Revista Internacional de Métodos Numéricos para Cálculo y Diseño en Ingeniería. Vol. 5 , 2, 263–275(1989)

UN POSTPROCESADOR GRAFICO INDEPENDIENTE DE LOS DISPOSITIVOS

LUIS QUIROZ
y
EMILIO DUFEU

Departamento de Ingeniería Mecánica,
Facultad de Ingeniería,
Universidad de Concepción,
Casilla 53-C, Concepción, Chile.

RESUMEN

Se presentan las consideraciones que se tomaron en cuenta para la construcción de un sistema postprocesador gráfico independiente de los dispositivos realizados en las representaciones, para el análisis por elementos finitos mediante el programa SAP–IV.

Se muestran los algoritmos seleccionados para obtener las gráficas necesarias y su implementación en un contexto de transportabilidad del código e independencia de dispositivos periféricos, gracias al uso de estándares como el lenguaje FORTRAN 77 y el sistema GRAPHICAL KERNEL SYSTEM.

SUMMARY

Considerations that were taken into account to build a graphic postprocessor independent from the utilized devices in representations, for finite elements analysis through SAP–IV program, are presented.

Selected algorithms to obtain needed graphics and their implementation in a portability of code and independence of peripherical devices context, thanks to the use of standards like FORTRAN 77 and GRAPHICAL KERNEL SYSTEM, are shown.

INTRODUCCION

Los sistemas de pre y postprocesamiento gráfico para programas de análisis por elementos finitos son, hoy en día, una herramienta imprescindible para minimizar el tiempo requerido en el aprendizaje y utilización de dichos programas. Más aún, en la actualidad no se concibe que un programa de E.F. no disponga de las facilidades gráficas que ofrecen aquellos sistemas.

El objetivo de este trabajo es desarrollar un postprocesador gráfico para el programa SAP–IV[1], instalado desde hace años en un VAX 11/780 utilizado en educación e investigación en problemas lineales de mecánica de sólidos. Se impone como restricción básica que su código fuente sea fácilmente transportable a otros computadores y

Recibido: Octubre 1988

 ISSN 0213–1315

que sus representaciones gráficas puedan ser implementadas en una amplia gama de monitores de diversa resolución, trazadores, digitalizadores e impresoras, así como almacenadas para ser transportadas e interpretadas por otros equipos. Es decir, que sea independiente de los dispositivos que se utilizarán para ejecutar el postprocesamiento. Este requirimiento se debe a la creciente disponibilidad de diversos microcomputadores (comunicados con el VAX) y equipo periférico para éstos, a que tienen acceso los usuarios del programa de análisis.

Para lograr un postprocesador con las características mencionadas se debe utilizar un lenguaje de programación y subrutinas gráficas disponibles en la mayoría de los computadores sin grandes variaciones. Esto sólo es posible si el lenguaje y las subrutinas permanecen bajo una norma aceptada universalmente. El FORTRAN es un lenguaje de programación estandarizado por la norma ANSI X3.9 - 1978 (FORTRAN 77) y está disponible para casi todos los computadores actuales, por lo que su uso garantiza la transportabilidad del código; sin embargo, no posee capacidades gráficas.

En la década pasada se hizo grandes esfuerzos por obtener un sistema gráfico que poseyera las propiedades de independencia de los dispositivos, y que, además contara con la aceptación de usuarios y fabricantes de equipos. Estos esfuerzos se concretaron finalmente en el sistema gráfico GRAPHICAL KERNEL SYSTEM, publicado por ISO como IS 7942 en agosto de 1985[2].

EL SISTEMA GKS

El Graphical Kernel System (GKS) es el primer sistema gráfico normalizado internacionalmente. Este consiste en una biblioteca de subrutinas que permiten graficar ciertas primitivas bidimensionales independientemente del dispositivo utilizado para visualizarlas. El FORTRAN 77, por sus características, fue el primer lenguaje usado en su implementación. La independencia mencionada se logra mediante una interfaz entre las rutinas del GKS y los programas que manejan los dispositivos (Device Drivers), denominada por algunos fabricantes Virtual Device Interface (VDI) y por otros, de acuerdo con ISO, Computer Graphics Interface (CGI). La Figura 1 muestra un diagrama de los niveles de dependencia o la ubicación de los DD e interfaz respecto a éstos. Adicionalmente, la salida puede enviarse a un archivo gráfico conocido como METAFILE, para ser transportada, almacenada o interpretada por otro programa en el mismo u otro computador, pudiendo ser ingresado nuevamente en una etapa de independencia de dispositivo.

