ANÁLISIS MATRICIAL DE ESTRUCTURAS CON SCILAB

Autor: Prof. Ing. José Luis Volta Grau *

Dirección: Valois Rivarola 828 esq. 9 de Setiembre
E-mail: jlvolta@rieder.net.py

Palabras claves: SCILAB, análisis matricial, aplicación informática, estructuras

RESUMEN

SCILAB es un software para cálculos matemáticos, de distribución libre y gratuita, poco conocido aún en nuestro medio científico-técnico, desarrollado en Francia por el "Consorcio Scilab" (INRIA/ENPC); puede bajarse desde el sitio http://www.scilab.org

Este trabajo tiene como propósito aportar información sobre las ventajas del uso de SCILAB para el desarrollo rápido de aplicaciones para cálculos técnicos que requieren el uso intensivo y extensivo del cálculo matricial.

En este sentido, el autor ha considerado interesante desarrollar una aplicación típica de cálculo matricial de estructuras, con el objetivo de evaluar las posibilidades y facilidades que ofrece SCILAB para técnicos con formación básica general en programación, pero con suficientes conocimientos de matemática (álgebra matricial, análisis, cálculos numéricos, gráficos, etc).

* Profesor Titular de la Cátedra de 'Estructuras II' y 'Estructuras III' de la Carrera de Ingeniería Civil de la Facultad de Ingeniería de la U.N.A.

1. Objetivos

El desarrollo de aplicaciones informáticas ("software") para cálculos de ingeniería, realizados por y para uso de los ingenieros en sus gabinetes de proyecto, ha sido tradicionalmente enfocado con planillas electrónicas.

Las ventajas de estas planillas para organizar los cálculos numéricos son más que conocidas. Además de contar con gran cantidad de funciones matemáticas, las planillas también permiten operar sobre rangos de celdas como si fueran matrices numéricas.

Sin embargo, la programación de cálculos matriciales para un número grande de matrices con muchos elementos resulta prácticamente imposible con el sólo uso de tales planillas.

En casi todo cálculo técnico de ingeniería se requiere, en algún punto, procesar matrices numéricas, dado que las magnitudes vectoriales son usuales en la mayoría de los fenómenos físicos y en otros campos.

2. Características de SCILAB

Cuales son las características de Scilab que facilitan el trabajo de programación a los ingenieros y técnicos sin preparación especial en informática ?

Sin intentar ninguna comparación con otros entornos de programación, comerciales o no, las características de Scilab más importantes que puedo mencionar brevemente son, a mi juicio:

a) su entorno integrado de trabajo, que es además, interactivo, los que facilita la depuración (mediante prueba y error, inspección de valores calculados, etc) sin necesidad de recurrir al depurador (aunque este está disponible);

b) puede usarse el propio entorno integrado de trabajo como interfaz de usuario colocando un menú en la barra de menús, usando funciones de cuadros de diálogos, etc; no es preciso conocer la programación orientada a objetos y guiada por eventos como lo requiere la construcción de una interfaz de usuario en otros entornos de programación;

c) las funciones de una aplicación pueden compilarse en una biblioteca de funciones, y ésta puede cargarse automáticamente al iniciar Scilab;

d) toda variable es considerada una matriz de algún tipo (número, texto, boolean, etc), el cual se define al asignarse el primer valor a cualquier elemento de la misma;

e) todas las matrices son dinámicas (su tamaño se define en tiempo de ejecución y pueden crecer automáticamente cuando se requiera, simplemente asignando nuevos elementos); como los datos se guardan en la memoria masiva, el límite de tamaño de un sistema de matrices sólo está limitado por la memoria RAM disponible;

f) los argumentos pasados a las funciones son matrices, así como los valores devueltos por éstas; el paso predeterminado es por valor, pero es posible hacerlo por referencia;

g) una función puede recibir y devolver una cantidad opcional de argumentos (usando los parámetros "varargin" y "varargout"); lo cual es una buena muestra de la gran flexibilidad de la estructura del lenguaje de Scilab;

