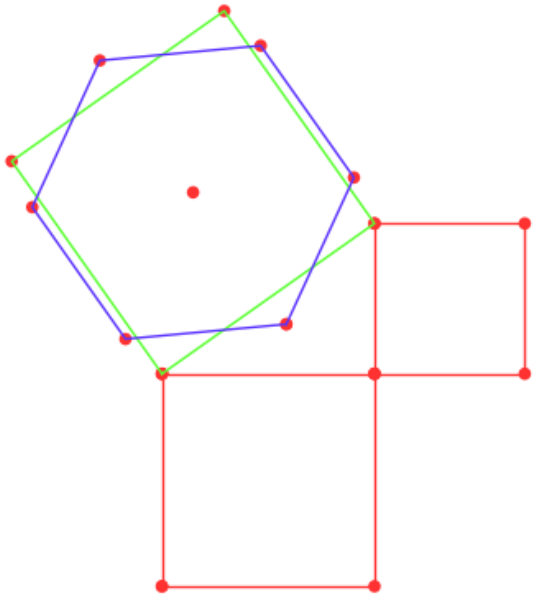
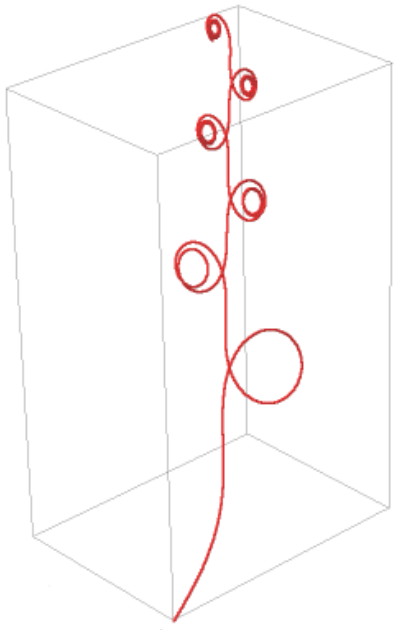
IMPLEMENTACIÓN E DESENVOLVEMENTO DE AULAS DE XEOMETRÍA EUCLÍDEA E DIFERENCIAL EN SAGE

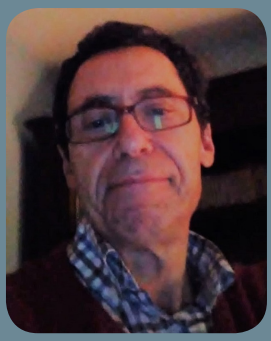
Francisco de Arriba Pérez
Alberto Castejón Lafuente
Eusebio Corbacho Rosas
Mª Carmen Somoza López
Ricardo Vidal Vázquez





Universidade Vigo
Servizo de Publicacións

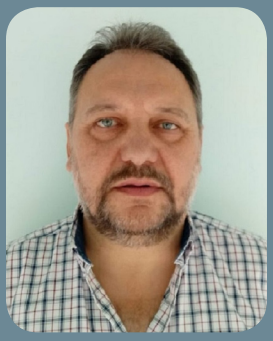
Francisco de
Arriba Pérez



Alberto
Castejón Lafuente

Eusebio
Corbacho Rosas

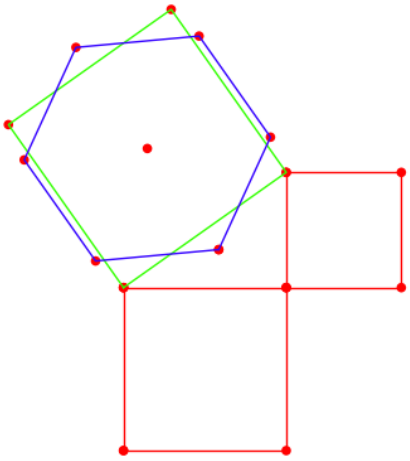
Mª. Carmen
Somoza López

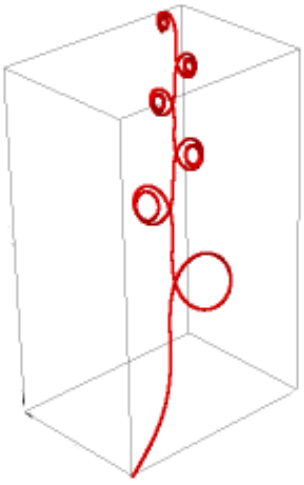


Ricardo
Vidal Vázquez

IMPLEMENTACIÓN E DESENVOLVEMENTO DE AULAS DE XEOMETRÍA EUCLÍDEA E DIFERENCIAL EN SAGE

Francisco de Arriba Pérez, farriba@uvigo.es
Alberto Castejón Lafuente, acaste@uvigo.es
Eusebio Corbacho Rosas, eusebio.corbacho@gmail.com
M. Carmen Somoza López, carmensomoza@uvigo.es
Ricardo Vidal Vázquez, rvidal@uvigo.es





Universidade de Vigo
Servizo de Publicacións
2020

MANUAIS DA UNIVERSIDADE DE VIGO, Nº. 73

Esta editorial é membro da UNE, o que garante a difusión e a comercialización das súas publicacións a nivel nacional e internacional.

Edición
Servizo de Publicacións da Universidade de Vigo
Edificio da Biblioteca Central
Campus de Vigo
36310 Vigo

ISBN: 978-84-8158-845-3
D.L.: VG 134-2020

Impresión: Tórculo Comunicación Gráfica, S.A.

PRÓLOGO

Ao iniciar a lectura deste libro, sobre todo para aquelas persoas que cursamos estudos de Enxeñería Industrial, un dáse conta da gran sorte que supón contar cunha obra onde se abordan dun xeito tan práctico, atractiva e pedagóxica conceptos relacionados coa xeometría euclídea e a xeometría diferencial. Abordar estes contidos nunha soa obra soamente está ao alcance daquelas persoas que, máis aló do amplo coñecemento que debe terse para levar a cabo a súa escritura nun libro como o que se presenta, demostran unha gran capacidade para centrarse en aspectos que son esenciais no coñecemento destas ramas. Nesta obra inclúense toda unha serie de nocións que poden considerarse importantes naquelas materias características dos estudos de enxeñería e que están acompañadas de exemplos e problemas que, ademais de facilitar a comprensión dos conceptos, resaltan os aspectos máis relevantes para o lector. Paréceme unha obra, tanto polos contidos como polo enfoque, moi apropiada para as materias de matemáticas que se cursan nos estudos do Mestrado en Enxeñería Industrial e tamén nalgúns dos grados da Rama Industrial.

Ante a lectura do libro tampouco puiden resistirme á tentación de recordar unha das miñas primeiras lecturas dun exemplar de matemáticas, máis aló dos típicos libros de texto que un manexa ao longo da súa formación. Nel falábase da Topoloxía como unha das ramas máis activas das matemáticas, sendo a Xeometría de Euclides un punto de partida esencial no seu desenvolvemento. No meu recordo está a banda de Moebius, nome que se corresponde co astrónomo alemán do século XIX. Como non recordar, tamén, un exemplo, que seguramente estará na mente de moitas persoas amantes destes temas, consistente en resolver o problema suscitado por tres casas situadas a pouca distancia unha doutra e ás que había que facerlles chegar os servizos de auga, gas e electricidade sen que se producise ningún cruzamento nas liñas que deberían abastecer estes servizos. Para o min supuxo unha enorme fascinación, aínda que tiña unha idade moi temperá, comprobar como a través das matemáticas podíanse resolver estes problemas, ás veces dun xeito sorprendentemente simple (soamente cando mo explicaban os meus profesores). Tamén destacar a importancia da xeometría diferencial que constitúe unha técnica que, mediante métodos diferenciais, dá resposta a numerosos problemas matemáticos moi presentes nos estudos de enxeñería. Por outra banda, destacar o gran acerto no desenvolvemento do libro ao resolver problemas a través dunha ferramenta que permite implicar ao alumnado na aprendizaxe. Contar cun texto que inclúa a resolución de problemas co manexo dunha ferramenta, como é o caso do software SAGE, representa un valor engadido que seguramente nun primeiro momento o lector non sexa capaz de percibir en toda a súa dimensión, pero teño por seguro espertaralle o interese polo desenvolvemento das súas propias ferramentas para a resolución de moitos problemas que lle poidan xurdir.

Por iso, para os estudantes de enxeñería, en particular os da Escola de Enxeñería Industrial da Universidade de Vigo, ao ter a sorte de recibir docencia por parte dos autores do libro, a lectura desta obra resulta altamente estimulante e un libro de texto que debe terse como referencia. Como alumno que fun da Escola de enxeñería Industrial, na que tiven o privilexio de ter como profesor ao Catedrático Eusebio Corbacho Rosas; agora como profesor, onde tamén tiven a oportunidade de compartir grandes momentos con algúns dos autores desta obra (Ricardo Vidal, Alberto Castejón, Carmen Somoza), destacaría de xeito moi especial a paixón que sempre han ter pola docencia e tamén **a preocupación pola boa formación como un elemento esencial na preparación de calquera enxeñeiro**. Ao mesmo tempo, sendo consciente do difícil que moitas veces resulta comprender algúns dos conceptos que se desenvolven na obra, os autores sempre demostraron ao longo das súas carreiras unha gran sensibilidade de como facer que esta dificultade vísese superada grazas a un esforzo constante por poñer **sempre ao alumnado no punto de mira do ensino**. Co paso do tempo un vaise dando conta da dimensión das persoas e comprende aqueles detalles que parecían non ter moita importancia. Recordo, ao iniciar os estudos do meu segundo curso de carreira, os 'corrillos' que había na Escola coa chegada do novo profesor que nos ía a dar matemáticas e que viña da Universidade de Zaragoza. Esa mesma expectación que eu puiden vivir naquel momento, puidérona experimentar unha gran cantidade de alumnos/as en todos estes anos. Escoitar ao alumnado que o enfoque dado polos autores nas súas clases representa unha bocalada de frescura e innovación, máis nestes tempos, é un dos maiores recoñecementos que pode recibir un docente.

Para rematar con este prólogo, non podo deixar de recordar cando fai uns cantos anos, co inicio do novo título do Mestrado en Enxeñaría Industrial, o profesor Eusebio Corbacho presentouse xunto co becario de último curso, Francisco de Arriba Pérez, nunha reunión ante Juan M. Pou Saracho, un referente para min e por aquel entón director da Escola de Enxeñería Industrial da Universidade de Vigo, para comentarnos que querían innovar na impartición das materias, introducindo nas súas clases prácticas o uso da linguaxe SAGE. Naquela reunión puiden comprobar en primeira persoa, aínda que poida soar un pouco remilgado, o amor dun entusiasta cara ao seu traballo e cara á súa paixón e vocación, **a formación en toda a súa amplitude**. Soamente as grandes persoas, aquelas que a súa única preocupación é tratar de que o seu alumnado aprenda dun xeito participativo, son capaces de acometer o desafiou de introducir esta ferramenta, por entón algo descoñecida para min (como atenuante ao meu favor, tamén para outra moita xente), para mellorar a aprendizaxe dos que deben ter un coñecemento destes contidos e poder resolver aqueles problemas cos que se terán que enfrontar, primeiro seguramente con aquelas outras materias que os utilizan e, despois, xa dun xeito máis esixente, ante os problemas reais aos que seguramente teñan que enfrontarse na súa vida profesional.

Creo que o momento presente que estamos vivindo pon de manifesto que as matemáticas nunca estiveron tan de actualidade (non de moda) como nos últimos tempos e só espero que as titulacións de enxeñería non perdan de vista o valor que representan na formación dos futuros enxeñeiros. **Só unha boa base pódenos facer sentir optimistas ante os importantes desafíos aos que deberemos enfrontarnos nos próximos tempos.**

Remato co meu máis sincero agradecemento por brindarnos a oportunidade da lectura deste libro e a miña admiración polo empeño, tesón e contido multidisciplinar que o fan de gran valor, sobre todo pola xenerosidade de non escatimar esforzos e así poder aproveitar a transmisión do coñecemento dos autores a través da súa obra.

O meu recoñecemento para as persoas, como os autores deste libro, que nos permiten compartir os seus coñecementos e experiencia, para que as matemáticas non constitúan soamente unha ciencia, senón tamén unha poderosa ferramenta que nos axude a seguir avanzando como sociedade.

Juan E. Pardo Froján.
Doutor Enxeñeiro Industrial.
Profesor Titular do Departamento de Organización de Empresas e Marketing

Táboa de contidos

1. Introdución

Este libro é continuación do manual "Implementación e desenvolvemento de aulas matemáticas avanzadas en Sage", IDAMAS, publicado pola Universidade de Vigo en 2018 e dedícase ao tratamento en Sage da Xeometría Euclídea e Xeometría Diferencial, que son parte esencial na formación do enxeñeiro e, en particular, do enxeñeiro industrial. IDAMAS debe ser considerado unha lectura necesaria para extraer toda a utilidade a este segundo libro. En *2.Instalación de Sage* dábanse as instrucións precisas para instalar Sage, nos sistemas operativos máis usuais, no ordenador do lector e en *3.Prácticas con Sage* describíanse as funcións máis importantes para o tratamento de conxuntos, listas, dicionarios, álxebra lineal, cálculo diferencial en variedades e problemas inversos. Os DATA cos códigos destas funcións e os correspondentes ao libro actual poden obterse en https://corbacho.webs.uvigo.es, poñendo a disposición do lector a importante labor de investigación realizada.

Este libro, como IDAMAS, está dedicado aos alumnos de Matemáticas da Especialidade e Complementos de Formación dos Graos de Tecnoloxías Industriais e Electrónica Industrial e Automática, e aos alumnos de Métodos Matemáticos do Mestrado de Enxeñería Industrial. Aínda que algúns contidos exceden os obxectivos destes cursos, iniciar ao alumno na programación e darlle a posibilidade de manexar potentes resultados de investigación actual seralle de gran utilidade en traballos futuros.

Tamén é interesante para profesores e personal investigador que queiran incorporarse á utilización de Sage. A principal vantaxe deste software é que é gratuíto e accesible a calquera docente ou estudante que desexe utilizalo. É habitual na docencia que cada profesor teña que dar clases en diferentes grupos de laboratorio e Sage nas súas versións 6.x permite crear no PC/pórtatil do profesor un "usuario-ubuntu" por cada laboratorio que imparte, que será un espazo novo de traballo, illado do resto, con utilidades adecuadas a cada unidade temática almacenada nos arquivos DATA e cos que os alumnos poden confeccionar as súas propias caixas de ferramentas matemáticas. Os exercicios que se realicen ao longo do curso vanse almacenando no servidor do profesor, tendo así constancia dos mesmos e facilitando a avaliación continua dos alumnos. Do mesmo xeito pódense crear terminais específicos para cada seminario de investigación que o lector estea dirixindo.

Este manual comeza coa Worksheet 2, para seguir a orde establecida en IDAMAS, que contén a Worksheet 0 e a Worksheet 1. A Worksheet 2 está

dedicada ao estudo da Xeometría Euclídea e Teoría de Grafos, tratando en profundidade os seguintes problemas (algúns dos cales están aínda sen resolver), para un conxunto finito de puntos V do plano complexo:

- Achar a envoltura convexa de V.
- Achar a teselación triangular da envoltura convexa de V que maximice o ángulo mínimo de todos os triángulos (teselación de Delone).
- Achar unha árbore xeneradora de lonxitude mínima ou $MST(V)$
- Salvo en casos excepcionais, existe unha familia $\mathcal{S}$ de subconxuntos R do plano complexo cumprindo que $V \cap R = \emptyset$ e que a lonxitude de $MST(V \cup R)$ é menor que a de $MST(V)$. Interesa achar un $S \in \mathcal{S}$ tal que

$$MST(V \cup S) = \min_{R \in \mathcal{S}} \{MST(V \cup R)\}$$

  Tal S é o conxunto de Steiner de V e $MST(V \cup S)$ é o Steiner Minimal Tree de V.
- Achar un ciclo hamiltoniano de lonxitude mínima ou $MHC(V)$.
- Achar un camiño hamiltoniano de lonxitude mínima ou $MHP(V)$.
- Achar en $V \cup \{v_i, v_f\}$ un camiño hamiltoniano de lonxitude mínima entre os que cumpren $g(v_i) = g(v_f) = 1$.

A Worksheet 3 está dedicada ao tratamento, sobre todo, numérico de ecuacións diferenciais ordinarias con exemplos de problemas de Cauchy en Ciencia e Tecnoloxía e Problemas de Cauchy en Xeometría.

Para concluír esta serie de manuais de matemáticas avanzadas en Sage, estamos preparando un terceiro volume dedicado a Variable Complexa e un cuarto a Transformadas Integrais.

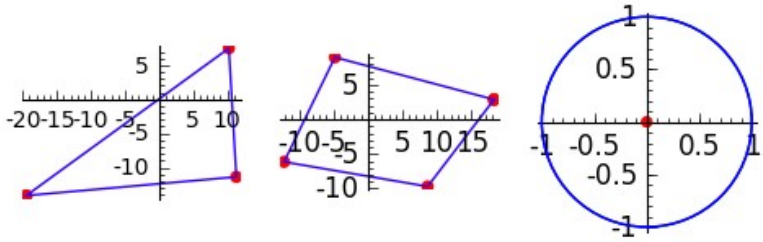
2. Worksheet 2

```
load (DATA + 'Worksheet0.sage')
load (DATA + 'Worksheet1.sage')
load (DATA + 'Worksheet2.sage')
```

2.1. Euclides

No libro I dos Elementos, século III a. C., Euclides introduciu definicións, principios e conceptos necesarios para tratar con rigor as figuras rectilíneas determinadas por tres, catro ou máis puntos e a figura circular, limitada pola súa circunferencia (Peripheria) e determinada polo seu centro (kentron) e o seu radio (diastema). Para construír figuras rectilíneas de menos de nove puntos definimos en Sage as funcións **optimiza()** e **optimizac()** que determinan a poligonal aberta e pechada de menor lonxitude.

```
T=listacomplejos(20,3)
DT=poligonal(T+[T[0]],.7,1)
C=listacomplejos(20,4)
Q=optimizac(C+[C[0]])[0]
DQ=poligonal(Q,.7,1)
DC=circle((0,0),1)+point([0,0],color='red',pointsize=25)
show(graphics_array([DT,DQ,DC]), figsize=[5,3])
```



As 48 proposicións establecidas nel son paradigma do método axiomático-dedutivo que impera desde entón en matemáticas. As 26 primeiras versan

sobre triángulos. Ao non dispoñer dun sistema de numeración eficiente, os razoamentos susténtanse na proporcionalidade de segmentos, a medida de ángulos e algunhas propiedades da circunferencia. Agora podémolas tratar coas funcións **triangle**(T), **circuncentro**(T), **incentro**(T), **ortocentro**(T) e **baricentro**(T) introducidas en Worksheet1.sage (véxase [1])

Exemplo 1:

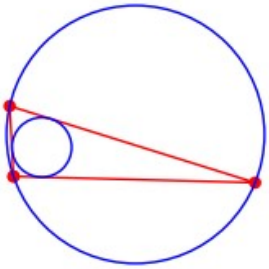
a) Comprobar que os ángulos dun triángulo suman π radiáns.

b) Debuxar as circunferencias inscrita e circunscrita a un triángulo dado.

```
T=listacomplejos(20,3)
show(pt(T),figsize=[1.5,1.5])
```

```
[P,A]=triangle(T)
print 'a)', sum(A)
print 'b)'
C=circuncentro(T);P=incentro(T)
show(pt(T)+circle(vectorizac(C[0]),C[1])+
circle(vectorizac(P[0]),P[1]),figsize=[2,2])
```

```
a) 3.14159265358979
b)
```



As 6 proposicións seguintes desenvolven a teoría das paralelas e as restantes tratan a cuadratura de diversas figuras.

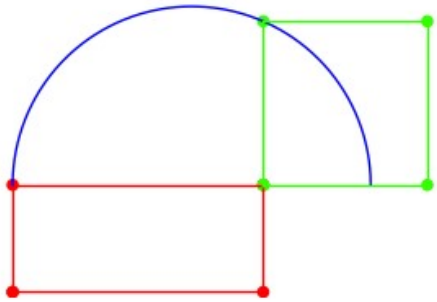
Exemplo 2:

Achar a cuadratura dun rectángulo:

```
a=7;b=3
R=[0,a,a+b*I,b*I,0]
show(poligonal(R),figsize=[2,2])
```

```
show(cuadrarec(3,7),figsize=[3,3])
```

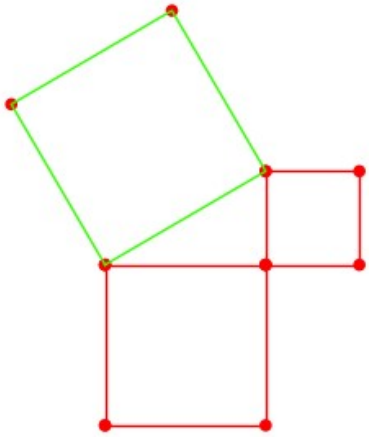
4.58257569495584



A cuadratura de calquera triángulo obtense cadrando o rectángulo de igual base e metade altura e a de calquera polígono conséguese descompoñéndoo en triángulos e sumando cadrados como Pitágoras nos ensinou:

```
show(Pitagoras(12,7),figsize=[3,3])
```

13.8924439894498



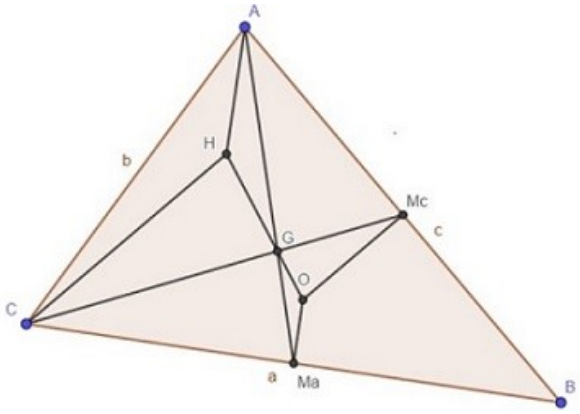
No século XVIII Euler obtivo dous novos teoremas sobre triángulos con demostracións de corte clásico:

Teorema E1:

En todo triángulo o ortocentro H, o baricentro G e o circuncentro O están aliñados, de modo que

$$GH = 2OG$$

Demostración



No triángulo ABC construímos o circuncentro O e o baricentro G e determinamos o punto H tal que $GH = 2OG$

É claro que os triángulos GHA e $GOMa$ son semellantes de razón 2 e en consecuencia OMa é paralelo a AH. Isto demostra que AH é o segmento contido na altura que desde o vértice A se traza sobre o lado a.

De igual modo, vemos que o triángulo CHG é semellante a $GOMc$ e en consecuencia CH será un segmento contido na altura que desde o vértice C se pode trazar sobre o lado c.

Así, o punto H é o ortocentro do triángulo ABC.

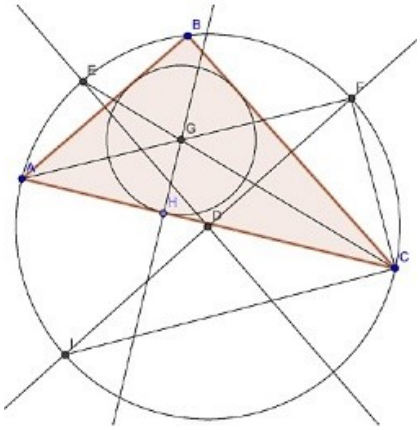
Teorema E2:

En todo triángulo, se R é o radio do circuncírculo, r o radio do incírculo e d é a distancia entre o circuncentro e o incentro, cúmprese que

$$2Rr = R^2 - d^2$$

Demostración:

Dado un triángulo $[A, B, C]$ e as súas circunferencias circunscrita e inscrita:



Por ser semellantes os triángulos rectángulos $[A,H,G]$ e $[I,C,F]$ tense que

$$2\cdot R.r = AG\cdot FC$$

Como a mediatriz dun lado e a bisectriz do ángulo oposto se intersecan sobre a circunferencia circunscrita, o triángulo $[G,F,C]$ é isóscele,

$$FC = GF \quad \text{e} \quad 2\cdot R\cdot r = AG\cdot GF$$

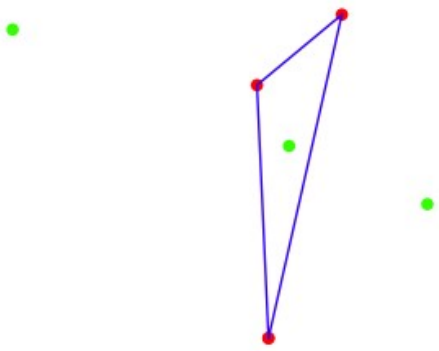
Como $2\cdot R\cdot r$ é a potencia de G respecto da circunferencia circunscrita,

$$2\cdot R\cdot r = (R+d)\cdot(R-d) = R^2 - d^2$$

Euler tamén nos ensinou a identificar cada punto do plano cun número complexo. Na Worksheet 0 (vexáse [1]) tratamos o conxunto dos números complexos tanto na súa forma binómica como módulo argumental e demos o xeito de obter listas aleatorias de n complexos situados nun rectángulo $T\times T$ mediante a función **listacomplejos(T,n)**. Isto aumenta extraordinariamente as posibilidades de experimentación en Xeometría plana. Como exemplo, comprobaremos experimentalmente os dous teoremas de Euler anteriores.

Comprobación experimental do teorema E1:

```
T=listacomplejos(10,3)
O=ortocentro(T)
B=baricentro(T)
D=circuncentro(T)
show(pt(T,.7,0)+plse([B,D[0],O],.3),figsize=[3,3])
```

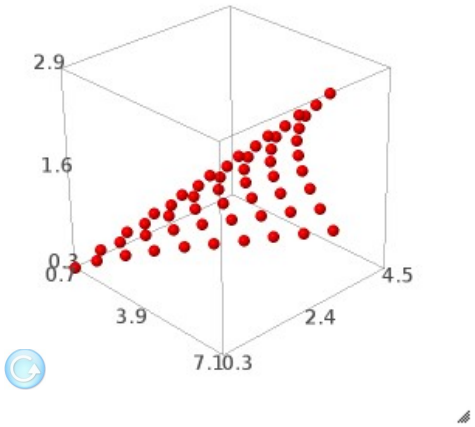


Comprobación experimental do teorema E2:

Grazas a Sage podemos deducir a relación $2 \cdot R \cdot r = R^2 - d^2$ axustando nubes de puntos:

Elixindo calquera familia de triángulos e representando en $\mathbb{R}^3$ os puntos (R,d,r) apreciamos a simple vista unha relación funcional entre eles

```
H=[1..10]
V=oplista(H,m,I)
LT=[[z,0,w] for [z,w] in cartesian_product([H,V])]
LT=depuralista(LT)
P=[[circuncentro(a)[1],abs(circuncentro(a)[0]-incentro(a)
[0]),incentro(a)[1]] for a in LT]
show(point3d(P,color='red',size=15),figsize=[3,3,3])
```



Sage pode axudarnos a determinala se utilizamos os métodos de axuste da Worksheet 1:

Fixamos sucesivamente, $R = 1, 2, 3, 4, 5$ e sobre a circunferencia de centro 0 e radio R consideramos a familia $\{T_i \mid i = 0, \cdots, N\}$ de todos os triángulos que se poden formar con vértices nos puntos $\{Re^{\frac{ik2\pi}{9}} \mid k = 0, 1, \ldots, 8\}$. Se calculamos a distancia d_i do incentro á orixe e o radio r_i da súa circunferencia inscrita, podemos determinar o mellor axuste entre $X = [d_0, d_1, \ldots, d_N]$ e $Y = [r_0, r_1, \ldots, r_N]$.

```
for R in [1..5]:
    L=[R*exp(I*k*2*pi/9).n() for k in range(9)]
    T=Combinations(L,3).list()
    P=[[abs(incentro(a)[0]),incentro(a)[1]] for a in T]
    DP=depuralista(P)
    DPr=[[round(a[0],6),round(a[1],6)] for a in DP]
    X=[a[0] for a in DPr]
    Y=[a[1] for a in DPr]
    F(d)=[1,d,d^2]
    N=nube(X,Y,F)
    print 'Para R=',R
    print 'r=',N[2](x=d)
```

```
Para R= 1
r= -0.500001103900085*d^2 + (2.07947903785310e-7)*d +
0.500000240440623
Para R= 2
r= -0.249999925505429*d^2 + (3.29419426181587e-8)*d +
0.999999850783890
Para R= 3
r= -0.166666768760854*d^2 + (1.08613134819358e-7)*d +
1.50000008452623
Para R= 4
r= -0.125000036341412*d^2 + (3.36021881319759e-7)*d +
1.99999939135009
Para R= 5
r= -0.100000018530218*d^2 - (2.51878881085688e-8)*d +
2.50000008227136
```

Destes resultados deducimos que $r = -\frac{1}{2R}d^2 + \frac{R}{2}$.

No século XIX Poncelet pregúntase se dado un círculo C e outro círculo D que conteña estritamente a C, existe un triángulo que os teña, respectivamente, por incírculo e circuncírculo. A tal triángulo chámaselle de Poncelet.

Teorema P:

Dado un círculo C e un círculo D que o conteña estritamente, ou non existe triángulo de Poncelet ou existen infinitos.

Comprobación experimental do teorema P:

É evidente que podemos limitarnos a estudar o caso en que C é o círculo

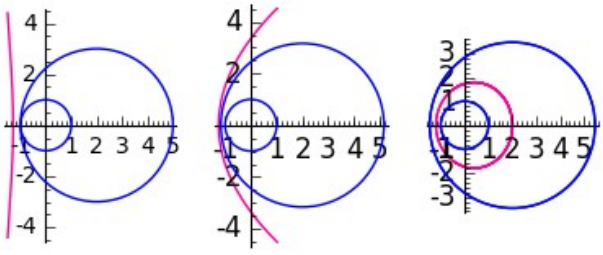
unidade e D ten o seu centro nun punto $(a,0)$ con $a \geq 0$ e o seu radio R cumpre que $R \geq 1+a$.

Para tratar este problema coa axuda de Sage cada tanxente a ∂C determina unha corda de ∂D e tomando os seus extremos como vértices dun triángulo circunscrito a C determinamos o lugar xeométrico do terceiro vértice mediante a función **lugar(a, R)**.

A expresión paramétrica de **lugar(a, R)** é intratable, pero Sage pode representalo eficazmente usando a orde **parametric_plot()** e mostrarnos que **lugar(a, R)** é unha cónica.

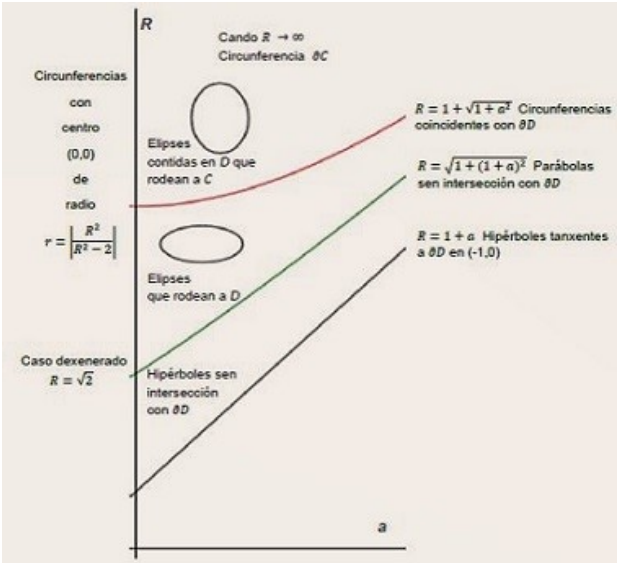
Para (a,R) con valores $(2,3)$, $(2,\sqrt{10})$ e $(2,\frac{3}{10}+\sqrt{10})$ obtemos as seguientes cónicas de Poncelet pintadas en vermello:

```
a=2
R1=3
L1=lugar(a,R1)
C=circle((0,0),1)
D1=circle((a,0),R1)
DL1=parametric_plot(L1,(t,-2.1,2.1), hue=.9, figsize=[3,3])
R2=sqrt(10)
L2=lugar(a,R2)
D2=circle((a,0),R2)
DL2=parametric_plot(L2,(t,-2.5,2.5), hue=.9, figsize=[3,3])
R3=.3+sqrt(10)
L3=lugar(a,R3)
D3=circle((a,0),R3)
DL3=parametric_plot(L3,(t,-4.5,4.5), hue=.9, figsize=[3,3])
show(graphics_array([C+D1+DL1,C+D2+DL2,C+D3+DL3]),figsize=
[4,6])
```



Para que exista triángulo de Poncelet é necesario e suficiente que a cónica corte a ∂D. Isto non sucede nunca no caso da hipérbole ou a parábola. Para

que suceda no caso elíptico, a cónica de Poncelet ten que coincidir con ∂D. Así pois, ou non existe triángulo de Poncelet ou se existe un, hai infinitos. O lector pode comprobar como exercicio que se verifica a seguinte casuística:

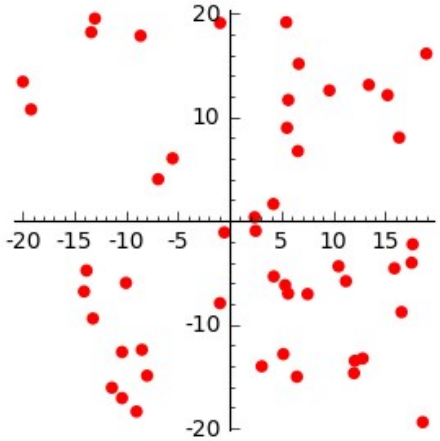


A posibilidade de xerar e manexar en Sage listas de complexos permítenos crear e controlar conxuntos V de puntos no plano que poden servir para modelizar problemas da vida real, de gran interese técnico e económico.

Cando eses conxuntos teñen un cardinal baixo, $|V| < 8$, eses problemas poden ser tratados, como xa vimos, por métodos da xeometría euclídea clásica e da combinatoria e probablemente o lector xa saiba abordalos. Pero cando $|V| > 8$, o uso do ordenador cun software eficaz para tratar listas faise imprescindible.

Con ordes sinxelas obtemos representacións de conxuntos V

```
V=listacomplejos(20,50)
show(plce(V),figsize=[3,3])
```



cuxos puntos se poden interpretar como plantas ornamentais que precisan unha instalación de rego por goteo, ou as tendas que debe visitar mensualmente un viaxante de comercio, ou os colectores que deben ser recollidos diariamente polo servizo de limpeza rural dun concello.

En calquera destas interpretacións é de interese atopar un adecuado conxunto de arestas ou segmentos con extremos en V, con mínima suma de lonxitudes.

2.2.Teoría de grafos no plano complexo

Un grafo é un par (V, E) onde V é un conxunto de vértices no plano complexo e E é unha lista de arestas con extremos en V. Chamamos subgrafo de (V, E) a calquera grafo (V', E') que cumpra $V' \subset V$ e $E' \subset E$.

Un camiño en (V, E) é un subgrafo (V', E') onde

$$V' = \{v_1, v_2, \ldots, v_k\} \quad \text{e} \quad E' = [(v_1, v_2), (v_2, v_3), \ldots, (v_{k-1}, v_k)]$$

Diremos que v_1 e v_k son, respectivamente a orixe e o final do camiño e que o camiño os une ou conecta. O camiño dise pechado se $v_1 = v_k$.

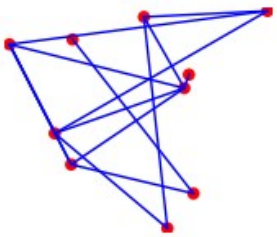
O grafo (V, E) dise conexo se calquera par de vértices se pode unir mediante un camiño.

Nun grafo (V, E), unha aresta (v, v) chámase lazo. Se unha aresta aparece repetida en E dise que o grafo ten arestas paralelas. Un grafo sen lazos nin arestas paralelas dise simple. Un grafo simple dise completo cando $|E| = \frac{|V|(|V|-1)}{2}$

Exemplo 1:

Debuxar un grafo simple de n vértices e m arestas.

```
n=10
m=15
V=listacomplejos(20,n)

if m>n*(n-1)/2:
    print 'imposible'

elif m==n*(n-1)/2:
    print 'grafo completo'
    A=[]
    for v in V:
        for w in restalistas(V,[v]):
            A.append([v,w])
    E=depura(A)
    show(plse(V)+sum([line([ri(e[0]),ri(e[1])]) for e in
E]),figsize=[3,3])

else:
    A=[]
    k=0
    while k<m:
        e=[choice(V),choice(V)]
        if e[0]!= e[1]:
            A.append(e)
            E=depura(A)
            k=len(E)

    show(plse(V)+sum([line([ri(e[0]),ri(e[1])]) for e in
E]),figsize=[2,2])
```



Cada grafo (V, E) determina a función grao $g_E : V \to \mathbb{N}$ onde $g_E(v)$ é o número de arestas que inciden en v. Diremos que o vértice v está desconectado se $g_E(v) = 0$. Implementamos en Sage a función **grado(**v, E**)** $= g_E(v)$. O grafo (V, E) dise k-regular se o seu grao é a función constante k.

É útil observar que

$$2|E| = \sum_{v \in V} g_E(v)$$

O grafo (V, E) contén ciclos se ten algún subgrafo 2-regular.

Chamamos árbore xeradora dun grafo conexo (V, E) a un subgrafo (V, E') conexo que non contén ciclos. Cúmprese que $|E'| = |V| - 1$.

Exemplo 2:

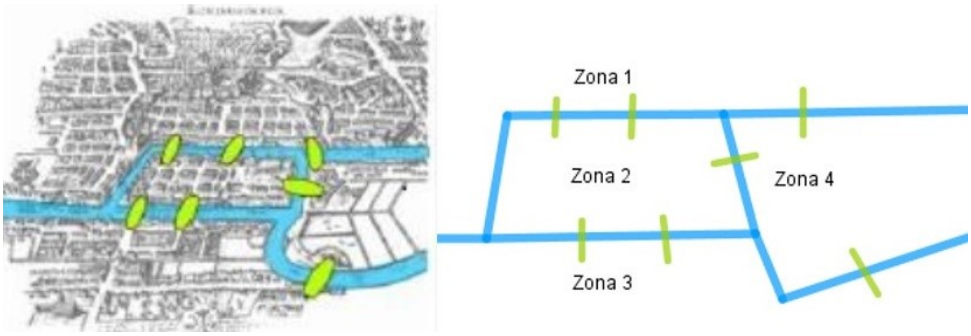
Achar o grao do grafo do Exemplo 1 e probar que cumpre a fórmula $2|E| = \sum_{v \in V} g_E(v)$.

```
show(grado(V,E))
print
2*len(E)==sum(grado(V,E))
```

$[3, 5, 3, 1, 5, 4, 1, 4, 2, 2]$

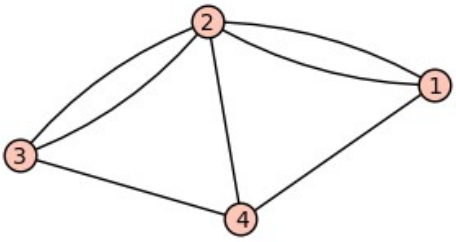
True

A función grao permitiu a Euler, en 1736, afirmar que era imposible deseñar un camiño pechado pola cidade de Königsberg (ver plano) pasando unha e só unha vez por cada unha das súas sete pontes sobre o río Pregel:



O paseo pódese modelar segundo o grafo

```
w=[[1,2],[1,2],[2,4],[1,4],[2,3],[2,3],[3,4]]
G=Graph(w,multiedges=True)
G.graphplot().show(figsize=[3,3])
```



Para que tal paseo fose posible, cada vértice debería ser de grao par e, como vemos, ningún o é.

Euler publicou este resultado baixo o título *Solutio problematis ad geometriam situs pertinentis* e está considerado por moitos autores como o nacemento da teoría de grafos. Tamén, por concentrar unha rexión nun punto e deformar rectas en curvas é considerado como o nacemento da Topoloxía.

Un grafo (V, HC) conexo e 2-regular é un ciclo hamiltoniano (Hamiltonian Cycle).

Un grafo (V, HP) conexo, con todos os seus puntos de grao 2, salvo dous de grao 1, é un camiño hamiltoniano (Hamiltonian Path).

Un grafo (V, P) conexo, no que as arestas só se cortan nos puntos de V, dise plano e divide o plano complexo nun conxunto de caras C, unha das cales é non acoutada. En 1750 Euler estableceu a seguinte conxectura:

$$|C| + |V| = |P| + 2$$

Hoxe podemos presentar unha sinxela proba da mesma: Unha árbore xeradora (V, T) é un caso particular de grafo plano con $|C| = 1$ e, xa que logo, a conxectura de Euler cúmprese trivialmente nel. En calquera outro grafo plano, suprimir unha aresta (sempre que o grafo resultante sega sendo conexo) supón suprimir unha cara e, así, a conxectura é certa sempre, xa que, a supresión sucesiva de arestas condúcenos inexorablemente a unha árbore xeradora.

Dado o grafo completo de vértices V, a procura dun conxunto adecuado de arestas E con suma de lonxitudes mínima pode modelizar diferentes tipos de problemas de gran interese científico e técnico. Por exemplo:

1. Achar a envoltura convexa de V.
2. Achar a teselación triangular da envoltura convexa de V que maximice o ángulo mínimo de todos os triángulos (teselación de Delone).
3. Achar unha árbore xeradora de lonxitude mínima ou $MST(V)$.
4. Salvo en casos excepcionais, existe unha familia $\mathcal{S}$ de subconxuntos R do plano complexo cumprindo que $V \cap R = \emptyset$ e que a lonxitude de $MST(V \cup R)$ é menor que a de $MST(V)$. Interesa achar un $S \in \mathcal{S}$ tal que

   $$MST(V \cup S) = \min_{R \in \mathcal{S}} \{MST(V \cup R)\}$$

   Tal S é o conxunto de Steiner de V e $MST(V \cup S)$ é o Steiner Minimal Tree de V.
5. Achar un ciclo hamiltoniano de lonxitude mínima ou $MHC(V)$.
6. Achar un camiño hamiltoniano de lonxitude mínima ou $MHP(V)$.
7. Achar en $V \cup \{v_i, v_f\}$ un camiño hamiltoniano de lonxitude mínima entre os que cumpren $g(v_i) = g(v_f) = 1$.

2.2.1. Envoltura convexa

En Worksheet0.sage (véxase [1], páx. 59) introducimos a función **envolturaconvexa(V)** que é unha versión particular do coñecido algoritmo de Jarvis, e dános os puntos expostos da envoltura convexa de V ordenados en

sentido antihorario a partir do de menor abscisa e menor ordenada. Como V queda modificado tras avaliar esta función, convén facer previamente unha copia W=**copy(**V**)**.

A partir da saída C desta función é inmediato calcular o perímetro da envoltura convexa, a súa área por triangulación e definir a función **interior(**z, C**)** cuxo valor será True se e só se z está no interior da envoltura convexa. Ademais a orde de Sage **polygon2d(**C**,** fill = True, color='red'**)** representa a rexión en vermello.

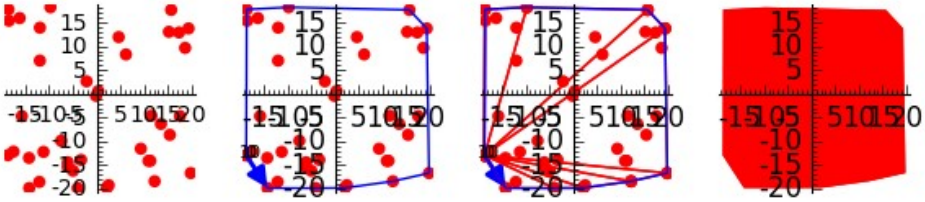
Exemplo 3:

Achar o perímetro e a área da envoltura convexa dun conxunto aleatorio de n complexos.

```
n=30
V=listacomplejos(20,n)
```

```
W=copy(V)
C=envolturaconvexa(W)
VC=vercamino(C)
VP=plce(V)
PER=lon(C)[0]
T=[]
for k in [1..len(C)-2]:
    T.append([C[0],C[k],C[k+1]])
A=sum([pt(t) for t in T])+VC
AREA=sum([heron(t) for t in T])
PC=polygon2d(C,fill=True, color='red')
show(graphics_array([VP,VC+VP,A+VP,PC]),figsize=[6,4])
print 'perímetro=', PER, 'área=', AREA
```

```
perímetro= 143.246976219835 área= 1394.41312771894
```



```
interior(0,C)
```

```
True
```

Se $co(V)$ é a envoltura convexa de V e $\delta(co(V))$ é a súa fronteira, definimos

$$V_1 = V \cap \delta(co(V)),$$

$$V_2 = V \cap \delta(co(V \setminus V_1)),$$

$$V_3 = V \cap \delta(co(V \setminus (V_1 \cup V_2)),$$

$$\cdots$$

$$V_n = V \cap \delta(co(V \setminus (V_1 \cup \cdots \cup V_{n-1})).$$

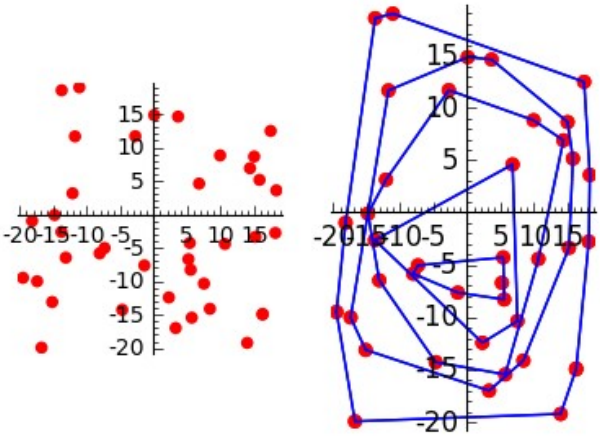
A función **cebolla(V)** devólvenos a lista $[V_1, \cdots, V_n]$, que nos proporciona a partición do conxunto V nos seus niveis de convexidade $\{V_1, \cdots, V_n\}$.

Exemplo 4:

Nun conxunto aleatorio V de corenta puntos dicir cantos niveis de convexidade ten e representalos.

```
V=listacomplejos(20,40)
C=cebolla(V)
print 'ten', len(C[0]), 'niveis de convexidade'
show(graphics_array([plse(V),C[1]]),figsize=[4,3])
```

```
ten 6 niveis de convexidade
```



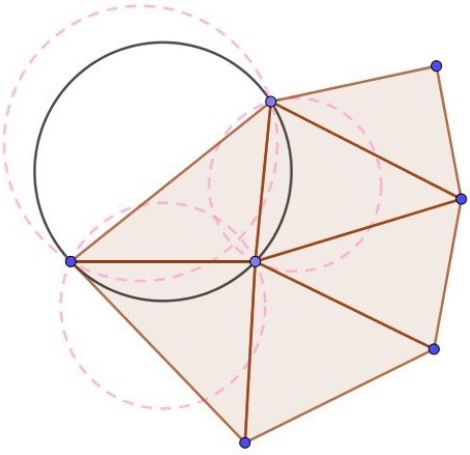
2.2.2. Teselación de Delone

Borís Nikoláievich Delone (1890 - 1980) foi un matemático e alpinista ruso. Como matemático, ideou o algoritmo denominado Triangulación de Delone, utilizado no modelado de superficies definidas por puntos. A ortografía **Delone** é unha transliteración directa do cirílico utilizada en publicacións recentes, mentres que en antigas publicacións francesas e alemás se utilizou a forma afrancesada de **Delaunay**.

Chamamos conxunto de Delone a un conxunto finito de puntos non colineais do plano complexo que non ten subconxuntos cocíclicos de cardinal maior que 3.

A envoltura convexa dun conxunto de Delone admite unha única teselación triangular, tal que os círculos circunscritos a devanditos triángulos non conteñen puntos do conxunto de Delone no seu interior (véxase [3], [10], [16]).

O grafo $G = (V, E)$ é un grafo de Delone se V é un conxunto de Delone e E é o conxunto de lados dos triángulos da súa teselación de Delone. Polas condicións impostas a V, unha aresta (u, v) está en E se e só se hai unha circunferencia que pasa por u e v cuxo círculo aberto non contén puntos de V:

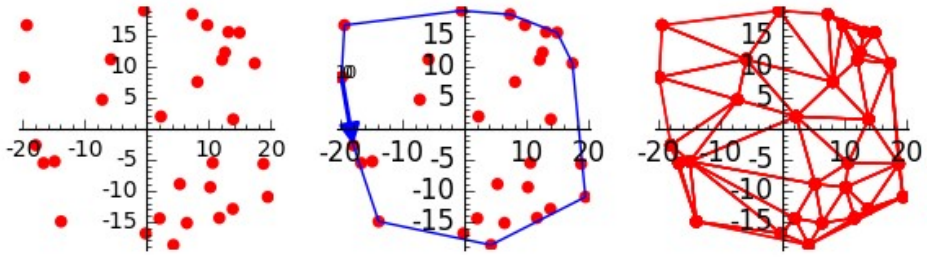


Se V é un conxunto de Delone, a función **delone**(V) determina a lista do circuncentro e os vértices de todos os triángulos da súa teselación e a función **TD**(V) debuxa a teselación e calcula a súa área total.

```
T=TD(V)
```

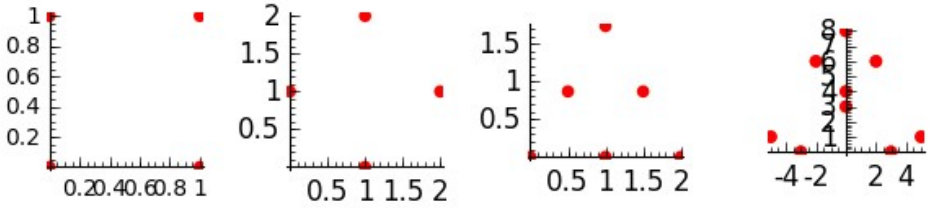
```
show(graphics_array([VP,VC+VP,T[0]]),figsize=[6,3])
print 'área=', T[1]
```

```
area= 1210.64250846728
```



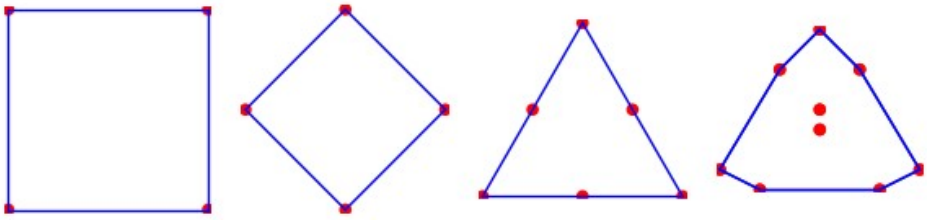
Os seguintes conxuntos Q, R, T e D

```
Q=[0,1,1+I,I]
R=[I,1,2+I,1+2*I]
T=triangular(3,1,sqrt(3)/2)
D=[3,-3, 5+I,-5+I,3*I,4*I,8*I,-2+6*I,2+6*I]

show(graphics_array([plse(Q),plse(R), plse(T),plse(D)]),figsize=
[6,2])
```



non son de Delone porque teñen subconxuntos colineais de cardinal 3 ou subconxuntos cocíclicos de cardinal 4

```
show(graphics_array([pco(Q)+plse(Q),pco(R)+plse(R),
pco(T)+plse(T),pco(D)+plse(D)]),axes=False,figsize=[6,2])
```



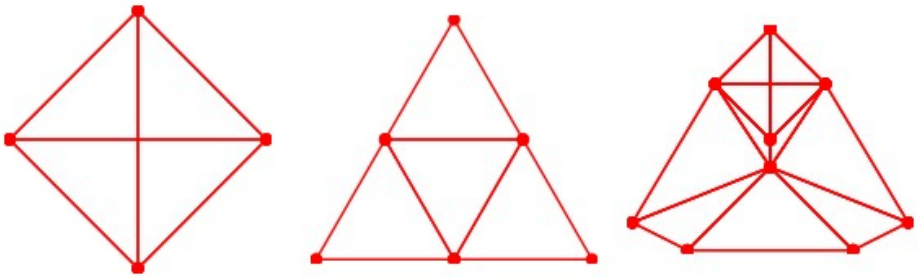
Neles a función **delone**() dá os seguintes resultados:

```
L=[Q,R,T,D]
for a in L:
    print delone(a)
    print
```

```
[]

[[I + 1, 0, 1, 2], [I + 1, 0, 1, 3], [I + 1, 0, 2, 3], [I + 1,
3]]

[[1/6*I*sqrt(3) + 1/2, 0, 1, 3], [1/6*I*sqrt(3) + 3/2, 1, 2, 4
[1/12*sqrt(3)*(sqrt(3) + I) + 1/4*I*sqrt(3) + 3/4, 1, 3, 4],
[2/3*I*sqrt(3) + 1, 3, 4, 5]]

[[0, 0, 1, 4], [17/6*I + 17/6, 0, 2, 4], [17/6*I - 17/6, 1, 3,
[121/38*I + 113/38, 2, 4, 8], [121/38*I - 113/38, 3, 4, 7], [7
5/2, 4, 5, 7], [7/2*I + 5/2, 4, 5, 8], [6*I, 5, 6, 7], [6*I, 5
8], [6*I, 5, 7, 8], [6*I, 6, 7, 8]]
```

Así, para Q non ofrece teselación, para R ofrece dúas, para T ofrece unha e para D ofrece dúas:

```
show(graphics_array([TD(R)[0],TD(T)[0],TD(D)
[0]]),axes=False,figsize=[6,2])
```



A determinación das zonas de proximidade dos puntos dun conxunto V no plano complexo, ou rexións de Voronoi de V, é un problema dual ao da teselación de Delone de V. A zona de proximidade de $v \in V$ é:

$$Z_v(V) = \{z \in \mathbb{C} \,|\, |z - v| \leq |z - w| \quad \forall w \in V\}$$

Implementamos en Sage, a función **voronoi(V)** para determinalas.

