

MEMÒRIA I MANUAL DEL SISTEMA  
DE CàLCUL DE MASSES I SUPERFÍCIES DE FIGURES 3D

Carles Pina i Estany  
Gener 2007  
Matemàtiques

# Índex de continguts

1	Objectiu de l'aplicació.....	3
2	Integració numèrica amb ordinador.....	4
2.1	Introducció (integració tri-dimensional general).....	4
2.2	Optimitzacions.....	5
2.3	Algorisme càlcul de masses.....	6
2.4	Masses calculades d'altres maneres.....	8
2.4.1	Con-con.....	8
2.4.2	Cilindre-cilindre.....	9
2.4.3	Comentaris del mètode.....	9
3	Integrals de superfície.....	10
4	Comentaris informàtics.....	10
5	Manual d'ús.....	11
6	Línies de futur.....	22
7	Comentaris.....	24
8	Comentaris personals.....	24
8.1	Pedagogia.....	24
8.2	PFC.....	24

# 1 Objectiu de l'aplicació

El motiu principal i final de l'aplicació és mostrar que he assimilat els conceptes de l'assignatura de matemàtiques de forma correcta per aprovar l'assignatura.

Per mostrar això es van marcar uns objectius per l'aplicació: fer integrals de massa i de superfície, interseccionant diversos cossos. Deixant introduir la densitat a l'usuari, posicions de les figures, etc.

L'aplicació principalment és una mostra de com fer integrals triples per calcular la massa, on els límits venen determinats pels cossos. Per tant, l'entrada de l'aplicació són els cossos i paràmetres d'aquests i la sortida és la massa de la intersecció tenint en compte la funció densitat així com la superfície d'intersecció.

L'aplicació també fa una gràfica-mostra dels cossos, per tenir una idea de com és el dibuix. La gràfica cal veure-la com orientativa. Les funcions són pintades correctament, però hi pot haver problemes escalant els eixos o sobretot amb la zona pintada de l'eix Z.

En aquesta memòria s'intentarà explicar què s'ha fet, com s'ha fet, alguns problemes trobats i com s'han resolt. També una breu part informàtica, línies de futur, etc.

## 2 Integració numèrica amb ordinador

En aquest apartat s'explica com s'ha realitzat la integració numèrica amb integrals triples. De totes maneres, el comentat és fàcilment aplicable a integrals dobles, simples, etc.

### 2.1 Introducció (integració tri-dimensional general)

Integrar simbòlicament, a mà, pot ser molt difícil. Alguns cops s'ha sentit "Integrar és un art, derivar una tècnica". Es poden tenir una sèrie de regles i passos per fer integració simbòlica informàticament, però sembla que només un programa ho té implementat (Mathematica). Els altres programes "matemàtics" fan integració numèrica (i com veurem fan una integració numèrica més limitada que el programa aquí realitzat)

En el cas de fer integrals de massa, es farà una integral triple (sigui en coordenades cartesianes, paramètriques o esfèriques). El concepte és, a cada punt X, Y, Z (o variables corresponents pel sistema de coordenades) dins el volum interseccionat calcular el valor de la funció densitat (entrada per l'usuari)

Computacionalment això és un procés molt costós ja que cal tenir 3 bucles anuats. Per exemple, si volem calcular la densitat d'un cub de 10 x 10 x 10 i agafant uns diferencials de volum de 0.001 cal fer:

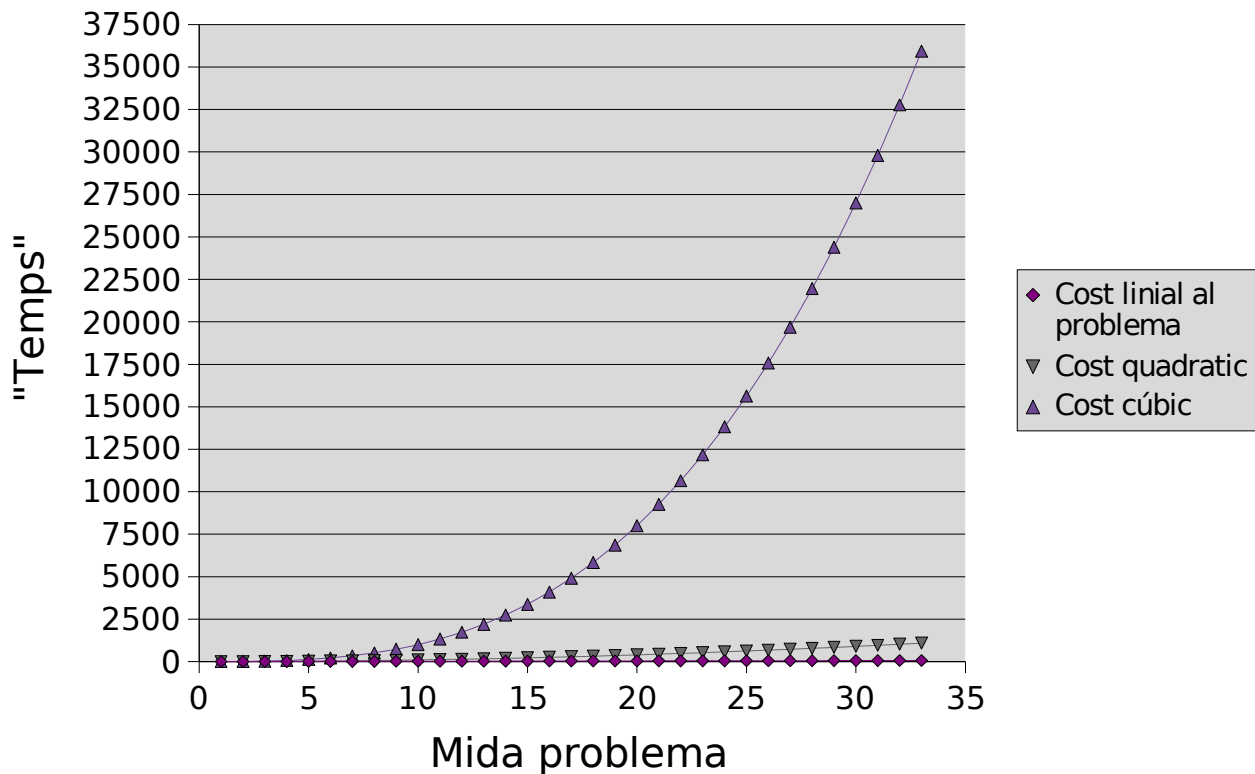
- Iterar de 0 a 10 cada 0.1 per l'eix X
  - Iterar de 0 a 10 cada 0.1 per l'eix Y
    - Iterar de 0 a 10 cada 0.1 per l'eix Z
      - Massa acumulada = densitat(X,Y,Z) \* 0.001
    - Z=Z+0.001
  - Y=Y+0.001
  - Z=0
- X=X+0.001
- Y=0
- Z=0

Computacionalment té un cost cúbic ( $X^3$ ).