El sistema GKS utiliza una gran cantidad de conceptos básicos de infografía, como estación de trabajo, estados de operación y otros; los que fueron definidos durante el tiempo que se dedicó a la estandarización de un sistema de este tipo. Los libros de Enderle et al.[2], Foley et al.[3] y de Newman et al.[4] son excelentes referencias al respecto.

ALGORITMOS PARA EL SISTEMA GRAFICO POSTPROCESADOR

De acuerdo a los objetivos planteados, se requiere que el sistema posea las siguientes posibilidades de representación:

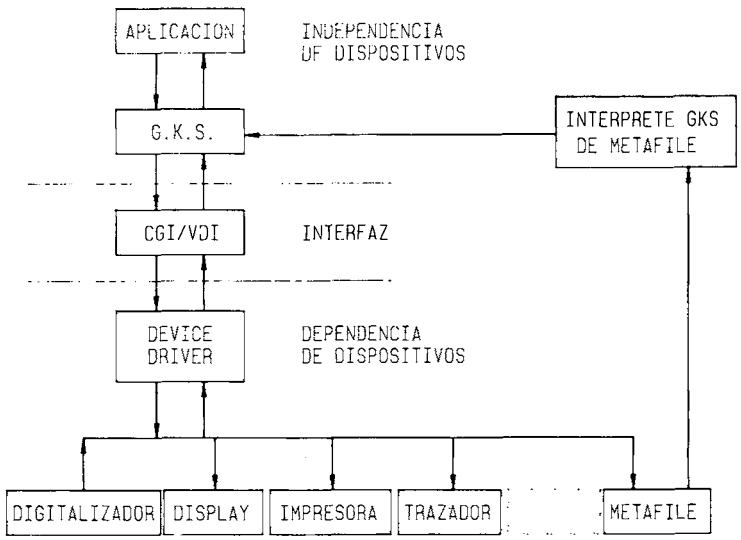

Figura 1. El sistema GKS y niveles de dependencia.

1) Representar el objeto, visto desde cualquier posición, borrando o no las líneas ocultas.
2) Representar el objeto deformado, visto desde cualquer posición, borrando o no las líneas ocultas.
3) Representar, en la superficie del objeto, las cantidades secundarias (esfuerzos, momentos, etc.) para zonas de igual valor.
4) Representar los modos normales de vibrar.
5) Representar la respuesta de un punto del objeto.

Los algoritmos necesarios para obtener dichas representaciones se pueden resumir en:

a) Algoritmo de visualización en un plano. Transformación de coordenadas 3-D a coordenadas 2-D.
b) Algoritmo de eliminación de líneas ocultas.
c) Algoritmo para graficar curvas de deformación de elementos lineales con restricciones de rotación (vigas y tuberías).
d) Algoritmo para determinar los puntos de cambio de nivel de valores nodales en la superficie de un objeto.
e) Algoritmo para graficar curvas a partir de puntos discretos para representar la respuesta en el tiempo y en frecuencia.

Algoritmo de visualización en un plano

El tipo de proyección más utilizado en la Ingeniería Mecánica es la proyección paralela. Esta está definida por un vector **V** que, a su vez, define la dirección de la proyección y por un plano de proyección definido por su normal.

El punto objeto **P** está en coordenadas absolutas (x, y, z) Figura 2.

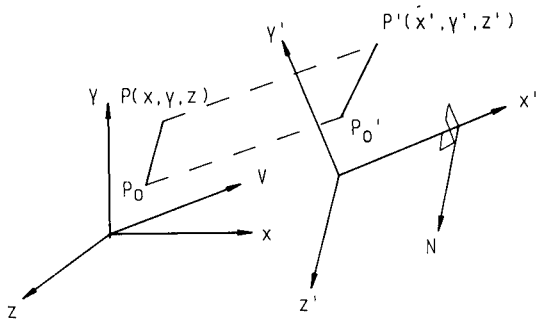

Figura 2. Proyección paralela.

El problema será determinar las transformaciones geométricas de traslación, escalamiento y rotación para encontrar las coordenadas $\mathbf{x'}$ $\mathbf{y'}$ $\mathbf{z'}$ del punto de imagen $\mathbf{P'}$ correspondiente a la proyección de $\mathbf{P(x,y,z)}$.

Debido a que la traslación se define en forma diferente (como una suma) del escalamiento y la rotación (multiplicaciones), será necesario, para poder combinarlas fácilmente, tratarlas en forma homogénea.