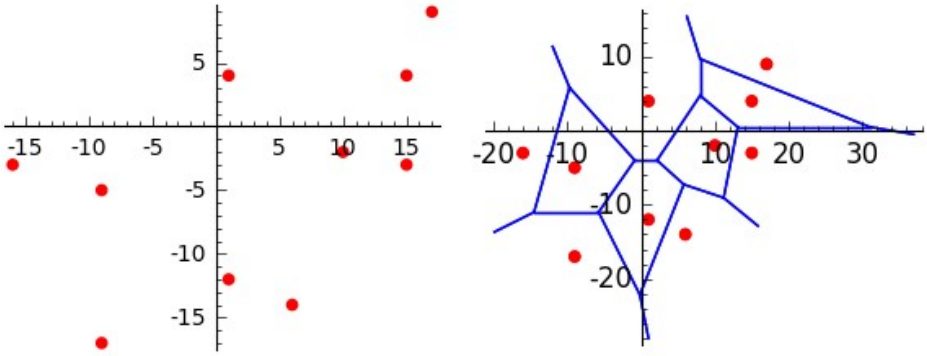
Exemplo 5:

Calcular as zonas de proximidade do conxunto

$$V = [6 - 14i, 10 - 2i, -16 - 3i, 1 + 4i, 1 - 12i,$$

$$15 + 4i, 17 + 9i, 15 - 3i, -9 - 5i, -9 - 17i]$$

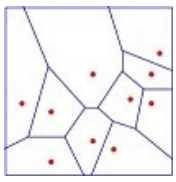
```
V=[6 - 14*I, 10 - 2*I, -16 - 3*I, 1 + 4*I, 1 - 12*I, 15 + 4*I,
17 + 9*I, 15 - 3*I, -9 - 5*I, -9 - 17*I]
VR=voronoi(V)
G=graphics_array([plse(V),VR])
G.show(figsize=[6,5])
```



Para restrinxir as rexións de Voronoi a unha zona $\mathcal{R}$ do plano complexo podemos recortar a rexión desexada cun programa gráfico.

Exemplo 6:

Representar as zonas de Voronoi do conxunto anterior V, sendo $\mathcal{R}$ o cadrado centrado na orixe paralela aos eixes e de lado 40 ou o círculo centrado na orixe de radio 20.



2.2.3. Árbore xeradora de lonxitude mínima

Os problemas de tipo 3 foron abordados por Otakar Boruvka (1926, [5]) no seu intento de electrificar unha zona rural no sur de Moldavia, pero foi Joseph Kruskal (1956, [15]) quen deseñou o algoritmo para calcular a árbore xeradora de lonxitude mínima nun conxunto finito V.

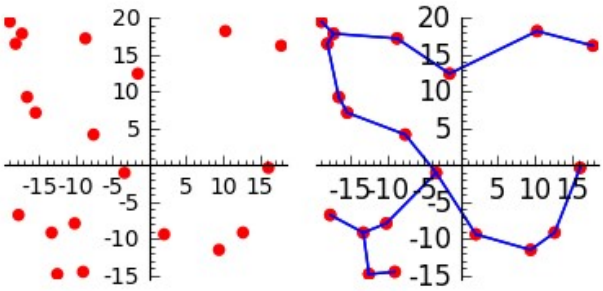
O Algoritmo de Kruskal consiste en ordenar as lonxitudes das arestas de menor a maior e en cada paso seleccionar a aresta de menor lonxitude, de tal xeito que nunca se forme un ciclo. Termínase cando todos os vértices do grafo quedan conectados.

Teorema de Kruskal:

Nun grafo conexo (V, E) o algoritmo de Kruskal produce sempre unha árbore xeradora mínima.

Para o grafo completo (V, K), a función **kruskal**(V) debúxanos a árbore xeradora mínima de V, calcula a suma das lonxitudes de todas as súa arestas, presenta unha lista G onde G[i] é o número de vértices de grado $i+1$ e ordena as arestas de menor a maior.

```
V=listacomplejos(20,20)
K=kruskal(V)
show(graphics_array([plce(V),K[0]]),figsize=[4,3])
```



```
print 'suma de lonxitudes=', K[1]
print 'G=',K[2]
print 'arestas ordeadas'
K[3]
```

```
suma de lonxitudes= 125.512551681647
G= [5, 12, 3]
arestas ordeadas
[[1.61268195516233, 5, 19],
 [2.31670025444501, 7, 19],
 [2.42250928669241, 1, 2],
 [3.36950589908136, 15, 18],
 [3.52063389256103, 6, 12],
 [4.01598711079683, 11, 13],
 [5.10639844453556, 0, 18],
 [5.66569609927876, 12, 18],
 [6.67461885783273, 4, 8],
 [7.38629038257714, 2, 5],
 [7.74679119785861, 10, 13],
 [7.83952052097826, 9, 16],
 [8.40437335724484, 1, 4],
 [8.58047438674583, 14, 17],
 [8.64625440048952, 17, 19],
 [9.42697582607007, 3, 11],
 [9.63323467565941, 8, 15],
 [9.92742288065250, 8, 10],
 [13.2164822529849, 9, 14]]
```

Exemplo 7:

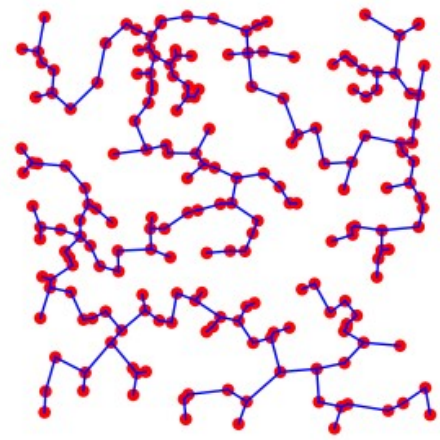
Encárgannos o rego por goteo de 200 limoeiros situados aleatoriamente no cadrado $[-70, 70] \times [-70, 70]$. Presupostar a instalación tendo en conta que:

a. A mangueira dos tramos menores de 7 m custa 3.25€/m e a dos tramos maiores de 7 metros custa 5€/m.
b. O noso operario coloca 50m/h e cobra 5€ á hora.
c. En cada punto de grao 1 debemos poñer un tapón de 1 €.
d. En cada punto de grao 2 debemos poñer un sten de 3 €.
e. En cada punto de grao n>2 desperdiciamos n dm de mangueira e debemos poñer un distribuidor de 3n €.

f. O IVA é o 21%.

g. Queremos gañar o 30%.

```
L=listacomplejos(70,200)
K=kruskal(L)
K[0].show(figsize=[3,3])
```



```
print 'a)'
V=[a[0] for a in K[3]]
m=0
for v in V:
    if v<7:
        m=m+3.25
    if v>7:
        m=m+5
print 'o prezo da mangueira é =', round(m,2), 'euros'
```

```
a)
o prezo da mangueira é = 793.75 euros
```

```
print 'b)'
print 'o operario cobra', round((m/50)*5,2), 'euros'
```

```
b)
o operario cobra 79.38 euros
```

```
print 'c)'
print 'o número de vértices de grado 1 é', K[2][0]
print 'gastamos en tapóns', K[2][0], 'euros'
```

```
c)
o número de vértices de grado 1 é 52
```

```
gastamos en tapóns 52 euros
```

```
print 'd)'
print 'o número de vértices de grado 2 é', K[2][1]
print 'gastamos en sten', 3*K[2][1], 'euros'
```

```
d)
o número de vértices de grado 2 é 100
gastamos en sten 300 euros
```

```
print 'e)'
print 'hai', K[2][2], 'puntos de grado 3 e desperdiciamos',
K[2][2], 'dm de mangueira'
print 'gastamos en distribuidores', 3*3*K[2][2], 'euros'
print 'hai', K[2][3], 'puntos de grado 4 e desperdiciamos',
K[2][3], 'dm de mangueira'
print 'gastamos en distribuidores', 3*4*K[2][3], 'euros'
```

```
e)
hai 46 puntos de grado 3 e desperdiciamos 46 dm de mangueira
gastamos en distribuidores 414 euros
hai 2 puntos de grado 4 e desperdiciamos 2 dm de mangueira
gastamos en distribuidores 24 euros
```

```
print 'f)'
g=(m+(m/50)*5 +K[2][0]+3*K[2][1]+3*3*K[2][2]+3*4*K[2][3])*1.21
print 'En total gastamos', round(g,2), 'euros'
```

```
f)
En total gastamos 2012.38 euros
```

```
print 'g)'
print 'Debemos cobrar', round(g*1.3,2), 'euros'
```

```
g)
Debemos cobrar 2616.1 euros
```

Teorema de Kruskal - Delone:

Se V é un conxunto de Delone, a $MST(V)$ é subgrafo do grafo de Delone (V, A).

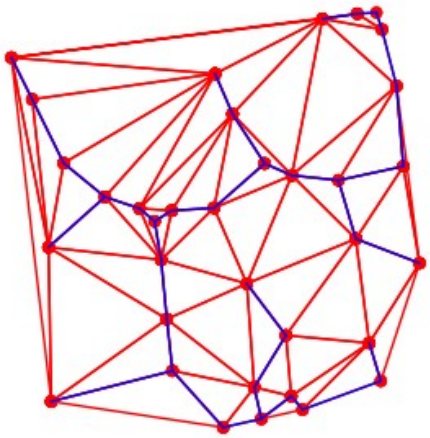
Demostración:

Se supoñemos que a aresta (x, y) da $MST(V)$ non está en A, o círculo aberto cuxo diámetro é xy debe conter un punto $z \in V$. Se eliminamos a aresta (x, y) da árbore $MST(V)$, esta divídese en dúas subárbores.

Asumamos que z queda na mesma subárbore que x. Engadindo a aresta (y, z) obtemos unha árbore xeradora de V de lonxitude menor que a $MST(V)$, o cal é absurdo.

Comprobación experimental de K-D:

```
n=35
V=listacomplejos(20,n)
K=kruskal(V)
show(TD(V)[0]+K[0],figsize=[3,3])
```



2.2.4. Problemas de Steiner

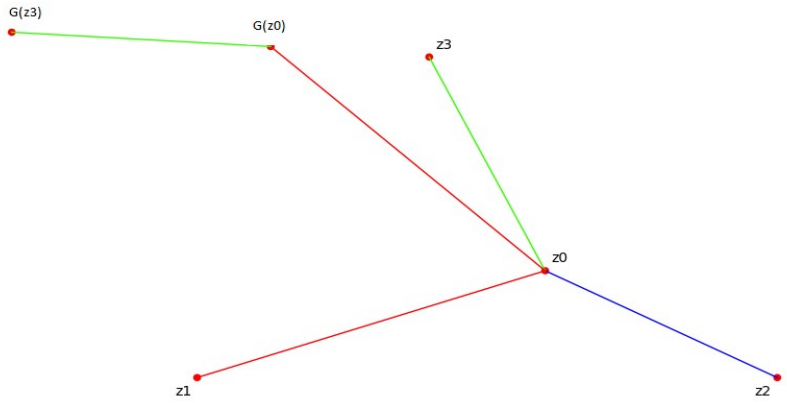
Para un conxunto finito de puntos V do plano complexo, queremos achar subconxuntos $R \subset \mathbb{C}$ tales que $V \cap R = \emptyset$ e a lonxitude da MST($V \cup R$) sexa menor que a lonxitude da MST(V) (véxase [17], [22]).

Os problemas deste tipo iniciounos Fermat no século $XVII$, para triángulos $V = \{z_1, z_2, z_3\}$ con todos os seus ángulos menores ca 120^o.

Preguntábase se existe $z_0 \in \mathbb{C}$ tal que

$$|z_1 - z_0| + |z_2 - z_0| + |z_3 - z_0| = \min\{|z_1 - z| + |z_2 - z| + |z_3 - z| \,|\, z \in \mathbb{C}\}$$

Torricelli, en 1640, resolveuno coa seguinte construción:



Se supoñemos que z_0 é a solución, ao facer un xiro G de 60º con centro en z_1, obtemos os puntos $G(z_0)$ e $G(z_3)$ e, se a suma de distancias $|z_1 - z_0| + |z_2 - z_0| + |z_3 - z_0|$ ha ser mínima, tamén haberá ser mínima a lonxitude da poligonal $[z_2, z_0, G(z_0), G(z_3)]$. En consecuencia, z_0 debe acharse sobre a recta $[z_2, G(z_3)]$ que está ben determinada porque $G(z_3)$ é o vértice do triángulo equilátero exterior ao dado que se pode construír sobre o lado $[z_1, z_3]$. Do mesmo xeito podemos asegurar que z_0 debe estar na recta determinada por z_1 e o vértice do triángulo equilátero exterior ao dado que se pode construír sobre o lado $[z_2, z_3]$. Xa que logo, z_0 será o punto de intersección de ambas rectas que estará no interior do triángulo dado oposto que esiximos que os seus ángulos sexan menores que 120º.

Por outra banda, se z_0 fose a solución, a recta $[z_3, z_0]$ debería ser ortogonal á elipse de focos z_1 e z_2 que pasa por z_0 e, xa que logo, os ángulos (z_3, z_0, z_1) e (z_3, z_0, z_2) deben ser iguais. Razoando do mesmo xeito coa recta $[z_1, z_0]$ e a elipse de focos z_2 e z_3 que pasa por z_0, concluiremos que os ángulos (z_3, z_0, z_1) e (z_1, z_0, z_2) tamén deben ser iguais. Así, o punto z_0 debe ser o punto isógono para o triángulo V, é dicir, desde ese punto débense ver os 3 vértices con ángulo 120º.

Así as cousas, podemos dar unha nova construción da solución z_0 baseándonos no coñecemento dos arcos capaces: as circunferencias circunscritas aos triángulos equiláteros exteriores a V construídos sobre os lados $[z_1, z_3]$ e $[z_2, z_3]$ débense cortar, ademais de no vértice z_3, no punto

isógono z_0.

Atopamos un $R = V \cup z_0$ que cumpre a condición requerida ao principio e, xa que logo, o problema de Steiner é consistente.

No caso dun triángulo cun ángulo maior ou igual que $120º$ é fácil observar que a solución z_0 coincide co vértice de incidencia dos dous lados máis curtos. Neste caso só $R = \emptyset$ cumpre as condicións requiridas.

Para calquera triángulo podemos establecer as seguintes propiedades da árbore de Steiner.

2.2.4.1. Propiedades da árbore de Steiner dun triángulo

1. De existir un punto de Steiner é único.
2. O grao do punto de Steiner é 3 e o grao de cada vértice é 1.
3. O punto de Steiner é un punto isógono.

Estas propiedades pódense xeneralizar para conxuntos V de $|V| > 3$.

2.2.4.2. Propiedades da árbore de Steiner

1. O grao de cada punto de Steiner é 3 e cada punto de Steiner é isógono.
   En efecto:
   O grao non pode ser 1 porque a aresta que o une co resto da árbore ten lonxitude positiva e iso contradiría a minimalidade.
   Non pode ser 2 porque as dúas arestas que o unen co resto da árbore, sv_1 e sv_2 suman maior lonxitude que a aresta v_1v_2.
   Non pode ser maior que 3 porque polo menos un dos ángulos que forman dúas arestas que inciden no punto de Steiner é menor que $120º$ e nese caso poderiamos substituír esas dúas arestas pola conexión de Steiner deses tres puntos reducindo así o grao do punto de Steiner.
   As tres arestas que inciden no punto de Steiner teñen que formar entre si ángulos de $120º$ polo tanto o punto de Steiner é isógono.
2. Un conxunto de Steiner S para V cumpre que $|S| \leq |V| - 2$:
   En efecto:
   Se T é unha árbore xeradora de $V \cup S$ teremos que $|T| = |V| + |S| - 1$ e, xa que logo,

$$2|V| + 2|S| - 2 = 2|T| = \sum_{v \in S} g(v) + \sum_{v \in V} g(v) \geq 3|S| + |V|$$

$$\Rightarrow \quad |V| - 2 \geq |S|$$

3. O grao de cada vértice dun SMT(V) é todo o máis 3.

2.2.4.3. Algoritmos de Steiner

Incluímos nos DATA a función **Steiner3(V)** que actúa sobre conxuntos de cardinal 3 resolvendo alxebricamente a construción de Torricelli. Abusando da definición, cando un dos ángulos é superior a 120^o, aínda que sabemos que non existe punto de Steiner, esta función propón como tal ao vértice no que inciden os dous lados menores.

Tamén incluímos nos DATA as funcións **Steiner4(V)**para conxuntos de cardinal 4. Esta función calcula os niveis de convexidade do conxunto V e, se ten dous niveis, fai un cálculo directo baseado na función **Steiner3()**. Se ten un só nivel de convexidade, fai unha programación iterativa **Steiner4a(V)** con inicio en **optimiza(V)** mediante **iter4(V)** que se detén cando os posibles puntos de Steiner a engadir achéganse suficientemente á isogonía.

As funcións **Steiner3(V)** e **Steiner4(V)** devólvennos catro saídas:

1. Os puntos de Steiner de V en sentido amplio.
2. O debuxo do SMT(V).
3. A lonxitude $S(V)$ do SMT(V).
4. A ganancia $G(V)$ definida como

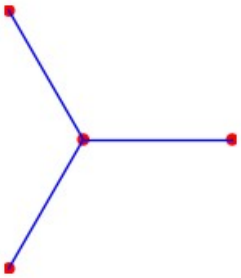
$$100\frac{MST(V) - SMT(V)}{MST(V)}$$

Exemplo 8:

Na configuración $V =$**radial(**3,1,1,0**)** a construción de Torricelli dános os resultado exactos SMT(V)=3 e $G(V) = 50(2 - \sqrt{3})$. Steiner3(V) dános SMT(V) =3 e $G(V) = 13.3974596215561$ con 14 cifras decimais exactas.

```
V=radial(3,1,1,0)
S=Steiner3(V)
print S[2]
print S[3]
show(S[1],figsize=[2,2])
```

```
3.00000000000000
13.3974596215561
```



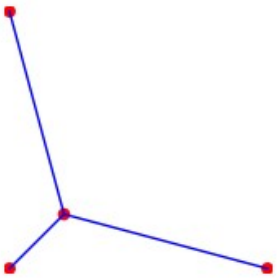
Exemplo 9:

Na configuración $V=[0,1,i]$ a construción de Torricelli dános os resultados exactos SMT$(V)=\frac{\sqrt{2}}{2}+\sqrt{\frac{3}{2}}$ e $G(V)=\frac{4-\sqrt{2}-\sqrt{6}}{4}$.

Steiner3(V) dános SMT$(V)=1.93185165257814$ e $G(V)=3.40741737109317$ con 14 cifras decimais exactas.

```
V=[0,1,I]
S=Steiner3(V)
print 'O punto de Steiner xeneralizado é', S[0]
print 'O SMT(V) é', S[2]
print 'A ganancia é', S[3]
show(S[1],figsize=[2,2])
```

```
O punto de Steiner xeneralizado é 0.211324865405187 +
0.211324865405187*I
O SMT(V) é 1.93185165257814
A ganancia é 3.40741737109318
```

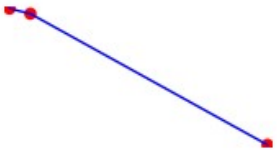


Exemplo 10:

Para un conxunto calquera V de cardinal 3, calcular **Steiner3(V)**.

```
V=listacomplejos(20,3)
S=Steiner3(V)
print 'O punto de Steiner xeneralizado é', S[0]
print 'O SMT(V) é', S[2]
print 'A ganancia é', S[3]
show(S[1],figsize=[2,2])
```

```
O punto de Steiner xeneralizado é 17.6769088912937 -
19.8789060175135*I
O SMT(V) é 38.8877888240593
A ganancia é 0.000000000000000
```

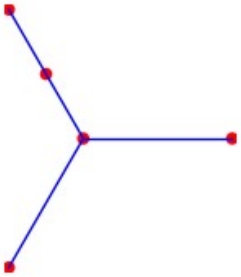


Exemplo 11:

Calcular os puntos de Steiner de $V =$**radial(**3,1,1,0**)+**$[0.5e^{i\frac{2\pi}{3}}]$.

```
V=radial(3,1,1,0)+[0.5*exp(I*2*pi/3)]
S=Steiner4(V)
print 'Os puntos de Steiner son', S[0].n()
print 'O SMT(V) é', S[2]
print 'A ganancia é', S[3]
show(S[1],figsize=[2,2])
```

```
Os puntos de Steiner son -2.22044604925031e-16 +
2.22044604925031e-16*I
O SMT(V) é 3.00000000000000
A ganancia é 4.63327506379598
```



Exemplo 12:

Calcular os puntos de Steiner de $V =$ **radial(**4, 1, 1, 0**)**.

```
V=radial(4,1,1,0)
S=Steiner4(V)
print 'Os puntos de Steiner son', S[0]
print 'O SMT(V) é', S[2]
print 'A ganancia é', S[3]
show(S[1],figsize=[2,2])
```

```
Os puntos de Steiner son [0.211324014792977 + 0.21132571601865
-0.211325580152501 - 0.211324150658758*I]
O SMT(V) é 3.86370330515706
A ganancia é 8.93163974768554
```

Exemplo 13:

Calcular os puntos de Steiner de $V = [0, i, 2 + i, 2]$

```
V=[0,I,2+I,2]
S=Steiner4(V)
print 'Os puntos de Steiner son', S[0]
print 'O SMT(V) é', S[2]
print 'A ganancia é', S[3]
show(S[1],figsize=[2,2])
```

```
Os puntos de Steiner son [0.288675134592770 + 0.50000153585083
1.71132486540651 + 0.500001234971038*I]
O SMT(V) é 3.73205080757059
A ganancia é 6.69872981073523
```

Exemplo 14:

Calcular os puntos de Steiner na configuración romboidal $R = [0, 1, 1 + e^{-i\frac{\pi}{3}}, e^{-i\frac{\pi}{3}}]$.

```
R=[0,1,I*exp(-I*pi/6),1+I*exp(-I*pi/6)]
S=Steiner4(R)
print 'Os puntos de Steiner son', S[0]
print 'O SMT(V) é', S[2]
print 'A ganancia é', S[3]
show(S[1],figsize=[2,2])
```

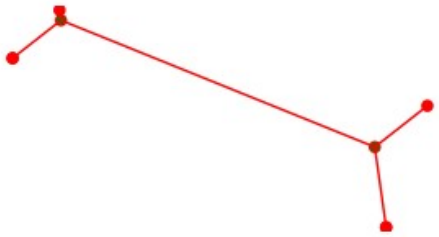
```
Os puntos de Steiner son [0.500000000000000 + 0.28867513459481
0.642857142857143 + 0.701068184015974*I]
O SMT(V) é 2.68222577003120
A ganancia é 10.5924743322934
```

Exemplo 15:

Para un conxunto calquera V de cardinal 4, calcular **Steiner4(V)**.

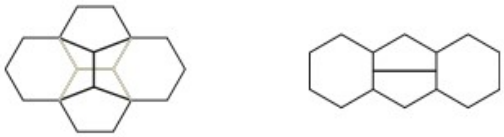
```
V=listacomplejos(20,4)
S=Steiner4(V)
print 'Os puntos de Steiner son', S[0]
print 'O SMT(V) é', S[2]
print 'A ganancia é', S[3]
show(S[1],figsize=[3,3])
```

```
Os puntos de Steiner son [-15.3244290900774 + 14.3431144544502
14.2663319333721 + 2.41609915428285*I]
O SMT(V) é 52.5831914240181
A ganancia é 3.04917131049260
```



Exemplo 16: Deformacións en membranas de grafeno

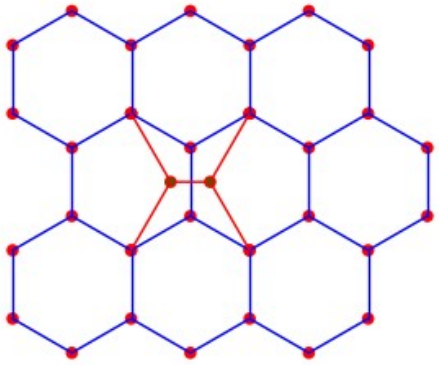
As dúas dislocacións que se producen nuna membrana de grafeno por causas aínda descoñecidas son as deformacións de Stone-Wales e a mitose (véxase [23], [24])



Esquemas de Zsoldos

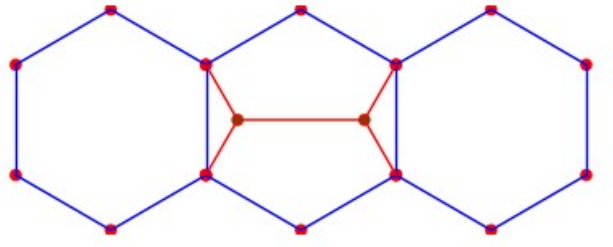
Mediante a función **Steiner4g(V)** podemos aplicar o método **iter4()** á lista V e non a **optimiza4(V)**. En consecuencia obtemos unha conexión de Steiner mínima, aínda que só localmente. Isto é o que cremos que sucede nas deformacións de Stone-Wales.

```
H=hexagonal(3,3,1)
V=num([H[0][2][1],H[0][5][1],H[0][5][2],H[0][2][2]])
show(H[1]+H[2]+Steiner4g(V)[1],figsize=[3,3])
```



Na mitose, os defectos prodúcense ao conectarse catro átomos de carbono por unha árbore de Steiner mínima, aínda que para iso teña que captar dous novos átomos de carbono que poderían ser considerados átomos de Steiner.

```
H=hexagonal(1,3,1)
V=num(H[0][1][1:3]+H[0][2][1:3])
show(H[1]+H[2]+Steiner4(V)[1],figsize=[4,4])
```



Incorporamos nos DATA a función **visualsteiner(**L, LI**)** que actúa sobre a gráfica subindicada de MST(L) para $|L| \geq 5$ e permítenos seleccionar visualmente os puntos de L máis adecuados para aplicarlles as reducións **Steiner3()** ou **Steiner4()**.

Exemplo 17:

Aplicar **visualsteiner()** á seguinte lista

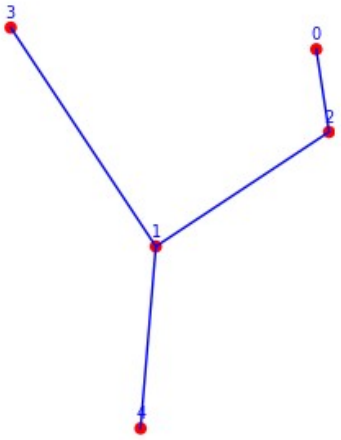
$$V = [16.3 + 10.3i, 3.7 - 5.2i, 17.3 + 3.8i, -7.7 + 12i, 2.5 - 19.5i]$$

Solución:

Pintamos a conexión de Kruskal indicando os vértices.

```
V=[16.3 + 10.3*I, 3.7 - 5.2*I, 17.3+ 3.8*I, -7.7+ 12*I, 2.5-
19.5*I]
CV=copy(V)
K=kruskal(CV)
print 'A lonxitude do MST(V) é', K[1]
VO=[[k,V[k]] for k in range(len(V))]
TOP=[]
for a in VO:
    TOP.append(text(a[0],ri(a[1]+1.1*I),fontsize=7))
show(K[0]+sum(TOP), figsize=[3,3])
```

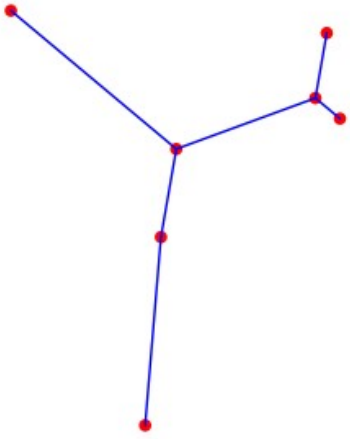
```
A lonxitude do MST(V) é 57.8699365020841
```



Á vista do resultado aplicamos un **Steiner4()** nos vértices $[3, 1, 2, 0]$

```
VI=[[3,1,2,0]]
kop=visualsteiner(V,VI)
print 'a lonxitude reducida é', kop[1]
show(kop[0],figsize=[3,3])
```

```
a lonxitude reducida é 56.2247783621150
```

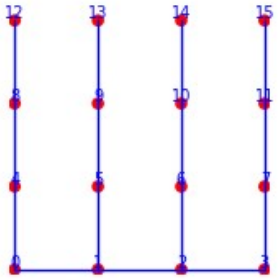


Exemplo 18:

Atopar en **rectangular(**4,4,1,1**)** unha conexión de Steiner menor cá de Kruskal.

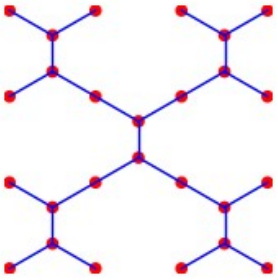
```
V=rectangular(4,4,1,1)
CV=copy(V)
K=kruskal(CV)
print 'A lonxitude do MST(V) é', K[1]
VO=[[k,V[k]] for k in range(len(V))]
TOP=[]
for a in VO:
    TOP.append(text(a[0],ri(a[1]+.1*I),fontsize=7))
show(K[0]+sum(TOP), figsize=[2,2])
```

```
A lonxitude do MST(V) é 15.0000000000000
```




```
VI=[[0,1,4,5],[2,3,6,7],[5,6,9,10],[8,9,12,13],[10,11,14,15]]
kop=visualsteiner(V,VI)
print 'a lonxitude reducida é', kop[1]
show(kop[0],figsize=[2,2])
```

```
a lonxitude reducida é 13.6602540378472
```



Exemplo 19:

Na lista de complexos

$$L = [-18.8 + 15.2i, -7.8 + 5.3i, -15.5 - 6.3i,$$

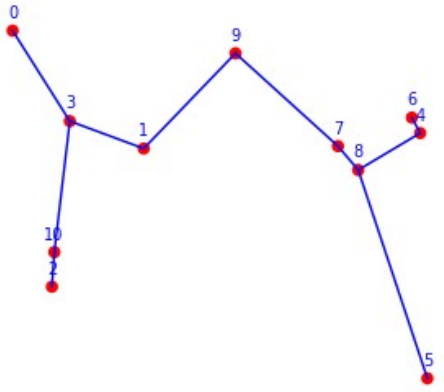
$$-14.0 + 7.6i, 15.4 + 6.6i, 16 - 14i, 14.7 + 7.9i,$$

$$8.5 + 5.5i, 10.2 + 3.5i, -0.1 + 13.3i, -15.3 - 3.4i]$$

añadir puntos isogónicos e obter unha conexión de menor lonxitude cá do MST(L).

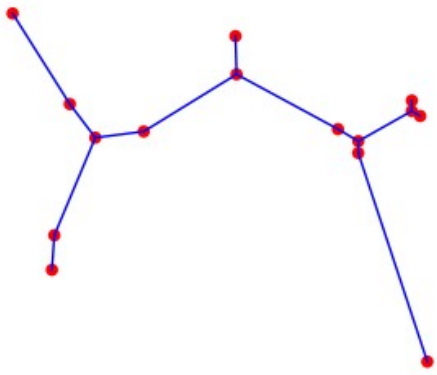
```
L=[-18.8 + 15.2*I, -7.8 + 5.3*I, -15.5 - 6.3*I, -14.0 + 7.6*I,
15.4 + 6.6*I, 16.0 - 14.0*I, 14.7 + 7.9*I, 8.5 + 5.5*I, 10.2 +
3.5*I, -0.1 + 13.3*I, -15.3 - 3.4*I]
K=kruskal(L)
print 'A lonxitude do MST(L) é', K[1]
LO=[[k,L[k]] for k in range(len(L))]
TOP=[]
for a in LO:
    TOP.append(text(a[0],ri(a[1]+1.5*I),fontsize=7))
show(K[0]+sum(TOP), figsize=[3,3])
```

```
A lonxitude do MST(L) é 80.8905251694785
```




```
LI=[[3,1,10],[1,9,7],[7,8,4,6]]
kop=visualsteiner(L,LI)
print 'a lonxitude reducida é', kop[1]
show(kop[0],figsize=[3,3])
```

```
a lonxitude reducida é 78.6900772626205
```



2.2.5. Ciclo Hamiltoniano Mínimo

Os problemas de tipo 4 tratan de atopar un ciclo hamiltoniano C en V de lonxitude mínima.

Historicamente os ciclos hamiltonianos de lonxitude mínima nun conxunto de puntos contidos nun rectángulo do plano complexo computáronse facendo particións en dito rectángulo (véxase [13]).

Se R é un rectángulo acoutado que contén a V atopáronse boas aproximacións á solución óptima usando particións rectangulares $\{R_i \mid i \in I\}$ de R que inducen particións $\{V_i \mid i \in I\}$ en V, tales que $V_i = V \cap R_i$ e $|V_i| < k \quad \forall i \in I$ sendo k un natural pequeno. Nestes conxuntos V_i pódense atopar facilmente camiños hamiltonianos óptimos e as computacións habituais seguiron diferentes estratexias para conectar os camiños óptimos de cada V_i e formar así o desexado ciclo hamiltoniano óptimo en V.

No traballo Designing Hamiltonian Cycles presentado no ACA2013 en Málaga por F. de Arriba, E. Corbacho e R. Vidal (véxase [2]), propúxose a partición do conxunto V nos seus diferentes niveis de convexidade, $\{V_1, \cdots, V_n\}$, pensando que a xeometría da distribución dos puntos facilitaría a conexión entre os diferentes elementos da partición no mesmo sentido que a integral de Lebesgue mellora a integral de Riemann.

As ideas básicas para conectar os diferentes niveis de convexidade son:

1. Se $V = V_1$, bastaría fixar unha orientación na fronteira e, de acordo con ela, ordenar os puntos de V. Deste xeito construiriamos un camiño hamiltoniano pechado en V de lonxitude mínima, posto que é o único que

non ten cruces.

2. Se existe un $z_0 \in V$ tal que $V\prime = V \setminus \{z_0\}$ ten a propiedade expresada en 1., podemos construír un ciclo hamiltoniano óptimo en $V\prime$ e reemplazar a máis axeitada aresta (z_i, z_{i+1}) pola poligonal $z_i z_0 z_{i+1}$ de tal xeito que o novo ciclo, que obviamente seguiría sendo hamiltoniano, aumentase a súa lonxitude o menos posible.
3. Se existise un $\{z_0, \cdots, z_n\} \subset V$ tal que $V\prime = V \setminus \{z_0, \cdots, z_n\}$ tivese a propiedade expresada en 1., poderiamos considerar o ciclo hamiltoniano óptimo en $V\prime$ e estudar a orde adecuada para incorporar os puntos z_i a devandito ciclo, de modo que se preserve a condición de hamiltoniano e creza a súa lonxitude o menos posible.

En calquera conxunto V pódese achar a súa partición en niveis de convexidade $\{V_1, \cdots, V_n\}$ e ao camiño hamiltoniano existente en V_1 incorporarlle os puntos de V_2 segundo a heurística de 3. Ao ciclo hamiltoniano así formado que designamos V_{12}, incorporámoslle os puntos de V_3 polo mesmo procedemento, e así sucesivamente ata chegar ao ciclo $V_{1\cdots n}$.

A función **pegalistas(V_1, V_2)** incorpora os puntos de V_2 ao ciclo V_1. A orde de incorporación decídese polo mínimo aumento de lonxitude do camiño. Os empates resólvense polo máximo ángulo de incorporación. Unha vez decidida a incorporación dun certo $w \in V_2$ entre os puntos (z_i, z_{i+1}) de V_1 reemplázase o tramo $[z_{i-1}, z_i, w, z_{i+1}, z_{i+2}]$ pola súa reordenación óptima e, sucesivamente, procédese a incorporar un novo elemento de V_2 a este novo ciclo.

A función **cicloham(V)** obtén a lista $[V_1, \cdots, V_n]$, computa V_{12} pegando V_2 a V_1, V_{123} pegando V_3 a V_{12} e, por iteración, propón en principio, como ciclo hamiltoniano óptimo a lista $C = V_{1\cdots n}$.

A función **mejoratramo(C, n, m)** que equivale a

pegalistas(restalistas($C, C[n : m+1]$),$C[n : m+1]$)

corta e pega de novo o tramo $C[n : m+1]$, intentando acurtar a lonxitude do ciclo C.

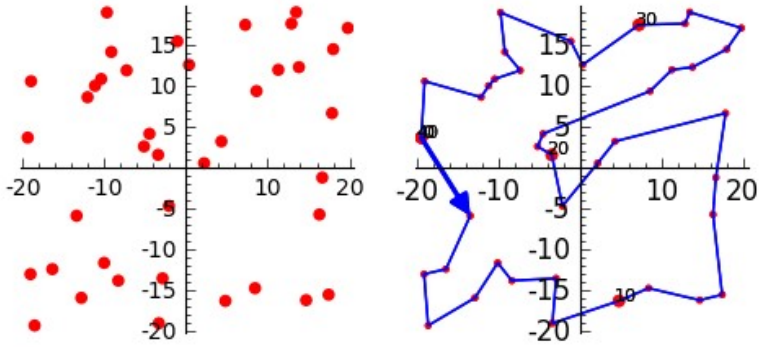
A función **ciclohamop(C)** establece un proceso recursivo no que utilizamos a función **mejoratramo()** en cada un dos fiordes que aparecen no ciclo C e a continuación a función **mejoratramo()** en grupos de tres puntos, por se o punto central debe cambiar de beira de fiorde para acurtar a lonxitude do ciclo

hamiltoniano. Este proceso finaliza nun número finito de pasos e execútase ao final de **cicloham()** devolvendo un ciclo hamiltoniano C que non é necesariamente óptimo, pois pode conter cruces de arestas, pero que como iremos comprobando é suficientemente bo.

Exemplo 20:

Nun conxunto V de corenta puntos obter o ciclo hamiltoniano mínimo.

```
V=listacomplejos(20,40)
show(graphics_array([plse(V),vercamino(cicloham(V))]),
aspect_ratio=1,figsize=[5,3])
```



Se C presenta un cruzamento das arestas $C[m:m+1]$ e $C[n-1:n]$ a función **descruza(**C, m, n**)** devólvenos un ciclo hamiltoniano sen ese cruzamento de arestas e, polo tanto, de menor lonxitude que C que tomaremos como óptimo.

Exemplo 21:

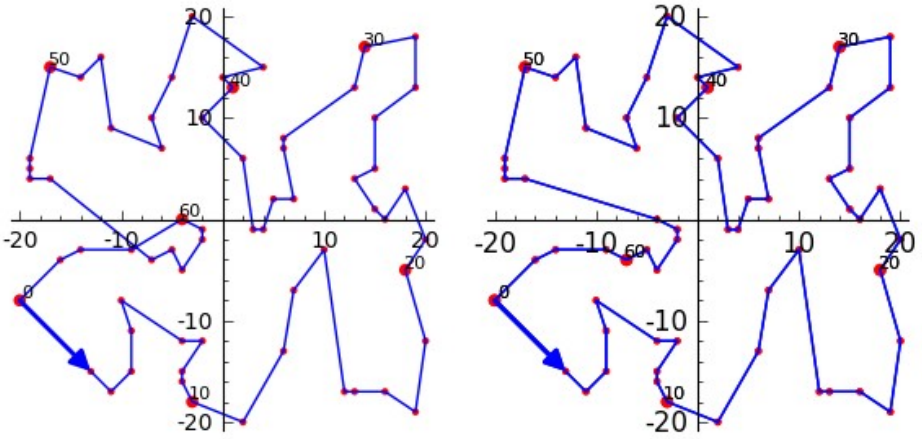
No conxunto de complexos,

$$V = [4i-19, 2i+5, -17i+16, -5i-4, -18i-3, -7i+7, 4i+13,$$
$$-17i+12, 13i+19, -i+4, 5i+15, -20i+2, 5i-19, -15i-9,$$
$$-3i-5, 10i+15, 15i-17, 6i-19, -12i-4, -8i-10, -i+3,$$
$$-19i+19, -2i-2, 7i+6, 17i+14, -3i-14, -5i+18, 8i+6,$$
$$-3i-9, 13i+1, -i-2, 4i-17, 14i, 10i-7, 18i+19, -12i+20,$$

$$7i-6, 2i+7, -13i+6, -15i-4, -15i-13, 14i-14, -2i+20,$$
$$20i-3, 16, -4, 10i-2, 15i+4, -8i-20, -3i+10, -17i+13,$$
$$14i-5, 6i+2, -4i-16, -11i-9, 9i-11, 13i+13, -12i-2,$$
$$16i-12, 3i+18, -4i-7, i+15, -17i-11, -16i-4]$$

achar o ciclo hamiltoniano óptimo.

```
V=[4*I - 19, 2*I + 5, -17*I + 16, -5*I - 4, -18*I - 3, -7*I + 7,
4*I + 13,-17*I + 12, 13*I + 19, -I + 4,
5*I + 15, -20*I + 2, 5*I - 19, -15*I - 9,-3*I - 5, 10*I + 15,
15*I - 17, 6*I - 19, -12*I - 4, -8*I - 10,
-I + 3,-19*I + 19, -2*I - 2, 7*I + 6, 17*I + 14, -3*I - 14, -5*I
+ 18, 8*I + 6,-3*I - 9, 13*I + 1, -I - 2, 4*I - 17, 14*I,10*I -
7, 18*I + 19, -12*I +20, 7*I - 6, 2*I + 7, -13*I + 6, -15*I - 4,
-15*I - 13,
14*I - 14, -2*I+ 20, 20*I - 3, 16, -4, 10*I - 2, 15*I + 4, -8*I
- 20, -3*I + 10, -17*I+ 13, 14*I - 5,
6*I + 2, -4*I - 16, -11*I - 9, 9*I - 11, 13*I + 13,-12*I - 2,
16*I - 12, 3*I + 18, -4*I - 7, I + 15,
-17*I - 11, -16*I - 4]
W=[z.n() for z in V]
C=cicloham(W)
VC=vercamino(C)
D=descruza(C,54,60)
show(graphics_array([VC,D[1]]),figsize=[6,3])
```



Nun intento de avaliar a valía do noso método **cicloham()** implementamos en Sage dúas funcións para calcular cotas inferiores da lonxitude do ciclo hamiltoniano óptimo:

1. A función **cotaK(**V**)**, que para o conxunto de vértices V suprime un vértice v e todas as súas arestas e no novo grafo resultante calcula a árbore xeradora de lonxitude mínima usando **kruskal(**$V \setminus \{v\}$**)** e engádelle as dúas arestas de lonxitude mínima que saen de v. Repítese este procedemento para todo $v \in V$ e quedamos co máximo dos valores obtidos. Así, a función **cotaK(**V**)** devólvenos unha boa cota inferior para a lonxitude do ciclo hamiltoniano mínimo de V.
2. A función **cotaV(**V**)** calcula para cada vértice de V a suma da lonxitude das súas dúas arestas menores, sendo o valor da cota o promedio de todas as sumas.

Como non sempre **cotaK(**V**)**$\leq$**cotaV(**V**)** nin o contrario, consideramos o máximo de ambas cotas implementado na función **cotaKV(**V**)**.

Se $C =$**cicloham(**V**)**, podemos tomar como estimación da valía do algoritmo, a porcentaxe

$$bondad(V) = 100 \cdot \frac{cotaKV(V)}{lon(C)[0]}$$

```
print 'bondade=',bondad(C)
```

```
bondade= 88.7581475525486
```

Exemplo 22:

O conxunto de puntos

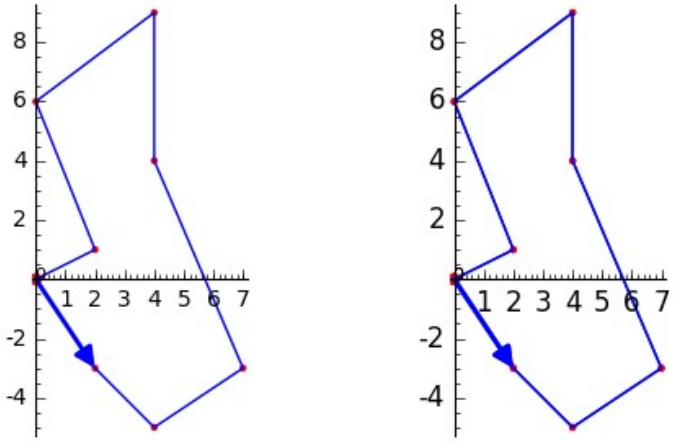
$$V = [0, 6i, 2 - 3i, 4 - 5i, 7 - 3i, 4 + 4i, 2 + i, 4 + 9i]$$

ten **bondad(**V**)**=100 e, xa que logo, é óptimo, como tamén podemos comprobar coa función **optimizac(**V**)**.

```
V=[0,6*I,2-3*I,4-5*I,7-3*I,4+4*I,2+I,4+9*I]
C=cicloham(V)
OC=optimizac(V)
VC=vercamino(C)
VOC=vercamino(OC[0])
show(graphics_array([VC,VOC]),figsize=[6,3])
```

```
print 'bondade =',bondad(C)
```

```
bondade = 100
```

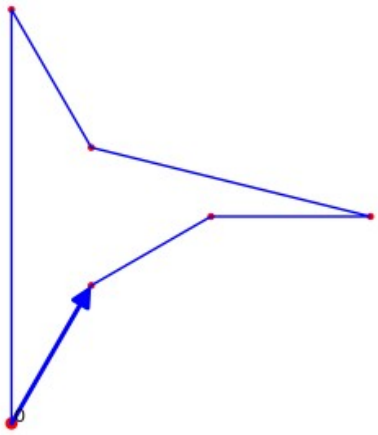


Exemplo 23:

Calcular o ciclo hamiltoniano óptimo do conxunto $R =$ **radial(**3,2,1,2**)**.

```
R=radial(3,2,1,2)
CR=cicloham(R)
VCR=vercamino(CR)
show(VCR,figsize=[3,3])
print 'bondade =',bondad(CR)
```

```
bondade = 100
```



Unha valía do 100% é condición suficiente pero non necesaria para ser óptimo, debido a que **cotaK(V)** e **cotaV(V)** poden non ser suficientemente grandes. Como non dispoñemos de cotas maiores, aceptamos que **cicloham(V)** pode ser un ciclo hamiltoniano óptimo en V aínda que a súa valía non sexa do 100%. A heurística dos exemplos realizados lévanos a confiar na función **cicloham()** para valías maiores ou iguais ao 80%.

Exemplo 24:

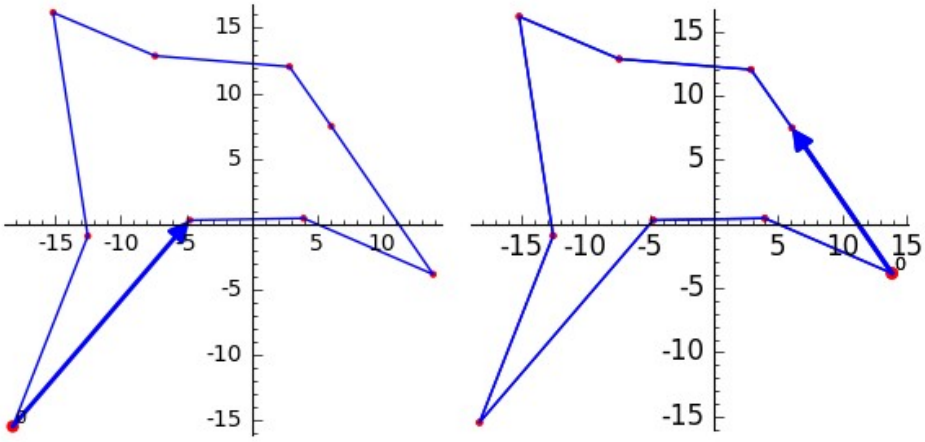
A lista

$$V = [3.97 + 0.439i, -7.38 + 12.86i, -12.52 - 0.92i,$$
$$-4.73 + 0.29i, 2.89 + 12.035i, 13.88 - 3.85i,$$
$$-18.26 - 15.48i, 6.09 + 7.48i, -15.17 + 16.17i]$$

ten valía de $88, 44$. Utilizando **optimizac(V)** comprobamos que **cicloham(V)** é un ciclo óptimo.

```
L=[3.97 + 0.439*I,  -7.38 + 12.86*I,  -12.52 - 0.92*I,  -4.73 +
0.29*I,  2.89 + 12.035*I,  13.88 - 3.85*I,  -18.26 - 15.48*I,
6.09 + 7.48*I,  -15.17 + 16.17*I]
C=cicloham(L); OC=optimizac(L)
VC=vercamino(C); VOC=vercamino(OC[0])
show(graphics_array([VC,VOC]),figsize=[6,3])
print 'bondade =',bondad(C)
```

```
bondade = 88.4399860381631
```



2.2.6. Camiño Hamiltoniano Mínimo

Os problemas de tipo 5 tratan de atopar unha árbore xeradora hamiltoniana de lonxitude mínima en V.

Para listas V de complexos con $|V| < 9$, a función **optimiza(V)** devólvenos a árbore hamiltoniana de mínima lonxitude. Con todo, para $|V| > 9$, o tempo de cálculo de **optimiza(V)** pódese facer insoportable. Convén, xa que logo, deseñar unha aproximación satisfactoria da árbore hamiltoniana mínima que se poida calcular con rapidez.

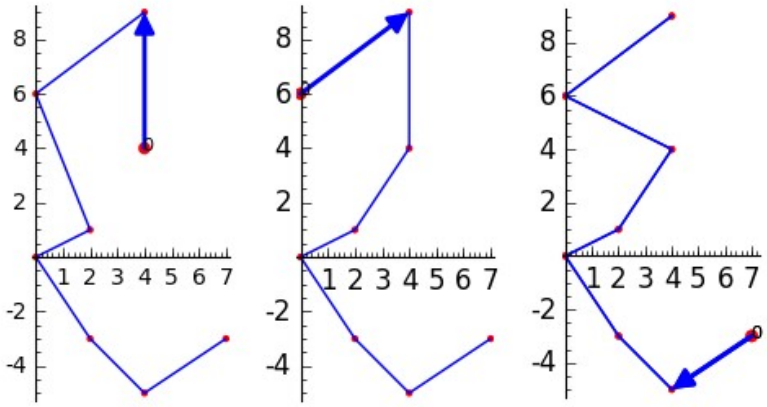
Unha primeira opción é suprimir de **C=cicloham(V)** a súa maior aresta mediante a función **CA(V)**, que nos dá unha árbore hamiltoniana de lonxitude $lon(C)[0] - lon(C)[1]$.

Con todo, como podemos comprobar coa lista V estudiada no exemplo 9, aínda que **cicloham(V)** é un ciclo hamiltoniano mínimo, **CA(V)** non é un camiño hamiltoniano mínimo en V:

Exemplo 25:

```
V=[0,6*I,2-3*I,4-5*I,7-3*I,4+4*I,2+I,4+9*I]
C=CA(V)
CH=caminoham(V)
OC=optimiza(V)
VC=vercamino(C)
VCH=vercamino(CH)
VOC=vercamino(OC[0])
show(graphics_array([VC,VCH,VOC]),figsize=[5,3])
print 'lonxitude de CA(V)=', lon(C)[0].n()
print 'lonxitude de caminoham(V)=', lon(CH)[0].n()
print 'lonxitude de optimiza(V)=', lon(OC[0])[0].n()
```

```
lonxitude de CA(V)= 27.6607624603085
lonxitude de caminoham(V)= 25.8811489286379
lonxitude de optimiza(V)= 25.3532848836375
```



A conexión intermedia obtívose a partir de **cicloham(V)** suprimindo as dúas arestas de maior lonxitude e conectando os dous camiños hamiltonianos que se producen de xeito óptimo.

Isto é o que establecemos como método xeral na función **caminoham(V)** que, aínda que, como pode verse non sempre devolve o camiño hamiltoniano óptimo en V, tomarémolo como unha aproximación suficiente para listas longas.

Exemplo 26:

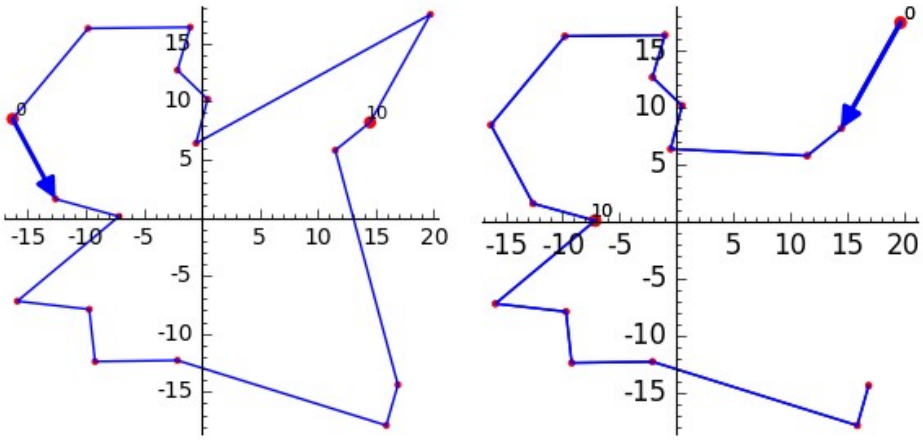
Comprobar que na lista

$$V = [-0.5 + 6.4i, 15.9 - 17.9i, 14.5 + 8.2i, -9.2 - 12.4i,$$
$$19.7 + 17.5i, -2.1 - 12.3i, -9.8 + 16.3i, -9.7 - 7.9i,$$
$$-7.14 + 0.1i, 11.5 + 5.8i, 0.5 + 10.2i, -1 + 16.4i, -2.1 + 12.7i,$$
$$-15.9 - 7.2i, 16.9 - 14.4i, -12.6 + 1.6i, -16.3 + 8.5i]$$

caminoham(V) non coincide con **cicloham(V)** sen a maior aresta.

```
V=[-0.5+ 6.4*I, 15.9-17.9*I, 14.5+ 8.2*I,-9.2-12.4*I,
19.7+17.5*I, -2.1-12.3*I,-9.8+16.3*I,-9.7-7.9*I,-7.14+ 0.1*I,
11.5+ 5.8*I, 0.5+ 10.2*I, -1+ 16.4*I, -2.1+ 12.7*I,-15.9 -
7.2*I, 16.9- 14.4*I, -12.6+ 1.6*I, -16.3+ 8.5*I]
```

```
C=cicloham(V)
VC=vercamino(C)
CH=caminoham(V)
VCH=vercamino(CH)
show(graphics_array([VC,VCH]),aspect_ratio=1,figsize=[6,3])
```



Co fin de avaliar a valía de **caminoham(**V**)** establecemos as seguintes cotas

1. **kruskal(**V**)**[1]
2. A función **cotaVa(**V**)**, que calcula para cada vértice de V a lonxitude das dúas arestas menores que inciden nel, sendo a cota o promedio da suma destas lonxitudes menos a lonxitude da maior das arestas consideradas.