A continuació s'ensenya una comparació de costos. Hi ha tres algorismes a la mostra:

- Un de cost lineal (per exemple, integral simple),
- algorisme de cost quadràtic (integral doble),
- algorisme de cost cúbic (integral triple)

# Comparació costos



Què indica aquesta gràfica?

Per exemple, podem pensar en tenir un problema de mida 15 i veiem que el cost cúbic és força més elevat que el quadràtic. Podem pensar que el cúbic serien 5000 segons (1h 20 min. aprox.) i el quadràtic comparativament despreciable.

Què passa si tenim un problema de mida 30, és a dir, només el doble. L'algorisme quadràtic tardaria una mica més, però el cúbic (integral triple) tardaria molt més: uns 5000 segons (6 hores i quart) .

Això indica que cal anar amb molt de compte amb la resolució que l'usuari escriu. Més precisió, molt més temps. Modificar la resolució de 0.1 a 0.01 pot incrementar el temps de resolució de segons/minuts a hores.

Comentari: té un cost molt alt en temps, però no en memòria. La memòria que es fa servir és la mateixa per una mida de problema que per una altra mida de problema.

## 2.2 Optimitzacions

Si només s'haguessin fet càlculs de volums és fàcil optimitzar l'algorisme. Si pensem una integral en 2D, enlloc de calcular l'àrea amb rectangles de base  $dx$  es farien trapezoides, maximitzant l'àrea calculada a cada iteració.

Aquí, per calcular el volum d'una esfera (figura en 3D) es podria maximitzar el paral·lelepípede més gran inscrit a l'objecte 3D, per després fer més paral·lelepípedes de mides màximes inscrites a les altres zones. D'aquesta manera, el cost seria MOLT menor (segurament logarítmic, menor que lineal).

No s'ha pogut fer cap optimització perquè en aquest cas s'avalua la massa, i cal avaluar la densitat per tots i cada un dels punts interiors de la figura. Si la funció

densitat tingués propietats concretes (simetries, no canviés en una direcció, fos constant, etc.) també es podria optimitzar.

Evidentment es podrien tenir dos algorismes: el general que sempre val i alguns optimitzats. No s'ha fet per considerar-ho innecessari en aquesta aplicació i pel propòsit que l'ocupa.

### 2.3 Algorisme càlcul de masses

A continuació es mostra l'algorisme d'integrals triples utilitzat per calcular les masses.

Aquesta és una versió de mostra, simplificada però totalment funcional. El llenguatge de programació és Python (veure secció "Comentaris informàtics").

```
#!/usr/bin/python

import math

def jacobia(x,y,z):
    return z*z*math.sin(y)

def densitat(x,y,z):
    return 1

#Exemple: calcul volum d'una esfera a partir de l'integral
R=1

xinf=0
xsup=2*math.pi

yinf=0
ysup=math.pi

zinf=0
zup=R

#inicialitzacions valors mínims
x=xinf
y=yinf
z=zinf

resolucio=0.02
total=0

dvolum=resolucio*resolucio*resolucio

while x<=xsup:
    while y<=ysup:
        while z<=zsup:
            total=total+jacobia(x,y,z)*densitat(x,y,z)*dvolum
            z=z+resolucio
        y=y+resolucio
        z=zinf
    x=x+resolucio
    y=yinf
    z=zinf

print "Total: ",total
```

Veure que aquest algorisme és l'aplicació directe del concepte d'integral. A l'exemple concret ens calcula el volum d'una esfera de radi 1. Observar que conceptualment és una integral en esfèriques (el plantejament dels límits, el Jacobià, etc.):

- X és Theta (de 0 a  $2\pi$ , equador)
- Y és Ro (de 0 a  $\pi$ , meridians)
- Z és radi (de 0 a radi)

Se segueix l'algorisme vist en apartats anteriors (els 3 whiles aniuats).

La línia:

```
total=total+jacobia(x,y,z)*densitat(x,y,z)*dvolum
```

Fa que el nou total és l'anterior més l'avaluació de la funció  $jacobia(x,y,z)$ , multiplicat per  $densitat(x,y,z)$  i multiplicat pel diferencial de volum. Si s'integra en coordenades cartesianes el jacobia és 1. Si es vol calcular el volum de la figura, la densitat serà 1.

Si executem el programa, i trigant uns 5 segons en un ordinador modern, ens ensenya per pantalla:

```
Total: 4.0747340362
```

Sabem que el volum d'una esfera és  $\frac{4}{3}\pi R^3$ . Si  $R=1$ , el resultat correcte és 4.18. Veiem que hi ha una lleugera diferència, culpa que els "diferencials" aplicats són de  $0.02^3$  u<sup>3</sup>, enlloc de ser realment infinitesimals.

Una petita taula amb 3 mides de diferencials, el resultat i el temps:

Mida diferencial	Resultat, error	Temps
0.02	4.07 (0.11)	5 segons
0.01	4.18 (0!)	40 segons
0.005	4.15 (0.03)	5 min 20 segons

Seria convenient fer més proves. El resultat final (provant amb diferencials més i més petits) és que a més resolució, més precisió. A més a més, i tal com s'intueix, segurament uns cops s'excedeix i altres cops es queda amb un valor inferior, segurament oscil·lant (com una funció oscil·lant amortiguada, tendint a error 0 per Mida diferencial tendint a 0). Ull, perquè el temps està creixent<sup>1</sup> molt, tal com s'havia predit en el punt "Cost de l'algorisme".

Evidentment si es pensés aquesta mateixa integral amb coordenades cartesianes, el jacobia a cada punt seria constant (1) i per tant es podria simplificar (per exemple, buscant un cub inscrit a la figura a integrar). D'aquesta manera s'aconseguiria un resultat molt més acurat amb molt menys temps.