Traslación:

$$\begin{vmatrix} x' \\ y' \\ z' \\ 1 \end{vmatrix} = \begin{vmatrix} 1 & 0 & 0 & D_x \\ 0 & 1 & 0 & D_y \\ 0 & 0 & 1 & D_z \\ 0 & 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} x \\ y \\ z \\ 1 \end{vmatrix} \quad \text{o} \quad \mathbf{p'} = \mathbf{T}\,\mathbf{p} \tag{1}$$

con Dx, Dy y Dz las componentes del desplazamiento en las respectivas coordenadas.

Escalamiento:

$$\begin{vmatrix} x' \\ y' \\ z' \\ 1 \end{vmatrix} = \begin{vmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 1 \\ 0 & 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} x \\ y \\ z \\ 1 \end{vmatrix} \quad \text{o} \quad \mathbf{p'} = \mathbf{S}\,\mathbf{p} \tag{2}$$

donde Sx, Sy y Sz son los factores de escala para cada coordenada.

Rotación:

En este caso, la matriz de transformación depende respecto de qué eje se efectuará la rotación (Figura 3).
En el eje $\mathbf{z}$ y rotación θ

$$\begin{vmatrix} x' \\ y' \\ z' \\ 1 \end{vmatrix} = \begin{vmatrix} \cos\theta & -\text{sen}\theta & 0 & 0 \\ \text{sen}\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} x \\ y \\ z \\ 1 \end{vmatrix} \quad \text{o} \quad \mathbf{p'} = \mathbf{R}\theta, z\,\mathbf{p} \tag{3}$$

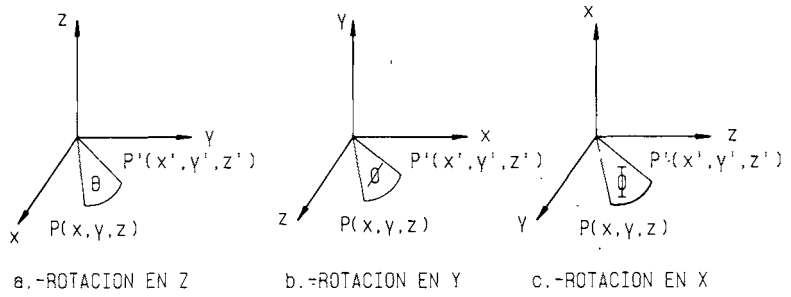

Figura 3. Rotaciones alrededor de ejes coordenados.

En el eje **y** y rotación ϕ

$$\begin{vmatrix} x' \\ y' \\ z' \\ 1 \end{vmatrix} = \begin{vmatrix} \cos\phi & 0 & \text{sen}\phi & 0 \\ 0 & 1 & 0 & 0 \\ -\text{sen}\phi & 0 & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} x \\ y \\ z \\ 1 \end{vmatrix} \quad \text{o} \quad \mathbf{p}' = \mathbf{R}\phi, z\,\mathbf{p} \tag{4}$$

En el eje **x** y rotación Φ

$$\begin{vmatrix} x' \\ y' \\ z' \\ 1 \end{vmatrix} = \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\Phi & -\text{sen}\Phi & 0 \\ 0 & \text{sen}\Phi & \cos\Phi & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} x \\ y \\ z \\ 1 \end{vmatrix} \quad \text{o} \quad \mathbf{p}' = \mathbf{R}\Phi, z\,\mathbf{p} \tag{5}$$

Una transformación acumulativa general en el espacio, será el resultado de premultiplicar **p** o **p'** por cada una de las transformaciones **S T** o **R**, lo cual indica que dicho resultado dependerá del orden en que se apliquen las transformaciones.

Una vez obtenida la transformación deseada se define el plano de visión como el formado por **x'** e **y'**, donde el valor de **z'** indicará la profundidad del punto respecto a dicho plano, mediante la transformación

$$\begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} x \\ y \\ z \\ 1 \end{vmatrix} \quad \text{o} \quad \mathbf{P}\,\mathbf{p} \tag{6}$$

Así se ha definido una transformación proyectiva paralela, cuyo vector de proyección es paralelo a la dirección **z'**, la que a la vez, es la dirección de la normal al plano de proyeccción.

Algoritmo de eliminación de líneas ocultas

En la literatura existen varios algoritmos más o menos eficientes[4,5]. En nuestro caso se debe elegir uno que, aparte de su eficiencia, no dependa del dispositivo utilizado para la visualización, por lo que no son útiles los que efectúan esta operación mediante comparación y barrido de puntos o pixels.