Como non sempre **kruskal(**V**)**$[1]$ $\leq$**cotaVa(**V**)** nin o contrario, consideramos o máximo de ambas cotas implementado na función **cotaKVa(**V**)** e definimos a valía aberta, **bondada(**V**)**, como a razón

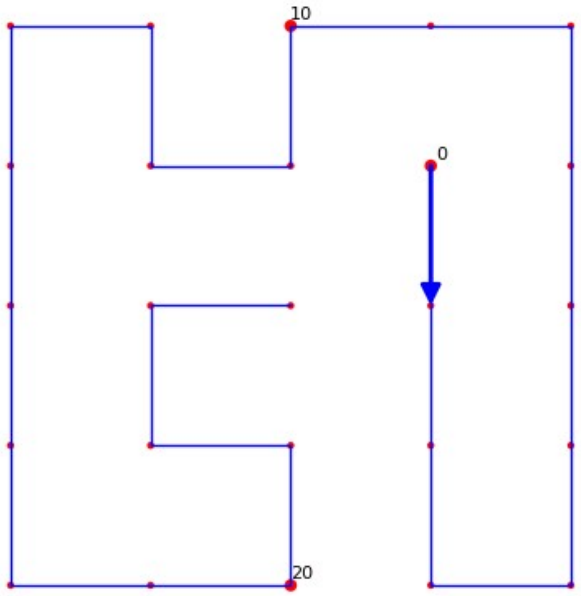
$$bondada(V) = 100 \cdot \frac{cotaKVa(V)}{lon(caminoham(V))[0]}$$

Exemplo 27:

Sexa $R =$**rectangular(**5,5,1,1**)**, comprobar que **caminoham(**R**)** é unha árbore hamiltoniana óptima.

```
R=rectangular(5,5,1,1)
CH=caminoham(R)
VCH=vercamino(CH)
```

```
show(VCH,figsize=[4,4])
```




```
print 'bondada(R)=', bondada(R)
```

```
bondada(R)= 100
```

Cando **bondada(**R**)**$= 100$, como sucede para $R =$**rectangular(**5,5,1,1**)**, podemos asegurar que **caminoham(**R**)** é un camiño hamiltoniano mínimo.

O recíproco non se verifica:

Exemplo 28:

Sexa a lista

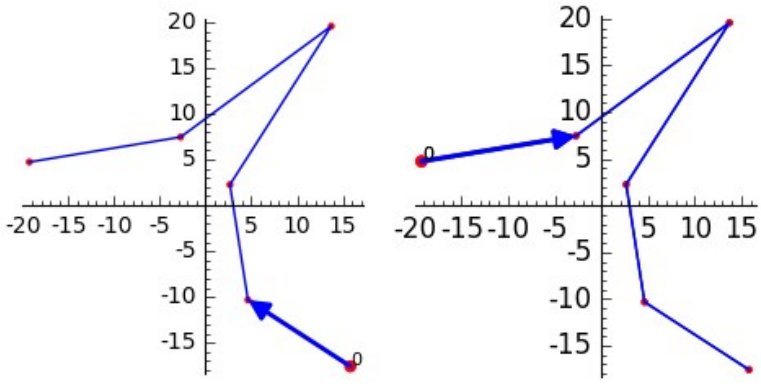
$$V = [13.74 + 19.57i, -19.28 + 4.73i, -2.73 + 7.46i,$$

$$4.63 - 10.33i, 15.83 - 17.58i, 2.66 + 2.26i]$$

```
V=[13.74 + 19.57*I, -19.28 + 4.73*I, -2.73 + 7.46*I, 4.63 -
10.33*I, 15.83 -17.58*I, 2.66 + 2.26*I]
C=caminoham(V)
O=optimiza(V)
L=vercamino(C)
LO=vercamino(O[0])
```

```
show(graphics_array([L,LO]), figsize=[5,5])
print 'bondada(V)=', bondada(V)
```

```
bondada(V)= 84.4217644500736
```



Vemos que **caminoham(V)** e **optimiza(V)** teñen a mesma lonxitude e **bondada(V)**=84,42%.

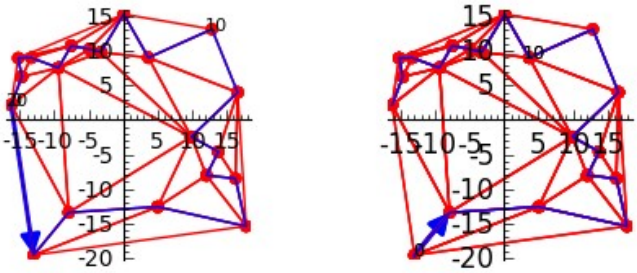
Daremos por boa a proposta de **caminoham(V)** se **bondada(V)**>80%.

Cando $|V| < 9$, **cicloham(V)** e **caminoham(V)** poden ser sustituídos, respectivamente, polos algoritmos exactos **optimizac(V)** e **optimiza(V)**, pero para $|V| \geq 9$ estes algoritmos exactos son demasiado lentos.

Exemplo 29:

O Teorema Kruskal - Delone fai que nos preguntemos se para calquera conxunto de Delone V, os grafos MHC(V) e MHP(V) son subgrafos do grafo de Delone (V, A), En moitos casos, é certo:

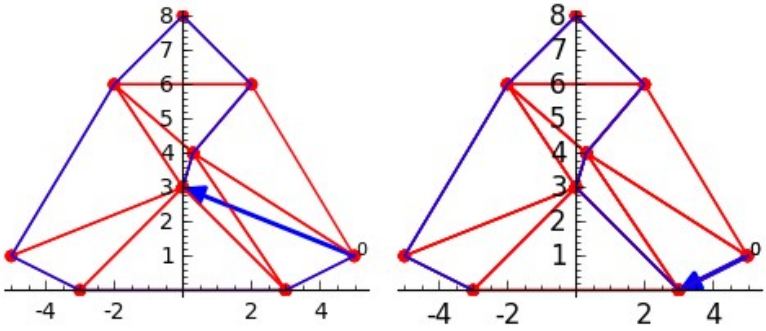
```
V=listacomplejos(20,20)
A=TD(V)[0]
C=cicloham(V)
HC=vercamino(C)
P=caminoham(V)
HP=vercamino(P)
show(graphics_array([A+HC,A+HP]),figsize=[5,2])
```



Exemplo 30:

Kantabtra e Dillencourt (véxase [10], [14]) atoparon un contraexemplo para a primeira afirmación que cumpre a segunda:

```
V=[3,-3, 5+I,-5+I,3*I,.3+4*I,8*I,-2+6*I,2+6*I]
A=TD(V)[0]
HC=optimizac(V)
C=vercamino(HC[0])
HP=optimiza(V)
P=vercamino(HP[0])
show(graphics_array([A+C,A+P]),figsize=[5,3])
```



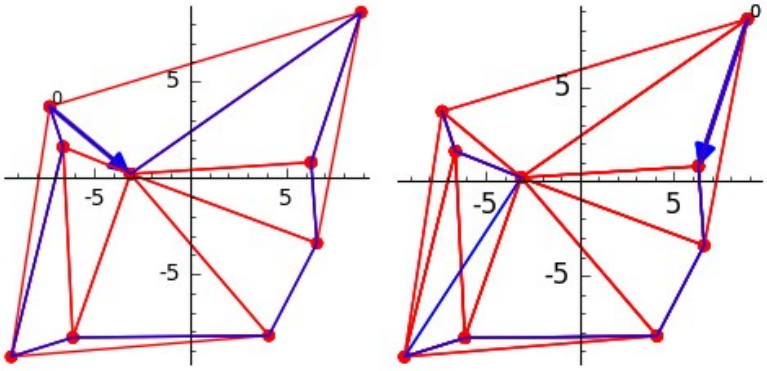
Exemplo 31:

$$V = [6.3 + 0.8i, -3.1 + 0.2i, -6.6 + 1.6i, -6.1 - 8.3i,$$

$$4.1 - 8.2i, -9.3 - 9.3i, 8.9 + 8.6i, 6.6 - 3.4i, -7.3 + 3.7i]$$

é un conxunto de Delone tal que o $MHC(V)$ é un subgrafo do grafo de Delone (V, A) e o $MHP(V)$ non o é.

```
V=[6.3 + 0.8*I, -3.1 + 0.2*I, -6.6 + 1.6*I, -6.1 - 8.3*I, 4.1 -
8.2*I, -9.3 - 9.3*I, 8.9 + 8.6*I, 6.6 - 3.4*I, -7.3 + 3.7*I]
A=TD(V)[0]
HC=optimizac(V)
C=vercamino(HC[0])
HP=optimiza(V)
P=vercamino(HP[0])
show(graphics_array([A+C,A+P]),figsize=[5,3])
```



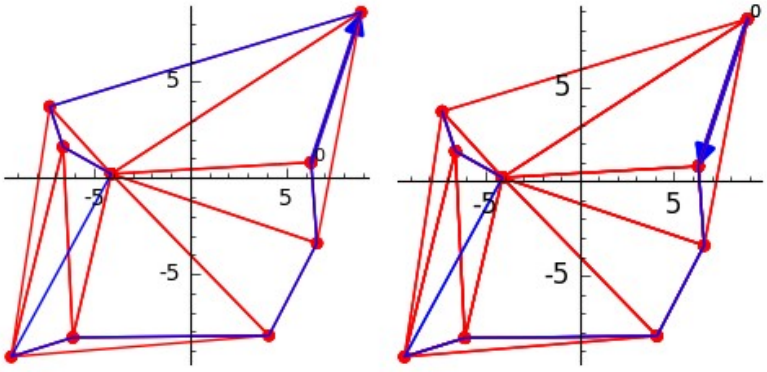
Exemplo 32:

$$V = [6.3 + 0.8i, -4.1 + 0.2i, -6.6 + 1.6i, -6.1 - 8.3i,$$

$$4.1 - 8.2i, -9.3 - 9.3i, 8.9 + 8.6i, 6.6 - 3.4i, -7.3 + 3.7i]$$

é un conxunto de Delone tal que nin o MHC(V) nin o MHP(V) son subgrafos do grafo de Delone (V, A).

```
V=[6.3 + 0.8*I, -4.1 + 0.2*I, -6.6 + 1.6*I, -6.1 - 8.3*I, 4.1 -
8.2*I, -9.3 - 9.3*I, 8.9 + 8.6*I, 6.6 - 3.4*I, -7.3 + 3.7*I]
A=TD(V)[0]
HC=optimizac(V)
C=vercamino(HC[0])
HP=optimiza(V)
P=vercamino(HP[0])
show(graphics_array([A+C,A+P]),figsize=[5,3])
```



2.2.7. Camiño hamiltoniano mínimo con extremos fixos

Tratamos de atopar nun conxunto $V \cup \{s, m\}$ un camiño hamiltoniano mínimo T que cumpra

$$g(s) = g(m) = 1 \quad \text{e} \quad g(v) = 2 \quad \forall v \in V$$

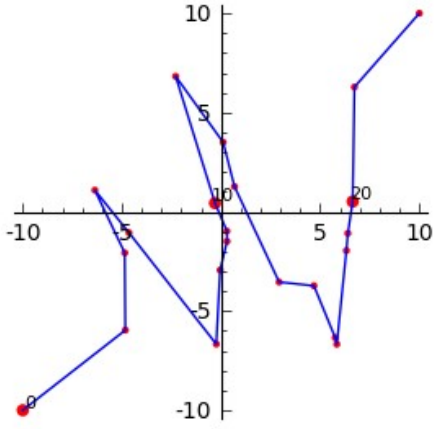
Exemplo 33:

Organizar o transporte escolar do colexio Hamilton situado en $10 + 10i$, con 20 puntos de recolleita en $[-7, 7] \times [-7, 7]$, mediante un autobús que vai a unha velocidade media de 50 $\frac{km}{h}$ e cuxo garaxe está en $-10 - 10i$.

a. Cal é a lonxitude da liña?
b. Cantos minutos ao día están no autobús os nenos de cada parada?

```
V=listacomplejos(7,20)
g=-10-10*I
c=10+10*I
E=[g,c]
C=cicloham(V+E)
m=1+C.index(c)
P=pegalistas(C[:m],C[m:])
M=mejora(P,6)
show(vercamino(M),axes=True,figsize=[3,3])
print 'a.', lon(M)[0],'kms'
```

```
a. 72.3934220645919 kms
```




```
print 'b.', 'Os nenos da parada'
for i in [1,2..20]:
    print i,',',ceil((2*lon(M[i:])[0]/50)*60), 'min'
```

```
b. Os nenos da parada
1 , 174 min
2 , 158 min
3 , 149 min
4 , 141 min
5 , 134 min
6 , 117 min
7 , 108 min
8 , 105 min
9 , 103 min
10 , 100 min
11 , 84 min
12 , 74 min
13 , 68 min
14 , 55 min
15 , 51 min
16 , 44 min
17 , 44 min
18 , 32 min
19 , 30 min
20 , 26 min
```

2.2.8. Teoría de grafos na esfera terrestre

Podemos proxectar os puntos dunha zona da esfera terrestre no plano tanxente no seu punto promedio e deste xeito aproximar a teoría de grafos na esfera pola teoría de grafos no plano.

Sexa P unha lista de n puntos da esfera terrestre cós seus topónimos e súas coordenadas xeográficas:

$$P = [[Name_k, lon_k, lat_k]\ for\ k\ in\ range(n)]$$

Tomando o radio da Terra $R = 6367650$ e facendo

$$LOM = \frac{\sum lon_k}{n} \quad LAM = \frac{\sum lat_k}{n}$$

$$\begin{cases} x_k = Rcos(lat_k \frac{\pi}{180}) tan((lon_k - LOM)\frac{\pi}{180}) \\ y_k = R(lat_k - LAM)\frac{\pi}{180} \end{cases}$$

transformamos a lista P nunha lista de puntos con nome, do plano complexo centrado en C=(LOM,LAM):

$$P = [[Name_k, x_k + y_k * I]\ for\ k\ in\ range(n)].$$

Exemplo 34:

Os nomes dos sete outeiros da Roma antiga son: Aventino, Celio, Palatino, Capitolio, Quirinal, Viminal, e Esquilino.

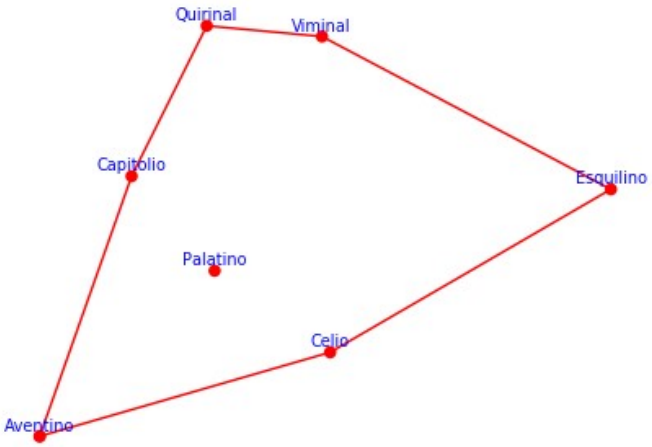
a. Facer un mapa con eses topónimos da cidade.

b. Estimar a súa extensión.

c. Dar unha cota inferior da lonxitude da muralla Serviana que a rodeaba.

Incluímos nos DATA a función **Roma()** cós sete outerios.

```
ROMA=[Roma][0]()[0]
COL=[a[1] for a in ROMA]
MS=envolturaconvexa(COL)
PLANO=poligonal(MS)+plse(COL)
TOP=[]
for a in ROMA:
    TOP.append(text(a[0],ri(a[1]+50*I),fontsize=7))
show(PLANO+sum(TOP),figsize=[4,4])
```




```
T=[]
for k in [1..len(MS)-2]:
    T.append([MS[0],MS[k],MS[k+1]])
AREA=sum([heron(t) for t in T])
print 'Área=',AREA
print
print 'Lonxitude=',lon(MS)[0]
```

```
Área= 2.56205978814832e6

Lonxitude= 7034.19168546871
```

Exemplo 35:

Facer un mapa toponímico do camiño francés de Santiago con inicio en Roncesvalles.

Solución:

Creamos e incorporamos nos DATA a función **caminofrances()** cos 29 principais núcleos de poboación entre Roncesvalles e Santiago de Compostela.

```
CF=[caminofrances][0]()[0]
PVH=[a[1] for a in CF]
K=kruskal(PVH)
TOP=[]
for a in CF:
    TOP.append(text(a[0],ri(a[1]+10000*I),fontsize=5))
show(K[0]+sum(TOP),figsize=[5,12])
```

Incluímos nos DATA unha función por cada rexión R de España. Se n son as centenas de milleiro de habitantes de R, expresamos R como a lista das súas n cidades máis importantes coas súas coordenadas xeográficas e transformámolas en listas de complexos con nome.

Tamén incluímos nos DATA unha función para cada país P da Unión Europea, cunha cidade por cada millón de habitantes. Se n son os millóns de habitantes do país P podemos expresar P como a lista das súas n cidades mais importantes, coas súas coordenadas xeográficas.

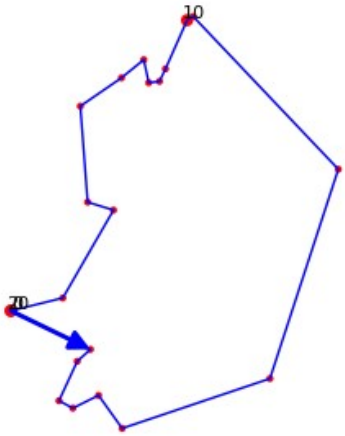
A función **viaje**(P) executa as funcións **genera_mapa()**, que pinta o ciclo hamiltoniano mínimo resaltando as cidades que ocupan os lugares múltiplos de dez e **pintar_nom_ciudades()**, que presenta as listas das cidades de cada tramo decenal para que o usuario poida establecer facilmente a correspondencia entre os nomes e o seu lugar no mapa evitando que se superpoñan os topónimos de lugares próximos complicando a súa lectura.

Exemplo 36:

Obter o ciclo hamiltoniano mínimo das 20 cidades principais de Galicia.

```
G=[galicia]
show(viaje(G)[1],figsize=[3,3])
```

```
0 [Santa_Eugenia, Pontevedra, Marin, Cangas, Vigo, Redondela,

10 [Ferrol, Oleiros, Cambre, Culleredo, Corunha, Arteixo, Carb

20 [Santa_Eugenia]
```

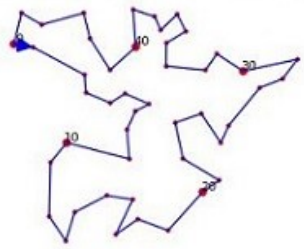


Exemplo 37:

Obter o ciclo hamiltoniano mínimo das capitais de provincia da España peninsular.

```
P=[espanha]
show(viaje(P)[1], figsize=[3,3])
```

```
0 [Pontevedra, Orense, Zamora, Salamanca, Avila, Segovia, Guadalajara, MADRID, Toledo, Ciudad_Real
10 [Caceres, Badajoz, Huelva, Cadiz, Sevilla, Cordoba, Jaen, Malaga, Granada, Almeria]
20 [Murcia, Alicante, Albacete, Cuenca, Teruel, Valencia, Castellon, Tarragona, Barcelona, Gerona]
30 [Lerida, Huesca, Zaragoza, Soria, Logronho, Pamplona, San_Sebastian, Vitoria, Bilbao, Santander
40 [Burgos, Valladolid, Leon, Oviedo, Lugo, Corunha, Pontevedra]
```



Podemos poñer sobre o mesmo plano complexo as cidades de dous países $[P_0, P_1]$ sumando a cada complexo de P_1 o complexo cuxo módulo é a distancia xeodésica entre os centros C_0 e C_1 de ambos países e cuxo

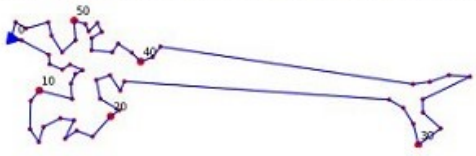
argumento é o ángulo que forma o círculo máximo determinado por C_0 e C_1 co paralelo que pasa por C_0. Isto faino a función **desplazamiento**($P_0()[1], P_1()[1]$)

Exemplo 38:

Obter o ciclo hamiltoniano mínimo das principais cidades de España e Grecia.

```
EG=[espanha,grecia]
show(viaje(EG)[1],figsize=[5,3])
```

```
0 [Pontevedra, Ourense, Zamora, Salamanca, Avila, Segovia, Guadalajara, MADRID, Toledo, Ciudad_Real]
10 [Caceres, Badajoz, Huelva, Cadiz, Sevilla, Cordoba, Jaen, Malaga, Granada, Almeria]
20 [Murcia, Alicante, Albacete, Cuenca, Teruel, Valencia, Castellon, Kerkira, Arta, Patra]
30 [Kalamata, ATHINA, Lamia, Larisa, Alexandropouli, Kavala, Thessaloniki, Ptolemaida, Gerona, Barcelona]
40 [Tarragona, Lerida, Huesca, Zaragoza, Soria, Logronho, Pamplona, San_Sebastian, Vitoria, Bilbao]
50 [Santander, Burgos, Valladolid, Leon, Oviedo, Lugo, Corunha, Pontevedra]
```



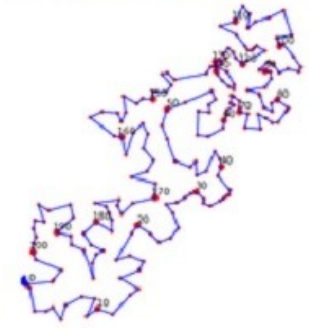
Para un conxunto de países $[P_0, P_1, \cdots, P_m]$ a función **viaje**($[P_0, P_1, \cdots, P_m]$) executa sucesivamente o **desplazamiento**(P_i() [1],P_{i+1}()[1]) para $i = 0, 1, \cdots, m-1$ e logra unha lista única de todas as cidades de devanditos países.

Exemplo 39:

Obter o ciclo hamiltoniano mínimo das principais cidades de Portugal, España, Francia e Alemaña.

```
PEFA=[portugal,espanha,francia,alemania]
show(viaje(PEFA)[1],figsize=[4,4])
```

```
0 [LISBOA, Setubal, Portimao, Tavira, Huelva, Cadiz, Sevilla, Cordoba, Jaen, Malaga]
10 [Granada, Almeria, Murcia, Alicante, Albacete, Valencia, Castellon, Teruel, Cuenca, Zaragoza]
20 [Huesca, Lerida, Tarragona, Barcelona, Gerona, Perpignan, Carcassonne, Beziers, Montpellier, Nimes]
30 [Avignon, Aix_Provence, Marseille, Toulon, Frejus, Cannes, Nice, Menton, Tende, Briancon]
40 [Val_dIsere, Chamonix, Thonon_les_Bains, Grenoble, Valence, Lyon, S_Etienne, Cl_Ferrand, Bourges, Orleans]
50 [PARIS, Reims, Trier, Saarbrucken, Metz, Nancy, Bar_le_Duc, Dijon, Besancon, Mulhouse]
60 [Friburgo, Strasbourg, Haguenau, Pforzheim, Karlsruhe, Ludwigshafen, Mannheim, Heidelberg, Heilbronn, Stuttgart]
70 [Reutlingen, Ulm, Lindau, Garmisch_Partenkirchen, Augsburg, Munchen, Rosenheim, Traunstein, Passau, Hauzenberg]
80 [Cham, Ratisbona, Landshut, Ingolstadt, Nuremberg, Erlangen, Wurzburg, Bayreuth, Plauen, Gera]
90 [Jena, Erfurt, Halle, Leipzig, Chemnitz, Dresden, Zittau, Goerlitz, Cottbus, BERLIN]
100 [Potsdam, Neuruppin, Schwedt_Oder, Sassnitz, Rostock, Lubeck, Harburg, Hamburg, Kiel, Flensburg]
110 [Cuxhaven, Norden, Bremen, Hanover, Hildesheim, Salzgitter, Brunswick, Braunschweig, Wolfsburg, Magdeburgo]
120 [Gotingen, Kassel, Paderborn, Bielefeld, Osnabruck, Munster, Hamm, Dortmund, Hagen, Recklinghausen]
130 [Essen, Oberhausen, Duisburgo, Mulheim, Solingen, Siegen, Frankfurt, Darmstadt, Mainz, Koblenz]
140 [Bonn, Koln, Dusseldorf, Monchengladbach, Aachen, Lille, Dunkerque, Calais, Arras, Amiens]
150 [Dappa, Rouen, Le_Havre, Caen, Cherbourg, Avranches, Rannes, Saint_Malo, Lanion, Brest]
160 [Saint_Nazaire, Nantes, La_Rochelle, Angers, Le_Mans, Tours, Poitiers, Limoges, Perigueux, Bordeaux]
170 [Cahors, Toulouse, Pau, Pamplona, San_Sebastian, Bayonne, Santander, Bilbao, Vitoria, Logronho]
180 [Soria, Burgos, Valladolid, Segovia, Guadalajara, MADRID, Toledo, Ciudad_Real, Avila, Salamanca]
190 [Zamora, Braganza, Leon, Oviedo, Lugo, Corunha, Orense, Pontevedra, Valenza, Braga]
200 [Porto, Aveiro, Coimbra, Caceres, Badajoz, Elvas, LISBOA]
```



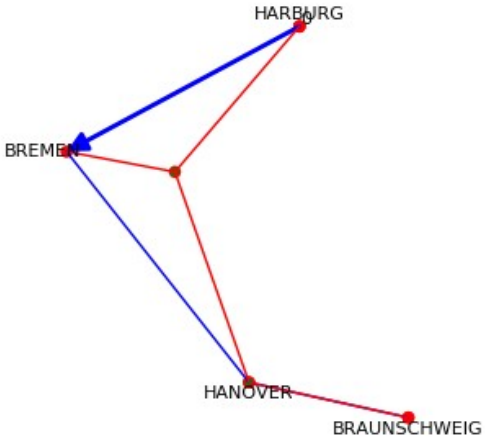
Exemplo 40:

En 1836, Gauss suscitou ao seu amigo Schuhmacher (véxase [11]) canto se podería reducir a conexión por estrada entre as cidades alemás de Bremen, Harburg, Hannover e Braunschweig.

```
P=[alemania][0]()[0]
NOM=[Harburg,Bremen,Braunschweig,Hanover]
Gauss=[a for a in P if a[0] in NOM]
CG=[a[1] for a in Gauss]
```

```
CAG=CA(CG)
Harburg=text('HARBURG',ri(CAG[0]+4000*i),color='black',fontsize=8)
Bremen=text('BREMEN',ri(CAG[1]-8000),color='black',fontsize=8)
Hanover=text('HANOVER',ri(CAG[2]-
4000*i),color='black',fontsize=8)
Braunschweig=text('BRAUNSCHWEIG',ri(CAG[3]-
4000*i),color='black',fontsize=8)
show(vercamino(CAG)+Steiner4(CAG)
[1]+Harburg+Bremen+Hanover+Braunschweig, figsize=[3,3])
print 'S(L)=', Steiner4(CAG)[2]
NC=[]
for a in CAG:
    for b in NOM:
        if [b,a] in Gauss:
            NC.append(b)
print 'Orde de recorrido das cidades no camiño hamiltoniano', NC
print 'G(L)=', Steiner4(CAG)[3]
print 'Puntos de Steiner=', Steiner4(CAG)[0]
```

```
S(L)= 233128.223185305
Orde de recorrido das cidades no camiño hamiltoniano [Harburg,
Bremen, Hanover, Braunschweig]
G(L)= 4.67606206040729
Puntos de Steiner= [-43224.4606465724 + 237738.673049200*I,
-18104.9108544127 + 166431.131056219*I]
```




```
z=29407.3496365613 + 265629.192560511*I
coorgeo(z,alemania)
```

```
(53.2684705357259, 10.4412866320434)
```

Se z é o complexo do punto de Steiner, a función **coorgeo(**z,alemania**)** devólvenos as súas coordenadas xeográficas e pegando en HTML, por exemplo, a dirección:

<iframe width="100\%" height=850tx frameborder="0" scrolling="no" marginheight="0" marginwidth="0" src="http://www.mundivideo.com/co.htm?id=sp\&lat=40.7060081\&lng=-74.0088151\&x_convertirUTM=583730.1\&y_convertirUTM=4506594.37\&zon=18\&hem=Norte$\&dir=Wall Street, Nueva York"></iframe>

introducindo en "Lugar" a saída de **coorgeo(**z,alemania**)** e clicando en "Ver", aparece un sinalizador do punto de ditas coordenadas. Pasando a modo Mapa, podemos coñecer as cidades da contorna de devandito punto.


```
z=29407.3496365613 + 265629.192560511*I
coorgeo(z,alemania)
```

```
(53.2684705357259, 10.4412866320434)
```

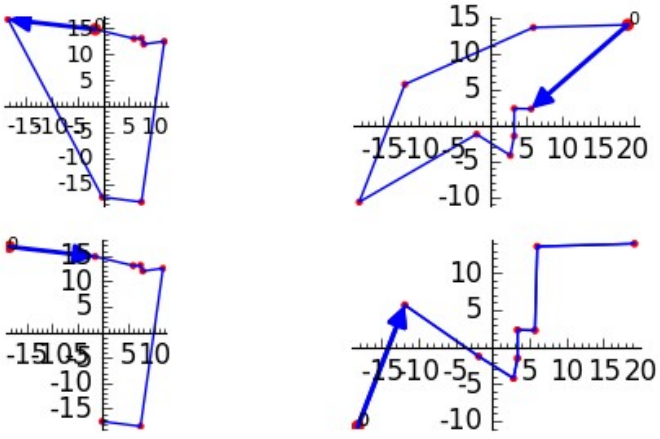
2.3. EXERCICIOS

Exercicio 1:

Estimar os tempos de cálculo para os algoritmos exactos **optimizac(**V**)** e **optimiza(**V**)** para $|V| = 8$ e $|V| = 9$

```
V8=listacomplejos(20,8)
V9=listacomplejos(20,9)
T0=walltime();OC8=optimizac(V8)
T1=walltime(T0)
print 'Tempo de cálculo de optimizac para 8 puntos'
print T1,'seg.'
T0=walltime();OC9=optimizac(V9)
T1=walltime(T0)
print 'Tempo de cálculo de optimizac para 9 puntos'
print T1,'seg.'
T0=walltime();O8=optimiza(V8)
T1=walltime(T0)
print 'Tempo de cálculo de optimiza para 8 puntos'
print T1,'seg.'
T0=walltime();O9=optimiza(V9)
T1=walltime(T0)
print 'Tempo de cálculo de optimiza para 9 puntos'
print T1,'seg.'
show(graphics_array([[vercamino(OC8[0]),vercamino(OC9[0])],
[vercamino(O8[0]),vercamino(O9[0])]]),figsize=[6,3])
```

```
Tempo de cálculo de optimizac para 8 puntos
4.29503393173 seg.
Tempo de cálculo de optimizac para 9 puntos
40.9143989086 seg.
Tempo de cálculo de optimiza para 8 puntos
14.8195409775 seg.
Tempo de cálculo de optimiza para 9 puntos
125.224471092 seg.
```

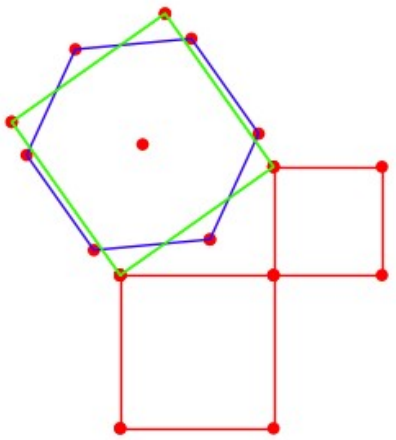


Exercicio 2:

Construír un cadrado que teña a mesma área que un hexágono regular de lado 1.

```
H=cuadrarec(1,sqrt(3)/4)[0]
L=Pitagoras(H,H)[0]
P=Pitagoras(2*H,L)
show(P[0])
C=sum(P[2][0:4])/4
P1=C+(P[2][-1]-P[2][-2])/abs(P[2][-1]-P[2][-2])
L1=[C+(P1-C)*exp(k*I*pi/3) for k in range(7)]
E=poligonal(L1,.7)
show(P[1]+plse([C])+E+P[3],figsize=[3,3])
```

1.61185489773531



Exercicio 3:

Nun conxunto aleatorio V de 20 puntos do plano complexo achar:

a. A envoltura convexa, o seu perímetro e a área encerrada.
b. Teselación de Delone e zonas de proximidade de V.
c. Os niveis de convexidade do conxunto V e as áreas comprendidas entre os niveis consecutivos.

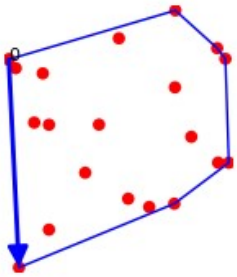
```
print 'a.'
V=listacomplejos(20,20);C=cebolla(V)
```

```
EC=vercamino(C[0][0]);show(plce(V)+EC,figsize=[2,2])
print 'perímetro da envoltura convexa =', lon(C[0][0])[0]
T=[]
for k in [1..len(C[0][0])-2]:
    T.append([C[0][0][0],C[0][0][k],C[0][0][k+1]])
AREA=sum([heron(t) for t in T])
print 'área da envoltura convexa =', AREA
```

a.

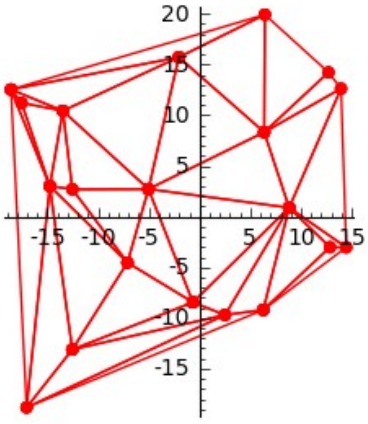
perímetro da envoltura convexa = 119.152080562540

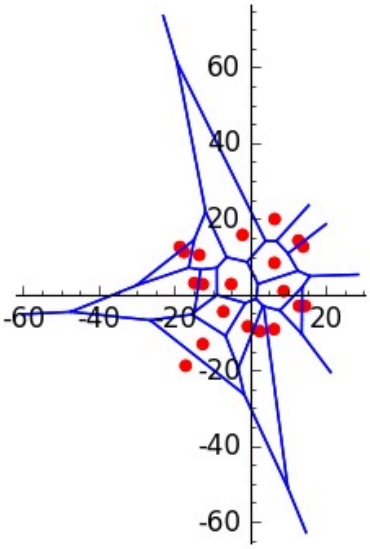
área da envoltura convexa = 914.333573363297



```
print 'b.'
T=TD(V)
VR=voronoi(V)
show(graphics_array([T[0],VR]),figsize=[5,4])
```

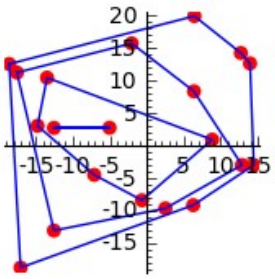
b.





```
print 'c.'
print 'V ten', len(C[0]), 'niveis de convexidade'
show(C[1],figsize=[2,2])
```

```
c.
V ten 4 niveis de convexidade
```



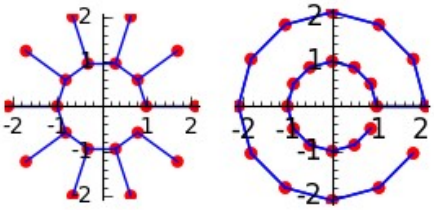

```
LA=[]
for k in range(len(C[0])):
    CN=C[0][k]
    if len(CN)<4:
        AN=0
        LA.append(AN)
    else:
        TN=[]
        for m in [1..len(CN)-2]:
            TN.append([CN[0],CN[m],CN[m+1]])
        AN=sum([heron(t) for t in TN])
        LA.append(AN)
print 'As áreas comprendidas entre os niveis consecutivos son'
ANC=[LA[k]-LA[k+1] for k in range(len(LA)-1)]+[LA[-1]]
ANC
```

```
As áreas comprendidas entre os niveis consecutivos son
[344.758938979077, 351.068594058921, 218.506040325299, 0]
```

Exercicio 4:

Determinar os MST para as listas de puntos V_1 =**radial(**6,2,1,2**)** e V_2 =**radial(**8,2,1,2**)**.

```
V1=radial(10,2,1,1.1);KV1=kruskal(V1)[0]
V2=radial(12,2,1,1.1);KV2=kruskal(V2)[0]
show(graphics_array([KV1,KV2]),figsize=[3,2])
```



Podes explicar as diferenzas entre estas dúas conexións?

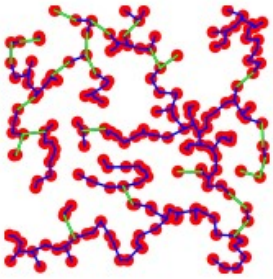
Exercicio 5:

Si L é unha lista aleatoria de 200 árbores a regar por goteo nun xardín cadrado de 0.16 ha, pintar de verde os tramos de mangueira do MST(L) que midan entre 3 e 4 metros.

Solución:

$E =$**kruskal(**L**)**[3] son as arestas do MST(L) de modo que cada $e \in E$ é a lista da súa lonxitude e os índices dos seus extremos en L.

```
L=listacomplejos(20,200);K=kruskal(L)
E=[e for e in K[3] if 3<=e[0]<=4]
V=[]
for e in E:
    V.append(line((ri(L[e[1]]),ri(L[e[2]])),hue=.3,thickness=1))
show(K[0]+add(V),figsize=[2,2])
```



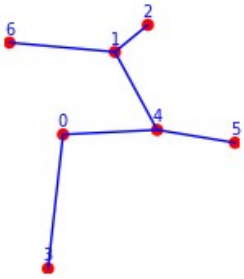
Exercicio 6:

Xera aleatoriamente unha lista V de 7 complexos e intenta achar outra R tal que

$$MST(V \cup R) \leq \frac{96}{100} MST(V)$$

Cando o logres, pinta V en vermello e R en verde.

```
V=listacomplejos(20,7);KV=kruskal(V)
VO=[[k,V[k]] for k in range(len(V))]
TOP=[]
for a in VO:
    TOP.append(text(a[0],ri(a[1]+1.5*I),fontsize=7))
show(KV[0]+sum(TOP), figsize=[2,2])
```

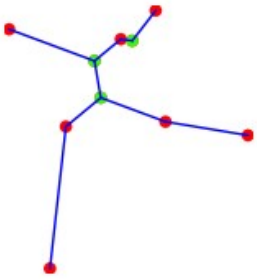



```
print 'cota 96% lonxitude inicial', 96*KV[1]/100
```

```
cota 96% lonxitude inicial 57.0947326166177
```

```
VI=[[6,1,0,4],[1,2,4,5]]
kop=visualsteiner(V,VI)
VA=kop[4]
PS=restalistas(VA,V)
PPS=plse(PS,h=.3)
print 'a lonxitude reducida é', kop[1]
show(kop[0]+PPS,figsize=[2,2])
```

```
a lonxitude reducida é 56.9786977750426
```

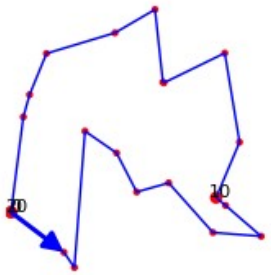


Exercicio 7:

Obter unha lista aleatoria L de 20 puntos do plano complexo. Determinar o ciclo hamiltoniano óptimo indicando a súa valía.

```
L=listacomplejos(20,20)
CL=cicloham(L)
PCL=vercamino(CL)
show(PCL,figsize=[2,2])
print 'valía =',bondad(CL)
```

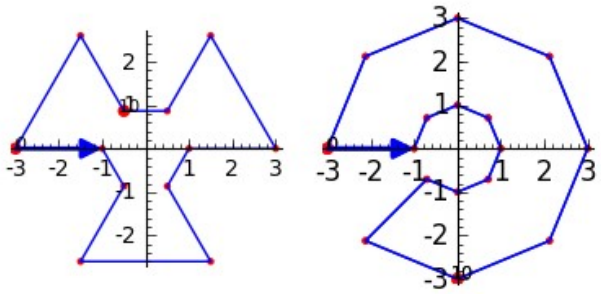
```
valía = 98.9109345500573
```



Exercicio 8:

Determinar os ciclos hamiltonianos óptimos das listas de puntos V_1 =**radial(**6, 2, 1, 2**)** e V_2 =**radial(**8, 2, 1, 2**)**.

```
V1=radial(6,2,1,2);CV1=cicloham(V1)
PCV1=vercamino(CV1)
V2=radial(8,2,1,2);CV2=cicloham(V2)
PCV2=vercamino(CV2)
show(graphics_array([PCV1,PCV2]),figsize=[4,3])
```



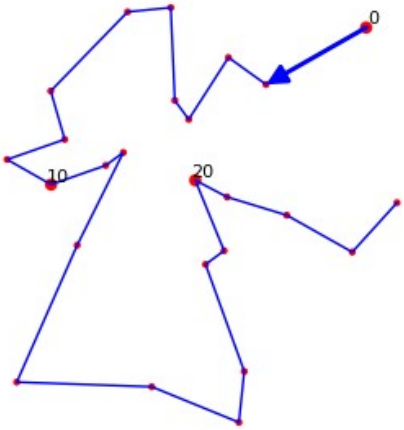
Podes explicar as diferenzas entre estes dous ciclos hamiltonianos? (véxase [20])

Exercicio 9:

a. Obter unha lista aleatoria L de 25 puntos do plano complexo. Determinar o camiño hamiltoniano óptimo indicando a súa lonxitude e a súa bondade.
b. Na lista anterior, calcular o camiño hamiltoniano mínimo con extremos $L[10]$ e $L[20]$ indicando a súa lonxitude.

```
print 'a.'
L=listacomplejos(20,25)
CL=caminoham(L)
PCL=vercamino(CL)
show(PCL,figsize=[3,3])
print 'lonxitude do camiño é',lon(CL)[0]
print 'bondade do camiño é',bondada(CL)
```

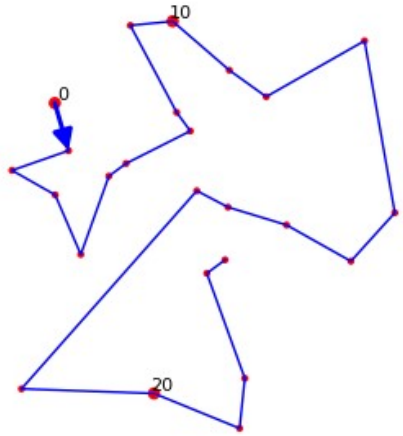
```
a.
lonxitude do camiño é 154.963621513148
bondade do camiño é 86.7083445664479
```




```
print 'b.'
P=CL.index(L[10])
F=CL.index(L[20])
if P<F:
    if P==len(L)-1:
        P=-1
    PG=pegalistas(reverso(CL[:P+1])+reverso(CL[F:]),CL[P+1:F])
else:
```

```
    if F==len(L)-1:
        F=-1
    PG=pegalistas(reverso(CL[:F+1])+reverso(CL[P:]),CL[F+1:P])
M=mejora(PG,6)
print 'lonxitude do camiño é',lon(M)[0]
show(vercamino(M),axes=False,figsize=[3,3])
```

```
b.
lonxitude do camiño é 171.419752747844
```



Exercicio 10:

Vigo, igual que Roma, ten sete montes senlleiros: Guía, Castro, Alba, Galiñeiro, Beade, Fragoselo e Madroa.

a. Facer un mapa con eses topónimos da zona metropolitana.

b. Estimar a súa extensión.

c. Calcular os vértices do maior triángulo de Delone da zona.

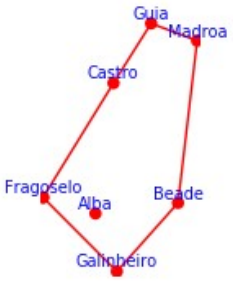
Solución:

Creamos e incorporamos nos DATA a función **Vigo()** cos sete montes que a circundan.

```
print 'a.'
VIGO=[Vigo][0]()[0]
COL=[a[1] for a in VIGO]
V=copy(COL)
MS=envolturaconvexa(COL)
```

```
PLANO=poligonal(MS)+plse(COL)
TOP=[]
for a in VIGO:
    TOP.append(text(a[0],ri(a[1]+500*I),fontsize=7))
show(PLANO+sum(TOP),figsize=[2,2])
```

a.




```
print 'b.'
T=[]
for k in [1..len(MS)-2]:
    T.append([MS[0],MS[k],MS[k+1]])
AREA=sum([heron(t) for t in T])
print 'Área =',AREA
```

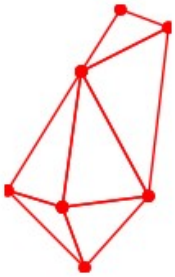
b.

Área = 4.96626689856092e7

```
print 'c.'
D=delone(V);show(TD(V)[0], figsize=[2,2])
T=[]
for d in D:
    T.append([V[d[1]],V[d[2]],V[d[3]]])
M=maximos([heron(a) for a in T])
[a[0] for a in VIGO if a[1] in T[M[1][0]]]
```

c.

[Castro, Beade, Madroa]



Exercicio 11:

O Instituto Xeográfico Nacional desexa realizar un estudo orográfico dunha parte de Castela a Mancha, para iso ten que:

a. Representar nun mapa os 16 núcleos máis poboados cos seus topónimos e calcular o perímetro do menor polígono convexo que os contén.
b. Facer unha triangulación da zona verificando que os círculos circunscritos a devanditos triángulos non conteñan no seu interior ningún núcleo de poboación.
c. Calcular os vértices do maior triángulo da zona.
d. Calcular as coordenadas xeográficas dos circuncentros de devanditos triángulos.

Solución:

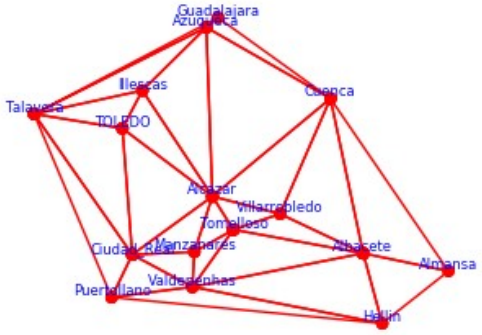
Incluímos nos DATA a función **casmancha()** que inclúe os 16 núcleos de poboación máis poboados.

```
print 'a.'
CM=[casmancha][0]()[0]
V=[a[1] for a in CM];W=copy(V)
C=envolturaconvexa(W)
MAPA=poligonal(C)+plse(V)
NOM=[]
for a in CM:
    NOM.append(text(a[0],ri(a[1]+5000*I),fontsize=6))

show(MAPA+sum(NOM),figsize=[3,3])
print 'perimetro =', lon(C)[0]
```

```
a.
perimetro = 858768.643382953
```

```
print 'b. A triángulación pedida é unha triangulación de Delone'
show(TD(V)[0]+sum(NOM), figsize=[3,3])
```

b. A triángulación pedida é unha triangulación de Delone




```
print 'c.'
D=delone(V)
T=[]
for d in D:
   T.append([V[d[1]],V[d[2]],V[d[3]]])
M=maximos([heron(a) for a in T])
[a[0] for a in CM if a[1] in T[M[1][0]]]
```

c.
[Cuenca, Azuqueca, Alcazar]

```
print 'd. A función Delone() dá a lista dos circuncentros e a
función coorgeo(z,casmancha) devolve as coordenadas xeográficas'
LC=[d[0] for d in D]
CLC=[]
for z in LC:
    CLC.append(coorgeo(z,casmancha))
print CLC
```

d. A función Delone() dá a lista dos circuncentros e a función coorgeo(z,casmancha) devolve as coordenadas xeográficas

[(39.5357458280120, -1.97626826488910), (39.6514309076420, -1.26668750615145), (38.5956971219600, -2.55524373855139), (38.6083772556114, -2.55220431789434), (38.5867298377192, -2.55305613887020), (38.8060165886169, -1.51527716558266), (43.1069315895503, -6.04862126635731), (39.3904595392036, -4.52856805918087), (40.1714420706784, -4.37867961651244), (39.1748148606811, -4.92188374345006), (41.2103229292184, -4.69787529860977), (40.2034498036571, -2.78682923173853), (39.4343183479618, -3.80352017220000), (39.7976092919411, -3.45060532239431), (38.7358432940024, -3.74992376568764),

```
(39.2927165662125, -3.66925123662060), (38.8934834184596,
-3.64292887040421), (39.9854748178531, -2.95057059410561),
(39.7962633289230, -2.74711138323806), (35.7264214793433,
-3.24941193684359), (39.3858367703529, -2.88645947588444),
(39.1942260668024, -3.27941817378471), (38.8681640627088,
-3.04134466066703), (39.9816346552990, -3.08896062137473)]
```

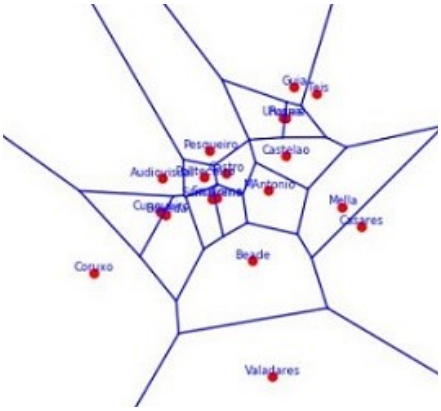
Exercicio 12:

Calcular as zonas de proximidade dos institutos de ensino secundario de Vigo.

Solución:

Incluímos nos DATA a función **iesvigo()** cos 19 institutos de ensino secundario de Vigo.

```
IES=[iesvigo][0]()[0];V=[a[1] for a in IES];VR=voronoi(V);NOM=[]
for a in IES:
    NOM.append(text(a[0],ri(a[1]+200*I),fontsize=6))
show(VR+sum(NOM),figsize=[8,8])
```



Exercicio 13:

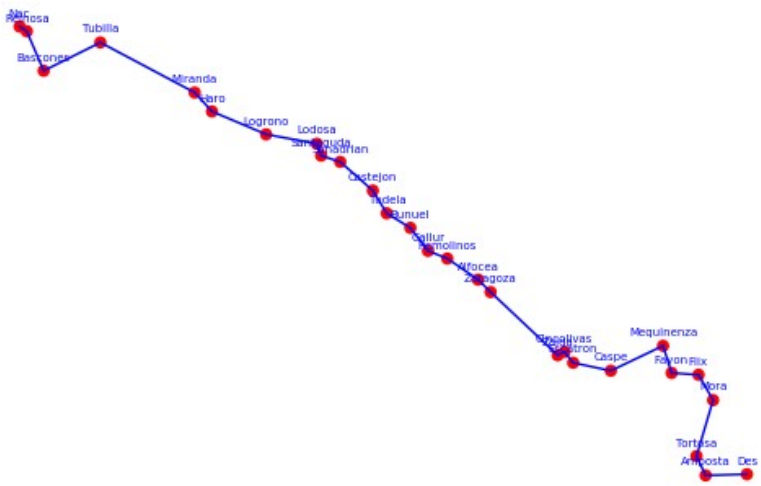
O Ebro nace preto de Fontibre e desemboca no Mediterráneo, preto de Amposta. Sabendo que a súa lonxitude é de 930 km, achar unha lista de localidades da súa ribeira cuxa poligonal alcance, polo menos, o 60% da lonxitude do río.

Solución:

Creamos e incorporamos nos DATA a función **ebro()** coas suficientes poboacións da ribeira.

```
L=ebro()
EB=[ebro][0]()[0]
LP=[a[1] for a in L[0]]
KLP=kruskal(LP)
show(lon(LP)[0])
TOP=[]
for a in EB:
    TOP.append(text(a[0],ri(a[1]+8000*I),fontsize=5))
show(KLP[0]+sum(TOP),figsize=[5,5])
```

578048.693846232



Exercicio 14:

Facer un mapa toponímico do camiño portugués de Santiago con inicio en Lisboa. Calcular a súa lonxitude total e indicar entre que vilas discorren as etapas máis longa e máis curta.

Solución:

Creamos e incorporamos nos DATA a función **caminoportugues()** cos 23 núcleos de poboación máis importantes intermedios.

```
CP=[caminoportugues][0]()[0]
```

```
N=[a[0] for a in CP]
L=[a[1] for a in CP]
K=kruskal(L)
K[0]
TOP=[]
for a in CP:
    TOP.append(text(a[0],ri(a[1]),fontsize=7))
K[0]+sum(TOP)
```


```
print 'Etapa máis longa',[N[10],N[11],29929]
print 'Etapa máis curta',[N[11],N[12],12215]
```

```
Etapa máis longa [Oporto, Sao_Pedro_de_Rates, 29929]
Etapa máis curta [Sao_Pedro_de_Rates, Barcelos, 12215]
```

Exercicio 15:

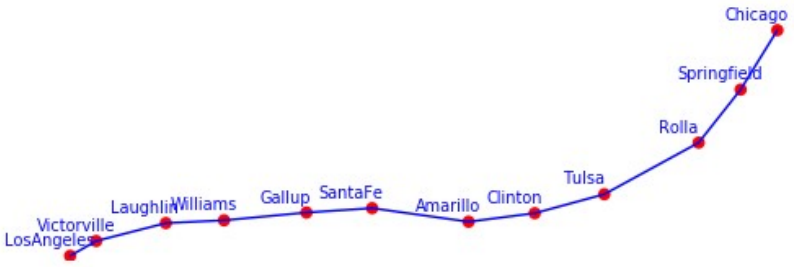
Facer un plano da ruta 66 cara ao Far West e calcular a súa lonxitude total. Calcular entre que cidades discorren as etapas máxima e mínima.