#### Comentari:

Volia contrastar els meus resultats en uns casos no trivials (l'esfera és fàcil contrastar el resultat). La idea era verificar el mateix mètode amb algun altre program. Es va investigar els programes Matlab, Octave i Mathematica. Sembla que cap dels 3 programes és capaç de resoldre numèricament integrals tridimensionals amb límits variables. Entenc com a límit variable que el límit de la integral Z (per exemple) depèn de X i Y, de fora cap a dins. El programa aquí

<sup>1</sup> Aquest és un cas típic que una persona va molt més ràpid que un ordinador: la persona pot fer la integral indefinida i avaluar-la amb un temps finit, en canvi amb aquest sistema si el diferencial fos realment tendint a 0 no acabaria mai

plantejat sí que ho fa.

Per totes les figures, el programa incorpora els límits d'integració paramatritzats en funció a les dades que entra l'usuari per tal de realitzar la integral que pertoca.

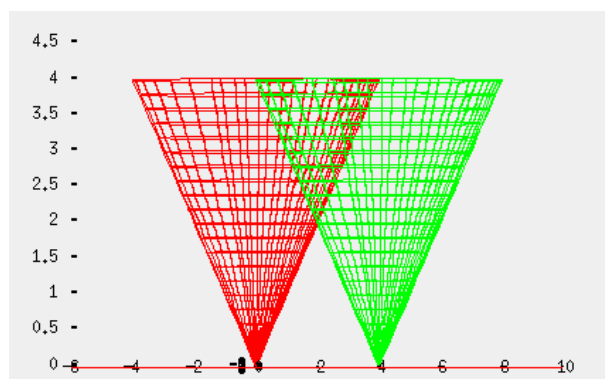
## 2.4 Masses calculades d'altres maneres

Hi ha hagut dos volums que es calcula la massa sense seguir el plantejament anterior (és a dir, sense fer una integral triple amb els límits que pertoca). Es descriu el mètode aplicat a continuació. Les figures són con-con i cilindre-cilindre.

### 2.4.1 Con-con

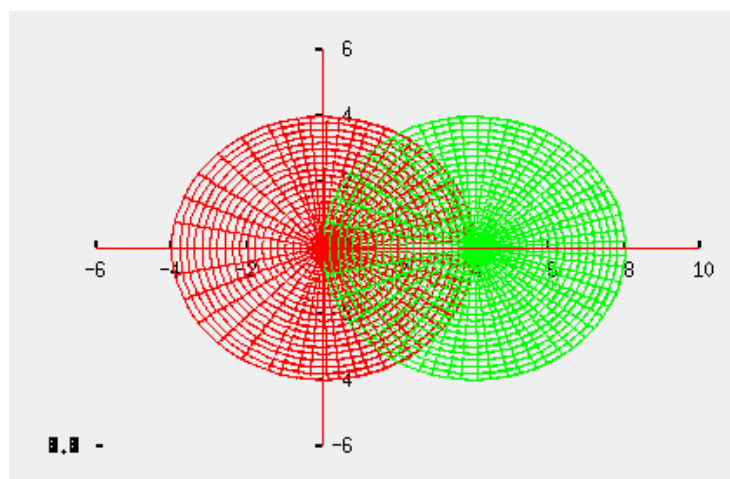
En el con-con es van buscar els límits, però sembla que vaig cometre algun error: donava una arrel amb operand negatiu. Però calia trobar una altra solució per poder calcular la massa.

Cal visualitzar el següent gràfic:



L'algorisme a seguir, si fos en 2D, hauria estat buscar els punts de la figura que corresponen als dos "triangles" (per exemple, podríem dir que la imatge del punt X de les dues funcions sigui menor que 4). Si un punt X forma part dels dos cons, cal calcular l'aportació de massa aquesta columna entre  $Y = \text{punt\_inferior}$  fins a  $Y = \text{punt\_superior}$  (alçada dels "triangles"/cons, 4 a l'exemple)

Evidentment, s'ha generalitzat aquest algorisme 2D (que només fa cerques en X) per un algorisme per tenir tots els punts que formen part dels dos cons. Es fa un escanor X-Y:



D'aquesta manera poder calcular tota l'aportació en massa de tots els punts X-Y que formen part de les dues figures. Sempre, des de Z inferior a Z superior, i sempre cal tenir una funció que ens digui si és cert o fals que un punt X, Y pertany a les dues funcions.

La manera fàcil de fer proves amb aquest algorisme és superposar els dos cons i fer densitat=1. La massa, en aquest cas, és el volum del con ( $1/3 * \pi * r^3 * h$ ). realment funciona. O posant que la densitat és "x" a dos cons centrats a l'origen la massa és 0 ja la meitat esquerra del con té massa negativa i la meitat dreta del con té massa positiva.

## 2.4.2 Cilindre-cilindre

Un cop es va tenir el con-con, s'ha aplicat una tècnica semblant pel cilindre-cilindre. En el cas del cilindre-cilindre es deixa que tinguin una alçada arbitrària. En el cas del con-con també es podria haver fet, ja que no afegeix cap complexitat: només cal calcular la Z fins a l'alçada menor dels dos cons (per tenir la intersecció de cons).

## 2.4.3 Comentaris del mètode

Es podria haver aplicat aquest mètode per totes les figures (i hagués estat molt directe!). Dona uns resultats acurats, i és molt independent de les figures: només cal "escanejar" una zona de l'espai i cal saber si cada punt està o no està dins les figures (i si el punt X, Y, Z està dins la figura doncs només cal avaluar la funció densitat i sumar a la massa anterior).

Es comenten uns pros i contres del mètode. Pros:

- Com ja s'ha comentat, pot calcular la massa de formes arbitràries. És igual quina forma hi hagi, si és descrita per equacions es podrà saber si el punt està dins la figura.
- Com a conseqüència de l'anterior, no cal calcular límits per les figures en concret, menys feina previa a la resolució

Contra:

- En principi és l'usuari qui defineix la presumpte zona on hi haurà la intersecció. L'usuari cal que prengui certes decisions (que tingui cert sentit comú o coneixements). Tot i això, amb la gràfica de les figures ja es pot avaluar prou bé. Si els límits són molt/massa grans, el programa pot trigar massa temps per calcular la integral. Si es tenen els límits d'integració, l'algorisme és més òptim.

Nota:

- És relativament fàcil acotar informàticament la zona d'intersecció de les figures. Per exemple, fer una búsqueda dicotòmica en diverses zones predefinides, o molt millor, buscar analíticament els punts de tall de les figures per tenir un marge d'error inferior. No s'ha fet en aquest programa però es podria fer en un futur.

