UBA – Facultad de Ciencias Exactas y Naturales – Departamento de Computación

Metodos Numericos

Segundo cuatrimestre de 2017

6 de septiembre de 2017

Fotometria Estereo

1. Introducción Teórica

El foco de este trabajo practico es la resolucion de sistmas de ecuaciones lineales. La idea principal de la resolucion de dichos sistemas es despejar todas sus variables. En este trabajo pretendemos resolver sistemas lineales que tienen muchas incognitas lo cual hace que sea impractico intentar resolverlos en lapiz y papel. Razon por la cual utilizamos multiples metods numericos que simplifican la tarea de encontrar la solucion de estos sistemas. El metodo mas comun es el de eliminacion gaussiana que consiste en aplicar tres operaciones basicas una cantidad finita de veces para transormar al sistema de ecuaciones en otro equivalente(traingula el sistema) donde despejar las incognitas es una tarea trivial utilizando "backwards substitution". Estas tres operaciones basicas son multiplicar una fila por un escalar, sumar o restar un multiplo de una fila con otra y intercambiar filas. Siempre teniendo en cuenta que una fila no denota solo los coeficientes de las variables de una ecuacion sino tambien su termino independiente.

Aunque el metodo de eliminacion gaussiana es muy comun y bastante simple de implementar su complejidad temporal es O(n^3) y cuando tenemos que resolver multiples sistema de ecuaciones lineales esta complejida no es tan buena. Para estos casos, si las ecuaciones de los sistemas lineales tienen coeficientes identicos pero distintos terminos independientes, existe el metodo de factorizacion LU donde la L es una matriz triangular inferior con unos es su diagonal y U es una matriz triangular superior. Apesar de que encontrar la factorizacion LU de una matriz tiene costo O(n^3), la resolucion del sistema Ax = LUx = b tiene costo O(n^2), lo cual hace que en las situaciones donde hay que resolver muchas veces el sistema Ax = b para diferentes b sea mucho mas eficiente que utilizar el algoritmo de eliminacion de gauss. La demostracion de porque los algoritmos de eliminacion guassiana y el de factorizacion LU tienen complejidad O(n^3) y O(n^2) respectivamente, se puede ver en el libro Numerical Analysis de Burden.

La existencia de la factorizacion LU no esta garantizada para cualquier matriz. Dada una matriz inversible, las condiciones necesarias y suficientes para que exista la factorizacion LU son que no aparezca un zero como pivote en cada iteracion del algoritmo de eliminacion de gauss o que todas las submatrices principales sean inversibles.

Aunque la factorizacion LU no siempre existe, la factorizacion PLU si. Esta factorizacion consiste de tres matrices cuadradas P , L y U donde L es una matriz triangular inferior y U es una matriz triangular superior. La matriz P es designada una matriz de permutacion que es necesaria cuando al aplicar la eliminacion de gauss hace falta hacer intercambios de filas. A diferencia de la facotrizacion LU, donde la L tiene que tener unos en la diagonal, la L de la PLU puede no tener todos unos es su diagonal. Una propiedad importante de la matriz P es que siempre existe su inversa, P^{-1}, y ademas $P^{-1} = P^t$.

Un problema que no debe pasar desapercibido cuando trabajamos con operaciones aritmeticas en la computadora es el de perdida de digitos significativos. Esto ocurre porque la computadora trabaja con aritmetica de digitos finitos lo cual quiere decir que la representacion de los numeros en la computadora tienen una cantidad de digitos finitos para ser representados. Los errores mas comunes son el error de redondeo, el de cancelacion catastrofica y el error que obtenemos al dividir por un numero muy pequeno. El error de redondeo ocurre cuando trabajamos con un numero cuya representacion requiere mas digitos de los que disponemos en la computadora. El error de cancelacion catastrofica surge cuando restamos dos numeros que son tan parecidos que se se pierden demaciados digitos significativos. El problema que ocurre con una division donde el denominador es mucho mas pequeno que el numerador es que si el numerador tiene acumulado errores de operaciones anterirores la division por un numero mucho mas chico va a propagar esos errores.