Solución:

Creamos e incorporamos nos DATA a función **ruta66()** que representa o camiño de Chicago aos Ángeles a través dos 10 núcleos de poboación intermedios máis importantes.

```
R66=[ruta66][0]()[0]
N=[a[0] for a in R66]
L=[a[1] for a in R66]
K=kruskal(L)
print' A lonxitude total es',round(K[1]/1000,2),'km'
K[0]
TOP=[]
for a in R66:
    TOP.append(text(a[0],ri(a[1]+58000*I-80000),fontsize=7))
show(K[0]+sum(TOP),figsize=[5,5])
```

```
A lonxitude total es 3059.02 km
```




```
print 'Etapa máis longa =',[N[2],N[3]]
print 'Etapa máis curta =',[N[9],N[11]]
```

```
Etapa máis longa = [Rolla, Tulsa]
Etapa máis curta = [Victorville, LosAngeles]
```

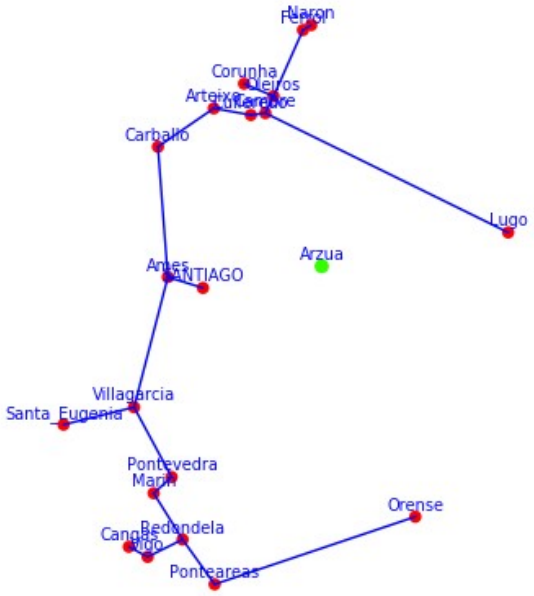
Exercicio 16:

A Xunta construirá o MST de fibra óptica para as 20 cidades máis poboadas de Galicia e permitirá ao resto de núcleos urbanos conectarse a el un por un e por conta propia. Presupostar a conexión de Arzúa se o metro de fibra vale 30 €, o enlace a un vértice, 1000 € e o enlace a unha aresta, 6000 €.

Solución:

Creamos e incorporamos nos DATA a función **Garzua()** que inclúe a Arzúa e as 20 cidades máis importantes de Galicia.

```
GAR=[Garzua][0]()[0]
AR=[a[1] for a in GAR if a[0]==Arzua]
L=[a[1] for a in GAR if a[0]!=Arzua]
K=kruskal(L)
TOP=[]
for a in GAR:
    TOP.append(text(a[0],ri(a[1]+3000*I),fontsize=7))
show(K[0]+sum(TOP)+point(ri(AR[0]),hue=.3,pointsize=40),figsize=
[4,4])
```




```
print 'A conexión ao vértice máis próximo vale',
round(min([abs(AR[0]-a) for a in L])*7+1000,2)
```

```
    A conexión ao vértice máis próximo vale 223443.35
```

```
print 'A aresta máis próxima a Arzúa é', [GAR[K[3][-1][1]]
[0],GAR[K[3][-1][2]][0]]
print 'e a conexión a ela vale', altura(AR[0],GAR[3][1],GAR[18]
[1])*7+10000
```

```
    A aresta máis próxima a Arzúa é [Lugo, Cambre]
    e a conexión a ela vale 217512.745923090
```

Exercicio 17:

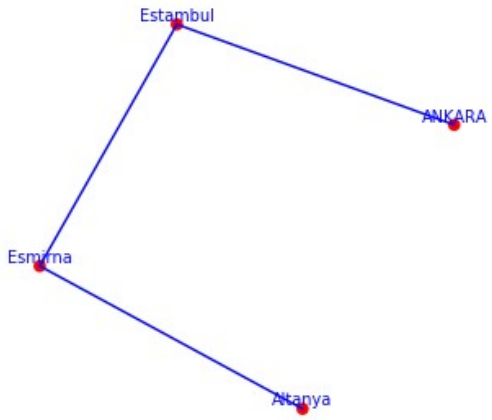
O goberno turco pensa construír unha conexión ferroviaria de alta velocidade entre Estambul, Ankara, Esmirna e Altanya. Se cada km se estíma en 15 millóns de euros, cal é o orzamento global?

Solución:

Incluímos nos DATA a función **turquia()** con esas catro cidades e as súas coordenadas xeográficas.

```
TUR=[turquia][0]()[0]
L=[a[1] for a in TUR]
K=kruskal(L)
TOP=[]
for a in TUR:
    TOP.append(text(a[0],ri(a[1]+10000*I),fontsize=7))
show(K[0]+sum(TOP), figsize=[3,3])
print 'A lonxitude do MST é', K[1]/1000, 'km'
```

```
A lonxitude do MST é 1036.55875195294 km
```

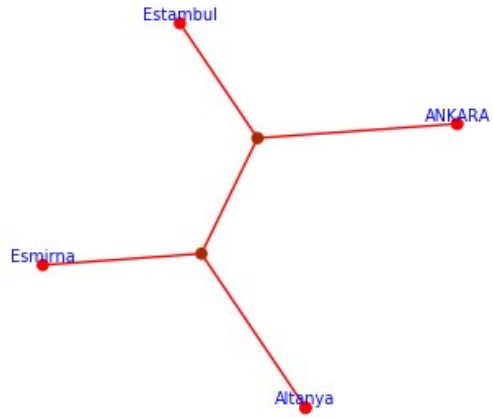


Podemos acurtar a lonxitude da conexión facendo un **Steiner4(**L**)**.

```
S=Steiner4(L)
show(S[1]+sum(TOP),figsize=[3,3])
print 'A lonxitude do SMT é', S[2]/1000, 'km'
print 'Presupostamos a conexión en', 15*S[2]/1000, 'millóns de
euros'
```

```
A lonxitude do SMT é 965.633298218769 km
```

```
Presupostamos a conexión en 14484.4994732815 millóns de euros
```



Esta conexión necesita dous nós ferroviarios que son os puntos de Steiner S[0] de coordenadas complexas.

```
print S[0][0]
print S[0][1]
```

```
    13019.9736700957 + 79495.7915470722*I
    -53728.8592297583 - 57867.1682242232*I
```

Aplicándolles a función **coorgeo(***z*,turquia**)** obtemos as súas coordenadas xeográficas.

```
CGT=[]
for z in S[0]:
    CGT.append(coorgeo(z,turquia))
print CGT[0]
print CGT[1]
```

```
    (39.7808554227592, 30.0760064587299)
    (38.5448707413900, 29.3054598737962)
```

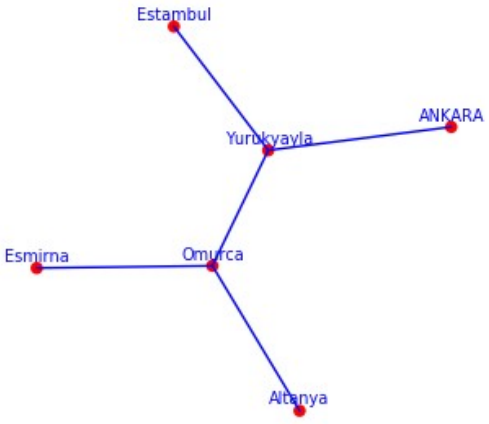
que están nas proximidades de Yürükyayla e Omurca. Podemos construír a lista nós.

```
def nudos():
    var('Yurukyayla Omurca')
    TUR=[[Yurukyayla,39.7808554227592, 30.0760064587299],
[Omurca,38.5448707413900,
29.3054598737962]]
```

```
    R=6367650
    LAM=sum([a[1] for a in TUR])/len(TUR)
    LOM=sum([a[2] for a in TUR])/len(TUR)
    PXY=[[a[0],(R*cos(a[1]*pi/180)*tan((a[2]-LOM)*pi
/180)).n()+I*(R*(a[1]-LAM)*pi/180).n()] for a in TUR]
    return PXY,[LOM,LAM]
```

engadila a **turquia()** e mediante **kruskal()** obter o MST da suma

```
NT=[turquia][0]()[0]+[nudos][0]()[0]
L=[a[1] for a in NT]
K=kruskal(L)
TOP=[]
for a in NT:
    TOP.append(text(a[0],ri(a[1]+15000*I),fontsize=7))
show(K[0]+sum(TOP), figsize=[3,3])
```



No orzamento habería que engadir o prezo das estacións dos dous nós.

Exercicio 18:

No hospital Vall d'Hebron de Barcelona, estudan a viabilidade de recoller as células nai que diariamente se doan nos hospitais das outras 15 cidades máis poboadas de Cataluña mediante un dron.

Se se desprazase a 130 km/hora e en cada manobra de despegue e aterraxe invertese 10 minutos:

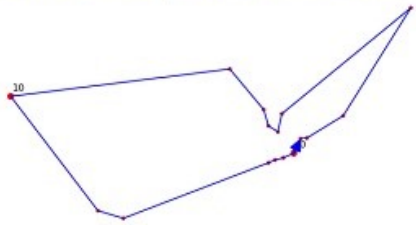
a. Cantas horas diarias debería traballar o telepiloto do dron?

b. Cal sería o percorrido óptimo?

c. Se a autonomía do dron fose de 130 km, en que hospitais se lle deberían cambiar as baterías?

```
CAT=[catalunha][0]()[0]
VH=CAT[0:16]
PVH=[a[1] for a in VH]
D=cicloham(PVH)
print 'a. Horas de piloto=', lon(D)[0]/130000+16/6
print
print 'b. '
NL=[]
for z in D:
    NL=NL+[a for a in VH if a[1]==z]
HR=[a[0] for a in NL]
n=HR.index(BARCELONA)
HR=HR[n:-1]+HR[:n+1]
D=D[n:-1]+D[:n+1]
show(vercamino(D), figsize=[5,5])
for i in range(ceil(len(HR)/10)):
    print html(""+str (10*i)+" "+str(HR[10*i:10*(i+1)])+"")
print'c. '
Parciales=[round(lon(D[:5])[0],1),round(lon(D[4:10])
[0],1),round(lon(D[9:11])[0],1),round(lon(D[10:13])
[0],1),round(lon(D[12:])[0],1)]
print 'Cambio de baterias:', [HR[0],HR[4],HR[9],HR[10],HR[12]]
```

```
a. Horas de piloto= 6.28505850087925

b.
0 [BARCELONA, Santa_Coloma_de_Gramenet, Badalona, Mataro, Girona, Sabadell, Sant_Cugat_del_Valles, Rubi, Terrassa, Manresa]

10 [Lleida, Reus, Tarragona, Sant_Boi_de_Llobregat, Cornella_de_Llobregat, Hospitalet_de_Llobregat, BARCELONA]

c.
Cambio de baterias: [BARCELONA, Girona, Manresa, Lleida, Tarragona]
```

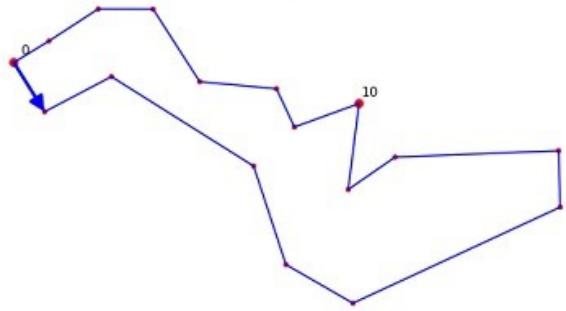


Exercicio 19:

Deseñar un circuíto turístico polas principais cidades da antiga Checoslovaquia. Calcular a lonxitude total do traxecto e indicar a etapa de menor lonxitude entre cidades.

```
CE=[chequia,eslovaquia]
VCE=viaje(CE)
show(VCE[1],figsize=[5,5])
```

```
0 [Karlovi_Varo, Pilsen, PRAGA, Brno, BRATISLAVA, Komarno, Kosice, Bardejov, Zilina, Trenc
10 [Ostrava, Olomouc, Sumperk, Pardubice, Ostrava, Decin, Chomulov, Karlovi_Varo]
```




```
print 'A lonxitude total do circuito é',lon([a[1] for a in
VCE[0]])[0]/1000,'km'
```

```
A lonxitude total do circuito é 1676.62058726735 km
```

```
L=[a[1] for a in VCE[0]]
LE=[abs(L[k]-L[k+1]) for k in range(len(L)-1)]
MLE=minimos(LE)
print 'A etapa de menor lonxitude transcorre entre'
print VCE[0][MLE[1][0]][0],'e',VCE[0][MLE[1][0]+1][0]
print 'e mide', MLE[0]/1000,'km'
```

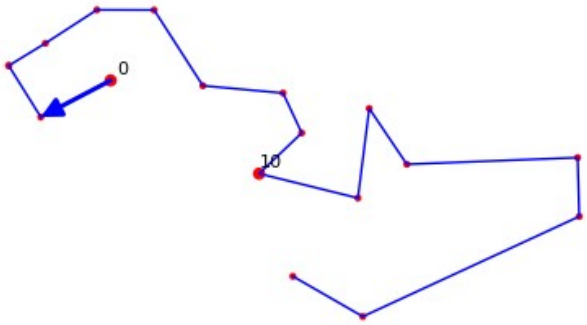
```
A etapa de menor lonxitude transcorre entre
Chomulov e Karlovi_Varo
e mide 45.9293996665139 km
```

Exercicio 20:

Describir o menor camiño hamiltoniano da antiga Checoslovaquia que comece en Praga e termine en Bratislava e calcular a súa lonxitude.

```
C=[a[1] for a in VCE[0]]
N=[a[0] for a in VCE[0]]
P=N.index(PRAGA)
B=N.index(BRATISLAVA)
PG=pegalistas(reverso(C[:P+1])+reverso(C[B:]),C[P+1:B])
M=mejora(PG,6)
show(vercamino(M),axes=False,figsize=[4,4])
print 'A lonxitude é', lon(M)[0]/1000,'km'
```

```
A lonxitude é 1477.21836707078 km
```



3. Worksheet 3

```
load (DATA + 'Worksheet0.sage')
load (DATA + 'Worksheet1.sage')
load (DATA + 'Worksheet2.sage')
load (DATA + 'Worksheet3.sage')
```

3.1. Problema de Cauchy

Dada unha función $f : [t_0, t_0 + T] \times \mathbb{R}^n \to \mathbb{R}^n$ e un punto $\mathrm{x}_0 \in \mathbb{R}^n$, búscase unha curva $c : [t_0, t_0 + T] \to \mathbb{R}^n$ tal que

$$\begin{cases} c(t_0) = \mathrm{x}_0 \\ c'(t) = f(t, c(t)) \quad \forall t \in [t_0, t_0 + T] \end{cases}$$

De forma abreviada enúnciase como segue: Buscar a solución da ecuación diferencial $x' = f(t, x)$ de condición inicial x_0.

Tamén poden suscitarnos unha ecuación diferencial $x'' = g(t, x, x')$ que chamaremos de segunda orde, pero facendo

$$\begin{cases} x' = v \\ v' = g(t, x, v) \end{cases}$$

estaremos no caso dunha ecuación de primeira orde $\mathrm{x}' = f(t, \mathrm{x})$ con $\mathrm{x} = (x, v)$ e $f(t, \mathrm{x}) = (v, g(t, x, v))$. A condición inicial será $\mathrm{x}_0 = (x(t_0), x'(t_0))$. Este esquema é xeneralizable para ecuacións diferenciais de calquera orde do tipo $x^{n)} = g(t, x, x', \cdots, x^{n-1)})$, polo que nos centraremos en ecuacións diferenciais vectoriais de orde 1.

Se f é continua e existe un número real λ tal que

$$\|f(t,x)-f(t,y)\| \leq \lambda \|x-y\| \quad \forall (t,x,y) \in [t_0, t_0+T] \times \mathbb{R}^n \times \mathbb{R}^n$$

o teorema de Picard-Lindelöf asegura a existencia e unicidade de solución.

Exemplo 1: Ecuación de Malthus $x' = rx$

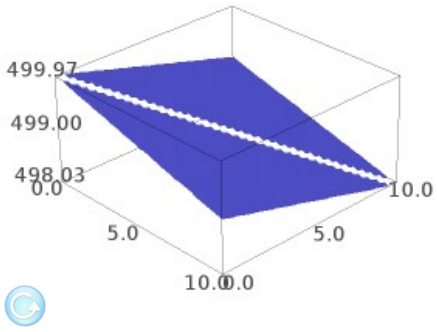
É claro que $|rx - ry| \leq |r||x-y|$ e a única solución de condición inicial x_0 é, trivialmente,

$$c(t) = x_0 e^{rt}$$

Exemplo 2: Ecuación loxística ou de Verhulst $x' = kx(K-x)$

É claro que $\frac{|kx(K-x)-ky(K-y)|}{|x-y|}$ non é acoutado en $\mathbb{R}^2$

```
k=.1; K=5000
f(x,y)=abs(k*x*(K-x)-k*y*(K-y))/abs(x-y)
show(plot3d(f,(x,0,10),(y,0,10)),figsize=[3,3])
```



e non podemos deducir a existencia e unicidade de solución do teorema de Picard Lindelöf. Con todo, por tratarse dunha ecuación diferencial en variables separadas, é fácil comprobar que

$$c(t) = \frac{K\frac{x_0}{K-x_0}e^{Kkt}}{1+\frac{x_0}{K-x_0}e^{Kkt}}$$

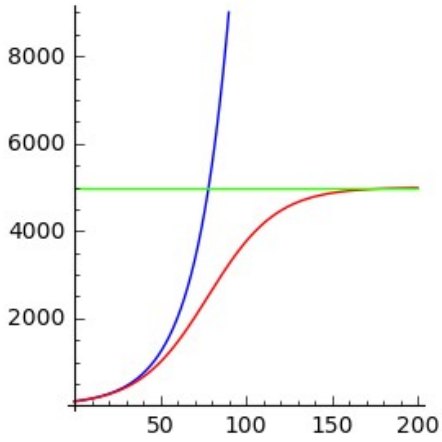
é unha solución de condición inicial x_0:

```
var('k K x0')
c(t)=K*(x0/(K-x0))*exp(K*k*t)/(1+(x0/(K-x0))*exp(K*k*t))
cp(t)=diff(c,t)
print 'cp(t)-k*c(t)*(K-c(t))=',(cp(t)-k*c(t)*(K-
c(t))).full_simplify()
print 'c(0)=',c(0).full_simplify()
```

```
cp(t)-k*c(t)*(K-c(t))= 0
c(0)= x0
```

Comparamos as solucións das ecuacións de Malthus e Verhulst para $k = .00001, \quad K = 5000, \quad r = k \cdot K$ e $x_0 = 100$

```
k=.00001
K=5000
r=k*K
x0=100
M(t)=x0*exp(r*t)
c(t)=K*(x0/(K-x0))*exp(K*k*t)/(1+(x0/(K-x0))*exp(K*k*t))
CM=plot(M,(t,0,90))
CV=plot(c,(t,0,200),hue=0)
CC=plot(K,(t,0,200),hue=.3)
show(CM+CV+CC,figsize=[3,3])
```



Exemplo 3: Ecuación lineal de coeficientes constantes $f(t, \mathbf{x}) = A\mathbf{x} + \mathbf{u}(t)$

Sexa $f(t, \mathbf{x}) = A\mathbf{x} + \mathbf{u}(t)$ con $A \in \mathfrak{M}_n(\mathbb{R})$ e $\mathbf{u} : [t0, t0 + T] \to \mathbb{R}^n$ unha función continua. Como

$$\|f(t, \mathbf{x}) - f(t, \mathbf{y})\| \leq \|A\|\|\mathbf{x} - \mathbf{y}\|$$

o teorema de Picard-Lindelöf asegura a existencia e unicidade de solución para unha condición inicial $\mathbf{x}_0$.

En cursos elementais de ecuacións diferenciais utilízase a exponencial de matrices para expresar a solución na forma

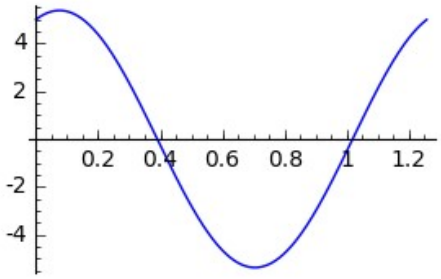
$$c(t) = e^{A\cdot(t-t_0)} \cdot \mathbf{x}_0 + \int_{t_0}^{t} e^{A\cdot(t-s)} \cdot \mathbf{u}(s)ds$$

Aquí podemos utilizar a nosa función **expm(**A,t**)**.

Así, se nos piden a solución de $x'' = -25x$ coas condicións iniciais $x(0) = 5$ e $x'(0) = 10$ resolveremos

$$\begin{pmatrix} x \\ v \end{pmatrix}' = \begin{pmatrix} 0 & 1 \\ -25 & 0 \end{pmatrix} \cdot \begin{pmatrix} x \\ v \end{pmatrix} \quad \text{con} \quad \mathbf{x_0} = \begin{pmatrix} 5 \\ 10 \end{pmatrix}$$

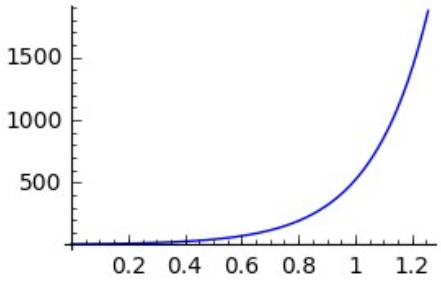
```
var('t')
A=matrix([[0,1],[-25,0]])
x0=vector([5,10])
fa(t)=(expm(A,t)*x0)[0]
show(plot(fa,(t,0,2*pi/5)),figsize=[3,2])
```



e se nos piden a solución de $x'' = 25x$ coas condicións iniciais $x(0) = 5$ e $x'(0) = 10$ resolveremos

$$\begin{pmatrix} x \\ v \end{pmatrix}' = \begin{pmatrix} 0 & 1 \\ 25 & 0 \end{pmatrix} \cdot \begin{pmatrix} x \\ v \end{pmatrix} \quad \text{con} \quad \mathbf{x_0} = \begin{pmatrix} 5 \\ 10 \end{pmatrix}$$

```
A=matrix([[0,1],[25,0]])
x0=vector([5,10])
fna(t)=(expm(A,t)*x0)[0]
show(plot(fna,(t,0,2*pi/5)),figsize=[3,2])
```



Exemplo 4: Ecuación autónoma $\mathrm{x}' = f(\mathrm{x})$

Cando a función $f(t, \mathrm{x})$ non depende explicitamente do tempo, a ecuación

diferencial chámase **autónoma**. Se $f:\Omega\subset\mathbb{R}^n\to\mathbb{R}^n$ é diferenciable no aberto Ω e se x_e é un cero da función f, a curva constante $\mathrm{x}(t)=\mathrm{x}_e$ é, trivialmente, unha solución particular de $\mathrm{x}'=f(\mathrm{x})$ e, por iso, diremos que x_e é un punto estacionario da ecuación. Na contorna de x_e, podemos aproximar $\mathrm{x}'=f(\mathrm{x})$ pola ecuación lineal

$$\mathrm{x}'=f(\mathrm{x}_e)+Df(\mathrm{x}_e)(\mathrm{x}-\mathrm{x}_e)=Df(\mathrm{x}_e)\mathrm{x}-Df(\mathrm{x}_e)\mathrm{x}_e$$

Por exemplo, a función $f(x,y)=[0.25x-0.01xy,-y+0.01xy]$ ten dous puntos estacionarios

```
f(x,y)=[.25*x-.01*x*y,-y+0.01*x*y]
S=solve(list(f(x,y)),[x,y],solution_dict=true)
print S
```

```
[{y: 0, x: 0}, {y: 25, x: 100}]
```

Nunha contorna do punto estacionario $\mathbf{x}_e=(0,0)$, a ecuación de partida pódese aproximar pola ecuación lineal $\mathbf{x}'=A\cdot\mathbf{x}+\mathbf{u}$, que determinamos como segue

```
xe=vector([0,0])
A=JAC(f)(x=0,y=0)
u=-A*xe
print 'A='
show(A)
print 'u='
show(u)
```

```
A=
```

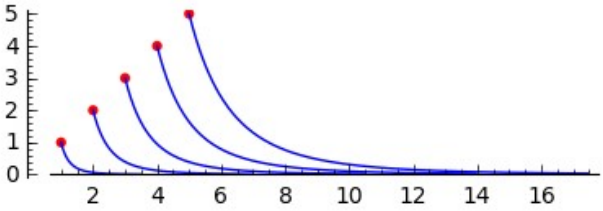
$$\begin{pmatrix} 0.250000000000000 & -0.000000000000000 \\ 0.000000000000000 & -1.00000000000000 \end{pmatrix}$$

```
u=
```

$$(0.000000000000000,\ 0.000000000000000)$$

Para condicións iniciais $\mathbf{x}_0 = (k, k)$ con $k \in [1, .., 5]$ obtemos as seguintes solucións da ecuación lineal

```
var('s')
D=[]
for k in [1,..,5]:
    x0=vector([k,k])
    V=expm(A,t)*x0
    assume(t>0)
    W=integrate(expm(A,t-s)*u,s,0,t)
    c(t)=list(V+W)
    D.append(parametric_plot(c,
(t,0,5))+point([k,k],pointsize=20,hue=0))
show(sum(D),figsize=[4,4])
```



Analogamente, nunha contorna do punto estacionario $\mathbf{x}_e = (100, 25)$, a ecuación de partida pódese aproximar pola ecuación lineal

```
xe=vector([100,25])
A=JAC(f)(x=100,y=25)
u=-A*xe
print 'A='
show(A)
print 'u='
show(u)
```

A=

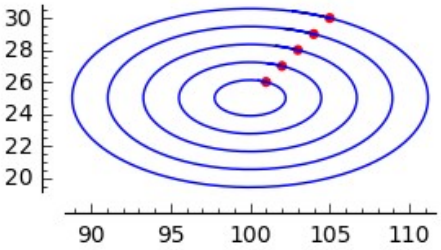
$$\begin{pmatrix} 0.000000000000000 & -1.00000000000000 \\ 0.250000000000000 & 0.000000000000000 \end{pmatrix}$$

u=

$$(25.0000000000000,\ -25.0000000000000)$$

Para condicións iniciais $\mathbf{x}_0 = (100+k, 25+k)$ con $k \in [1,..,5]$, obtemos as seguintes solucións da ecuación lineal

```
var('s'); D=[]
for k in [1,..,5]:
    x0=vector([100+k,25+k])
    V=expm(A,t)*x0
    assume(t>0)
    W=integrate(expm(A,t-s)*u,s,0,t)
    c(t)=list(V+W)
    D.append(parametric_plot(c,
(t,0,13))+point([100+k,25+k],pointsize=20,hue=0))
show(sum(D),figsize=[3,3])
```



Aínda que podiamos asegurar a existencia de solución $c(t)$ para o problema de Cauchy $\mathbf{x}' = f(t, \mathbf{x})$ de condición inicial $\mathbf{x}_0$, non sempre a podemos expresar en termos elementais, polo que optamos por facer unha partición equidistante

$$t_0 < t_1 < \cdots < t_k < t_{k+1} < \cdots < t_N = t_0 + T$$

do intervalo $[t_0, t_0+T]$, $t_{k+1} - t_k = \frac{T}{N} := h \quad \forall k$, e dar unha lista de puntos $[\mathrm{x}_k\ ]$ que aproxime a lista $[c(t_k)]$ mediante un proceso recursivo da forma

$$\begin{cases} \mathrm{x}_0 = c(t_0) \\ \mathrm{x}_{k+1} = \mathrm{x}_k + h\phi(t_k, \mathrm{x}_k; h) \end{cases}$$

que se chama *método dun paso* cando a función ϕ utilizada para achar x_{k+1} só

depende de t_k, x_k e o paso h. Exemplos de métodos *dun paso* son:

3.1.1. Método de Euler

Posto que $c' = f(t, c)$, aproximando a integral pola área do paralelogramo de base h e altura $f(t_k, x_k)$ deducimos que

$$c(t_{k+1}) - c(t_k) = \int_{t_k}^{t_{k+1}} f(t, c(t))dt \approx hf(t_k, \mathrm{x}_k)$$

$$\text{onde} \quad \phi(t_k, \mathrm{x}_k; h) = f(t_k, \mathrm{x}_k)$$

Se $c(t_0) = \mathrm{x}_0$ obtemos a lista $[\mathrm{x}_k]$ por iteración: $\mathrm{x}_{k+1} = \mathrm{x}_k + hf(t_k, \mathrm{x}_k) \quad \forall k = 0, 1, .., N$, que implementamos en Sage na función **euler(**f, CI, I, N**)**.

3.1.2. Método de Runge-Kutta

Se definimos as magnitudes

$$\begin{cases} F_{k1} = f(t_k, \mathrm{x}_k) \\ F_{k2} = f(t_k + \frac{h}{2}, \mathrm{x}_k + \frac{h}{2} F_{k1}) \\ F_{k3} = f(t_k + \frac{h}{2}, \mathrm{x}_k + \frac{h}{2} F_{k2}) \\ F_{k4} = f(t_{k+1}, \mathrm{x}_k + hF_{k3}) \end{cases}$$

e tomamos $\phi(t_k, \mathrm{x}_k; h) = \frac{F_{k1} + 2F_{k2} + 2F_{k3} + F_{k4}}{6}$ obtemos a aproximación da integral

$$c(t_{k+1}) - c(t_k) = \int_{t_k}^{t_{k+1}} f(t, c(t))dt \approx h\phi(t_k, \mathrm{x}_k; h)$$

que é outro método *dun paso* que implementamos en Sage na función **rungekutta(**f, CI, I, N**)**.

Estes métodos son casos particulares dun procedemento xeral que designamos $(p, \mathrm{c}, \mathrm{b}, A)$:

Dado $p \in \mathbb{N}$, os vectores $\mathrm{c},\ \mathrm{b}\ \in \mathbb{R}^p$ e a matriz subdiagonal $A \in \mathfrak{M}_{(p,p)}(\mathbb{R})$:

$$\mathrm{c} = \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ \vdots \\ c_p \end{pmatrix}, \quad \mathrm{b} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_p \end{pmatrix}, \quad A = \begin{pmatrix} 0 & 0 & \cdots & 0 & 0 \\ a_{21} & 0 & \cdots & 0 & 0 \\ a_{31} & a_{32} & \cdots & 0 & 0 \\ \vdots & \vdots & \cdots & 0 & 0 \\ a_{p1} & a_{p2} & \cdots & a_{pp-1} & 0 \end{pmatrix},$$

defínense as magnitudes

$$\begin{cases} F_{k1} = f(t_k + c_1 h, \mathrm{x}_k) \\ F_{k2} = f(t_k + c_2 h, \mathrm{x}_k + a_{21} F_{k1} h) \\ F_{k3} = f(t_k + c_3 h, x_k + a_{31} F_{k1} h + a_{32} F_{k2} h) \\ \qquad \cdots \\ F_{kp} = f(t_k + c_p h, \mathrm{x}_k + a_{p1} F_{k1} h + a_{p2} F_{k2} h + \cdots + a_{pp-1} F_{kp-1} h) \end{cases}$$

e considérase a función

$$\phi(t_k, \mathrm{x}_k, h) = b_1 F_{k1} + \cdots + b_p F_{kp}$$

Para $p = 1$, $\mathrm{c} = 0_p$, $\mathrm{b} = 1_p$, $A = 0_{pp}$, obtemos a $\phi(t_k, \mathrm{x}_k; h) = f(t_k, \mathrm{x}_k)$ do método de Euler.

Para $p = 4$, $\mathrm{c} = \begin{pmatrix} 0 \\ \frac{1}{2} \\ \frac{1}{2} \\ 1 \end{pmatrix}$, $\mathrm{b} = \begin{pmatrix} \frac{1}{6} \\ \frac{2}{6} \\ \frac{2}{6} \\ \frac{1}{6} \end{pmatrix}$, $A = \begin{pmatrix} 0 & 0 & 0 & 0 \\ \frac{1}{2} & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$,

temos a función $\phi(t_k, \mathrm{x}_k; h) = \frac{F_{k1} + 2F_{k2} + 2F_{k3} + F_{k4}}{6}$ do método de Runge-Kutta.

Para precisar a orde de aproximación destes métodos damos as siguientes

Definicións:

Un método d*un paso* de función ϕ dise:

1. **Consistente** se

$$\lim_{h\to 0}\left(\sum_{k=0}^{N-1}||\mathrm{x}(t_{k+1})-\mathrm{x}(t_k)-h\phi(t_k,\mathrm{x}(t_k);h)||\right)=0$$

2. **De orden p** se existe unha constante $K\geq 0$ tal que

$$\sum_{k=0}^{N-1}||\mathrm{x}(t_{k+1}-\mathrm{x}(t_k)-h\phi(t_k,\mathrm{x}(t_k);h)||\leq Kh^p$$

3. **Estable** se existe unha constante M independente de h tal que para todas as N-tuplas (y_k) e (ε_k) que verifiquen

$$\mathrm{y}_{k+1}=\mathrm{y}_k+h\phi(t_k,\mathrm{y}_k;h)+\varepsilon_k$$

cúmprese que

$$\max\{||\mathrm{y}_k-\mathrm{x}_k||\ |k=0,\cdots,N\}\leq M\left(||\mathrm{y}_0-\mathrm{x}_0||+\sum_{k=0}^{N-1}||\varepsilon_k||\right)$$

4. **Converxente** se o **erro de discretización**

$$E(h)=\max\{||\mathrm{x}_k-\mathrm{x}(t_k)||\ |k=0,\cdots,N\}$$

$$\text{cumpre que}\qquad \lim_{h\to 0}E(h)=0$$

Consecuencia inmediata destas definicións é o seguinte

Lema:

Se un método é consistente e estable, é converxente.

Demostración:

Sexa

$$\varepsilon_k = \mathrm{x}(t_{k+1}) - \mathrm{x}(t_k) - h\phi(t_k, \mathrm{x}(t_k); h)$$

Por ser o método estable, existe unha constante M tal que

$$E(h) \leq M\Big(\sum_{k=0}^{N-1} ||\mathrm{x}(t_{k+1}) - \mathrm{x}(t_k) - h\phi(t_k, \mathrm{x}(t_k); h)|| \Big)$$

e, por consistente, $\lim_{h\to 0} E(h) = 0$. $\diamondsuit$

Non son tan inmediatos os lemas seguintes cuxas demostracións poden verse en [9].

Lema:

Unha condición necesaria e suficiente para que un método d*un paso* sexa consistente é que

$$\phi(t, \mathrm{x}; 0) = f(t, \mathrm{x}) \quad \forall (t, \mathrm{x}) \in [t_0, t_0 + T] \times \mathbb{R}^n. \quad \diamondsuit$$

Lema:

Unha condición necesaria e suficiente para que un método dun paso sexa estable é que exista un $H > 0$ e unha constante L tales que

$$||\phi(t, \mathrm{x}; h) - \phi(t, \mathrm{y}; h)|| \leq L||\mathrm{x} - \mathrm{y}||$$

$$\forall t \in [t_0, t_0 + T], \quad \forall \mathrm{x}, \mathrm{y} \in \mathbb{R}^n, \quad \forall h \in [0, H]. \quad \diamondsuit$$

Lema:

A condición necesaria e suficiente para que o procedemento $(p, \mathrm{c}, \mathrm{b}, A)$ sexa

1. De orde 1 é que

$$(\mathrm{b}|1_p) = 1$$

2. De orde 4 é que

$$A1_p = C1_p, \quad (\mathrm{b}|C^3 1_p) = \frac{1}{4}, \quad (\mathrm{b}|AC^2 1_p) = \frac{1}{12},$$

$$(\mathrm{b}|A^2C1_p) = \frac{1}{24}, \quad (\mathrm{b}|CAC1_p) = \frac{1}{8}$$

onde C é a matriz diagonal de vector c.

Xa que logo, as nosas funcións **euler(**f, CI, I, N**)** e **rungekutta(**f, CI, I, N**)** son, respectivamente, métodos de orde 1 e de orde 4.

Exemplo 5:

A poboación de mosquitos da froita está rexida pola ecuación de Malthus m'(t) = 0.098m(t) co tempo en días. Se partimos de 111 mosquitos. Cal será a poboación de mosquitos ao cabo dun día? e ao cabo de 100 días?

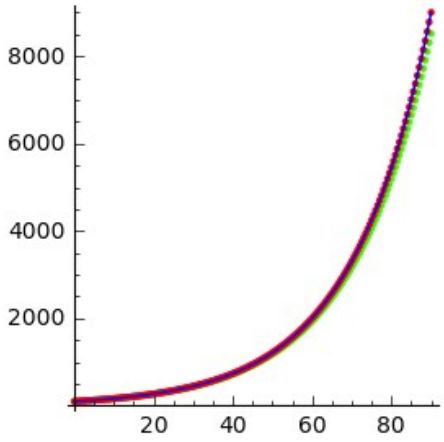
```
f(t,m)=[.098*m]
CI=vector([111])
I=[0,100]
N=24*I[1]
S=rungekutta(f,CI,I,N)
print 'Ao cabo dun día', round(S[24][0]), 'mosquitos'
print 'Ao cabo de 100 días', round(S[2400][0]), 'mosquitos'
```

```
Ao cabo dun día 122 mosquitos
Ao cabo de 100 días 2001746 mosquitos
```

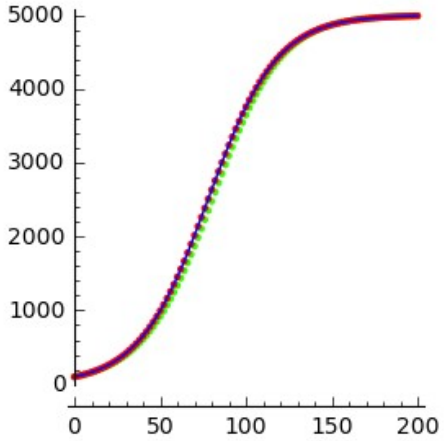
Exemplo 6:

Aplicar o método de Euler e o de Runge-Kutta ás ecuacións de Malthus e Verhulst e comparar as solucións coas exactas.

```
k=.00001; K=5000
r=k*K
x0=100; CI=vector([x0])
M(t)=x0*exp(r*t)
CM=plot(M,(t,0,90))
fm(t,x)=[r*x]
I=[0,90]; N=I[1]*2
em=euler(fm,CI,I,N)
EM=[a[0] for a in em]
rm=rungekutta(fm,CI,I,N)
RM=[a[0] for a in rm]
T=[0,1/2,..,90]
GEM=point(zip(T,EM),hue=.3)
GRM=point(zip(T,RM),hue=0)
show(GEM+GRM+CM, figsize=[3,3])
```



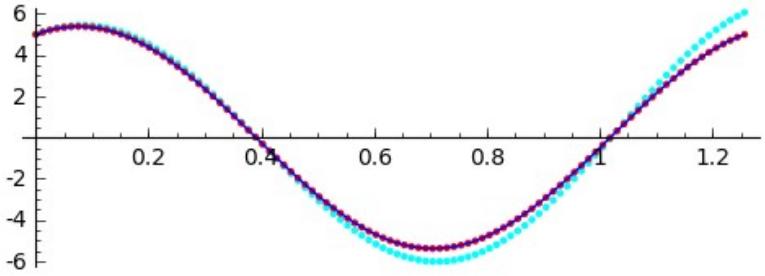

```
k=.00001; K=5000
x0=100; CI=vector([x0])
V(t)=K*(x0/(K-x0))*exp(K*k*t)/(1+(x0/(K-x0))*exp(K*k*t))
CV=plot(V,(t,0,200))
fv(t,x)=[k*x*(K-x)]
I=[0,200]; N=I[1]/2
ev=euler(fv,CI,I,N)
EV=[a[0] for a in ev]
rv=rungekutta(fv,CI,I,N)
RV=[a[0] for a in rv];T=[0,2,..,200]
GEV=point(zip(T,EV),hue=.3)
GRV=point(zip(T,RV),hue=0)
show(GEV+GRV+CV, figsize=[3,3])
```



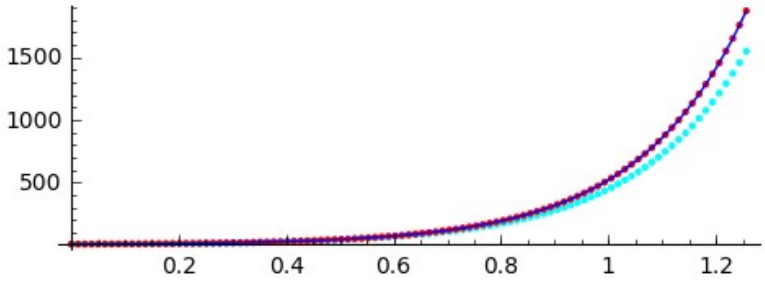
Exemplo 7:

Aplicar o método de Euler e o de Runge-Kutta ás ecuacións lineais do exemplo 3 e comparar as solucións coas exactas.

```
f(t,x,v)=[v,-25*x]
CI=vector([5,10])
L=(2*pi/5).n()
I=[0,L]
N=100
E=euler(f,CI,I,N)
RK=rungekutta(f,CI,I,N)
T=[0,L/100,..,L]
XE=[a[0] for a in E]
XRK=[a[0] for a in RK]
show(plot(fa,(t,0,L))+point(zip(T,XE),hue=0.5)+
point(zip(T,XRK),hue=0),figsize=[5,2])
```




```
f(t,x,v)=[v,25*x]
CI=vector([5,10])
L=(2*pi/5).n()
I=[0,L]
N=100
E=euler(f,CI,I,N)
RK=rungekutta(f,CI,I,N)
T=[0,L/100,..,L]
XE=[a[0] for a in E]
XRK=[a[0] for a in RK]
show(plot(fna,(t,0,L))+point(zip(T,XE),hue=0.5)+
point(zip(T,XRK),hue=0),figsize=[5,2])
```



A boa aproximación de orde 4 que dá a función **rungekutta(**f, CI, I, N**)** xustifica que a utilicemos de forma sistemática ata nas ecuacións lineais que, como sabemos, teñen solucións exactas expresables en termos elementais, máxime se temos en conta que a orde de Sage **exp**(A*t) e a nosa función **expm**(A,t) poden fallar nalgún caso.

Exemplo 8:

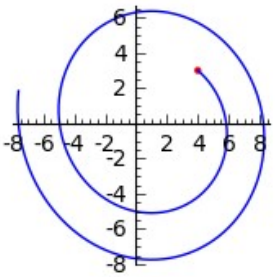
Achar a solucion da ecuación $\mathbf{x}' = A\mathbf{x} + \mathbf{u}(t)$ nos casos:

$$a) \quad A = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \quad , \quad \mathbf{u}(t) = \begin{pmatrix} \cos(t) \\ 1 \end{pmatrix} \quad \text{e} \quad CI = \begin{pmatrix} 4 \\ 3 \end{pmatrix}$$

$$b) \quad A = \begin{pmatrix} 1 & -1 \\ 0 & -3 \end{pmatrix} \quad , \quad \mathbf{u}(t) = \begin{pmatrix} \sin(t) \\ \cos(t) \end{pmatrix} \quad \text{e} \quad CI = \begin{pmatrix} 2 \\ 5 \end{pmatrix}$$

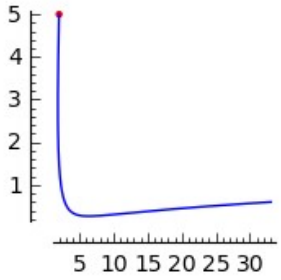
```
f(t,x,y)=[y+cos(t),1-x]
I=[0,10]
N=60*I[1]
CI=vector([4,3])
RK=rungekutta(f,CI,I,N)
print 'a.'
show(line(RK)+point([4,3],hue=0),figsize=[2,2])
```

```
a.
```




```
f(t,x,y)=[x-y+cosh(.3*t),-3*y+sinh(.5*t)]
I=[0,3]
N=60*I[1]
CI=vector([2,5])
RK=rungekutta(f,CI,I,N)
print 'b.'
show(line(RK)+point([2,5],hue=0),figsize=[2,2])
```

b.



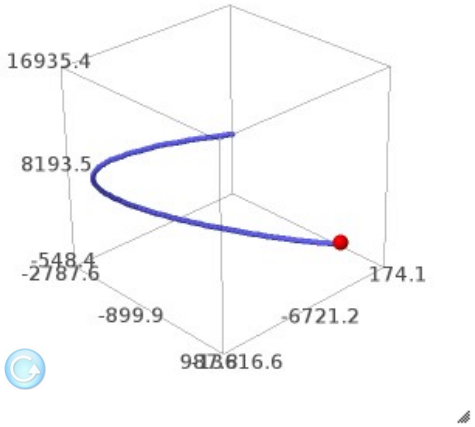
Exemplo 9:

Achar a solucion do problema de Cauchy $\mathbf{x}' = A\mathbf{x} + \mathbf{u}(t)$ sendo

$$A = \begin{pmatrix} 1 & 1 & 3 \\ 9 & 1 & 2 \\ 8 & -5 & 0 \end{pmatrix} \quad , \quad \mathbf{u}(t) = \begin{pmatrix} \sin(t) \\ \cos(t) \\ 1 \end{pmatrix} \quad \text{e} \quad CI = \begin{pmatrix} 4 \\ 3 \\ 5 \end{pmatrix}$$

```
f(t,x,y,z)=[x+y+3*z+sin(t),9*x+y+2*z+cos(t),8*x-5*y+1]
CI=vector([4,3,5])
I=[0,2]
N=100*I[1]
RK=rungekutta(f,CI,I,N)
```

```
show(line3d(RK,thickness=7)+point3d(CI,color='red',size=20),
figsize=[3,3,3])
```



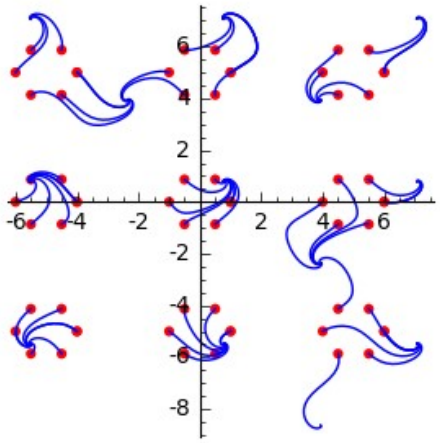
A facilidade de representar as solucións numéricas dun problema de Cauchy, permítenos considerar a superposición de gráficas con distintas condicións iniciais e visualizar a dependencia das mesmas.

Exemplo 10:

Estudar como dependen das condicións iniciais, as solucións da ecuación

$$\begin{cases} x' = cos(x+y) \\ y' = sin(x-y) \end{cases}$$

```
n=5
L=[[-n,n],[-n,0],[-n,-n],[0,n],[0,0],[0,-n],[n,n],[n,0],[n,-n]]
f(t,x1,x2)=[cos(x1+x2),sin(x1-x2)]
I=[0,40]
N=200
D=[]
for P in L:
    G=[]
    for j in [0,2*pi/6..2*pi]:
        CI=vector(P)+vector([cos(j),sin(j)])
        X=rungekutta(f,CI,I,N)
        G=G+[line2d(X)+point([CI],color='red',pointsize=20)]
    D.append(sum(G))
show(sum(D),figsize=[3,3])
```

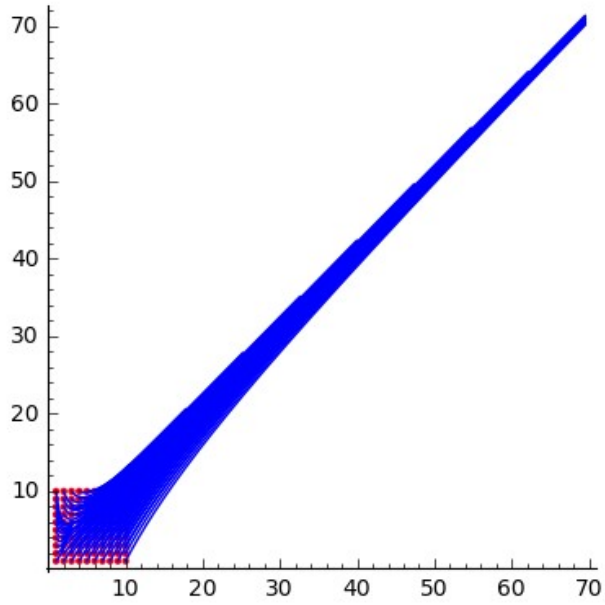


Exemplo 11:

Estudar como dependen das condicións iniciais no primeiro cuadrante, as solucións da ecuación:

$$\begin{pmatrix} x \\ y \end{pmatrix}' = \begin{pmatrix} 1 & 0 \\ 2 & -1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} -t \\ \sin(t) \end{pmatrix}$$

```
f(t,x,y)=[-t+x,2*x-y+sin(t)]
I=[0,2]
Q=[]
P=[1,2,..,10]
for n in P:
    for m in P:
        CI=vector([n,m])
        RK=rungekutta(f,CI,I,N)
        X=[a[0] for a in RK]
        Y=[a[1] for a in RK]
        Q.append(line(zip(X,Y))+point([n,m],hue=0))
show(sum(Q),figsize=[4,4])
```



3.2. Problemas de Cauchy en Ciencia e Tecnoloxía

A maior dificultade que atopa o alumno ante un problema diferencial suscitado retoricamente é detectar a función $f : [t_0, t_0 + T] \times \mathbb{R}^n \to \mathbb{R}^n$ subxacente. Na maioría dos casos, f expresa unha lei experimental descuberta no laboratorio ou observada na Natureza.

3.2.1. Paso ao continuo

Exemplo 1:

A evolución simple dunha poboación P no ano n comezouse suscitando a seguinte ecuación en diferenzas $P[n] = k \cdot P[n-1]$. As seguintes manipulacións obvias

$$P[n] = P[n-1] \cdot (1 + k - 1)$$

$$P[n] = P[n-1] + (k-1)P[n-1]$$

lévannos a

$$\frac{P[n]-P[n-1]}{n-(n-1)}=(k-1)P[n-1]$$

e, pasando ao continuo, suxírenos unha familia de ecuacións diferenciais do tipo Malthus

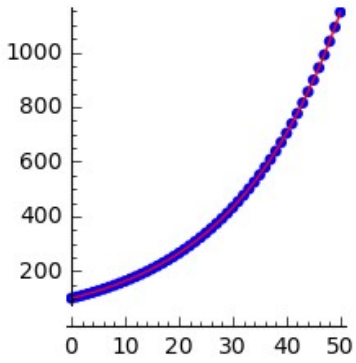
$$x'=r\cdot x$$

Se resolvemos a ecuación en diferenzas coa condición inicial $P[0]$, para cada n podemos determinar unha r_n, tal que a ecuación diferencial $x'=r_n\cdot x$ coa condición inicial $P[0]$ teña unha solución $c(t)$, tal que $c(n)=P[n]$.

Sabemos que a ecuación diferencial $x'=r\cdot x$ con condición inicial $P[0]$ ten a solución $c(t)=P[0]\cdot e^{rt}$ e, polo tanto, $r_n=\frac{log(\frac{P[n]}{P[0]})}{n}$, sendo log o logaritmo neperiano.

Para $k=1.05$, P[0]=100 e $n=50$ resolvemos a ecuación en diferenzas e a ecuación diferencial correspondente e visualizamos os resultados

```
P=[100]; r=1.05
for n in range(50):
    P.append(P[-1]*r)
T=[0,..,50]
SD=point(zip(T,P),pointsize=25)
rn=(log(P[-1]/100))/50
c(t)=100*exp(rn*t)
SC=plot(c,(t,0,50),hue=0)
show(SD+SC,figsize=[2.5,2.5])
```



Exemplo 2:

Fibonacci supuxo que unha parella de coellos adquiría plena capacidade reprodutora aos dous meses de vida e que, unha vez alcanzada, podía procrear outra parella cada mes. Preguntouse cantas parellas podería ter ao cabo dun ano se empezaba cunha parella de coellos recén nada. Na súa famosa sucesión podemos obter a resposta

$$P[0]=1, P[1]=1, P[2]=2, \ldots, P[n+2]=P[n]+P[n+1]$$

A relación de recorrencia anterior, $P[n+2]-P[n+1]=P[n]$, pódese escribir na forma

$$\frac{\frac{P[n+2]-P[n+1]}{(n+2)-(n+1)}-\frac{P[n+1]-P[n]}{(n+1)-n}}{(n+1)-n}=P[n]-\frac{P[n+1]-P[n]}{(n+1)-n}$$

e, pasando ao continuo, suxírenos unha familia de ecuacións diferenciais $x''=k(x-x')$ con $k>0$.

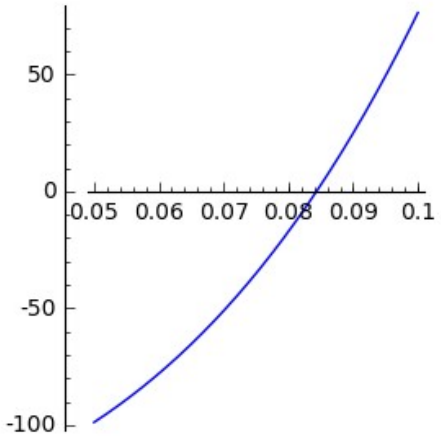
En calquera curso elemental de ecuacións diferenciais ensínannos que a ecuación $x''+kx'-kx=0$ ten como solución xeral $x(t)=Ae^{r_1 t}+Be^{r_2 t}$, sendo r_1 e r_2 raíces do polinomio característico $r^2+kr-k=0$.

Impoñendo as condicións iniciais $x(0)=1$ e $x'(0)=0$, obtemos a solución particular $c(t)=\frac{r_2}{r_2-r_1}e^{r_1 t}+\frac{r_1}{r_1-r_2}e^{r_2 t}$, que obviamente depende de k.