S'ha estat temptat, un cop s'ha provat aquest mètode, de fer que totes les integrals de massa fossin resoltes amb aquest mètode. De totes maneres no s'ha fet: l'altre mètode és més ràpid, no cal entrar límits i a més a més ja estava tot fet.

### 3 Integrals de superfície

Hi ha un cas que la integral de superfície és fàcil: la del casquet. Feta a l'assignatura de matemàtiques i comprovat amb els exercicis de l'assignatura. Com que cal fer una integral doble s'ha adaptat l'algorisme de la triple.

Per fer les integrals de superfície cal imaginar que els dos cossos són “buits” (en contraposició de les integrals de massa/volum, que estem calculant la massa que seria comuna als dos cossos).

Tenint dos cossos “buits” interseccionats dona com a resultat una corba. El què cal fer, és calcular la superfície de la corba (que sempre serà tancada).

S'ha provat de tenir un algorisme (informàtic) genèric de càlcul de superfícies, però no ha donat resultats prou satisfactoris. Si s'hagués trobat hagués estat relativament fàcil aplicar-ho a totes les figures, però les proves que s'han fet a volums simples no ha donat suficientment correcte. Caldria, segurament, treballar-hi més encara més.

La solució adoptada és una solució de compromís:

- En algunes figures, s'ha pogut fer el càlcul de forma general (amb una integral de superfície) i aquest resultat és el que es calcula i s'ensenya a l'usuari. Això és el què es volia, no hi ha cap problema.
- En altres volums es calcula la superfície amb unes posicions de les figures predefinides que faciliten el càlcul (per exemple, cilindre-esfera). En altres figures s'ha fet de les dues maneres i tot (tradicional i en posició especial, per exemple, cilindre-cilindre).

De totes maneres, per casos límit sembla que pot fallar el càlcul de superfície.

### 4 Comentaris informàtics

Es deixen uns comentaris de la part més informàtica de l'aplicació (llenguatge, llibreries, etc.).

- El programa s'ha fet amb el llenguatge Python. Python és un llenguatge de programació interpretat, lliure (es pot fer servir sense limitacions) i una característica important que s'ha aprofitat és que és trivial poder demanar una expressió a l'usuari i avaluar-la. Això és el què es fa demanant l'expressió densitat i avaluant-la. Amb altres llenguatges de programació hauria calgut fer un analitzador d'expressions (coneguts com parsers). Fer un analitzador d'expressions ben fet hagués estat prou difícil (temari de l'assignatura de Compiladors de cinqué d'informàtica) o hagués calgut utilitzar llibreries externes.
- Python és orientat a objectes, això ha facilitat el disseny de l'aplicació.
- Python és un llenguatge de programació multiplataforma. L'aplicació funciona tan amb entorn Microsoft Windows com en entorn GNU/Linux. També podria funcionar amb Mac OS X.
- Per limitacions amb el sistema de de dibuixat, hi ha figures que poden quedar “deformades”. Es realitza un auto-escalat de la zona a representar, però l'escala de les X pot ser diferent a l'escala de les Y i així es pot arribar a veure les figures amb forma incorrecte (per exemple, una esfera

representada en forma el·lipsoide i no esfèrica)

- Les llibreries de programació gràfica són les Qt. Són unes llibreries lliures, amb llicència lliure i de contrastat ús en varis entorns. Treballen molt bé conjuntament amb Python.