En los algoritmos que nosotros implementamos abordar este problema es imprescindible ya que al computar el algoritmo de eliminacion de gauss se hacen alrededor de n^3 sumas, restas, diviciones y multiplicaciones, donde n es la cantidad de filas y columnas de la matriz asociada al sistema. La mejor forma que nosotros encontramos de acotar este tipo de errores es implementado el metodo de pivoteo parcial en nuestro algoritmo de eliminacion gaussiana. El pivoteo parcial consiste en que para cada iteracion del algoritmo el pivote sea el elemento mayor de la columna en la que se encuentra.

Otra factorizacion de matrices que utilizamos es la factorizacion LL^t, conocida como factorizacion de cholesky. Hay dos condiciones que tiene que complir una matriz para garantizar la existencia de esta factorizacion. Primero que sea simetrica y segundo que sea definida-positiva. Se dice que una matriz es definida positiva cuando $x^tAx > 0 \ \forall$ x $\in \mathbb{R}^{nxn}$.Una propiedad importante para destacar es que si tenemso una matriz A simetrica que tiene factorizacion LU entonces se puede demostrar que A se puede escribir de la siguiente manera, A = LU = LDL^t. Si ademas todos los elementos de su diagonal son positivos entonces se puede demostrar que A tiene factorizacion chelosky, osea que A se puede escribir de la siguiente manera A = LL^t. Cuando aplicable, esta factorizacion resulta ser bastante mas eficiente que la factorizacion LU.

La estabilidad numerica de un sistema lineal es un concepto sumamente importante en el momento de conciderar el error relativo que tiene la solucion que obtenemos del sistema. Para ver si un sistema es estable hablamos del condicionamiento del sistema. A esta idea la cuantificamos utilizando el numero de condicion κ(A) = ||A|| $||A^{-1}||$. Se dice que un sistema lineal esta bien condicionado si el numero esta lo mas cerca posible de 1 y mal condicionado si es mucho mas grande que 1.

Aunque no hay ningun criterio exacto para saber cual es el punto de quiebre entre mal y bien condicionado, segun nuestras clases teoricas de la materia, si el numero de condicion es mayor a 10,000, usualmente se puede decir que el sistema esta mal condicionado.

2. Desarrollo

El programa desarrollado se puede dividir en dos partes: calibracion del sistema y reconstruccion del modelo 3D. La primera parte consiste en conseguir las direcciones de fuentes de luz y elegir de estas las que nos permitan hacer una reconstruccion optima de los modelos. En la segunda parte utilizando las direcciones obtenidas durante la calibracion, se estiman las dirreciones normales de la superficie del modelo y con estas se obtienen las profundidades

2.1. Calibracion

2.1.1. Obtencion de direcciones de fuentes de luz

Dado que la superficie de los modelos es Lambertiana tiene la prodiedad de que absorbe la luz uniformemente en cada punto, con lo cual la normal en el punto con mayor intensidad de luz de la superficie corresponde con la direccion de la iluminacion. Usamos para la calibracion el modelo de la esfera porque al conocer su geometria y las intensidades en cualquier parte de su superficie podemos calcular las normales en todo punto

Para determinar la intensidad de un pixel utilizamos un promedio de los tres colores(azul, verde, rojo) dandole mas peso al verde dado que este es el color que el ojo humano percibe en mayor proporcion.

//Por tener la imagen con intensidades Discretas, no necesariamente habria una con el mayor valor.

Nuestra primera estrategia para encontrar el punto con mayor intensidad de luz fue recorrer los pixeles de las imagenes uno por uno y guardar la posicion del de mayor intensidad. El problema que tuvimos es que como la esfera no es de color uniforme, pueden haber puntos que sean mas brillantes por tener un tono de color mas claro y no por ser el punto con mayor intensidad luz.

La segunda estrategia fue considerar vecindades fijas que es un poco mas simple que tomar vecindades que se pudieran contraer o expandir(ya que en los bordes esto podria fallar), obteniendo una mejor aproximacion al pixel buscado.

2.1.2. Eleccion de las direcciones de fuente de luz

Para eleccion de las direcciones de fuente de luz: Lo primero fue hacer las diferentes permutaciones de tres direcciones de luz sin tomar elementos repetidos y aplicar el algoritmo de la eliminacion de gauss para ver si la matriz es inversible o no. Una vez corroborado que la matriz fuera inversible, mediante el numero de condicion asociado a la matriz podemos discriminar entre las matrices para una mejor estimacion del valor, mirando el numero de condicion que esta relacionado con la estabilidad numerica de un sistema lineal ya que una matriz mal condicionada es propensa a tener graves errores y no darnos soluciones del sistema en las cuales podemos confiar. Nosotros supones que el error que vamos a obtener al calcular las normales va a estar directamente relacionado al condicionamiento de la matriz que utilizamos. Dicho de otra manera, el mejor gráfico de las normales va a estar dado por la matriz mejor condicionada. Para corroborar esto buscamos la matrices con el máximo y mínimo grado de condicionamiento y comparamos visualmente las diferencias entre los campos normales que obtuvimos utilizando cada una.