h) pueden definirse variables "globales", accesibles a cualquier función de la aplicación, con lo que puede evitarse la necesidad de paso de argumentos a funciones;

i) Scilab aporta funciones para leer directamente datos contenidos en libros de Excel (en Windows); (aunque no pueden grabarse datos en libros de Excel); es innecesario recalcar la importancia de esta característica para los cálculos técnicos;

j) las operaciones algebraicas con matrices se realizan como si fueran variables de tipo simple; Scilab aporta muchas funciones para operaciones matriciales y vectoriales;

k) Scilab agrega funciones para la solución de sistemas de ecuaciones lineales cuyas matrices son simétricas, de banda y "esparcidas" (solamente se requiere memoria para sus elementos no-nulos, con un considerable ahorro de memoria y

de tiempo de procesamiento); todas las matrices del análisis matricial de estructuras son de este tipo;

l) Scilab aporta una extensa biblioteca de funciones para realizar gráficos;

m) Es posible agregar ayuda en línea para las aplicaciones desarrolladas con Scilab, usando para esto páginas de hipertexto que se acceden con el sistema de ayuda del mismo entorno;

n) El lenguaje y el entorno de programación son parecidas al Basic, por mucho tiempo el lenguaje de programación de propósito general preferido por los no profesionales en informática, pero con un nivel superior en muchos sentidos.

3. Las funciones de una aplicación

En SCILAB, una aplicación consiste en un conjunto de funciones que forman una biblioteca. Es posible cargar automáticamente las funciones de nuestra aplicación y usarlas con sólo teclear su nombre.

Al arrancar, SCILAB ejecuta automáticamente, si lo hemos creado en nuestro directorio de usuario, el archivo de texto 'scilab.ini', que contiene los nombres de las bibliotecas compiladas de funciones de nuestra aplicación.

Por ejemplo, el archivo 'scilab.ini' necesario para cargar el programa 'Estructuras', con tres funciones compiladas ("lib"), contiene las siguientes líneas de texto simple:

// Programa para análisis y dimensionamiento estructural con Scilab
// cargando las bibliotecas de funciones compiladas
load(SCIHOME+'\Estructuras\Análisis\lib');
load(SCIHOME+'\Estructuras\Dimensionamiento\lib');
load(SCIHOME+'\Estructuras\lib');

SCIHOME es una variable de entorno donde Scilab guarda el "path" del directorio del usuario actual.

4. Interfaz de usuario de una aplicación

El entorno de Scilab permite agregar un menú (o varios, según se requiera) a su barra de menús, a fin de que la aplicación ofrezca una forma amigable de uso.

Igualmente es posible usar cuadros de diálogo de varias clases, indispensable en muchos casos.

Para agregar un menú se usa la función '**addmenu**'; usualmente se colocará esta función dentro de la función inicial de la aplicación.

Las 2 líneas de código siguientes, contenidas en la función 'Estructuras', crea el menú de nombre 'Analizar', con varios comandos para realizar el análisis de estructuras:

comandos=['Nuevo archivo de datos…','Abrir archivo de datos…','(otros comandos)'];
addmenu('Analizar',comandos,list(2,'Analisis'));

El primer argumento de la función 'addmenu' es el nombre del menú que aparece en la barra de menús; el segundo argumento ('comandos') es un vector fila conteniendo los nombres de los comandos del menú; el tercer argumento crea una lista para capturar los eventos del menú: en este caso, el 2 significa que los procedimientos de eventos estarán en una función, y 'Analisis' es el nombre de dicha función.

Usualmente agregamos un cuadro de diálogo o de bienvenida inicial, mostrando nombre de la aplicación, número de versión, fecha, nombre(s) de autor(es), etc. (las figuras 1 y 2 ilustran el proceso para la aplicación 'Estructuras')

