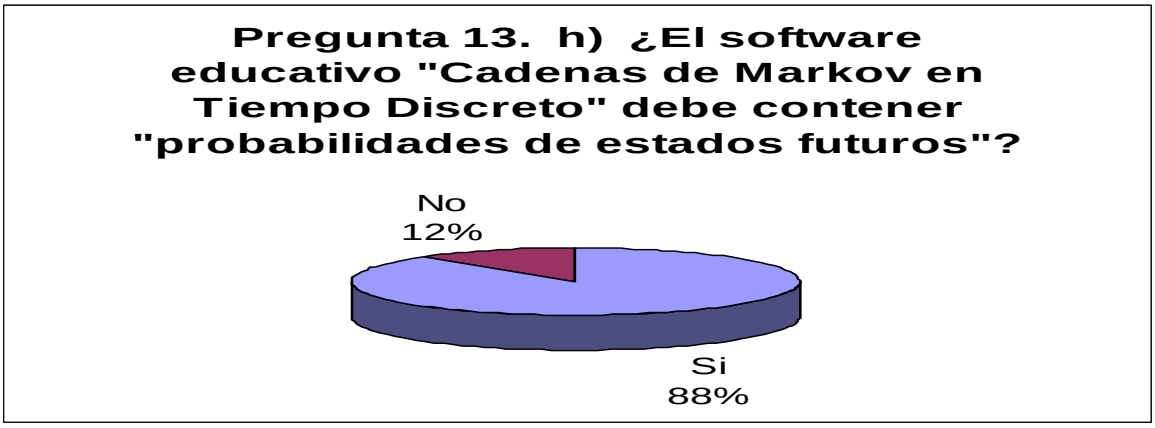


Pregunta 13. f) ¿El software educativo "Cadenas de Markov en Tiempo Discreto" debe contener "probabilidades transientes"?



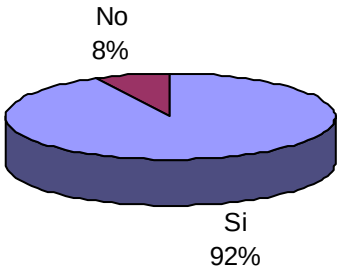
Pregunta 13. g) ¿El software educativo "Cadenas de Markov en Tiempo Discreto" debe contener "ecuaciones de Chapman - Kolmogorov"?



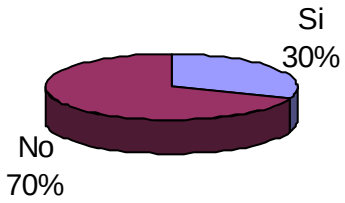

Pregunta 13. h) ¿El software educativo "Cadenas de Markov en Tiempo Discreto" debe contener "probabilidades de estados futuros"?
No
12%
Si
88%

ANEXO C ANÁLISIS GRÁFICO DE LA INVESTIGACIÓN FORMAL

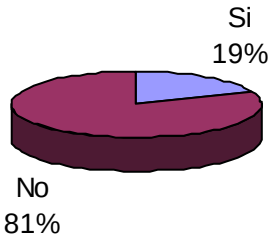
Pregunta 8. ¿Cree que en el proceso educativo de la asignatura de procesos estocásticos es importante el apoyo de herramientas computacionales?



Pregunta 9. a) ¿Conoce la existencia de un software de "cálculo de esperanza"?

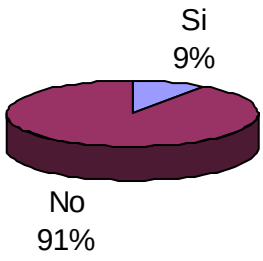


Pregunta 9. b) ¿Conoce la existencia de un software de "Proceso Poisson"?

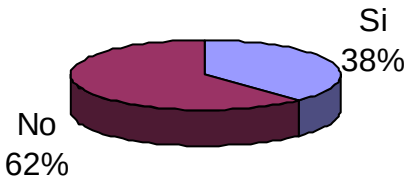


Pregunta 9. c) ¿Conoce la existencia de un software de "Cadenas de Markov en Tiempo Discreto"?

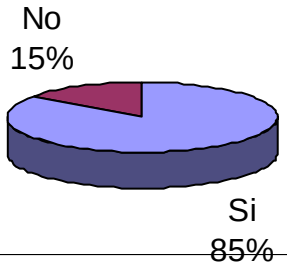
Pregunta 9. d) ¿Conoce la existencia de un software de "Cadenas de Markov en Tiempo Continuo"?



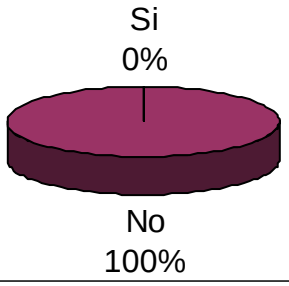
Pregunta 9. e) ¿Conoce la existencia de un software de "Teoría de Colas"?



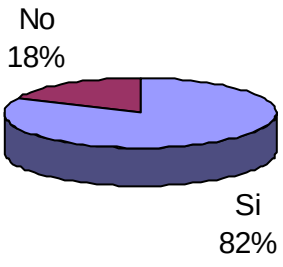
Pregunta 10. ¿Considera usted importante desde el punto de vista académico contar con un software educativo que apoye el desarrollo del tema "Cadenas de Markov en Tiempo Discreto"?



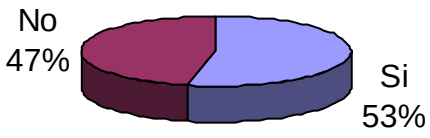
Pregunta 11. ¿Conoce la existencia de algún software que apoye específicamente el tema "Cadenas de Markov en Tiempo Discreto"?



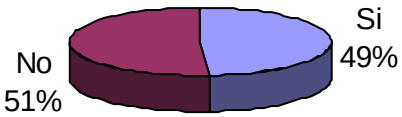
Pregunta 13. a) ¿El software educativo "Cadenas de Markov en Tiempo Discreto" debe contener "análisis de estados (transientes, recurrentes y absorbentes)"?



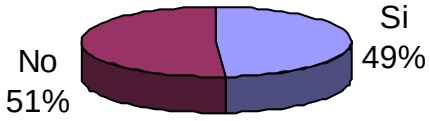
Pregunta 13. b) ¿El software educativo "Cadenas de Makov en Tiempo Discreto" debe contener "probabilidades estacionarias"?



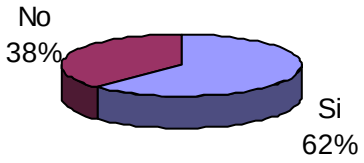
Pregunta 13. c) ¿El software educativo "Cadenas de Markov en Tiempo Discreto" debe contener "análisis del tipo de cadena (transiente y ergódica)"?



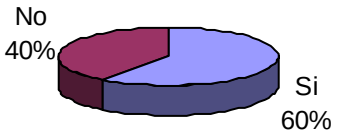
Pregunta 13. d) ¿El software educativo "Cadenas de Markov en Tiempo Discreto" debe contener "cadenas de estados transientes"?



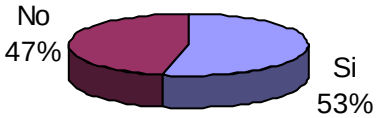
Pregunta 13. e) ¿El software educativo "Cadenas de Markov en Tiempo Discreto" debe contener "tiempos esperados en estados transientes"?



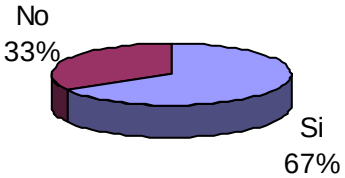
Pregunta 13. f) ¿El software educativo "Cadenas de Markov en Tiempo Discreto" debe contener "probabilidades transientes"?



Pregunta 13. g) ¿El software educativo "Cadenas de Markov en Tiempo Discreto" debe contener "ecuaciones de Chapman - Kolmogorov"?



Pregunta 13. h) ¿El software educativo "Cadenas de Markov en Tiempo Discreto" debe contener "probabilidades de estados futuros"?