Con eso calculamos el numero de condicion de cada matriz, para luego tomar los maximos y minimos para poder comparar los resultados.

2.2. Construccion de la matriz M

Dadas las ecuaciones 11 y 12, para generar el sistema de ecuaciones para todos los pixeles se pueden ver de dos formas para poder generar la matriz banda.

2.2.1. 1 er Forma

La primera que pensamos,fue dar una relacion de orden (entre los pixeles, apartir de la relacion de orden de los enteros), verlos como un grafo dirigido y tomar la matriz asociada al grafo. Para la cual ordenamos los pixeles para que se recorran de forma univoca, donde la relacion de orden estaba dada por la coordenada en Y y luego x , con lo cual (i,j) ¡(k,l) ¡=¿(l=¿j)——(k =¿i), con esto logramos obtener la matriz banda(anhelada). //Creo que no deberia ir ya que no vimos nada en la materia y deberiamos justificarlo de alguna forma (teorica)

2.2.2. 2 da Forma

La segunda forma (naiv), para la cual tomamos la misma relacion anterior, es ver la grilla de pixeles en forma de F1++F2++...++Fn concatenacion de los pixeles (donde F1 contiene a todos los pixeles correspondiente a todas las columnas de la fila 1 de la grilla, analogamente Fn). con los cual, si separamos las ecuaciones y formamos 2 matrices una para las ecuaciones correspondientes al eje X y otra matriz correspondiente al eje Y. Tenemos una Matriz Banda 1x2 Para el eje

X y 1x(cantidad De Columnas de las grilla +1) para el eje Y. Combinandolos de manera adecuada, se obtiene la matriz banda(anhelada).(X1/Y1/.../Xn/Yn)

MxMt es simetrica, es valido para toda matriz.

2.2.3. Descripcion de la matriz M

La matriz M tiene dimension 2NxN y es banda PxQ, por lo tanto Mt es de dimension Nx2N y es banda QxP, luego siendo A = MxMt tiene dimension NxN y es banda Qx(Q+1). Ya que como esta armada la matriz M, se forma una escalera donde al moverse por las columnas,mirando desde la q+1 columna, la primer coordenada tiene el primer elemento igual a cero y todas los demas no necesariamente distinto de cero.Al mirar la siguiente columna la cantidad de elementos que no son cero incrementan en dos y asi cada vez que se mira la siguiente columna. Luego al mirar las filas, se logra ver un patron en el cual se agrega un cero a las primeras coordenadas de la fila cuando se dan pasos de a dos filas. Lo cual forma un patron interesante al ser multiplicado por su transpuesta, esto relaciona la cantidad de pixeles con el ancho de banda de la matriz el cual queda dependiendo de la cantidad de pixeles y del ancho de la imagen.

2.3. Descripcion de la matriz A

2.3.1. Mirando ceros de la parte superior

Para la matriz A resultante al juntar lo visto, y siguiendo la misma idea con la matriz transpuesta, se observan los mismos patrones.Por lo tanto al hacer la multiplicacion, se puede observar un patron que siguen cada una de las filas de la matriz resultante , en particular la fila i(i ¡=Q) tiene sus coordenadas igual a cero desde la posicion Q+i+1, eso forma una diagonal de elementos no necesariamente igual a cero a partir de la fila 1 columna q.

2.3.2. Mirando los ceros de la parte inferior

Para la fila i(i ¿Q), tiene el primer cero a partir de la fila q+1 en la coordenada 1 y va incrementando un cero a las primeras coordenada del vector fila cada ves que avanza, con lo cual eso forma una diagonal de elementos no necesariamente igual a cero a partir de la fila q columna 1.

2.3.3. Forma de la matriz resultante

Para obtener la matriz A que da la multiplicacion de la matriz MxMt, dado que son matrices banda y que M es banda PxQ donde p=cantidad de elementos(pixeles)-1 y Q = cantidad de columnas de la imagen. Si la imagen es una tira de pixeles entonces P = Q, sino P ¿Q. La matriz resultante tiene dimension NxN y es banda Qx(Q+1).