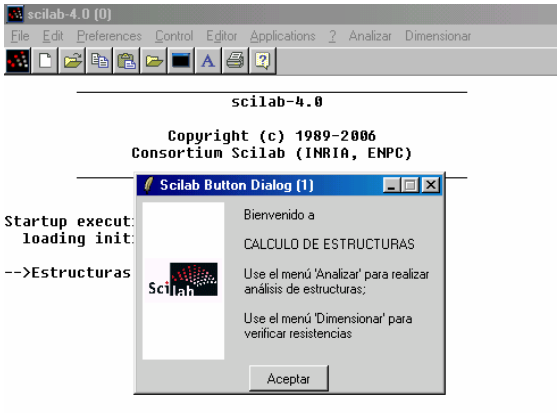

Fig. 1

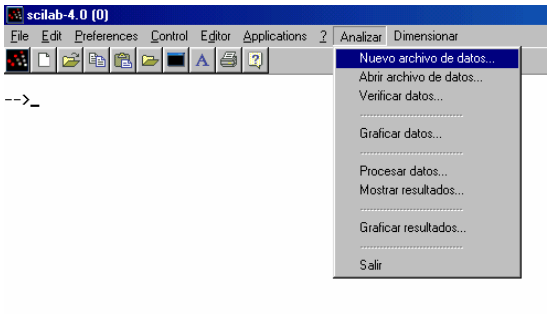

Fig. 2

Para seleccionar una tarea entre varias opciones se requiere un cuadro de diálogo de selección: la función '**x_choose**' sirve para este propósito, por ejemplo:

resp=**x_choose**(lista1,'Seleccione un tipo estructural');

'lista1' es un vector conteniendo los nombres de las opciones a presentar. El número de orden de la opción seleccionada (iniciando con 1 y no con 0 como en otros lenguajes), se devuelve en la variable 'resp', que permite luego dirigir el flujo del programa usando la instrucción 'select resp…' (ejemplo de este cuadro de diálogo en la figura 3).

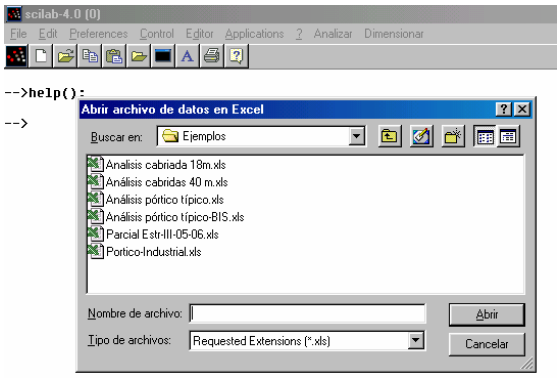

Fig. 3

El paso siguiente es abrir un archivo desde el cual leer los datos a procesar, o donde guardar los datos procesados. La función '**tk_getfile**' permite seleccionar un nombre de archivo, como en el caso siguiente (ver Fig. 4):

archdatos=**tk_getfile**('*xls',SCIHOME+'\Estructuras\Ejemplos',title='Abrir archivo de datos en Excel');

La siguiente línea de código abre el archivo seleccionado (un libro de Excel), usando la aplicación Excel (en Windows):

winopen(archdatos);

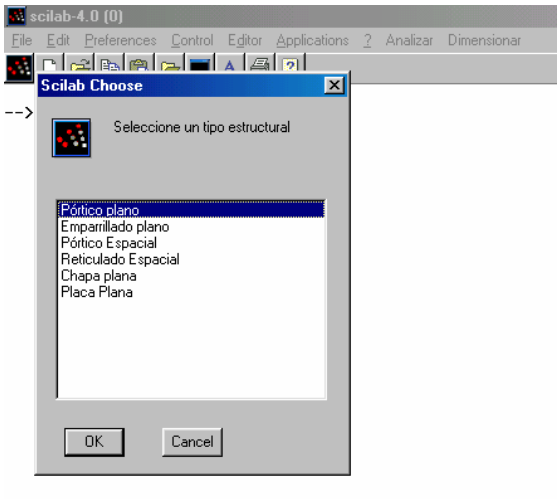

Fig. 4

5. El acceso a los datos

Es sabido que la parte más trabajosa de una aplicación puede ser preparar la interfaz de usuario. Ya se explicó antes cómo Scilab simplifica grandemente esta tarea.