ANEXO D CÓDIGO DEL PROGRAMA DESARROLLADO EN BUILDER C++

```
/*****************************************************************************/
/*                                                                           */
/*      PROGRAMA DISEÑADO PARA EL CALCULO DE LA MATRIZ DE MARKOV             */
/*                                                                           */
/*****************************************************************************/
/*       DECLARACION DE LAS UNIDADES UTILIZADAS POR EL PROGRAMA              */
/*****************************************************************************/
#include <vcl.h>
#include <math.h>
#pragma hdrstop
#include "Mat1.h"
#include "Mat2.h"
#include "Mat3.h"
#include "Unit5.h"
#include "Unit6.h"
/*****************************************************************************/
#pragma package(smart_init)
#pragma resource "*.dfm"
/*****************************************************************************/
/*      DECLARACION DE VARIABLES AUXILIARES (CONTADORES Y BANDERAS)          */
/*****************************************************************************/
TForm1 *Form1;
double res=0;
int tam ;
int p=1;
int aux=0;
int cont=0;
int auxf=1;
int conta=0;
int auxc=0;
int auxp=0;
int auxm=0;
int auxn=0;
int auxu=0;
bool finitas = false;
bool chapman = false;
bool grafos  = false;
bool grafos_t = false;
```

```
bool transicion = false;
bool ecuacion   = false;
/*****************************************************************************/
/*                INICIALIZACION DE LA FORMA PRINCIPAL                       */
/*****************************************************************************/
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
/*****************************************************************************/
/*                  PROCEDIMIENTO PARA CREAR LA MATRIZ                       */
/*****************************************************************************/
void __fastcall TForm1::BitBtn1Click(TObject *Sender)
{
 if (Edit1->Text != ""){
BitBtn4->Enabled = true;
BitBtn1->Caption = "Crear";

tam = StrToInt(Edit1->Text) + 1;

if (tam>2)
{
  StringGrid1->ColCount=tam;
  StringGrid1->RowCount=tam;
  for (int i=0; i<=tam; i++)
  {
    StringGrid1->Cells[i][0] = IntToStr(i);
  }
  for (int j=0; j<= tam; j++)
    StringGrid1->Cells[0][j] = IntToStr(j);
  for (int i=1;i<=tam;i++)
    for (int j=1;j<=tam;j++)
      StringGrid1->Cells[i][j] = IntToStr(0);
  for (int i=1;i<=tam-1;i++)
  {
    ComboBox1->Items->Add(IntToStr(i));
    ComboBox2->Items->Add(IntToStr(i));
  }
}
else if (tam<=2)
{
  ShowMessage(AnsiString("El tamaño de su matriz debe ser > 1"));
```

```
  Edit1->Text="";
}
GroupBox2->Visible=true;
Edit4->Text="1";
Edit5->Text="1";
}
else
    ShowMessage("No ha digitado el tamaño de la matriz!");

BitBtn1->Default = false ;
Edit2->SetFocus() ;
}
/*****************************************************************************/
/*                MUESTRA EL MENU DE EDICION DE LA MATRIZ                    */
/*****************************************************************************/
void __fastcall TForm1::BitBtn2Click(TObject *Sender)
{
GroupBox3->Visible=true;
}
/*****************************************************************************/
/*          PROCEDIMIENTO PARA ASIGNMAR LOS DATOS A LA MATRIZ                */
/*****************************************************************************/
void __fastcall TForm1::BitBtn4Click(TObject *Sender)
{
    double res2,dif ;
    float temp ;

  res2 = StrToFloat(Edit2->Text);
  if  (res2 < 0 || res2 > 1) {
     ShowMessage(AnsiString("Valor no permitido!!") );
     return ;
  }
  res=0;
  for (int i=1;i<=tam;i++){
        res = res + StrToFloat(StringGrid1->Cells[i][StrToInt(Edit4->Text)]);
  }

   res = res + StrToFloat(Edit2->Text);
   dif = res * 1;
  if (dif < 1) {
      if (dif >= 1 ) goto salto ;
```

```
        StringGrid1->Cells[StrToInt(Edit5->Text)][StrToInt(Edit4->Text)] = FloatToStr(StrToFloat(Edit2->Text));
        Edit2->Text="0";

        if (p<=tam-2){
                Edit5->Text=IntToStr(StrToInt(Edit5->Text)+1);
                p++;
        }
        else if(p>tam-2){
                if (res==1){
                        p=1;
                        Edit5->Text="1";
                        Edit4->Text=IntToStr(StrToInt(Edit4->Text)+1);
                }
                else if (res!=1){
                        ShowMessage(AnsiString("Edite por favor esta fila porque la suma de la misma es diferente de
1!!"));
                        return ;
                }
               }
  }
  else if (res==1){
        salto:
        StringGrid1->Cells[StrToInt(Edit5->Text)][StrToInt(Edit4->Text)] = FloatToStr(StrToFloat(Edit2->Text));
        Edit2->Text="0";
        res=0;
        p=1;
        Edit5->Text="1";
        Edit4->Text=IntToStr(StrToInt(Edit4->Text)+1);
  }
  else if (res>1){
  ShowMessage(AnsiString("El valor de la fila no puede ser superior a 1, por favor ingrese otro valor!"));
  res=0;
  Edit2->Text="0";
  return ;
  }
  if (Edit4->Text >= tam || Edit5->Text >= tam)
  {
        BitBtn4->Enabled=false;
        Resultados1->Enabled = true ;
  }

  if (StrToInt(Edit5->Text) == (tam - 1)) {
```

```
        dif = 1 - res ;
        Edit2->Text = FloatToStr(dif ) ;
        }

}
/*****************************************************************************/
/*  PROCEDIMIENTO PARA ASIGNAR LOS VALORES CORREGIDOS DENTRO DE LA MATRIZ  */
/*****************************************************************************/
void __fastcall TForm1::BitBtn3Click(TObject *Sender)
{

        StringGrid1->Cells[StrToInt(ComboBox2->Text)][StrToInt(ComboBox1->Text)]                                =
FloatToStr(StrToFloat(Edit3->Text));
        Edit3->Text="0";
        res=0;
        if (StrToInt(ComboBox2->Text)<=tam-2){
                Edit5->Text=IntToStr(StrToInt(ComboBox2->Text)+1);
                Edit4->Text=ComboBox1->Text;
                p++;
        }
        else if(StrToInt(ComboBox2->Text)>tam-2){
                p=1;
                Edit5->Text="1";
                Edit4->Text=IntToStr(StrToInt(ComboBox1->Text)+1);
        }
        BitBtn4->Enabled=true;
GroupBox3->Visible=false;
}
/*****************************************************************************/
/*      PROCEDIMIETO PARA INGRESAR A LA OPCION DE MATRIZ TRANSIENTE        */
/*****************************************************************************/
void __fastcall TForm1::crearresultados1Click(TObject *Sender)
{
  int entro = 0;
  for (int i=1; i<=tam; i++)
    for (int j=1; j<=tam; j++)
      if (((j==i) && (StringGrid1->Cells[i][j]=="1"))  &&
        ((j==i) && (StringGrid1->Cells[i+1][j+1] == "1")))
          auxf++;

  StringGrid2->RowCount =  tam;
  StringGrid2->ColCount =  tam;
```

```
for (int i=1;i<=tam;i++)
    for (int j=1;j<=tam;j++)
      StringGrid2->Cells[i][j] = StringGrid1->Cells[i][j];

for(int v=1; v<=auxf; v++)
   for (int i=1; i<=tam; i++)
      for (int j=1; j<=tam; j++)
      {
        if ((j==i) && (StringGrid2->Cells[i][j] == "1"))
        {
          entro = 1;
          for (int l=i;l<=tam;l++)
            for (int k=1;k<=tam;k++)
              StringGrid2->Cells[l][k] = StringGrid2->Cells[l+1][k];
          for (int l=j;l<=tam;l++)
            for (int k=1;k<=tam;k++)
              StringGrid2->Cells[k][l] = StringGrid2->Cells[k][l+1];
          cont++;
        }
        else
          StringGrid2->Cells[i][j] = StringGrid2->Cells[i][j];
      }

StringGrid2->ColCount = tam-cont;
StringGrid2->RowCount = tam-cont;
Form2->StringGrid1->ColCount = tam-cont; //Transiente
Form2->StringGrid1->RowCount = tam-cont; //Transiente
Form2->StringGrid2->ColCount = tam-cont; //Identidad
Form2->StringGrid2->RowCount = tam-cont; //Identidad

for (int i=0;i<=(tam-cont);i++)
{
  for (int j=0;j<=(tam-cont);j++)
  {
    Form2->StringGrid1->Cells[i][j]=StringGrid2->Cells[i][j];
      if ((j!=i)&&((j!=0)&&(i!=0)))
        Form2->StringGrid2->Cells[i][j]="0";
      else if((i==j)&&((j!=0)&&(i!=0)))
        Form2->StringGrid2->Cells[i][j]="1";
      else if ((j==0)||(i==0))
        Form2->StringGrid2->Cells[i][j]=StringGrid2->Cells[i][j];
```

```
   }
  }

  if (entro == 1)
  {
    Form2->Visible=true;
  }
  else
  {
     ShowMessage(AnsiString("No es una Matriz Transiente. Por favor Verifique la matriz, si desea más información por favor remitase a la ayuda"));
  }
  cont=0;
  auxf=1;
}
/*****************************************************************************/
/*      PROCEDIMIETO PARA INGRESAR A LA OPCION DE MATRIZ ERGODICA          */
/*****************************************************************************//*************************************************************************/
/*                    OPCION DE SALIR DEL PROGRAMA                           */
/*****************************************************************************/
void __fastcall TForm1::Salir1Click(TObject *Sender)
{
 Close ();
}
/*****************************************************************************/
/*         PROCEDIMIENTO PARA INGRESAR A LA OPCION DE GRAFOS                 */
/*****************************************************************************/
void __fastcall TForm1::CrearGrafos1Click(TObject *Sender)
{
 Panel1->Visible = true;
 Image1->Canvas->AfterConstruction();
 Image1->Canvas->Refresh();
 Image1->Refresh();
}
/*****************************************************************************/
/*         PROCEDIMIENTO PARA GRAFICAR LA MATRIZ EN EL GRAFO                 */
/*****************************************************************************/
/*  Este procedimiento funciona con la propiedad canvas contenida en la
    libreria standard de builder, el algoritmo esta diseñado para crear
    la cantidad de nodos digitados (no mayor a 10!), el evalua los datos
    con las respectivas sentencias if y se ejecutara en un ciclo for hasta
```

```
    que la matriz sea recorrida completamente.                          */
/*************************************************************************/
void __fastcall TForm1::Button1Click(TObject *Sender)
{
 Image1->Canvas->AfterConstruction();
 Image1->Visible = true;
 Image1->Canvas->Refresh();
 Image1->Refresh();
 Image1->Show();
 for (int y =1; y < tam; y++)
 {
  int x=1;

  Image1->Canvas->Font->Color = clRed ;
  Image1->Canvas->Font->Style = TFontStyles()<< fsBold ;
  Image1->Canvas->TextOutA (y * 40, y * 40, IntToStr(y));
  Image1->Canvas->Font->Color = clBlack;
  Image1->Canvas->Font->Style = TFontStyles() ;
    for (x; x < tam; x++)
  {
   if (StringGrid1->Cells[x][y] != NULL)
   {
    if (y < x)
    {
      if (y == 1)
      {
       Image1->Canvas->Rectangle(48,44,x*40,45);
       Image1->Canvas->Rectangle(x*40,45,x*40+1,x*40);
       Image1->Canvas->TextOutA(x*40+6,49,StringGrid1->Cells[x][y]);
      }
      if (y == 2)
      {
       Image1->Canvas->Rectangle(88,84,x*40,85);
       Image1->Canvas->Rectangle(x*40,85,x*40+1,x*40);
       Image1->Canvas->TextOutA(x*40+6,89,StringGrid1->Cells[x][y]);
      }
      if (y == 3)
      {
       Image1->Canvas->Rectangle(128,124,x*40,125);
       Image1->Canvas->Rectangle(x*40,125,x*40+1,x*40);
       Image1->Canvas->TextOutA(x*40+6,129,StringGrid1->Cells[x][y]);
      }
```

```
if (y == 4)
{
 Image1->Canvas->Rectangle(168,164,x*40,165);
 Image1->Canvas->Rectangle(x*40,165,x*40+1,x*40);
 Image1->Canvas->TextOutA(x*40+6,169,StringGrid1->Cells[x][y]);
}
if (y == 5)
{
 Image1->Canvas->Rectangle(208,204,x*40,205);
 Image1->Canvas->Rectangle(x*40,205,x*40+1,x*40);
 Image1->Canvas->TextOutA(x*40+6,209,StringGrid1->Cells[x][y]);
}
if (y == 6)
{
 Image1->Canvas->Rectangle(248,244,x*40,245);
 Image1->Canvas->Rectangle(x*40,245,x*40+1,x*40);
 Image1->Canvas->TextOutA(x*40+6,249,StringGrid1->Cells[x][y]);
}
if (y == 7)
{
 Image1->Canvas->Rectangle(288,284,x*40,285);
 Image1->Canvas->Rectangle(x*40,285,x*40+1,x*40);
 Image1->Canvas->TextOutA(x*40+6,289,StringGrid1->Cells[x][y]);
}
if (y == 8)
{
 Image1->Canvas->Rectangle(328,324,x*40,325);
 Image1->Canvas->Rectangle(x*40,325,x*40+1,x*40);
 Image1->Canvas->TextOutA(x*40+6,329,StringGrid1->Cells[x][y]);
}
if (y == 9)
{
 Image1->Canvas->Rectangle(368,364,x*40,365);
 Image1->Canvas->Rectangle(x*40,365,x*40+1,x*40);
 Image1->Canvas->TextOutA(x*40+6,369,StringGrid1->Cells[x][y]);
}
if (y == 10)
{
 Image1->Canvas->Rectangle(408,404,x*40,405);
 Image1->Canvas->Rectangle(x*40,405,x*40+1,x*40);
 Image1->Canvas->TextOutA(x*40+6,409,StringGrid1->Cells[x][y]);
}
```

```
  }
  if (y > x)
  {
    if (y == 2)
    {
     Image1->Canvas->Rectangle(45,84,76,85);
     Image1->Canvas->Rectangle(43,84,44,52);
     Image1->Canvas->TextOutA(64,94,StringGrid1->Cells[x][y]);
    }
    if (y == 3)
    {
     Image1->Canvas->Rectangle(116, 125, x*40+4, 124);
     Image1->Canvas->Rectangle(x*40+4, 124, x*40-1+4, x*40+15);
     Image1->Canvas->TextOutA(116-40*x, 125,StringGrid1->Cells[x][y]);
    }
    if (y == 4)
    {
     Image1->Canvas->Rectangle(156, 165, x*40+4, 164);
     Image1->Canvas->Rectangle(x*40+4, 164, x*40-1+4, x*40+15);
     Image1->Canvas->TextOutA(156-40*x, 165,StringGrid1->Cells[x][y]);
    }
    if (y == 5)
    {
     Image1->Canvas->Rectangle(196, 205, x*40+4, 204);
     Image1->Canvas->Rectangle(x*40+4, 204, x*40-1+4, x*40+15);
     Image1->Canvas->TextOutA(196-40*x, 205,StringGrid1->Cells[x][y]);
    }
    if (y == 6)
    {
     Image1->Canvas->Rectangle(236, 245, x*40+4, 244);
     Image1->Canvas->Rectangle(x*40+4, 244, x*40-1+4, x*40+15);
     Image1->Canvas->TextOutA(236-40*x, 245,StringGrid1->Cells[x][y]);
    }
    if (y == 7)
    {
     Image1->Canvas->Rectangle(276, 285, x*40+4, 284);
     Image1->Canvas->Rectangle(x*40+4, 284, x*40-1+4, x*40+15);
     Image1->Canvas->TextOutA(276-40*x, 285,StringGrid1->Cells[x][y]);
    }
    if (y == 8)
    {
     Image1->Canvas->Rectangle(316, 325, x*40+4, 324);
```

```
      Image1->Canvas->Rectangle(x*40+4, 324, x*40-1+4, x*40+15);
      Image1->Canvas->TextOutA(316-40*x, 325,StringGrid1->Cells[x][y]);
     }
     if (y == 9)
     {
      Image1->Canvas->Rectangle(356, 365, x*40+4, 364);
      Image1->Canvas->Rectangle(356-x*40, 364, 356-x*40-1, x*40+15);
      Image1->Canvas->TextOutA(356-40*x, 365,StringGrid1->Cells[x][y]);
     }
     if (y == 10)
     {
      Image1->Canvas->Rectangle(396, 405, 396-x*40, 404);
      Image1->Canvas->Rectangle(x*40+4, 404, x*40-1+4, x*40+15);
      Image1->Canvas->TextOutA(396-x*40, 405,StringGrid1->Cells[x][y]);
     }
    }
    if (y==x)
    {
     Image1->Canvas->Font->Color = clRed ;
     Image1->Canvas->TextOutA(y*40+8, y*40-4, "<--" );
     Image1->Canvas->Font->Color = clBlack ;
     Image1->Canvas->TextOutA(y*40+22, y*40-18, StringGrid1->Cells[x][y]);
     for (int k = 1; k <=15; k++)
     {
      Image1->Canvas->TextOutA (y*40+20, y*40-k+6,"|");
      Image1->Canvas->TextOutA (y*40+2, y*40-20, "___");
      if (k != 10)
        Image1->Canvas->TextOutA (y*40+2, y*40-25+k, "|");
     }
    }
     }
    }
   }
  }
/*****************************************************************************/
/*              OPCION PARA REGRESAR AL MENU PRINCIPAL                       */
/*****************************************************************************/
void __fastcall TForm1::Button2Click(TObject *Sender)
{
 Panel1->Visible = false;
 Image1->Refresh();
 Image1->Canvas->Refresh();
```

```
 Image1->Dragging();
 Image1->Hide();
 Image1->Canvas->CleanupInstance();
 Image1->CleanupInstance();
 Image1->Canvas->BeforeDestruction();
 Image1->Update();
}
/*****************************************************************************/
/*                        OPCION DE AYUDA                                    */
/*****************************************************************************/
/*       se activa elpanel donde se desplegara la ayuda del sistema       */
/*****************************************************************************/
void __fastcall TForm1::Acercade1Click(TObject *Sender)
{
 Panel2->Visible = true;
}
/*****************************************************************************/
/*                OPCION DE REGRESAR AL MENU PRINCIPAL                 */
/*****************************************************************************/
void __fastcall TForm1::Label7Click(TObject *Sender)
{
 Panel2->Visible = false;
 p_grafico->Visible = false;
 ver->Enabled = false;
 Memo1->Clear();
}
/*****************************************************************************/
/*     OPCION PARA CARGAR LOS DATOS DE INTRODUCCION DENTRO DE LA AYUDA     */
/*****************************************************************************/
void __fastcall TForm1::Label8Click(TObject *Sender)
{
 Memo1->Lines->LoadFromFile("marco teorico.txt");
 p_grafico->Visible = false;
 ver->Enabled = false;
 finitas = false;
 chapman = false;
 grafos  = false;
 grafos_t = false;
 transicion = false;
 ecuacion = false;
}
/*****************************************************************************/
```

```
/*  OPCION PARA CARGAR LOS DATOS DE CADENAS DE MARKOV DENTRO DE LA AYUDA   */
/*************************************************************************/
void __fastcall TForm1::Label9Click(TObject *Sender)
{
 Memo1->Lines->LoadFromFile("cadenas markov.txt");
 p_grafico->Visible = false;
 ver->Enabled = false;
 finitas = false;
 chapman = false;
 grafos  = false;
 grafos_t = false;
 transicion = false;
 ecuacion = false;
}
/*************************************************************************/
/*   OPCION PARA CARGAR LOS DATOS DE CADENAS FINITAS DENTRO DE LA AYUDA    */
/*************************************************************************/
void __fastcall TForm1::Label10Click(TObject *Sender)
{
 Memo1->Lines->LoadFromFile("finitas.txt");
 p_grafico->Visible = false;
 ver->Enabled = true;
 finitas = true;
 chapman = false;
 grafos  = false;
 grafos_t = false;
 transicion = false;
 ecuacion = false;
}
/*************************************************************************/
/* OPCION PARA CARGAR LOS DATOS DE FUNCIONES DE CHAPMAN DENTRO DE LA AYUDA */
/*************************************************************************/
void __fastcall TForm1::Label11Click(TObject *Sender)
{
 Memo1->Lines->LoadFromFile("chapman.txt");
 p_grafico->Visible = false;
 ver->Enabled = true;
 chapman = true;
 finitas = false;
 grafos  = false;
 grafos_t = false;
 transicion = false;
```

```
 ecuacion = false;
}
/*****************************************************************************/
/*                    CARGAR DESCRIPCION DE ESTADOS                          */
/*****************************************************************************/
void __fastcall TForm1::Label13Click(TObject *Sender)
{
 Memo1->Lines->LoadFromFile("estado.txt");
 p_grafico->Visible = false;
 ver->Enabled = false;
 finitas = false;
 chapman = false;
 grafos  = false;
 grafos_t = false;
 transicion = false;
 ecuacion = false;
}
/*****************************************************************************/
/*            CARGAR DESCRIPCION DE ESTADOS DE COMUNICACION                  */
/*****************************************************************************/
void __fastcall TForm1::Label14Click(TObject *Sender)
{
 Memo1->Lines->LoadFromFile("comunicacion.txt");
 p_grafico->Visible = false;
 ver->Enabled = false;
 finitas = false;
 chapman = false;
 grafos  = false;
 grafos_t = false;
 transicion = false;
 ecuacion = false;
}
/*****************************************************************************/
/*            CARGAR DESCRIPCION DE ESTADOS DE CLASE                         */
/*****************************************************************************/
void __fastcall TForm1::Label15Click(TObject *Sender)
{
 Memo1->Lines->LoadFromFile("clase.txt");
 p_grafico->Visible = false;
 ver->Enabled = false;
 finitas = false;
 chapman = false;
```

```
 grafos  = false;
 grafos_t = false;
 transicion = false;
 ecuacion = false;
}
/*****************************************************************************/
/*              CARGAR DESCRIPCION DE ESTADOS  RECURRENTES                   */
/*****************************************************************************/
void __fastcall TForm1::Label16Click(TObject *Sender)
{
 Memo1->Lines->LoadFromFile("estado recurrente.txt");
 p_grafico->Visible = false;
 ver->Enabled = false;
 finitas = false;
 chapman = false;
 grafos  = false;
 grafos_t = false;
 transicion = false;
 ecuacion = false;
}
/*****************************************************************************/
/*              CARGAR DESCRIPCION DE ESTADOS TRANSIENTES                    */
/*****************************************************************************/
void __fastcall TForm1::Label17Click(TObject *Sender)
{
 Memo1->Lines->LoadFromFile("estado transiente.txt");
 p_grafico->Visible = false;
 ver->Enabled = false;
 finitas = false;
 chapman = false;
 grafos  = false;
 grafos_t = false;
 transicion = false;
 ecuacion = false;
}
/*****************************************************************************/
/*              CARGAR DESCRIPCION DE ESTADOS ABSORBENTES                    */
/*****************************************************************************/
void __fastcall TForm1::Label18Click(TObject *Sender)
{
 Memo1->Lines->LoadFromFile("estado absorbente.txt");
 p_grafico->Visible = false;
```

```
 ver->Enabled = false;
 finitas = false;
 chapman = false;
 grafos  = false;
 grafos_t = false;
 transicion = false;
 ecuacion = false;
}
/*****************************************************************************/
/*            CARGAR DESCRIPCION DE ESTADOS IRREDUCTIBLES                    */
/*****************************************************************************/
void __fastcall TForm1::Label20Click(TObject *Sender)
{
 Memo1->Lines->LoadFromFile("irreductible.txt");
 p_grafico->Visible = false;
 ver->Enabled = false;
 finitas = false;
 chapman = false;
 grafos  = false;
 grafos_t = false;
 transicion = false;
 ecuacion = false;
}
/*****************************************************************************/
/*            CARGAR DESCRIPCION DE ESTADOS TRANSIENTES                      */
/*****************************************************************************/
void __fastcall TForm1::Label21Click(TObject *Sender)
{
 Memo1->Lines->LoadFromFile("transiente.txt");
 p_grafico->Visible = false;
 ver->Enabled = false;
 finitas = false;
 chapman = false;
 grafos  = false;
 grafos_t = false;
 transicion = false;
 ecuacion = false;
}
/*****************************************************************************/
/*            CARGAR DESCRIPCION DE MATRICES PERIODICAS                      */
/*****************************************************************************/
void __fastcall TForm1::Label22Click(TObject *Sender)
```

```
{
 Memo1->Lines->LoadFromFile("periodica.txt");
 p_grafico->Visible = false;
 ver->Enabled = false;
 finitas = false;
 chapman = false;
 grafos  = false;
 grafos_t = false;
 transicion = false;
 ecuacion = false;
}
/*****************************************************************************/
/*            CARGAR DESCRIPCION DE MATRICES APERIODICAS                     */
/*****************************************************************************/
void __fastcall TForm1::Label23Click(TObject *Sender)
{
 Memo1->Lines->LoadFromFile("aperiodica.txt");
 p_grafico->Visible = false;
 ver->Enabled = false;
 finitas = false;
 chapman = false;
 grafos  = false;
 grafos_t = false;
 transicion = false;
 ecuacion = false;
}
/*****************************************************************************/
/*            CARGAR DESCRIPCION DE MATRICES ERGODICAS                       */
/*****************************************************************************/
void __fastcall TForm1::Label24Click(TObject *Sender)
{
 Memo1->Lines->LoadFromFile("ergodica.txt");
 p_grafico->Visible = false;
 ver->Enabled = false;
 finitas = false;
 chapman = false;
 grafos  = false;
 grafos_t = false;
 transicion = false;
 ecuacion = false;
}
/*****************************************************************************/
```

```
/* PROCEDIMIENTO PARA CARGAR LAS RESPECTIVAS IMAGENES DE LAS DESCRIPCIONES */
/*************************************************************************/
/*cada descripcion que contenga imagen activara su respectiva bandera para
  que el sistema carge la imagen correspondiente                         */
/*************************************************************************/
void __fastcall TForm1::verClick(TObject *Sender)
{
 p_grafico->Visible = true;
 if (finitas)
    Image2->Picture->LoadFromFile("finitas.JPG");
 if (chapman)
    Image2->Picture->LoadFromFile("chapman.JPG");
 if (grafos)
    Image2->Picture->LoadFromFile("grafos.JPG");
 if (grafos_t)
    Image2->Picture->LoadFromFile("grafos trancientes.JPG");
 if (transicion)
    Image2->Picture->LoadFromFile("transicion.JPG");
 if (ecuacion)
    Image2->Picture->LoadFromFile("ecuacion.JPG");
}
/*************************************************************************/
/*                  CARGAR DESCRIPCION DE GRAFOS                         */
/*************************************************************************/
void __fastcall TForm1::Label25Click(TObject *Sender)
{
 Memo1->Lines->LoadFromFile("grafos.txt");
 p_grafico->Visible = false;
 ver->Enabled = true;
 grafos = true;
 chapman = false;
 finitas = false;
 grafos_t = false;
 transicion = false;
 Image2->Picture->LoadFromFile("transicion.JPG");
}
/*************************************************************************/
/*            CARGAR DESCRIPCION DE GRAFOS TRANSIENTES                   */
/*************************************************************************/
void __fastcall TForm1::Label26Click(TObject *Sender)
{
 Memo1->Lines->LoadFromFile("grafos trancientes.txt");
```

```
 p_grafico->Visible = false;
 ver->Enabled = true;
 grafos_t = true;
 chapman = false;
 finitas = false;
 grafos  = false;
 transicion = false;
 ecuacion = false;
}
/*****************************************************************************/
/*            CARGAR DESCRIPCION DE TRANSICIONES                             */
/*****************************************************************************/
void __fastcall TForm1::Label27Click(TObject *Sender)
{
 Memo1->Lines->LoadFromFile("transicion.txt");
 p_grafico->Visible = false;
 ver->Enabled = true;
 transicion = true;
 chapman = false;
 finitas = false;
 grafos  = false;
 grafos_t = false;
 ecuacion = false;
}
/*****************************************************************************/
/*            CARGAR DESCRIPCION DE ECUACIONES                               */
/*****************************************************************************/
void __fastcall TForm1::Label28Click(TObject *Sender)
{
 Memo1->Lines->LoadFromFile("ecuacion.txt");
 p_grafico->Visible = false;
 ver->Enabled = true;
 ecuacion = true;
 chapman = false;
 finitas = false;
 grafos  = false;
 grafos_t = false;
 transicion = false;
}
/*****************************************************************************/
/*                      EFECTO BOTON DE DEMOS                                */
/*****************************************************************************/
```

```
void __fastcall TForm1::Image4MouseMove(TObject *Sender, TShiftState Shift,
     int X, int Y)
{
 Image4->Picture->LoadFromFile("b-demos-on.JPG");
}
/*****************************************************************************/
/*              CANCELAR EFECTO BOTON DEMOS E INGRESAR                       */
/*****************************************************************************/
void __fastcall TForm1::Image3MouseMove(TObject *Sender, TShiftState Shift,
     int X, int Y)
{
 b_intro_on->Visible = false;
 b_demos_on->Visible = false;
}
/*****************************************************************************/
/*                      EFECTO BOTON DE INGRESAR                             */
/*****************************************************************************/
void __fastcall TForm1::Image5MouseMove(TObject *Sender, TShiftState Shift,
     int X, int Y)
{
 Image5->Picture->LoadFromFile("b-entrar-on.JPG");
}
/*****************************************************************************/
/*                    OPCION DE REGRESAR DEL DEMO 1                          */
/*****************************************************************************/
void __fastcall TForm1::Label34Click(TObject *Sender)
{
 demos->Visible = false;
 auxp=0;
 Label33->Enabled = false;
}
/*****************************************************************************/
/*                    INGRESAR A LA OPCION DE DEMOS                          */
/*****************************************************************************/
void __fastcall TForm1::Image5Click(TObject *Sender)
{
 p_principal->Visible = false;
}
/*****************************************************************************/
/*                     INICIALIZAR LOS DEMOS                                 */
/*****************************************************************************/
/*este procedimiento carga las primeras imagenes que salen en el menu de
```

```
  demos, como las imagenes estan contenidas en la carpeta el sistemas hace
  un llamado del archivo*/
/*****************************************************************************/
void __fastcall TForm1::Image4Click(TObject *Sender)
{
  demos->Visible = true;
  Label33->Enabled = false;
  atras2->Enabled  = false;
  atras3->Enabled  = false;
  atras4->Enabled  = false;
  Image6->Picture->LoadFromFile("pantalla 1.JPG");
  demo2->Picture->LoadFromFile ("pantalla 21.JPG");
  demo3->Picture->LoadFromFile ("pantalla 31.JPG");
  demo4->Picture->LoadFromFile ("pantalla 41.JPG");
}
/*****************************************************************************/
/*                            DEMOS                                          */
/*****************************************************************************/
/*Los procedimientos que siguen a continuacion funcionan base de contadores,
  cada uno de los demos tiene su respectivo, el contador se incrementa a
  medida de que cada uno de los demos es utilizado, los procedimientos
  cargan las imagenes a medida de los contadores se incrementan o se
  decrementan evaluandolos mediante condicionales switch                    */
/*****************************************************************************/
void __fastcall TForm1::Label32Click(TObject *Sender)
{
 auxp++;
 switch (auxp)
 {
  case 1 : {Image6->Picture->LoadFromFile("pantalla 2.JPG"); Label33->Enabled = true; Label32->Enabled =
true; break;}
  case 2 : {Image6->Picture->LoadFromFile("pantalla 3.JPG"); Label33->Enabled = true; Label32->Enabled =
true; break;}
  case 3 : {Image6->Picture->LoadFromFile("pantalla 4.JPG"); Label33->Enabled = true; Label32->Enabled =
true; break;}
  case 4 : {Image6->Picture->LoadFromFile("pantalla 5.JPG"); Label33->Enabled = true; Label32->Enabled =
true; break;}
  case 5 : {Image6->Picture->LoadFromFile("pantalla 6.JPG"); Label33->Enabled = true; Label32->Enabled =
true; break;}
  case 6 : {Image6->Picture->LoadFromFile("pantalla 7.JPG"); Label33->Enabled = true; Label32->Enabled =
true; break;}
```

```
  case 7 : {Image6->Picture->LoadFromFile("pantalla 8.JPG"); Label33->Enabled = true; Label32->Enabled =
false; break;}
 }
}
/*****************************************************************************/
void __fastcall TForm1::Label33Click(TObject *Sender)
{
 auxp--;
 if (auxp == 0)
   {
    Label33->Enabled = false;
    Image6->Picture->LoadFromFile("pantalla 1.JPG");
   }
 switch (auxp)
 {
  case 1 : {Image6->Picture->LoadFromFile("pantalla 2.JPG"); Label33->Enabled = true; Label32->Enabled =
true; break;}
  case 2 : {Image6->Picture->LoadFromFile("pantalla 3.JPG"); Label33->Enabled = true; Label32->Enabled =
true; break;}
  case 3 : {Image6->Picture->LoadFromFile("pantalla 4.JPG"); Label33->Enabled = true; Label32->Enabled =
true; break;}
  case 4 : {Image6->Picture->LoadFromFile("pantalla 5.JPG"); Label33->Enabled = true; Label32->Enabled =
true; break;}
  case 5 : {Image6->Picture->LoadFromFile("pantalla 6.JPG"); Label33->Enabled = true; Label32->Enabled =
true; break;}
  case 6 : {Image6->Picture->LoadFromFile("pantalla 7.JPG"); Label33->Enabled = true; Label32->Enabled =
true; break;}
  case 7 : {Image6->Picture->LoadFromFile("pantalla 8.JPG"); Label33->Enabled = true; Label32->Enabled =
true; break;}
 }
}
/*****************************************************************************/
void __fastcall TForm1::regresar2Click(TObject *Sender)
{
 demos->Visible = false;
 auxm=0;
 atras2->Enabled = false;
}
/*****************************************************************************/
void __fastcall TForm1::atras2Click(TObject *Sender)
{
 auxm--;
```

```
 if (auxm == 0)
   {
    atras2->Enabled = false;
    demo2->Picture->LoadFromFile("pantalla 21.JPG");
   }
 switch (auxm)
 {
  case 1 : {demo2->Picture->LoadFromFile("pantalla 22.JPG"); atras2->Enabled = true; adelante2->Enabled =
true; break;}
  case 2 : {demo2->Picture->LoadFromFile("pantalla 23.JPG"); atras2->Enabled = true; adelante2->Enabled =
true; break;}
  case 3 : {demo2->Picture->LoadFromFile("pantalla 24.JPG"); atras2->Enabled = true; adelante2->Enabled =
true; break;}
  case 4 : {demo2->Picture->LoadFromFile("pantalla 25.JPG"); atras2->Enabled = true; adelante2->Enabled =
true; break;}
  case 5 : {demo2->Picture->LoadFromFile("pantalla 26.JPG"); atras2->Enabled = true; adelante2->Enabled =
true; break;}
  case 6 : {demo2->Picture->LoadFromFile("pantalla 27.JPG"); atras2->Enabled = true; adelante2->Enabled =
true; break;}
  case 7 : {demo2->Picture->LoadFromFile("pantalla 28.JPG"); atras2->Enabled = true; adelante2->Enabled =
true; break;}
 }
}
/*************************************************************************/
void __fastcall TForm1::adelante2Click(TObject *Sender)
{
 auxm++;
 switch (auxm)
 {
  case 1 : {demo2->Picture->LoadFromFile("pantalla 22.JPG"); atras2->Enabled = true; adelante2->Enabled =
true; break;}
  case 2 : {demo2->Picture->LoadFromFile("pantalla 23.JPG"); atras2->Enabled = true; adelante2->Enabled =
true; break;}
  case 3 : {demo2->Picture->LoadFromFile("pantalla 24.JPG"); atras2->Enabled = true; adelante2->Enabled =
true; break;}
  case 4 : {demo2->Picture->LoadFromFile("pantalla 25.JPG"); atras2->Enabled = true; adelante2->Enabled =
true; break;}
  case 5 : {demo2->Picture->LoadFromFile("pantalla 26.JPG"); atras2->Enabled = true; adelante2->Enabled =
true; break;}
  case 6 : {demo2->Picture->LoadFromFile("pantalla 27.JPG"); atras2->Enabled = true; adelante2->Enabled =
true; break;}
```

```
  case 7 : {demo2->Picture->LoadFromFile("pantalla 28.JPG"); atras2->Enabled = true; adelante2->Enabled =
false; break;}
 }
}
/*****************************************************************************/
void __fastcall TForm1::regresar3Click(TObject *Sender)
{
 demos->Visible = false;
 auxn=0;
 atras3->Enabled = false;
}
/*****************************************************************************/
void __fastcall TForm1::adelante3Click(TObject *Sender)
{
 auxn++;
 switch (auxn)
 {
  case 1 : {demo3->Picture->LoadFromFile("pantalla 32.JPG"); atras3->Enabled = true; adelante3->Enabled =
true; break;}
  case 2 : {demo3->Picture->LoadFromFile("pantalla 33.JPG"); atras3->Enabled = true; adelante3->Enabled =
true; break;}
  case 3 : {demo3->Picture->LoadFromFile("pantalla 34.JPG"); atras3->Enabled = true; adelante3->Enabled =
true; break;}
  case 4 : {demo3->Picture->LoadFromFile("pantalla 35.JPG"); atras3->Enabled = true; adelante3->Enabled =
true; break;}
  case 5 : {demo3->Picture->LoadFromFile("pantalla 36.JPG"); atras3->Enabled = true; adelante3->Enabled =
true; break;}
  case 6 : {demo3->Picture->LoadFromFile("pantalla 37.JPG"); atras3->Enabled = true; adelante3->Enabled =
false; break;}
 }
}
/*****************************************************************************/
void __fastcall TForm1::atras3Click(TObject *Sender)
{
 auxn--;
 if (auxn == 0)
   {
    atras3->Enabled = false;
    demo3->Picture->LoadFromFile("pantalla 31.JPG");
   }
 switch (auxn)
 {
```

```
  case 1 : {demo3->Picture->LoadFromFile("pantalla 32.JPG"); atras3->Enabled = true; adelante3->Enabled =
true; break;}
  case 2 : {demo3->Picture->LoadFromFile("pantalla 33.JPG"); atras3->Enabled = true; adelante3->Enabled =
true; break;}
  case 3 : {demo3->Picture->LoadFromFile("pantalla 34.JPG"); atras3->Enabled = true; adelante3->Enabled =
true; break;}
  case 4 : {demo3->Picture->LoadFromFile("pantalla 35.JPG"); atras3->Enabled = true; adelante3->Enabled =
true; break;}
  case 5 : {demo3->Picture->LoadFromFile("pantalla 36.JPG"); atras3->Enabled = true; adelante3->Enabled =
true; break;}
  case 6 : {demo3->Picture->LoadFromFile("pantalla 37.JPG"); atras3->Enabled = true; adelante3->Enabled =
true; break;}
 }
}
/*****************************************************************************/
void __fastcall TForm1::regresar4Click(TObject *Sender)
{
 demos->Visible = false;
 auxu=0;
 atras4->Enabled = false;
}
/*****************************************************************************/
void __fastcall TForm1::adelante4Click(TObject *Sender)
{
 auxu++;
 switch (auxu)
 {
  case 1 : {demo4->Picture->LoadFromFile("pantalla 42.JPG"); atras4->Enabled = true; adelante4->Enabled =
true; break;}
  case 2 : {demo4->Picture->LoadFromFile("pantalla 43.JPG"); atras4->Enabled = true; adelante4->Enabled =
true; break;}
  case 3 : {demo4->Picture->LoadFromFile("pantalla 44.JPG"); atras4->Enabled = true; adelante4->Enabled =
true; break;}
  case 4 : {demo4->Picture->LoadFromFile("pantalla 45.JPG"); atras4->Enabled = true; adelante4->Enabled =
true; break;}
  case 5 : {demo4->Picture->LoadFromFile("pantalla 46.JPG"); atras4->Enabled = true; adelante4->Enabled =
true; break;}
  case 6 : {demo4->Picture->LoadFromFile("pantalla 47.JPG"); atras4->Enabled = true; adelante4->Enabled =
true; break;}
  case 7 : {demo4->Picture->LoadFromFile("pantalla 48.JPG"); atras4->Enabled = true; adelante4->Enabled =
false; break;}
 }
```

```
}
/*****************************************************************************/
void __fastcall TForm1::atras4Click(TObject *Sender)
{
 auxu--;
 if (auxu == 0)
   {
    atras4->Enabled = false;
    demo4->Picture->LoadFromFile("pantalla 41.JPG");
   }
 switch (auxu)
 {
  case 1 : {demo4->Picture->LoadFromFile("pantalla 42.JPG"); atras4->Enabled = true; adelante4->Enabled =
true; break;}
  case 2 : {demo4->Picture->LoadFromFile("pantalla 43.JPG"); atras4->Enabled = true; adelante4->Enabled =
true; break;}
  case 3 : {demo4->Picture->LoadFromFile("pantalla 44.JPG"); atras4->Enabled = true; adelante4->Enabled =
true; break;}
  case 4 : {demo4->Picture->LoadFromFile("pantalla 45.JPG"); atras4->Enabled = true; adelante4->Enabled =
true; break;}
  case 5 : {demo4->Picture->LoadFromFile("pantalla 46.JPG"); atras4->Enabled = true; adelante4->Enabled =
true; break;}
  case 6 : {demo4->Picture->LoadFromFile("pantalla 47.JPG"); atras4->Enabled = true; adelante4->Enabled =
true; break;}
  case 7 : {demo4->Picture->LoadFromFile("pantalla 48.JPG"); atras4->Enabled = true; adelante4->Enabled =
false; break;}
 }
}
/*****************************************************************************/
void __fastcall TForm1::b_intro_offMouseMove(TObject *Sender,
      TShiftState Shift, int X, int Y)
{
 b_intro_on->Visible = true;
}
/*****************************************************************************/
void __fastcall TForm1::b_demo_offMouseMove(TObject *Sender,
      TShiftState Shift, int X, int Y)
{
 b_demos_on->Visible = true;
}
/*****************************************************************************/
/*****************************************************************************/
```

```
/***************************************************************************/
void __fastcall TForm1::Borrar1Click(TObject *Sender)
{
     tam = StrToInt(Edit1->Text) + 1;

if (tam>2)
{
  StringGrid1->ColCount=tam;
  StringGrid1->RowCount=tam;
  for (int i=0; i<=tam; i++)
  {
    StringGrid1->Cells[i][0] = IntToStr(i);
  }
  for (int j=0; j<= tam; j++)
    StringGrid1->Cells[0][j] = IntToStr(j);
  for (int i=1;i<=tam;i++)
    for (int j=1;j<=tam;j++)
      StringGrid1->Cells[i][j] = IntToStr(0);
  for (int i=1;i<=tam-1;i++)
  {
    ComboBox1->Items->Add(IntToStr(i));
    ComboBox2->Items->Add(IntToStr(i));
  }
}

}
//---------------------------------------------------------------------------

void __fastcall TForm1::ProbabilidadesEstacionarias1Click(TObject *Sender)
{
  int cont = 0;
  for (int i=1; i<StringGrid1->RowCount; i++)
    for (int j=1; j<StringGrid1->ColCount; j++)
      if ((j==i) && (StringGrid1->Cells[i][j] == "1"))
        cont++;

  if (cont != 0)
    ShowMessage(AnsiString("No es una Matriz Ergodica. Por favor Verifique la matriz, si desea más
información por favor remitase a la ayuda"));
  else
  {
    for (int i=0; i<StringGrid1->RowCount; i++)
```

```
      for (int j=0; j<StringGrid1->ColCount; j++)
        Form3->StringGrid1->Cells[i][j] = StringGrid1->Cells[i][j];
      Form3->StringGrid1->RowCount = StringGrid1->RowCount;
      Form3->StringGrid1->ColCount = StringGrid1->ColCount;
      Form3->Visible=true;
  }

}
//---------------------------------------------------------------------------


void __fastcall TForm1::ProbabilidadesIncondicionalesenNpasos1Click(
      TObject *Sender)
{
  int cont = 0;
  for (int i=1; i<StringGrid1->RowCount; i++)
    for (int j=1; j<StringGrid1->ColCount; j++)
      if ((j==i) && (StringGrid1->Cells[i][j] == "1"))
        cont++;

  if (cont != 0)
   { ShowMessage(AnsiString("No es una Matriz Ergodica. Por favor Verifique la matriz, si desea más
información por favor remitase a la ayuda"));
      return ;}
    for (int i=0; i<StringGrid1->RowCount; i++)
      for (int j=0; j<StringGrid1->ColCount; j++)
      Form6->StringGrid1->Cells[i][j] = StringGrid1->Cells[i][j];

      Form6->StringGrid1->RowCount = StringGrid1->RowCount;
      Form6->StringGrid1->ColCount = StringGrid1->ColCount;
      Form6->Visible=true;


}
//---------------------------------------------------------------------------
```