## 5 Manual d'ús

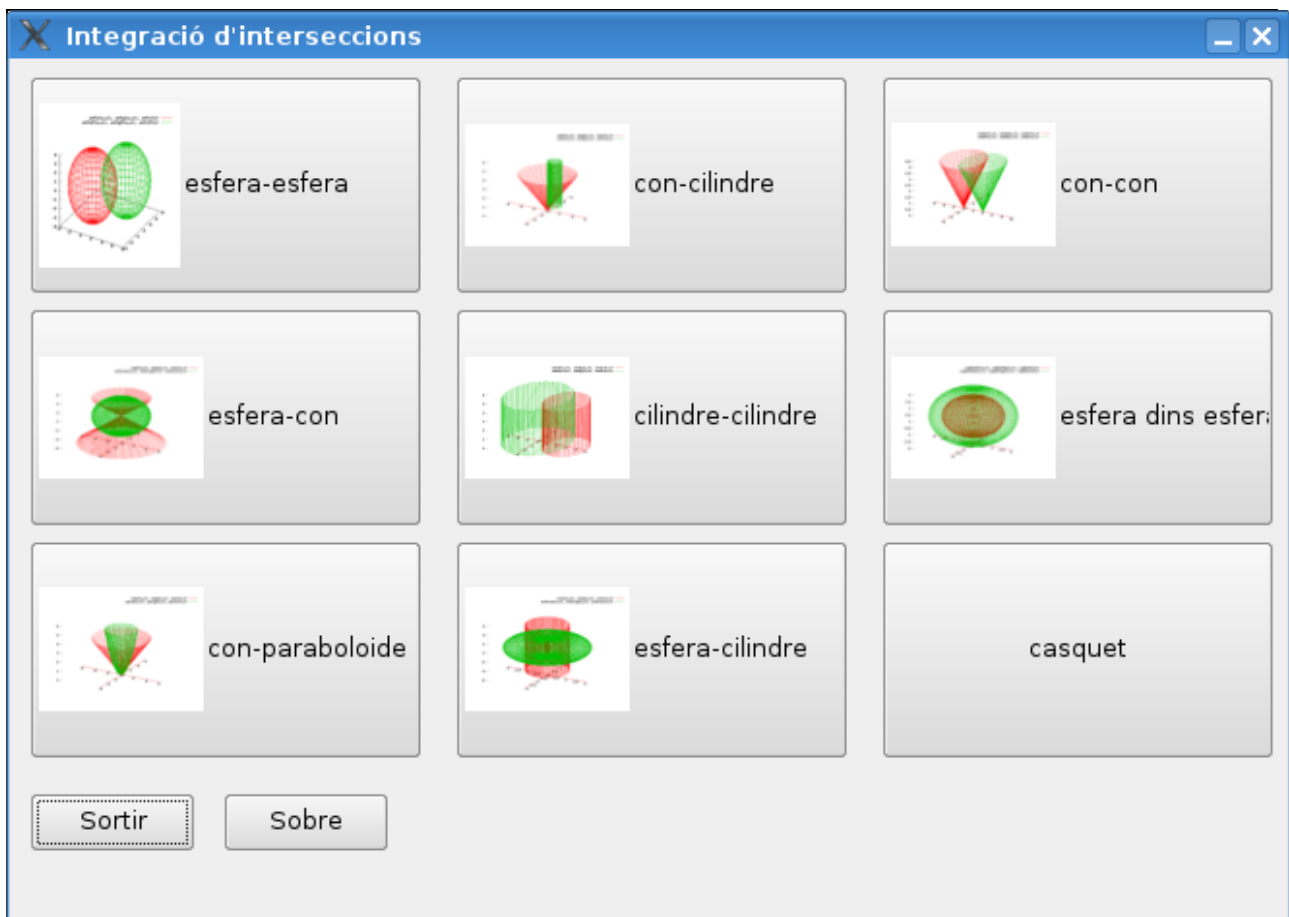
### 5.1 Instal·lació

- Descarregar el fitxer .zip de <http://salle.url.edu/~is08139/mates/>
- Descomprimir el fitxer zip a algun directori
- Executar (doble clic) el fitxer **main.exe**

### 5.2 Ús del programa

En aquest punt es veuen algunes de les pantalles de l'eina informàtica. És una guia per entendre com funciona l'aplicació així com a exemple de diverses figures.

Les captures de pantalla d'aquest document han estat fetes en una versió anterior a la definitiva, hi pot haver hagut canvis menors. A la versió definitiva hi ha alguns canvis a nivell d'interfície d'usuari, nous botons, a les captures de pantalla falten resultat de superfície, etc.



Pantalla principal de l'aplicació En aquesta pantalla es pot escollir quina operació es vol realitzar (apart de sortir o una breu informació sobre el programa)

Nota: Només en el cas d'interseccionar esferes només deixa calcular el volum, per tant no hi ha funció densitat. Totes les altres figures es calcula la massa (per tant, densitat). El motiu és que va ser la primera figura a realitzar-se i no s'ha afegit la massa (no seria problema fer-ho, sobretot després de veure el punt 2.4).

Nota: Només en el cas d'interseccionar esferes només deixa calcular el volum, per tant no hi ha funció densitat. Totes les altres figures es calcula la massa (per tant, densitat). El motiu és que va ser la primera figura a realitzar-se i no s'ha afegit la massa (no seria problema fer-ho, sobretot després de veure el punt 2.4).

**Esfera - esfera** [?] [-] [X]

Esfera 1 (vermella)		Esfera 2 (verda)	
Desplaçament X:	<input type="text" value="0 ."/>	Desplaçament X:	<input type="text" value="2 ."/>
Desplaçament Y:	<input type="text" value="0 ."/>	Desplaçament Y:	<input type="text" value="0 ."/>
Desplaçament Z:	<input type="text" value="0 ."/>	Desplaçament Z:	<input type="text" value="0 ."/>
Radi:	<input type="text" value="2 ."/>	Radi:	<input type="text" value="1 ."/>

Volum:   
 Intersecció:

Nota: Només en el cas d'interseccionar esferes només deixa calcular el volum, per tant no hi ha funció densitat. Totes les altres figures es calcula la massa (per tant, densitat). El motiu és que va ser la primera figura a realitzar-se i no s'ha afegit la massa (no seria problema fer-ho, sobretot després de veure el punt 2.4).

La majoria de pantalles són semblants: unes caselles de text per entrar radis, posicions i desplaçaments. Per defecte, ja ve una mostra de com podria ser.

**Esfera - con**

**Esfera (centrada)**

Desplaçament X:

Desplaçament Y:

Desplaçament Z:

Radi:

**Con (centrat)**

Desplaçament X:

Desplaçament Y:

Desplaçament Z:

Obertura (graus):

Densitat:

Resolució

Massa:

0%

En alguns casos, es disposa del botó superfície per entrar altres dades, si calen, per la superfície. En algunes figures cal calcular la superfície amb una posició diferent del volum (per problemes que he tingut en plantejar integrals de superfície).

**Esfera - con**

Esfera (centrada)

Desplaçament X:

Desplaçament Y:

Desplaçament Z:

Radi:

Con (centrat)

Desplaçament X:

Desplaçament Y:

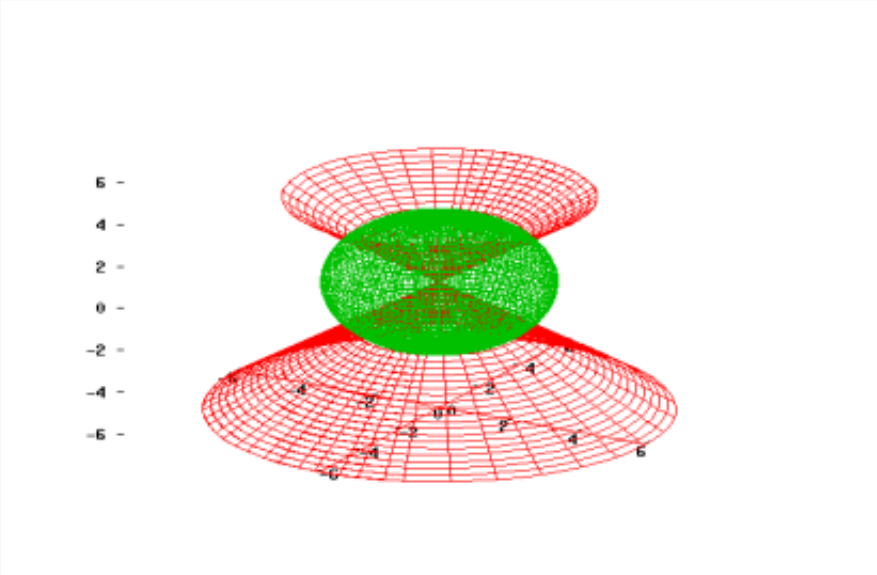
Desplaçament Z:

Obertura (graus):

Densitat:

Resolució

Massa:



0%

En alguns casos, es disposa del botó superfície per entrar altres dades, si calen, per la superfície. En algunes figures cal calcular la superfície amb una posició diferent del volum (per problemes que he tingut en plantejar integrals de superfície).

Observar com la funció densitat pot tenir variables. Fent clic a l'interrogant ens diu quines variables són permeses a la funció densitat.