5:Dudas. ¿Porque no importa la cantidad de pixeles a la hora de ver la anchura de la matriz?.

2.4. Justificacion de utilizacion de la factorizacion de cholesky Para la matriz A

Para poder justificar cholesky, nos alcanza con demostrar que las columnas de la matriz M son Li ya que por la definicion de matriz definida positiva, vale que para $w^tAw > 0$ ∀ w ∈ $\mathbb{R}^{nxn}$, la demostracion es por el contraresiproco con lo cual suponemos lo contrario existe un $(\exists w)Aw = 0 \wedge w \neq 0$, como A = $M^tM \rightarrow w^tM^t = (Mw)^t(Mw) =||Mw||^2$ y como esto es la norma del vector, la norma de ese vector es 0, como demostramos que las columas eran Li, con lo cual llegamos a un absurdo que provino de suponer que existe tal w, por lo tanto la matriz A es definida positiva. Y usando la Propiedad*, tenemos que la matriz A tine factorizacion cholesky unica.

Propiedad*: Una matriz es definida positiva ¡=¿tiene factorizacion cholesky unica.

//

En el desarrollo de este trabajo practico lo primero que hicimos fue la calibracion del sistema. Lo cual consistio en encontrar las coordenadas del punto mas brillante de 12 imagenes de una esfera. Luego utilizamos la coordenadas del punto mas brillante junto con la ecuacion de la esfera para poder encontrar la direccion de la luz incidente a ella.

Para encontrar el punto mas brillante, nuestra primer idea fue analizar uno por uno todos los pixeles de cada imagen hasta encontrar el pixel con la máxima intensidad en el rango de 1 a 255, pero nos dimos cuenta que esta no era la mejor estrategia. El problema que surgió al tratar de seleccionar el pixel mas brillante de esta manera es que había mas de un pixel con el máximo valor de brillo en una imagen y en estos casos decidimos elegir uno de todos los pixeles con mayor intensidad al azar. Esto llevo a que en muchos casos eligiésemos mal el puntos mas brillante. Lo que no nos habíamos dado cuenta al principio fue que las imágenes discretisan el espacio y esto es lo que lleva que aya mad de un pixel que tengan la máxima intensidad de la imagen, pero esto no necesariamente quiere decir que la normal de todos estos puntos sean la dirección de la luz. Ademas no es muy lógico elegir el punto mas brillante al azar ya que este representa la dirección de la iluminación la cual es unica. Luego pensamos en que el brillo forma una aureola y que la esfera tiene algunas irregularidades sobre su superficie, entonces no alcanza con buscar el pixel mas brillante, por lo tanto se utilizamos las vecindades para lograr agrandar el espectro. Una vez seleccionadas las coordenadas del punto mas brillante de una imagen utilizamos la ecuación de la esfera, r^2=$(x - x_0)^2$ + $(y - y_0)^2$+ $(z - z_0)^2$, para obtener las direcciones de la iluminación. De esta ecuacion los parametros que conocemos son las coordenadas (x_0,y_0) del centro de la esfera, el radio r y las coordenadas del pixel que consideramos mas brillante (x, y). Si C es el centro de la esfera y P el punto donde se encuentra el pixel mas brillante, entonces el vector que representa la dirección de la iluminación es C-P = (x_0 - x, y_0 - y, z_0 - ($\sqrt{r^2 - (x - x_0)^2 - (y - y_0)^2 + z_0}$)) = ($x_0$ - x, y_0 - y, $\sqrt{r^2 - (x - x_0)^2 - (y - y_0)^2}$).