La segunda mayor dificultad es acceder en forma segura a los datos a procesar y verificar su consistencia con lo que el programa espera leer en cada fase. Esta tarea se completa guardando en un archivo de texto los datos tal como fueron leídos por el programa (archivo de réplica o 'eco') para una verificación más detallada antes del procesamiento.

El acceso seguro requiere que los datos se guarden en un formato preciso, respetando una cierta 'plantilla' de datos. Auque esto puede hacerse desde un archivo de texto plano, ahora es mucho más ventajoso hacerlo en una o varias hojas de cálculo de Excel u otra aplicación similar. Al tiempo de poder ubicar los datos de cada categoría en un sitio preciso (usualmente será una columna de la hoja de cálculos), los datos mismos podrán ser el resultado de un cálculo o de una función evaluada en la misma hoja.

Scilab aporta funciones para leer directamente datos desde libros de Excel (en Windows), como explicamos a continuación.

El contenido de cada hoja de cálculos se transfiere entera a una matriz de números reales; los datos que no son numéricos (texto), se consideran de tipo '**Nan**' (Not a number, aunque en realidad es un tipo especial de número definido por Scilab, ya

que puede ser un elemento de una matriz numérica). Al mismo tiempo guarda los datos de texto en una matriz de texto, y define una matriz de índices para relacionar cada celda '**Nan**' con el texto correspondiente a dicha celda.

Las dos funciones necesarias para esto son '**xls_open**' y '**xls_read**' en ese orden; la primera abre el archivo de Excel:

[idarch,SST,NomHojas,PosHojas]=**xls_open**(archdatos);

Esta función asigna un identificador al archivo 'archdatos' (extensión '.xls') en la variable 'idarch'; también guarda en la matriz 'SST' todos los datos no numéricos (texto) de todas las hojas del libro 'archdatos'; asimismo guarda en la matriz 'NomHojas' los nombres de las hojas del libro, y finalmente guarda en la matriz 'PosHojas' los índices para acceder a cada hoja.

La cantidad de hojas del libro puede obtenerse a partir del tamaño de la matriz 'PosHojas':

CantHojas=**length**(PosHojas);

La matriz de datos numéricos de la hoja número 'ind' se lee con la siguiente función:

[rango,indices]=**xls_read**(idarch,PosHojas(ind));

La función guarda los datos numéricos en la matriz 'rango' (los elementos no-numéricos se asignan de tipo '**Nan**' como ya se explicó); la matriz 'indices' guarda un índice para cada posición (números de fila y de columna de las celdas no numéricas), que permite acceder al valor del texto extrayéndolo de la matriz 'SST', como sigue:

texto(fila,columna)=SST(indices(fila,columna)) ;

Con un poco de habilidad es posible escribir los algoritmos necesarios para obtener los datos de cada hoja (con un formato conocido) de un libro de Excel.

Para realizar un análisis estructural, los datos del mismo tipo pueden disponerse en una hoja de Excel, y el conjunto de datos en un libro de trabajo (archivo .xls) con varias hojas (unas 17 o 18 hojas para un análisis estructural), con un nombre identificatorio en la pestaña de cada hoja.

6. El proceso del análisis matricial de estructuras

Es conocido el significado físico-mecánico de los coeficientes de las matrices de un sistema estructural: estos expresan las relaciones de equilibrio estático y deformaciones elásticas de los elementos estructurales.

Mi intención aquí es solamente ilustrar el proceso matemático del análisis estructural, que es facilitado por Scilab mediante el recurso de las funciones que procesan matrices como si fueran variables de tipos simples (como un número entero, o uno racional, o una cadena de texto, un valor boolean, etc.). Los mismos recursos podrán ser aplicables a otros análisis teóricos expresables con matrices.

Scilab ahorra a sus usuarios la necesidad de escribir y depurar funciones para realizar operaciones matriciales, que consumirían tiempo de desarrollo de la aplicación y que serían mucho más lentas que las funciones aportadas por Scilab.

Otro aspecto importante que incorpora Scilab son las funciones para solución de sistemas de ecuaciones lineales, y en especial, aquellas cuyas matrices son 'raleadas', es decir con muchos elementos nulos; frecuentemente las matrices estructurales tienen menos del 20% de elementos no-nulos, dependiendo de la topología de la estructura.