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "Mat2.h"
#include "math.h"
#include "Mat1.cpp"
#include "MATRIZ.CPP"


//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"

//typedef matrix Matrix;

TForm2 *Form2;
float mat[50][50];
//---------------------------------------------------------------------------
__fastcall TForm2::TForm2(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm2::Button1Click(TObject *Sender)
{
StringGrid3->RowCount=StringGrid1->RowCount;
StringGrid3->ColCount=StringGrid1->ColCount;

for (int i=0;i<=StringGrid3->RowCount;i++){
        for (int j=0;j<=StringGrid3->ColCount;j++){
         if ((j==0)||(i==0)){
            StringGrid3->Cells[i][j]=StringGrid1->Cells[i][j];
         }
         else{
         StringGrid3->Cells[i][j]=FloatToStr(StrToFloat(StringGrid2->Cells[i][j])-                StrToFloat(StringGrid1-
>Cells[i][j]));
         }
        }
}
```

```
for (int i=1;i<=StringGrid3->RowCount;i++){
        for (int j=1;j<=StringGrid3->ColCount;j++){
                mat[i][j]=StrToFloat(StringGrid3->Cells[i][j]);
                }
        }
}
//---------------------------------------------------------------------------


void __fastcall TForm2::SpeedButton1Click(TObject *Sender)
{
  float dato;
  int col,fil;
  CMatriz <float> a, b;

  a.nColumnas=StrToInt(StringGrid3->ColCount-1);
  a.nFilas=StrToInt(StringGrid3->RowCount-1);
  a.Matriz = new float *[a.nColumnas]; // Creo el vector de vectores...
  for(int i=0;i<a.nFilas&&i>=0;i++) // Se crean los vectores...
    a.Matriz[i] = new float[a.nFilas];
  for(int i=0;i<a.nColumnas;i++)
    for(int j=0;j<a.nFilas;j++)
       a.Matriz[i][j] = (float)0;


  for (int i=1;i<=a.nFilas;i++)
  {
    for (int j=1;j<=a.nColumnas;j++)
    {
      dato = (float)StrToFloat(StringGrid3->Cells[i][j]);
      a.Matriz[i-1][j-1] =(float)dato;
    }
  }

  b.nColumnas = StrToInt(StringGrid3->ColCount-1);
  b.nFilas = StrToInt(StringGrid3->RowCount-1);
  b.Matriz = new float *[b.nColumnas]; // Creo el vector de vectores...
  for(int i=0;i<b.nFilas&&i>=0;i++) // Se crean los vectores...
    b.Matriz[i] = new float[b.nFilas];
  for(int i=0;i<b.nColumnas;i++)
    for(int j=0;j<b.nFilas;j++)
```

```
      b.Matriz[i][j] = (float)0;

  a.Inversa(a,b);

  StringGrid4->ColCount = StringGrid3->ColCount;
  StringGrid4->RowCount = StringGrid3->RowCount;

  for (int i=0;i<=StringGrid3->RowCount;i++)
  {
    for (int j=0;j<=StringGrid3->ColCount;j++)
    {
      if ((j==0)||(i==0))
      {
        StringGrid4->Cells[i][j]=StringGrid3->Cells[i][j];
      }
    }
  }

  for (int i=1;i<=b.nFilas;i++)
  {
    for (int j=1;j<=b.nColumnas;j++)
    {
      StringGrid4->Cells[i][j] = b.Matriz[i-1][j-1];
    }
  }
}
//---------------------------------------------------------------------------

void __fastcall TForm2::BitBtn1Click(TObject *Sender)
{
  float res = 0;
  int cont = 0;

  for (int i=1; i<=StringGrid4->RowCount-1; i++)
    for (int j=1; j<=StringGrid4->ColCount-1; j++)
    {
      if (i == j)
        res = (float)((StrToFloat(StringGrid4->Cells[i][j]) - 1) / StrToFloat(StringGrid4->Cells[i][i]));
      else
        res = (float)(StrToFloat(StringGrid4->Cells[i][j])) / StrToFloat(StringGrid4->Cells[i][i]);
      StringGrid5->Cells[cont][0] = AnsiString("F") + FloatToStr(i) + FloatToStr(j);
      StringGrid5->Cells[cont][1] = (float)res;
```

```
      cont++;
    }
  StringGrid5->ColCount = cont;
}
//---------------------------------------------------------------------------
void __fastcall TForm2::Button2Click(TObject *Sender)
{

  Form2->Visible = false;
  for (int w=1; w < StringGrid1->ColCount; w++)
     for (int y=1; y < StringGrid1->RowCount; y++)
        StringGrid1->Cells[w][y]="";
  for (int w=1; w < StringGrid2->ColCount; w++)
     for (int y=1; y < StringGrid2->RowCount; y++)
        StringGrid2->Cells[w][y]="";
  for (int w=1; w < StringGrid3->ColCount; w++)
     for (int y=1; y < StringGrid3->RowCount; y++)
        StringGrid3->Cells[w][y]="";
  for (int w=1; w < StringGrid4->ColCount; w++)
     for (int y=1; y < StringGrid4->RowCount; y++)
        StringGrid4->Cells[w][y]="";
  for (int w=0; w <= StringGrid5->ColCount; w++)
     for (int y=0; y <= StringGrid5->RowCount; y++)
        StringGrid5->Cells[w][y]="";

}
//---------------------------------------------------------------------------

//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "Mat3.h"
#include "MATRIZ.CPP"
#include "Unit4.h"
#include "Mat1.cpp"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm3 *Form3;
//---------------------------------------------------------------------------
```

```
__fastcall TForm3::TForm3(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm3::BitBtn1Click(TObject *Sender)
{
  String formula = "";
  for (int j=1; j<StringGrid1->ColCount; j++)
  {
    formula = AnsiString("ß") + IntToStr(j) + AnsiString("=");
    for (int i=1; i<StringGrid1->RowCount; i++)
    {
      if (i != StringGrid1->RowCount - 1)
        formula += StringGrid1->Cells[i][j] + "ß" + IntToStr(i) + " + ";
      else
       formula += StringGrid1->Cells[i][j] + "ß" + IntToStr(i);
    }
    ListBox1->Items->Add(formula);
  }
  formula = "";
  for (int i=1; i<StringGrid1->RowCount; i++)
    if (i != StringGrid1->RowCount - 1)
      formula += AnsiString("ß") + IntToStr(i) + " + ";
    else
      formula += AnsiString("ß") + IntToStr(i);
  formula += AnsiString(" = 1");
  ListBox1->Items->Add(formula);
}
//---------------------------------------------------------------------------

void __fastcall TForm3::BitBtn2Click(TObject *Sender)
{
     Form4->Show();
}
//---------------------------------------------------------------------------


void __fastcall TForm3::FormShow(TObject *Sender)
{
    CMatriz <float> a, b;
    for (int i=0; i<StringGrid1->RowCount; i++)
```

```
for (int j=0; j<StringGrid1->ColCount; j++)
{
  if (j==0)
    Form4->StringGrid2->Cells[i][j] = i;
  else
    Form4->StringGrid2->Cells[i][j] = j;
}

for (int i=1; i<StringGrid1->RowCount; i++)
  for (int j=2; j<StringGrid1->ColCount; j++)
  {
    if (i==j)
      Form4->StringGrid2->Cells[i][j-1] = StringGrid1->Cells[i][j]-1;
    else
      Form4->StringGrid2->Cells[i][j-1] = StringGrid1->Cells[i][j];
  }
Form4->StringGrid2->RowCount = StringGrid1->RowCount;
Form4->StringGrid2->ColCount = StringGrid1->ColCount;

for (int i=1; i<Form4->StringGrid2->ColCount; i++)
  Form4->StringGrid2->Cells[i][Form4->StringGrid2->ColCount-1] = 1;

a.nColumnas=StrToInt(Form4->StringGrid2->ColCount-1);
a.nFilas=StrToInt(Form4->StringGrid2->RowCount-1);
a.Matriz = new float *[a.nColumnas]; // Creo el vector de vectores...
for(int i=0;i<a.nFilas&&i>=0;i++) // Se crean los vectores...
  a.Matriz[i] = new float[a.nFilas];
for(int i=0;i<a.nColumnas;i++)
  for(int j=0;j<a.nFilas;j++)
    a.Matriz[i][j] = (float)0;

b.nColumnas=StrToInt(Form4->StringGrid2->ColCount-1);
b.nFilas=StrToInt(Form4->StringGrid2->RowCount-1);
b.Matriz = new float *[b.nColumnas]; // Creo el vector de vectores...
for(int i=0;i<b.nFilas&&i>=0;i++) // Se crean los vectores...
  b.Matriz[i] = new float[b.nFilas];
for(int i=0;i<b.nColumnas;i++)
  for(int j=0;j<b.nFilas;j++)
    b.Matriz[i][j] = (float)0;


for (int i=1;i<=a.nFilas;i++)
```

```
    {
      for (int j=1;j<=a.nColumnas;j++)
      {
        float dato = (float)StrToFloat(Form4->StringGrid2->Cells[i][j]);
        a.Matriz[i-1][j-1] =(float)dato;
      }
    }

    String pis,pis2;
    Form4->ListBox1->Items->Clear() ;
    Form4->ListBox2->Items->Clear() ;
    String determinantes = "Los determinantes son: ";
    Form4->ListBox2->Items->Add(determinantes);
    float determinante = (float) a.Determinante(a);
    float determinanteA = (float) a.Determinante(a);
    if (determinante == (float) 0)
    {
      BitBtn2->Enabled = false;
    }
    Form4->txtDeterminante->Text = determinante;

    for (int j=0; j<a.nColumnas;j++)
    {
      b = a;
      for (int i=0;i<(a.nFilas-1);i++)
      {
        b.Matriz[i][j] = (float)0;
      }
      determinante = (float) b.Determinante(b);
      determinantes = FloatToStr(determinante);
      Form4->ListBox2->Items->Add(determinantes);
      pis = AnsiString("μ") + (j+1) + " = " + FloatToStr(determinante/determinanteA);
      Form4->ListBox2->Items->Add(pis);
      pis2 = AnsiString("m") + (j+1) + " = " + FloatToStr((1/(determinante/determinanteA)));
      Form4->ListBox1->Items->Add(pis2);
    }
}
//---------------------------------------------------------------------------

void __fastcall TForm3::BitBtn4Click(TObject *Sender)
{
/*  int tam = StringGrid1->ColCount;
```

```
 if (tam > 1)
 {
   Form1->StringGrid1->ColCount=tam;
   Form1->StringGrid1->RowCount=tam;

   for (int i=0; i<=tam; i++)
   {
     Form1->StringGrid1->Cells[i][0] = IntToStr(i);
   }

   for (int j=0; j<=tam; j++)
     Form1->StringGrid1->Cells[0][j] = IntToStr(j);

   for (int i=1;i<=tam;i++)
     for (int j=1;j<=tam;j++)
       Form1->StringGrid1->Cells[i][j] = StringGrid1->Cells[i][j];
 }

 Form1->GroupBox2->Visible=true;
 Form1->Edit4->Text="1";
 Form1->Edit5->Text="1";
 Form1->BitBtn1->Caption = "Nuevo";
 Form1->Show();     */
 Form4->ListBox2->Clear();
 ListBox1->Clear();
 Form3->Visible = false;
  for (int w=1; w < StringGrid1->ColCount; w++)
    for (int y=1; y < StringGrid1->RowCount; y++)
      StringGrid1->Cells[w][y]="";
  for (int w=1; w < Form4->StringGrid2->ColCount; w++)
    for (int y=1; y < Form4->StringGrid2->RowCount; y++)
      Form4->StringGrid2->Cells[w][y]="";
}
//---------------------------------------------------------------------------

//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "Unit4.h"
```

```
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm4 *Form4;
//---------------------------------------------------------------------------
__fastcall TForm4::TForm4(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------

//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "Unit5.h"

#include "Mat2.h"
#include "math.h"
#include "Mat1.cpp"
#include "MATRIZ.CPP"

//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm5 *Form5;

//---------------------------------------------------------------------------
__fastcall TForm5::TForm5(TComponent* Owner)
        : TForm(Owner)
{
 int vv;
 vv = tam ;
 for (int y =1; y < vv; y++)
 {
  Image1->Canvas->TextOutA (y * 30, y * 30, IntToStr(y));
 }
}
//---------------------------------------------------------------------------

#ifndef matrix.h
```

```
#define matrix.h

////////////////////////////////
// Matrix TCL Lite v1.12
// Copyright (c) 1997-2001 Techsoft Pvt. Ltd. (See License.Txt file.)
//
// Matrix.h: Matrix C++ template class include file
// Web: http://www.techsoftpl.com/matrix/
// Email: matrix@techsoftpl.com
// Author: Somnath Kundu
//

////////////////////////////////
// Installation:
//
// Copy this "matrix.h" file into include directory of your compiler.
//

////////////////////////////////
// Note: This matrix template class defines majority of the matrix
// operations as overloaded operators or methods. It is assumed that
// users of this class is familiar with matrix algebra. We have not
// defined any specialization of this template here, so all the instances
// of matrix will be created implicitly by the compiler. The data types
// tested with this class are float, double, long double, complex<float>,
// complex<double> and complex<long double>. Note that this class is not
// optimized for performance.
//
// Since implementation of exception, namespace and template are still
// not standardized among the various (mainly old) compilers, you may
// encounter compilation error with some compilers. In that case remove
// any of the above three features by defining the following macros:
//
//  _NO_NAMESPACE:  Define this macro to remove namespace support.
//
//  _NO_EXCEPTION:  Define this macro to remove exception handling
//                  and use old style of error handling using function.
//
//  _NO_TEMPLATE:   If this macro is defined matrix class of double
//                  type will be generated by default. You can also
//                  generate a different type of matrix like float.
//
```

```
//  _SGI_BROKEN_STL: For SGI C++ v.7.2.1 compiler.
//
//  Since all the definitions are also included in this header file as
//  inline function, some compiler may give warning "inline function
//  can't be expanded". You may ignore/disable this warning using compiler
//  switches. All the operators/methods defined in this class have their
//  natural meaning except the followings:
//
//  Operator/Method                    Description
//  ---------------                    -----------
//   operator ()   :   This function operator can be used as a
//                     two-dimensional subscript operator to get/set
//                     individual matrix elements.
//
//   operator !    :   This operator has been used to calculate inversion
//                     of matrix.
//
//   operator ~    :   This operator has been used to return transpose of
//                     a matrix.
//
//   operator ^    :   It is used calculate power (by a scalar) of a matrix.
//                     When using this operator in a matrix equation, care
//                     must be taken by parenthesizing it because it has
//                     lower precedence than addition, subtraction,
//                     multiplication and division operators.
//
//   operator >>   :   It is used to read matrix from input stream as per
//                     standard C++ stream operators.
//
//   operator <<   :   It is used to write matrix to output stream as per
//                     standard C++ stream operators.
//
// Note that professional version of this package, Matrix TCL Pro 2.0
// is optimized for performance and supports many more matrix operations.
// Matrix TCL Pro 2.0 is available as shareware from our web site at
// http://www.techsoftpl.com/matrix/.
//

#ifndef __cplusplus
#error Must use C++ for the type matrix.
#endif
```

```
#if !defined(__STD_MATRIX_H)
#define __STD_MATRIX_H

//////////////////////////////
// First deal with various shortcomings and incompatibilities of
// various (mainly old) versions of popular compilers available.
//

#if defined(__BORLANDC__)
#pragma option -w-inl -w-pch
#endif

#if ( defined(__BORLANDC__) || _MSC_VER <= 1000 ) && !defined( __GNUG__ )
#  include <stdio.h>
#  include <stdlib.h>
#  include <math.h>
#  include <iostream.h>
#  include <string.h>
#else
#  include <cmath>
#  include <cstdio>
#  include <cstdlib>
#  include <string>
#  include <iostream>
#endif

#if defined(_MSC_VER) && _MSC_VER <= 1000
#  define _NO_EXCEPTION        // stdexception is not fully supported in MSVC++ 4.0
typedef int bool;
#  if !defined(false)
#    define false  0
#  endif
#  if !defined(true)
#    define true   1
#  endif
#endif

#if defined(__BORLANDC__) && !defined(__WIN32__)
#  define _NO_EXCEPTION        // std exception and namespace are not fully
#  define _NO_NAMESPACE        // supported in 16-bit compiler
#endif
```

```
#if defined(_MSC_VER) && !defined(_WIN32)
#  define _NO_EXCEPTION
#endif

#if defined(_NO_EXCEPTION)
#  define _NO_THROW
#  define _THROW_MATRIX_ERROR
#else
#  if defined(_MSC_VER)
#    if _MSC_VER >= 1020
#      include <stdexcept>
#    else
#      include <stdexcpt.h>
#    endif
#  elif defined(__MWERKS__)
#      include <stdexcept>
#  elif (__GNUC__ >= 2 || (__GNUC__ == 2 && __GNUC_MINOR__ >= 8))
#     include <stdexcept>
#  else
#     include <stdexcep>
#  endif
#  define _NO_THROW               throw ()
#  define _THROW_MATRIX_ERROR     throw (matrix_error)
#endif

#ifndef __MINMAX_DEFINED
#  define max(a,b)    (((a) > (b)) ? (a) : (b))
#  define min(a,b)    (((a) < (b)) ? (a) : (b))
#endif

#if defined(_MSC_VER)
#undef _MSC_EXTENSIONS      // To include overloaded abs function definitions!
#endif

#if ( defined(__BORLANDC__) || _MSC_VER ) && !defined( __GNUG__ )
inline float abs (float v) { return (float)fabs( v); }
inline double abs (double v) { return fabs( v); }
inline long double abs (long double v) { return fabsl( v); }
#endif

#if  defined(__GNUG__)  ||  defined(__MWERKS__)  ||  (defined(__BORLANDC__)  &&  (__BORLANDC__  >=
0x540))
```

```
#define FRIEND_FUN_TEMPLATE <>
#else
#define FRIEND_FUN_TEMPLATE
#endif

#if defined(_MSC_VER) && _MSC_VER <= 1020   // MSVC++ 4.0/4.2 does not
#  define _NO_NAMESPACE                     // support "std" namespace
#endif

#if !defined(_NO_NAMESPACE)
#if defined( _SGI_BROKEN_STL )              // For SGI C++ v.7.2.1 compiler
namespace std { }
#endif
using namespace std;
#endif

#ifndef _NO_NAMESPACE
namespace math {
#endif

#if !defined(_NO_EXCEPTION)
class matrix_error : public logic_error
{
   public:
          matrix_error (const string& what_arg) : logic_error( what_arg) {}
};
#define REPORT_ERROR(ErrormMsg)  throw matrix_error( ErrormMsg);
#else
inline void _matrix_error (const char* pErrMsg)
{
   cout << pErrMsg << endl;
   exit(1);
}
#define REPORT_ERROR(ErrormMsg)  _matrix_error( ErrormMsg);
#endif

#if !defined(_NO_TEMPLATE)
#  define MAT_TEMPLATE  template <class T>
#  define matrixT  matrix<T>
#else
#  define MAT_TEMPLATE
#  define matrixT  matrix
```

```
#  ifdef MATRIX_TYPE
    typedef MATRIX_TYPE T;
#  else
    typedef double T;
#  endif
#endif


MAT_TEMPLATE
class matrix
{
public:
  // Constructors
  matrix (const matrixT& m);
  matrix (size_t row = 6, size_t col = 6);

  // Destructor
  ~matrix ();

  // Assignment operators
  matrixT& operator = (const matrixT& m) _NO_THROW;

  // Value extraction method
  size_t RowNo () const { return _m->Row; }
  size_t ColNo () const { return _m->Col; }

  // Subscript operator
  T& operator () (size_t row, size_t col) _THROW_MATRIX_ERROR;
  T  operator () (size_t row, size_t col) const _THROW_MATRIX_ERROR;

  // Unary operators
  matrixT operator + () _NO_THROW { return *this; }
  matrixT operator - () _NO_THROW;

  // Combined assignment - calculation operators
  matrixT& operator += (const matrixT& m) _THROW_MATRIX_ERROR;
  matrixT& operator -= (const matrixT& m) _THROW_MATRIX_ERROR;
  matrixT& operator *= (const matrixT& m) _THROW_MATRIX_ERROR;
  matrixT& operator *= (const T& c) _NO_THROW;
  matrixT& operator /= (const T& c) _NO_THROW;
  matrixT& operator ^= (const size_t& pow) _THROW_MATRIX_ERROR;
```