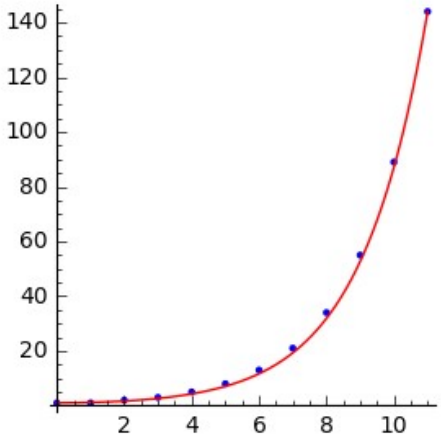
Se esiximos que $c(n)=P[n]$, podemos determinar unha k_n que nos proporcione unha ecuación diferencial cuxa solución $c(t)$ cumpra que $c(0)=1, c'(0)=0$ e $c(n)=P[n]$. Por exemplo, para $n=11$ temos

```
var('k')
r2=-k+sqrt(k^2+4*k)
r1=-k-sqrt(k^2+4*k)
A=r2/(r2-r1)
B=-r1/(r2-r1)
g(t,k)=A*exp(r1*t)+B*exp(r2*t)
G(k)=g(11,k)-Fibonacci(11)
show(plot(G,[0.05,0.1]),figsize=[3,3])
print 'k11=', biseccion(G,.05,.1,.0000001)
```

```
k11= 0.0843514442443848
```




```
SF=[Fibonacci(n) for n in range(12)]
T=range(12)
SD=point(zip(T,SF))
xk11(t)=g(t,.08435129)
show(plot(xk11,[0,11],hue=0)+SD,figsize=[3,3])
```



Exemplo 3:

A evolución loxística dunha poboación P no ano n debeu tratarse, inicialmente, mediante a seguinte ecuación en diferenzas

$$P[n] = k \cdot P[n-1](R - P[n-1])$$

Sinxelas manipulacións lévannos a

$$P[n] - P[n-1] = P[n-1] \cdot (kR - 1 - k \cdot P[n-1])$$

ou equivalentemente

$$P[n] - P[n-1] = kP[n-1] \cdot (\frac{kR-1}{k} - P[n-1])$$

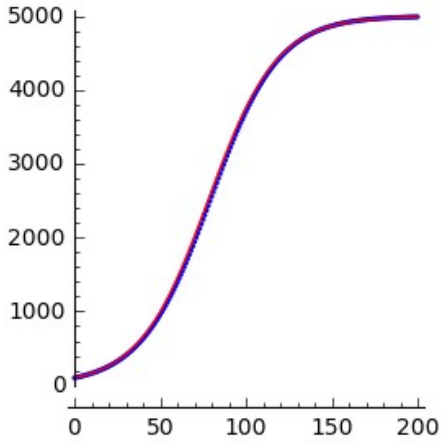
que pasando ao continuo, dá a familia de ecuacións diferenciais loxísticas

$$x' = kx(K - x) \quad \text{con} \quad K = \frac{kR-1}{k}$$

Como teñen capacidade de carga K admiten a solución particular constante $c(t) = K$ e o control $c(n) = P[n]$ que impoñiamos nos casos de Malthus e Fibonacci está incorporado, de saída, para n suficientemente grande.

Con $k = .00001$, $R = 105000$ e $P[0] = 100$, obtemos a solución da ecuación en diferenzas e comparámola coa solución da ecuación diferencial correspondente

```
P=[100]
k=.00001
R=105000
for n in range(201):
    P.append(k*P[-1]*(R-P[-1]))
T=[0,..,200]
SD=point(zip(T,P),pointsize=5)
x0=P[0]
K=(k*R-1)/k
V(t)=K*(x0/(K-x0))*exp(K*k*t)/(1+(x0/(K-x0))*exp(K*k*t))
CV=plot(V,(t,0,200),hue=0)

show(CV+SD,figsize=[3,3])
```



Exemplo 4:

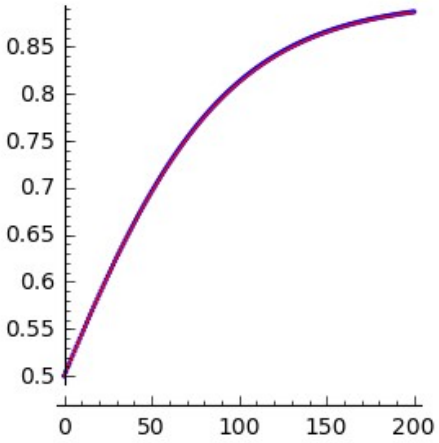
En economía, Goodwin (1913-1996) suscita a evolución da taxa de emprego no ano n mediante a ecuación en diferenzas

$$E[n] = E[n-1](1 + \frac{r}{E^\star} \cdot (E^\star - E[n-1]))$$

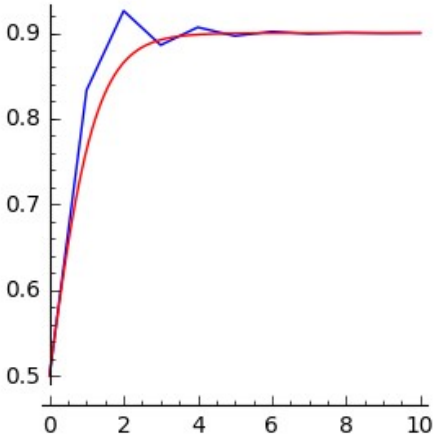
que, pasando ao continuo, asociámoslle a ecuación loxística $x' = \frac{r}{E^\star} \cdot x(E^\star - x)$.

Para $E[0] = 0.5$, $E^\star = 0.9$ e diferentes valores de r vemos que a similitude das solucións da ecuación en diferenzas e as solucións das correspondentes ecuacións loxísticas non sempre son satisfactorias

```
E=[.5]; r=.02; Es=.9
for n in range(201):
    E.append(E[-1]*(1+r/Es*(Es-E[-1])))
T=[0,..,200]
SD=point(zip(T,E),pointsize=5)
x0=E[0]
K=Es
k=r/Es
V(t)=K*(x0/(K-x0))*exp(K*k*t)/(1+(x0/(K-x0))*exp(K*k*t))
GV=plot(V,(t,0,200),hue=0)
show(SD+GV,figsize=[3,3])
```

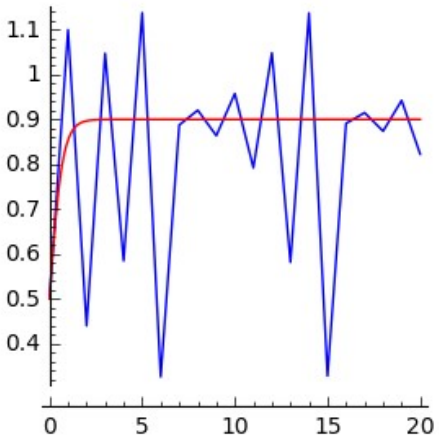



```
E=[.5]; r=1.5; Es=.9
for n in range(10):
    E.append(E[-1]*(1+r/Es*(Es-E[-1])))
T=[0,..,10]
SD=line(zip(T,E))
x0=E[0]
K=Es; k=r/Es
V(t)=K*(x0/(K-x0))*exp(K*k*t)/(1+(x0/(K-x0))*exp(K*k*t))
GV=plot(V,(t,0,10),hue=0)
show(SD+GV,figsize=[3,3])
```




```
E=[.5]; r=2.7; Es=.9
for n in range(21):
    E.append(E[-1]*(1+r/Es*(Es-E[-1])))
```

```
T=[0,..,20]
SD=line(zip(T,E))
x0=E[0]
K=Es
k=r/Es
V(t)=K*(x0/(K-x0))*exp(K*k*t)/(1+(x0/(K-x0))*exp(K*k*t))
GV=plot(V,(t,0,20),hue=0)
show(SD+GV,figsize=[3,3])
```



Estas discrepancias entre as solucións das ecuacións diferenciais e as ecuacións en diferenzas xustifican que, habitualmente, se traten como temas independentes.

3.2.2. Quecemento-Arrefriamento

En problemas de equilibrio térmico poden presentarse dúas situacións destacables: unha instantánea, como obter a temperatura T da mestura de dúas substancias que están a temperaturas diferentes, e outra temporal, como obter a temperatura $T(t)$ dun obxecto que no instante $t = 0$ se abandona nunha habitación climatizada a temperatura $T_a(t)$.

No primeiro caso, a calor cedida pola substancia a maior temperatura T_1, con masa m_1 e calor específica c_1 terá que ser igual á calor absorbida pola substancia a menor temperatura T_2, con masa m_2 e calor específica c_2. É dicir,

$$m_1 c_1 (T_1 - T) = m_2 c_2 (T - T_2)$$

e, xa que logo, a temperatura da mestura será

$$T = \frac{m_1 c_1}{m_1 c_1 + m_2 c_2} T_1 + \frac{m_2 c_2}{m_1 c_1 + m_2 c_2} T_2$$

O segundo caso está rexido pola lei de Newton que di que a razón de cambio da temperatura do obxecto é proporcional á diferenza entre a temperatura ambiente e a do obxecto:

$$T'(t) = k(T_a(t) - T(t))$$

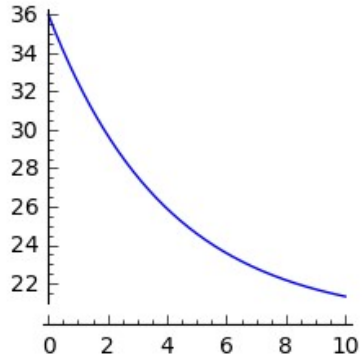
onde k é una constante, obviamente positiva, que se chama parámetro de arrefriamento do obxecto.

Exemplo 5:

Ás 22 horas áchase un cadáver nun edificio climatizado a $20º$ e o forense determina que a súa temperatura corporal é de $30º$. Achar a hora da morte supoñendo que estaba a 36º e o parámetro de arrefriamento dun cadáver é 0.25

```
f(t,x)=[.25*(20-x)]
CI=vector([36])
I=[0,10]
N=I[1]*60
S=rungekutta(f,CI,I,N)
R=[a[0] for a in S]
T=[0,1/60,..,10]
show(line(zip(T,R)),figsize=[2.5,2.5])
G=[r for r in R if r>30]
print 'a morte produciuse ás', 22-len(G)/60.n()
```

```
a morte produciuse ás 20.1166666666667
```



Exemplo 6:

Nun local a 20º de temperatura serven o café a 80º nunha cunca ao 75% da súa capacidade e acompáñano dunha xerra con leite a 5º.

Un cliente completa a súa cunca e espera tres minutos e outro, espera tres minutos e completa a súa cunca. Se a calor específica do leite é 0.93 e supoñemos que os parámetros de arrefriamento da cunca e a xerra son, respectivamente, 0.35 e 0.23, a que temperatura toma cada un dos dous clientes o seu café con leite?

```
80*0.75+5*0.25
```

```
61.2500000000000
```

```
N=.35
f(t,x)=[N*(20-x)]
CI=vector([245/4])
I=[0,3]
S=rungekutta(f,CI,I,180)
print 'O primeiro cliente tómao a', S[180][0],'°'
N=.35
f(t,x)=[N*(20-x)]
CI=vector([80])
I=[0,3]
S=rungekutta(f,CI,I,180)
C=S[180][0]
N=0.23
f(t,x)=[N*(20-x)]
CI=vector([5])
I=[0,3]
S=rungekutta(f,CI,I,180)
L=S[180][0]
CL=C*.75+L*.25
print 'O segundo cliente tómao a', CL,'°'
```

```
O primeiro cliente tómao a 34.4349321509821 °
O segundo cliente tómao a 33.8662884511623 °
```

3.2.3. Procesos de fabricación

A presión $f(x)$ nos cilindros de laminación en frío de pletinas planas obedece, no intervalo $[b, 0]$, á ecuación diferencial

$$f' = 2Rf\frac{x + \frac{f^{-\frac{1}{3}}}{3}}{h_a + Rx^2}$$

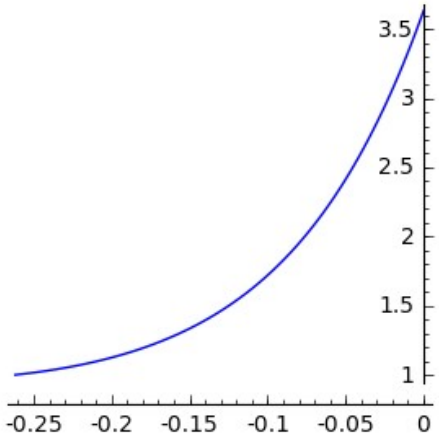
onde R é o radio dos cilindros e h_a é o espesor na saída.

Exemplo 7:

Se $b = -\pi/12, \quad R = 200, \quad h_a = 10$ e a condición inicial é $f(b) = 1$, representar f en $[b, 0]$ e calcular $\int_b^0 f(x)dx$.

```
R=200; h=10
b=-pi/12.n()
F(x,f)=[2*R*f*(x+(f^(-1/3))/3)/(h+R*x^2)]
CI=vector([1])
RG=rungekutta(F,CI,[b,0],100)
X=[b,b-b/100..0]
show(line([[X[k],RG[k][0]] for k in range(101)]),figsize=[3,3])
print 'A integral vale',((-b/100)*(2*sum([RG[k][0] for k in
range(101)])-RG[0][0]-RG[100][0])/2).n()
```

```
A integral vale 0.451507943487826
```



3.2.4. Oferta-demanda

Sexa p(t) a función prezo diario dun ben. O número de unidades do ben que desexan os consumidores por unidade de tempo chámase demanda D e, en xeral, é función de t, de p e das súas derivadas. Analogamente, o número de unidades do ben dispoñible por unidade de tempo chámase oferta F e, en xeral,

tamén é función de t, de p e das súas derivadas.

Os economistas dinnos que o prezo está determinado pola condición D=F.

Exemplo 8:

Achar a evolución do prezo diario p(t) dun ben, sabendo que p(0)=20 e que a oferta e a demanda son, respectivamente:

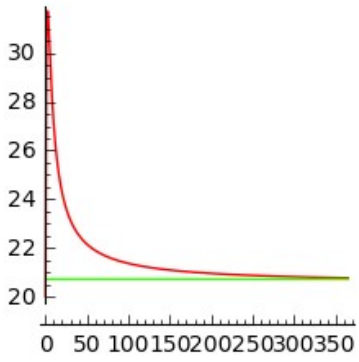
$$F(t,p,p') = 160 - \frac{15}{10+t}p - 4(1+\sqrt{t})p'$$

$$D(t,p,p') = 40 + \frac{20t}{15+2t}p + (1+t^2)p'$$

a. Estabilízase o prezo no tempo?

b. En que día alcanza o prezo máximo?

```
f(t,p)=[(120-((20*t)/(15+2*t)*p+15*p/(10+t)))/(4*(1+sqrt(t))+
(1+t^2))]
CI=vector([20]);I=[0,365];N=24*I[1]
S=rungekutta(f,CI,I,N)
P=[a[0] for a in S]
T=[0,1/24,..,365]
GP=line(zip(T,P),hue=0)
f(x)=S[-1][0]
P=plot(f,(x,0,365),hue=0.3)
show(GP+P,figsize=[2.5,2.5])
```




```
print 'A curva ten unha asíntota horizontal y=',S[-1][0]
print 'O prezo máximo alcánzase ás',maximos(S)[1][0],'horas'
```

```
A curva ten unha asíntota horizontal y= 20.7578849475742
O prezo máximo alcánzase ás 63 horas
```

Exemplo 9:

Achar a evolución do precio diario $p(t)$ dun ben, sabendo que $p(0) = 20$ e $p'(0) = 0$, coas seguintes oferta e demanda

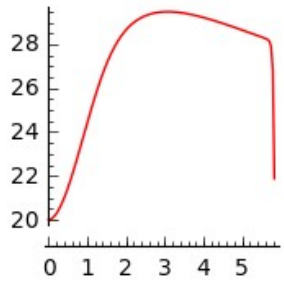
$$F(t, p, p', p'') = 160 - \frac{15}{10+t}p - 4(1+\sqrt{t})p' + \frac{t}{1+t^2}p''$$

$$D(t, p, p', p'') = 40 + \frac{20t+1}{15+2t}p + (1+t^2)p' - \frac{t}{1+t^2}p''$$

a. En que día alcanza o ben o maior prezo?

b. Estabilízase o prezo no tempo?

```
f(t,p,q)=[q,((1+t^2)*(120-((20*t+1)*p/(15+2*t)+15*p/(10+t))-
(4*(1+sqrt(t))+(1+t^2))*q))/(2*(t+3))]
CI=vector([20,0])
I=[0,30]
N=24*I[1]
S=rungekutta(f,CI,I,N)
SV=[a for a in S if a[0]>0]
P=[a[0] for a in S]
T=[0,1/24,..,30]
T=T[:141]
P=P[:141]
GP=line(zip(T,P),hue=0)
show(GP,figsize=[2,2])
```



```
print 'O prezo máximo alcánzase ás', maximos(P)[1][0],'horas'
print 'O prezo cae en picado ao final do quinto día'
```

```
O prezo máximo alcánzase ás 74 horas
O prezo cae en picado ao final do quinto día
```

3.2.5. Circuítos eléctricos

Na Worksheet 1 (ver [1] páx. 207) tratamos as redes eléctricas, con xeradores de tensión ou intensidade e resistencias. Agora consideraremos tamén elementos almacenadores de enerxía, como condensadores e bobinas. Referirémonos a estas redes como circuítos eléctricos. As leis que rexen os circuítos son as xa mencionadas leis de Ohm e de Kirchoff e dúas novas regras para o control dos condensadores e as bobinas. De acordo con [Apuntes de clase do Prof. C.Garrido] denotaremos $u(t)$ á tensión, $q(t)$ á carga, $i(t) = q'(t)$ á intensidade, R á resistencia, C á capacidade, L á inductancia e enunciamos:

Regra 1: $i(t) = C \cdot u'(t) \quad o \quad \frac{q(t)}{C} = u(t)$

Regra 2: $u(t) = L \cdot i'(t)$

Os circuítos cun só elemento almacenador de enerxía (circuítos RC ou RL) dan lugar a ecuacións diferenciais lineais de primeira orde e os circuítos con dous elementos almacenadores de enerxía (circuítos RLC) dan lugar a ecuacións diferenciais lineais de segunda orde.

Exemplo 10:

Un circuíto montado en serie consta dun xerador, unha resistencia $R = 1\Omega$ e un condensador $C = 0.5F$. Supoñemos que a tensión fornecida polo xerador é

$$u(t) = \begin{cases} 20 & \text{se } 0 \leq t \leq 3 \\ 0 & \text{se } t > 3 \end{cases}$$

Calcular a tensión no condensador.

Solución:

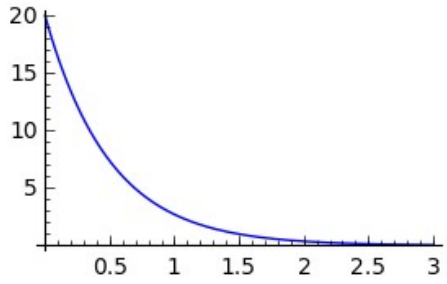
As leis de Kirchoff aseguran que $u(t) = i(t) \cdot R + u_C(t)$ e derivando,

$$0 = R \cdot i'(t) + u_C'(t) = R \cdot i'(t) + \frac{1}{C} \cdot i(t) \quad \Rightarrow \quad i' = -\frac{1}{RC} i$$

Integrando esta ecuación diferencial en [0,3] con i(0)=20 temos

```
var('t')
C=1/2
R=1
A=matrix([(-1/(R*C))])
i0=vector([20])
i(t)=(expm(A,t)*i0)[0]
show(i(t))
I0=plot(i,(t,0,3))
show(I0,figsize=[3,2])
```

$$20\, e^{(-2\, t)}$$

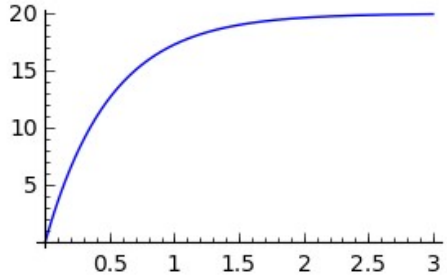


Usando a Regra 1, calculamos a tensión no condensador $uc : [0, 3] \to \mathbb{R}$

```
uc(t)=(1/C)*integrate(i,t,0,t)
show(uc(t))
show(uc(3).n())
U0=plot(uc,(t,0,3))
show(U0,figsize=[3,2])
```

$$-20\, e^{(-2\, t)} + 20$$

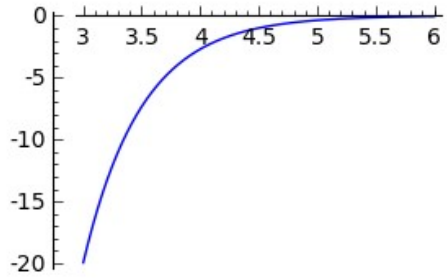
$$19.9504249564667$$



No intervalo $[3,\infty)$, a relación entre a intensidade e a tensión no condensador vén dada por $0=i_1(t)\cdot R+u_{1C}(t)$ e derivando $i_1'=-\frac{1}{RC}i_1$. Integrando esta ecuación diferencial coa condición inicial $i_1(3)=-\frac{u_{1C}(3)}{R}\approx -19.95$

```
i10=vector([-uc(3)])
i1(t)=(expm(A,t-3)*i10)[0]
show(i1(t).full_simplify())
I1=plot(i1,(t,3,6))
show(I1,figsize=[3,2])
```

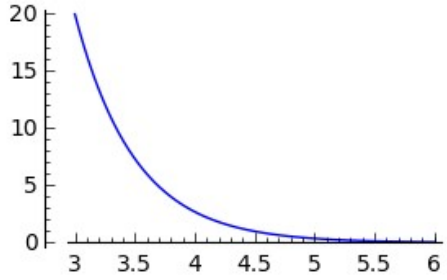
$$-20\left(e^{6}-1\right)e^{(-2\,t)}$$



e usando a Regra 1, calculamos a tensión no condensador no intervalo $[3,\infty)$

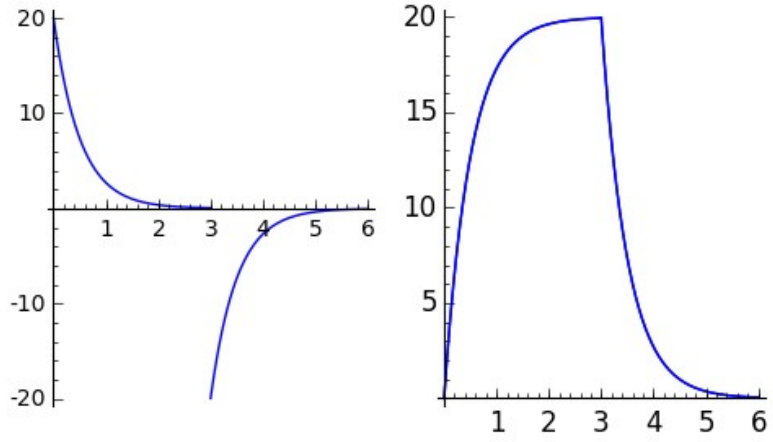
```
u1c(t)=uc(3)+(1/C)*integrate(i1,t,3,t)
show(u1c(t).full_simplify())
U1=plot(u1c,(t,3,6))
show(U1,figsize=[3,2])
```

$$20\left(e^{6}-1\right)e^{(-2\,t)}$$



As gráficas conxuntas da intensidade e a tensión no tempo son

```
show(graphics_array([I0+I1,U0+U1]),figsize=[5,3])
```



Exemplo 11:

Un circuíto montado en serie consta dun xerador, unha resistencia $R = 1\Omega$ e unha bobina $L = 0.5H$. Supoñemos que a tensión fornecida polo xerador é

$$u(t) = \begin{cases} 20 & \text{se } 0 \leq t \leq 3 \\ 0 & \text{si } t > 3 \end{cases}$$

Calcular a intensidade e a tensión na bobina.

Solución:

As leis de Kirchoff aseguran que

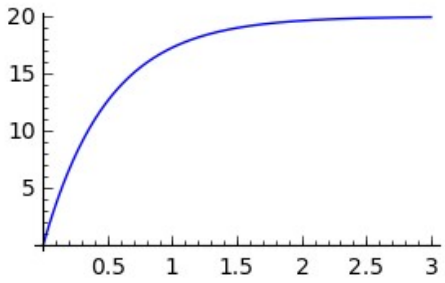
$$u(t) = i(t) \cdot R + u_L(t) = i(t) \cdot R + L \cdot i'(t)$$

Integrando esta ecuación diferencial en [0,3] con i(0)=0 por estar a bobina descargada no momento inicial

```
var('t s')
L=1/2; R=1
u(s)=(1/L)*20
A=matrix([-R/L])
i0=vector([0])
i(t)=(expm(A,t)*i0)[0]+integrate(expm(A,t-s)[0,0]*u(s),s,0,t)
show(i(t))
show(i(3).n())
I0=plot(i,(t,0,3))
show(I0,figsize=[3,2])
```

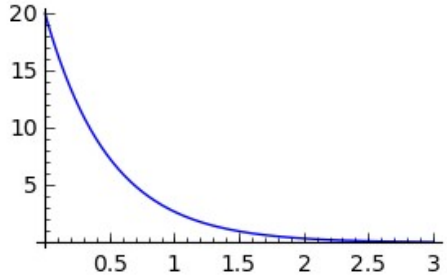
$$-20\,e^{(-2\,t)} + 20$$

$$19.9504249564667$$



e usando a Regra 2, calculamos a tensión na bobina

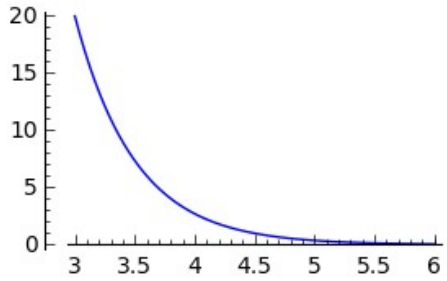
```
uL(t)=L*diff(i(t),t)
U0=plot(uL,(t,0,3))
show(U0,figsize=[3,2])
```



No intervalo $[3,\infty)$, a relación entre a intensidade e a tensión na bobina vén dada por $0 = i_1(t)\cdot R + u_{1L}(t)$ e usando a Regra 2 temos a ecuación diferencial $i_1' = \frac{-R}{L}\cdot i_1$ coa condición inicial $i_1(3) = i(3) \approx -19.95$

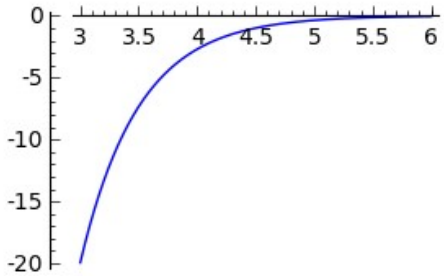
```
i10=vector([i(3)])
i1(t)=(expm(A,t-3)*i10)[0]
show(i1(t).full_simplify())
I1=plot(i1,(t,3,6))
show(I1,figsize=[3,2])
```

$$20\left(e^{6}-1\right)e^{(-2\,t)}$$



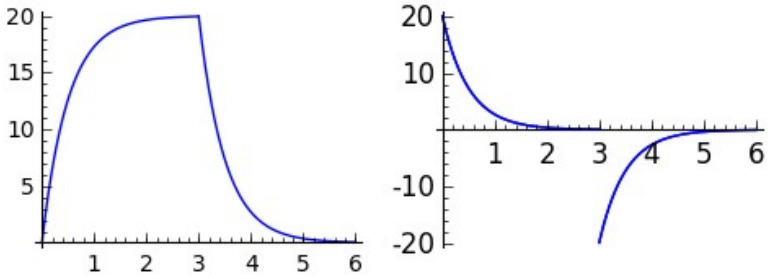
e usando a Regra 2, calculamos a tensión na bobina

```
u1L(t)=L*diff(i1(t),t)
U1=plot(u1L,(t,3,6))
show(U1,figsize=[3,2])
```



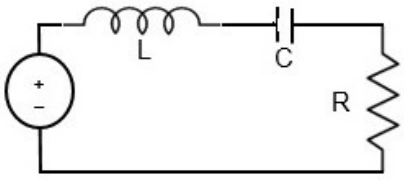
As gráficas conxuntas da intensidade e a tensión no tempo son

```
show(graphics_array([I0+I1,U0+U1]),figsize=[5,2])
```



Exemplo 12:

No seguinte circuíto RLC montado en serie



temos $R = 5\Omega$, $L = 0.8H$ e $C = 0.9F$. Calcular a intensidade e a tensión nas diferentes compoñentes se o xerador fornece unha tensión $u(t) = 2\sin 5t$ e no instante inicial a carga é 2 e a intensidade 1.

Solución:

É claro que

$$u(t) = u_L(t) + u_C(t) + u_R(t) = L \cdot i'(t) + \frac{1}{C} q(t) + R \cdot i(t)$$

que é a ecuación diferencial lineal de segunde orde en q :

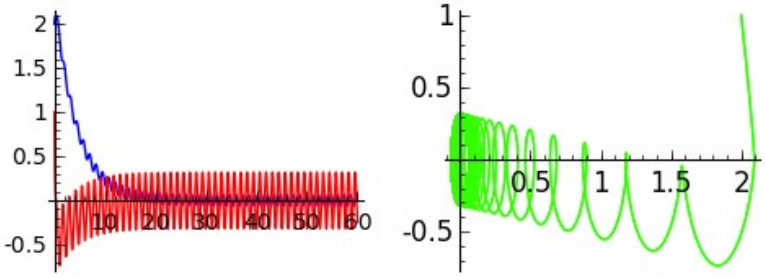
$$q'' + \frac{R}{L} q' + \frac{1}{LC} q = \frac{u}{L}$$

ou o sistema lineal de primera orde en $\begin{pmatrix} q \\ i \end{pmatrix}$

$$\begin{pmatrix} q \\ i \end{pmatrix}' = \begin{pmatrix} 0 & 1 \\ -\frac{1}{LC} & -\frac{R}{L} \end{pmatrix} \begin{pmatrix} q \\ i \end{pmatrix} + \begin{pmatrix} 0 \\ \frac{u}{L} \end{pmatrix}$$

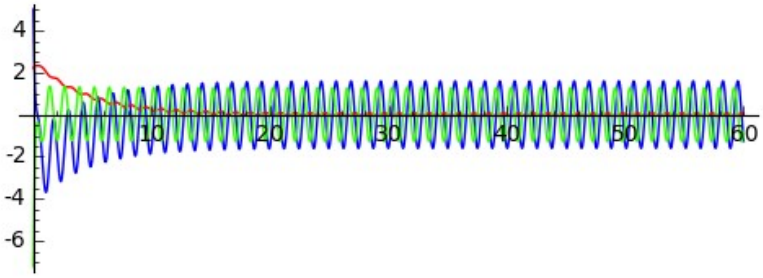
Utilizamos **rungekutta()** para integrala e obtemos as gráficas de $q(t)$, $i(t)$ e a relación entre ambas

```
R=5
C=.9
L=.8
u(t)=2*sin(5*t)
CI=vector([2,1])
f(t,q,i)=[i,-q/(L*C)-i*(R/L)+u(t)/L]
I=[0,60]
N=60*I[1]
CI=vector([2,1])
RK=rungekutta(f,CI,I,N)
T=[0,1/60,..,60]
Q=[a[0] for a in RK]
In=[a[1] for a in RK]
show(graphics_array([line(zip(T,Q))+line(zip(T,In),hue=0),
line(zip(Q,In),hue=.3)]),figsize=[5,2])
```



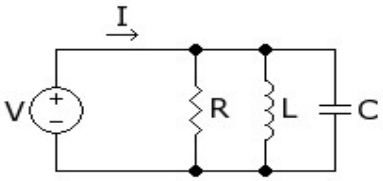
A partir da intensidade discretizada In, obtemos o potencial discretizado na resistencia (azul). A partir da carga discretizada Q, obtemos o potencial discretizado no condensador (vermello) e, restando da discretización de u(t) a suma de ambos, obtemos o potencial discretizado na bobina (verde):

```
UR=[R*a for a in In]
UC=[a/C for a in Q]
UL=[u(T[k])-UR[k]-UC[k] for k in range(len(T))]
show(line(zip(T,UR))+line(zip(T,UC),hue=0)+
line(zip(T,UL),hue=.3),figsize=[5,2])
```



Exemplo 13:

No seguinte circuíto RLC montado en paralelo



temos $R = 5\Omega$, $L = 0.8H$ e $C = 0.9F$. Calcular a intensidade nas diferentes compoñentes se o xerador fornece unha intensidade total $I(t) = 2\sin 5t$ e no instante inicial a carga é 2 e a intensidade 1.

Solución:

É claro que $i_C + i_R + i_L = I$ e, xa que logo, $Cu'(t) + \frac{1}{R}u(t) + \frac{1}{L}\int_0^t u(t)dt = I(t)$. Derivando, obtemos a ecuación de segunda orde

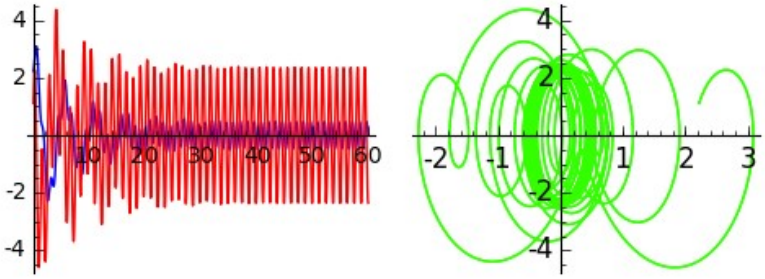
$$Cu'' + \frac{1}{R}u' + \frac{1}{L}u = I'$$

ou o sistema lineal de primera orde en $\begin{pmatrix} u \\ v \end{pmatrix}$

$$\begin{pmatrix} u \\ v \end{pmatrix}' = \begin{pmatrix} 0 & 1 \\ -\frac{1}{LC} & -\frac{1}{RC} \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} + \begin{pmatrix} 0 \\ \frac{I'}{C} \end{pmatrix}$$

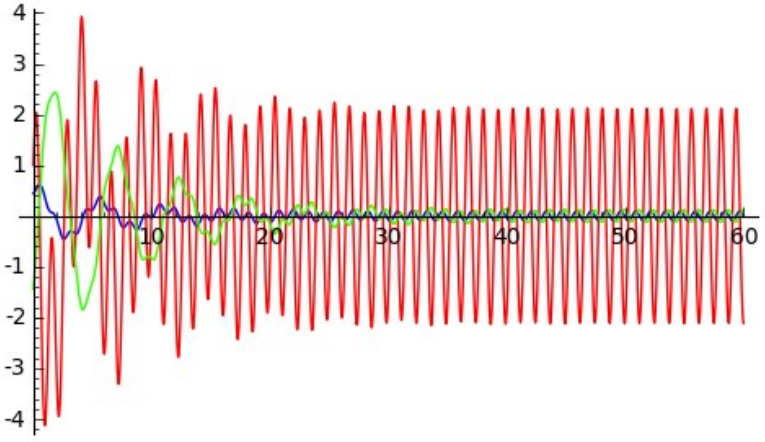
Utilizamos **rungekutta()** para integrar esta ecuación coas condicións iniciais $u(0) = \frac{q(0)}{C} = \frac{20}{9}$ e $v(0) = u'(0) = \frac{i(0)}{C} = \frac{10}{9}$

```
R=5
C=.9
L=.8
I(t)=2*sin(5*t)
I1(t)=diff(I,t)
CI=vector([20/9,10/9])
f(t,u,v)=[v,-u/(L*C)-v/(R*L)+I1(t)/C]
J=[0,60]
N=60*J[1]
CI=vector([20/9,10/9])
RK=rungekutta(f,CI,J,N)
T=[0,1/60,..,60]
U=[a[0] for a in RK]
V=[a[1] for a in RK]
show(graphics_array([line(zip(T,U))+line(zip(T,V),hue=0),
line(zip(U,V),hue=.3)]),figsize=[5,2])
```



A partir da tensión discretizada U, obtemos a intensidade discretizada na resistencia (azul), no condensador (vermello) e na bobina (verde):

```
IR=[a/R for a in U]
IC=[a*C for a in V]
IL=[I(T[k])-IR[k]-IC[k] for k in range(len(T))]
show(line(zip(T,IR))+line(zip(T,IC),hue=0)+
line(zip(T,IL),hue=.3),figsize=[5,3])
```



3.2.6. Reaccións químicas

Nunha reacción química elemental onde os produtos X_0 e X_1 reaccionan dando lugar ao produto X_2, se $x_i(t)$ é a concentración de X_i no instante t, cúmprese a *lei de conservación da materia*:

$$x_i(t) + x_2(t) = C_i \text{ ou } \frac{dx_i}{dt} + \frac{dx_2}{dt} = 0 \text{ para } i = 0, 1.$$

A función $\frac{dx_2}{dt}$ chámase velocidade da reacción e a *lei de acción de masas* asegura que existe unha constante k tal que

$$\frac{dx_2}{dt} = kx_0x_1.$$

Se $x = (x_0, x_1, x_2)$ e $\mathrm{u} = (-1, -1, 1)$, unha reacción química de constante k estará gobernada pola ecuación diferencial $x' = f(t, x)$ onde

$$\begin{aligned} f : [0, \infty) \times \mathbb{R}^3 &\to \quad \mathbb{R}^3 \\ (t, \mathrm{x}) \quad &\mapsto \; kx_0x_1\mathrm{u} \end{aligned}$$

As condicións iniciais fixarán as concentracións $x_i(0)$ para $i = 0, 1, 2$.

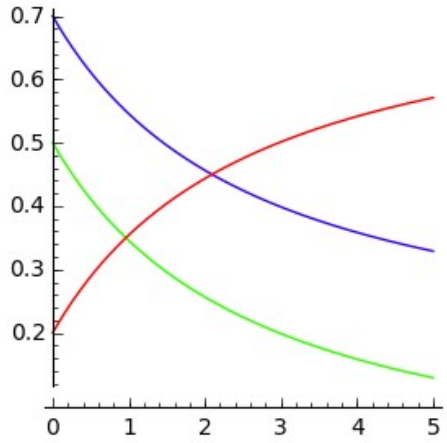
Exemplo 14:

As substancias X_0, X_1, X_2 reaccionan segundo o esquema $X_0 + X_1 \to X_2$ con $k = 0.6$. Se as condicións iniciais son $\mathbf{x}(0) = (0.5, 0.7, 0.2)$, calcular a concentración de cada substancia no intervalo de tempo $[0, 5]$

En que instante coinciden as concentracións de X_0 e X_2?

E as de X_1 e X_2?

```
k=.6; f(t,x0,x1)=[-k*x0*x1,-k*x0*x1,k*x0*x1]
CI=vector([.5,.7,.2]); I=[0,5]; N=300; T=[0,1/60,..,5]
S=rungekutta(f,CI,I,N)
X0=[a[0] for a in S]
X1=[a[1] for a in S]
X2=[a[2] for a in S]
GX0=line(zip(T,X0),hue=0.3)
GX1=line(zip(T,X1),hue=0.7)
GX2=line(zip(T,X2),hue=1)
show(GX0+GX1+GX2,figsize=[3,3])
```




```
X0X2=[abs(X0[n]-X2[n]) for n in range(301)]
print 'X0=X2 aos 58 segundos', '(',T[X0X2.index(min(X0X2))],')'
X1X2=[abs(X2[n]-X1[n]) for n in range(301)]
print 'X1=X2 aos 126 segundos',
'(',T[X1X2.index(min(X1X2))],')'
```

```
X0=X2 aos 58 segundos ( 29/30 )
X1=X2 aos 126 segundos ( 21/10 )
```

Exemplo 15:

As substancias X_0 , X_1 , X_2 , X_3 , X_4 e X_5 reaccionan quimicamente segundo o esquema de Brusselator:

$X_0 \rightarrow X_4$ con $k_1 = 1.2$

$X_1 + X_4 \rightarrow X_5 + X_2$ con $k_2 = 0.8$

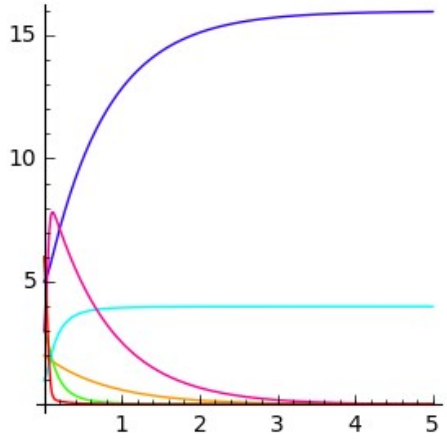
$2X_4 + X_5 \rightarrow 3X_4$ con $k_3 = 0.9$

$X_4 \rightarrow X_3$ con $k_4 = 1.3$

Se as condicións iniciais son $\mathbf{x}(0) = (2, 3, 1, 5, 3, 6)$, calcular a concentración de cada substancia no intervalo de tempo $[0, 5]$.

```
k1=1.2
k2=.8
k3=.9
k4=1.6
```

```
f(t,x0,x1,x2,x3,x4,x5)=[-k1*x0,-k2*x1*x4,k2*x1*x4,k4*x4,k1*x0-
k2*x1*x4+k3*x4^2*x5-k4*x4,k2*x1*x4-k3*x4^2*x5]
CI=vector([2,3,1,5,3,6])
I=[0,5]
N=300
T=[0,1/60,..,5]
S=rungekutta(f,CI,I,N)
X0=[a[0] for a in S]
X1=[a[1] for a in S]
X2=[a[2] for a in S]
X3=[a[3] for a in S]
X4=[a[4] for a in S]
X5=[a[5] for a in S]
GX0=line(zip(T,X0),hue=.1)
GX1=line(zip(T,X1),hue=.3)
GX2=line(zip(T,X2),hue=.5)
GX3=line(zip(T,X3),hue=.7)
GX4=line(zip(T,X4),hue=.9)
GX5=line(zip(T,X5),hue=1)
show(GX0+GX1+GX2+GX3+GX4+GX5,figsize=[3,3])
```



3.2.7. Lotka-Volterra

En 1925 Lotka e Volterra modelaron a evolución das poboacións $x(t)$ e $y(t)$ de herbíboros e carnívoros convivindo nun hábitat ideal, por un sistema de ecuacións diferenciais do tipo

$$\begin{cases} x' = \alpha x - \beta\, x\, y \\ y' = -\gamma y + \delta\, x\, y \end{cases} \quad \text{con} \quad \alpha, \beta, \gamma, \delta \in \mathbb{R}^+$$

Como vemos, a evolución dos herbívoros segue unha lei de Malthus lastrada polos seus encontros cos carnívoros e a dos carnívoros, unha lei de Malthus de constante negativa porque compiten entre si, favorecida polos seus encontros cos herbívoros.

Posteriormente, estudáronse evolucións de ecosistemas considerando crecementos loxísticos das poboacións de herbívoros, é dicir, en hábitats con capacidade de carga limitada.

En 1967, Goodwin suscitou ecuacións diferenciais similares para estudar a evolución de ciclos económicos ligando a participación do traballo na produción, $u(t)$, e a taxa de emprego, $v(t)$, mediante ecuacións diferenciais do tipo:

$$\begin{cases} v' = (\frac{1}{\sigma} - (\alpha + \beta))v - \frac{1}{\sigma} vu \\ u' = -(\alpha + \gamma)u + \rho vu \end{cases}$$

Exemplo 16:

Nun prado con posibilidade ilimitada de produción de herba, sóltanse no instante inicial, 104 cordeiros e 25 lobos. Estudar a evolución das dúas especies supoñendo que seguen unha ecuación de Lotka e Volterra con

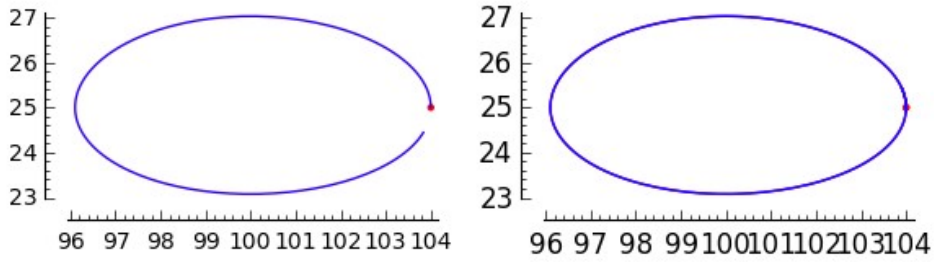
$$(\alpha, \beta, \gamma, \delta) = (0.25, 0.01, 1, 0.01)$$

Solución:

Resolvendo coa función **rungekutta(),** sucesivamente en intervalos de 12 e trece anos, vemos que a solución é cíclica e o seu período é menor que trece anos

```
f(t,x,y)=[.25*x-.01*x*y,-y+.01*x*y]
CI=vector([104,25])
I=[0,12]
```

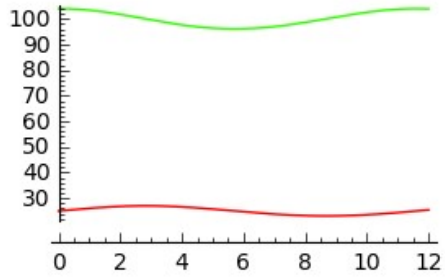
```
N=12*I[1]
S12=rungekutta(f,CI,I,N)
D12=line(S12,hue=.7,aspect_ratio=1)+point(CI,hue=0)
I=[0,13]
S13=rungekutta(f,CI,I,N)
D13=line(S13,hue=.7,aspect_ratio=1)+point(CI,hue=0)
show(graphics_array([D12,D13]),figsize=[6,2])
```



Tamén podemos estudar a evolución de cada unha das especies en función do tempo e calcular os seus máximos e mínimos e os instantes en que se acadan:

```
T=[0,1/12,..,13]
X=[a[0] for a in S13]
GX=line(zip(T,X),hue=.3)
Y=[a[1] for a in S13]
GY=line(zip(T,Y),hue=0)
show(GX+GY,figsize=[3,2])
print 'Máximo número de cordeiros aos', X.index(max(X)),'meses'
print 'Mínimo número de cordeiros aos', X.index(min(X)), 'meses'
print 'Máximo número de lobos aos', Y.index(max(Y)),'meses'
print 'Mínimo número de lobos aos', Y.index(min(Y)), 'meses'
```

```
Máximo número de cordeiros aos 0 meses
Mínimo número de cordeiros aos 68 meses
Máximo número de lobos aos 34 meses
Mínimo número de lobos aos 104 meses
```



Exemplo 17:

Diremos que o ecosistema do Exemplo 16 é sostible se tanto o número de lobos como o de cordeiros é igual ou maior que dous en todo momento. Determinar o conxunto de condicións iniciais que o fan sostible, o que denominaremos zona de sostibilidade.

Solución:

A ecuación do ecosistema é a autónoma estudada no Exemplo 4 da primeira sección desta Worksheet. Sabemos que ten os puntos estacionarios $(0,0)$ e $(100,24)$. Na contorna do primeiro, a aproximación lineal da ecuación prognostica insostibilidade por desaparición dos lobos mentres que na contorna do segundo, temos un sistema sostible.

Resolvemos con **rungekutta(**f, CI, I, N**)** o sistema con condicións iniciais na recta que une os dous puntos estacionarios

$$\begin{cases} x = 100 \cdot t \\ y = 24 \cdot t \end{cases}$$

e mediante un proceso de tipo bisección, atopamos que o paso de non sostibilidade a sostibilidade prodúcese para un $t \in [0.4000, 0.4001]$

```
f(t,x,y)=[.25*x-.01*x*y,-y+.01*x*y]
I=[0,15]
N=12*I[1]; LGS=[]; LGNS=[]
for k in [0.40,0.4001]:
    CI=vector([100*k,24*k])
    S=rungekutta(f,CI,I,N)
    C=[a[0] for a in S]
    L=[a[1] for a in S]
    if min(C)<2 or min(L)<2:
        print 'ecosistema non sostible con'
        print 'CI=',CI
        LGNS.append(line(S,hue=k))
    else:
        print 'ecosistema sostible con'
        print 'CI=',CI
        LGS.append(line(S,hue=2*k))
TE=text('zona de sostibilidade',[120,40],fontsize=6)
show(sum(LGS)+TE,figsize=[2,2])
```

```
ecosistema non sostible con
CI= (40.0000000000000, 9.60000000000000)
ecosistema sostible con
CI= (40.0100000000000, 9.60240000000000)
```

A zona de sostibilidade é a rexión pechada cuxa fronteira é a curva púrpura da gráfica anterior e a de non sostibilidade, a súa complementaria.

Exemplo 18:

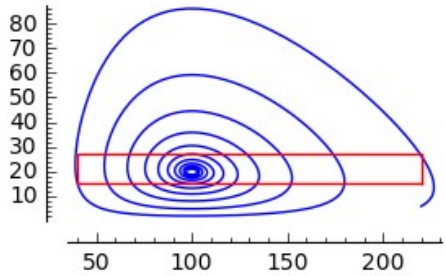
Nun prado con capacidade de carga de 500 cordeiros, sóltanse no instante inicial 220 cordeiros e 6 lobos. Estudar a evolución das especies se a ecuación que a rexe é

$$\begin{cases} x' = 0.0005x(500-x) - 0.01xy \\ y' = -y + 0.01xy \end{cases}$$

Probar que o rectángulo $[40, 220] \times [15, 27]$ está contido na zona de estabilidade da ecuación.

```
f(t,x,y)=[.0005*x*(500-x)-.01*x*y,-y+.01*x*y]
I=[0,150];N=12*I[1]
CI=vector([220,6])
S=rungekutta(f,CI,I,N)
R=line([[220,15],[220,27],[40,27],[40,15],[220,15]],hue=0)
show(line(S)+R,figsize=[3,2])
print 'O número mínimo de lobos é', min([a[1] for a in S])
print 'O número mínimo de cordeiros é', min([a[0] for a in S])
```

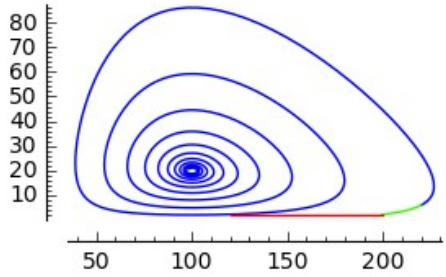
```
O número mínimo de lobos é 2.04420786853889
O número mínimo de cordeiros é 38.9558337158167
```



Como a solución de condición inicial $(220, 6)$ ten todos os seus puntos de abscisas e ordenadas maiores que 2, vemos que o rectángulo está contido na zona de sostibilidade do ecosistema. Todas as solucións con condición inicial nel tenden co tempo ao punto $(100, 20)$, que é un atractor do sistema.

Para estender a zona de sostibilidade do sistema, estudaremos a extensión da solución de condición inicial (220,6) a tempos negativos, tramo verde, e pecharemos a curva dentro da zona de sostibilidade, segmento vermello.

```
IN=[0,-50]
NN=-12*IN[1]
CI=vector([220,6])
SN=rungekutta(f,CI,IN,NN)
for k in range(NN):
    if SN[k][1]>2:
        Ik=k
    else:
        break
LSN=SN[0:Ik]
show(line(S)+line(LSN,hue=.3)+
line([[120,2.3],[200,2.3]],hue=0),figsize=[3,2])
```



Exemplo 19:

Un ecosistema de tres especies está rexido polo sistema de ecuacións diferenciais

$$\begin{cases} x' = 0.25x + 0.01\,x\,y - 0.01\,x\,z \\ y' = 0.3\,y + 0.001\,x\,y - .1\,y\,z \\ z' = -\,z + 0.01\,x\,z + 0.01\,y\,z \end{cases}$$

No instante inicial o número de individuos de cada especie é $(x_o, y_o, z_o) = (100, 80, 20)$:

a. Mostrar a evolución conxunta das tres especies en 30 anos.

b. Cal das especies se extingue e en canto tempo?

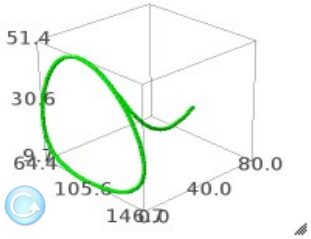
c. Probar que as especies restantes constitúen un sistema ciclicamente estable e calcular o seu período.