Entonces, solamente los elementos no-nulos se guardan en memoria durante el procesamiento, con el consiguiente ahorro de memoria, siendo, además, el procesamiento mucho más rápido. Por ejemplo, una estructura algo grande podría requerir un sistema de 2000 a 3000 ecuaciones, lo que podría significar procesar una matriz completa de 4 a 9 millones de elementos; con un 20% de elementos no nulos se requerirá almacenar sólo entre 800 mil y 1.8 millones de números en la memoria RAM de la CPU.

7. El sistema de ecuaciones de compatibilidad elástica

La siguiente ecuación expresa el equilibrio estático de una estructura elásticamente deformable, supuesta constituída por la unión de elementos finitos (barras, superficies o segmentos de magnitud no diferencial) en

algunos puntos o secciones (nudos) de sus respectivos bordes o contornos:

KS * d = (P-Po)

Siendo **KS** la matriz del sistema (matriz de rigidez), cuyos valores numéricos dependen de las dimensiones de los elementos estructurales y de la elasticidad del material de los mismos; **d** es el vector de desplazamientos (incógnitas) de los nudos, desde una posición correspondiente a la estructura sin carga; **P** es el vector de cargas actuantes directamente en los nudos y **Po** el vector de tensiones internas de cada elemento, actuantes en los nudos de su contorno.

La solución de este sistema de ecuaciones expresa los valores de los desplazamientos **d** necesarios para producir el equilibrio estático de la estructura solicitada por cargas **P** y **Po**.

Formular y resolver este sistema es una tarea más o menos compleja, según sea el tipo de estructura y los elementos constituyentes a emplear, lo que se expresa como el 'modelo teórico' (y en nuestro caso, el 'modelo digital') de la estructura.

Los pasos de este proceso son:

7.1 definir la matriz de cada elemento estructural, en un sistema de coordenada 'local' (referido al elemento). A modo de ejemplo sencillo utilizaré la matriz de un elemento 'barra plana de sección constante', solicitada por fuerzas en su plano (elemento usado en los pórticos planos).

$$\mathbf{Kbas} := \begin{pmatrix} \frac{\mathbf{E}\cdot\mathbf{A}}{\mathbf{L}} & 0 & 0 \\ 0 & 4\frac{\mathbf{E}\cdot\mathbf{I}}{\mathbf{L}} & 2\frac{\mathbf{E}\cdot\mathbf{I}}{\mathbf{L}} \\ 0 & 2\frac{\mathbf{E}\cdot\mathbf{I}}{\mathbf{L}} & 4\frac{\mathbf{E}\cdot\mathbf{I}}{\mathbf{L}} \end{pmatrix}$$

Esta matriz básica 'Kbas' de la barra se obtiene de las ecuaciones tensión-deformación de una barra simplemente apoyada (articulación fija en un extremo y deslizante en otro):

E = Módulo de elasticidad del material de la barra;
A = área de la sección recta de barra;
I = momento de inercia de la sección;
L = longitud de la barra;

Esta matriz de 3*3 expresa las tensiones en una barra que sufre deformaciones axiales y de flexión en sus dos extremos. Soporta tensión axial y flexión en ambos extremos. Puede modificarse fácilmente para otras condiciones de vinculación.

7.2 usando esta matriz básica, es posible 'expandirla', con el fin de incluir las deformaciones transversales de la barra (situación que se presenta normalmente cuando la barra forma parte de una estructura), y al mismo tiempo expresar las deformaciones locales mediante los desplazamientos locales de los extremos de la barra,

Esta 'expansión', que transforma la matriz básica de 3*3 en una de 6*6, se realiza mediante un doble producto matricial, usando una matriz de 'compatibilidad cinemática' **A**, de 3*6, cuyos elementos expresan las relaciones entre deformaciones y desplazamientos de la barra, en el sistema local:

$$\mathbf{A} := \begin{pmatrix} -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & \frac{1}{\mathbf{L}} & 1 & 0 & \frac{-1}{\mathbf{L}} & 0 \\ 0 & \frac{1}{\mathbf{L}} & 0 & 0 & \frac{-1}{\mathbf{L}} & 1 \end{pmatrix}$$

Donde L es la longitud de la barra.