Inicialmente se consideró el algoritmo propuesto por Janssen[6], muy popular porque publica el programa fuente; pero se mostró muy ineficiente cuando existe una gran

cantidad de elementos en el modelo. Además no es muy útil cuando se requiere la representación de los valores nodales en la superficie de la pieza, ya que para esto, es más conveniente uno que borre las superficies ocultas más que las líneas. Por lo anterior se desarrolló un método basado en parte en el trabajo de Gordon[7] y el algoritmo de Newell-Sancha[5]. Este método consiste en eliminar las superficies ocultas determinando la visibilidad de las superficies de cada elemento en forma individual, de acuerdo al valor del área proyectada por ellas sobre el plano de visión. Para esto es necesario que los nodos que forman cada superficie estén ordenados en forma antihoraria respecto a la normal exterior de ella. Primero se eliminan las que son comunes a dos elementos adyacentes, las que obviamente son invisibles. Luego, si el área proyectada de alguna de las que quedan es negativa o cero, la superficie es invisible y se elimina. Para las que quedan se determina su profundidad máxima y se ordenan en forma descendente de acuerdo a este parámetro para posteriormente dibujarlas en ese orden, traslapándose las que ocultan parcialmente a otras. En el caso de elementos de cáscara delgada o placas, sólo se utiliza esta última parte del algoritmo.

Algoritmo para graficar curvas de deformación de elementos de vigas y tuberías[3]

En el caso de elementos de vigas y tuberías, no es posible representar su deformada en el espacio mediante una recta que una la posición final de los nodos extremos. Es necesario tomar en cuenta la rotación que se produce en dichos nodos y por lo tanto su representación será una curva alabeada.

Como los datos que se obtienen del procesador son los desplazamientos y rotaciones nodales, parecería que la forma paramétrica de Hermite es la más adecuada para el efecto. Sin embargo, como es más simple efectuar las transformaciones geométricas sólo a puntos y no a puntos y tangentes, se calculan puntos auxiliares con las rotaciones nodales y se utiliza la forma paramétrica de Bezier (Figura 4). Inicialmente se efectúa la transformación geométrica deseada obteniendo los puntos de Bezier en coordenadas $\mathbf{x'}$ $\mathbf{y'}$ del plano de visión. Así, si $\mathbf{0 < t < 1}$

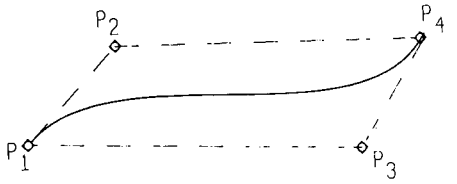

Figura 4. Puntos de Bezier.

$$x'(t) = T\ Mb\ Gb_{x'} \tag{7}$$
$$y'(t) = T\ Mb\ Gb_{y'} \tag{8}$$

donde

$$T = [t^3\ t^2\ t\ 1]$$

$$Mb = \begin{vmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{vmatrix} \qquad Gb = \begin{vmatrix} P_1' \\ P_2' \\ P_3' \\ P_4' \end{vmatrix}$$

Algoritmo para determinar los puntos de cambio de nivel de valores nodales en la superficie de un objeto

Se modificó el SAP–IV para los elementos que no calculan las variables secundarias (esfuerzos, momentos y otros) en los nodos, calculando su valor en los puntos definidos por Barlow[8], asignándolo a los nodos y promediando los valores asignados a cada nodo por todos los elementos comunes a él. Con esto se obtiene un campo relativamente suave de valores nodales secundarios.

El algoritmo para dibujar los niveles se puede resumir en los siguientes pasos:

a) Se determina el mínimo y el máximo valor nodal existente en el modelo y se construye una escala desde el mínimo al máximo en un número cualquiera de tramos, definiendo así los límites entre tramos, o sea, los cambios de nivel de valores nodales.
b) Se establece el nivel o tramo al que corresponde el valor nodal en cada nodo.
c) Se encuentran los puntos sobre la frontera de la superficie de un elemento, para los cuales existe un cambio de nivel, mediante interpolación lineal.
d) Se generan polígonos, dentro del elemento, a partir de estos puntos de cambio de nivel, de modo que, todos los puntos dentro de un polígono corresponden a un mismo nivel.
e) Se aplica el procedimiento a todas las superficies de cada elemento y a todos los elementos.

La Figura 5 muestra un esquema de un elemento triangular sobre el cual se ha aplicado el algoritmo.

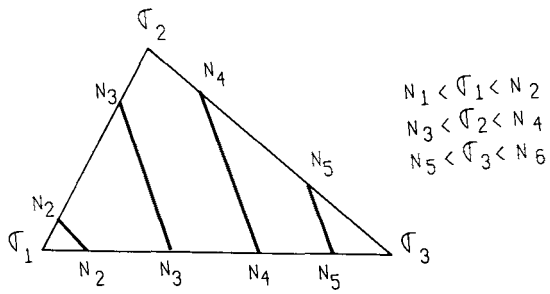

Figura 5. Elemento triangular con líneas de cambio de nivel de esfuerzos.

Algoritmo para graficar curvas 2-D a partir de puntos discretos para representar la respuesta[9]

Este consiste fundamentalmente en dibujar una curva plana que pase por **n** puntos. Dicha curva debe, además, ser continua en su primera derivada para cada punto. Por lo tanto, a lo menos, la interpolación debe ser parabólica de segundo grado y para definirla se requiere un mínimo de tres puntos.

Sin embargo, se conoce que una parábola cúbica es más suave y se aproxima mejor al trazado a mano de una interpolación (no representa variaciones bruscas) y para definirla se necesita conocer cuatro puntos sobre ella.

La interpolación más conocida que pasa por todos los puntos es la de Lagrange, la cual, en su forma paramétrica, consiste en encontrar las funciones de interpolación de Lagrange $\mathbf{B_i(t)}$ tales que:

$$\mathbf{x(t)} = \sum_{\mathbf{i=1}}^{\mathbf{n}} \mathbf{x_i}\ \mathbf{B_i(t)} \tag{9}$$

$$\mathbf{y(t)} = \sum_{\mathbf{i=1}}^{\mathbf{n}} \mathbf{y_i}\ \mathbf{B_i(t)} \tag{10}$$

Como se tratará con cúbicas, $\mathbf{n = 4}$ y las funciones de interpolación quedan:

$$\mathbf{B_1(t)} = \frac{\mathbf{t(t-1)\ (t-2)}}{\mathbf{-6}} \tag{11}$$

$$\mathbf{B_2(t)} = \frac{\mathbf{(t+1)\ (t-1)\ (t-2)}}{\mathbf{2}} \tag{12}$$

$$\mathbf{B_3(t)} = \frac{\mathbf{(t+1)\ t\ (t-2)}}{\mathbf{-2}} \tag{13}$$

$$\mathbf{B_4(t)} = \frac{\mathbf{(t+1)\ t\ (t-1)}}{\mathbf{-6}} \tag{14}$$

con $\mathbf{t = -1}$ para el primer punto, $\mathbf{t = 0}$ para el segundo, $\mathbf{t = 1}$ para el tercero y $\mathbf{t = 2}$ para el cuarto.

Si hay que interpolar **N** puntos, se toman los primeros cuatro ($t = -1$, $t = 0$, $t = 1$ y $t = 2$) y se dibuja la curva entre $\mathbf{t = -1}$ y $\mathbf{t = 1}$. Luego se consideran los tres últimos de éstos, el cuarto será el punto siguiente, y se grafica la curva entre el segundo y el tercero ($\mathbf{t = 0}$ y $t = 1$), y así sucesivamente hasta llegar a considerar el punto **N** en que, para el conjunto de los cuatro últimos puntos, se dibuja la curva entre $\mathbf{t = 0}$ y $\mathbf{t = 2}$ (Figura 6).

Implementación del programa postprocesador

Para la construción del postprocesador se utilizó un microcomputador IBM-AT y un compilador FORTRAN 77, teniendo especial cuidado de no utilizar ninguna llamada al sistema operativo DOS 3.1 para mantener la independencia. El sistema GKS se disponía

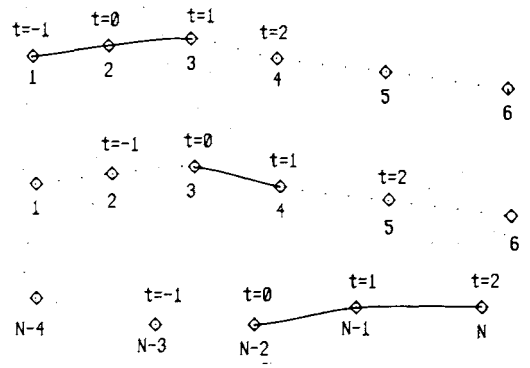

Figura 6. Construcción de una curva de interpolación bidimensional.

sólo para este sistema operativo, pero existe en el mercado versiones para Unix, VMS, CMS y otros. Debido a esto se construyó en dicho microcomputador y no fue posible ensayarlo en su totalidad en otra máquina, salvo que el programa fuente en FORTRAN fue compilado en un VAX 11/780 con S.O. VMS, un DEC 1020 con TOPS10 y un Burroghs B1800 sin indicación de errores excepto en las sentencias OPEN en el DEC y el Burroghs, las que fueron modificadas apropiadamente sin mayores dificultades. El programa hace uso de un máximo de ocho colores en sus salidas gráficas, lo que en cierto modo implica una dependencia. Para obviar esto, se le pregunta al usuario, antes de las opciones de visualización, si posee dicha capacidad de colores. Si no es así, en lugar de diferenciar por colores lo hace por tipos de líneas y achurados.

Para probar la independencia de las salidas gráficas respecto a los periféricos se usaron cuatro tipos diferentes de pantallas: una CGA de 320 × 200 pixels y 3 colores, una EGA de 640 × 350 y 16 colores, una PGA de 640 × 480 y 256 colores, y una monocromática de 720 × 350. En todos se obtuvo sin problemas la visualización después de cargar el controlador de dispositivo correspondiente. Por supuesto existió siempre una gran diferencia en la salida de la CGA respecto a las otras en cuanto a la definición de la figura, por lo que no es recomendable su uso, aunque actualmente este tipo ha sido prácticamente abandonado reemplazándose por la EGA que es la que se dispuso para el desarrollo de este trabajo.

Además se envió la salida a un archivo gráfico METAFILE y desde ahí se obtuvo la representación en otros periféricos como una impresora IBM Graphics, un trazador con lenguaje HPGL y una impresora a color de chorro de tinta que se utilizó también para efectuar barridos de pantalla de la EGA.

EJEMPLOS

La fotografía de la Figura 7 muestra los niveles de esfuerzo equivalente de Von Mises en un sector de cigüeñal sometido a cuatro fuerzas nodales tangentes, que simulan un torque. Este problema se desarrolló, originalmente, para determinar la influencia del espesor del alma en la rigidez torsional. Dicha fotografía se obtuvo directamente de la pantalla EGA. La misma representación se envió a un archivo gráfico METAFILE y luego a una impresora de matriz de puntos, donde, como no se dispone de colores, la

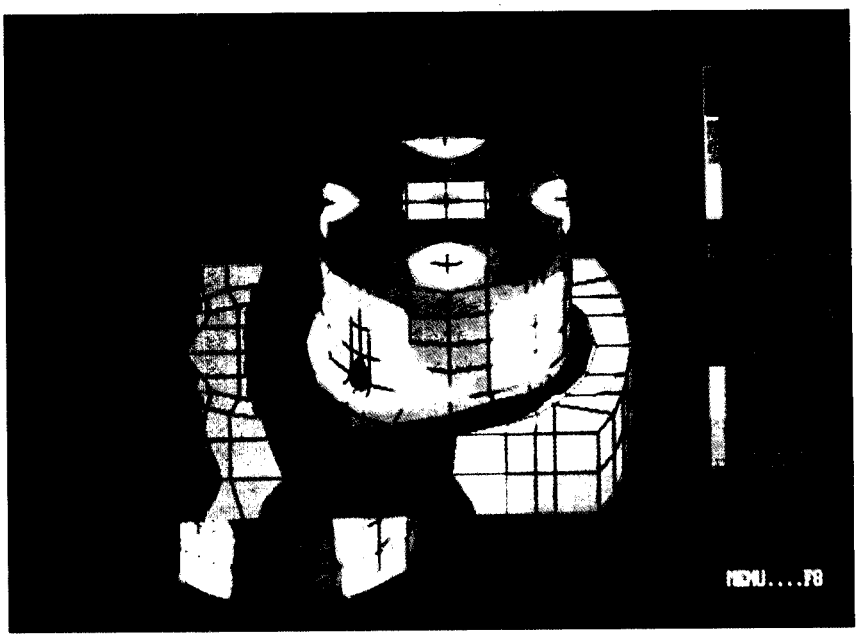

Figura 7. Esfuerzos equivalentes de Von Mises. Pantalla E.G.A..

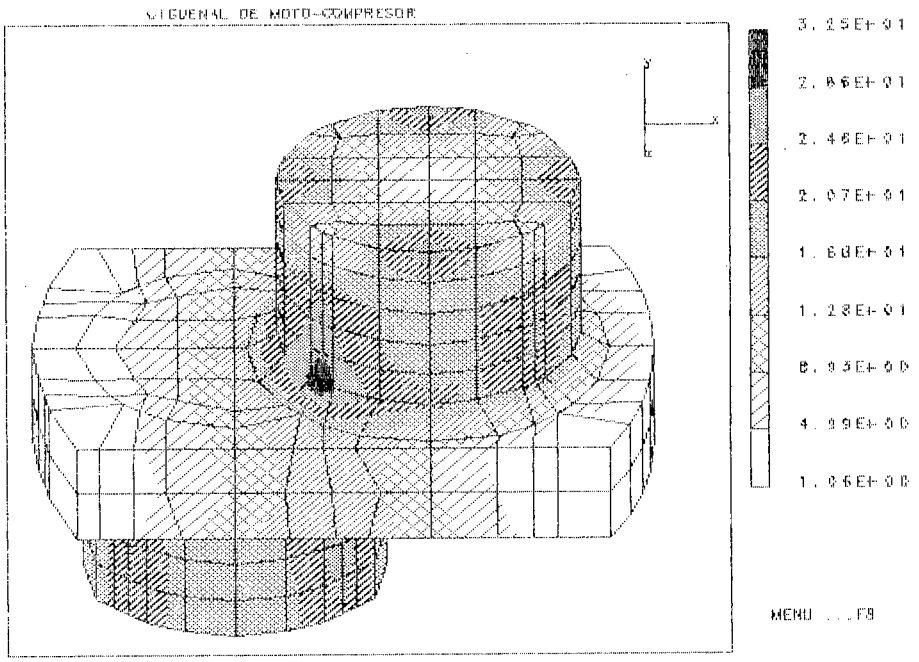

Figura 8. Esfuerzos equivalentes de Von Mises. Impresora.

diferencia de niveles es por tipos de achurados (Figura 8). Aquí se han utilizado los algoritmos de visualización en un plano, de eliminación de líneas ocultas y de cambio de nivel de valores nodales.

Para mostrar un ejemplo del algoritmo para graficar curvas de deformación de elementos de vigas tuberías, se analizó una red de tuberías tomada del libro de The Kellogg Co.[10]. La Figura 9 muestra la deformada estática superpuesta a la red, fotografiada desde la pantalla. El mismo tipo de salida se obtiene graficando los modos normales de vibrar de la red.

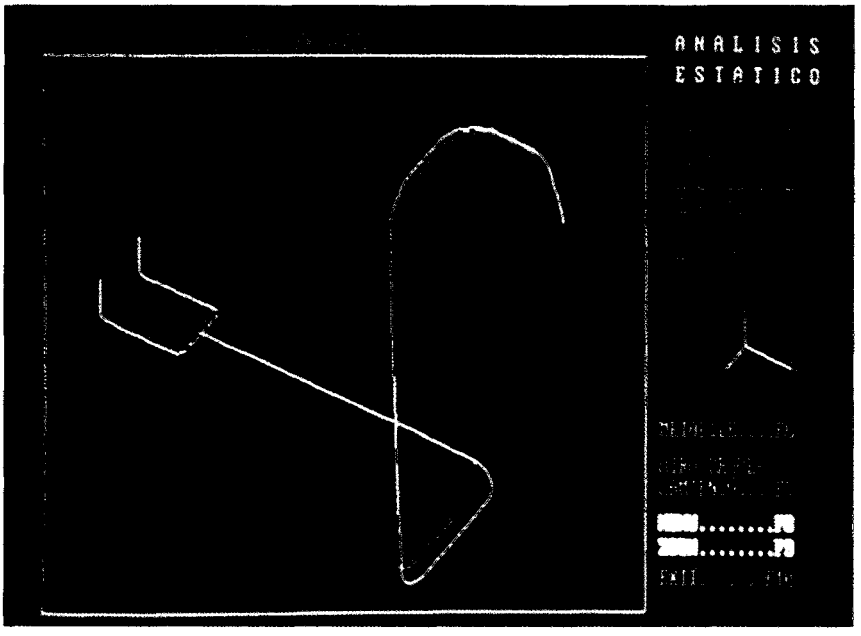

Figura 9. Red deformada ejemplo Kellogg.

Finalmente se muestra la aplicación del algoritmo para dibujar curvas bidimensionales a la misma red anterior en un análisis de la respuesta frente a una excitación. En la Figura 10 se observa la curva de respuesta en frecuencia obtenida con dicho algritmo. Una representación similar se obtiene para la respuesta en el tiempo.

CONCLUSIONES

Se han escogido los algoritmos disponibles en la literatura que permitan preservar las características de independencia, tanto respecto del computador utilizado como de los equipos periféricos.

No ha sido posible ensayar el sistema gráfico en otros computadores por no tener éstos disponible el sistema GKS pero sí se ha probado con diferentes dispositivos periféricos para microcomputadores del tipo PC. Sin embargo, por las características del lenguaje utilizado y del GKS, los autores esperan que el programa funcionará con muy pocas o ninguna modificación en otras máquinas.

Actualmente está por concluirse el desarrollo de un preprocesador gráfico para el SAP-IV con las mismas propiedades del postprocesador presentado aquí, pero

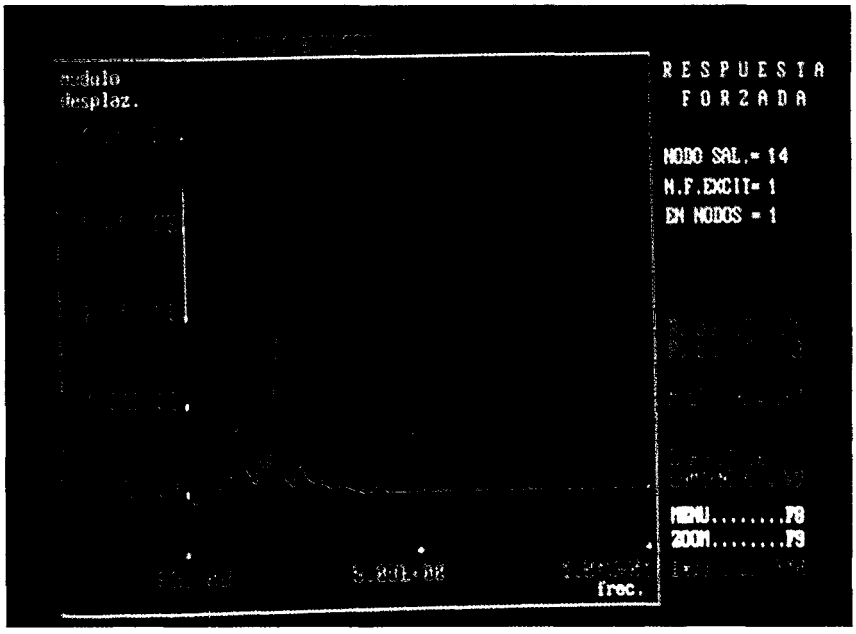

Figura 10. Respuesta en frecuencia ejemplo Kellogg.

cuyas necesidades de interfaces gráficas e interactividad con el usuario presentan una problemática muy diferente, lo que será un tema de un próximo artículo.

AGRADECIMIENTOS

Se agradece el aporte recibido por parte de la Dirección de Investigación de la Universidad de Concepción para el proyecto 20.94.12, en cuyo marco se inscribe este trabajo.

REFERENCIAS

1. K.J. Bathe, E.L. Wilson y F.E. Peterson, "SAP-IV. A Structural Analysis Program for Linear Systems", *Report EERc 73-11*, Universidad de California, Berkeley, (1974).
2. G. Enderle, K. Kansky y G. Pfaff, *"Computer Graphics Programming - GKS - The Graphics Standard"*, Springer-Verlag, Berlín, (1987).
3. J.D. Foley y A. Van Dam, *"Fundamentals of Interactive Computer Graphics"*, Addison Wesley, Reading, Mass, (1983).
4. W.M. Newman y R.F. Sproull, *"Principles of Interactive Computer Graphics"*, Graw-Hill International, 2^a Ed., (1979).
5. I. Sutherland, R. Sproull y R. Schumaker, "A Caracterization of Ten Hidden Surface Algorithms", *Computing Surveys*, Vol. **6**, no. 1, pp. 1–55, (1974).
6. T. Janssen, "A Simple Efficient Hidden Line Algorithm", *Computers and Structures*, Vol. **17**, pp. 33–37, (1983).
7. J.J. Gordon y C.L. Foodzeit, "COIFES – An Efficient Structural Graphics Program Using the Hidden Line Technique", *Computers and Structures*, Vol. 12, pp. 699–712, (1980).

8. J. Barlow, "Optimal Stress Locations in Finite Element Models", *Int. J. Num. Meth. Eng.*, Vol. **10**, pp. 243–251, (1976).
9. P. Beckers, *"Introduction to CAD Systems"*, Universidad de Liege, Bélgica, (1987).
10. The M.W. Kellogg Co., *"Design of Piping Systems"*, John Wiley and Sons Inc., Nueva York, (1967).