```
  // Miscellaneous -methods
  void Null (const size_t& row, const size_t& col) _NO_THROW;
  void Null () _NO_THROW;
  void Unit (const size_t& row) _NO_THROW;
  void Unit () _NO_THROW;
  void SetSize (size_t row, size_t col) _NO_THROW;

  // Utility methods
  matrixT Solve (const matrixT& v) const _THROW_MATRIX_ERROR;
  matrixT Adj () _THROW_MATRIX_ERROR;
  matrixT Inv () _THROW_MATRIX_ERROR;
  T Det () const _THROW_MATRIX_ERROR;
  T Norm () _NO_THROW;
  T Cofact (size_t row, size_t col) _THROW_MATRIX_ERROR;
  T Cond () _NO_THROW;

  // Type of matrices
  bool IsSquare () _NO_THROW { return (_m->Row == _m->Col); }
  bool IsSingular () _NO_THROW;
  bool IsDiagonal () _NO_THROW;
  bool IsScalar () _NO_THROW;
  bool IsUnit () _NO_THROW;
  bool IsNull () _NO_THROW;
  bool IsSymmetric () _NO_THROW;
  bool IsSkewSymmetric () _NO_THROW;
  bool IsUpperTriangular () _NO_THROW;
  bool IsLowerTriangular () _NO_THROW;

private:
   struct base_mat
   {
          T **Val;
          size_t Row, Col, RowSiz, ColSiz;
          int Refcnt;

          base_mat (size_t row, size_t col, T** v)
          {
             Row = row; RowSiz = row;
             Col = col; ColSiz = col;
             Refcnt = 1;

             Val = new T* [row];
```

```
            size_t rowlen = col * sizeof(T);

            for (size_t i=0; i < row; i++)
            {
                Val[i] = new T [col];
                if (v) memcpy( Val[i], v[i], rowlen);
            }
        }
        ~base_mat ()
        {
            for (size_t i=0; i < RowSiz; i++)
                delete [] Val[i];
            delete [] Val;
        }
    };
    base_mat *_m;

    void clone ();
    void realloc (size_t row, size_t col);
    int pivot (size_t row);
};

#if defined(_MSC_VER) && _MSC_VER <= 1020
#  undef  _NO_THROW              // MSVC++ 4.0/4.2 does not support
#  undef  _THROW_MATRIX_ERROR    // exception specification in definition
#  define _NO_THROW
#  define _THROW_MATRIX_ERROR
#endif

// constructor
MAT_TEMPLATE inline
matrixT::matrix (size_t row, size_t col)
{
  _m = new base_mat( row, col, 0);
}

// copy constructor
MAT_TEMPLATE inline
matrixT::matrix (const matrixT& m)
{
    _m = m._m;
    _m->Refcnt++;
```

```
}

// Internal copy constructor
MAT_TEMPLATE inline void
matrixT::clone ()
{
    _m->Refcnt--;
    _m = new base_mat( _m->Row, _m->Col, _m->Val);
}

// destructor
MAT_TEMPLATE inline
matrixT::~matrix ()
{
   if (--_m->Refcnt == 0) delete _m;
}

// assignment operator
MAT_TEMPLATE inline matrixT&
matrixT::operator = (const matrixT& m) _NO_THROW
{
    m._m->Refcnt++;
    if (--_m->Refcnt == 0) delete _m;
    _m = m._m;
    return *this;
}

//  reallocation method
MAT_TEMPLATE inline void
matrixT::realloc (size_t row, size_t col)
{
   if (row == _m->RowSiz && col == _m->ColSiz)
   {
      _m->Row = _m->RowSiz;
      _m->Col = _m->ColSiz;
      return;
   }

   base_mat *m1 = new base_mat( row, col, NULL);
   size_t colSize = min(_m->Col,col) * sizeof(T);
   size_t minRow = min(_m->Row,row);
```

```
  for (size_t i=0; i < minRow; i++)
    memcpy( m1->Val[i], _m->Val[i], colSize);

  if (--_m->Refcnt == 0)
     delete _m;
  _m = m1;

  return;
}

// public method for resizing matrix
MAT_TEMPLATE inline void
matrixT::SetSize (size_t row, size_t col) _NO_THROW
{
  size_t i,j;
  size_t oldRow = _m->Row;
  size_t oldCol = _m->Col;

  if (row != _m->RowSiz || col != _m->ColSiz)
    realloc( row, col);

  for (i=oldRow; i < row; i++)
    for (j=0; j < col; j++)
          _m->Val[i][j] = T(0);

  for (i=0; i < row; i++)
    for (j=oldCol; j < col; j++)
          _m->Val[i][j] = T(0);

  return;
}

// subscript operator to get/set individual elements
MAT_TEMPLATE inline T&
matrixT::operator () (size_t row, size_t col) _THROW_MATRIX_ERROR
{
  if (row >= _m->Row || col >= _m->Col)
    REPORT_ERROR( "matrixT::operator(): Index out of range!");
  if (_m->Refcnt > 1) clone();
  return _m->Val[row][col];
}
```

```
// subscript operator to get/set individual elements
MAT_TEMPLATE inline T
matrixT::operator () (size_t row, size_t col) const _THROW_MATRIX_ERROR
{
  if (row >= _m->Row || col >= _m->Col)
    REPORT_ERROR( "matrixT::operator(): Index out of range!");
  return _m->Val[row][col];
}

// input stream function
MAT_TEMPLATE inline istream&
operator >> (istream& istrm, matrixT& m)
{
  for (size_t i=0; i < m.RowNo(); i++)
    for (size_t j=0; j < m.ColNo(); j++)
    {
      T x;
      istrm >> x;
      m(i,j) = x;
    }
  return istrm;
}

// output stream function
MAT_TEMPLATE inline ostream&
operator << (ostream& ostrm, const matrixT& m)
{
  for (size_t i=0; i < m.RowNo(); i++)
  {
    for (size_t j=0; j < m.ColNo(); j++)
    {
      T x = m(i,j);
      ostrm << x << '\t';
    }
    ostrm << endl;
  }
  return ostrm;
}


// logical equal-to operator
MAT_TEMPLATE inline bool
```

```
operator == (const matrixT& m1, const matrixT& m2) _NO_THROW
{
  if (m1.RowNo() != m2.RowNo() || m1.ColNo() != m2.ColNo())
    return false;

  for (size_t i=0; i < m1.RowNo(); i++)
    for (size_t j=0; j < m1.ColNo(); j++)
              if (m1(i,j) != m2(i,j))
                return false;

  return true;
}

// logical no-equal-to operator
MAT_TEMPLATE inline bool
operator != (const matrixT& m1, const matrixT& m2) _NO_THROW
{
   return (m1 == m2) ? false : true;
}

// combined addition and assignment operator
MAT_TEMPLATE inline matrixT&
matrixT::operator += (const matrixT& m) _THROW_MATRIX_ERROR
{
  if (_m->Row != m._m->Row || _m->Col != m._m->Col)
    REPORT_ERROR( "matrixT::operator+= : Inconsistent matrix sizes in addition!");
  if (_m->Refcnt > 1) clone();
  for (size_t i=0; i < m._m->Row; i++)
    for (size_t j=0; j < m._m->Col; j++)
          _m->Val[i][j] += m._m->Val[i][j];
  return *this;
}

// combined subtraction and assignment operator
MAT_TEMPLATE inline matrixT&
matrixT::operator -= (const matrixT& m) _THROW_MATRIX_ERROR
{
  if (_m->Row != m._m->Row || _m->Col != m._m->Col)
    REPORT_ERROR( "matrixT::operator-= : Inconsistent matrix sizes in subtraction!");
  if (_m->Refcnt > 1) clone();
  for (size_t i=0; i < m._m->Row; i++)
    for (size_t j=0; j < m._m->Col; j++)
```

```
            _m->Val[i][j] -= m._m->Val[i][j];
  return *this;
}

// combined scalar multiplication and assignment operator
MAT_TEMPLATE inline matrixT&
matrixT::operator *= (const T& c) _NO_THROW
{
   if (_m->Refcnt > 1) clone();
   for (size_t i=0; i < _m->Row; i++)
          for (size_t j=0; j < _m->Col; j++)
             _m->Val[i][j] *= c;
   return *this;
}

// combined matrix multiplication and assignment operator
MAT_TEMPLATE inline matrixT&
matrixT::operator *= (const matrixT& m) _THROW_MATRIX_ERROR
{
  if (_m->Col != m._m->Row)
    REPORT_ERROR( "matrixT::operator*= : Inconsistent matrix sizes in multiplication!");

  matrixT temp(_m->Row,m._m->Col);

  for (size_t i=0; i < _m->Row; i++)
    for (size_t j=0; j < m._m->Col; j++)
    {
      temp._m->Val[i][j] = T(0);
      for (size_t k=0; k < _m->Col; k++)
        temp._m->Val[i][j] += _m->Val[i][k] * m._m->Val[k][j];
    }
  *this = temp;

  return *this;
}

// combined scalar division and assignment operator
MAT_TEMPLATE inline matrixT&
matrixT::operator /= (const T& c) _NO_THROW
{
   if (_m->Refcnt > 1) clone();
   for (size_t i=0; i < _m->Row; i++)
```

```
            for (size_t j=0; j < _m->Col; j++)
               _m->Val[i][j] /= c;

    return *this;
}

// combined power and assignment operator
MAT_TEMPLATE inline matrixT&
matrixT::operator ^= (const size_t& pow) _THROW_MATRIX_ERROR
{
            matrixT temp(*this);

            for (size_t i=2; i <= pow; i++)
      *this = *this * temp;

            return *this;
}

// unary negation operator
MAT_TEMPLATE inline matrixT
matrixT::operator - () _NO_THROW
{
   matrixT temp(_m->Row,_m->Col);

   for (size_t i=0; i < _m->Row; i++)
      for (size_t j=0; j < _m->Col; j++)
            temp._m->Val[i][j] = - _m->Val[i][j];

   return temp;
}

// binary addition operator
MAT_TEMPLATE inline matrixT
operator + (const matrixT& m1, const matrixT& m2) _THROW_MATRIX_ERROR
{
   matrixT temp = m1;
   temp += m2;
   return temp;
}

// binary subtraction operator
MAT_TEMPLATE inline matrixT
```

```
operator - (const matrixT& m1, const matrixT& m2) _THROW_MATRIX_ERROR
{
  matrixT temp = m1;
  temp -= m2;
  return temp;
}

// binary scalar multiplication operator
MAT_TEMPLATE inline matrixT
operator * (const matrixT& m, const T& no) _NO_THROW
{
  matrixT temp = m;
  temp *= no;
  return temp;
}


// binary scalar multiplication operator
MAT_TEMPLATE inline matrixT
operator * (const T& no, const matrixT& m) _NO_THROW
{
  return (m * no);
}

// binary matrix multiplication operator
MAT_TEMPLATE inline matrixT
operator * (const matrixT& m1, const matrixT& m2) _THROW_MATRIX_ERROR
{
  matrixT temp = m1;
  temp *= m2;
  return temp;
}

// binary scalar division operator
MAT_TEMPLATE inline matrixT
operator / (const matrixT& m, const T& no) _NO_THROW
{
   return (m * (T(1) / no));
}


// binary scalar division operator
```

```
MAT_TEMPLATE inline matrixT
operator / (const T& no, const matrixT& m) _THROW_MATRIX_ERROR
{
    return (!m * no);
}

// binary matrix division operator
MAT_TEMPLATE inline matrixT
operator / (const matrixT& m1, const matrixT& m2) _THROW_MATRIX_ERROR
{
    return (m1 * !m2);
}

// binary power operator
MAT_TEMPLATE inline matrixT
operator ^ (const matrixT& m, const size_t& pow) _THROW_MATRIX_ERROR
{
   matrixT temp = m;
   temp ^= pow;
   return temp;
}

// unary transpose operator
MAT_TEMPLATE inline matrixT
operator ~ (const matrixT& m) _NO_THROW
{
   matrixT temp(m.ColNo(),m.RowNo());

   for (size_t i=0; i < m.RowNo(); i++)
      for (size_t j=0; j < m.ColNo(); j++)
      {
         T x = m(i,j);
                  temp(j,i) = x;
      }
   return temp;
}

// unary inversion operator
MAT_TEMPLATE inline matrixT
operator ! (const matrixT m) _THROW_MATRIX_ERROR
{
   matrixT temp = m;
```

```
  return temp.Inv();
}

// inversion function
MAT_TEMPLATE inline matrixT
matrixT::Inv () _THROW_MATRIX_ERROR
{
  size_t i,j,k;
  T a1,a2,*rowptr;

  if (_m->Row != _m->Col)
    REPORT_ERROR( "matrixT::operator!: Inversion of a non-square matrix");

  matrixT temp(_m->Row,_m->Col);
  if (_m->Refcnt > 1) clone();



  temp.Unit();
  for (k=0; k < _m->Row; k++)
  {
    int indx = pivot(k);
    if (indx == -1)
              REPORT_ERROR( "matrixT::operator!: Inversion of a singular matrix");

    if (indx != 0)
    {
              rowptr = temp._m->Val[k];
              temp._m->Val[k] = temp._m->Val[indx];
              temp._m->Val[indx] = rowptr;
    }
    a1 = _m->Val[k][k];
    for (j=0; j < _m->Row; j++)
    {
              _m->Val[k][j] /= a1;
              temp._m->Val[k][j] /= a1;
    }
    for (i=0; i < _m->Row; i++)
            if (i != k)
            {
              a2 = _m->Val[i][k];
              for (j=0; j < _m->Row; j++)
              {
```

```
                _m->Val[i][j] -= a2 * _m->Val[k][j];
                temp._m->Val[i][j] -= a2 * temp._m->Val[k][j];
              }
            }
  }
  return temp;
}

// solve simultaneous equation
MAT_TEMPLATE inline matrixT
matrixT::Solve (const matrixT& v) const _THROW_MATRIX_ERROR
{
  size_t i,j,k;
  T a1;

  if (!(_m->Row == _m->Col && _m->Col == v._m->Row))
    REPORT_ERROR( "matrixT::Solve():Inconsistent matrices!");

  matrixT temp(_m->Row,_m->Col+v._m->Col);
  for (i=0; i < _m->Row; i++)
  {
    for (j=0; j < _m->Col; j++)
        temp._m->Val[i][j] = _m->Val[i][j];
    for (k=0; k < v._m->Col; k++)
        temp._m->Val[i][_m->Col+k] = v._m->Val[i][k];
  }
  for (k=0; k < _m->Row; k++)
  {
    int indx = temp.pivot(k);
    if (indx == -1)
        REPORT_ERROR( "matrixT::Solve(): Singular matrix!");

    a1 = temp._m->Val[k][k];
    for (j=k; j < temp._m->Col; j++)
        temp._m->Val[k][j] /= a1;

    for (i=k+1; i < _m->Row; i++)
    {
        a1 = temp._m->Val[i][k];
        for (j=k; j < temp._m->Col; j++)
          temp._m->Val[i][j] -= a1 * temp._m->Val[k][j];
    }
```

```
  }
  matrixT s(v._m->Row,v._m->Col);
  for (k=0; k < v._m->Col; k++)
    for (int m=int(_m->Row)-1; m >= 0; m--)
    {
          s._m->Val[m][k] = temp._m->Val[m][_m->Col+k];
          for (j=m+1; j < _m->Col; j++)
            s._m->Val[m][k] -= temp._m->Val[m][j] * s._m->Val[j][k];
    }
  return s;
}

// set zero to all elements of this matrix
MAT_TEMPLATE inline void
matrixT::Null (const size_t& row, const size_t& col) _NO_THROW
{
   if (row != _m->Row || col != _m->Col)
          realloc( row,col);

   if (_m->Refcnt > 1)
          clone();

   for (size_t i=0; i < _m->Row; i++)
          for (size_t j=0; j < _m->Col; j++)
             _m->Val[i][j] = T(0);
   return;
}

// set zero to all elements of this matrix
MAT_TEMPLATE inline void
matrixT::Null() _NO_THROW
{
   if (_m->Refcnt > 1) clone();
   for (size_t i=0; i < _m->Row; i++)
          for (size_t j=0; j < _m->Col; j++)
                 _m->Val[i][j] = T(0);
   return;
}

// set this matrix to unity
MAT_TEMPLATE inline void
matrixT::Unit (const size_t& row) _NO_THROW
```

```
{
    if (row != _m->Row || row != _m->Col)
            realloc( row, row);

    if (_m->Refcnt > 1)
            clone();

    for (size_t i=0; i < _m->Row; i++)
            for (size_t j=0; j < _m->Col; j++)
                _m->Val[i][j] = i == j ? T(1) : T(0);
    return;
}

// set this matrix to unity
MAT_TEMPLATE inline void
matrixT::Unit () _NO_THROW
{
    if (_m->Refcnt > 1) clone();
    size_t row = min(_m->Row,_m->Col);
    _m->Row = _m->Col = row;

    for (size_t i=0; i < _m->Row; i++)
            for (size_t j=0; j < _m->Col; j++)
                _m->Val[i][j] = i == j ? T(1) : T(0);
    return;
}

// private partial pivoting method
MAT_TEMPLATE inline int
matrixT::pivot (size_t row)
{
  int k = int(row);
  double amax,temp;

  amax = -1;
  for (size_t i=row; i < _m->Row; i++)
    if ( (temp = abs( _m->Val[i][row])) > amax && temp != 0.0)
     {
       amax = temp;
       k = i;
     }
  if (_m->Val[k][row] == T(0))
```

```
    return -1;
  if (k != int(row))
  {
    T* rowptr = _m->Val[k];
    _m->Val[k] = _m->Val[row];
    _m->Val[row] = rowptr;
    return k;
  }
  return 0;
}

// calculate the determinant of a matrix
MAT_TEMPLATE inline T
matrixT::Det () const _THROW_MATRIX_ERROR
{
   size_t i,j,k;
   T piv,detVal = T(1);

   if (_m->Row != _m->Col)
     REPORT_ERROR( "matrixT::Det(): Determinant a non-square matrix!");

   matrixT temp(*this);
   if (temp._m->Refcnt > 1) temp.clone();

   for (k=0; k < _m->Row; k++)
   {
     int indx = temp.pivot(k);
     if (indx == -1)
           return 0;
     if (indx != 0)
           detVal = - detVal;
     detVal = detVal * temp._m->Val[k][k];
     for (i=k+1; i < _m->Row; i++)
     {
           piv = temp._m->Val[i][k] / temp._m->Val[k][k];
           for (j=k+1; j < _m->Row; j++)
             temp._m->Val[i][j] -= piv * temp._m->Val[k][j];
     }
   }
   return detVal;
}
```

```
// calculate the norm of a matrix
MAT_TEMPLATE inline T
matrixT::Norm () _NO_THROW
{
  T retVal = T(0);

  for (size_t i=0; i < _m->Row; i++)
    for (size_t j=0; j < _m->Col; j++)
          retVal += _m->Val[i][j] * _m->Val[i][j];
  retVal = sqrt( retVal);

  return retVal;
}

// calculate the condition number of a matrix
MAT_TEMPLATE inline T
matrixT::Cond () _NO_THROW
{
  matrixT inv = ! (*this);
  return (Norm() * inv.Norm());
}

// calculate the cofactor of a matrix for a given element
MAT_TEMPLATE inline T
matrixT::Cofact (size_t row, size_t col) _THROW_MATRIX_ERROR
{
  size_t i,i1,j,j1;

  if (_m->Row != _m->Col)
    REPORT_ERROR( "matrixT::Cofact(): Cofactor of a non-square matrix!");

  if (row > _m->Row || col > _m->Col)
    REPORT_ERROR( "matrixT::Cofact(): Index out of range!");

  matrixT temp (_m->Row-1,_m->Col-1);

  for (i=i1=0; i < _m->Row; i++)
  {
    if (i == row)
          continue;
    for (j=j1=0; j < _m->Col; j++)
    {
```

```
            if (j == col)
              continue;
            temp._m->Val[i1][j1] = _m->Val[i][j];
            j1++;
      }
      i1++;
    }
    T  cof = temp.Det();
    if ((row+col)%2 == 1)
      cof = -cof;

    return cof;
}


// calculate adjoin of a matrix
MAT_TEMPLATE inline matrixT
matrixT::Adj () _THROW_MATRIX_ERROR
{
    if (_m->Row != _m->Col)
      REPORT_ERROR( "matrixT::Adj(): Adjoin of a non-square matrix.");

    matrixT temp(_m->Row,_m->Col);

    for (size_t i=0; i < _m->Row; i++)
      for (size_t j=0; j < _m->Col; j++)
            temp._m->Val[j][i] = Cofact(i,j);
    return temp;
}

// Determine if the matrix is singular
MAT_TEMPLATE inline bool
matrixT::IsSingular () _NO_THROW
{
    if (_m->Row != _m->Col)
      return false;
    return (Det() == T(0));
}

// Determine if the matrix is diagonal
MAT_TEMPLATE inline bool
matrixT::IsDiagonal () _NO_THROW
```

```
{
  if (_m->Row != _m->Col)
    return false;
  for (size_t i=0; i < _m->Row; i++)
    for (size_t j=0; j < _m->Col; j++)
          if (i != j && _m->Val[i][j] != T(0))
            return false;
  return true;
}

// Determine if the matrix is scalar
MAT_TEMPLATE inline bool
matrixT::IsScalar () _NO_THROW
{
  if (!IsDiagonal())
    return false;
  T v = _m->Val[0][0];
  for (size_t i=1; i < _m->Row; i++)
    if (_m->Val[i][i] != v)
          return false;
  return true;
}

// Determine if the matrix is a unit matrix
MAT_TEMPLATE inline bool
matrixT::IsUnit () _NO_THROW
{
  if (IsScalar() && _m->Val[0][0] == T(1))
    return true;
  return false;
}

// Determine if this is a null matrix
MAT_TEMPLATE inline bool
matrixT::IsNull () _NO_THROW
{
  for (size_t i=0; i < _m->Row; i++)
    for (size_t j=0; j < _m->Col; j++)
          if (_m->Val[i][j] != T(0))
            return false;
  return true;
}
```

```
// Determine if the matrix is symmetric
MAT_TEMPLATE inline bool
matrixT::IsSymmetric () _NO_THROW
{
  if (_m->Row != _m->Col)
    return false;
  for (size_t i=0; i < _m->Row; i++)
    for (size_t j=0; j < _m->Col; j++)
          if (_m->Val[i][j] != _m->Val[j][i])
            return false;
  return true;
}

// Determine if the matrix is skew-symmetric
MAT_TEMPLATE inline bool
matrixT::IsSkewSymmetric () _NO_THROW
{
  if (_m->Row != _m->Col)
    return false;
  for (size_t i=0; i < _m->Row; i++)
    for (size_t j=0; j < _m->Col; j++)
          if (_m->Val[i][j] != -_m->Val[j][i])
            return false;
  return true;
}

// Determine if the matrix is upper triangular
MAT_TEMPLATE inline bool
matrixT::IsUpperTriangular () _NO_THROW
{
  if (_m->Row != _m->Col)
    return false;
  for (size_t i=1; i < _m->Row; i++)
    for (size_t j=0; j < i-1; j++)
          if (_m->Val[i][j] != T(0))
            return false;
  return true;
}

// Determine if the matrix is lower triangular
MAT_TEMPLATE inline bool
```

```
matrixT::IsLowerTriangular () _NO_THROW
{
  if (_m->Row != _m->Col)
    return false;

  for (size_t j=1; j < _m->Col; j++)
    for (size_t i=0; i < j-1; i++)
          if (_m->Val[i][j] != T(0))
            return false;

  return true;
}

#ifndef _NO_NAMESPACE
}
#endif

#endif //__STD_MATRIX_H

#define ERR_NO_ERROR     0x00 // Se definen los tipos de error que puede haber.
 #define ERR_INVALID_DATA 0x01 // No me he "mascao" diferenciandolos.


template <class TipoDato>
 class CMatriz
 {
   public:
     int       nFilas;
     int       nColumnas;
     TipoDato **Matriz;            // Matriz=Vector de Vectores...
     static int Error;

     CMatriz(); // Constructor por defecto...
     CMatriz(int,int);      // Constructor de MxN...
     CMatriz(CMatriz &);       // Constructor de copia...
     CMatriz(const CMatriz &); // Otro constructor de copia...
     ~CMatriz();               // El Destructor..., ¨como no?

     CMatriz <TipoDato>  operator = (CMatriz <TipoDato> &op);

     int SetSize(int,int);      // "Resiza" la matriz...
     int                Filas();
```

```
    int             Columnas();
    int             Dimension();   /* Falta por implementar. Hacer M,todo y Funci¢n. */
    CMatriz <TipoDato>   Kerf();       /* Falta por implementar. Hacer M,todo y Funci¢n. Hacerlo usando Gauss
*/
    CMatriz <TipoDato>   Imf();        /* Falta por implementar. Hacer M,todo y Funci¢n. Hacerlo usando Gauss
*/
    CMatriz <TipoDato>   MatrizIdentidad(int,TipoDato,TipoDato);
    CMatriz <TipoDato>   MatrizNula(int,TipoDato);
    CMatriz <TipoDato>   Inversa(CMatriz <TipoDato> op,CMatriz <TipoDato> &op2);
    TipoDato          Absoluto(TipoDato);
    TipoDato          Determinante(CMatriz <TipoDato>);
};
//-----------------------
template <class TipoDato>
int CMatriz<TipoDato>::Error; // En caso de error en alguna operaci¢n,
                        // el resultado devuelto ser  una matriz de 0x0
                        // y en esta variable "static" se guardar  el porqu, del
                        // error hasta que se haga otra operaci¢n.
//-----------------------
template <class TipoDato>
CMatriz <TipoDato> CMatriz <TipoDato>::MatrizIdentidad(int Size,TipoDato cero,TipoDato uno)
{                       // Devuelve la matriz identidad para matrices cuadradas de
  int i,j;              // tama¤o "Size". La raz¢n del parametro "uno" es
  CMatriz <TipoDato> id;      // porque el compilador necesita saber con que
                        // tipo se est  trabajando.
  id.SetSize(Size);
  for(i=0;i<Size;i++)
  {
    for(j=0;j<Size;j++)
      id.Matriz[i][j] = cero;
    id.Matriz[i][i] = uno;
  }
  return(id);
}

//-----------------------
template <class TipoDato>
CMatriz <TipoDato> MatrizNula(int Size,TipoDato cero)
{                     // Lo mismo que la matriz identidad, pero devolviendo
  int i,j;            // una matriz nula.
  CMatriz <TipoDato> null;
```

```
  null.SetSize(Size);
  for(i=0;i<Size;i++)
    for(j=0;j<Size;j++)
          null.Matriz[i][j] = cero;
  return(null);
}

//-----------------------
template <class TipoDato>
CMatriz<TipoDato>::CMatriz() // Constructor por defecto...
{                          // Simplemente lo inicializa todo para que
  Matriz = NULL;           // al mirar por el tama¤o de la matriz se sepa que
  nFilas = nColumnas = 0;  // no se ha usado a£n.
  Error = ERR_NO_ERROR;
}

//-----------------------
template <class TipoDato>
CMatriz<TipoDato>::CMatriz(int M,int N) // Constructor para matrices de las
{                // que se conoce ya el tama¤o. En caso de que sean cuadradas,
  int i,j;       // no es necesario pasar como parametro el n£mero de columnas.

  if((nFilas>0)&&(nColumnas>0)) // Si el tama¤o de la matriz es correcto...
  {
    Matriz = new TipoDato *[M]; // Creo el vector de vectores...
    for(i=0;i<N&&i>=0;i++) // Se crean los vectores...
      Matriz[i] = new TipoDato[N];
    for(i=0;i<M;i++)
      for(j=0;j<N;j++)
        Matriz[i][j] = (TipoDato)0;
    Error = ERR_NO_ERROR;
  }
  else
    Error = ERR_INVALID_DATA;
}

//-----------------------
template <class TipoDato>
CMatriz<TipoDato>::CMatriz(CMatriz <TipoDato> &copia)
{                          // Constructor de copia. ¨Necesita m s comentarios?
  int i,j;
```

```
  nFilas = copia.nFilas;
  nColumnas = copia.nColumnas;
  if(nFilas>0 && nColumnas>0) // Si tiene las medidas correctas...
  {
    Matriz = new TipoDato *[nFilas];
    for(i=0;i<nFilas && i>=0;i++)
      Matriz[i] = new TipoDato[nColumnas];
    for(i=0;i<nFilas;i++)
      for(j=0;j<nColumnas;j++)
        Matriz[i][j] = copia.Matriz[i][j];
    Error = ERR_NO_ERROR;
  }
  else
    Error = ERR_INVALID_DATA;
}



//-----------------------
template <class TipoDato>
CMatriz<TipoDato>::CMatriz(const CMatriz <TipoDato> &copia)
{                 // Otro constructor de copia. Para matrices constantes...
  int i,j;

  nFilas = copia.nFilas;
  nColumnas = copia.nColumnas;
  if(nFilas>0 && nColumnas>0) // Si tiene las medidas correctas...
  {
    Matriz = new TipoDato *[nFilas];
    for(i=0;i<nFilas && i>=0;i++)
      Matriz[i] = new TipoDato[nColumnas];
    for(i=0;i<nFilas;i++)
      for(j=0;j<nColumnas;j++)
    Matriz[i][j] = copia.Matriz[i][j];
    Error = ERR_NO_ERROR;
  }
  else
    Error = ERR_INVALID_DATA;
}

//-----------------------
template <class TipoDato>
```

```
CMatriz<TipoDato>::~CMatriz() // Destructor. Elimina la memoria usada por
{                             // la matriz.
  int i;

  if(Matriz)
  {
    for(i=0;i<nFilas;i++)
      delete Matriz[i];
    delete Matriz;
  }
}

//-----------------------
template <class TipoDato>
int CMatriz<TipoDato>::SetSize(int NFilas,int NColumnas)
{
  int i,j;
  int CopiarFilas, CopiarColumnas;
  TipoDato **OldMatriz;

  OldMatriz = Matriz;
  if(NColumnas==-1)
    NColumnas = NFilas;
  if(NFilas>0 && NColumnas>0) // Las dimensiones de la matriz deben ser positivas.
  {
    Matriz = new TipoDato *[NFilas];
    for(i=0;i<NFilas&&i>=0;i++)
      Matriz[i] = new TipoDato[NColumnas];
    CopiarFilas = NFilas>nFilas ? nFilas : NFilas;
    CopiarColumnas = NColumnas>nColumnas ? nColumnas : NColumnas;
    for(i=0;i<NFilas;i++)
    {
      for(j=0;j<NColumnas;j++)
        if(i<=CopiarFilas && j<=CopiarColumnas)
          Matriz[i][j] = OldMatriz[i][j]; // Copio el antiguo resultado...
        else
          Matriz[i][j] = (TipoDato)0; // Dado que es una posici¢n nueva, la creo a "0".
    }
    if(OldMatriz)
    {
      for(i=0;i<nFilas;i++)
        delete OldMatriz[i];
```

```
      delete OldMatriz; // Termino de borrar la matriz antigua.
    }
    nFilas = NFilas;
    nColumnas = NColumnas;
    Error = ERR_NO_ERROR;
  }
  else
  {
    if(nFilas)
    {
      for(i=0;i<nFilas;i++)
        delete Matriz[i];
      delete Matriz;
    }
    nFilas = 0;
    nColumnas = 0;
    Error = ERR_INVALID_DATA;
  }
  return(Error);
}

//-----------------------
template <class TipoDato>
int CMatriz<TipoDato>::Filas() // M,todo para saber el n£mero de filas de la matriz.
{
  return(nFilas);
}

//-----------------------
template <class TipoDato>
int CMatriz<TipoDato>::Columnas() // M,todo para saber el n£mero de columnas de la matriz
{
  return(nColumnas);
}

// operadores...
//-----------------------
template <class TipoDato>
CMatriz <TipoDato> CMatriz <TipoDato>::operator = (CMatriz <TipoDato>& op)
{                 // Sobrecarga del operador "=" para que se pueda asignar
  int i,j;        // una matriz a otra.
```

```
  if(nFilas==op.nFilas && nColumnas==op.nColumnas) // Si tienen el mismo tama¤o...
  {
    for(i=0;i<nFilas;i++)
          for(j=0;j<nColumnas;j++)
            Matriz[i][j] = op.Matriz[i][j];
    Error = ERR_NO_ERROR;
  }
  else
    Error = ERR_INVALID_DATA;
  return(*this);
}



//-----------------------
template <class TipoDato> // Halla el valor absoluto de un elemento. Para cuando no es tipo predefinido.
TipoDato absoluto(TipoDato dato)
{
  TipoDato resultado;

  if(dato<(TipoDato)0) // Si es menor que cero...
    resultado = ((TipoDato)(-1))*dato; // se multiplica por el -1...
  else
    resultado = dato;
  return(resultado);
}

//-----------------------
template <class TipoDato> // Calcula el determinante de una matriz, convirtiendola
TipoDato CMatriz <TipoDato>::Determinante(CMatriz <TipoDato> dato) // en una matriz triangular inferior
{                       // pero de determinante equivalente.
  int i, j, k, Size;
  TipoDato signo, factor, *buffer;

  if(dato.nFilas>0 && dato.nColumnas>0) // Si tiene las dimensiones adecuadas...
  {
    if(dato.nFilas==dato.nColumnas) // Si es una matriz cuadrada...
    {
          Size = dato.nFilas;
          signo = 1;
          for(i=0;i<Size;i++) // Se van recorriendo las filas a partir de la segunda...
          {
            if(dato.Matriz[i][i]==(TipoDato)0) // Si en esa posicion hay un cero...
```

```
            {
              for(j=i+1;j<Size;j++)
              {
                if(dato.Matriz[j][i]!=(TipoDato)0)
                {
                        buffer = dato.Matriz[j];
                        dato.Matriz[j] = dato.Matriz[i];
                        dato.Matriz[i] = buffer;
                        signo = -signo;
                        j = Size+1;
                }
              }
              if(j==Size)
                i = Size+1; // Codigo para salir (det=0)
            }
            for(j=i+1;j<Size;j++) // Se recorre el resto de la columna poniendo "ceros"...
            {
              factor = dato.Matriz[j][i]/dato.Matriz[i][i];
              for(k=0;k<Size;k++)
                dato.Matriz[j][k] = dato.Matriz[j][k]-dato.Matriz[i][k]*factor;
            }
          }
      factor = (TipoDato)1;
      for(i=0;i<Size;i++)
      factor = factor*dato.Matriz[i][i]; // det=prod(diagonal principal)
      dato.Error = ERR_NO_ERROR;
    }
    else
    dato.Error = ERR_INVALID_DATA;
  }
  else
    dato.Error = ERR_INVALID_DATA;
  if(dato.Error) // Si hubo un error, se devuelve un nulo, por devolver algo...
    factor = (TipoDato)0;
  return(signo*factor);
}

//-----------------------
template <class TipoDato>
CMatriz <TipoDato> CMatriz <TipoDato>::Inversa(CMatriz <TipoDato> op,CMatriz <TipoDato> &op2)
{                   // Calcula la inversa de una matriz mediante el metodo
  int i, j, k, x, y;    // de la pivotaci¢n total.
```

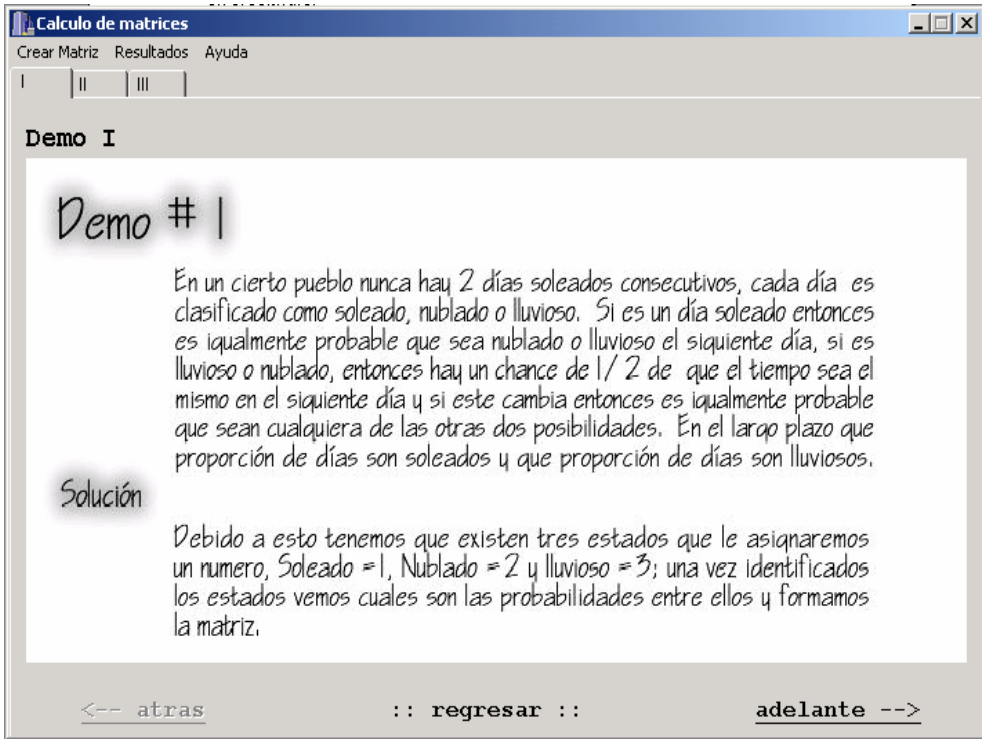
```
int Size,*Cambios;
TipoDato temp, *buffer;
CMatriz <TipoDato> invertir,identidad;

if((op.nFilas>0 && nFilas>0) && (op.nFilas==nColumnas)) // Si son matrices validas...
{
  if((Determinante(op)) != 0)
  {
    Size = nFilas;

invertir.nColumnas = Size;
invertir.nFilas = Size;
invertir.Matriz = new float *[nColumnas]; // Creo el vector de vectores...
for(int i=0;i<invertir.nColumnas&&i>=0;i++) // Se crean los vectores...
    invertir.Matriz[i] = new float[invertir.nColumnas];
for(int i=0;i<invertir.nColumnas;i++)
    for(int j=0;j<invertir.nColumnas;j++)
          invertir.Matriz[i][j] = (float)0;

identidad.nColumnas = Size;
identidad.nFilas = Size;
identidad.Matriz = new float *[nColumnas]; // Creo el vector de vectores...
for(int i=0;i<identidad.nColumnas&&i>=0;i++) // Se crean los vectores...
    identidad.Matriz[i] = new float[identidad.nColumnas];
for(int i=0;i<identidad.nColumnas;i++)
    for(int j=0;j<identidad.nColumnas;j++)
      identidad.Matriz[i][j] = (float)0;

int i,j;                    // tama¤o "Size". La raz¢n del parametro "uno" es
for(i=0;i<Size;i++)
{
  for(j=0;j<Size;j++)
    identidad.Matriz[i][j] = 0;
  identidad.Matriz[i][i] = 1;
}


invertir = op; // las matrices con las que trabajar,...
Cambios = new int[Size]; // Se almacenan los cambios...
for(i=0;i<Size;i++)
    Cambios[i] = i; // En principio no se han hecho cambios...
for(i=0;i<Size;i++)
```

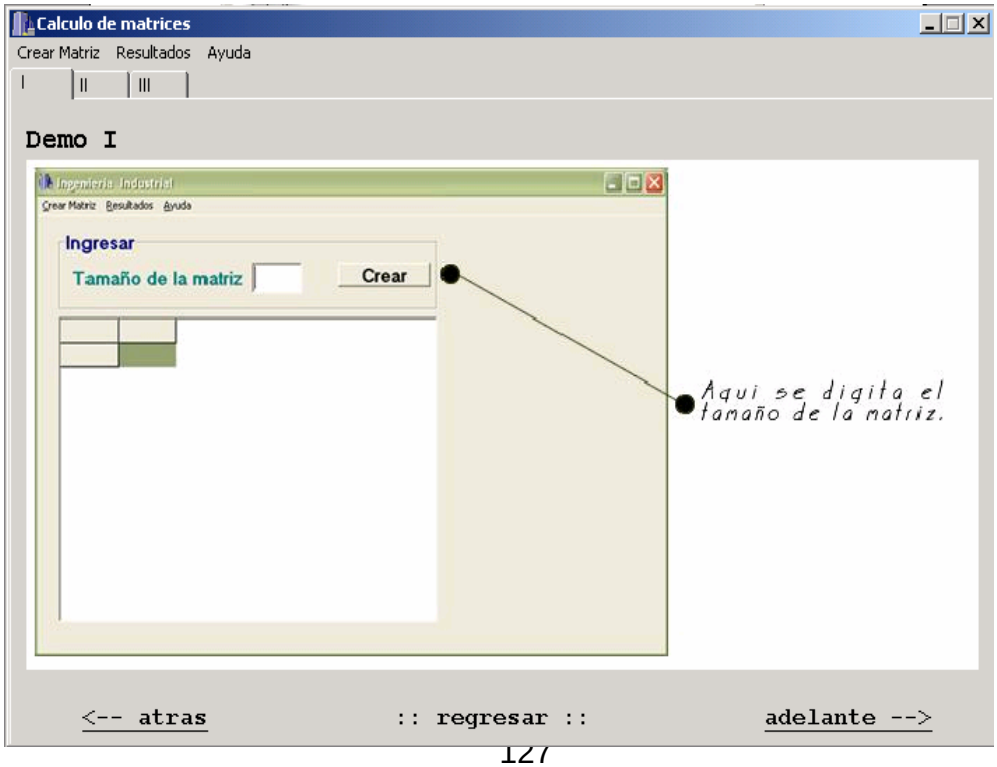
```
{
    temp = (TipoDato)0; // Todav¡a no se sabe cual es el mayor...
    for(j=i;j<Size;j++) // Para cada fila...
    {
      for(k=i;k<Size;k++) // Para cada elemento de la fila...
      {
        if( absoluto(invertir.Matriz[j][k]) >= temp ) // mirar la posicion en el x...
        {
          temp = absoluto(invertir.Matriz[j][k]);
          x = k; // Guardo la posici¢n del mayor...
          y = j; // idem...
        }
      }
    }
    if(temp==(TipoDato)0)
      i = Size; // Si el mayor valor absoluto es el cero, se sale del bucle...
    else        // aunque creo que este caso no se dar  nunca.
    {
      if(y!=i) // Si no est  en la fila que debe estar...
      { // Se intercambian las filas...
        buffer = invertir.Matriz[y]; // en una matriz...
        invertir.Matriz[y] = invertir.Matriz[i];
        invertir.Matriz[i] = buffer;
        buffer = identidad.Matriz[y]; // y en la otra...
        identidad.Matriz[y] = identidad.Matriz[i];
        identidad.Matriz[i] = buffer;
      }
      if(x!=i) // Si no est  en la columna que debe estar...
      { // Se intercambian las columnas...
        for(j=0;j<Size;j++)
        {
          temp = invertir.Matriz[j][x];  // Guardo la columna mayor,
          invertir.Matriz[j][x] = invertir.Matriz[j][i]; // , muevo la col. mayor a la col. i,
          invertir.Matriz[j][i] = temp; // y "repongo" el valor guardado en la nueva posici¢n...
        }
        Cambios[x] += Cambios[i]; // Se almacenan los cambios hechos en las columnas...
        Cambios[i] = Cambios[x]-Cambios[i];
        Cambios[x] -= Cambios[i];
      }
         // Una vez que se ha encontrado el mayor se realizan las sumas...