# Con - paraboloides

Con  $Z^2 = A^2(X^2 + Y^2)$

Desplaçament X:

Desplaçament Y:

Desplaçament Z:

A:

Paraboloides  $Z = B(X^2 + Y^2)$

Desplaçament X:

Desplaçament Y:

Desplaçament Z:

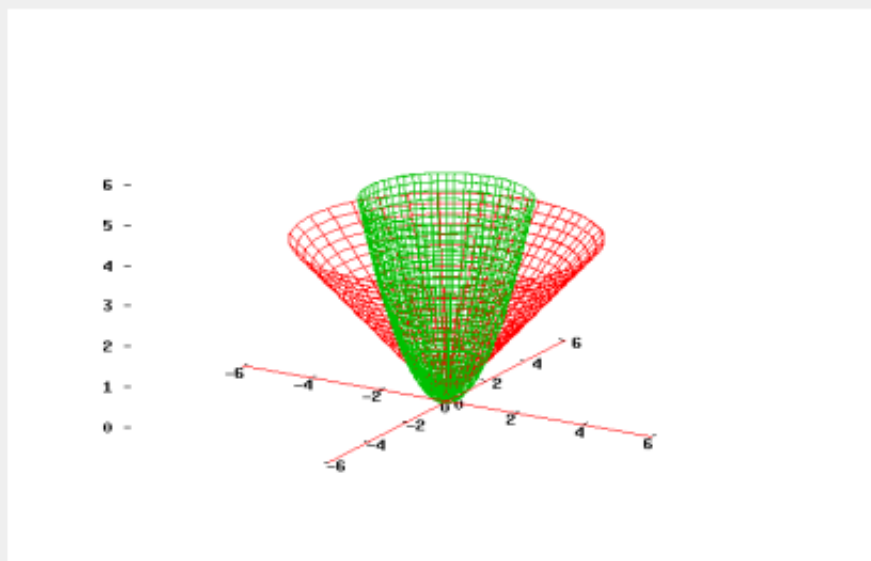
B:

Densitat:

Resolució:

Massa:

Intersecció:



Seguint el mateix esquema que el cas anterior. Veure que en aquest cas A i B són els paràmetres a entrar. Es podria calcular haver demanat radi i alçada i calcular A i B (en futures versions es podria deixar escollir: entrar A i B o bé entrar radi i alçada)

# Con - cilindre

Con  $X=A*\sqrt{(X^2+Y^2)}$

Desplaçament X:

Desplaçament Y:

Desplaçament Z:

A:

Cilindre

Desplaçament X:

Desplaçament Y:

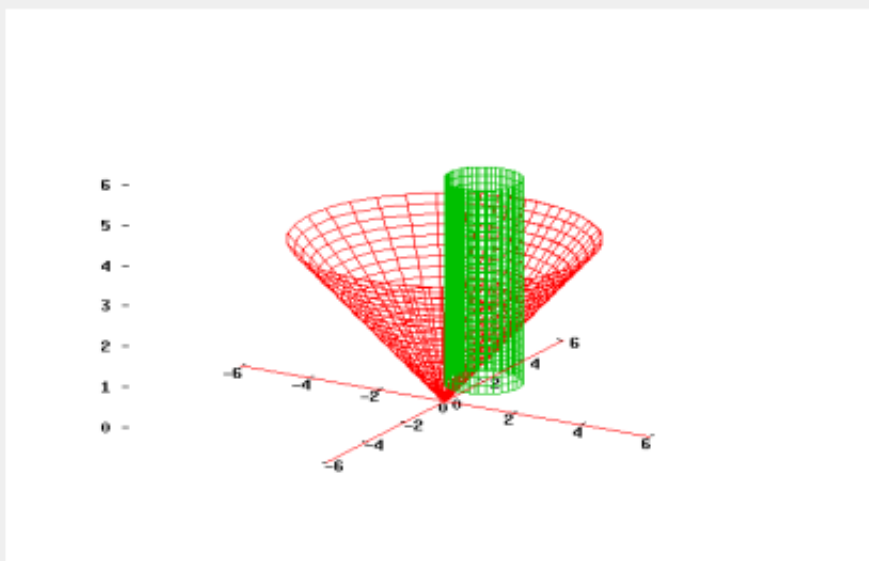
Desplaçament Z:

Radi:

Densitat:

Resolucio

Massa:



0%

Se segueix el mateix esquema anterior sense problemes. Es veu quins són els paràmetres a entrar (Y del centre del cilindre, A -factor d'obertura- pel con)

**Cilindre - cilindre**

Cilindre 1  
 Desplaçament X: 3 .  
 Desplaçament Y: 0 .0  
 Radi: 2 .  
 Alçada: 2 .

Cilindre 2  
 Desplaçament X: 0 .0  
 Desplaçament Y: 0 .0  
 Radi: 3 .  
 Alçada: 2 .

Densitat: ?

Resolucio ?

0.1

Pintar

Calcular

Limits integració  
 xinf: -5. xsup: 5 . yinf: -5. ysup: 5 . ?

Massa: 10.66

Superfície: 7.60

Sortir

Superfície 2

0%

Aquest cas, com a novetat, es demana el límit d'integració. Aquesta és la zona que el programa "rastregarà" buscant punts comuns a les dues figures per calcular la massa.

Si el límit no inclou tota la zona interseccionada, el resultat és incorrecte. Si el límit inclou massa zona interseccionada tardarà massa a resoldre.

Veure que no demana el límit en Z's, aquest és calculat entre 0 i l'alçada menor.







# Esfera - cilindre

Esfera

Desplaçament X:

Desplaçament Y:

Desplaçament Z:

Radi:

Cilindre

Desplaçament X:

Desplaçament Y:

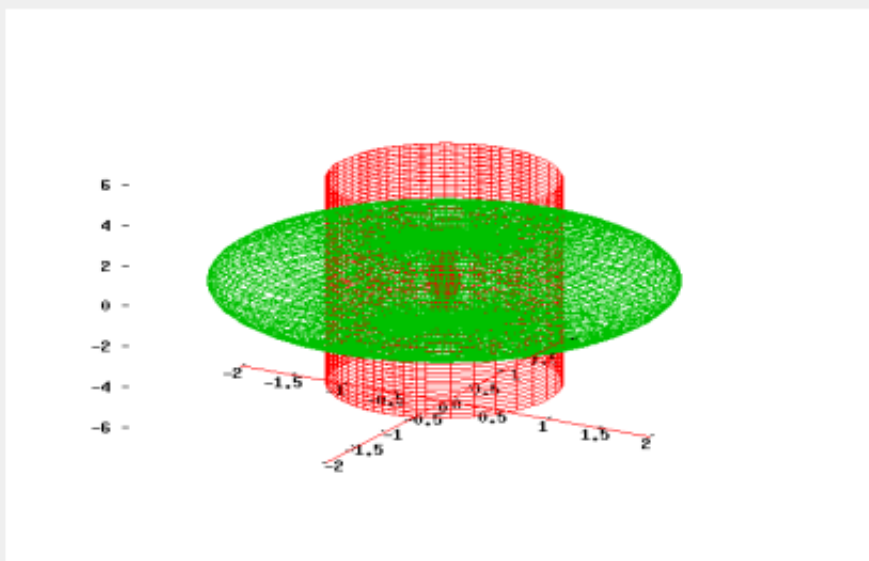
Desplaçament Z:

Radi:

Densitat:

Resolucio

Massa:



Mitjançant el botó superfície s'accedeix al quadre de diàleg per calcular la superfície intersecció (tenint en compte una posició diferent de les figures)

Con - con

Con 1  $Z=A*\sqrt{(X^2+Y^2)}$

Desplaçament X: 0.0

Desplaçament Y: 0.0

Desplaçament Z: 0.0

Alçada: 5 .

A: 1 .

Con 2  $Z=A*\sqrt{(X^2+Y^2)}$

Desplaçament X: 2.

Desplaçament Y: 0.0

Desplaçament Z: 0.0

Alçada: 1 .

A: 1 .

Densitat: ?

Resolucio ?

0.1

Pintar

Calcular

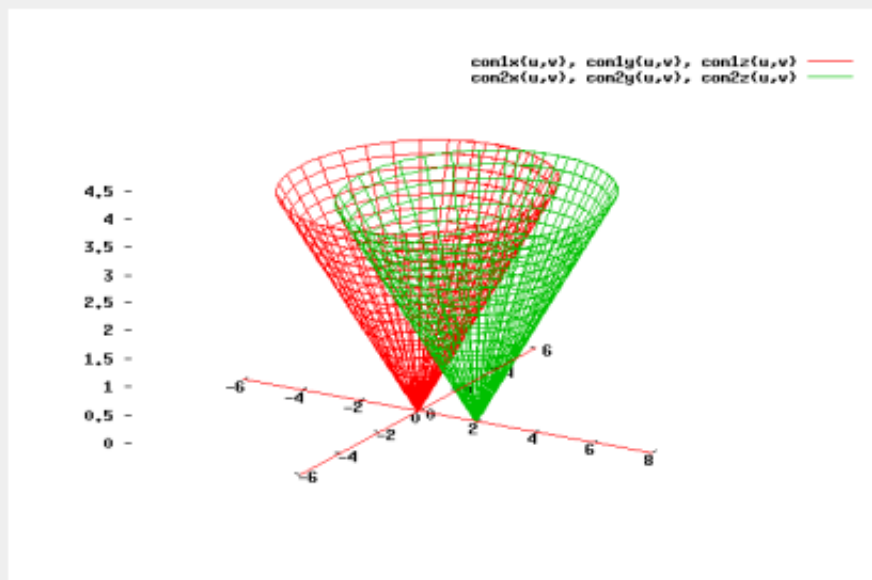
Massa:

85.660

Sortir

Limits integració

xinf: -5. xsup: 5 . yinf: -5. ysup: 5 . ?



0%

Igual que el cilindre-cilindre, es demanen els límits d'integració.

**Con - con** [?] [-] [X]

Con 1  $Z=A*\sqrt{(X^2+Y^2)}$       Con 2  $Z=A*\sqrt{(X^2+Y^2)}$

Desplaçament X:       Desplaçament X:

Desplaçament Y:       Desplaçament Y:

Desplaçament Z:       Desplaçament Z:

Alçada:       Alçada:

A:       A:

Densitat:

Resolucio:

Massa:

Limits integració

xinf:     xsup:     yinf:     ysup:     ?

Igual que el cilindre-cilindre, es demanen els límits d'integració.  
El funcionament és com els altres.

**Casquet** [?] [-] [X]

Casquet

Radi:

Paral·lel (graus):

Resolucio:

Area:

Volum:

S'ha també realitzat el casquet. Es demana el volum (ull, influeix amb al temps de càlcul i els graus del paral·lel a calcular.

**Esfera dins d'esfera**

Esfera 1 (vermella, centrada) Radi:

Esfera 2 (verda, centrada) Radi:

Resolució:

Densitat:

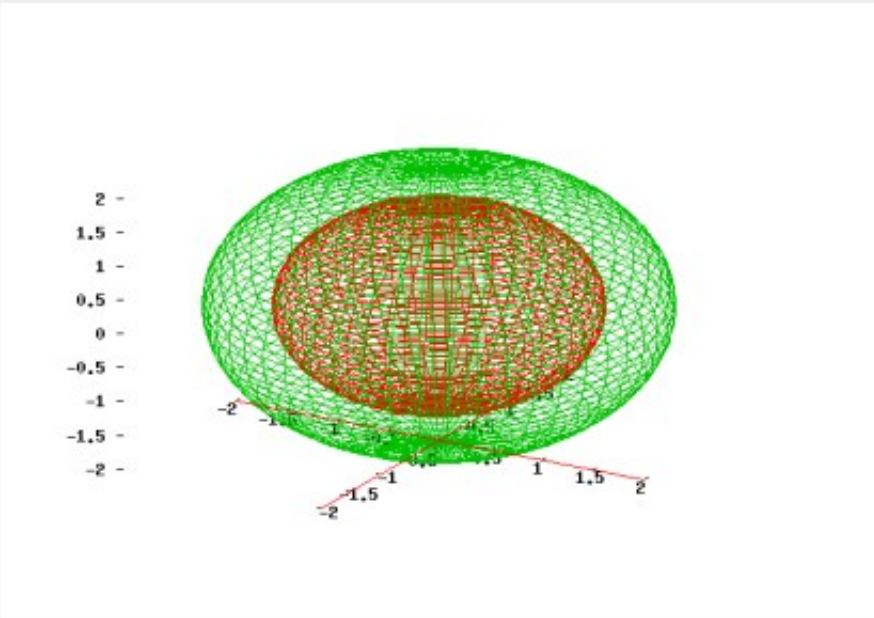
Límits integració (esfèriques)

< THETA <

< PHI <

pi és valor vàlid.  
Graus en radians.

Massa:



0%

Per últim es mostren dues esferes centrades. En aquest cas, l'usuari pot definir quins són els límits d'integració. El sistema calcula la massa entre les dues esferes.