Una vez obtenidos las direcciones de iluminacion, el siguiente paso del desarrollo fue elegir las tres direcciones de luz para después obtener las normales. Nuestra idea fue elegir aquella combinacion de direcciones que tengan el menor número de condición. Para hacer esto programamos un algoritmo que calcula todas las posibles permutaciones de tres vectores, teniendo 12 vectores para elegir, junto con el numero de condición asociado a cada una de estas matrices. Hay que tener cuenta que al calcular las posibles permutaciones de las direcciones consideramos a una elección de tres vectores como un conjunto, ya que el numero de condicion no cambia si permutamos las columnas o filas de una matriz. El condicionamiento de una matriz, dado por su numero de condicion, nos pareció la mejor forma de escoger las direcciones ya que, por lo que aprendimos en clase una matriz mal condicionada es propensa a tener graves errores y no darnos soluciones del sistema en las cuales podemos confiar. Pensando en el numero de condiciones en términos geométricos, también nos dimos cuenta de que el condicionamiento de una matriz va a mejorar a medida que las ecuaciones lineales de la matriz esten lo mas lejano posible de ser paralelas, lo cual tiene sentido ya que tener iluminación dispersa ayuda a ver un objeto mejor en el mundo real. Nosotros supones que el error que vamos a obtener al calcular las normales va a estar directamente relacionado al condicionamiento de la matriz que utilizamos. Dicho de otra manera, el mejor gráfico de las normales va a estar dado por la matriz mejor condicionada. Para corroborar esto buscamos la matrices con el máximo y mínimo grado de condicionamiento y comparamos visualmente las diferencias entre los campos normales que obtuvimos utilizando cada una.

El procedimiento lo hicimos en C++. Para conseguir todas la conbinaciones posibles de las direcciones de luz pusimos en una lista todas las filas (direcciones de luz) sin ninguna repetición. A esta lista se la recorre con 3 indices, los cuales nunca comparten el mismo valor y nunca se cruzan por lo tanto no se crean nuevas matrices con filas conmutadas, ni matrices con filas idénticas. Al mismo tiempo se le calcula el numero de condición correspondiente a cada matriz y se guarda solamente la que posee el numero de condición menor.

Con las tres direcciones de iluminación ya elegidas el siguiente paso del desarrollo es el calculo de las normales en cada pixel. Esta es la etapa del trabajo practico en la cual implementamos y utilizamos los algoritmos de eliminación de gauss y la factorizacion LU.

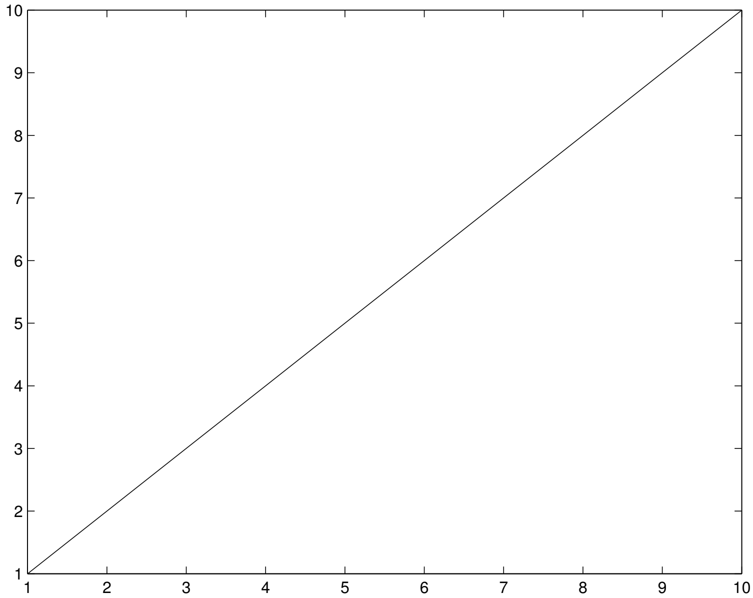
Nuestro algoritmo de eliminación gaussiana, hace n iteraciones donde n es la cantidad de filas de la matriz a triangular. En la i-esima iteración toma el elemento a_{ii} como pivote. Si el pivote en cualquiera de las iteraciones es zero entonces no existe factorizacion LU pura, aunque si existe factorizacion PLU. Si el pivote no es cero el algoritmo hace otras k iteraciones donde k es la cantidad de filas que tiene por debajo. El algoritmo procede a calcular el múltiplo exacto por el cual tiene que multiplicar al pivote para poder eliminar al elemento, que este en la misma columna que el, y en la fila debajo si. Esto lo repite k veces, una ves para cada fila que se encuentra por debajo del pivote actual. Una vez terminadas las k iteraciones para el i-esimo pivote, el algoritmo incrementa en 1 a i y pasa al siguiente pivote. Resulta evidente que los pivotes va a ser todos los elementos de la diagonal.