Solución:

A segunda especie queda con menos dun individuo entre os meses 18 e 19:

```
f(t,x0,x1,x2)=
[.25*x0+.01*x0*x1-.01*x0*x2,.3*x1+.001*x0*x1-.1*x1*x2,-
x2+.01*x0*x2+.01*x1*x2]
CI=vector([100,80,20])
I=[0,30]
S=rungekutta(f,CI,I,360)
print 'S[18]=',S[18]
print 'S[19]=',S[19]
g=line3d(S,rgbcolor=(0,1,0),thickness=5)
show(g,figsize=[2,2,2])
```

```
S[18]= (128.395227091330, 1.18426921299880, 43.9923062525366)
S[19]= (126.429092626355, 0.846836106738938, 45.0469548856762)
```

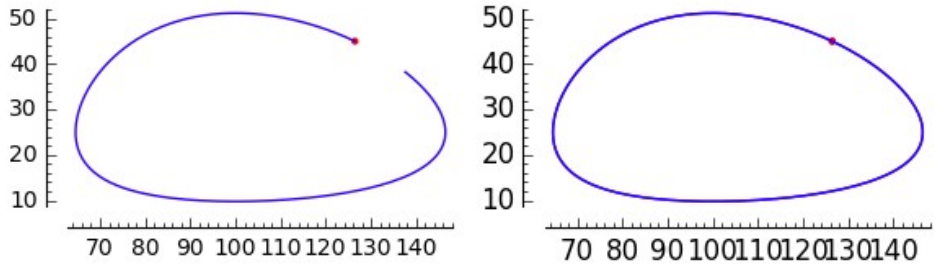


Cando se extinga a segunda especie a ecuación que rexe o ecosistema das dúas restantes é:

$$\begin{cases} x' = 0.25x - 0.01\,x\,z \\ z' = -z + 0.01\,x\,z \end{cases}$$

e as condicións iniciais serán $(S[19][0], S[19][2])$. Como se ve nas seguintes gráficas o período é de 13 anos

```
f(t,x0,x2)=[.25*x0-.01*x0*x2,-x2+.01*x0*x2]
CI=vector([S[19][0],S[19][2]])
I=[0,12.5]
N=12*I[1]
S12=rungekutta(f,CI,I,N)
D12=line(S12,hue=.7,aspect_ratio=1)+point(CI,hue=0)
I=[0,13]
S13=rungekutta(f,CI,I,N)
D13=line(S13,hue=.7,aspect_ratio=1)+point(CI,hue=0)
show(graphics_array([D12,D13]),figsize=[6,2])
```



Exemplo 20:

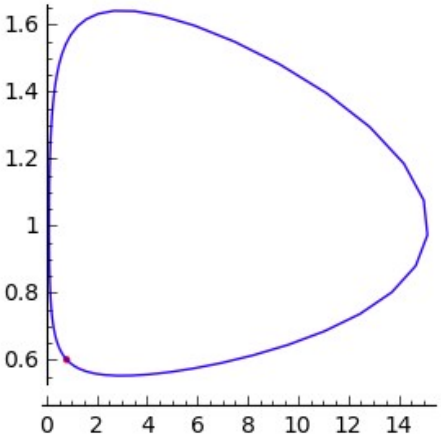
Estudar o ciclo económico de Goodwin dado polo sistema de ecuacións diferenciais:

$$\begin{cases} v' = 5v - 5vu \\ u' = -0.3001u + 0.1vu \end{cases}$$

onde $v(t)$ é a taxa de emprego, $u(t)$ a participación do traballo na produción e

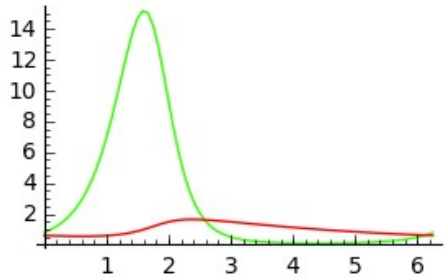
as condicións iniciais son $u(0) = 0,8$ e $v(0) = 0,6$.

```
f(t,v,u)=[5*v-5*v*u,-.3001*u+.1*v*u]
CI=vector([.8,.6])
I=[0,6.3]
N=12*I[1]
S=rungekutta(f,CI,I,N)
show(line(S,hue=.7)+point(CI,hue=0),figsize=[3,3])
```




```
T=[0,1/12,..,6.3]
X=[a[0] for a in S]
GX=line(zip(T,X),hue=.3)
Y=[a[1] for a in S]
GY=line(zip(T,Y),hue=0)
show(GX+GY,figsize=[3,2])

print 'Máxima taxa de emprego aos', X.index(max(X)),'meses'
print 'Mínima taxa de emprego aos', X.index(min(X)), 'meses'
print 'Máxima participación do traballo aos',
Y.index(max(Y)),'meses'
print 'Mínima participación do traballo aos', Y.index(min(Y)),
'meses'
```

```
Máxima taxa de emprego aos 19 meses
Mínima taxa de emprego aos 53 meses
Máxima participación do traballo aos 29 meses
Mínima participación do traballo aos 7 meses
```



3.2.8. Epidemias

Nunha poboación na que se produce unha epidemia, debemos considerar os sans $x(t)$, os enfermos $y(t)$, os inmunizados $z(t)$ e, no caso de epidemias graves, os mortos $m(t)$. Como nos apartados anteriores, a variación dos enfermos crece co seu contacto cos sans e diminúe pasando a inmunizados ou mortos como podemos ver nos seguintes exemplos.

Exemplo 21:

Nunha poboación de 1000 habitantes declárase unha epidemia leve, sen mortes nin nacementos e inmunización permanente, gobernada polas ecuacións:

$$\begin{cases} x' = -0.0004xy \\ y' = 0.0004xy - 0.2y \\ z' = 0.2y \end{cases}$$

onde $x(t)$, $y(t)$ e $z(t)$ é o número de sans, enfermos e inmunizados no día t.

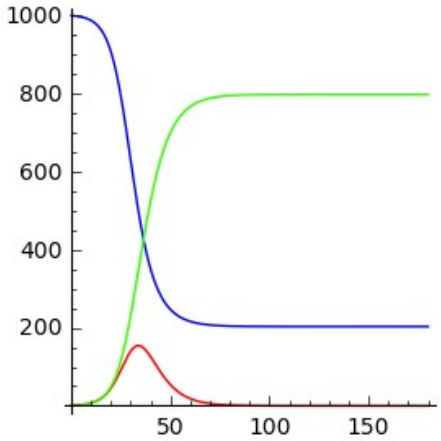
a. Canto dura o brote epidémico?

b. Que día haberá maior afluencia no servizo de urxencias?

c. Que porcentaxe da poboación enferma?

```
f(t,x,y,z)=[-.0004*x*y,.0004*x*y-.2*y,.2*y]
CI=vector([999,1,0])
I=[0,180]
N=24*I[1]
```

```
T=[0,1/24,..,180]
S=rungekutta(f,CI,I,N)
X=[a[0] for a in S]
Y=[a[1] for a in S]
Z=[a[2] for a in S]
show(line(zip(T,X))+line(zip(T,Y),hue=0)+
line(zip(T,Z),hue=.3),figsize=[3,3])
```




```
D=len([a for a in Y if a>1])/24.n()
print 'a.A epidemia dura', floor(D),'días e',ceil((D-
floor(D))*24),'horas'
M=maximos(Y)[1][0]/24.n()
print 'b.A maior afluencia a urxencias ten lugar o día',
floor(M),'ás',ceil((M-floor(M))*24),'horas'
m=min(X)
print 'c.Enferma o', round((1000-m)/10,2),'% da poboación'
```

```
a.A epidemia dura 84 días e 17 horas
b.A maior afluencia a urxencias ten lugar o día 33 ás 19 horas
c.Enferma o 79.72 % da poboación
```

Exemplo 22:

Nunha poboación de 1000 habitantes, declárase unha epidemia leve, sen mortes nin nacementos e inmunización temporal, gobernada polas ecuacións:

$$\begin{cases} x' = -0.0004xy + 0.001z \\ y' = 0.0004xy - 0.2y \\ z' = 0.2y - 0.001z \end{cases}$$

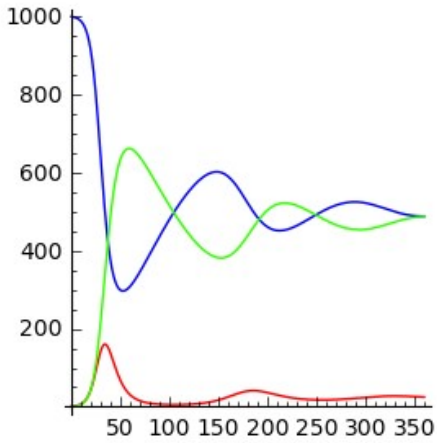
onde $x(t)$, $y(t)$ e $z(t)$ é o número de sans, enfermos e inmunizados no día t.

a. Convértese nunha enfermidade endémica?

b. Que día do primeiro ano hai maior afluencia ao servizo de urxencias?

c. Que día do primeiro ano se produce a segunda maior afluencia ao servizo de urxencias?

```
f(t,x,y,z)=[-.0004*x*y+.01*z,.0004*x*y-.2*y,.2*y-.01*z]
CI=vector([999,1,0]);I=[0,360]
N=24*I[1];T=[0,1/24,..,360]
S=rungekutta(f,CI,I,N)
X=Round([a[0] for a in S],5)
Y=Round([a[1] for a in S],5)
Z=Round([a[2] for a in S],5)
show(line(zip(T,X))+line(zip(T,Y),hue=0)+
line(zip(T,Z),hue=.3),figsize=[3,3])
```




```
print 'a.Ao cabo dun ano hai', Y[-1],'enfermos'
M=maximos(Y)[1][0]/24.n()
print 'b.A maior afluencia a urxencias ten lugar o día',
floor(M),'ás',ceil((M-floor(M))*24),'h.'
M1=100+maximos(Y[100*24:])[1][0]/24.n()
print 'c.O maior rebrote ten lugar o día',
floor(M1),'ás',ceil((M1-floor(M1))*24),'h.'
```

```
a.Ao cabo dun ano hai 24.82441 enfermos
b.A maior afluencia a urxencias ten lugar o día 34 ás 6 h.
c.O maior rebrote ten lugar o día 185 ás 11 h.
```

Exemplo 23:

Nunha poboación de 1000 habitantes, declárase unha epidemia grave, con mortes e inmunización temporal, gobernada por as ecuacións:

$$\begin{cases} x' = -0.0004xy + 0.001z \\ y' = 0.0004xy - 0.208y \\ z' = 0.2y - 0.001z \\ m' = 0.008y \end{cases}$$

onde $x(t)$, $y(t)$, $z(t)$ e $m(t)$ é o número de sans, enfermos, inmunizados e mortos pola enfermidade no día t.

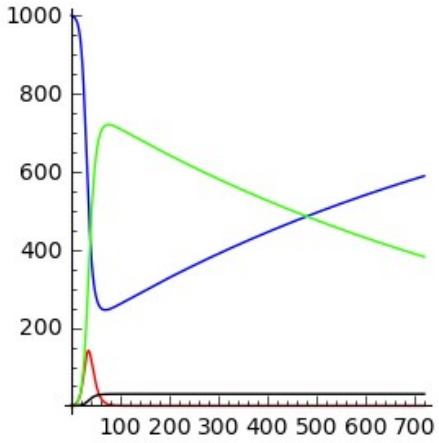
a. Canto durou a epidemia?

b. Que día houbo maior afluencia ao servizo de urxencias?

c. Cantas mortes produciu a epidemia ?

d. Entre que días se produciron as mortes?

```
f(t,x,y,z,m)=
[-.0004*x*y+.001*z,.0004*x*y-.208*y,.2*y-.001*z,.008*y]
CI=vector([999,1,0,0])
I=[0,720]
N=24*I[1]
T=[0,1/24,..,720]
S=rungekutta(f,CI,I,N)
X=Round([a[0] for a in S],5)
Y=Round([a[1] for a in S],5)
Z=Round([a[2] for a in S],5)
M=Round([a[3] for a in S],5)
show(line(zip(T,X))+line(zip(T,Y),hue=0)+
line(zip(T,Z),hue=.3)+line(zip(T,M),color='black'),figsize=
[3,3])
```




```
D=len([a for a in Y if a>1])/24.n()
print 'a.A epidemia durou', floor(D),'días e',ceil((D-
floor(D))*24),'horas'
M1=maximos(Y)[1][0]/24.n()
print 'b.A maior afluencia a urxencias tuvo lugar o día',
floor(M1),'ás',ceil((M1-floor(M1))*24),'horas'
print 'c.A epidemia causou', floor(M[-1]),'mortes'
EM=len([m for m in M if m<1])/24
TM=len([m for m in M if m<30])/24
print 'd.As mortes producíronse entre o día ',floor(EM), 'e o
día ', ceil(TM)
```

```
a.A epidemia durou 87 días e 11 horas
b.A maior afluencia a urxencias tuvo lugar o día 34 ás 15 hora
c.A epidemia causou 30 mortes
d.As mortes producíronse entre o día  17 e o día  79
```

3.3. Problemas de Cauchy en Xeometría

3.3.1. Circuítos automobilísticos

Exemplo 1:

A posición $x(t)$ e a velocidade $v(t)$ dun móbil nun circuito están ligadas pola ecuación $v(t) = A(t) \cdot x(t)$ sendo

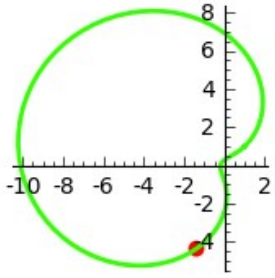
$$A(t) = \begin{pmatrix} \cos\frac{t}{2} & -\sin\frac{t}{2} \\ \sin\frac{t}{2} & \cos\frac{t}{2} \end{pmatrix}$$

a. Achar a traxectoria no período $[0, 4\pi]$ sabendo que $x(0) = (1, 1)$.

b. En que instante está o móbil máis lonxe do punto de saída?

c. Cal é a máxima velocidade que acada o móbil?

```
f(t,x0,x1)=[x0*cos(t/2)-x1*sin(t/2),x0*sin(t/2)+x1*cos(t/2)]
CI=vector([1,1])
I=[0,4*pi]
S=rungekutta(f,CI,I,200)
g=line(S,hue=.3,thickness=2)
movil=[]
for k in srange(0,200,4):
    movil.append(g+point(S[k],hue=0,pointsize=50))
a=animate(movil)
#a.show(delay=10)
show(a[20],figsize=[2,2])
```



Exemplo 2:

A posición $x(t)$ e a velocidade $v(t)$ dun móbil nun circuíto están ligadas pola ecuación $v(t) = A(t) \cdot x(t)$ sendo

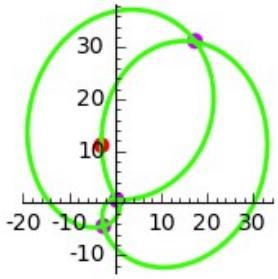
$$A(t) = \begin{pmatrix} \cos\frac{t}{2} & -\sin\frac{t}{4} \\ \sin\frac{t}{4} & \cos\frac{t}{2} \end{pmatrix}$$

a. Cal é o período do movemento?

b. Achar a traxectoria nun período sabendo que $x(0) = (5, 1)$ e marcar os puntos dobres do circuíto.

```
f(t,x0,x1)=[x0*cos(t/2)-x1*sin(t/4),x0*sin(t/4)+x1*cos(t/2)]
```

```
CI=vector([5,1])
I=[0,8*pi]
S=rungekutta(f,CI,I,300)
g=line(S,hue=.3,thickness=2)+point([S[32],S[75],S[117]],hue=.8,poin
movil=[]
for k in srange(0,300,4):
    movil.append(g+point(S[k],hue=0,pointsize=50))
a=animate(movil)
#a.show(delay=10)
show(a[40], figsize=[2,2])
```



3.3.2. Atractores estraños

Exemplo 3: Lorenz

A posición $\mathbf{x}$ e a velocidade $\mathbf{x}'$ dunha partícula en $\mathbb{R}^3$ están ligadas pola ecuación de Lorenz

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix}' = \begin{pmatrix} -10 & 10 & 0 \\ 28 & -1 & -x \\ 0 & x & -\frac{8}{3} \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

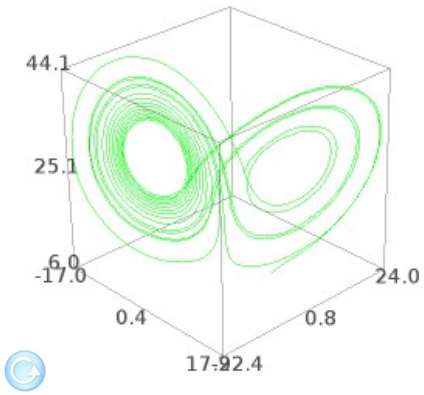
a. Representar a traxectoria de condición inicial $\mathbf{x}(0) = (6,6,6)$ no intervalo $[0,15]$.

b. Animar o movemento da partícula.

```
FL(t,x,y,z)=[10*(-x+y),28*x-y-x*z,-8*z/3+x*y]
print
CI=vector([6,6,6])
I=[0,15]
N=600
```

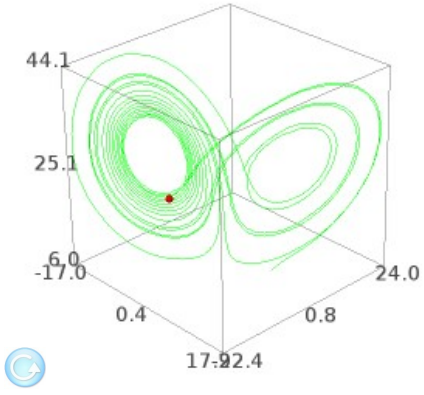
```
SL=rungekutta(FL,CI,I,N)
gL=line3d(SL,rgbcolor=(0,1,0),thickness=1)
print 'a.'
show(gL,figsize=[3,3,3])
```

a.




```
print 'b.'
G2=[]
for a in SL:
    G2.append(gL+point3d(a,rgbcolor=(1,0,0),size=10))
AG2=animate(G2)
#show(AG2)
show(AG2[50],figsize=[3,3,3])
```

b.



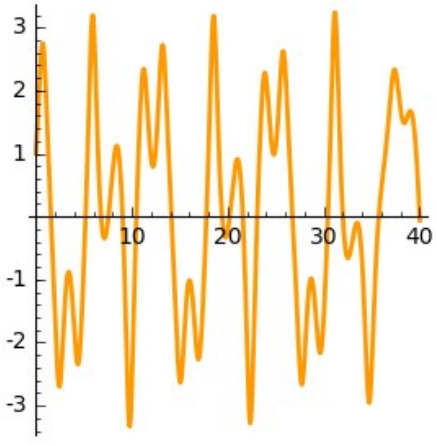
Exemplo 4: Ueda

Sexa a ecuación diferencial $x'' + 2\gamma x' + x^3 = k\cos(t)$ con $\gamma = 0.025$ e $k = 7.5$.

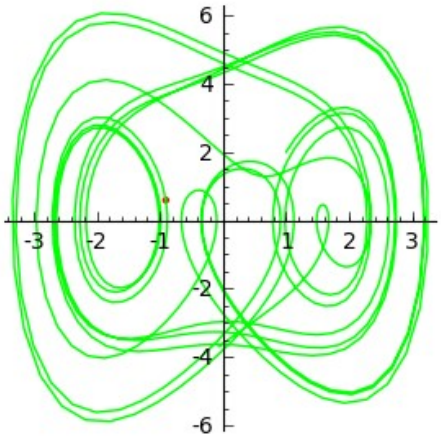
a. Representar a solución de condicións iniciais $x(0) = 1,\ x'(0) = 2$ no intervalo de tempo $[0, 40]$.

b. Animar a gráfica da solución no espazo de fases.

```
Fu(t,x,y)=[y,-2*0.025*y-x^3+7.5*cos(t)]
CI=vector([1,2])
I=[0,40]
N=600
S=rungekutta(Fu,CI,I,N)
g=line(S,rgbcolor=(0,1,0),thickness=1)
TI=[0,4/60,..,40]
XT=[a[0] for a in S]
g2=line(zip(TI,XT),hue=.1,thickness=2)
show(g2,figsize=[3,3])
```




```
G=[]
for a in S:
    G.append(g+point2d(a,rgbcolor=(1,0,0),pointsize=10))
AG=animate(G)
#show(AG)
show(AG[50],figsize=[3,3])
```



3.3.3. Catenaria

A forma f(x) que toma un anaco de corda homoxénea e flexible de densidade δ kg/m, cando está en equilibrio suspendida entre os puntos $(x_1, f(x_1))$ e $(x_2, f(x_2))$ baixo a acción da gravidade e dunha tensión horizontal T en cada extremo, obedece, segundo Bernoulli, á ecuación diferencial:

$$f'' = \frac{\delta}{T}\sqrt{1+f'^2}$$

Exemplo 5:

Se δ= 1, $x_1 = 0$, $f(x_1) = 5$, $f'(x_1) = -\frac{1}{3}$ e $x_2 = 20$:

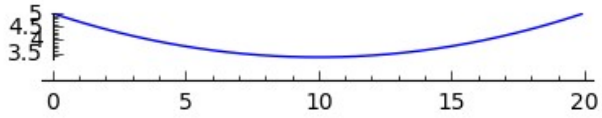
a. Cal debe ser a tensión da corda para que $f(x_2) = 5$?

b. Cal é a lonxitude do anaco de corda?

```
d=1
T=30.5
f(t,x0,x1)=[x1,(d/T)*sqrt(1+x1^2)]
CI=vector([5,-1/3])
R=rungekutta(f,CI,[0,20],200)
X=[0,.1..20]
Y=[R[k][0] for k in range(201)]
show(line([(X[k],Y[k]) for k in
range(200)],aspect_ratio=1),figsize=[4,4])
```

```
print 'a. Con T=',T,'temos f(20)=',R[200][0]
Z=[X[k]+I*Y[k].n() for k in range(201)]
D=[abs(Z[k]-Z[k+1]) for k in range(200)]
print 'b. Lonxitude da corda',sum(D)
```

```
a. Con T= 30.5000000000000 temos f(20)= 5.00852488530680
b. Lonxitude da corda 20.3602505038298
```



3.3.4. Osciladores

Un campo de forzas lineal $L:\mathbb{R}^3\to\mathbb{R}^3$ vén dado por unha matriz $M\in\mathfrak{M}_{(3,3)}(\mathbb{R})$. Dúas posibilidades caben: ou M ten tres valores propios reais, ou M ten un valor propio real e dous imaxinarios conxugados. No primeiro caso, existe unha base na que podemos representar L como unha matriz diagonal $D=\begin{pmatrix}k_1&0&0\\0&k_2&0\\0&0&k_3\end{pmatrix}$.

No segundo caso, todo o máis que podemos aspirar é a representar L por unha matriz real $ND=\begin{pmatrix}k&0&0\\0&a&-b\\0&b&a\end{pmatrix}$ con $b\neq 0$. Neste caso, o rotacional de L é $(2b,0,0)$ e L non pode ser un campo conservativo.

Un oscilador é un campo de forzas lineal conservativo baixo cuxa acción unha masa puntual realiza un movemento acoutado.

Se D é a matriz diagonal que o representa, unha masa m baixo a súa acción no baleiro, seguirá unha traxectoria $\mathbf{x}(t)$, solución da ecuación de Newton $D\mathbf{x}=m\mathbf{x}''$. As súas coordenadas deben cumprir $x_i''=\frac{k_i}{m}x_i$ e serán acoutadas se e só se $k_i<0\quad\forall i=1,2,3$

No que segue, identificaremos cada oscilador cunha terna de números reais negativos. Esencialmente, preséntanse tres casos: que os números sexan todos iguais, que sexan proporcionais a cadrados perfectos ou calquera outro.

Exemplo 6:

Filmar o movemento no baleiro dunha partícula de masa 10 e $CI =$**vector(**[1,7,3,1,1,2]**)**, baixo a acción dos osciladores:

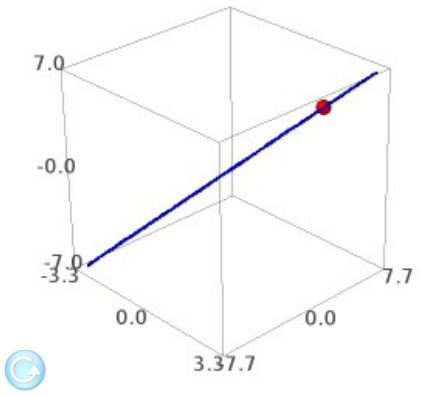
a. $D1 = [-1, -1, -1]$

b. $D2 = [-1, -4, -9]$

c. $D3 = [-1, -2, -3]$

```
print 'a.'
[k1,k2,k3]=[-1,-1,-1]
f(t,x1,x2,x3,v1,v2,v3)=[v1,v2,v3,.1*(k1*x1),.1*(k2*x2),.1*
(k3*x3)]
CI=vector([1,7,3,1,1,2])
I=[0,19.9]; N=6*I[1];T=[0,1/6,..,I[1]]
S=rungekutta(f,CI,I,N)
g=line3d([a[0:3] for a in S],color='blue',thickness=.5)
particula=[]
for k in range(len(T)):
    particula.append(g+point3d(S[k][0:3],color='red',size=20))
a=animate(particula)
show(a[0],figsize=[3,3,3]); #a.show(delay=10)
```

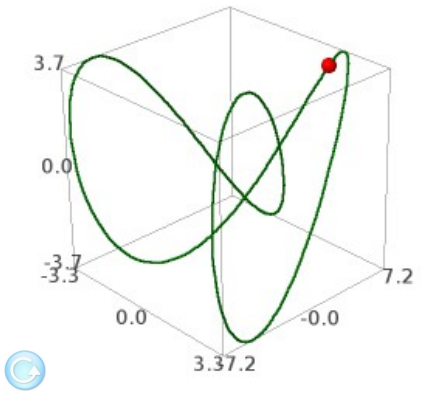
a.




```
print 'b.'
[k1,k2,k3]=[-1,-4,-9]
f(t,x1,x2,x3,v1,v2,v3)=[v1,v2,v3,.1*(k1*x1),.1*(k2*x2),.1*
```

```
(k3*x3)]
CI=vector([1,7,3,1,1,2])
I=[0,19.9]
N=6*I[1]
T=[0,1/6,..,I[1]]
S=rungekutta(f,CI,I,N)
g=line3d([a[0:3] for a in S],color='green',thickness=.5)
particula=[]
for k in range(len(T)):
    particula.append(g+point3d(S[k][0:3],color='red',size=20))
a=animate(particula)
show(a[0],figsize=[3,3,3])
#a.show(delay=10)
```

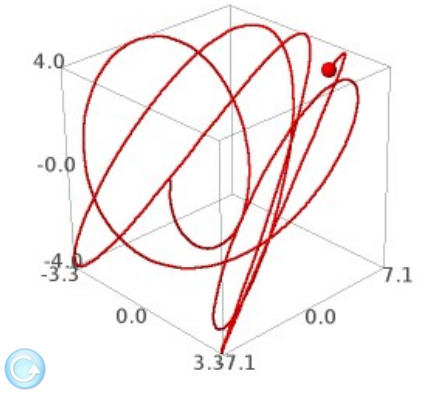
b.




```
print 'c.'; [k1,k2,k3]=[-1,-5,-6]
f(t,x1,x2,x3,v1,v2,v3)=[v1,v2,v3,.1*(k1*x1),.1*(k2*x2),.1*
(k3*x3)]
CI=vector([1,7,3,1,1,2])
I=[0,40]
N=6*I[1]
T=[0,1/6,..,I[1]]
S=rungekutta(f,CI,I,N)
g=line3d([a[0:3] for a in S],color='red',thickness=.5)
particula=[]
for k in range(len(T)):
    particula.append(g+point3d(S[k][0:3],color='red',size=20))
a=animate(particula)
show(a[0],figsize=[3,3,3])
```

```
#a.show(delay=10)
```

c.



Exemplo 7:

Filmar o movemento nun medio viscoso de coeficiente b=.3, dunha partícula de masa 10 e $CI = [1, 7, 3, 1, 1, 2]$, baixo a acción dos osciladores:

a. $D1 = [-1, -1, -1]$

b. $D2 = [-1, -4, -9]$

c. $D3 = [-1, -2, -3]$

Solución:

A ecuación diferencial é agora $D\mathbf{x} - b\mathbf{x}' = m\mathbf{x}''$ ou, equivalentemente,

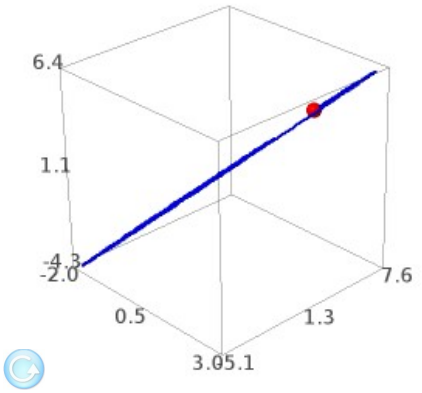
$$\begin{pmatrix} \mathbf{x} \\ \mathbf{v} \end{pmatrix}' = \begin{pmatrix} 0 & I \\ \frac{1}{m}D & -\frac{b}{m}I \end{pmatrix} \cdot \begin{pmatrix} \mathbf{x} \\ \mathbf{v} \end{pmatrix}$$

Aplicaremos **rungekutta(**f, CI, I, N**)** á función

$$f(t, x_1, x_2, x_3, v_1, v_2, v_3) = [v_1, v_2, v_3, \frac{k_1}{m}x_1 - \frac{b}{m}v_1, \frac{k_2}{m}x_2 - \frac{b}{m}v_2, \frac{k_3}{m}x_3 - \frac{b}{m}v_3]$$

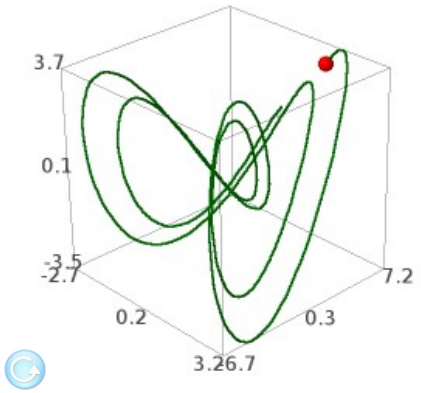
```
print 'a.'
b=.8
[k1,k2,k3]=[-1,-1,-1]
f(t,x1,x2,x3,v1,v2,v3)=[v1,v2,v3,.1*(k1*x1)-b/10*v1,.1*(k2*x2)-
b/10*v2,.1*(k3*x3)-b/10*v3]
CI=vector([1,7,3,1,1,2])
I=[0,40];N=6*I[1]
T=[0,1/6,..,I[1]]
S=rungekutta(f,CI,I,N)
g=line3d([a[0:3] for a in S],color='blue',thickness=.5)
particula=[]
for k in range(len(T)):
    particula.append(g+point3d(S[k][0:3],color='red',size=20))
a=animate(particula)
show(a[0],figsize=[3,3,3]);#a.show(delay=15)
```

a.



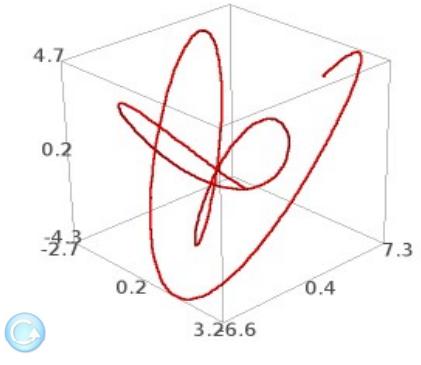

```
print 'b.';[k1,k2,k3]=[-1,-4,-9]
f(t,x1,x2,x3,v1,v2,v3)=[v1,v2,v3,.1*(k1*x1)-.03*v1,.1*
(k2*x2)-.03*v2,.1*(k3*x3)-.03*v3]
CI=vector([1,7,3,1,1,2])
I=[0,40];N=6*I[1];T=[0,1/6,..,I[1]]
S=rungekutta(f,CI,I,N)
g=line3d([a[0:3] for a in S],color='green',thickness=.5)
particula=[]
for k in range(len(T)):
    particula.append(g+point3d(S[k][0:3],color='red',size=20))
a=animate(particula)
show(a[0],figsize=[3,3,3]); #a.show(delay=10)
```

b.




```
print 'c.';[k1,k2,k3]=[-1,-2,-3]
f(t,x1,x2,x3,v1,v2,v3)=[v1,v2,v3,.1*(k1*x1)-.03*v1,.1*
(k2*x2)-.03*v2,.1*(k3*x3)-.03*v3]
CI=vector([1,7,3,1,1,2])
I=[0,40];N=6*I[1];T=[0,1/6,..,I[1]]
S=rungekutta(f,CI,I,N)
g=line3d([a[0:3] for a in S],color='red',thickness=.5)
particula=[]
for k in range(len(T)):
    particula.append(g+point3d(S[k]
[0:3],color='red',pointsize=20))
a=animate(particula)
show(a[0],figsize=[3,3,3]);#a.show(delay=10)
```

c.



Exemplo 8:

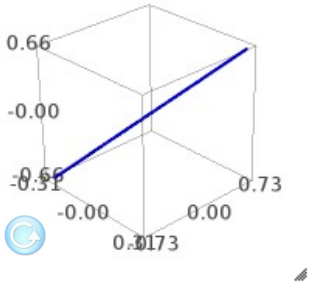
Que forza externa debemos aplicar á partícula para compensar a viscosidade do medio en cada un dos casos anteriores?

Solución:

Queremos que a ecuación $D\mathbf{x} - b\mathbf{x}' + F(t) = m\mathbf{x}''$ teña a mesma solución que $D\mathbf{x} = m\mathbf{x}''$.

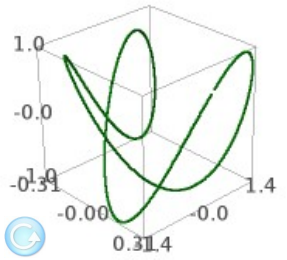
Xa que logo, $F(t) = b\mathbf{x}'$, sendo $\mathbf{x}$ a solución da segunda ecuación

```
[k1,k2,k3]=[-1,-1,-1]
f(t,x1,x2,x3,v1,v2,v3)=[v1,v2,v3,.1*(k1*x1),.1*(k2*x2),.1*
(k3*x3)]
CI=vector([1,7,3,1,1,2])
I=[0,19.8]
N=60*I[1]
T=[0,1/60,..,I[1]]
S=rungekutta(f,CI,I,N)
show(line3d([.3*vector(a[3:6]) for a in S],
color='blue',thickness=.5),figsize=[2,2,2])
```

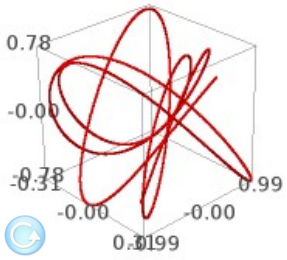



```
[k1,k2,k3]=[-1,-4,-9]
f(t,x1,x2,x3,v1,v2,v3)=[v1,v2,v3,.1*(k1*x1),.1*(k2*x2),.1*
(k3*x3)]
CI=vector([1,7,3,1,1,2])
I=[0,19.8]
N=60*I[1]
T=[0,1/60,..,I[1]]
S=rungekutta(f,CI,I,N)
```

```
show(line3d([.3*vector(a[3:6]) for a in S],
color='green',thickness=.5),figsize=[2,2,2])
```




```
[k1,k2,k3]=[-1,-2,-3]
f(t,x1,x2,x3,v1,v2,v3)=[v1,v2,v3,.1*(k1*x1),.1*(k2*x2),.1*
(k3*x3)]; CI=vector([1,7,3,1,1,2])
I=[0,60]
N=60*I[1]
T=[0,1/60,..,I[1]]
S=rungekutta(f,CI,I,N)
show(line3d([.3*vector(a[3:6]) for a in S],
color='red',thickness=.5),figsize=[2,2,2])
```



3.3.5. Lanzamentos

Un exemplo paradigmático é determinar a traxectoria $\mathrm{x}(t)$ en $\mathbb{R}^3$ dunha masa m sometida a unha forza F. A segunda lei de Newton asegura que a forza actuante é a derivada da cantidade de movemento,

$$F = \frac{d(m \cdot \mathrm{x}')}{dt}$$

Se a masa é constante no tempo, obtemos a famosa ecuación de segunda orde

$F = m \cdot \mathrm{x}''$ que se pode escribir como dúas ecuacións de primeira orde

$$\begin{cases} \mathrm{x}' = \mathrm{v} \\ \mathrm{v}' = \frac{F}{m} \end{cases}$$

ou como unha ecuación vectorial de primeira orde onde $f : [t_0, t_0 + T] \times \mathbb{R}^6 \to \mathbb{R}^6$ está dada por

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ v_1 \\ v_2 \\ v_3 \end{pmatrix}' = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ v_1 \\ v_2 \\ v_3 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \\ \frac{F_1}{m} \\ \frac{F_2}{m} \\ \frac{F_3}{m} \end{pmatrix}$$

ou, abreviadamente, $X' = AX + C$. Posto que se trata dunha ecuación diferencial lineal, o teorema de Picard Lindelöf asegura a existencia e unicidade de solución para unha condición inicial X_0.

Exemplo 9:

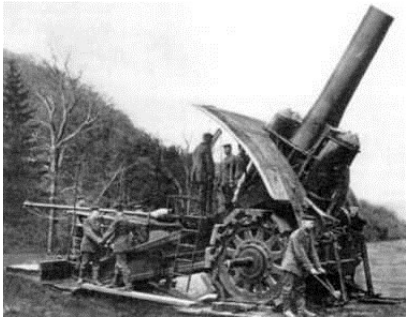


Figura: Gran Bertha en Verdún 1916

O Gran Bertha era un canón desenvolvido polas industrias Krupp durante a Primeira Guerra Mundial que disparaba obuses de 830 kg cunha velocidade inicial de 400 m/s. Debuxar a traxectoria e calcular o alcance dun obús lanzado cun ángulo de 30º, sabendo que o coeficiente de viscosidade do aire é de

$b = 1.73 \cdot 10^{-5}$.

Solución:

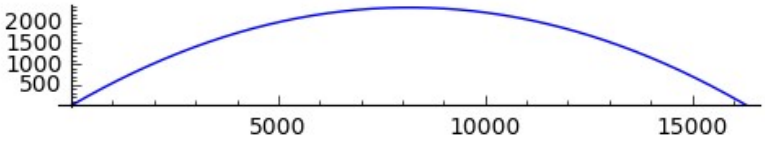
Supoñendo que a traxectoria é plana e que non hai máis forzas que a gravidade e o rozamento co aire $-b\mathrm{v}$, teremos $F = m\mathbf{g} - b\mathrm{v}$ e o sistema a resolver será:

$$\begin{cases} \mathrm{x}' = \mathrm{v} \\ \mathrm{v}' = \mathbf{g} - \frac{b}{m}\mathrm{v} \end{cases}$$

ou, en coordenadas,

$$\begin{pmatrix} x_1 \\ x_2 \\ v_1 \\ v_2 \end{pmatrix}' = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -\frac{b}{m} & 0 \\ 0 & 0 & 0 & -\frac{b}{m} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ v_1 \\ v_2 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \\ -g \end{pmatrix}$$

```
b=1.73*10^(-5)
m=830; g=9.81
v0=430; I=[0,60]
N=I[1]*30
CI=vector([0,0,v0*cos(pi/6),v0*sin(pi/6)])
f(t,x1,x2,v1,v2)=[v1,v2,-b*v1/m,-b*v2/m-g]
S=rungekutta(f,CI,I,N)
D=[[a[0],a[1]] for a in S if a[1]>0 ]
show(line(D,aspect_ratio=1),figsize=[5,3])
```



O alcance deste canón que pesaba 43000 kg e había que desplazalo en ferrocarril, é de

```
print round(D[-1][0]/1000,3),'km'
```

```
16.311 km
```

Exemplo 10:

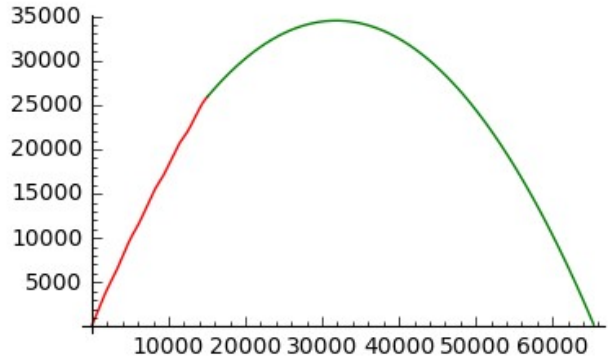
Un obús de 5kg é lanzado desde $X_0 = [0, 100]$ cunha velocidade $V_0 = [500, 1000]$. Durante os primeiros 30 s actúa tamén unha forza externa $F(t) = (100\cos(t), 300\sin(t))$ pero, despois, queda só baixo o influxo da gravidade e a viscosidade do aire, cuxo coeficiente é $b = 1.73 \cdot 10^{-5}$.

Calcular:

a. Punto de impacto do obús contra o chan.

b. Tempo que tarda en producirse devandito impacto.

```
var('t')
m=5; b=1.73*10^(-5)
T0=30
I0=[0,T0]
F1=100*cos(t)
F2=300*cos(t)
N=60*T0
f(t,x0,x1,x2,x3)=[x2,x3,-(b/m)*x2+F1,-9.8-(b/m)*x3+F2]
CI=vector([0,100,500,1000])
R=rungekutta(f,CI,I0,N)
P=[[R[k][0],R[k][1]] for k in range(N) if R[k][1]>0]
DP=line(P,color='red')
if len(P)<N:
    show(DP)
else:
    CI=R[N]
    I=[T0,20*T0]
    N=500
    f(t,x0,x1,x2,x3)=[x2,x3,-(b/m)*x2,-9.8-(b/m)*x3]
    RR=rungekutta(f,CI,I,N)
    PP=[[RR[k][0],RR[k][1]] for k in range(N) if RR[k][1]>0]
    DPP=line(PP,color='green')
    show(DP+DPP,figsize=[4,2.5])
```



Se en lugar dunha masa constante lanzamos un foguete de K kg de carga e P kg de propulsante que se consome uniformemente en T segundos producindo gases que saen do cohete a $v_g m/s$, a derivada da cantidade de movemento nun instante t é

$$\lim_{\triangle t \to 0} \frac{(m+\triangle m)(\mathrm{v}+\triangle \mathrm{v}) - \triangle m(\mathrm{v} - v_g \frac{\mathrm{v}}{||\mathrm{v}||}) - m\mathrm{v}}{\triangle t} = m\mathrm{v}' + m' v_g \frac{\mathrm{v}}{||\mathrm{v}||}$$

Así chegamos á ecuación de Tsiolkovski $F = m\mathbf{v}' + m' v_g \frac{\mathbf{v}}{\|\mathbf{v}\|}$, onde

$$m = K + \frac{P(T-t)}{T} \quad \text{e, xa que logo,} \quad m' = -\frac{P}{T}$$

Exemplo 11:

O comenzo das festas de San Fermín anúnciase co chupinazo desde o balcón central do Concello de Pamplona. Este foguete de cana de 200 g de masa e 100 g de pólvora lánzase desde unha rampa situada a 6 m de altura e inclinada 80º coa horizontal. Se a pólvora, se consome uniformemente en 10 s e a velocidade de saída dos gases é de 1100 m/s, determinar a traxectoria do foguete e calcular o seu alcance, sabendo que o coeficiente de viscosidade do aire é $1.73 \cdot 10^{-5}$.

Solución:

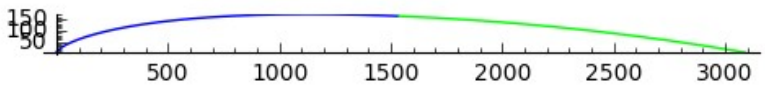
Podemos supoñer que o foguete se move no plano perpendicular á fachada do concello e sae da rampa de lanzamento con velocidade inicial de $0.5\, m/s$

(impulsado por algún resorte, por exemplo). Durante os 10 s iniciais, a ecuación do movemento será:

$$m\mathbf{v}' + m' v_g \frac{\mathbf{v}}{\|\mathbf{v}\|} = m\mathbf{g} - k\mathbf{v} \quad \text{con} \quad m = 0.2 + 0.1\frac{10-t}{10} \quad \text{y} \quad m' = 0$$

O resto do movemento realizarase a masa constante, coas condicións iniciais alcanzadas pola propulsión da pólvora, ata alcanzar a altura 0.

```
var('t x0 x1 x2 x3')
u=[x0,x1]
v=[x2,x3]
U=vector(u)
V=vector(v)
G=vector([0,-9.81])
b=1.73*10^(-5)
K=0.2
P=0.08
vg=1100
T=10
m=K+P*(T-t)/T
mpri=-P/T
CV=(-b-vg*mpri/norma(V))/m
VPRI=CV*V+G
f(t,x0,x1,x2,x3)=[V[0],V[1],VPRI[0],VPRI[1]]
CI=vector([0,6,0.5*cos(80*pi/180),0.5*sin(80*pi/180)])
I=[0,10]
N=960
V2=rungekutta(f,CI,I,N)
P=[[V2[k][0],V2[k][1]] for k in range(N)]
DP=line(P,aspect_ratio=1)
CI=V2[-1]
I=[10,490]
f(t,x0,x1,x2,x3)=[x2,x3,-(b/K)*x2,-9.8-(b/K)*x3]
RR=rungekutta(f,CI,I,N)
PP=[[RR[k][0],RR[k][1]] for k in range(N) if RR[k][1]>0]
DPP=line(PP,rgbcolor=(0,1,0))
show(DP+DPP,figsize=[5,4])
```



O alcance é

```
print round(PP[-1][0]/1000,3), 'km'
```

```
3.092 km
```

Exemplo 12:

O $V2$ foi un mísil balístico desenvolvido en Alemaña a principios da Segunda Guerra Mundial que se empregou contra Bélxica e Inglaterra.

Figura: V2 en Peenemünde 1942

Desde Peenemünde (54.1367132, 13.7747756) lánzase un $V2$ con $K = 300$ kg de carga e $P = 2000$ kg de propulsante que se consome uniformemente en $T = 80$ segundos producindo gases que saen del a $v_g = 2450$ m/s.

Se o radio da Terra é $R = 6367650$ Km e o coeficiente de viscosidade do aire é $b = 1.73 \cdot 10^{-5}$, determinar as condicións iniciais do lanzamento para alcanzar o centro de Londres sabendo que a velocidade vertical inicial é de 9 m/s.

Solución:

Elixindo o sistema de referencia inercial ortonormal asociado ao centro da Terra, ao Ecuador, ao Polo Norte e ao meridiano de Greenwich, a ecuación de Tsiolkovski queda na forma

$$m\mathbf{v}' + m' v_g \frac{\mathbf{v}}{\|\mathbf{v}\|} = -G\frac{M\cdot m}{\|\mathbf{x}\|^2}\frac{\mathbf{x}}{\|\mathbf{x}\|} - b\mathbf{v}$$

onde

$$m = K + \frac{P(T-t)}{T} \quad \text{y} \quad m' = -\frac{P}{T}$$

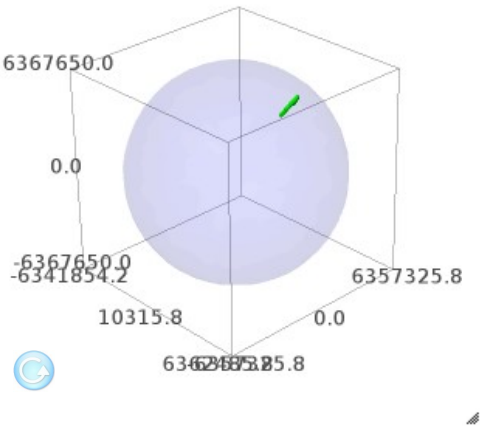
sendo $G = 6.67 \cdot 10^{-11}$ a constante de gravitación universal, $M = 5.972 \cdot 10^{24}$ a masa da Terra.

Podemos tratalo como un problema de Cauchy onde a posición inicial é Peenemünde e a velocidade inicial expresámola como combinación adecuada das compoñentes (**zenital**, **leste**, **norte**) en Peenemünde. Na compoñente **leste** teremos en conta a velocidade debida á rotación da Terra.

Como no exemplo anterior, estudaremos dous movementos: o producido pola propulsión nos 80 segundos iniciais e a súa continuación, a masa constante, ata que a distancia ao centro da Terra volva ser R. Usamos as constantes k_e e k_n para modificar as compoñentes **leste** e **norte** da velocidade inicial. Tras varios experimentos quedamos con $k_e = -0.000915$ e $k_n = -0.019$.

```
var('t x y z u v w')
R=6367650
ke=-.000915
kn=-.019
LAP=54.136713*pi/180
LOP=13.774775*pi/180
x0=(R*cos(LAP)*cos(LOP)).n()
y0=(R*cos(LAP)*sin(LOP)).n()
z0=(R*sin(LAP)).n()
ve=ke*
((pi/(12*3600))*vector([-R*cos(LAP)*sin(LOP),R*cos(LAP)*cos(LOP),0]
vn=kn*vector([-sin(LAP)*cos(LOP),-
sin(LAP)*sin(LOP),cos(LOP)]).n()
vz=9*vector([cos(LAP)*cos(LOP),cos(LAP)*sin(LOP),sin(LAP)]).n()
v0=ve+vn+vz
G=6.67*10^(-11)
U=vector([x,y,z])
V=vector([u,v,w])
M=5.972*10^(24)
b=1.73*10^(-5)
```

```
K=300
P=2000
vg=2450
T=80
m=K+P*(T-t)/T
mpri=-P/T
FG=-G*M/(norma(U)^3)*U
CV=((-mpri*vg-b*norma(V))/(m*norma(V)))*V
VPRI=CV+FG
I=[0,80]
N=240
CI=vector([x0,y0,z0,v0[0],v0[1],v0[2]])
f(t,x,y,z,u,v,w)=[u,v,w,VPRI[0],VPRI[1],VPRI[2]]
V2=rungekutta(f,CI,I,N)
IP=[[V2[k][0],V2[k][1],V2[k][2]] for k in range(N)]
DP=line3d(IP,thickness=5)
esf(fi,teta)=[R*cos(fi)*cos(teta),R*cos(fi)*sin(teta),R*sin(fi)]
ES=parametric_plot3d(esf,(fi,-pi/2,pi
/2),(teta,0,2*pi),opacity=.1)
CI2=V2[-1]
I2=[80,1580]
N2=3000
f2(t,x,y,z,u,v,w)=[u,v,w,FG[0]-b*u/K,FG[1]-b*v/K,FG[2]-b*w/K]
RR=rungekutta(f2,CI2,I2,N2)
PP=[[RR[k][0],RR[k][1],RR[k][2]] for k in range(N2) if
norma(vector([RR[k][0],RR[k][1],RR[k][2]])).n()>R ]
DPP=line3d(PP,rgbcolor=(0,1,0),thickness=5)
show(ES+DP+DPP,figsize=[3,3,3])
```

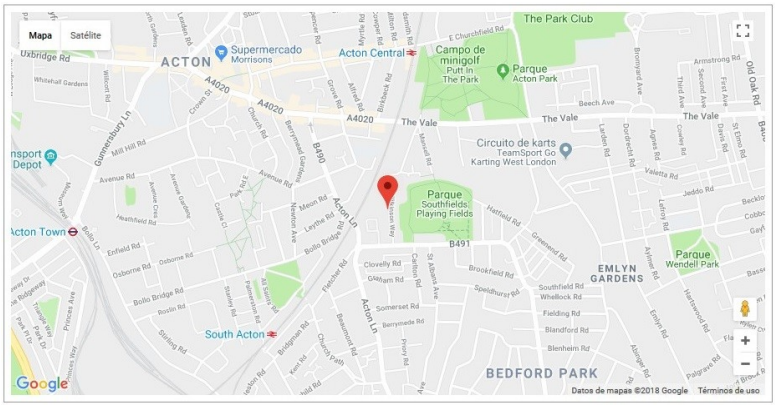


Finalmente, para achar as coordenadas xeográficas do punto de impacto debemos ter en conta o xiro da Terra no tempo de voo do foguete.

```
tempo=(I[1]+len(PP)/2).n()
Xiro=tempo*180/(12*3600)
R0=norma(vector(PP[-1]))
LALLR=(asin(PP[-1][2]/R0)).n()
LOLLR=acos(PP[-1][0]/(R0*cos(LALLR)).n())
if abs(sin(LOLLR)-(PP[-1][1]/(R0*cos(LALLR))).n())>10^(-4):
    LOLLR=-LOLLR
CGLL=[(LALLR*180/pi).n(), (LOLLR*180/pi).n()]
CGIM=[CGLL[0],CGLL[1]+Xiro]
CGIM
```

```
[51.5036952679636, -0.264430681534109]
```

Introducindo na páxina web www.mundivideo.com as coordenadas obtidas podemos visualizar o lugar do impacto



3.3.6. Curvas de persecución

Imaxinemos que un móbil, de traxectoria coñecida $\mathrm{c}:[t_0,t_0+T]\to\mathbb{R}^3$, é perseguido por outro móbil. A traxectoria $\mathrm{x}(t)$ do perseguidor cumprirá

$$\mathrm{x}'(t)=k(t)\frac{\mathrm{c}(t)-\mathrm{x}(t)}{||\mathrm{c}(t)-\mathrm{x}(t)||}\quad\text{onde}\quad |k(t)|=||\mathrm{x}'(t)||.$$

Se supoñemos coñecida a relación $r(t)=\dfrac{||\mathrm{x}'(t)||}{||\mathrm{c}'(t)||}$, poderemos concluír que

$$\mathrm{x}'(t) = f(t, \mathrm{x}(t)) \quad \text{con} \quad f(t, \mathrm{x}(t)) = r(t)||\mathrm{c}'(t)|| \frac{\mathrm{c}(t) - \mathrm{x}(t)}{||\mathrm{c}(t) - \mathrm{x}(t)||}$$

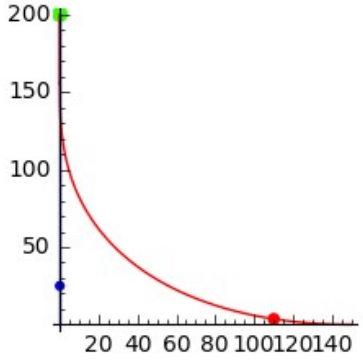
e, para cada posición inicial do perseguidor $\mathrm{x}(t_0) = \mathrm{x}_0$, teremos unha única curva solución.