## 6 Línies de futur

En aquesta secció es planteja el què es podria fer per futures versions. A la vegada, és una manera de veure el què no ha quedat completament satisfactori i que és conegut. En general, no són errades sinó idees o millores. Evidentment a la primera versió del programa no és possible abarcar totes les idees que es tenen (per qüestió de temps, centrar esforços, etc.). Moltes de les millores pensades no són a nivell matemàtic sinó a nivell d'interfície d'usuari.

Les dues primeres són les més importants:

- Figures que no interseccionen: avisar a l'usuari
- Comprovar millor casos límit: els casos proposats per defecte i variacions d'aquest funcionen correctament. Però casos límit (figures que es toquen tangencialment, figures que estan incloses totalment dins una altra) caldria comprovar-ho i segurament millorar-ho.

- Caldria fer un quadre de diàleg per entrar la densitat. Aquest quadre de diàleg validaria millor que la densitat fos correcta per la funció, ensenyaria a l'usuari quines funcions té disponibles (sin, cos, tan, etc. i permetre clicar a les funcions per seleccionar-les i fer-ne ús). També ensenyaria les variables disponibles en cada cas (x, y i z?  $\rho$ ,  $\theta$ ?). Fins i tot, l'usuari podria escollir amb quines coordenades desitja entrar la funció densitat i el programa podria fer un canvi de coordenades per les coordenades internes d'integració. Les coordenades internes d'integració són diferents a cada figura, en funció del què era més fàcil.
- Interfície d'usuari. L'interfície d'usuari té varis punts millorables. Per exemple s'hauria de fer que quan es modifiquen les característiques de les figures, es redibuixa la gràfica de forma automàtica (valorar si és prou ràpid)
- Donar passos intermitjos. Pot ser convenient donar més passos intermitjos. Actualment es fan internament al programa, es podrien ensenyar a l'usuari. Caldria, però, ajustar força passos: la manera que els estudiants fem alguns passos és diferent de com ho fan el programa, per plantejaments diferents. A més a més, és diferent per cada figura.
- Afegir més figures o interseccions amb figures conegudes. També permetre calcular la densitat de figures soles (sense intersecció entre figures)
- Millorar les gràfiques. Algunes pinten parts que no corresponen exactament al càlcul (especialment referent a l'eix Z). També, per limitacions del sistema utilitzat per les gràfiques, algunes poden quedar escalades incorrectament (és a dir, que les dues esferes poden quedar representades sense ser realment esfera perquè algun dels eixos té una escala diferent).
- Fer fils d'execució per avortar durant el càlcul. Això és purament informàtic però molt convenient. Actualment, un cop iniciat el càlcul no és possible aturar el procés fins que acaba (o tancar el programa). A més a més, el programa queda bloquejat sense permetre interacció amb l'usuari. Caldria executar en un "fil" o "thread" separat el càlcul, no bloquejar el programa i permetre avortar el càlcul. El càlcul, com s'ha vist anteriorment, pot ser massa llarg, sobretot si la resolució no és adequada en funció a la mida de l'intersecció. Això també permetria fer diversos càlculs a la vegada.
- Més millores de d'interfície d'usuari: anar mostrant la massa calculada fins al moment, en intervals de cada segon
- Optimització. Depenent de com és la funció densitat, optimitzar el càlcul. Per exemple, si la funció densitat és constant, es podria calcular el volum de l'intersecció (aplicant les optimitzacions vistes en punts anteriors) i un cop es tingués el volum, multiplicar per la densitat constant, o altres maneres.
- Estimació del temps restant. És trivial fer-ho, sabent el percentatge i el temps que ha transcorregut.
- Afegir mètodes intel·ligents comentats al punt "Intersecció con-con"
- Proves unitaris per cada figura.

## 7 Comentaris

### Interseccions utilitzant base matemàtica i aprofitant funcions informàtiques

Hi ha interseccions que és molt difícil trobar la intersecció de forma analítica (límits convencionals). En dos casos s'ha aplicat el comentat al punt 4.2.

Hi ha encara un punt intermig, que és aprofitar funcions matemàtiques per calcular els límits. Hi ha un cas que s'han fet proves integrant d'aquesta manera:

- Límit inferior:  $\max(f_a(x,y,z), f_b(x,y,z))$
- Límit superior:  $\min(f_a(x,y,z), f_b(x,y,z))$

D'aquesta manera, tot i ser difícil per les funcions  $f_a$  i  $f_b$  trobar de forma analítica els límits de la integral; al fer-ho numèricament és possible buscant la intersecció amb les funcions max i min. Va bé tenir aquests recursos a mà per si calen.

El cost que té el càlcul d'integrals numèriques definides és un problema i, en general, no és possible resoldre-ho. Està inherent a l'algorisme.

## 8 Comentaris personals

### 8.1 *Pedagogia*

El primer cop que un aplica el mètode d'integració tal com sempre l'havia imaginat (3 iteracions aniuades), sobre coordenades esfèriques, amb el Jacobià i dona el resultat esperat és molt gratificant. Sobretot, quan un pensa amb les capacitats que es tenen: qualsevol funció densitat pot ser avaluada, qualsevols límits poden ser integrats (sense tenir problemes amb integrals simbòliques).

Crec que als alumnes informàtics de matemàtiques els hi seria de profit tenir aquest algorisme o un semblant per provar-ho i realment veure-ho. O encara millor: que ho fessin ells (sense la rigidesa que comportaria fer-ho dins una part de l'assignatura de càlcul numèric, que caldria seguir algorismes ja pre-fabricats, més òptims però menys didàctics)

Jo sempre havia imaginat les integrals triples d'aquesta manera (3 iteracions). Però d'imaginar-ho a fer-ho (i que funcioni a la primera) hi ha un petit salt.

### 8.2 *PFC*

L'aplicació ha estat molt interessant, tot i que he evitat invertir massa temps en qüestions informàtiques (es pot veure tot el què queda per fer a línies de futur). Potser m'agradaria dedicar-hi més temps en un futur, i fins i tot estic pensant que podria ser el projecte final de carrera (de la superior, que ja tenia algunes idees però no hi ha encara feina feta).

Evidentment, polint molt més el què he comentat a línies de futur i afegint-hi segurament més funcionalitat matemàtica. Però el disseny actual del programa (caldrà canviar algunes coses) és prou robust com per créixer.

És només una idea per parlar-ho (més endavant).