Se obtiene así la matriz **Kloc** (de 6*6) de la barra en el sistema local, mediante el doble producto matricial (se requiere una única línea de código para esta operación):

Kloc = A' * Kbas * A

expresión en la cual **A'** es la matriz transpuesta de **A**

7.3 seguidamente es necesario 'orientar' las componentes de la matriz **KLoc** con respecto al sistema global de coordenadas, realizando un doble producto con la matriz de cosenos directores del sistema local respecto al global. Con la siguiente línea de código obtenemos la matriz **Kglob** de la barra en el sistema global:

Kglob = Cos' * Kloc * Cos

Siendo **Cos** la matriz 6*6 de los cosenos directores de la barra, y **Cos'** su transpuesta:

$$\mathbf{Cos} := \begin{pmatrix} \mathbf{cosx} & \mathbf{cosy} & 0 & 0 & 0 & 0 \\ -\mathbf{cosy} & \mathbf{cosx} & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & \mathbf{cosx} & \mathbf{cosy} & 0 & 0 \\ 0 & 0 & -\mathbf{cosy} & \mathbf{cosx} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Donde:
cosx = (x2 – x1) / L
cosy = (y2 – y1) / L

x1, y1,x2,y2 son las coordenadas de los extremos de la barra;

7.4 Por último, es necesario definir un vector de 'incidencias' **T**, que relacione cada coeficiente de rigidez de la barra con el número de la componente del sistema general de ecuaciones que recibirá el 'aporte' de rigidez correspondiente de la barra:

$$\mathbf{T} := \begin{pmatrix} 3\cdot\mathbf{ext1} - 2 \\ 3\cdot\mathbf{ext1} - 1 \\ 3\cdot\mathbf{ext1} \\ 3\cdot\mathbf{ext2} - 2 \\ 3\cdot\mathbf{ext2} - 1 \\ 3\cdot\mathbf{ext2} \end{pmatrix}$$

Donde **ext1** es el número del nudo correspondiente al extremo inicial de la barra y **ext2** idem para el extremo final de la barra;

7.5 La matriz "esparcida" del sistema

Este proceso se realiza en la función 'GenerarSistema', y se hace uso de la solución para matrices 'raleadas' ('sparse matrix') que aporta Scilab.

En este proceso se asigna memoria solamente para los elementos no nulos de la matriz del sistema, para lo cual es necesario previamente generar dos vectores:

fila_col : este vector de 2 columnas y número variable (dinámico) de filas, guarda los índices (fila,columna) de cada elemento no-nulo de la matriz; y

vectK : este vector de una columna e igual número de filas que el anterior, guarda los valores no-nulos de la matriz, correspondiente a cada par fila-columna;

Para ubicar cada elemento no-nulo de la matriz de una barra en el sistema general de la estructura se hace uso del vector de orientación **T** calculado en el punto d).

Con un bucle doble **for i=1:6** y **for j=1:6** se calculan:

fila=**T(i)**;
columna=**T(j)**;

si fila==columna entonces los elementos de la diagonal principal del sistema se suman al elemento i,j del vector **vectK**:

vectK(fila)=vectK(fila)+Kglob(i,j)

y su ubicación en el sistema global se almacena en el vector **fila_col**:

fila_col(fila,1)=fila**;**
fila_col(fila,2)=columna;

si fila<>columna, se agrega una nueva fila no nula a los dos vectores

filann=filann+1
vectK(filann)=vectK(filann)+Kglob(i,j)

y:
fila_col(filann,1)=fila**;**
fila_col(filann,2)=columna;

La matriz 'esparcida' ('sparse matrix') **KS** del sistema se obtiene a partir de los vectores **fila_col** y **vectK** calculados arriba, usando la función '**sparse**':

KS = sparse(fila_col,vectK)

7.6 Los vectores de cargas

Los vectores **P** y **Po** se obtienen siguiendo una rutina similar de expansiones matriciales a partir de dos vectores 'básicos', proceso que no detallamos aquí por razones de espacio.

7.7 La solución del sistema

La solución del sistema de ecuaciones se realiza con la función '**lusolve**' de Scilab.

La solución de la ecuación **KS*d = PS** se obtiene con la siguiente línea de código:

PS = lusolve(KS,PS)

guardándose en el mismo vector de cargas **PS** (y no en un vector adicional **d,** que no es necesario) los valores de los desplazamientos calculados, ahorrándose memoria.

7.8 Tensiones en las barras

Las tensiones en las barras (en el sistema local de cada barra) se calculan, una vez calculados los desplazamientos de los nudos de la estructura, utilizando las ecuaciones de barra calculados en los pasos 7.1 a 7.4, siguiendo un proceso inverso de transformaciones matriciales similares a las ya detalladas.

No es necesario recalcar, por esto, las ventajas que ofrece Scilab para simplificar el código de este tipo de aplicaciones y aumentar la velocidad del procesamiento.

8. 'Perfomance' de las aplicaciones con Scilab

A lo largo de la exposición anterior recalcamos la serie de ventajas que ofrece Scilab a los técnicos para desarrollar rápidamente una aplicación de cálculos numéricos.

Pero también es necesario conocer otras características tan o más importantes que las mencionadas, como son, por ejemplo: la capacidad de memoria disponible para datos, la precisión de los cálculos, la velocidad del procesamiento, etc.

En primer lugar, tenemos el hecho de que Scilab no crea archivos auto-ejecutables, es decir, que nuestras aplicaciones, aún estando compiladas y reunidas en bibliotecas, sólo pueden correr en el ambiente de Scilab. Esto significa también que los enlaces con las bibliotecas se hacen en tiempo de corrida y no de compilación.

Por lo tanto, la velocidad del procesamiento puede ser más o menos veloz, dependiendo de nuestros algoritmos. En general, será más rápido cuando más funciones de Scilab usemos (las que están desarrolladas en C o Fortran), en lugar de crear rutinas especiales.

Los datos numéricos tienen una declaración implícita (Scilab no aporta declaraciones explícitas de ningún tipo de variable), usando siempre variables reales de tipo doble precisión (double: 8 bytes, con 15-16 dígitos significativos), suficiente para la mayoría de los cálculos científicos.

En cuanto a la capacidad de memoria de que puede disponerse para los cálculos, consideremos los siguientes puntos:

a) como mencionamos al comienzo, para Scilab todas las variables son matrices dinámicas de algún tipo. Esto significa que, entre bambalinas, Scilab usa variables de tipo 'puntero' para gestionar la memoria de estas matrices, y que todos los datos se colocan en el 'montón'; así, la cantidad de datos que pueden almacenarse depende solamente de la cantidad de RAM de la CPU y no del tamaño de la 'pila';

b) El hecho de poder codificar algoritmos para matrices dinámicas como si fuesen variables de tipo simple libera al usuario de la necesidad de trabajar con variables tipo puntero, (inevitable con algunos lenguajes como C). En realidad, ni siquiera es necesaria una definición previa del tamaño de la matriz antes de poder asignar valores a sus elementos (cosa inevitable en la mayoría de los lenguajes) y por lo tanto, tampoco se requiere re-definir su tamaño para agregar nuevos elementos más allá del último (es decir que, en realidad, nunca hay un último elemento en una matriz de Scilab);

c) La función 'stacksize' permiten conocer (y modificar) el tamaño de la 'pila' de Scilab (la cantidad de memoria total disponible), y la función 'gstacksize' permite conocer y modificar el tamaño de la memoria disponible para variables globales; esta última, sin embargo, se modifica automáticamente cuando la cantidad de variables globales lo requiere.

BIBLIOGRAFIA

La ayuda en línea ha sido la única bibliografía que el autor ha encontrado sobre Scilab, y ésta sólo está disponible en inglés o francés.

Textos adicionales pueden bajarse desde el sitio 'www.scilab.org'.