Obtener la factorizacion LU no es mas que guardar los múltiplos que utilizamos para diagonalizar una matriz en el algoritmo de eliminacion gaussiana. Es importante notar que nosotros implementamos un algoritmo de factorizacion PLU donde la factorizacion LU es el caso especifico donde la matriz P es la matriz identidad. Por lo cual, la complejidad temporal de obtener la factorizacion LU es igual a la complejidad del algoritmo de eliminacion de gauss, la cual es O($(n)^3$).

Aunque sus complejidades son iguales, la diferencia se encuentra al momento de resolver varios sistemas. La eliminación gaussiana va a tener la misma complejidad cada vez que la utilizamos. En contraste, la factorización LU solo se obtiene una vez y luego solo se debe resolver dos sistemas triangulares, esto lleva a que su complejidad es O($(n)^2$). Como dice el enunciado del TP, los valores sx,sy,sz, de la ecuación 5 no cambian pixel a pixel sino que solamente se modifica el valor de la intensidad en (x,y) para la imagen correspondiente Ii, y por lo tanto poseen la misma matriz pero con distinto término independiente. Por lo cual, si se encuentra una convinación de s_1, s_2 y s_3 que posea factorización LU se podria bajar sustancialmente la complejidad de obtener las normales de una imagen, ya que estas poseen una gran cantidad de pixeles.

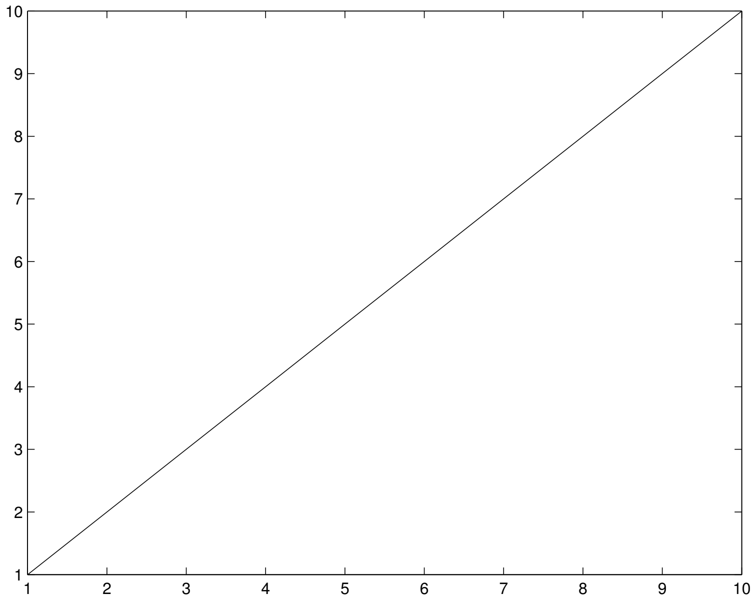
Nuestra primer idea para tratar de comprobar nuestra hipótesis de que la factorizacion LU es mas rápida que la eliminación de gauss fue tomar distintos tamaños de matrices y trangularlos una vez con cada algoritmo para ver que método era mas rápido. Este experimento no pone a prueba nuestra hipótesis porque lo que intentábamos comprobar es que la factorizacion LU es mas rápida que la eliminación de gauss para resolver un sistema lineal despues de la primera vez que se resuelve el sistema. Si resolvíamos una sola vez cada tamaño de matriz, ambos algoritmos tardarían lo mismo. Por lo tanto, decidimos hacer un experimento donde mantuvimos constante el tamaño del sistema lineal a resolver y lo que variamos en cada instancia de teste fueron las cantidades de términos independientes. Por ejemplo, la primera instancia resolvimos el sistema Ax=b con 100 b diferentes y luego incrementamos la cantidad de términos independientes de 100 en 100, hasta llegar a 1000. Es importante remarcar que estos términos independientes fueron generados aleatoriamente con una función encontrada en la siguiente pagina:

SANTI TENES QUE PONER LA PAGINA ACA!!



3. Resultados

Hicimos la elección de direcciones de iluminación con el procedimiento explicado en la parte de desarrollo y obtuvimos la matriz 0,4,10 que tiene el minimo numero de condición y la matriz 3,4,8 que tiene el máximo numero de condición. Luego generamos los campos de las normales y obtuvimos los siguientes resultados:



4. Discusión

Aca escribimos la discusion.

5. Conclusiones

Aca escribimos las conclusiones.

6. Apéndices

Aca escribimos el apendices.

7. Referencias

Aca escribimos las referencias.