Exemplo 13:

No instante $t = 0$, un coello pasa pola orixe (0,0) correndo a 5 m/s cara ao seu tobo que está en (0,200) e un cazador, apostado en (150,0), solta ao seu can que persegue ao coello con celeridade constante de 8 m/s. A que distancia do tobo o alcanza?

```
T=[0,.2,..,40]
c(t)=[0,5*t]
var('x y'); X=vector([x,y])
f(t,x,y)=list(8*(c(t)-X)/norma(c(t)-X))
CI=vector([150,0]); I=[0,40]; N=200
P=rungekutta(f,CI,I,200)
L=line([[0,0],
[0,200]])+point([0,200],hue=.3,pointsize=50)+line(P,hue=0)
concan=[]
for n in range(200):
    if norma(c(n*.2)-vector(P[n]))>.5:
        concan.append(L+point(c(n*.2),pointsize=20)+point(P[n],
        hue=0,pointsize=30))
    else:
        break
a=animate(concan);
#a.show(delay=10)
show(a[25],figsize=[2.5,2.5])
print 'Alcánzao a',200-c(n*.2)[1],'m do tobo'
```

```
Alcánzao a 46.0000000000000 m do tobo
```

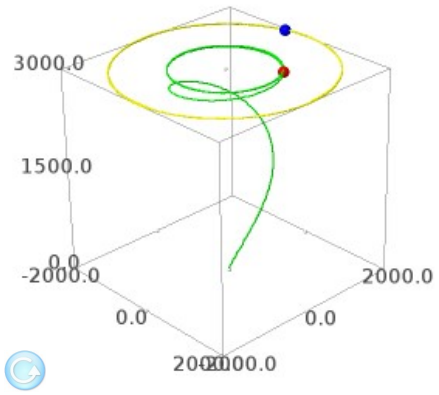


Exemplo 14:

Un avión espía voa en círculo a 3000 m de altura sobre a nosa posición (0,0,0), describindo unha circunferencia de 2000 m de radio cunha velocidade constante de 200 m/s.
Para advertirlle que nos molesta a súa presenza lanzámoslle un mísil intelixente que o persegue cunha celeridade igual a 1/2 da súa. Animar o movemento do avión e o mísil.

```
var('x0 x1 x2')
X=vector([x0,x1,x2])
c(t)=[2000*cos(t/10),2000*sin(t/10),3000]
v=diff(c,t); r=.5
f(t,x0,x1,x2)=list(r*norma(v(t))*((c(t)-X)/norma(c(t)-X)))
CI=vector([0,0,0]); I=[0,300]; N=2*I[1]
S=rungekutta(f,CI,I,N)
g=line3d(S,rgbcolor=(0,1,0),thickness=2)
a=parametric_plot3d(c,(t,0,I[1]),rgbcolor=(1,1,0),thickness=2)
FXI=point3d([-2000,0,0],color='white')
FXD=point3d([2000,0,0],color='white')
FYI=point3d([0,-2000,0],color='white')
FYD=point3d([0,2000,0],color='white')
FZB=point3d([0,0,0],color='white')
FZA=point3d([0,0,3000],color='white')
A=[]; k=0; R=[0,3,..,300]
for r in R:
    P=point3d(list(c(t=r)),size=15,color='blue',figsize=
[12,12,12])
    Q=point3d(list(S[6*k]),size=15,color='red',figsize=
[12,12,12])
```

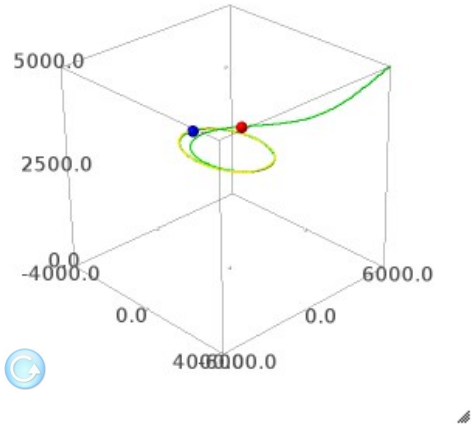
```
    A.append(FXI+FYI+FZB+g+a+P+Q+FXD+FYD+FZA)
    k=k+1
an=animate(A); #show(an)
show(an[27],figsize=[3,3,3])
```



Como insiste en espiarnos, un dos nosos cazas dispáralle outro mísil con celeridade 10/9 da súa, desde (4000,6000,5000) cando o avión pasa por (2000, 0, 3000). Achar a traxectoria deste mísil e determinar o tempo e o lugar do impacto.

```
var('x0 x1 x2'); X=vector([x0,x1,x2])
c(t)=[2000*cos(t/10),2000*sin(t/10),3000]
v=diff(c,t); r=10/9
f(t,x0,x1,x2)=list(r*norma(v(t))*((c(t)-X)/norma(c(t)-X)))
CI=vector([4000,6000,5000])
I=[0,300]; N=2*I[1]; S=rungekutta(f,CI,I,N)
g=line3d(S,rgbcolor=(0,1,0),thickness=2)
a=parametric_plot3d(c,(t,0,I[1]),rgbcolor=(1,1,0),thickness=2)
FXI=point3d([-4000,0,0],color='white')
FXD=point3d([4000,0,0],color='white')
FYI=point3d([0,-6000,0],color='white')
FYD=point3d([0,6000,0],color='white')
FZB=point3d([0,0,0],color='white')
FZA=point3d([0,0,5000],color='white')
A=[]; k=0; R=[0,3,..,300]
for r in R:
    P=point3d(list(c(t=r)),size=15,color='blue',figsize=
[12,12,12])
```

```
    Q=point3d(list(S[6*k]),size=15,color='red',figsize=
[12,12,12])
    A.append(FXI+FYI+FZB+g+a+P+Q+FXD+FYD+FZA)
    k=k+1
an=animate(A); #show(an)
show(an[10],figsize=[3,3,3])
```



3.3.7. Curvas de arrastre

Imaxinemos que un móbil, de traxectoria coñecida $\mathrm{c} : [t_0, t_0 + T] \to \mathbb{R}^3$, leva ligada, mediante unha articulación esférica, unha vara de lonxitude ℓ en cuxo extremo hai unha bóla. Se sobre a bóla actúa, ademais, unha forza exterior $\phi : [t_0, t_0 + T] \to \mathbb{R}^3$ e unha fricción viscosa proporcional á velocidade, a traxectoria da bóla cumprirá

$$\mathrm{x}(t) = \mathrm{c}(t) + \ell \mathrm{u}(t) \quad \text{con} \quad ||\mathrm{u}(t)|| = 1,$$

a súa velocidade cumprirá

$$\mathrm{x}'(t) = \mathrm{c}'(t) + \ell \mathrm{u}'(t) \quad \text{con} \quad (\mathrm{u}(t)|\mathrm{u}'(t)) = 0$$

e, segundo as leis de Newton, a súa aceleración cumprirá

$$\mathrm{x}''(t) = \mathrm{c}''(t) + \ell\mathrm{u}''(t) = k(t)\mathrm{u}(t) - \beta(\mathrm{c}'(t) + \ell\mathrm{u}'(t)) + \phi(t)$$

$$\text{con} \quad (\mathrm{u}(t)|\mathrm{u}'(t)) = 0 \quad \text{e} \quad (\mathrm{u}(t)|\mathrm{u}''(t)) = -(\mathrm{u}'(t)|\mathrm{u}'(t))$$

onde $k(t)$ é unha función escalar e β é o coeficiente de viscosidade. Multiplicando por $\mathrm{u}(t)$ teremos

$$k(t) = (\mathrm{c}''(t)|\mathrm{u}(t)) - \ell(\mathrm{u}'(t)|\mathrm{u}'(t)) + \beta(\mathrm{c}'(t)|\mathrm{u}(t)) - (\phi(t)|\mathrm{u}(t))$$

e, en consecuencia,

$$\mathrm{u}'' = \frac{-\mathrm{c}'' + ((\mathrm{c}''|\mathrm{u}) - \ell(\mathrm{u}'|\mathrm{u}') + \beta(\mathrm{c}'|\mathrm{u}) - (\phi|\mathrm{u}))\mathrm{u} - \beta\mathrm{c}' - \beta\ell\mathrm{u}' + \phi}{\ell}$$

Esta ecuación de segunda orde é equivalente ao sistema

$$\begin{cases} \mathrm{u}' = \mathrm{v} \\ \mathrm{v}' = \frac{-\mathrm{c}'' + ((\mathrm{c}''|\mathrm{u}) - \ell(\mathrm{v}|\mathrm{v}) + \beta(\mathrm{c}'|\mathrm{u}) - (\phi|\mathrm{u}))\mathrm{u} - \beta\mathrm{c}' - \beta\ell\mathrm{v} + \phi}{\ell} \end{cases}$$

e designando $U = \begin{pmatrix} \mathrm{u} \\ \mathrm{v} \end{pmatrix}$ podemos escribilo como unha ecuación vectorial $U' = f(t, U)$. Para cada condición inicial $U(t_0) = U_0$ teremos unha única solución $U(t)$ cuxas tres primeiras compoñentes constituirán un $\mathrm{u}(t)$ que nos permitirá escribir a traxectoria da bóla

$$\mathrm{x}(t) = \mathrm{c}(t) + \ell\mathrm{u}(t).$$

No caso particular de que a función $\mathrm{c} : [t_0, t_0 + T] \to \mathbb{R}^3$ sexa identicamente nula, a bóla moverase nunha esfera de centro 0 e radio ℓ describindo unha traxectoria $\ell\mathrm{u}$ onde

$$\begin{cases} \mathrm{u}' = \mathrm{v} \\ \mathrm{v}' = \frac{(-\ell(\mathrm{v}|\mathrm{v})-(\phi|\mathrm{u}))\mathrm{u}-\beta\ell\mathrm{v}+\phi}{\ell} \end{cases}$$

No caso particular de que o coeficiente de viscosidade sexa moi grande podemos supoñer que a traxectoria da bóla é

$$\mathrm{x}(t) = \mathrm{c}(t) + \ell\mathrm{u}(t) \quad \text{con} \quad ||\mathrm{u}(t)|| = 1 \quad \mathrm{y} \quad \mathrm{x}'(t)||\mathrm{u}(t).$$

Entón, o problema queda de primeira orde e independente de calquera forza externa ϕ, pois terá que existir unha certa función escalar $h(t)$ tal que

$$\mathrm{c}'(t) + \ell\mathrm{u}'(t) = h(t)\mathrm{u}(t) \quad \text{con} \quad (\mathrm{u}(t)|\mathrm{u}'(t)) = 0.$$

Multiplicando por $\mathrm{u}(t)$ temos que $h(t) = (\mathrm{c}'(t)|\mathrm{u}(t))$ e, xa que logo,

$$\mathrm{u}'(t) = f(t, \mathrm{u}(t)) \quad \text{con} \quad f(t, \mathrm{u}(t)) = \frac{(\mathrm{c}'(t)|\mathrm{u}(t))\mathrm{u}(t) - \mathrm{c}'(t)}{\ell}$$

Así, para cada vector unitario inicial $\mathrm{u}(t_0) = \mathrm{u}_0$, teremos unha única solución $\mathrm{u}(t)$ e unha única traxectoria $\mathrm{x}(t) = \mathrm{c}(t) + \ell\mathrm{u}(t)$.

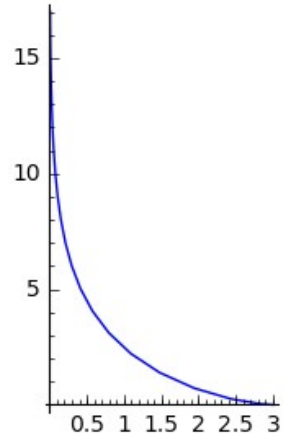
Exemplo 15:

Unha persoa anda polo eixe OY a $3.6\,km/h$ e leva ao seu can atado cunha correa de $3m$. Cando pasa pola orixe, o can está en (3,0) e, a partir dese momento, resistese a camiñar. Canto percorre o can nos 20 segundos seguintes?

```
f(t,u0,u1)=[u0*u1/3,(u1^2-1)/3]
CI=vector([1,0])
I=[0,21]
N=21
P=rungekutta(f,CI,I,N)
T=[t for t in range(N)]
C=line([vector([0,t])+3*P[t] for t in T])
show(C,figsize=[2,3])
```

```
V=[vector([0,t])+3*P[t] for t in T]
L=[norma(V[t]-V[t+1]) for t in range(N-1)]
print 'O can percorre',sum(L),'m'
```

```
O can percorre 17.9137398339802 m
```



Exemplo 16:

Do punto [0,0,10], centro do teito do noso laboratorio, colga, mediante unha articulación esférica, unha vara de $7m$ e masa desprezable, acabada nunha bóla de $50kg$. Se no instante inicial colocamos o centro da bóla en

$$(0,0,10)+7\cdot(cos\frac{\pi}{4}\cdot cos\frac{\pi}{6},cos\frac{\pi}{4}\cdot sen\frac{\pi}{6},-sen\frac{\pi}{4})$$

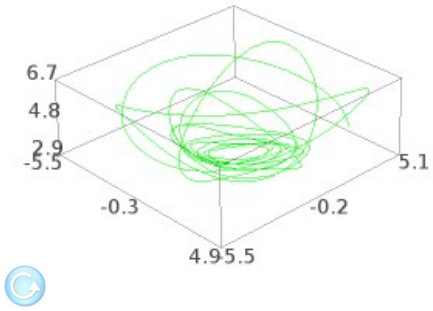
e dámoslle unha velocidade de $(-sen\frac{\pi}{4},cos\frac{\pi}{4},0)$:

a. Debuxar a traxectoria da bóla durante o primeiro minuto se o coeficiente de viscosidade da atmosfera do laboratorio é $b=0.3$.

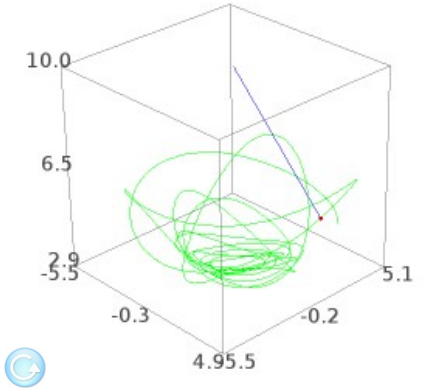
b. Animar o movemento da vara e a bóla.

```
var('t x0 x1 x2 x3 x4 x5')
C=vector([0,0,10]); L=7; b=3; m=50
u=[x0,x1,x2]; U=vector(u)
v=[x3,x4,x5]; V=vector(v)
FI=vector([0,0,-9.8])
CI=vector([cos(pi/4)*cos(pi/6),cos(pi/4)*sin(pi/6),-sin(pi/4),-
sin(pi/4),cos(pi/4),0])
K=((-m*L*(V*V)-m*(FI*U))*U-b*L*V+m*FI)/(m*L)
```

```
f(t,x0,x1,x2,x3,x4,x5)=v+list(K)
I=[0,60]; N=300; h=60/300
H=rungekutta(f,CI,I,N)
H1=[C+L*H[k][0:3] for k in [0..N]]
g=line3d(H1,rgbcolor=(0,1,0),thickness=1)
show(g,aspect_ratio=1,figsize=[3,3,3])
```




```
GG=[]
for k in range(N/2):
    GG.append(g+line3d((C,H1[2*k]),rgbcolor=
(0,0,1),thickness=1)+point3d(H1[2*k][0:3],rgbcolor=
(1,0,0),size=5))
AGG=animate(GG);#show(AGG); show(AGG[10], figsize=[3,3,10])
```



3.3.8. Loxodrómicas na superficie terrestre

Supoñemos a superficie dada pola carta $s:(-\frac{\pi}{2},\frac{\pi}{2})\times(0,2\pi)\to R^3$ con

$$s(u,v)=(R\cos u\cos v, R\cos u\sin v, R\sin u)$$

e R=6367650.

Unha loxodrómica é unha curva sobre ela que forma ángulo constante cos meridianos. É importante en navegación porque se percorre a rumbo fixo. Podemos expresarla dando unha relación $u=u(v)$ co que a súa ecuación será

$$\mathrm{x}(v)=(R\cos u(v)\cos v, R\cos u(v)\sin v, R\sin u(v))$$

Como $\mathrm{x}'=\frac{\partial s}{\partial u}u'+\frac{\partial s}{\partial v}$ e os vectores tanxentes aos meridianos e paralelos $\frac{\partial s}{\partial u}$ e $\frac{\partial s}{\partial v}$, son ortogonales e teñen normas R e $R\cos u$, se A é o ángulo constante que forma x' con $\frac{\partial s}{\partial u}$, débese cumprir que $\cos A=\frac{u'}{\sqrt{u'^2+\cos^2 u}}$. Despexando obtemos a ecuación diferencial da loxodrómica de rumbo A en variables separadas $dv=\tan A\frac{du}{\cos u}$ e, polo tanto

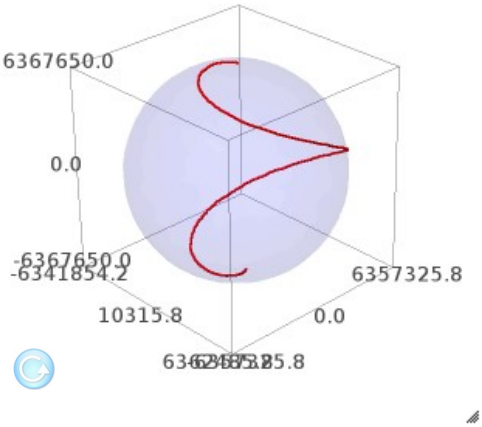
$$v=\tan A\int\frac{du}{\cos u}=C+\tan A\,\log\left(\frac{1+\sin u}{1-\sin u}\right)^{1/2}$$

Exemplo 17:

Debuxar sobre a superficie terrestre a loxodrómica de rumbo $\frac{\pi}{4}$ e $C=2$.

```
R=6367650

s(u,v)=[R*cos(u)*cos(v),R*cos(u)*sin(v),R*sin(u)]
T=parametric_plot3d(s,(u,-pi/2,pi/2),(v,0,2*pi),opacity=.1)

K=tan(2*pi/3)
C=20
v(u)=C+K*(1/2)*log((1+sin(u))/(1-sin(u)))
L(u)=[R*cos(u)*cos(v(u)),R*cos(u)*sin(v(u)),R*sin(u)]
```

```
PL=parametric_plot3d(L,(u,-pi/2,pi/2),thickness=3,color='red')
show(T+PL,figsize=[3,3,3])
```



Exemplo 18:

a. Con qué rumbo fixo debemos zarpar de Maracaibo (10.6876898, -71.5975723) para chegar a Vigo (42.22128, -8.733557) dando menos dunha volta ao mundo?

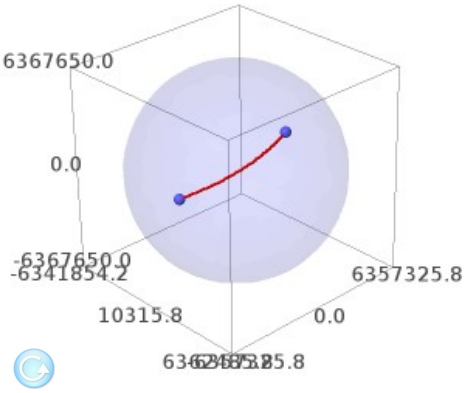
b. Calcular a lonxitude da viaxe.

```
print 'a.'
var('C B')
f1=C+B*(1/2)*log((1+sin(10.6876898*pi/180))/(1-sin(10.6876898*pi
/180)))+71.5975723*pi/180
f2=C+B*(1/2)*log((1+sin(42.22128*pi/180))/(1-sin(42.22128*pi
/180)))+8.733557*pi/180
S=solve([f1,f2],[B,C],solution_dict=True)
print 'O rumbo é =', (atan(S[0][B])*180/pi).n(),'graos'
```

```
a.
O rumbo é = 60.2636123267762 graos
```

```
R=6367650
s(u,v)=[R*cos(u)*cos(v),R*cos(u)*sin(v),R*sin(u)]
T=parametric_plot3d(s,(u,-pi/2,pi/2),(v,0,2*pi),opacity=.1)
K=(S[0][B]).n()
C=(S[0][C]).n()
v(u)=C+K*(1/2)*log((1+sin(u))/(1-sin(u)))
L(u)=[R*cos(u)*cos(v(u)),R*cos(u)*sin(v(u)),R*sin(u)]
```

```
PL=parametric_plot3d(L,(u,10.6876898*pi/180,42.22128*pi
/180),thickness=3,color='red')
show(T+PL+point3d(s(10.6876898*pi/180,-71.5975723*pi
/180),size=15)+point3d(s(42.22128*pi/180,
-8.733557*pi/180),size=15),figsize=[3,3,3])
```




```
print 'b.'
LonMV=loncurva([10.6876898*pi/180,42.22128*pi/180],L)
print'A lonxitude da viaxe é',round(LonMV/1000,3), 'km'
```

```
b.
A lonxitude da viaxe é 7065.295 km
```

3.3.9. Principios variacionais

Algunhas curvas importantes, $\mathrm{x} : [t_1, t_2] \to \mathbb{R}^n$, se presentan na natureza como puntos extremais dun funcional de tipo

$$\mathcal{J}(\mathrm{x}) = \int_{t_1}^{t_2} F(t, \mathrm{x}(t), \mathrm{x}'(t))dt$$

onde a función F é, polo menos, continua.

Podemos ver nas Notas Persoais de José Luis Rubio de Francia que a condición necesaria e suficiente para que x sexa un punto extremal de $\mathcal{J}$ é que cumpra a ecuación de Euler.

$$\frac{\partial F}{\partial x_i} - \frac{d}{dt}\Big(\frac{\partial F}{\partial x_i'}\Big) = 0 \quad \forall i = 1, \cdots, n$$

Casos particulares sinxelos:

1. Se $F = F(\mathrm{x}')$ resulta que $x_k'' = 0$ para $k = 1, 2, \ldots, n$ e, xa que logo as solucións da ecuación de Euler son rectas.

2. Se $F = F(t, \mathrm{x}')$ entón $\frac{d}{dt}(\frac{\partial F}{\partial \mathrm{x}'}) = 0$ e, polo tanto $\frac{\partial F}{\partial \mathrm{x}'} = C$ (constante).

3. Se $F = F(\mathrm{x}, \mathrm{x}')$ a ecuación de Euler pode escribirse na forma $\frac{d}{dt}(F - \frac{\partial F}{\partial \mathrm{x}'} \cdot \mathrm{x}') = 0$ logo hai que resolver a ecuación de primeiro orde $F - \frac{\partial F}{\partial \mathrm{x}'} \cdot \mathrm{x}' = C$ (constante).

Exemplo 19: Xeodésicas en $\mathbb{R}^n$

Atopar a curva $\mathrm{x} : [t_1, t_2] \to \mathbb{R}^n$ con $\mathrm{x}(t_1) = \mathrm{a}$ e $\mathrm{x}(t_2) = \mathrm{b}$ de lonxitude mínima.

Solución:

Temos que minimizar $\mathcal{J}(\mathrm{x}) = \int_{t_1}^{t_2} ||\mathrm{x}'|| dt$ e, xa que logo, $F(t, \mathrm{x}, \mathrm{x}') = ||\mathrm{x}'||$.

Estamos no caso 1 e, xa que logo, a solución é $\mathrm{x}(t) = \mathrm{c}_1 t + \mathrm{c}_2$ que é una recta. Impoñendo as condicións nos extremos vemos que $\mathrm{c}_1 = \frac{\mathrm{b}-\mathrm{a}}{t_2 - t_1}$ e $\mathrm{c}_2 = \frac{\mathrm{a}t_2 - \mathrm{b}t_1}{t_2 - t_1}$

Exemplo 20: Xeodésicas nun cilindro recto de $\mathbb{R}^3$

A ecuación dun cilindro recto de radio R é unha superficie $s : (0, 2\pi) \times (0, H) \to \mathbb{R}^3$ onde $s(u, v) = (R\cos u, R \sin u, v)$. A xeodésica será unha curva sobre dito cilindro e supoñemos que podemos escribila en función do parámetro u na forma $\mathrm{x}(u) = (R\cos u, R \sin u, v(u))$. Atopar o tramo de xeodésica entre os puntos $(0, -R, 0)$ e $(0, R, h)$.

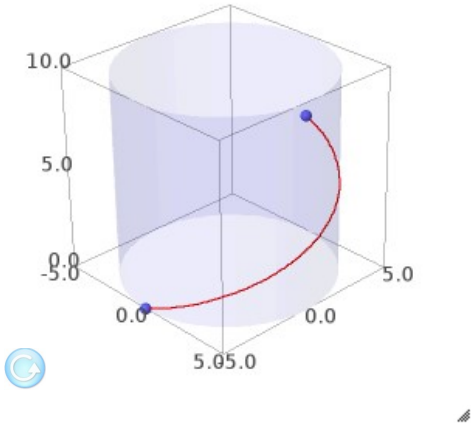
Solución:

Debemos minimizar o funcional

$$\mathcal{J}(\mathrm{x}) = \int_{u_2}^{u_1} ||\mathrm{x}'||du$$

Estamos no primeiro caso particular e debe cumprirse que $\mathrm{x}'' = 0$ co que $v''(u) = 0$ e, xa que logo, $v = C_1 u + C_2$. Tendo en conta as condicións nos extremos temos que $C_1 = \frac{h}{\pi}$ e $C_2 = -\frac{3h}{2}$.

```
R=5
H=10
h=6
C1=h/pi
C2=-3*h/2
s(u,v)=[R*cos(u),R*sin(u),v]
DC=parametric_plot3d(s,(u,0,2*pi),(v,0,H),opacity=.1)
x(u)=[R*cos(u),R*sin(u),C1*u+C2]
Dx=parametric_plot3d(x,(u,3*pi/2,5*pi
/2),thickness=2,color='red')
show(DC+Dx+point3d((0,-R,0),size=15)+point3d((0,R,h),size=15),
figsize=[3,3,3])
```



Exemplo 21: Xeodésicas na superficie terrestre

Como en 3.3.8 aproximamos a superficie pola carta $s : (-\frac{\pi}{2}, \frac{\pi}{2}) \times (0, 2\pi) \to \mathbb{R}^3$ con $s(u, v) = (R\cos u \cos v, R\cos u \sin v, R\sin u)$ e R=6367650. Expresar a xeodésica dando unha relación $v = v(u)$ co que a súa ecuación será

$$\mathrm{x}(u) = (R\cos u\cos v(u), R\cos u\sin v(u), R\sin u)$$

Solución:

Debemos minimizar o funcional

$$\mathcal{J}(\mathrm{x}) = \int_{u_2}^{u_1} ||\mathrm{x}'||du$$

Como $\mathrm{x}'(u) = \frac{\partial s}{\partial u} + \frac{\partial s}{\partial v}v'$ e os vectores $\frac{\partial s}{\partial u}$ e $\frac{\partial s}{\partial v}$ son ortogonais de normas respectivas R e $R\cos u$, temos que

$$F = ||\mathrm{x}'|| = R\sqrt{1 + v'^2\cos^2 u}$$

Estamos nosegundo caso particular e debe cumprirse que $\frac{\partial F}{\partial v'} = C$ (constante), é dicir,

$$\frac{dv}{du} = \frac{1}{\cos u\sqrt{A^2\cos^2 u - 1}} \quad \text{onde} \quad A = \frac{R}{C} > 1$$

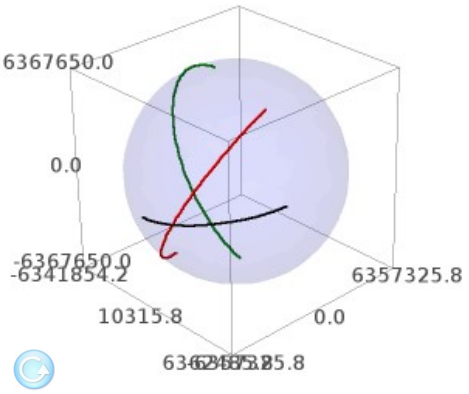
Esta ecuación diferencial en variables separadas se integra doadamente

$$v = K + \int \frac{1}{\cos u\sqrt{A^2\cos^2 u - 1}}du$$

Para cada par de valores (A, K) obtemos unha xeodésica sobre a Terra. Como exemplos, presentamos en verde a xeodésica (9,3), en vermello a (2,5) e en negro a (1.02, -1):

```
R=6367650
s(u,v)=[R*cos(u)*cos(v),R*cos(u)*sin(v),R*sin(u)]
T=parametric_plot3d(s,(u,-pi/2,pi/2),(v,0,2*pi),opacity=.1)
A=9
K=3
v(u)=integrate(1/(cos(u)*sqrt(A^2*cos(u)^2-1)),u)+K
x(u)=[R*cos(u)*cos(v(u)),R*cos(u)*sin(v(u)),R*sin(u)]
D93=parametric_plot3d(x,(u,-pi/2,pi
/2),thickness=3,color='green')
```

```
A=2
K=5
v(u)=integrate(1/(cos(u)*sqrt(A^2*cos(u)^2-1)),u)+K
x(u)=[R*cos(u)*cos(v(u)),R*cos(u)*sin(v(u)),R*sin(u)]
D25=parametric_plot3d(x,(u,-pi/2,pi/2),thickness=3,color='red')
A=1.02
K=-1
v(u)=integrate(1/(cos(u)*sqrt(A^2*cos(u)^2-1)),u)+K
x(u)=[R*cos(u)*cos(v(u)),R*cos(u)*sin(v(u)),R*sin(u)]
D121=parametric_plot3d(x,(u,-pi/2,pi
/2),thickness=3,color='black',figsize=[3,3,3])
show(T+D93+D25+D121, figsize=[3,3,3])
```




```
LL=LonMV/1000
LG=longeo([10.6876898, -71.5975723],[42.22128, -8.733557])/1000
LL-LG
```

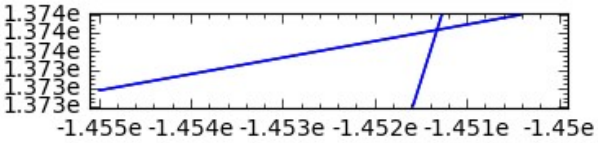
```
81.9026638639216
```

Exemplo 22:

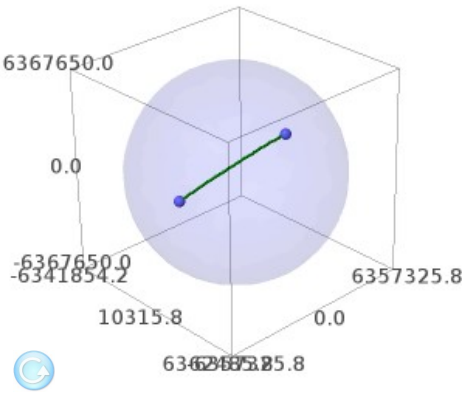
Achar as constantes (A, K) da xeodésica entre Maracaibo $(10.6876898, -71.5975723)$ e Vigo $(42.22128, -8.733557)$.

```
u0=10.6876898
v0=-71.5975723
u1=42.22128
v1=-8.733557
var('A K')
```

```
assume(A>1)
v(u)=K+integrate(1/(cos(u)*sqrt(A^2*cos(u)^2-1)),u)
U0=u0*pi/180
V0=v0*pi/180
U1=u1*pi/180
V1=v1*pi/180
f1=v(U0)-V0
f2=v(U1)-V1
DF1=implicit_plot(f1,(K,-1.455,-1.45),(A,1.373,1.374))
DF2=implicit_plot(f2,(K,-1.455,-1.45),(A,1.373,1.374))
show(DF1+DF2,figsize=[4,4])
```



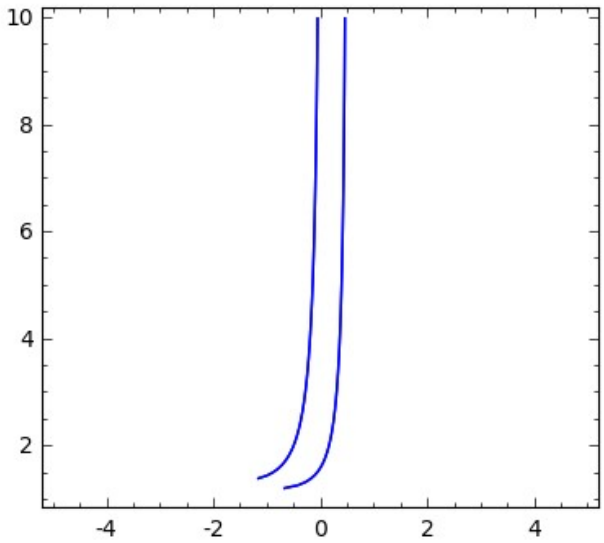

```
A=1.374
K=-1.4515
R=6367650
s(u,v)=[R*cos(u)*cos(v),R*cos(u)*sin(v),R*sin(u)]
T=parametric_plot3d(s,(u,-pi/2,pi/2),(v,0,2*pi),opacity=.1)
v(u)=K+integrate(1/(cos(u)*sqrt(A^2*cos(u)^2-1)),u)
x(u)=[R*cos(u)*cos(v(u)),R*cos(u)*sin(v(u)),R*sin(u)]
Px=parametric_plot3d(x,(u,U0,U1),thickness=3,color='green')
show(Px+T+point3d(s(U0,V0),size=15)+point3d(s(U1,V1),size=15),figsi
[3,3,3])
```



Exemplo 23:

Comprobar que o método anterior non funciona para determinar as constantes (A, K) entre Barcelona e Alejandría.

```
u0=41.3880386
v0=2.1700101
u1=31.2105007
v1=29.9125309
var('A K')
assume(A>1)
v(u)=K+integrate(1/(cos(u)*sqrt(A^2*cos(u)^2-1)),u)
U0=u0*pi/180
V0=v0*pi/180
U1=u1*pi/180
V1=v1*pi/180
f1=v(U0)-V0
f2=v(U1)-V1
DF1=implicit_plot(f1,(K,-5,5),(A,1.01,10))
DF2=implicit_plot(f2,(K,-5,5),(A,1.01,10))
show(DF1+DF2,figsize=[4,4])
```



Como neste caso as curvas non se cortan non podemos utilizar o método anterior.

Movendo as imaxes das xeodésicas obtidas podemos ver que son arcos de

círculos máximos. Dous puntos da superficie terrestre de coordenadas xeográficas (u_0, v_0), (u_1, v_1) e polo Polo Norte son vértices dun triángulo esférico cuxos lados son a xeodésica que os une e os meridianos $v = v_0$ e $v = v_1$. A este triángulo esférico se lle pode aplicar o teorema do coseno da trigonometría esférica e isto permítenos definir a función de Sage **loxeo(**u_0, v_0, u_1, v_1, h**)**, que nos debuxa sempre a loxodrómica dun transporte que viaxa a h metros de altura no rectángulo esférico determinado polos puntos e a xeodésica que pode ou non pertencer a dito rectángulo esférico.

Exemplo 24:

Aplicar a función loxeo():

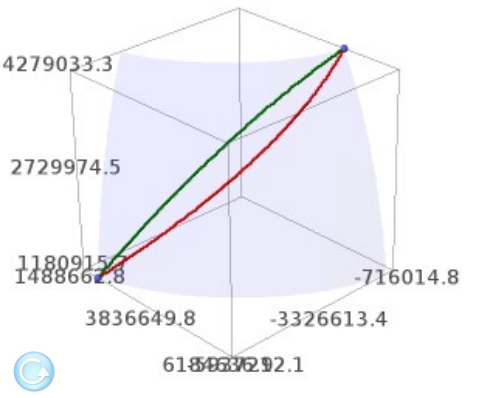
a. Para transatlánticos entre Maracaibo e Vigo.

b. Para voos a 10000 m de altura entre Madrid e Tblisi.

c. Para voos a 8000 m de altura entre Barcelona e Alexandría.

```
print 'a.'
u0=10.6876898
v0=-71.5975723
u1=42.22128
v1=-8.733557
h=0
A=loxeo(u0,v0,u1,v1,h)
show(A[0],figsize=[3,3,3])
```

```
a.
```

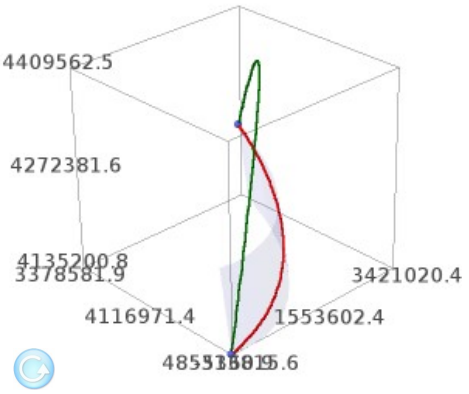


```
print'Rumbo loxodromica=',round(A[1],3),'°'
print'Atallo xeodésico=',round((A[2]-A[3])/1000,2),'km'
```

```
Rumbo loxodromica= 60.264 °
Atallo xeodésico= 81.9 km
```

```
print 'b.'

u0=40.4202805
v0=-3.70577
u1=41.7090797
v1=44.7961197
h=10000
A=loxeo(u0,v0,u1,v1,h)

show(A[0],figsize=[3,3,3])
```

```
b.
```



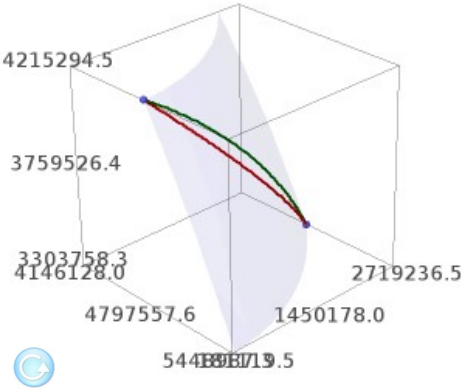

```
print'Rumbo loxodromica=',round(A[1],3),'°'
print'Atallo xeodésico=',round((A[2]-A[3])/1000,2),'km'
```

```
Rumbo loxodromica= 87.981 °
Atallo xeodésico= 32.99 km
```

```
print 'c.'

u0=41.3880386
v0=2.1700101
u1=31.2105007
v1=29.9125309
h=8000
A=loxeo(u0,v0,u1,v1,h)
```

```
show(A[0],figsize=[3,3,3])
```

c.




```
print'Rumbo loxodromica=',round(A[1],3),'°'
print'Atallo xeodésico=',round((A[2]-A[3])/1000,2),'km'
```

```
Rumbo loxodromica= -65.466 °
Atallo xeodésico= 9.42 km
```

Exemplo 25: Curva Braquistócrona

Atopar a curva $\mathrm{x} : [t_1, t_2] \to \mathbb{R}^2$ con $\mathrm{x}(t_1) = \mathrm{a} = (0, a)$ e $\mathrm{x}(t_2) = \mathrm{b} = (b, 0)$, con $a, b > 0$, que é percorrida en menor tempo, por un corpo que comeza no punto a con velocidade 0, e que debe desprazarse ao longo da curva ata chegar ao punto b, baixo a acción da gravidade e sen rozamentos.

Solución:

Supoñamos que a curva admite unha ecuación explícita $y = y(x)$ e debemos extremar o funcional

$$\mathcal{J}(y) = \int_0^b \frac{\sqrt{1 + y'^2}}{\sqrt{2g(a - y)}} dx$$

Xa que logo estamos no terceiro caso particular con

$$F(y,y') = \frac{\sqrt{1+y'^2}}{\sqrt{2g(a-y)}}$$

e, en consecuencia, temos que integrar a ecuación diferencial

$$F - y' \frac{\partial F}{\partial y'} = C$$

que no noso caso é

$$y' = \pm \sqrt{\frac{1}{2gC^2(a-y)} - 1}$$

A cicloide

$$\begin{cases} x = \frac{1}{4gC^2}(\theta - \sin\theta) \\ y = a + \frac{1}{4gC^2}(1 - \cos\theta) \end{cases}$$

cumpre as condicións requeridas con $C^2 = \frac{\cos b - 1}{4ga}$

posto que

$$\frac{dy}{dx} = \frac{\frac{dy}{d\theta}}{\frac{dx}{d\theta}} = \frac{\sin\theta}{1-\cos\theta} = \sqrt{\frac{1+\cos\theta}{1-\cos\theta}} = \sqrt{\frac{1}{2gC^2(a-y)} - 1}$$

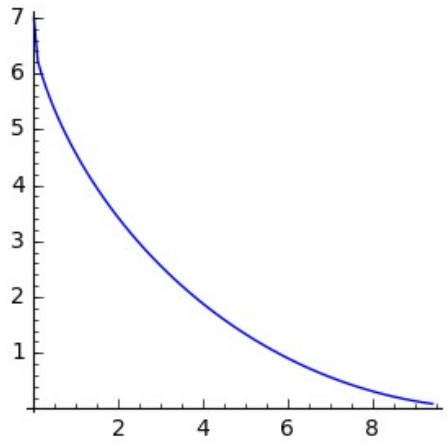
Integrando por métodos numéricos a ecuación

$$y' = \pm \sqrt{\frac{1}{2gC^2(a-y)} - 1}$$

para $a = 7$ e $b = 3\pi$ temos

```
a=6.99
b=(3*pi).n()
g=9.8
C=sqrt((1-cos(b))/(4*g*a))
```

```
f(t,y)=[-sqrt(1/(2*g*C^2*(7-y))-1)]
CI=vector([a])
I=[0,b]
N=100
h=(I[1]-I[0])/N
RK=rungekutta(f,CI,I,N)
T=[0,0+h,..,I[1]]
PT=[]
show(line(zip(T,[a[0] for a in RK])), figsize=[3,3])
```



3.3.10. Principios variacionais en forma hamiltoniana

Veremos aquí como foi maxistralmente xeneralizada por Hamilton a técnica de pasar ao espazo de fases para suscitar problemas mecánicos baixo a forma dunha ecuación diferencial $\mathrm{x}' = f(t, \mathrm{x})$:

Se a enerxía cinética dun sistema é $\mathcal{T}(t, \mathrm{x}, \mathrm{x}')$ e a potencial é $\mathcal{V}(t, \mathrm{x})$, a súa diferenza é a **lagragiana** do sistema

$$L(t, \mathrm{x}, \mathrm{x}') = \mathcal{T}(t, \mathrm{x}, \mathrm{x}') - \mathcal{V}(t, \mathrm{x}).$$

A traxectoria debe vir dada por unha función $\mathrm{x}_0 : [t_1, t_2] \to \mathbb{R}^n$ que sexa punto extremal do funcional

$$\mathcal{J}(\mathrm{x}) = \int_{t_1}^{t_2} L(t, \mathrm{x}(t), (x'(t))dt$$

e, así, debe cumprir a ecuación de Euler-Lagrange

$$\frac{\partial L}{\partial \mathrm{x}_i} - \frac{d}{dt}\Big(\frac{\partial L}{\partial \mathrm{x}_i'}\Big) = 0 \quad \forall i = 1, \cdots, n$$

que debe ser resolto para atopar as compoñentes de $\mathrm{x}_o(t) = (\mathrm{x}_{o1}(t), \cdots, \mathrm{x}_{on}(t))$. Facendo cambio de variables

$$\mathrm{y}_i = \frac{\partial L}{\partial \mathrm{x}_i'} \quad \forall i = 1, \cdots, n$$

pasamos ao sistema de $2n$ ecuacións de primeira orde:

$$(\star) \qquad \begin{cases} \mathrm{y}_i = \frac{\partial L}{\partial \mathrm{x}_i'} \\ \mathrm{y}_i' = \frac{\partial L}{\partial \mathrm{x}_i} \end{cases}$$

Hamilton tivo a idea de considerar a función

$$H = (\mathrm{x}'|\mathrm{y}) - L(t, \mathrm{x}, \mathrm{x}')$$

e non é difícil probar que se pode expresar en termos das coordenadas xeneralizadas $(t, \mathrm{x}, \mathrm{y})$. Así, supoñendo

$$H = H(t, \mathrm{x}, \mathrm{y}),$$

a derivada total de H respecto de t pódese calcular de dous xeitos:

$$\begin{cases} \frac{dH}{dt} = \frac{\partial H}{\partial \mathrm{x}}\mathrm{x}' + \frac{\partial H}{\partial \mathrm{y}}\mathrm{y}' + \frac{\partial H}{\partial t} \\ \frac{dH}{dt} = (\mathrm{x}''|\mathrm{y}) + (\mathrm{x}'|\mathrm{y}') - \frac{\partial L}{\partial \mathrm{x}}\mathrm{x}' - \frac{\partial L}{\partial \mathrm{x}'}\mathrm{x}'' - \frac{\partial L}{\partial t} = -\frac{\partial L}{\partial \mathrm{x}}\mathrm{x}' + (\mathrm{x}'|\mathrm{y}') - \frac{\partial L}{\partial t} \end{cases}$$

e da súa igualdade deducimos que

$$\frac{\partial H}{\partial \mathrm{x}} = -\frac{\partial L}{\partial \mathrm{x}}, \quad \frac{\partial H}{\partial \mathrm{y}} = \mathrm{x}', \quad \frac{\partial H}{\partial t} = -\frac{\partial L}{\partial t}$$

En consecuencia, o sistema $(\star)$ pódese escribir na forma hamiltoniana

$$\begin{cases} \mathrm{x}' = \frac{\partial H}{\partial \mathrm{y}} \\ \mathrm{y}' = -\frac{\partial H}{\partial \mathrm{x}} \end{cases}$$

e considerando o vector $X = \begin{pmatrix} \mathrm{x} \\ \mathrm{y} \end{pmatrix}$ escribilo en forma de ecuación vectorial $X' = f(t, X)$.

A expresión máis xeral da enerxía cinética é

$$\mathcal{T} = (A\mathrm{x}'|\mathrm{x}') + (\mathrm{b}|\mathrm{x}') + c$$

onde $A \in \mathcal{M}_n(\mathbb{R})$, $\mathrm{b} \in \mathbb{R}^n$ e $c \in \mathbb{R}$. Entón,

$$\mathrm{y} = \frac{\partial L}{\partial \mathrm{x}'} = \frac{\partial \mathcal{T}}{\mathrm{x}'} = 2A\mathrm{x}' + \mathrm{b}$$

e, xa que logo,

$$H = 2(A\mathrm{x}'|\mathrm{x}') + (\mathrm{b}|\mathrm{x}') - \mathcal{T} + \mathcal{V} = (A\mathrm{x}'|\mathrm{x}') - c + \mathcal{V}.$$

É frecuente que $\mathrm{b} = \mathrm{o}$ e $c = 0$ e, en tal caso,

$$H = \mathcal{T} + \mathcal{V} = \mathcal{E} = \text{enerxía total do sistema.}$$

A función de **Hamilton** para partículas de masa 1 baixo a acción dalgúns campos de forzas coñecidos é a seguinte:

1. Para un campo constante $k : \mathbb{R}^3 \to \mathbb{R}^3$ temos

$$\begin{cases} \mathcal{T} = \frac{1}{2}(\mathrm{v}|\mathrm{v}) \\ \mathcal{V} = -(\mathrm{x}|k) \end{cases} \quad \text{e, xa que logo,} \quad H = \frac{1}{2}(\mathrm{v}|\mathrm{v}) - (\mathrm{x}|k)$$

2. Para o campo identidade $I : \mathbb{R}^3 \to \mathbb{R}^3$ temos

$$\begin{cases} \mathcal{T} = \frac{1}{2}(\mathrm{v}|\mathrm{v}) \\ \mathcal{V} = -\frac{1}{2}(\mathrm{x}|\mathrm{x}) \end{cases} \quad \text{e, xa que logo,} \quad H = \frac{1}{2}((\mathrm{v}|\mathrm{v}) - (\mathrm{x}|\mathrm{x}))$$

3. Para un campo newtoniano de constante K

$$\begin{array}{rcl} N : \mathbb{R}^3 \backslash \{0\} & \to & \mathbb{R}^3 \\ \mathrm{x} & \mapsto & K \dfrac{\mathrm{x}}{||\mathrm{x}||^3} \end{array}$$

temos

$$\begin{cases} \mathcal{T} = \frac{1}{2}(\mathrm{v}|\mathrm{v}) \\ \mathcal{V} = \frac{-K}{||\mathrm{x}||} \end{cases} \quad \text{e, xa que logo,} \quad H = \frac{1}{2}(\mathrm{v}|\mathrm{v}) - \frac{K}{||\mathrm{x}||}$$

Exemplo 26:

Comprobar que o movemento no baleiro dun péndulo simple de masa m e lonxitude L baixo a acción da gravidade é periódico e o período depende das condicións iniciais.

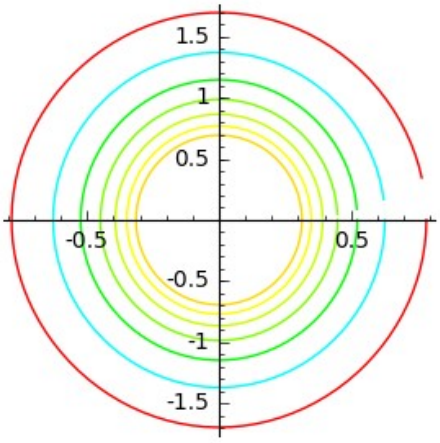
Solución:

Un péndulo simple de $50gr$ de masa e $2m$ de lonxitude que se move no baleiro baixo a acción da gravidade ten enerxía cinética $\mathcal{T} = \frac{1}{2}ms'^2$ sendo s o arco percorrido. Se θ é o ángulo medido desde a vertical, $s = L \cdot \theta$ e $\mathcal{T} = \frac{1}{2}mL^2\theta'^2$. Ademais, se supoñemos o nivel cero de enerxía potencial no punto máis baixo do percorrido, a enerxía potencial é $\mathcal{V} = mgL(1 - \cos\theta)$. Xa que logo, a hamiltoniana do sistema é

$$\mathcal{H} = \mathcal{T} + \mathcal{V} = \frac{1}{2}mL^2\theta'^2 + mgL(1 - \cos\theta)$$

```
g=9.81
m=50
L=2
H(t,x0,x1)=m*L^2*x1^2/2+m*g*L*(1-cos(x0))
q=vector([x0])
p=vector([x1])
Hp=[diff(H,x1)]
Hq=[-diff(H,x0)]
f(t,x0,x1)=Hp+Hq
CIS=[vector([pi/n,0]) for n in [4,5,..,10]]
I=[0,.01425]
N=100
```

```
D=[]
for n in [4,5,..,10]:
    P=rungekutta(f,CIS[n-4],I,N)
    D.append(line(P,hue=1/(n-3)))
show(sum(D),figsize=[3,3])
```

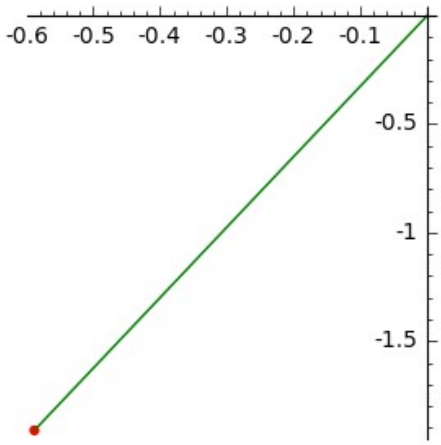


Estas gráficas no espazo de fases proban que os movementos de condicións iniciais

$$(\frac{\pi}{10},0),\quad (\frac{\pi}{9},0),\quad (\frac{\pi}{8},0),\quad (\frac{\pi}{7},0),\quad (\frac{\pi}{6},0),\quad (\frac{\pi}{5},0),\quad (\frac{\pi}{4},0)$$

son periódicos. O primeiro ten un período aproximado de 0.1425 segundos e, os demais teñen períodos maiores. Na seguinte cela de cálculo presentamos a animación do primeiro:

```
CI=vector([pi/10,0])
P=rungekutta(f,CI,I,N)
A=[P[n][0] for n in range(N)]
G=[]
for a in A:
    G.append(line([[0,0],[2*sin(a),-
2*cos(a)]],color='green',thickness=1)+point([2*sin(a),-
2*cos(a)],color='red',pointsize=20))
a=animate(G,xmin=-2,ymin=-2,xmax=2,ymax=0,figsize=[3,3])
show(a[45],figsize=[3,3])
```



3.3.11. Ecuacións intrínsecas de curvas

Unha **curva** en $\mathbb{R}^3$ é unha variedade M 1-dimensional (véxase 3.1.14 de [1]). Se está descrita pola carta local $((a,b),\alpha)$ teremos que $M=\{\alpha(u)\,|\,u\in(a,b)\}\subset\mathbb{R}^3$ pero é claro que se $\phi:(c,d)\to(a,b)$ é un difeomorfismo, $((c,d),\alpha\circ\phi)$ será outra carta local de M. En particular, o difeomorfismo $\phi:(a,b)\to(a,b)$, onde $\phi(u)=a+b-u$, proporciónanos a nova carta local $((a,b),\alpha^\star)$ onde $\alpha^\star(u)=\alpha(a+b-u)$.

En $\mathbf{x}=\alpha(u)$ o espazo tanxente admite a base $D\alpha(u)(1)=\alpha'(u)=\mathbf{t}(u)$. Pero tamén $\mathbf{x}=\alpha^\star(v)$ con $v=a+b-u$ e o espazo tanxente admitirá a base $D\alpha^\star(v)(1)=\alpha^{\star\prime}(v)=(\alpha\circ c)'(v)=-\alpha'(u)=-\mathbf{t}(u)$. Dicimos, xa que logo, que α e $\alpha^\star$ inducen **orientacións** distintas na curva M.

O cambio de lonxitude local xerado pola aplicación $\alpha:(a,b)\to\mathbb{R}^3$ en $\mathbf{x}=\alpha(u)$ determínao a aplicación lineal $D\alpha(u)$ e, como vemos, vale $\|\mathbf{t}(u)\|$. Xa que logo, a lonxitude da curva, desde $\alpha(a)$ ata $\alpha(u)$ pode calcularse como o límite da suma de devanditos cambios locais, é dicir, como a integral

$$\ell(u)=\int_a^u\|\mathbf{t}(u)\|\mathrm{d}u$$

Así, podemos definir a función **lonxitude de arco**:

$$\ell : (a,b) \to (0,L) \quad \text{con} \quad \ell(u) = \int_a^u \|\mathbf{t}(u)\| \mathrm{d}u \quad \text{e} \quad \ell'(u) = \|\mathbf{t}(u)\| > 0$$

Esta función é crecente e, xa que logo, a súa función inversa $\ell^{-1} : (0,L) \to (a,b)$ está ben definida e é derivable, aínda que tanto ℓ como ℓ^{-1} poden non ser expresables en termos elementais. A función composta $\gamma = \alpha \circ \ell^{-1}$ proporciónanos unha nova carta local $\gamma : (0,L) \to \mathbb{R}^3$ de M que induce a mesma orientación que α pois

$$\mathbf{t}(s) = \gamma'(s) = (\alpha \circ \ell^{-1})'(s) = \frac{\alpha'(u)}{\|\alpha'(u)\|} = \frac{\mathbf{t}(u)}{\|\mathbf{t}(u)\|}$$

A vantaxe xeométrica da carta canónica $((0,L),\gamma)$ fronte á $((a,b),\alpha)$ é grande e estriba en que o vector $\gamma'(s)$ é unitario. Así pois, o vector $\mathbf{t}(s)$ só cambia de dirección e por iso é razoable chamar **curvatura** á magnitude $\kappa(s) = \|\mathbf{t}'(s)\|$. Ademais, se M fose unha variedade de clase $\mathcal{C}^2$ poderiamos derivar $(\mathbf{t}(s)|\mathbf{t}(s)) = 1$ obtendo que $\mathbf{t}'(s) \perp \mathbf{t}(s)$. Aparece así un novo vector unitario

$$\mathbf{n}(s) := \frac{\mathbf{t}'(s)}{\|\mathbf{t}'(s)\|}$$

chamado vector **normal**, que destaca entre todos os vectores perpendiculares a $\mathbf{t}(s)$. Definindo o vector **binormal** de modo que $\mathbf{b}(s) := \mathbf{t}(s) \wedge \mathbf{n}(s)$, temos en cada punto $\gamma(s)$ da curva, a base ortonormal de **Frenet** $\{\mathbf{t},\mathbf{n},\mathbf{b}\}(s)$ intrinsecamente ligada á xeometría da curva.
Esta base determina o **triedro de Frenet** constituído polo plano **normal** $[\mathbf{t}(s)]^\perp$, o **rectificante** $[\mathbf{n}(s)]^\perp$ e o **osculador** $[\mathbf{b}(s)]^\perp$.

Podemos expresar calquera vector $\mathbf{y} \in \mathbb{R}^3$ respecto da base de Frenet suprimindo a dependencia do arco para alixeirar a notación:
$\mathbf{y} = (\mathbf{y}|\mathbf{t})\mathbf{t} + (\mathbf{y}|\mathbf{n})\mathbf{n} + (\mathbf{y}|\mathbf{b})\mathbf{b}$ e falaremos do seu compoñente tanxencial $(\mathbf{y}|\mathbf{t})$, o seu compoñente normal $(\mathbf{y}|\mathbf{n})$ ou o seu compoñente binormal $(\mathbf{y}|\mathbf{b})$. Cando M sexa de clase $\mathcal{C}^3$ é particularmente interesante a expresión de $\mathbf{t}'$, $\mathbf{n}'$ e $\mathbf{b}'$:

Como $\mathbf{t}$, $\mathbf{n}$ e $\mathbf{b}$ son unitarios, $\quad (\mathbf{t}'|\mathbf{t}) = (\mathbf{n}'|\mathbf{n}) = (\mathbf{b}'|\mathbf{b}) = 0.$.

Ademais,

$$\begin{array}{rcrcl} (\mathbf{t}|\mathbf{n}) = 0 & \Rightarrow & (\mathbf{t}'|\mathbf{n}) = -(\mathbf{n}'|\mathbf{t}) & = & \kappa \\ (\mathbf{t}|\mathbf{b}) = 0 & \Rightarrow & (\mathbf{t}'|\mathbf{b}) = -(\mathbf{b}'|\mathbf{t}) & = & 0 \\ (\mathbf{n}|\mathbf{b}) = 0 & \Rightarrow & (\mathbf{n}'|\mathbf{b}) = -(\mathbf{b}'|\mathbf{n}) & := & \tau \end{array}$$

Así, obtemos as ecuacións diferenciais lineais ou **fórmulas de Frenet**:

$$\begin{array}{rcccccc} \mathbf{t}' = & & & & \kappa\mathbf{n} & & \\ \mathbf{n}' = & -\kappa\mathbf{t} & + & & & + & \tau\mathbf{b} \\ \mathbf{b}' = & & & -\tau\mathbf{n} & & & \end{array}$$

A función $\tau(s) = (\mathbf{n}'|\mathbf{b})(s) = -(\mathbf{b}'|\mathbf{n})(s)$ mide a variación do plano osculador posto que $|\tau(s)| = \|\mathbf{b}'(s)\|$, é dicir, mide a non planariedade da curva, o que se torce, e por iso chámase **torsión**. É fácil calculala:

$$\tau(s) = -(\mathbf{b}'|\mathbf{n})(s) = -\left((\mathbf{t}\wedge\mathbf{n})'|\mathbf{n}\right)(s) = [\mathbf{t},\mathbf{n},\mathbf{n}'](s) = \frac{[\gamma',\gamma'',\gamma''']}{\kappa^2}(s)$$

Coñecidas $\kappa(s)$ e $\tau(s)$, a integración das ecuacións de Frenet proporciónanos o coñecemento xeral da base $\{\mathbf{t},\mathbf{n},\mathbf{b}\}(s)$ e, polo tanto, o da función $\gamma(s)$ salvo unhas constantes de integración que poderemos determinar se impoñemos condicións iniciais. É dicir, a curvatura e a torsión proporciónannos o coñecemento da curva $\mathbf{x} = \gamma(s)$ salvo xiros e traslacións.

Cando $\tau(s) = 0$ a curva é plana e podemos considerala contida en $\mathbb{R}^2$. As ecuacións de Frenet redúcense a:

$$\begin{aligned} \mathbf{t}' &= \kappa\mathbf{n} \\ \mathbf{n}' &= -\kappa\mathbf{t} \end{aligned}$$

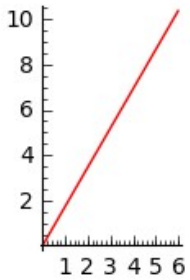
e son facilmente integrables.

Exemplo 27:

Representar a curva plana con $\kappa(s) = 0$ que parte de $(0,0)$ e en devandito

punto ten tanxente $(\cos(\alpha), \sin(\alpha))$ e normal $(-\sin(\alpha), \cos(\alpha))$.

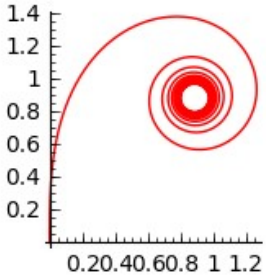
```
k(s)=0
f(s,t1,t2,n1,n2)=[k(s)*n1,k(s)*n2,-k(s)*t1,-k(s)*t2]
al=pi/3
CIT=vector([cos(al),sin(al),-sin(al),cos(al)])
I=[0,12]; N=100*I[1]; h=I[1]/N
F=rungekutta(f,CIT,I,N)
FT=[vector([a[0],a[1]]) for a in F]
FTN=[a/norm(a) for a in FT]
PI=vector([0,0])
C=[PI]
for k in range(N-2):
    C.append(C[-1]+vector((h*(FTN[k]+FTN[k+1])/2).n()))
show(line(C[0:N],color='red',aspect_ratio=1), figsize=[3,2])
```



Exemplo 28: Espiral de Cornu

Representar a curva plana con $\kappa(s) = s$ que parte do punto $(0,0)$ e en devandito punto ten tanxente $(0,1)$ e normal $(1,0)$.

```
k(s)=s
f(s,t1,t2,n1,n2)=[k(s)*n1,k(s)*n2,-k(s)*t1,-k(s)*t2]
CIT=vector([0,1,1,0])
I=[0,12]; N=100*I[1]; h=I[1]/N
F=rungekutta(f,CIT,I,N)
FT=[vector([a[0],a[1]]) for a in F]
FTN=[a/norm(a) for a in FT]
PI=vector([0,0]); C=[PI]
for k in range(N-2):
    C.append(C[-1]+vector((h*(FTN[k]+FTN[k+1])/2).n()))
show(line(C[0:N],color='red',aspect_ratio=1), figsize=
[2.5,2.5])
```



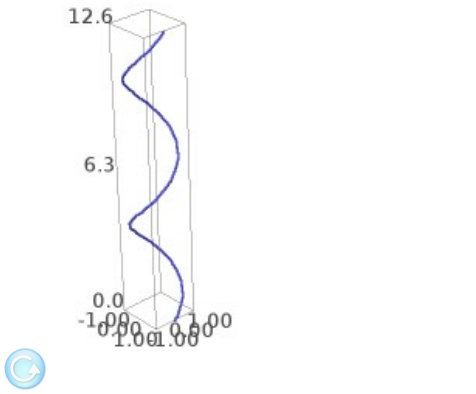
Cando $\tau(s)$ non é identicamente nula a curva non é plana e as ecuacións de Frenet son as xerais anteriormente expresadas que consisten nun sistema de nove ecuacións lineais e nove incógnitas que tamén podemos integrar usando **rungekutta()**.

Exemplo 29:

Calcular a curvatura e a torsión de $\alpha : (0, 4\pi) \to \mathbb{R}^3$ con $\alpha(u) = (\cos(u), \sin(u), u)$.

Calcular o punto $\alpha(0)$ e os vectores tanxente, normal e binormal nel.

```
a(u)=[cos(u),sin(u),u]
show(parametric_plot3d(a,(u,0,4*pi),thickness=3),
aspect_ratio=[1,1,1],figsize=[3,3,3])
```



A función lonxitude de arco é

```
assume(u>0)
l(u)=integrate(norma(diff(a(u),u)),u,0,u)
print 'l(u)=', l(u)
```

```
l(u)= sqrt(2)*u
```

e a ecuación canónica da curva é $g : (0, 4\pi\sqrt{2}) \to \mathbb{R}^3$ con

```
g(s)=RS([cos(s/sqrt(2)),sin(s/sqrt(2)),s/sqrt(2)])
print 'g(s)='
show(g(s))
```

```
g(s)=
```

$$\left(\cos\left(\frac{1}{2}\sqrt{2}s\right), \sin\left(\frac{1}{2}\sqrt{2}s\right), \frac{1}{2}\sqrt{2}s\right)$$

A partir dela podemos achar facilmente a curvatura e a torsión:

```
g1(s)=list(diff(g(s),s))
g2(s)=list(diff(g1(s),s))
g3(s)=list(diff(g2(s),s))
K(s)=norma(diff(g1(s),s)).full_simplify()
TOR(s)=(g1(s)*
(g2(s).cross_product(g3(s)))/K(s)^2).full_simplify()
print 'K(s)=',K(s)
print 'TOR(s)=', TOR(s)
```

```
K(s)= 1/2
TOR(s)= 1/2
```

A posición inicial Xo e os vectores tanxente To, normal No, e binormal Bo, son:

```
Xo=g(0)
print 'Xo=', g(0)
To=g1(0)
print 'To=', To
No=g2(0)/norm(g2(0))
print 'No=', No
Bo=To.cross_product(No)
print 'Bo=', Bo
```

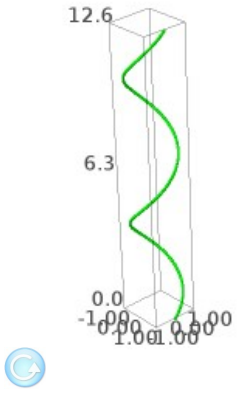
```
Xo= (1, 0, 0)
To= (0, 1/2*sqrt(2), 1/2*sqrt(2))
No= (-1, 0, 0)
```

```
Bo= (0, -1/2*sqrt(2), 1/2*sqrt(2))
```

Exemplo 30:

Resolver o recíproco do exemplo anterior; é dicir, representar a curva de $\mathbb{R}^3$ de $\kappa(s) = 1/2$ e $\tau = 1/2$ que parte do punto Xo e nel ten tanxente, normal e binormal To, No e Bo.

```
K=1/2
TOR=1/2
f(s,t1,t2,t3,n1,n2,n3,b1,b2,b3)=[K*n1,K*n2,K*n3,TOR*b1-
K*t1,TOR*b2-K*t2,TOR*b3-K*t3,-TOR*n1,-TOR*n2,-TOR*n3]
CI=vector(list(To)+list(No)+list(Bo))
I=[0,4*pi*sqrt(2)]
N=100*ceil(I[1]-I[0])
h=(I[1]-I[0])/N
F=rungekutta(f,CI,I,N)
Tan=[a[0:3].n() for a in F]
C=[Xo]
for k in range(N-2):
    C.append(C[-1]+(h*(Tan[k]+Tan[k+1])/2).n())
show(line3d(C[0:N],rgbcolor=
(0,1,0),thickness=3,aspect_ratio=1),figsize=[3,3,3])
```



3.4. Exercicios

Exercicio 1:

Sexa unha familia de rectángulos de lados paralelos aos eixes e un vértice fixo na orixe. Por que curva do primeiro cuadrante debe moverse o vértice oposto á orixe para que o ritmo de cambio da área A con respecto a x sexa A?

Solución:

Supoñemos que a curva podemos dala como $y = y(x)$

Como $A'(x) = A$ temos que $y + xy' = xy$

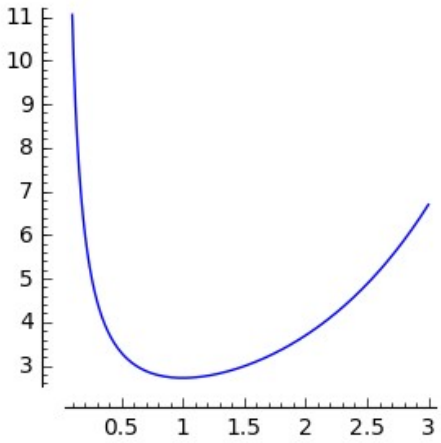
Esta é unha ecuación diferencial en variables separadas que podemos escribir na forma

$$\frac{dy}{y} = (1 - \frac{1}{x})dx$$

cuxa solución é $log\, y = x - log\, x$

Xa que logo a curva é $y = \frac{e^x}{x}$.

```
f(x)=exp(x)/x
show(plot(f,(x,0.1,3)),figsize=[3,3])
```



Exercicio 2:

Achar as curvas que en cada punto verifican que a proxección sobre o eixe OX do segmento da normal entre o punto e o eixe OX mide 1.

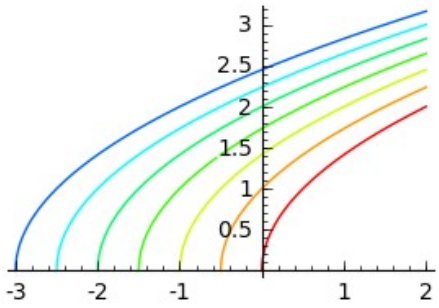
Solución:

Se cada curva admite a ecuación explícita $y = y(x)$, débese cumprir a ecuación vectorial

$$\begin{pmatrix} x \\ y \end{pmatrix} + m \begin{pmatrix} -y' \\ 1 \end{pmatrix} = \begin{pmatrix} x+1 \\ 0 \end{pmatrix}$$

Xa que logo $x + yy' = x + 1$ que é unha ecuación diferencial en variables separadas. A familia de curvas é $y^2 = 2x + k$. Presentamos algunhas parábolas da familia para diferentes valores de k.

```
F=[]
for K in range(7):
    f(x)=sqrt(2*x+K)
    F.append(plot(f,(x,-K/2,2),hue=K/10,aspect_ratio=1))
show(sum(F),figsize=[3,3])
```



Exercicio 3:

Achar as curvas que en cada punto verifican que o punto medio do punto e o pé da normal no eixe OX está sobre a parábola $y^2 = x$

Solución:

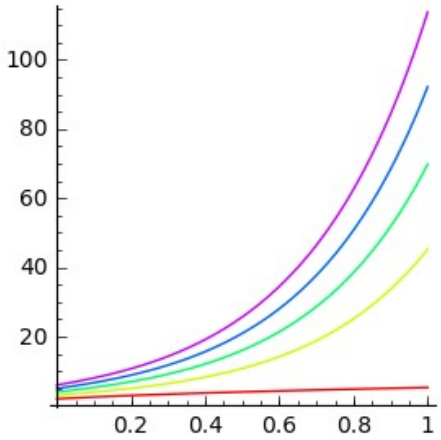
Supoñendo que cada curva da familia admite a ecuación explícita $y = y(x)$ e seguindo os pasos do exercicio anterior deducimos que o punto medio ten as

coordenadas $\left(\frac{2x+yy'}{2}, \frac{y}{2}\right)$. Debe cumprir a ecuación diferencial

$$y' = \frac{y^2 - 4x}{2y}$$

Como non é unha ecuación diferencial en variables separadas utilizaremos a nosa función **rungekutta()** para presentar algunhas curvas da familia.

```
f(x,y)=[(y^2-4*x)/(2*y)]
D=[]
for K in range(5):
    CI=vector([K+2])
    I=[0,6]
    N=60
    T=[0,1/60,..,I[1]]
    R=rungekutta(f,CI,I,N)
    S=[a[0] for a in R]
    D.append(line(zip(T,S),hue=2*K/10))
show(sum(D),figsize=[3,3])
```

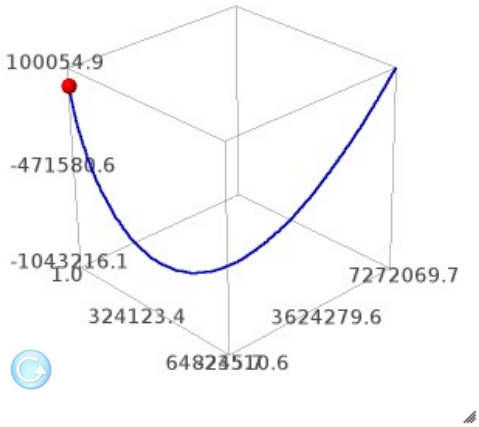


Exercicio 4:

Estudar o movemento no baleiro dunha partícula de masa 10 e $CI = [1, 7, 3, 1, 1, 2]$ baixo a acción do campo lineal non conservativo

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & -1 \\ 0 & 1 & 1 \end{pmatrix}$$

```
f(t,x1,x2,x3,v1,v2,v3)=[v1,v2,v3,.1*x1,.1*(x2-x3),.1*(x2+x3)]
CI=vector([1,7,3,1,1,2])
I=[0,40]
N=6*I[1]
T=[0,1/6,..,I[1]]
S=rungekutta(f,CI,I,N)
g=line3d([a[0:3] for a in S],color='blue',thickness=.5)
particula=[]
for k in range(len(T)):
    particula.append(g+point3d(S[k][0:3],color='red',size=20))
a=animate(particula)
show(a[8],figsize=[3,3,3])
#a.show(delay=10)
```



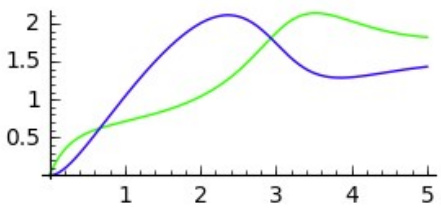
Exercicio 5:

Un proceso Brusselator segue o seguinte esquema:

$$\begin{cases} A \to X \quad \text{con} \quad k_1 = 1.2 \\ B + X \to Y + C \quad \text{con} \quad k_2 = 0.8 \\ 2X + Y \to 3X \quad \text{con} \quad k_3 = 0.9 \\ X \to D \quad \text{con} \quad k_4 = 1.3 \end{cases}$$

onde son constantes as concentracións de A, 2, B, 3, C, 4 e D, 6. Achar as concentracións de X e Y en función do tempo sabendo que no instante inicial son nulas.

```
K=[1.2,0.8,0.9,1.3]
P=[2,3,4,6]
f(t,x,y)=[K[0]*P[0]-K[1]*P[1]*x+K[2]*x^2*y-K[3]*x,K[1]*P[1]*x-
K[2]*x^2*y]
CI=vector([0,0])
I=[0,5]
N=300
T=[0,1/60,..,5]
S=rungekutta(f,CI,I,N)
X=[a[0] for a in S]
Y=[a[1] for a in S]
GX=line(zip(T,X),hue=0.3,aspect_ratio=1)
GY=line(zip(T,Y),hue=0.7,aspect_ratio=1)
show(GX+GY,figsize=[3,3])
```



Exercicio 6:

As ecuacións de Lotka-Volterra para dinámicas de poboacións que compiten por un mesmo recurso, son similares ás dos modelos de depredador-presa. A ecuación de evolución de cada especie ten un termo de iteración propia e outros termos de iteración coas demais especies, e son do tipo:

$$\frac{dx_i}{dt} = r_i x_i (1 - \frac{\sum_{j=1}^{N} \alpha_{ij} x_j}{K_i}),\ i = 1, \cdots, N$$

donde $\alpha_{ij} \geq 0$.

Un exemplo para 4 especies ven dado pola matriz de coeficientes (α_{ij}):

$$\begin{pmatrix} 1 & 1.09 & 1.52 & 0 \\ 0 & 1 & 0.44 & 1.36 \\ 2.33 & 0 & 1 & 0.47 \\ 1.21 & 0.51 & 0.35 & 1 \end{pmatrix}$$

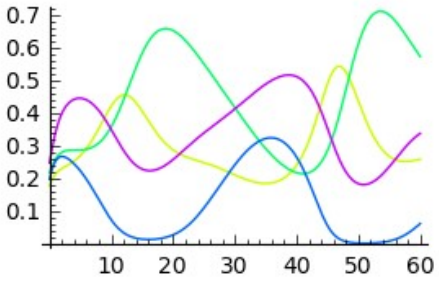
e o vector $(r_i) = (1, 0.72, 1.53, 1.27)$ con $K_i = 1,\ i = 1, \cdots, 4$.

Estudar a evolución deste ecosistema durante 60 anos partindo dunhas condicións iniciais de $CI = [0.18, 0.22, 0.2, 0.25]$ (entendemos en miles de individuos).

a. Representar as curvas de evolución das 4 especies nese período de tempo, nunha mesma gráfica.
b. Determinar a especie que acada o número máximo de individuos nese periódo e o número de individuos que acada.

```
print 'a.'
f(t,x1,x2,x3,x4)=[x1*(1-(x1+1.09*x2+1.52*x3)),0.72*x2*
(1-(x2+0.44*x3+1.36*x4)),1.53*x3*
(1-(2.33*x1+x3+0.47*x4)),1.27*x4*
(1-(1.21*x1+0.51*x2+0.35*x3+x4))]
CI=vector([0.18,0.22,0.2,0.25])
I=[0,60]
N=12*I[1]
S=rungekutta(f,CI,I,N)
T=[0,1/12,..,60]
X1=[a[0] for a in S]
GX1=line(zip(T,X1),hue=.2)
X2=[a[1] for a in S]
GX2=line(zip(T,X2),hue=.4)
X3=[a[2] for a in S]
GX3=line(zip(T,X3),hue=.6)
X4=[a[3] for a in S]
GX4=line(zip(T,X4),hue=.8)
show(GX1+GX2+GX3+GX4,figsize=[3,2])
```

```
a.
```




```
print'b.'
[M,K]=maximos(X2)
[M,K]
```

```
b.
[0.712646329256942, [643]]
```

Como vemos na gráfica, a especie 2 é a que acada maior poboación con 712 individuos aos 53 anos e 7 meses.

Exercicio 7:

Nunha poboación de 800 individuos con crecemento loxístico de capacidade de carga 1000, declárase unha epidemia grave con mortes e inmunidade temporal, gobernada polas ecuacións diferenciais

$$\begin{cases} x' = 0.01x - 0.00001x^2 - 0.0004xy + 0.0001z \\ y' = 0.0004xy - 0.208y \\ z' = 0.2y - 0.0001z \\ m' = 0.008y \end{cases}$$

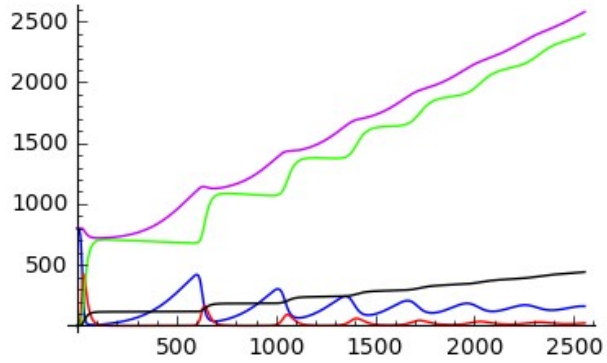
onde $x(t)$, $y(t)$, $z(t)$ e $m(t)$ é o número de sans, enfermos, inmunizados e mortos pola enfermidade no día t.

a. Faise endémica a enfermidade?

b. A epidemia fai que se supere co tempo a capacidade de carga da poboación?

c. En canto tempo se triplica a poboación?

```
f(t,x,y,z,m)=
[.008*x-.00001*x^2-.0004*x*y+.0001*z,.0004*x*y-.058*y,.05*y-.0001*z
CI=vector([799,1,0,0])
I=[0,365*7]; N=24*I[1]
T=[0,1/24,..,I[1]]; S=rungekutta(f,CI,I,N)
X=Round([a[0] for a in S],5)
Y=Round([a[1] for a in S],5)
Z=Round([a[2] for a in S],5)
M=Round([a[3] for a in S],5)
V=Round([sum(a[0:3]) for a in S],5)
show(line(zip(T,X))+line(zip(T,Y),hue=0)+line(zip(T,Z),hue=.3)+
line(zip(T,M),color='black')+ line(zip(T,V),hue=.8),figsize=
[4,2.5])
```




```
print 'a.Como ao final do período hai máis de', floor(Y[-1]),
'enfermos,'
print '   é unha endemia.'
print 'b.A gráfica púrpura chega a superar os 1000 individuos.'
D=len([a for a in V if a<2400])/24
print 'c.A poboación triplícase no día',floor(D)
```

```
a.Como ao final do período hai máis de 21 enfermos,
   é unha endemia.
b.A gráfica púrpura chega a superar os 1000 individuos
c.A poboación triplícase no día 2324
```

Exercicio 8:

Desde o centro xeométrico do Santiago Bernabeu, Messi prepárase para disparar a porta. Sabe que se o seu chut chega ao marco con celeridade inferior a 20m/s, o porteiro párao seguro. Achar unha velocidade inicial do balón que posibilite un gol por todo o escadro.

Solución:

Internet dinos que as medidas do Bernabeu son 105×68, as das súas porterías 7.32×2.44, o peso do balón é 450 g e o seu radio 11cm. Supoñemos que non hai vento e que o coeficiente de viscosidade do aire é 0.003.

Tendo en conta o rozamento do aire, a segunda lei de Newton dános a ecuación diferencial

$$x'' = \frac{F}{m} - \frac{k}{m}v$$

sendo F a forza gravitatoria, k o coeficiente de viscosidade do aire e m a masa do balón. Utilizando o método de **Runge-Kutta** e deixando o vector velocidade inicial en función dos parámetros c_1, c_2, c_3, tras algúns experimentos, atopamos a seguinte solución

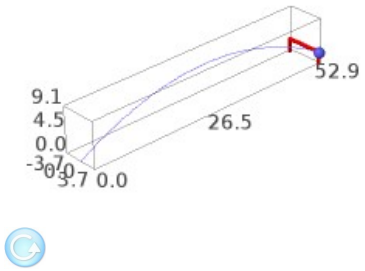
```
k=.003
m=.45
g=9.81
I=[0,3]
N=I[1]*30
(c1,c2,c3)=(3.8,4,2.55)
CI=vector([0,0,0,c1*0.366,c2*5.25,c3*5.25])
f(t,x1,x2,x3,v1,v2,v3)=[v1,v2,v3,k*v1/m,-k*v2/m,-k*v3/m-g]
S=rungekutta(f,CI,I,N)
D=[Round([a[0],a[1],a[2],norma(a[3:])],2) for a in S if a[1]<53
]
print Round([c1*0.366,c2*5.25,c3*5.25],2)
print D[-1]
```

```
    [1.39, 21.0, 13.39]
    [3.55, 52.75, 2.33, 23.67]
```

que nos indica que se o vector velocidade inicial é $(1.39, 21, 13.39)$ o centro do balón, se non o para o porteiro, estará no punto $(3.55, 52.75, 2.33)$ con velocidade de $23.67 m/s$

```
B=[[a[0],a[1],a[2]] for a in D]
POR=[[-3.66,52.5,0],[-3.66,52.5,2.44],[3.66,52.5,2.44],
[3.66,52.5,0]]
show(line3d(B)+line3d(POR,color='red',thickness=5)+
point3d([3.48,52.93,2.11],size=15),aspect_ratio=1,figsize=
```

```
[3,3,3])
```

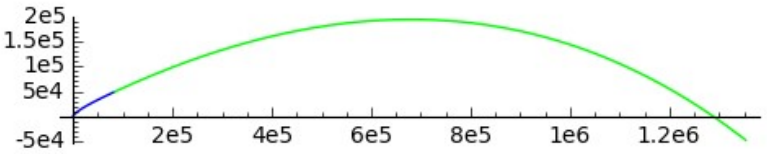


Exercicio 9:

Desde [0,0] lanzamos un foguete de 300 kg de carga e 3000 kg de propulsante que se consome uniformemente en 80 segundos producindo gases que saen del a 2000 m/s. Se a lanzadeira lle proporciona unha velocidade inicial [1,12], a onde chega o foguete? e, en canto tempo?

```
var('t x0 x1 x2 x3')
u=[x0,x1]
U=vector(u)
v=[x2,x3]
V=vector(v)
b=.3
G=vector([0,-9.8])
K=300; P=3000
vg=2000; T=80
mpri=P/T
m=K+P*(T-t)/T
CV=(-b+vg*mpri/norma(V))/m
VPRI=CV*V+G
f(t,x0,x1,x2,x3)=[V[0],V[1],VPRI[0],VPRI[1]]
CI=vector([0,0,1,12])
I=[1,80]; N=200
V2=rungekutta(f,CI,I,N)
P=[[V2[k][0],V2[k][1]] for k in range(N)]
DP=line(P,aspect_ratio=1)
```

```
CI=V2[200]
I=[80,480]
f(t,x0,x1,x2,x3)=[x2,x3,-(b/K)*x2,-9.8-(b/K)*x3]
RR=rungekutta(f,CI,I,N)
PP=[[RR[k][0],RR[k][1]] for k in range(N)]
DPP=line(PP,rgbcolor=(0,1,0))
show(DP+DPP,figsize=[5,4])
```



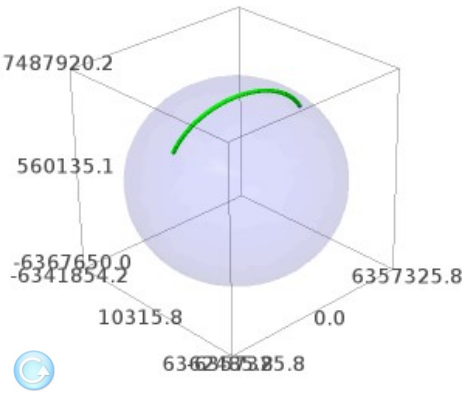
Exercicio 10:

Desde Moscú (55.7522202, 37.6155586) lánzase un $V2$ con $K = 17700$ kg de carga e $P = 275200$ kg de propulsante que se consome uniformemente en $T = 120$ segundos producindo gases que saen del a $v_g = 2700$ m/s.

Se o radio da Terra é $R = 6367650$ Km e o coeficiente de viscosidade do aire é $b = 1.73 \cdot 10^{-5}$, determinar as condicións iniciais do lanzamento para alcanzar Washington sabendo que a velocidade vertical inicial é de 9 m/s.

```
var('t x y z u v w')
R=6367650
ke=-0.0007117
kn=.174685
LAP=55.7522202*pi/180
LOP=37.6155586*pi/180
x0=(R*cos(LAP)*cos(LOP)).n()
y0=(R*cos(LAP)*sin(LOP)).n()
z0=(R*sin(LAP)).n()
ve=ke*
((pi/(12*3600))*vector([-R*cos(LAP)*sin(LOP),R*cos(LAP)*cos(LOP),0]
vn=kn*vector([-sin(LAP)*cos(LOP),-
sin(LAP)*sin(LOP),cos(LOP)]).n()
vz=9*vector([cos(LAP)*cos(LOP),cos(LAP)*sin(LOP),sin(LAP)]).n()
v0=ve+vn+vz
G=6.67*10^(-11)
U=vector([x,y,z])
V=vector([u,v,w])
M=5.972*10^(24)
b=1.73*10^(-5)
```

```
K=17700
P=275200
vg=2700
T=120
m=K+P*(T-t)/T
mpri=-P/T
FG=-G*M/(norma(U)^3)*U
CV=((-mpri*vg-b*norma(V))/(m*norma(V)))*V
VPRI=CV+FG
I=[0,120]
N=360
CI=vector([x0,y0,z0,v0[0],v0[1],v0[2]])
f(t,x,y,z,u,v,w)=[u,v,w,VPRI[0],VPRI[1],VPRI[2]]
V2=rungekutta(f,CI,I,N)
IP=[[V2[k][0],V2[k][1],V2[k][2]] for k in range(N)]
DP=line3d(IP,thickness=5)
esf(fi,teta)=[R*cos(fi)*cos(teta),R*cos(fi)*sin(teta),R*sin(fi)]
ES=parametric_plot3d(esf,(fi,-pi/2,pi
/2),(teta,0,2*pi),opacity=.1)
CI2=V2[-1]
I2=[120,2080]
N2=3500
f2(t,x,y,z,u,v,w)=[u,v,w,FG[0]-b*u/K,FG[1]-b*v/K,FG[2]-b*w/K]
RR=rungekutta(f2,CI2,I2,N2)
PP=[[RR[k][0],RR[k][1],RR[k][2]] for k in range(N2) if
norma(vector([RR[k][0],RR[k][1],RR[k][2]])).n()>R ]
DPP=line3d(PP,rgbcolor=(0,1,0),thickness=5)
show(ES+DP+DPP,figsize=[3,3,3])
```



Podemos achar as coordenadas xeográficas do punto de impacto

```
tempo=(I[1]+len(PP)/2).n()
Xiro=tempo*180/(12*3600)
R0=norma(vector(PP[-1]))
LALLR=(asin(PP[-1][2]/R0)).n()
LOLLR=acos(PP[-1][0]/(R0*cos(LALLR)).n())
if abs(sin(LOLLR)-(PP[-1][1]/(R0*cos(LALLR))).n())>10^(-4):
    LOLLR=-LOLLR
CGLL=[(LALLR*180/pi).n(), (LOLLR*180/pi).n()]
CGIM=[CGLL[0],CGLL[1]+Xiro]
CGIM
```

```
[38.8694928998564, -77.0534522548272]
```

Introducindo na páxina web www.mundivideo.com as coordenadas obtidas visualizamos o lugar do impacto

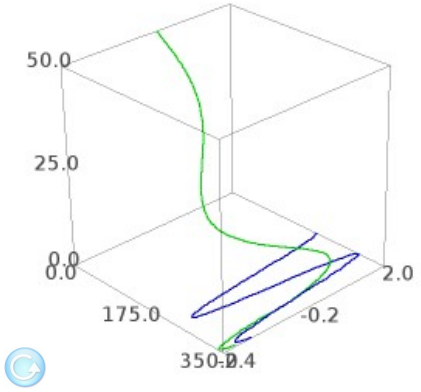
Exercicio 11:

Unha aguia está no punto (0,0,100) e un coello no punto (200,2,0). Este, ao notar a presenza da aguia, foxe cara ao seu tobo situado en $(350, 2\cos 10, 0)$, con movemento de ecuación $c(t) = (20015t, 2\cos t, 0)$. Nese mesmo instante, a aguia inicia a persecución do coello, con celeridade de 40 m/s.

Representar o movemento da presa e o depredador.

Indicar se o coello é alcanzado antes de chegar ao tobo e, no seu caso, determinar o instante de captura.

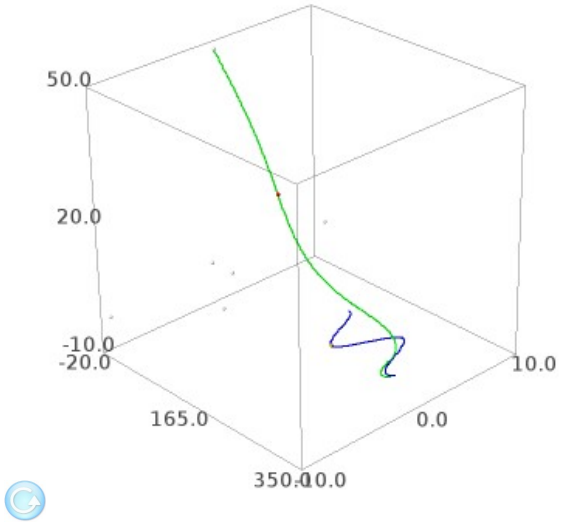
```
var('x0 x1 x2'); X=vector([x0,x1,x2])
c(t)=[200+15*t,2*cos(t),0]
f(t,x0,x1,x2)=list(40*((c(t)-X)/norma(c(t)-X)))
CI=vector([0,0,50]); I=[0,10]; N=200
S=rungekutta(f,CI,I,N)
g=line3d(S,rgbcolor=(0,1,0),thickness=2)
a=parametric_plot3d(c,(t,0,I[1]),rgbcolor=(0,0,1),thickness=2)
show(g+a,figsize=[3,3,3])
```




```
FXI=point3d([-20,0,0],color='white')
FXD=point3d([20,0,0],color='white')
FYI=point3d([0,-10,0],color='white')
FYD=point3d([0,10,0],color='white')
FZB=point3d([0,0,50],color='white')
FZA=point3d([0,0,-10],color='white')
A=[]; k=0; R=[0,.2,..,10]
for r in R:
    P=point3d(list(c(t=r)),color='orange',size=5)
    Q=point3d(list(S[4*k]),color='red',size=5)
    A.append(FXI+FYI+FZB+g+a+P+Q+FXD+FYD+FZA)
    if norm(c(t=r)-S[4*k]).n()<.5:
        print 'captura no instante t=',r
        break
    k=k+1
```

```
an=animate(A); #show(an)
show(an[15],figsize=[4,3,4])
```

captura no instante t= 8.20000000000000



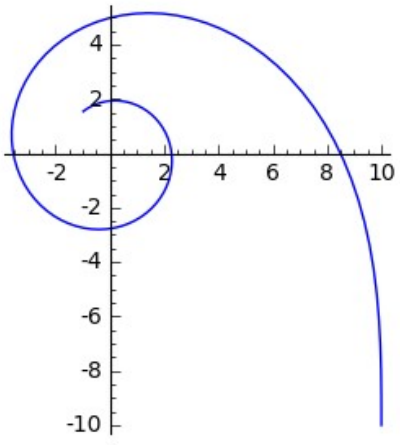
Exercicio 12:

Un tractor leva ligada, mediante unha articulación esférica, unha vara de $10m$ de lonxitude, acabada nunha bóla, e percorre, nunha praia, unha circunferencia de centro $(0,0)$ e $10m$ de radio a $1m/s$. Se no instante inicial o tractor está en $(10,0)$ e a bóla en $(10,-10)$, cando estará a bóla a $2m$ da orixe?

```
f(t,u0,u1)=[(-u0^2*sin(t/10)+u0*u1*cos(t/10)+sin(t/10))/10,
(-u0*u1*sin(t/10)+u1^2*cos(t/10)-cos(t/10))/10]
CI=vector([0,-1]); I=[0,100]; N=100
P=rungekutta(f,CI,I,N)
T=range(N)
C=line([vector([10*cos(t/10),10*sin(t/10)])+10*P[t] for t in T])
show(C,aspect_ratio=1,figsize=[3,3])
V=[vector([10*cos(t/10),10*sin(t/10)])+10*P[t] for t in T]
D=[norma(v).n() for v in V]
print 'D[89:91]=',D[89:91]
print 'Logo t=',89.497487437185930,'s'
```

D[89:91]= [2.00996936323504, 1.99006951592709]

```
Logo t= 89.49748743718593 s
```

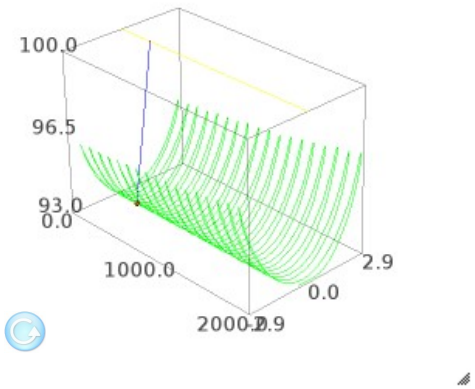


Exercicio 13:

Un helicóptero do servizo de extinción de incendios voa a $100m$ de altura sobre o eixe X con velocidade uniforme $20m/s$ portando unha articulación esférica de $1m^3$ de auga ligada ao helicóptero mediante unha vara de $7m$. Se a cantidade de fume é tal que o coeficiente de viscosidade do aire é $b = 0.2$ e no instante inicial a posición e a velocidade relativa da carga respecto ao helicóptero son $[0, 0, -1]$ e $[1, 1/2, 0]$, animar o movemento do helicóptero e a carga no primeiro minuto de voo.

```
var('t x0 x1 x2 x3 x4 x5')
C=vector([20*t,0,100])
DC=diff(C,t)
L=7
b=.2
m=2000
u=[x0,x1,x2]
v=[x3,x4,x5]
U=vector(u)
V=vector(v)
FI=vector([0,0,-9.8])
K=((-m*L*(V*V)+b*(DC*U)-m*(FI*U))*U-b*DC-b*L*V+m*FI)/(m*L)
f(t,x0,x1,x2,x3,x4,x5)=v+list(K)
CI=vector([0,0,-1,1,.5,0])
I=[0,100]
N=300
```

```
h=(I[1]-I[0])/N
S=rungekutta(f,CI,I,N)
C1=[C(t=k*h) for k in [0..N]]
S1=[C(t=k*h)+L*S[k][0:3] for k in [0..N]]
g=line3d(S1,rgbcolor=(0,1,0),thickness=1)
a=parametric_plot3d(C,(t,I[0],I[1]),rgbcolor=
(1,1,0),thickness=1,aspect_ratio=[1,200,200])
GGH=[]
for k in range(N/2):
    GF=g+a+line3d((C1[2*k],S1[2*k]),rgbcolor=
(0,0,1),thickness=1)+point3d(S1[2*k],rgbcolor=
(1,0,0),size=7)+point3d(C1[2*k],rgbcolor=(1,0,0),size=2)
    GGH.append(GF)
AGGH=animate(GGH)
#show(AGGH)
show(AGGH[25],figsize=[3,3,3])
```



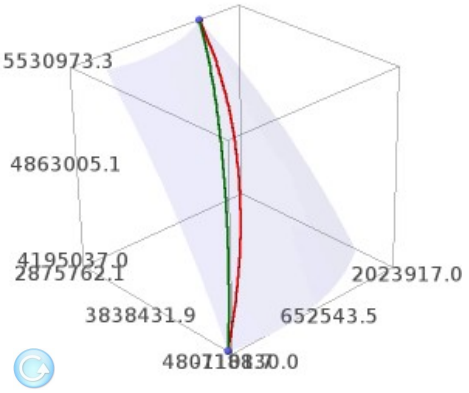
Exercicio 14:

Un avión voa a 8000 m de altura a rumbo fixo entre Porto (41.1458397,-8.6108103) e Helsinki (60.1711617,.24.9326496). Achar o rumbo, debuxar a correspondente loxodrómica e comparala coa xeodésica.

```
print 'a.'
u0=41.1458397
v0=-8.6108103
u1=60.1711617
v1=24.9326496
```

```
h=8000
A=loxeo(u0,v0,u1,v1,h)
show(A[0],figsize=[3,3,3])
```

a.




```
print'Rumbo loxodromica=',round(A[1],3),'°'
print'Atallo xeodésico=',round((A[2]-A[3])/1000,2),'km'
```

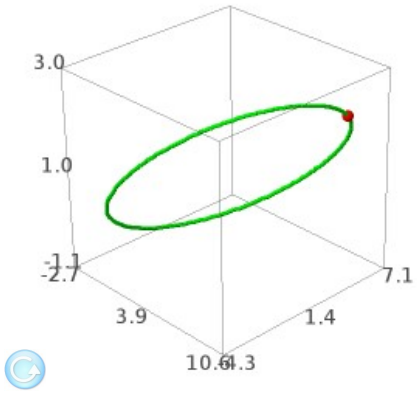
```
Rumbo loxodromica= 47.647 °
Atallo xeodésico= 27.32 km
```

Exercicio 15:

A función hamiltoniana dun planeta de masa unidade nun campo newtoniano é $H(\mathbf{x},\mathbf{v}) = \frac{(\mathbf{v}|\mathbf{v})}{2} - \frac{1}{\|\mathbf{x}\|}$. Animar o movemento do planeta para as condicións iniciais $\mathbf{x}(0) = (10,0,3)$ e $\mathbf{v}(0) = (0.1,0.2,0)$

```
var('t x0 x1 x2 x3 x4 x5')
q=vector([x0,x1,x2])
p=vector([x3,x4,x5])
H=(p*p)/2-1/norma(q)
Hp=[diff(H,x3),diff(H,x4),diff(H,x5)]
Hq=[-diff(H,x0),-diff(H,x1),-diff(H,x2)]
f(t,x0,x1,x2,x3,x4,x5)=Hp+Hq
CI=vector([10,0,3,.1,.2,0])
I=[0,118]
N=236
P=rungekutta(f,CI,I,N)
P0=[P[k][0:3] for k in [0..N]]
```

```
g=line3d(P0,rgbcolor=(0,1,0),thickness=5)
GP=[]
for k in range(N/2):
    GF=g+point3d(P0[2*k],rgbcolor=(1,0,0),size=15)
    GP.append(GF)
AGP=animate(GP)
#show(AGP)
show(AGP[25],figsize=[3,3,3])
```

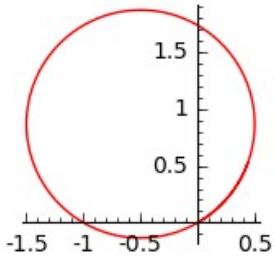


Exercicio 16:

Representar a curva plana con $\kappa(s) = 1$ que parte de $(0,0)$ e en devandito punto ten tanxente $(\cos(\alpha), \sin(\alpha))$ e normal $(-\sin(\alpha), \cos(\alpha))$.

```
k(s)=1
f(s,t1,t2,n1,n2)=[k(s)*n1,k(s)*n2,-k(s)*t1,-k(s)*t2]
al=pi/6
CIT=vector([cos(al),sin(al),-sin(al),cos(al)])
I=[0,7]
N=100*I[1]
h=I[1]/N
F=rungekutta(f,CIT,I,N)
FT=[vector([a[0],a[1]]) for a in F]
FTN=[a/norm(a) for a in FT]
PI=vector([0,0])
C=[PI]
for k in range(N-2):
    C.append(C[-1]+vector((h*(FTN[k]+FTN[k+1])/2).n()))
```

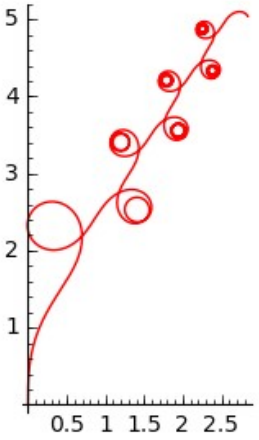
```
show(line(C[0:N],color='red',aspect_ratio=1), figsize=[2,2])
```



Exercicio 17:

Representar a curva plana con $\kappa(s) = s\cos(s)$ que parte do punto $(0,0)$ e en devandito punto ten tanxente $(\cos(\alpha), \sin(\alpha))$ e normal $(\sin(\alpha), -\cos(\alpha))$.

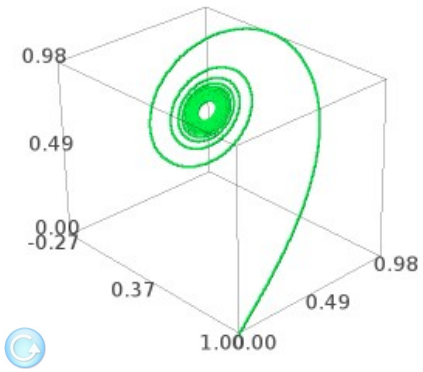
```
k(s)=s*cos(s)
f(s,t1,t2,n1,n2)=[k(s)*n1,k(s)*n2,-k(s)*t1,-k(s)*t2]
al=pi/2; CIT=vector([cos(al),sin(al), sin(al),-cos(al)])
I=[0,24]; N=100*I[1]; h=I[1]/N
F=rungekutta(f,CIT,I,N)
FT=[vector([a[0],a[1]]) for a in F]
FTN=[a/norm(a) for a in FT]
PI=vector([0,0])
C=[PI]
for k in range(N-2):
    C.append(C[-1]+vector((h*(FTN[k]+FTN[k+1])/2).n()))
show(line(C[0:N],color='red',aspect_ratio=1),figsize=[4,3])
```



Exercicio 18:

Representar a curva con curvatura $\kappa(s) = s$ e $\tau = 0$ como curva de $\mathbb{R}^3$ que parte do punto $Xo = (1, 0, 0)$ e nel ten tanxente $To = (0, \frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2})$ e normal $No = (-1, 0, 0)$.

```
K(s)=s
TOR=0
Xo=vector([1, 0, 0])
To=vector([0, 1/2*sqrt(2), 1/2*sqrt(2)])
No=vector([-1, 0, 0])
Bo=To.cross_product(No)
f(s,t1,t2,t3,n1,n2,n3,b1,b2,b3)=[K*n1,K*n2,K*n3,TOR*b1-
K*t1,TOR*b2-K*t2,TOR*b3-K*t3,-TOR*n1,-TOR*n2,-TOR*n3]
CI=vector(list(To)+list(No)+list(Bo))
I=[0,4*pi*sqrt(2)]
N=100*ceil(I[1]-I[0])
h=(I[1]-I[0])/N
F=rungekutta(f,CI,I,N)
Tan=[a[0:3].n() for a in F]
C=[Xo]
for k in range(N-2):
    C.append(C[-1]+(h*(Tan[k]+Tan[k+1])/2).n())
show(line3d(C[0:N],rgbcolor=
(0,1,.3),thickness=3,aspect_ratio=1),figsize=[3,3,3])
```



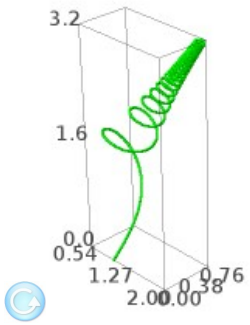
Como pode verse obtemos unha espiral de Cornu no plano determinado por To

e No que pasa por Xo.

Exercicio 19:

Representar a curva de $\mathbb{R}^3$ con curvatura $\kappa(s) = s$ e $\tau = 1$ que parte do punto Xo e nel ten tanxente, normal e binormal To, No e Bo.

```
K(s)=s; TOR=1
f(s,t1,t2,t3,n1,n2,n3,b1,b2,b3)=[K*n1,K*n2,K*n3,TOR*b1-
K*t1,TOR*b2-K*t2,TOR*b3-K*t3,-TOR*n1,-TOR*n2,-TOR*n3]
CI=vector(list(To)+list(No)+list(Bo))
I=[0,4*pi*sqrt(2)]
N=100*ceil(I[1]-I[0])
h=(I[1]-I[0])/N
F=rungekutta(f,CI,I,N)
Tan=[a[0:3].n() for a in F]
C=[Xo]
for k in range(N-2):
    C.append(C[-1]+(h*(Tan[k]+Tan[k+1])/2).n())
show(line3d(C[0:N],rgbcolor=
(0,1,0),thickness=3,aspect_ratio=1),figsize=[2.5,2.5,2.5])
```

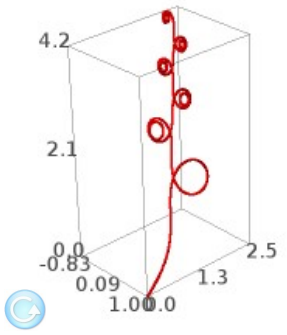


Exercicio 20:

Representar a curva de $\mathbb{R}^3$ con curvatura $\kappa(s) = s\cos(s)$ e $\tau = 1$ que parte do punto Xo e nel ten tanxente, normal e binormal To, No e Bo.

```
K(s)=s*cos(s); TOR=1/(1+s)
f(s,t1,t2,t3,n1,n2,n3,b1,b2,b3)=[K*n1,K*n2,K*n3,TOR*b1-
K*t1,TOR*b2-K*t2,TOR*b3-K*t3,-TOR*n1,-TOR*n2,-TOR*n3]
```

```
CI=vector(list(To)+list(No)+list(Bo))
I=[0,20]; N=100*ceil(I[1]-I[0]); h=(I[1]-I[0])/N
F=rungekutta(f,CI,I,N)
Tan=[a[0:3].n() for a in F]
C=[Xo]
for k in range(N-2):
    C.append(C[-1]+(h*(Tan[k]+Tan[k+1])/2).n())
show(line3d(C[0:N],rgbcolor=
(1,0,0),thickness=3,aspect_ratio=1),figsize=[2.5,2.5,2.5])
```



Bibliografía

1. de Arriba, F., Corbacho, E., Somoza, M.C., and Vidal, R. Implementación e desenvolvemento de aulas de matemáticas avanzadas en Sage. Universidade de Vigo. Vigo 2018.

2. de Arriba, F., Corbacho, E., and Vidal, R. Designing Hamiltonian Cycles. ACA2013, Málaga, 2013.

3. Aurenhammer, F. Voronoi Diagrams - A Survey of a Fundamental Geometric Data Structure. ACM Computing Surveys, Vol. 23, No. 3 pp. 345-405, 1991.

4. Bollobás, B. Modern graph theory. New York, Springer-Verlag, 1998.

5. Borûvka, O. On a minimal problem.Práce Moravské Pridovedecké Spolecnosti, vol. 3, 1926.

6. Casamayou, N., Cohen, G., Connan, T., Dumont, L., Fousse, F., Maltey, M., Meulien, M., Mezzarobba, C., Pernet, N. M., Thiéry, P., and Zimmermann. Calcul mathématique avec Sage. http://sagebook.gforge.inria.fr

7. Corbacho Rosas, E., Rodríguez de Miguel, V. Cálculo avanzado, 2017. https://corbacho.webs.uvigo.es

8. Corbacho Rosas, E., Vidal Vázquez, V. Matemáticas de la Especialidad y Métodos Matemáticos, 2017. https://corbacho.webs.uvigo.es

9. Crouzeix, M., Mignot, A.L. Analyse numérique des équations différentielles. Paris, Masson, 1984.

10. Dillencourt, M.B. Toughness and Delaunay triangulations. In Proceedings of the 3rd Annual ACM Symposium on Computational geometry. pp. 186 - 194, 1987.

11. Gauss, C.F. Briefwechsel Gauss-Shuhmacher. In Werke Bd. X, 1, Göttingen, págs. 459--468, 1917.

12. Harary, F. Graph theory. Encyclopedia of Mathematics and its applications, vol. 21, Gian-Carlo Rota Editor. Department of Mathematics. Massachusetts Institute of Technology. Cambridge. Published by the Press Syndicate of the University of Cambridge 1984.

13. Har-Peled, S. Geometric Approximation Algorithms. American Mathematical Soc., 2011.

14. Kantabtra, V. Traveling salesman cycles are not always subgraphs of Voronoi duals. Inf Process. