Programa del Medio Elíptico

# OBJETO

Este programa desarrollado en Python/Matlab consigue:

* Proporcionar funciones básicas para la manipulación de elipses definidas ya sea en ejes locales (mediante los parámetros a y b –longitud de los semiejes-, x0 e y0 –posición del centro en ejes globales- y alfa –inclinación con respecto al sistema fijo-) o en ejes globales (mediante la matriz A de la cónica)
* Generar de forma aleatoria un número arbitrario de elipses en la parte superior de un medio rectangular ( de dimensiones LxH), cuyas dimensiones máximas y mínimas pueden definirse (maxa,maxb;mina.minb).
* Mover estas elipses en dirección descendente hasta alcanzar la base del cajón u otra elipse (en este último caso no se sitúan en el punto de tangencia exacto, sino que la solución es aproximada).

En base a esto se podría programar en un futuro el comportamiento del suelo como medio discreto, para lo cual habría que:

* Tratar de mejorar la intersección entre elipses de forma que la posición final de las elipses sea tal que sólo sean tangentes entre sí (no secantes). [No mandatorio]
* Para cada situación, resolver el equilibrio entre las elipses existentes y determinar las reacciones en los puntos de contacto. (Aplicando simplificaciones si no queremos recurrir a la teoría de la elasticidad, ya que en el caso general, se formaría una “estructura” hiperestática)
* Conocidas las reacciones en los contactos, determinar si se produce el deslizamiento, la rodadura o ambas cosas en los contactos.
* Aplicar las ecuaciones cinemáticas de forma ordenada a las diferentes elipses de forma que se dé lugar a una nueva situación más próxima a la de equilibrio.

Adicionalmente, sería interesante añadir la opción de aplicar cargas externas (lo cual no debería suponer un problema una vez programados los puntos anteriores).

# RESPECTO AL LENGUAJE DE PROGRAMACIÓN EMPLEADO:

Originalmente el programa se ha realizado en Python 3.0.

Personalmente considero que este lenguaje resulta más cómodo a la hora de programar y que la posibilidad de emplear programación orientada a objetos podría ser muy útil en un futuro de cara a obtener un código más limpio y versátil (aunque para ello habría que reescribir gran parte del código ya realizado, que hace uso de este paradigma de programación con objeto de mantener una relación más próxima con Matlab)

La versión en Matlab se ha realizado con posterioridad y las diferencias en cuanto a funcionalidad son mínimas entre ambas versiones (existen mínimas diferencias como los avisos en caso de introducir ciertos datos de manera incorrecta, o el distinto coloreado de las diferentes elipses, que no están implementados en Matlab).

Para mi sorpresa, la versión en Matlab ha resultado ser más eficiente en términos de tiempo de ejecución. Desconozco si esto se debe a que hace un mejor uso de los múltiples núcleos del procesador de mi ordenador, o a que he hecho uso de alguna función poco optimizada en Python.

Por tanto, y puesto que la velocidad de cálculo me parece importante en un programa de estas características, creo que la opción de emplear Matlab si estos resultados se mantienen en otros equipos debería ser tenida en cuenta a pesar de ser un lenguaje menos versátil.

# FUNCIONES DEFINIDAS:

Las funciones definidas en el programa son las siguientes.

|  |  |  |
| --- | --- | --- |
| **Función** | **Descripción** | **Salida** |
| **dibuja\_conica(x,y,A,color)** | Dibuja la cónica definida por la matriz A en el espacio delimitado por los vectores x e y, y en el color (string) indicado. Devuelve un error si la longitud de los vectores x e y no son iguales. Devuelve los valores de X, Y y Z en toda la malla (los puntos con Z=0 son el contorno de la elipse). | X,Y,Z |
| **analisis\_espectral(A)** | Devuelve los autovectores y autovalores de una matriz dada | Autovalores, Autovectores |
| **es\_elipse(A)** | Comprueba que la cónica dada por A corresponda a una elipse real. | Comprobación (verdadero o falso) |
| **haz\_de\_elipses(x,y,A1,A2)** | Dibuja en unos ejes discretizados según los vectores x e y (que deben tener la misma longitud) el haz de elipses que pasa por las intersecciones de las dos elipses dadas por A1 y A2. *(La utilidad de esta función es meramente visualizar este haz de cara a realizar comprobaciones en la formulacion)* |  |
| **determinante(A)** | Devuelve el determinante de la matriz A | detA |
| **adjuntos(A)** | Devuelve la matriz de adjuntos de la matriz A | Adj |
| **es\_tangente\_fondo(A,H)** | Comprueba que la elipse dada por A es tangente al fondo (profundidad H). Lo es si devuelve un valor 0. | Comprobación |
| **es\_tangente\_izquierda(A)** | Comprueba si la elipse dada por A es tangente a la pared izquierda (Situada en el origen). Lo es si devuelve un valor de 0. | Comprobación |
| **es\_tangente\_derecha(A,L)** | Comprueba si la elipse dada por A es tangente a la pared derecha (situada a una distancia L del origen). Lo es si devuelve un valor de 0. | Comprobación |
| **son\_tangentes(A,B)** | Comprueba si las elipses dadas por A y B son tangentes, lo serán si el valor devuelto es de 0. *A diferencia de las funciones anteriores, esta función me ha resultado poco útil ya que los valores devueltos no parecen aproximarse de forma suave a cero a medida que las elipses se acercan. Me ha parecido mal indicador ya que solo se cumple cuando la tangencia es perfecta*. | Comprobación |
| **centro(A)** | Devuelve un vector con las coordenadas x0 e y0 del centro de la cónica definida por la matriz A. | Centro |
| **orientación\_ejes(A)** | Devuelve la inclinación con respecto a los ejes fijos de la elipse dada por la matriz A (en radianes) | Alfa |
| **cambio\_locales\_fijos(a,b,x0,y0,alfa)** | Devuelve una matriz A que define la elipse dados los semiejes a y b, las coordenadas x0 e y0 de su centro y su inclinación alfa (en radianes) con respecto al sistema global. | A |
| **genera\_elipse\_globales(maxa,maxb, mina, minb,L)** | Esta función genera una elipse de tamaño del semieje según x comprendido entre maxa y mina y de tamaño del semieje según y comprendido entre maxb y minb, y la sitúa en la zona superior de la “caja”. La función devuelve los parámetros a,b,x0,y0 y alfa que definen la elipse en ejes locales y su transformación a ejes fijos. | a,b,x0,y0,alfa |
| **genera\_elipse\_fijos(maxa,maxb, mina, minb,L)** | Esta funciónes análoga a la anterior pero devuelve la matriz A que define a la elipse en ejes fijos. | A |
| **punto\_inferior(a,b,A)** | Devuelve la coordenada y del punto inferior de la elipse. Cuando el valor de esta función alcance el valor de H, se produce el contacto con el fondo.  Se trata de una alternativa a la función **es\_tangente\_fondo** que simplifica y optimiza la colocación de la elipse en contacto con el fondo con precisión máxima. Con ella, llegados a una cierta zona donde no se esperen contactos con otras elipses, con esta función se puede mover el centro de la elipse la distancia exacta necesaria para alcanzar el fondo. | inferior |
| **existe\_contacto\_elipses(x,y,A1,A2)** | Esta función calcula los valores de la forma cuadrática de las elipses A1 y A2 en un mallado dado por los vectores x e y, y determina si ambas elipses son tangentes evaluando si de todos esos puntos de la malla, existe alguno en el cual los residuos de las ecuaciones de las dos elipses son menores que cero, lo cual indicaría que existen puntos interiores a ambas elipses simultáneamente, y que por tanto, las elipses se intersecan.  Esta función da resultados correctos, y en base a ella se ha podido ampliar el programa principal para que vaya bajando las elipses y se detenga cuando estas se intersequen con otras. Sin embargo, me gustaría encontrar otra forma de determinar el momento en que las elipses se intersecan con un coste computacional más reducido… | True/False |
| **simetriza(A)** | Dada una matriz A que debería ser triangular inferior, genera una matriz simétrica a partir de ella. Devuelve un error si la matriz no es cuadrada. | A |

# CUERPO DEL PROGRAMA:

A continuación voy a comentar el cuerpo del programa tratando de explicar su funcionamiento. Utilizare para ello la versión en Matlab, ya que parece funcionar más rápidamente y ser el lenguaje más empleado en la escuela.

EL código del cuerpo del programa (“encapsulado” en la función “*medio\_eliptico*”) es el siguiente:

function[comprobacion\_contacto\_elipses,elipses,matriz\_contactos]=medio\_eliptico()

Defino la función que encapsula el cuerpo del programa (y he marcado como salidas de la función algunas variables interesantes para la evaluación del comportamiento del programa, pero estas no son en absoluto necesarias más allá de para su revisión). En cuanto a los argumentos (entradas), de momento no he establecido ninguno y asigno los valores necesarios en el propio cuerpo del programa (por facilidad en esta fase de desarrollo)

clc

clear all

hold on.

Procedo a limpiar las variables de ejecuciones previas y a indicar que no deseo que cada nuevo gráfico suplante al anterior

H=10; %Profundidad del fondo (positivo hacia abajo)

L=10; %Longitud de la franja de terreno estudiada (encajada en paredes verticales)

N=50; %Resolución

x=linspace(0,H,N); %Espacio x de trabajo

y=linspace(0,L,N); %Espacio y de trabajo

maxa=1; maxb=.75; mina=0.75; minb=0.5; %Valores máximos y mínimos de la longitud de los ejes

maxelipses=15; %Numero máximo de elipses introducidas.

Procedo a definir algunas variables básicas del programa: las dimensiones del medio (H y L), la resolución de la malla (N) , los valores máximos y mínimos de los ejes de las elipses y el número de elipses aintroducir.

elipses=ones(3,3,maxelipses);

Esta variable de 3 dimensiones almacena todas las elipses que se generarán. Sus dos primeras dimensiones son las “matrices” de cada elipse, y su tercera dimensión indica la elipse a la que nos referimos. Así elipses(1,2,5) sería la componente (1,2) de la elipse 5. De momento inicializaré esta variable haciendo que todos sus elementos valgan 1.

[a,b,x0,y0,alfa]=genera\_elipse\_locales(maxa,maxb, mina, minb,L);

Genero una primera elipse (locales).

fprintf('Moviendo elipse 1 de %d',maxelipses)

Comienza el proceso de movimiento de esta primera elipse:

while true==true;

A=cambio\_locales\_fijos(a,b,x0,y0,alfa);

if punto\_inferior(A)<(H-min(mina,minb));

x0=x0+min(mina,minb);

elseif punto\_inferior(A)<H;

x0=x0+(H-punto\_inferior(A));

else

elipses(:,:,1)=A(:,:);

break

end

end

dibuja\_conica(x,y,A,'green');

Este trozo de código pasa la elipse a ejes globales y comprueba que no su punto más bajo (obtenido mediante la función “punto\_inferior”) no sea inferior al fondo del medio. Si no lo es, procede a moverla hacia abajo. Si detecta que ha superado el fondo, significa que ha llegado al fondo sin chocar con otro elemento, así que la mueve automáticamente hasta recuperar su posición tangente al suelo. Cuando se da este caso, guarda la matriz de la elipse en la variable “elipses” y termina el bucle. Finalmente se dibuja la elipse en su posición final.

matriz\_contactos=zeros(maxelipses,maxelipses);

Inicializo esta variable cuya función será contar con una forma sencilla de comprobar qué matrices se intersecan entre sí.

De esta forma queda completado el movimiento de la primera elipse, que resulta la mas sencilla dado que no puede chocar con ninguna otra elipse.

El próximo bucle “for” realiza en cada iteración el movimiento del resto de las elipses (cuya numeración queda indicada por la variable “numeroelipse”). Su funcionamiento es similar al empleado para la primera elipse (más sencilla dado que no puede chocar con otra elipse).

for numeroelipse=2:maxelipses;

[a,b,x0,y0,alfa]=genera\_elipse\_locales(maxa,maxb, mina, minb,L);

fprintf('Moviendo elipse %d de %d',numeroelipse,maxelipses)

for i=1:1000;

La primera diferencia con el código anterior es que en este caso se limita el máximo número de iteraciones (movimientos) realizados con cada elipse, por una cuestión de facilidad de depuración del código.

A=cambio\_locales\_fijos(a,b,x0,y0,alfa);

comprobacion\_contacto\_elipses=zeros(maxelipses,1);

Esta variable se introduce para registrar con qué elipses contacta la nueva elipse introducida. Se trata de un vector de longitud “maxelipses” que inicialmente estará compuesto únicamente de ceros.

for j=1:numeroelipse;

comprobacion\_contacto\_elipses(j)=existe\_contacto\_elipses(x,y,A,elipses(:,:,j));

end

Antes que nada, se comprueba con qué elipses contacta la nueva elipse (una por una con todas las existentes hasta este momento), almacenando los resultados positivos (contactos) como unos (o trues booleanos) en el vector “comprobación\_contacto\_elipses”.

if any(comprobacion\_contacto\_elipses)

matriz\_contactos(numeroelipse,:)=true;

break

Si la elipse está en contacto con cualquier otra (existe cualquier uno en el vector “comprobación\_contacto\_elipses”), se detiene el movimiento de la elipse actual, y se anotan los contactos oportunos en “matriz\_contactos”, ya que la variable “comprobación\_contacto\_elipses” será sobreescrita al mover la siguiente elipse.

elseif punto\_inferior(A)<(H-min(mina,minb))

x0=x0+min(mina,minb)/10; %Aquí se indica cuánto avanza en cada iteración

elseif punto\_inferior(A)<H

x0=x0+(H-punto\_inferior(A));

La detección del contacto con el fondo es análoga al caso anterior. Sin embargo en este caso es importante mencionar que podemos determinar el paso de avance de las elipses, reduciéndolo si queremos obtener una mejor aproximación a la tangencia (con mayor esfuerzo de cálculo)

elseif i==999

%'ERROR: Número de Iteraciones Insuficiente'

break

else

break

end

end

dibuja\_conica(x,y,A,'blue');

elipses(:,:,numeroelipse)=A(:,:);

end

El resto del código es análogo al ya comentado anteriormente.

matriz\_contactos=simetriza(matriz\_contactos);

Por último se simetriza la variable “matriz\_contactos”, que tal como se había definido era triangular inferior, pero obviamente, debe ser simétrica, ya que el contacto entre elipses es bidireccional.

hold off

end

# COMENTARIOS FINALES:

No quiero dejar de comentar un aspecto que puede resultar algo extraño al ver el programa por primera vez, y es que las elipses se mueven hacia la derecha en el gráfico obtenido.

Esto es así porque el señor Alcibíades emplea en su teoría unos ejes x,y girados 90 grados en el sentido de las agujas del reloj. Por lo tanto, he creído oportuno ser consistente con dicha notación.

El eje x sería por tanto el eje vertical (positivo) hacia abajo. Por limitaciones del programa (que si no me equivoco, podrían ser solventadas con ciertas funciones de ploteo), este eje x es el habitual, dando lugar al efecto de que las elipses se mueven hacia la derecha y no hacia abajo.