      for(j=0;j<Size;j++) // Se va "reduciendo"...
```

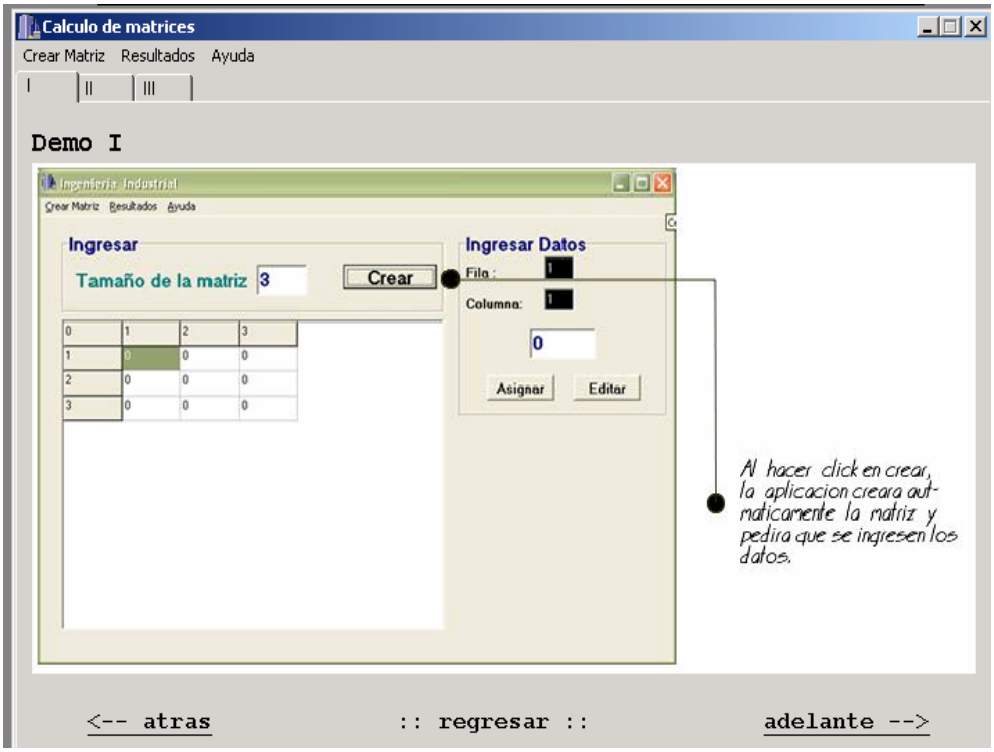
```
          {
            if(i==j) // Si es la fila en la que se est  trabajando...
            {
              temp = ((TipoDato)1)/(invertir.Matriz[j][j]); // invertir.Matriz[j][j] !=0. Asegurado...
              for(k=0;k<Size;k++)
              { // Se multiplica la fila por (1/Mayor) de manera que identidad[i][i]==1...
                identidad.Matriz[j][k] = identidad.Matriz[j][k]*temp;
                invertir.Matriz[j][k] = invertir.Matriz[j][k]*temp;
              }
            }
            else // Si no es la fila en la que se est  trabajando...
            {
              temp = invertir.Matriz[j][i]/invertir.Matriz[i][i];
              for(k=0;k<Size;k++) // Para cada elemento de la fila...
              {
                identidad.Matriz[j][k] = identidad.Matriz[j][k]-temp*identidad.Matriz[i][k];
                invertir.Matriz[j][k] = invertir.Matriz[j][k]-temp*invertir.Matriz[i][k];
              }
            }
          }
        }
        for(i=0;i<Size;i++) // Se deshacen los cambios...
        {
          if(Cambios[i]!=i) // Si se ha cambiado la columna i...
          {
            for(j=i;j<Size&&Cambios[j]!=i;j++)
              ;
            buffer = identidad.Matriz[j]; // Se cambian las filas en la matriz ya "casi invertida"...
            identidad.Matriz[j] = identidad.Matriz[i];
            identidad.Matriz[i] = buffer;
            Cambios[i] = Cambios[i]+Cambios[j]; // y se marca como deshecho el cambio...
            Cambios[j] = Cambios[i]-Cambios[j];
            Cambios[i] = Cambios[i]-Cambios[j];
          }
        }
      }
    }
    delete Cambios;
    op2=identidad;
}
/*****************************************************************************/
```

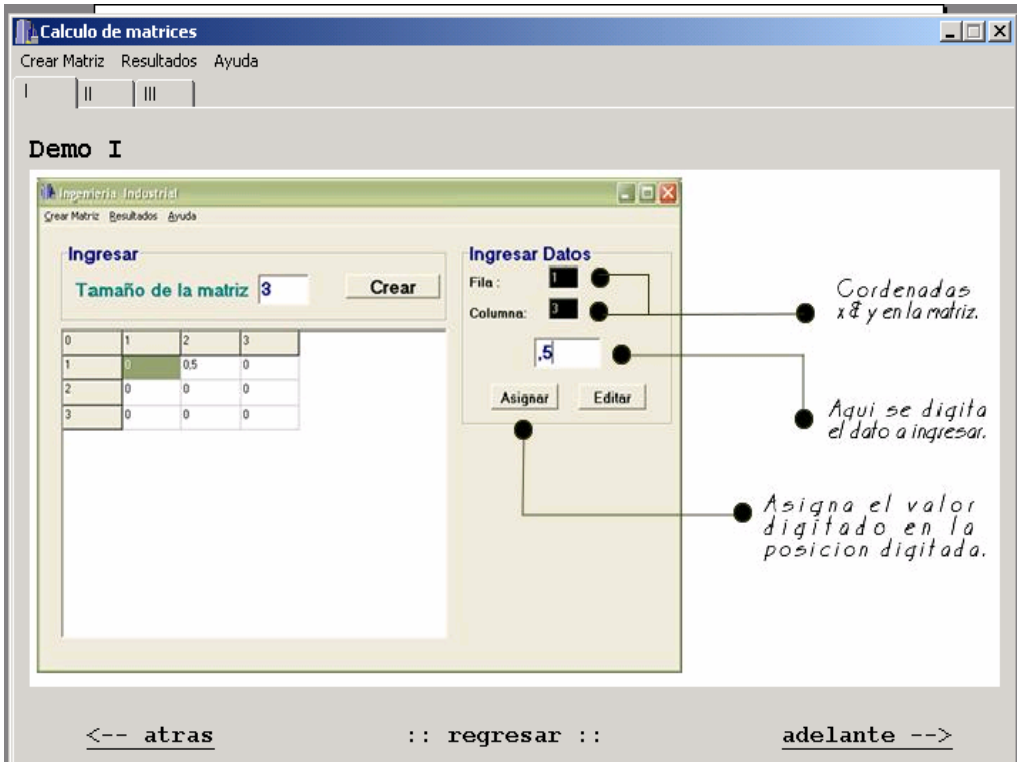
ANEXO E DEMOS (EJERCICIOS DE APLICACIÓN)

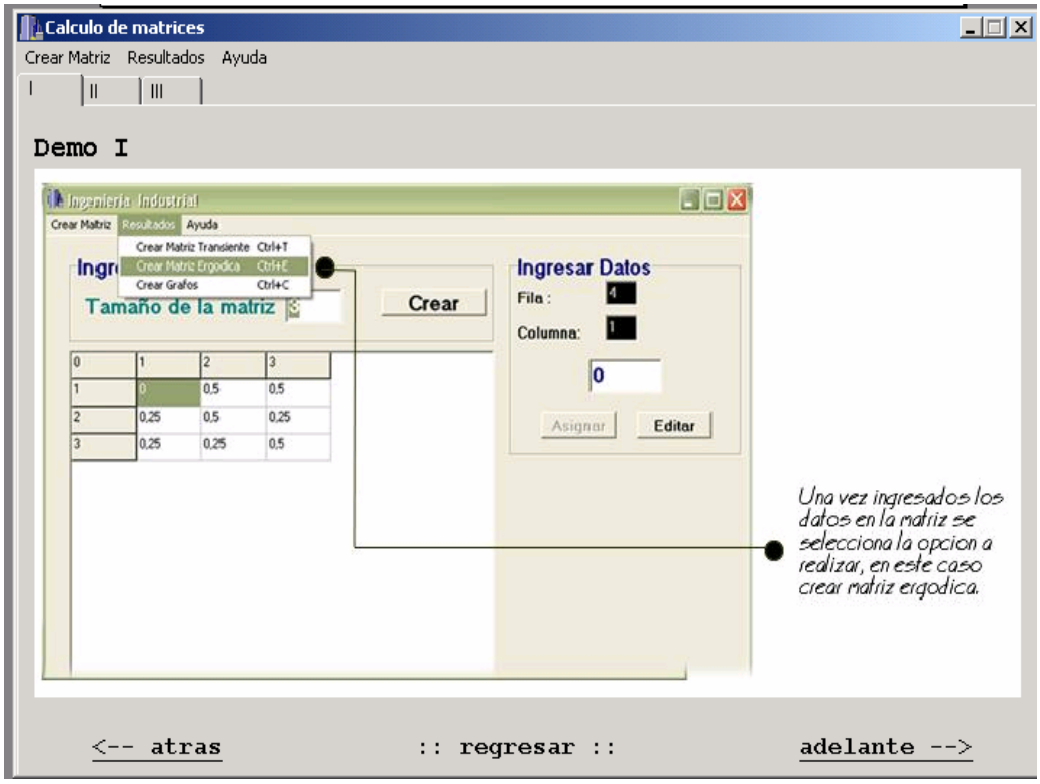
DEMO # 1 (Solución para Matrices Ergódicas)

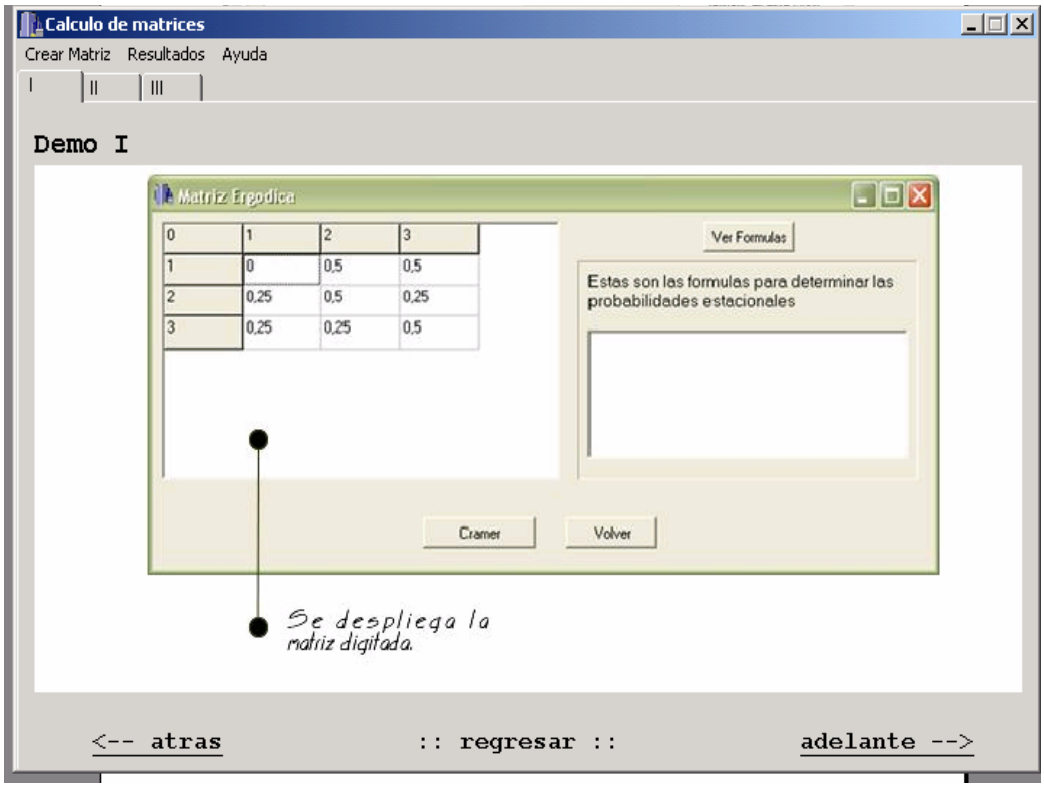


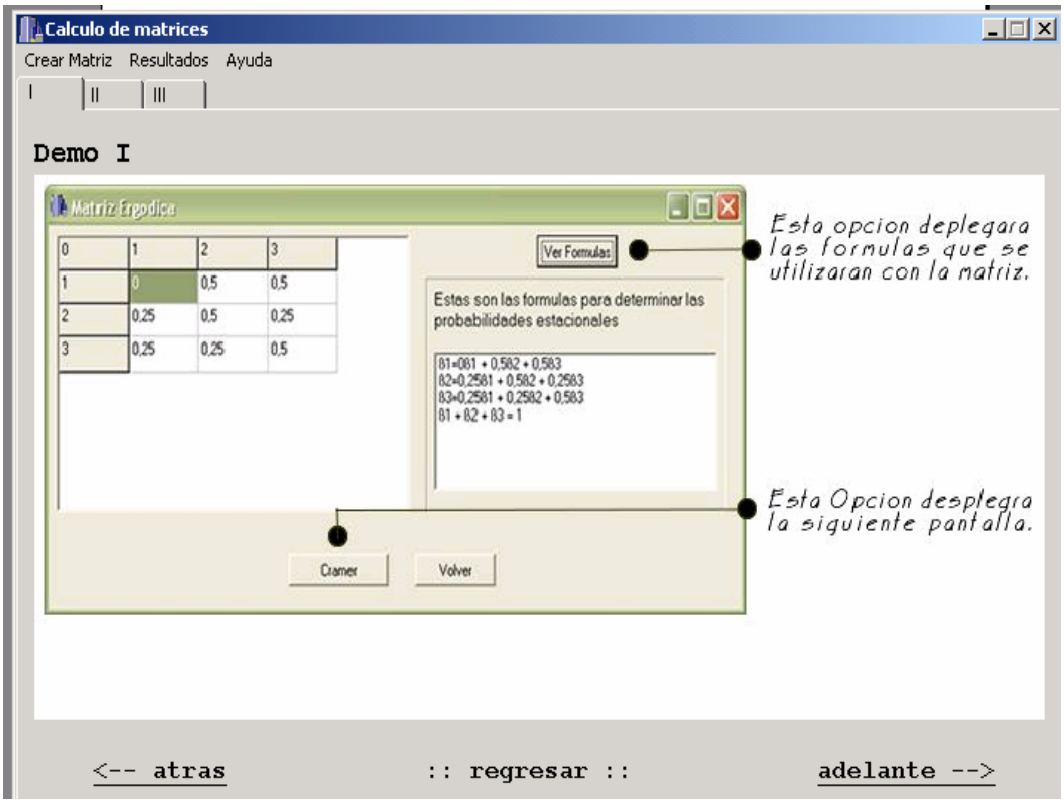


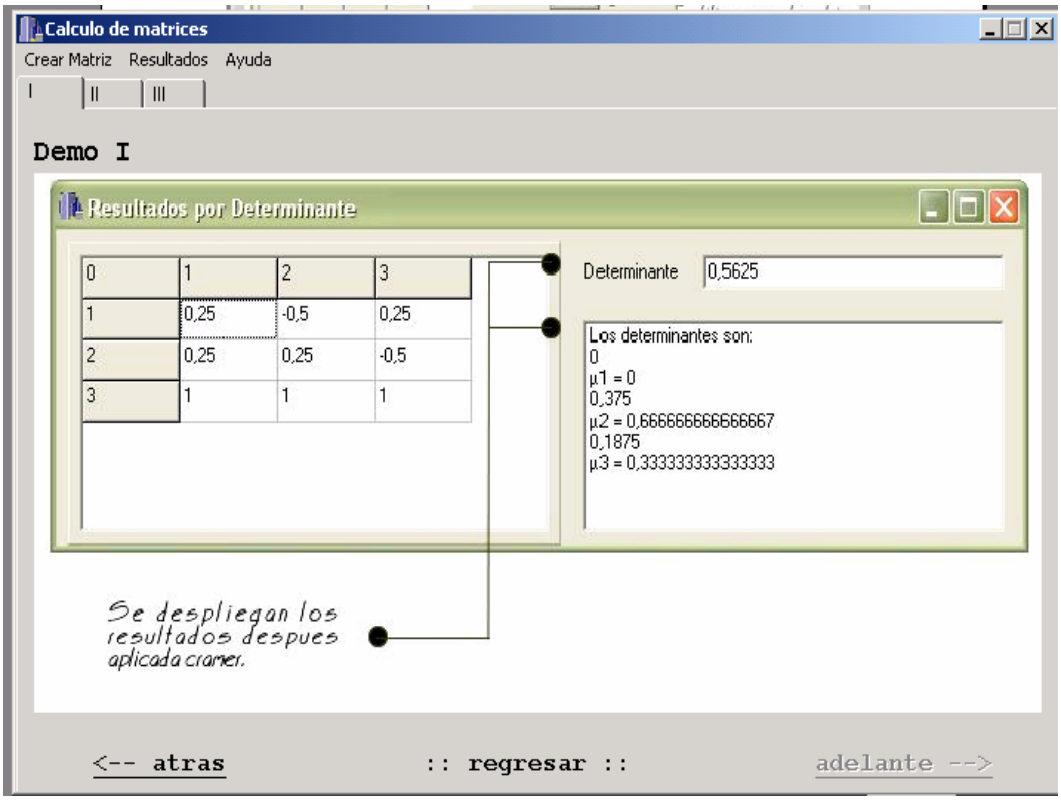

Calculo de matrices
Crear Matriz Resultados Ayuda
I II III
Demo I
Ingenieria Industrial
Crear Matriz Resultados Ayuda
Ingresar
Tamaño de la matriz 3
Crear
Ingresar Datos
Fila: 1
Columna: 1
0
Asignar
Editar
Al hacer click en crear, la aplicacion creara automaticamente la matriz y pedira que se ingresen los datos.
<-- atras
:: regresar ::
adelante -->


Calculo de matrices
Crear Matriz Resultados Ayuda
I II III
Demo I
Ingenieria Industrial
Crear Matriz Resultados Ayuda
Ingresar
Tamaño de la matriz 3
Crear
Ingresar Datos
Fila: 1
Columna: 3
.5
Asignar
Editar
Cordenadas x & y en la matriz.
Aqui se digita el dato a ingresar.
Asigna el valor digitado en la posicion digitada.
<-- atras
:: regresar ::
adelante -->

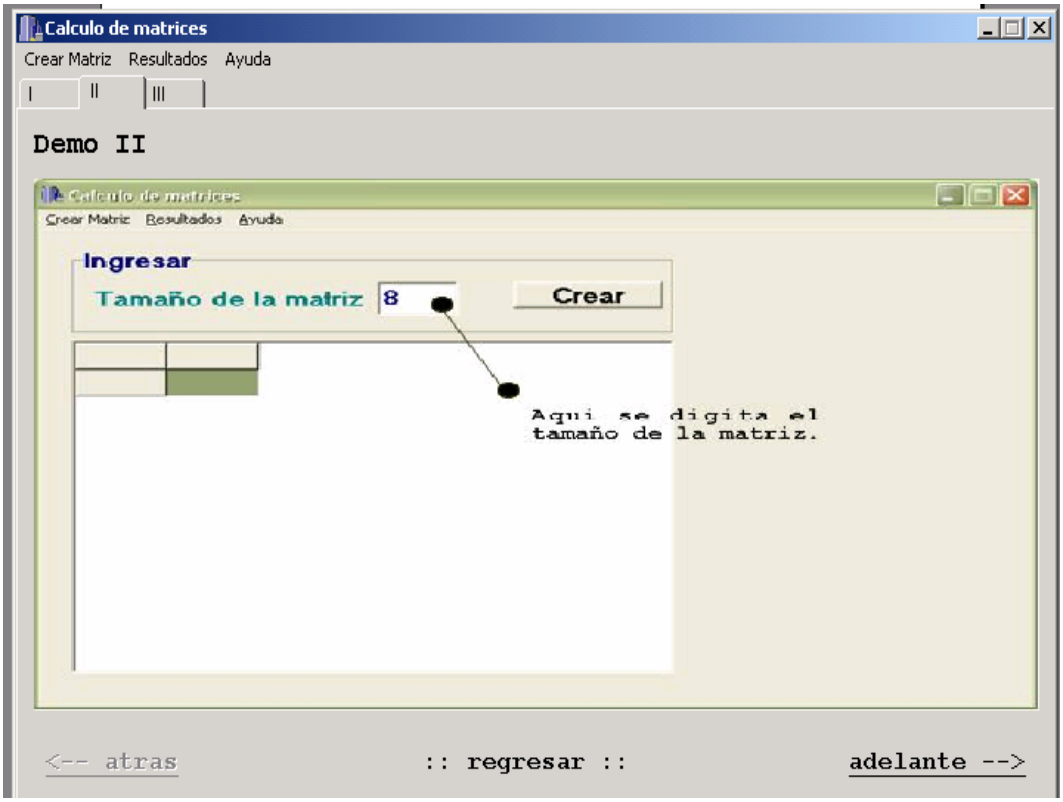

Calculo de matrices
Crear Matriz
Resultados
Ayuda
I
II
III
Demo I
Ingeniería Industrial
Crear Matriz
Resultados
Ayuda
Crear Matriz Transiente Ctrl+T
Crear Matriz Ergodica Ctrl+E
Crear Grafos Ctrl+C
Tamaño de la matriz
Crear
Ingresar Datos
Fila :
Columna:
Asignar
Editar
Una vez ingresados los datos en la matriz se selecciona la opcion a realizar, en este caso crear matriz ergodica.
<-- atras
:: regresar ::
adelante -->

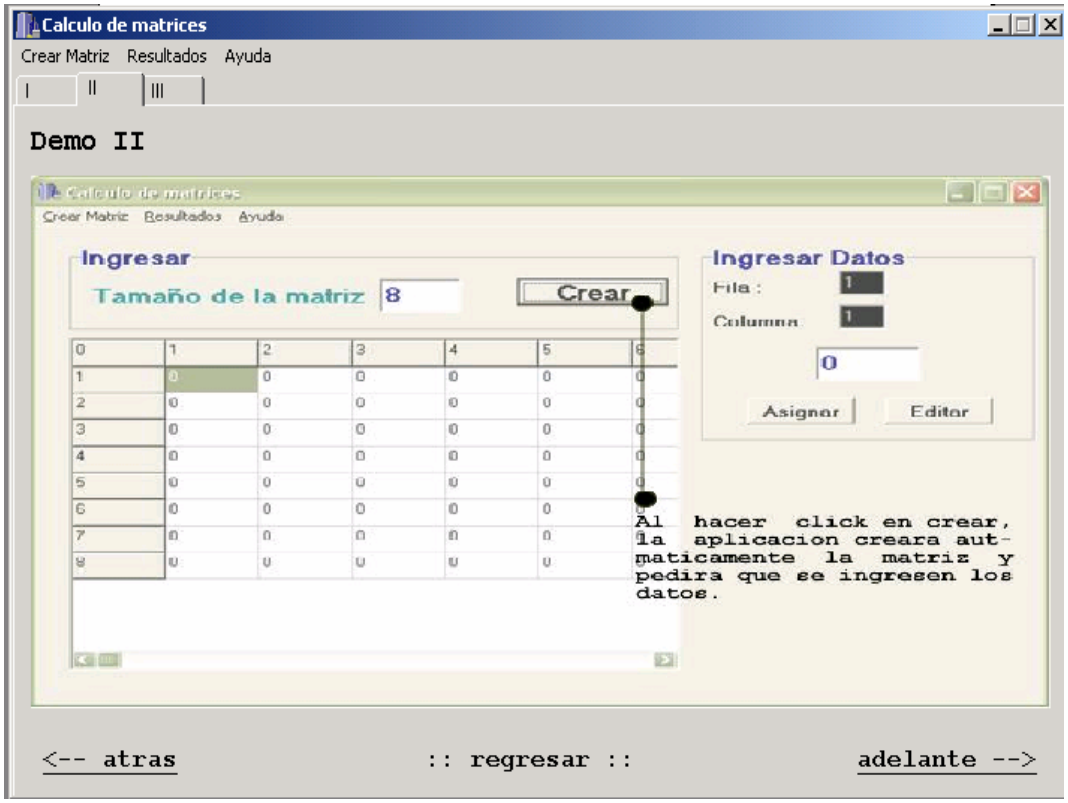

Calculo de matrices
Crear Matriz
Resultados
Ayuda
I
II
III
Demo I
Matriz Ergodica
Ver Formulas
Estas son las formulas para determinar las probabilidades estacionales
Cramer
Volver
Se despliega la matriz digitada.
<-- atras
:: regresar ::
adelante -->

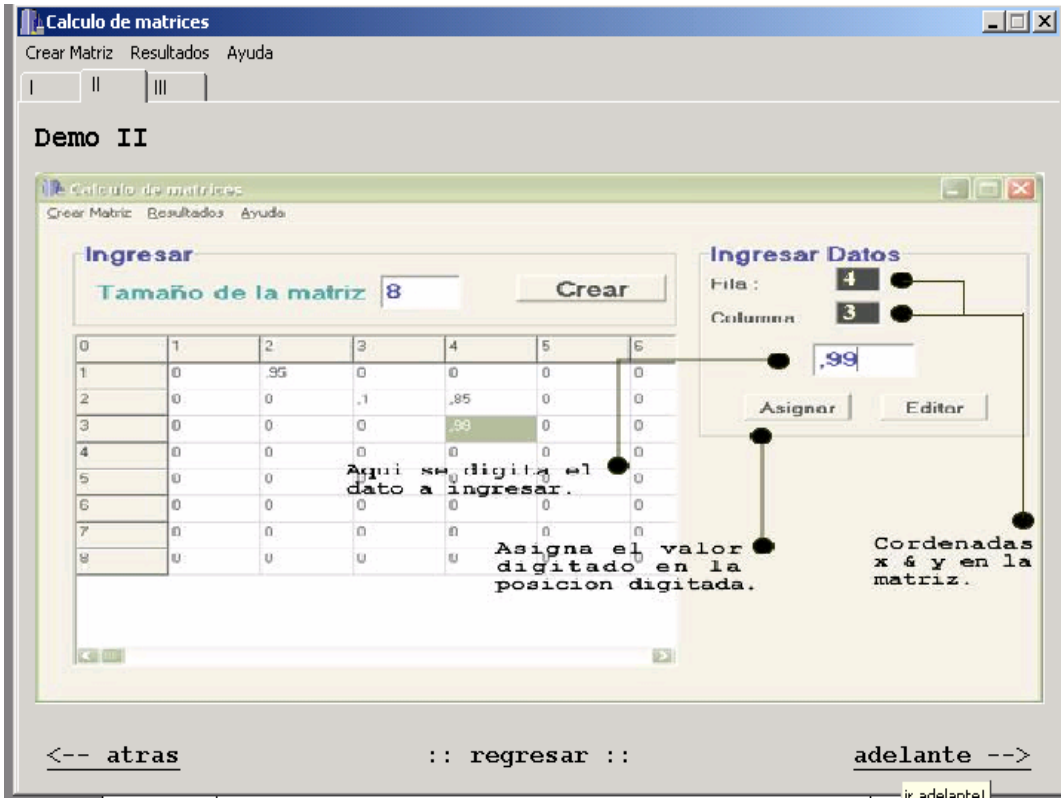

Calculo de matrices
Crear Matriz
Resultados
Ayuda
I
II
III
Demo I
Matriz Ergodica
Ver Formulas
Estas son las formulas para determinar las probabilidades estacionales
Cramer
Volver
Esta opcion deplegara las formulas que se utilizaran con la matriz.
Esta Opcion desplegra la siguiente pantalla.
<-- atras
:: regresar ::
adelante -->

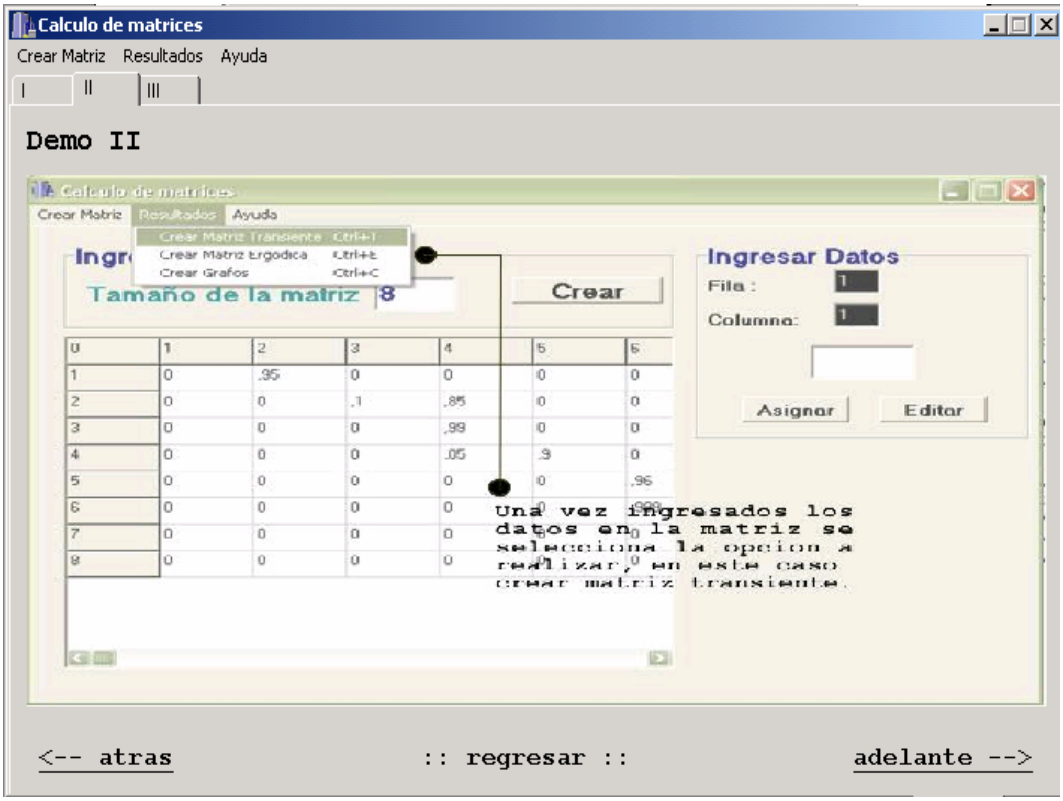

Calculo de matrices
Crear Matriz
Resultados
Ayuda
I
II
III
Demo I
Resultados por Determinante
Determinante
0,5625
Los determinantes son:
Se despliegan los resultados despues aplicada cramer.
<-- atras
:: regresar ::
adelante -->

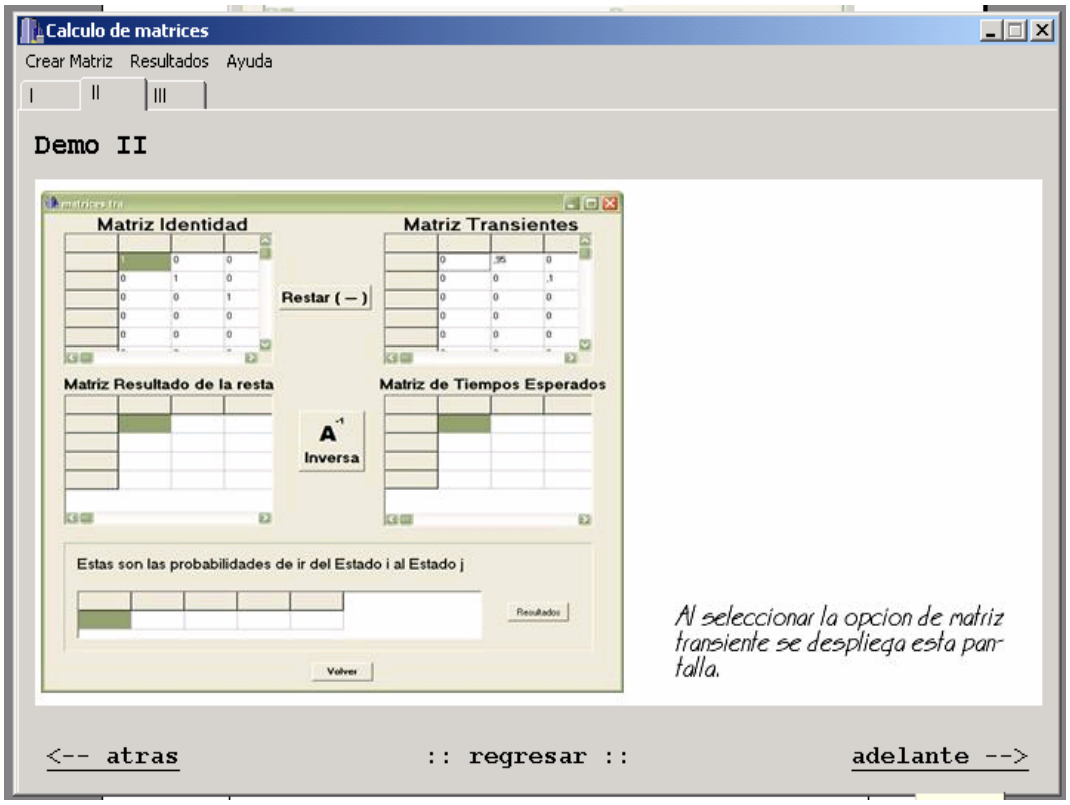
DEMO # 2 (Solución para Matrices Transientes)

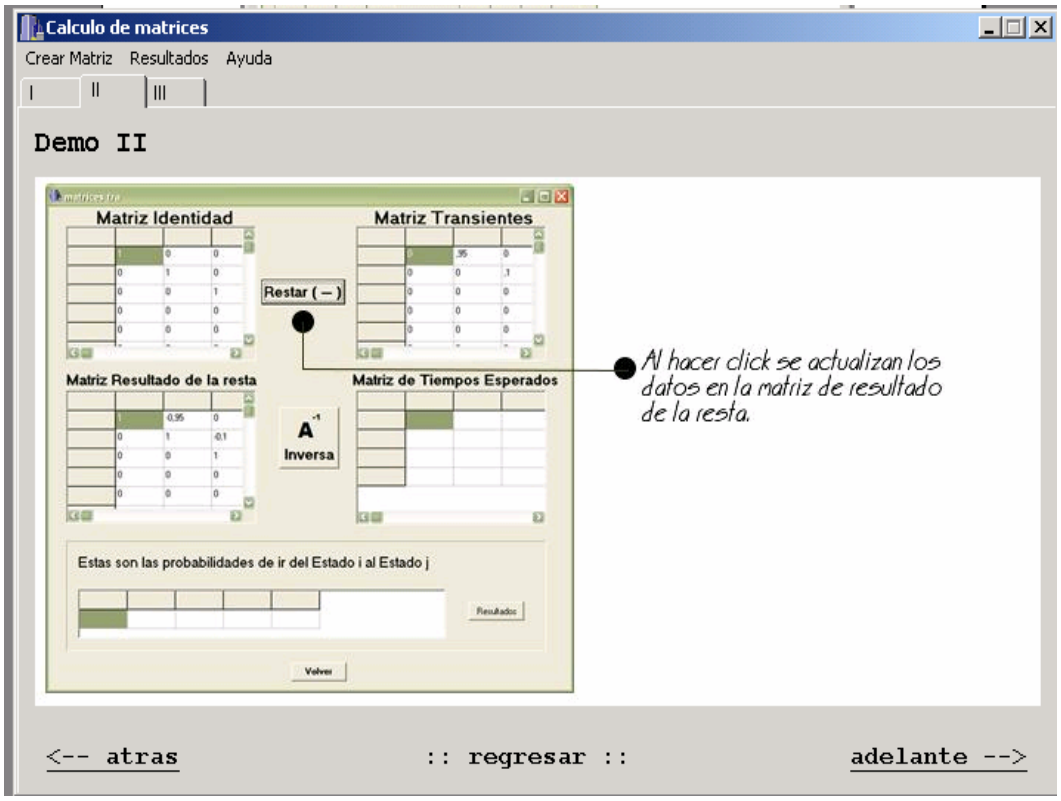


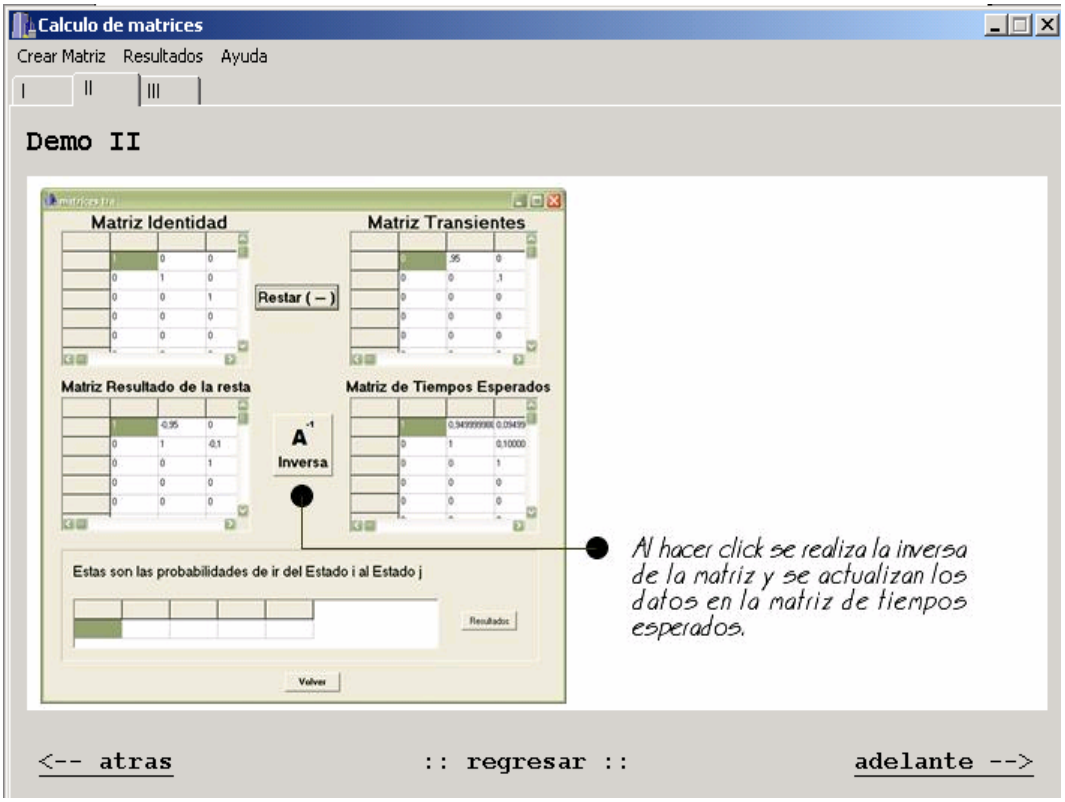


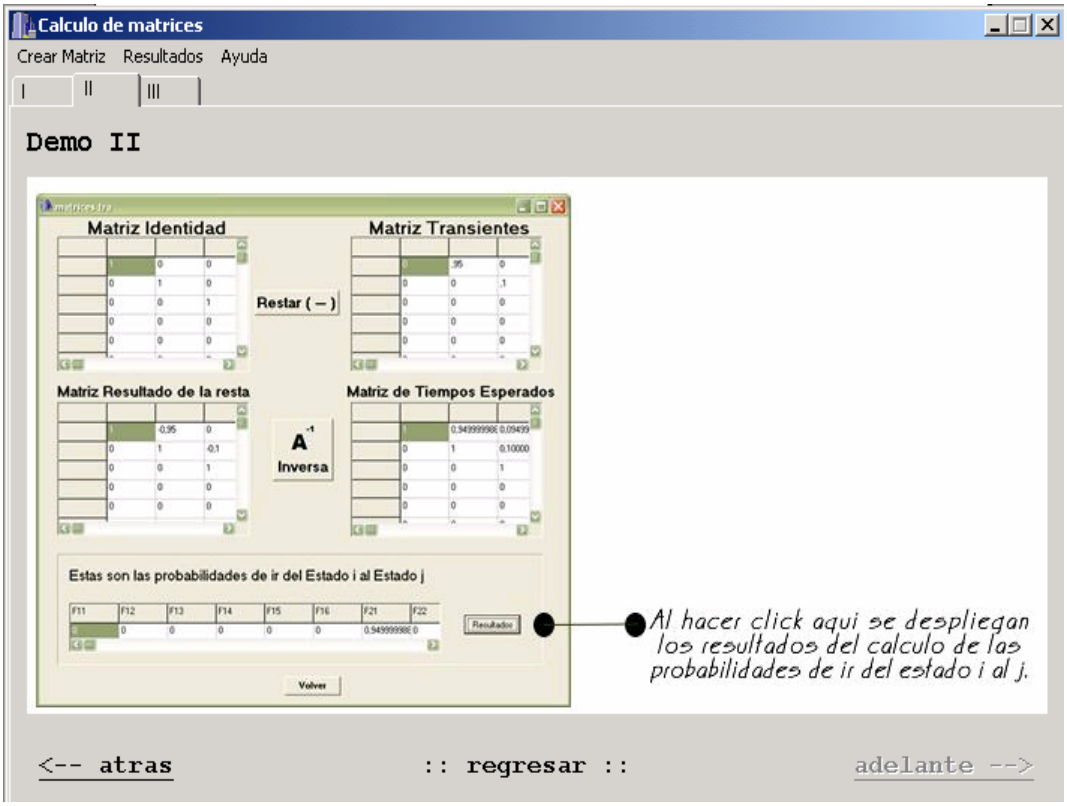

Calculo de matrices
Crear Matriz Resultados Ayuda
Demo II
Ingresar
Tamaño de la matriz 8
Crear
Ingresar Datos
Fila : 4
Columna 3
,99
Asignar
Editar
Aqui se digita el dato a ingresar.
Asigna el valor digitado en la posicion digitada.
Cordenadas x & y en la matriz.
<-- atras
:: regresar ::
adelante -->

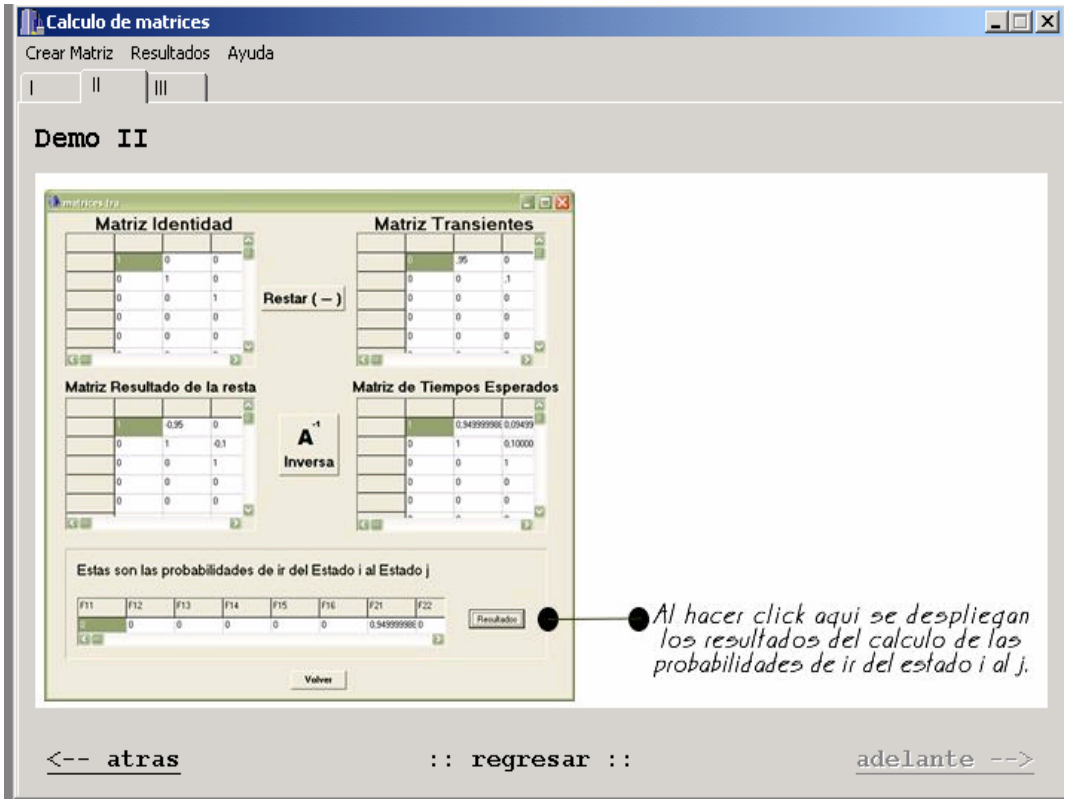

Calculo de matrices
Crear Matriz Resultados Ayuda
Demo II
Crear Matriz Transiente Ctrl+T
Crear Matriz Ergodica Ctrl+E
Crear Grafos Ctrl+C
Tamaño de la matriz 8
Crear
Ingresar Datos
Fila : 1
Columna: 1
Asignar
Editar
Una vez ingresados los datos en la matriz se selecciona la opcion a realizar, en este caso crear matriz transiente.
<-- atras
:: regresar ::
adelante -->


Calculo de matrices
Crear Matriz
Resultados
Ayuda
I
II
III
Demo II
Matriz Identidad
Matriz Transientes
Restar (–)
Matriz Resultado de la resta
Matriz de Tiempos Esperados
A-1
Inversa
Estas son las probabilidades de ir del Estado i al Estado j
Resultados
Volver
Al seleccionar la opcion de matriz transiente se despliega esta pantalla.
<-- atras
:: regresar ::
adelante -->


Calculo de matrices
Crear Matriz
Resultados
Ayuda
I
II
III
Demo II
Matriz Identidad
Matriz Transientes
Restar (–)
Matriz Resultado de la resta
Matriz de Tiempos Esperados
A-1
Inversa
Estas son las probabilidades de ir del Estado i al Estado j
Resultados
Volver
Al hacer click se actualizan los datos en la matriz de resultado de la resta.
<-- atras
:: regresar ::
adelante -->


Calculo de matrices
Crear Matriz Resultados Ayuda
I II III
Demo II
Matriz Identidad
Matriz Transientes
Restar (–)
Matriz Resultado de la resta
Matriz de Tiempos Esperados
A-1
Inversa
Estas son las probabilidades de ir del Estado i al Estado j
Resultados
Volver
Al hacer click se realiza la inversa de la matriz y se actualizan los datos en la matriz de tiempos esperados.
<-- atras
:: regresar ::
adelante -->


Calculo de matrices
Crear Matriz Resultados Ayuda
I II III
Demo II
Matriz Identidad
Matriz Transientes
Restar (–)
Matriz Resultado de la resta
Matriz de Tiempos Esperados
A-1
Inversa
Estas son las probabilidades de ir del Estado i al Estado j
Resultados
Volver
Al hacer click aqui se despliegan los resultados del calculo de las probabilidades de ir del estado i al j.
<-- atras
:: regresar ::
adelante -->

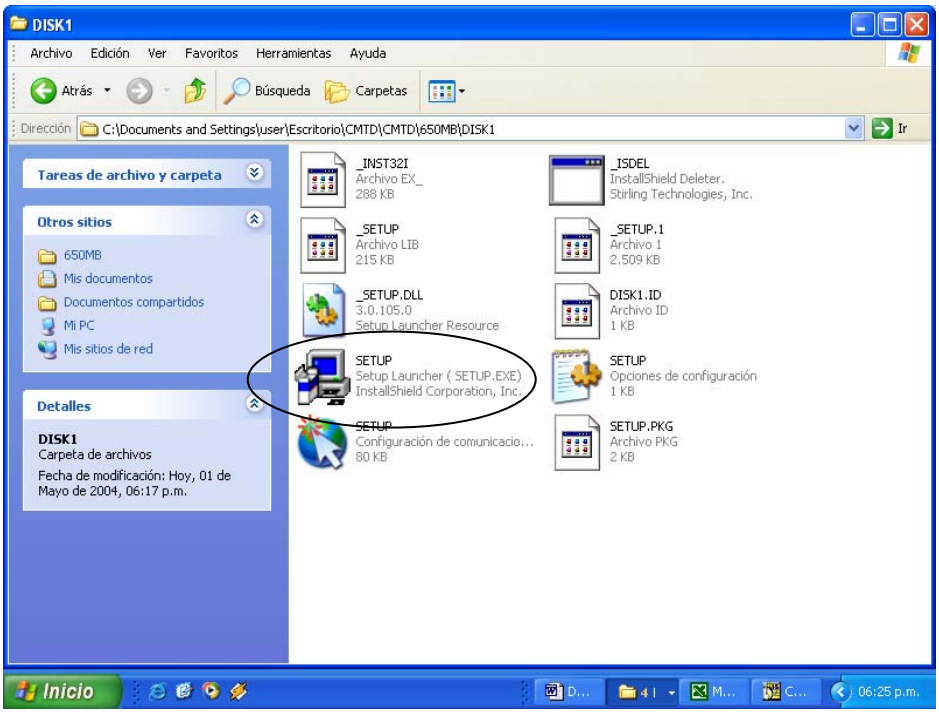

Calculo de matrices
Crear Matriz
Resultados
Ayuda
I
II
III
Demo II
Matriz Identidad
Matriz Transientes
Restar (–)
Matriz Resultado de la resta
Matriz de Tiempos Esperados
A-1
Inversa
Estas son las probabilidades de ir del Estado i al Estado j
F11
F12
F13
F14
F15
F16
F21
F22
Resultados
Volver
Al hacer click aqui se despliegan los resultados del calculo de las probabilidades de ir del estado i al j.
<-- atras
:: regresar ::
adelante -->

ANEXO F MANUAL INSTRUCTIVO DE USO

TABLA DE CONTENIDO MANUAL DE USUARIO

Proceso de instalación

1. Introducir el CD en la unidad.
2. Copiar la carpeta “(CMTD)” en el escritorio de Windows.
3. Ingresar a la carpeta y dentro de ella, Abrir la carpeta “CMTD”.
4. Seguidamente abrir la carpeta “650 MB”
5. ingresar a la carpeta “DISK 1” y ejecutar el Setup



6. Seguir las intrucciones del Setup.

Nota: no deben borrarse de la carpeta los archivos con extensión dll.

BIENVENIDOS!

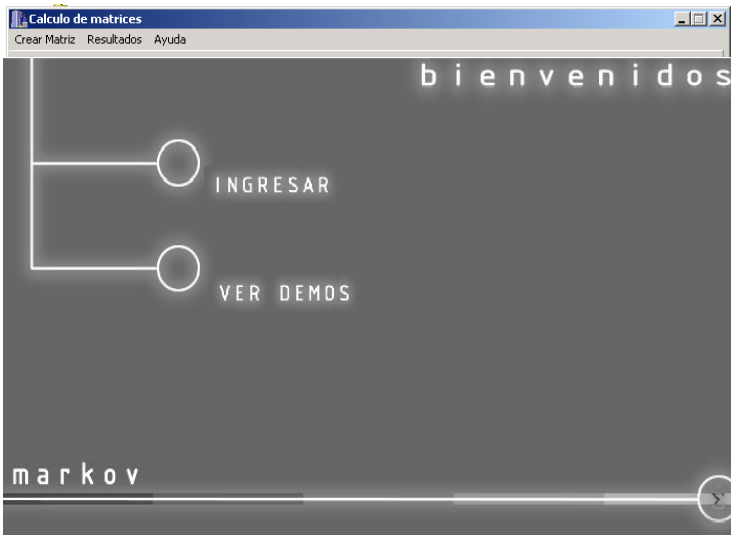
Transiente

Figura # 1

Para ejecutar el programa, el usuario debe ubicarse en el escritorio de Windows y pulsar “Inicio” posteriormente “programas” y buscar en el menú la carpeta CMTD y dar doble click al icono “CMarcCofv” ,. Ver figura 1.

BIENVENIDA DEL SISTEMA:

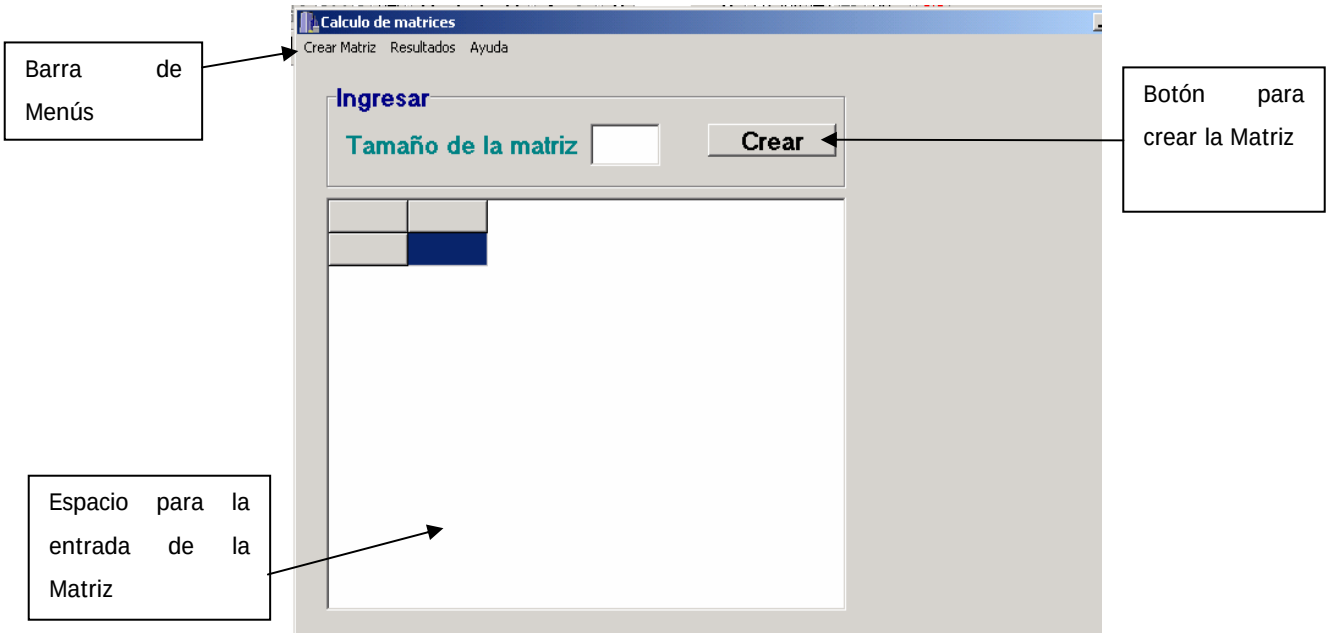
En la pantalla de inicio aparecerán 2 botones de opciones:



INGRESAR : Muestra a los estudiantes los tipos de ejercicios que se pueden realizar ene el programa.

VER DEMOS : Al hacer clic se entra a la interfaz principal del software educativo.

1 VENTANA : INTERFAZ

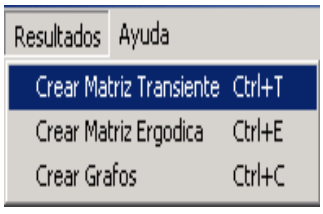


COMPONENTES DE LA INTERFAZ PRINCIPAL

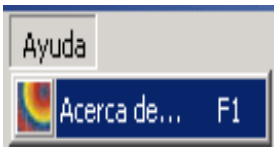
Los componentes de la interfaz principal son:

Barra de Menú: Permite el acceso a los siguientes submenús:

Submenú Crear Matriz: Este menú permite borrar el contenido de una matriz, y también salir de la aplicación.



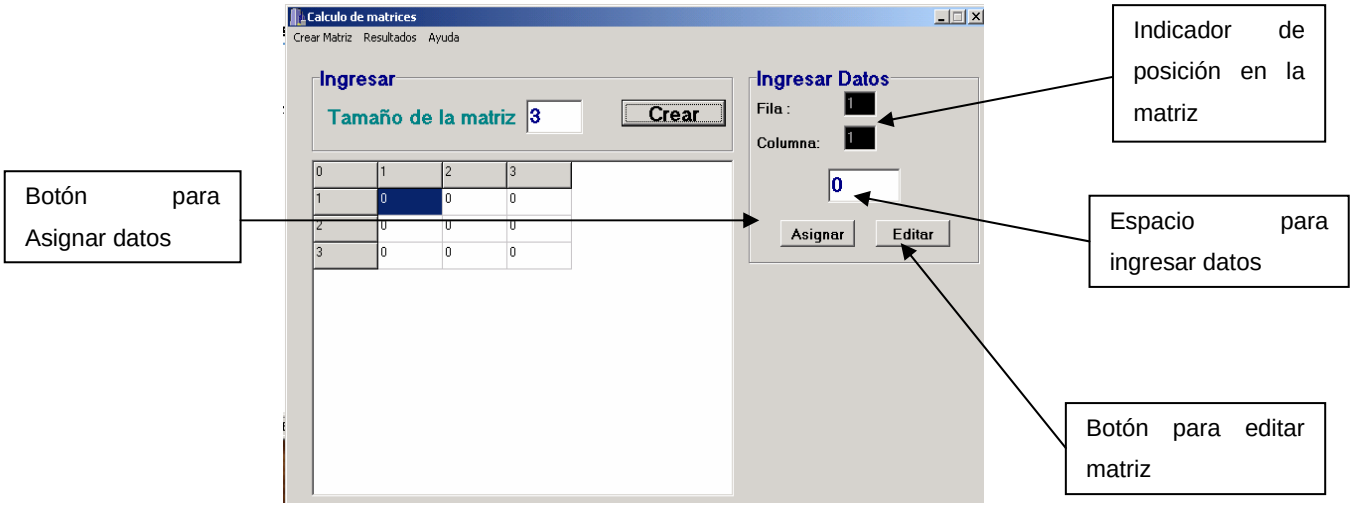
Submenú Resultados: Este menú permite seleccionar el tipo de resultados que se genera según el tipo de matriz, de igual forma tiene la opción de generar grafos. Cada ítem de este menú se explicará con más detalle más adelante



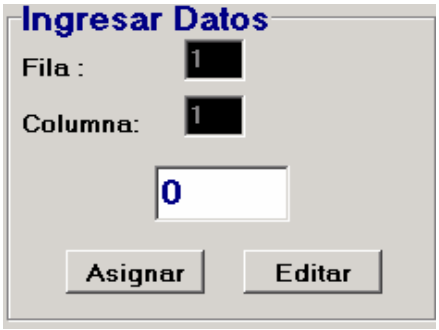
Submenú Ayuda: Permite al usuario retomar todos los conceptos relacionados con Cadenas de Markov en Tiempo Discreto (CMTD), este menú se explicará con más detalle más adelante.

.

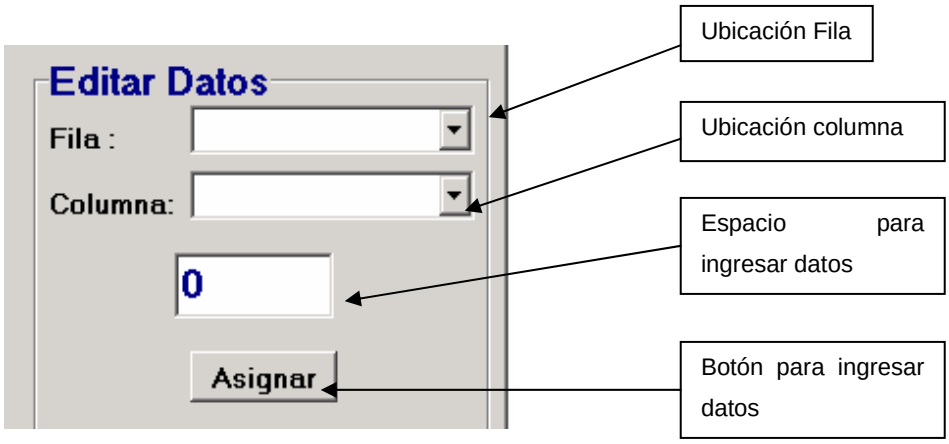
Botón Crear Matriz: Una vez se a presionado este botón se activa en la interfaz principal la opción de ingresar datos.



2.1) Ingresar Datos: Este submenú se dispuso para ingresar y modificar los datos de la matriz, todo dato que se ingrese debe ir con “coma” o con “punto”, en su defecto.

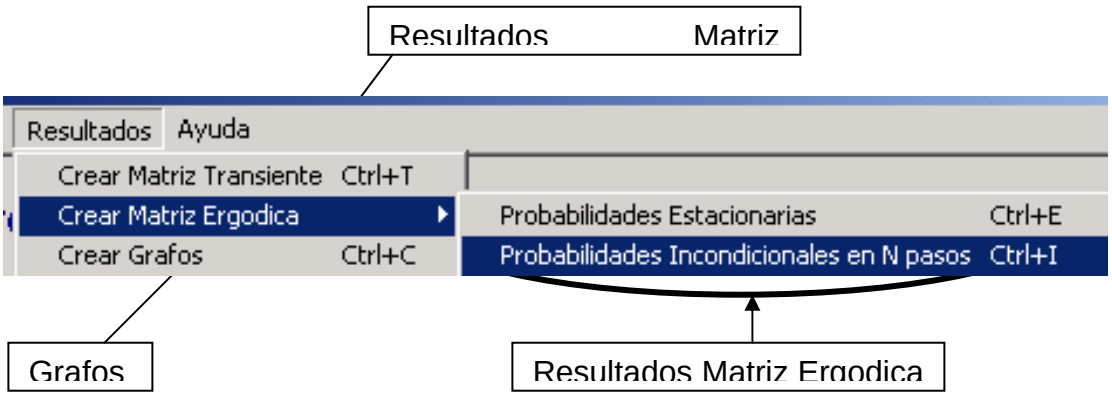


2.1.1) Botón Editar: Este botón una vez accionado despliega un submenú para editar los datos de la matriz, en donde se puede ubicar la posición del valor a editar por filas y columnas.



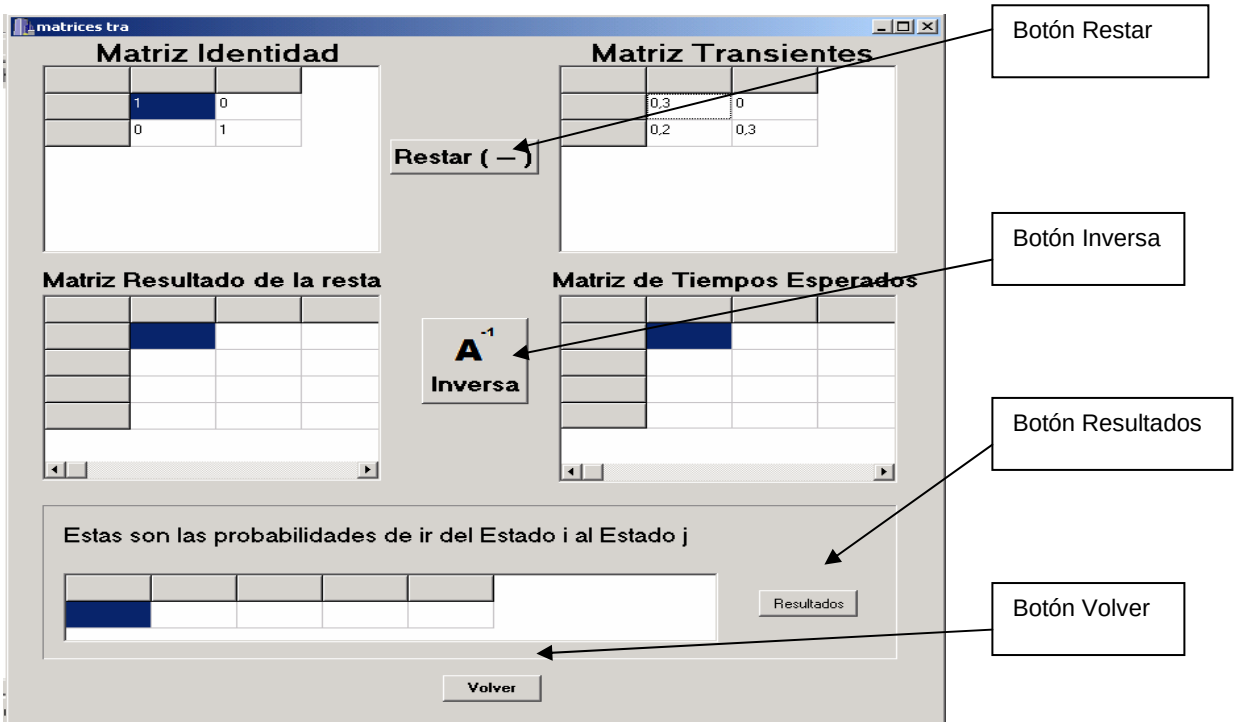
SUBMENÚ DE RESULTADOS

En el submenú de “Resultados” en la barra de Menús, el usuario puede generar dos tipos de resultados dependiendo del tipo de matriz, y de igual forma obteniendo el grafo para la matriz inicial.



Resultados Para Una Matriz Transiente

Se desplegará en la pantalla la siguiente ventana



El usuario solo debe oprimir los botones en el siguiente orden; Restar, Inversa y por ultimo resultados, si no lo hace de esta forma generará un error.

Restar (–) Este botón hace la resta entre la matriz identidad y la matriz de estados Transiente y genera la matriz “resultados de la resta”.

A-1 Inversa Este botón hace la inversa de la matriz de resultados de la resta y por medio de un algoritmo interno genera la matriz de Tiempos Esperados.

Resultados Este botón genera las probabilidades de ir del estado i al j. Introduciendo esta información en una tabla.

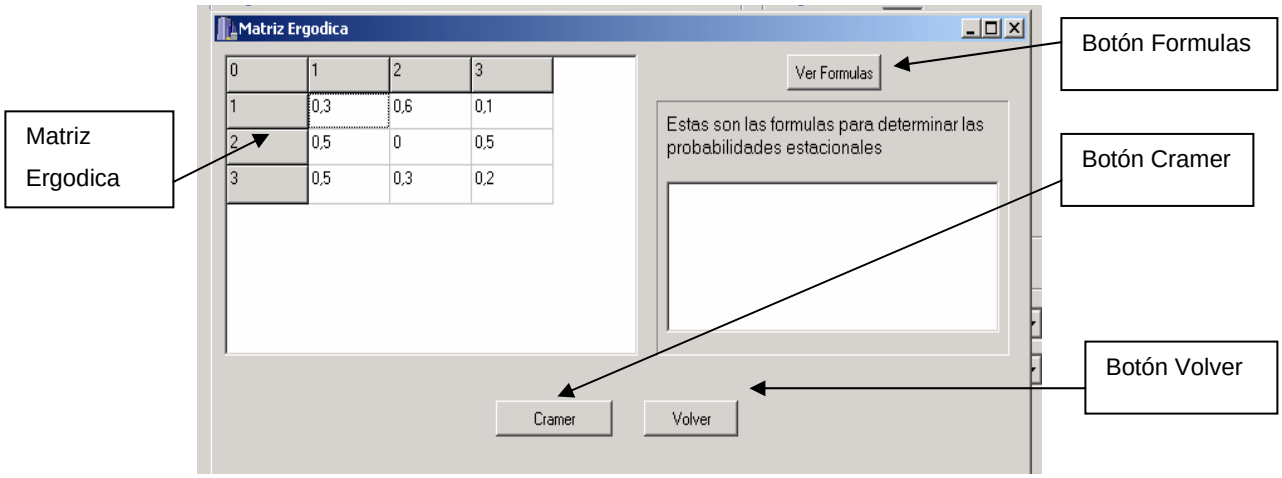
Volver Este Botón regresa al usuario a la interfaz principal sin hacer modificaciones a los datos.

Resultados Para Una Matriz Ergódica.

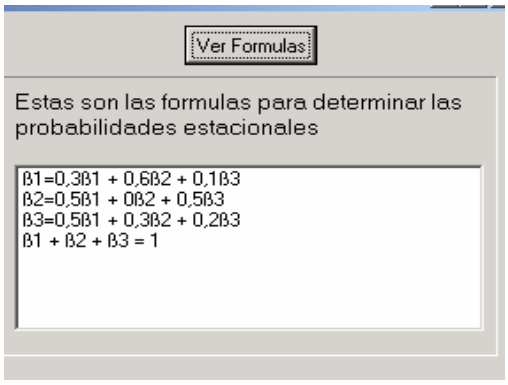
Este menú tiene dos posibles respuestas dependiendo de la elección del usuario.

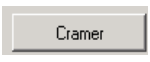
Probabilidades Estacionarias

Se desplegará en la pantalla la siguiente ventana

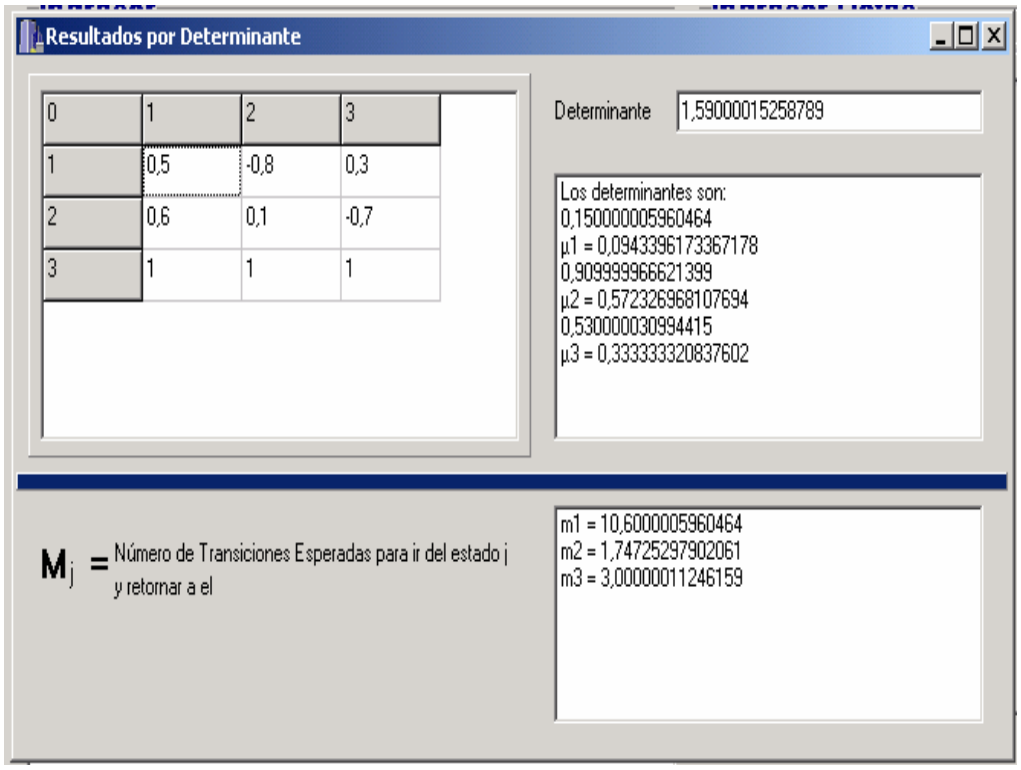


Este botón despliega en la tabla las formulas necesarias para que el usuario genere las probabilidades estacionarias





Este botón despliega la siguiente pantalla.

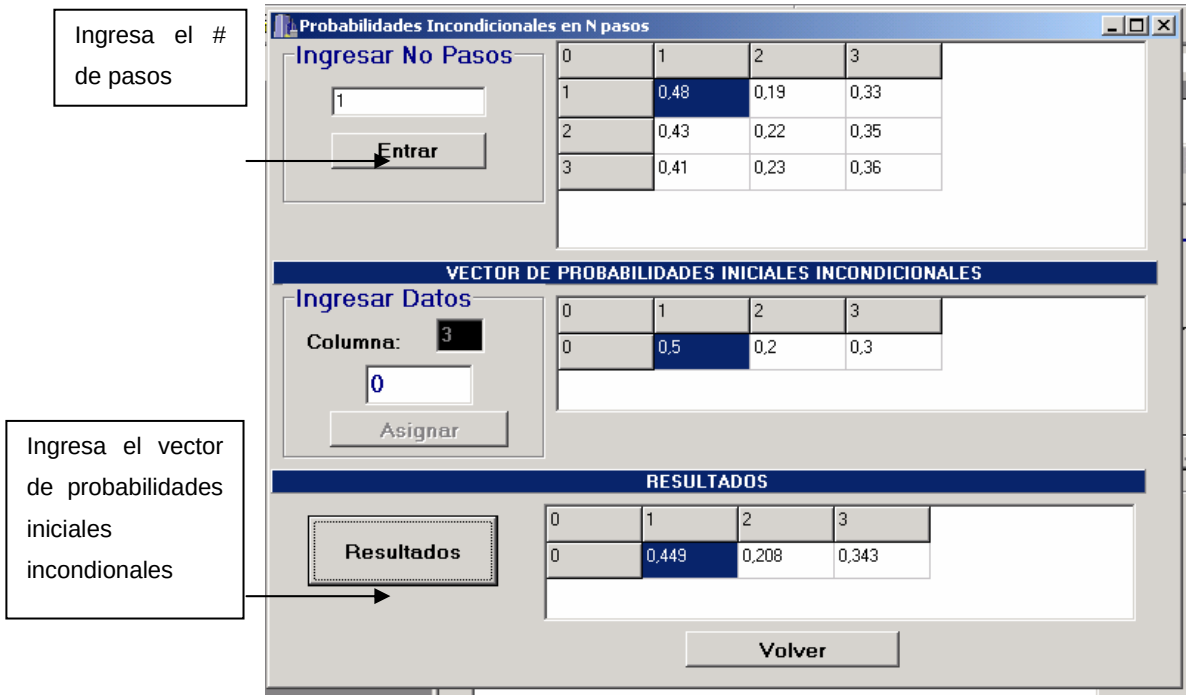


En donde el usuario encontrará los resultados de las formulas, por el método de Cramer, encontrando las probabilidades estacionarias y adicionalmente el numero de transiciones esperadas para ir al estado j y retornar a él.

Volver Este Botón regresa al usuario a la interfaz principal sin hacer modificaciones a los datos.

Probabilidades Incondicionales en N Pasos

Si el usuario escoge esta opción ingresara en la siguiente pantalla.

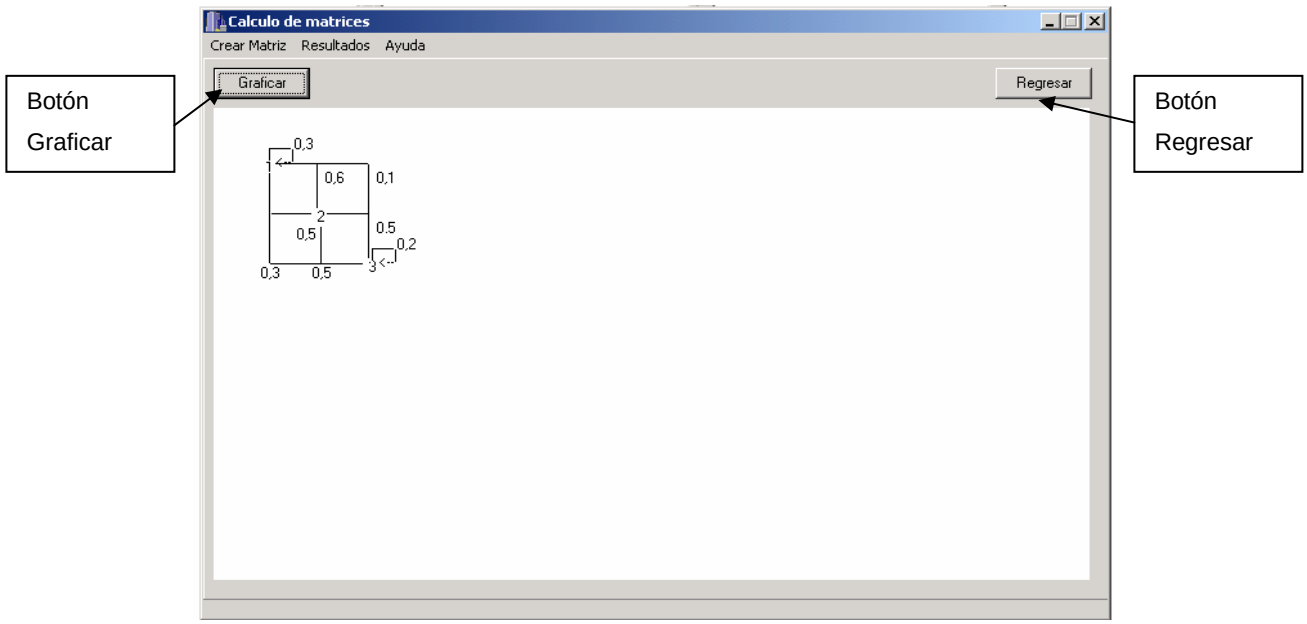


Este Botón muestra los resultados de las probabilidades

Este Botón regresa al usuario a la interfaz principal sin hacer modificaciones a los datos.

Crear Grafos

Se desplegará en la pantalla la siguiente ventana



Este Botón genera el grafo

Este Botón regresa al usuario a la interfaz principal sin hacer modificaciones a los datos.

SUBMENÚ DE AYUDA

En el submenú de “Ayuda” en la barra de Menús, el usuario puede ingresar a una nueva pagina de ayuda, esta ventana se presenta a continuación.

En esta ventana el usuario solo debe presionar con el mouse el link que desee ver, si dentro de la definición existen gráficos el usuario debe dar click en el link de “Ver Gráficos”, si desea volver a ver el texto debe dar click de nuevo al link de donde se encontraba.

ANEXO G EVALUACIÓN DEL SOFTWARE EDUCATIVO

EVALUACIÓN DEL SOFTWARE EDUCATIVO: CADENAS DE MARKOV EN TIEMPO DISCRETO

PROCESOS ESTOCÁSTICOS

1. NOMBRE:_____
2. TELEFONO:_____
3. SEXO: F_____ M_____
4. FECHA:_____

5. El Software educativo le brinda al estudiante la posibilidad de:

	Si	**No**
a. Interactuar con una interfaz amigable		
b. Reforzar imágenes con explicaciones escritas		
c. Controlar la cantidad de ejercicios		
d. Controlar cuando abandonar		
e. Controlar cuando reiniciar		

6. El profesor puede tener la posibilidad de:

	Si	**No**
a. Editar los ejercicios		

7. El software desarrolla los siguientes contenidos:

	Si	No
a. Análisis de estados (transientes, recurrentes y absorbentes)		
b. Probabilidades estacionarias		
c. Análisis del tipo de cadena (transiente y ergódica)		
d. Cadenas de estados transientes		
e. Tiempos esperados en estados transientes		
f. Probabilidades transientes		
g. Ecuaciones de Chapman – Kolmogorov		
h. Probabilidades de estados futuros		

8. El manual de usuario debe facilitar:

	Si	No
a. La forma de instalación		
b. El uso del software		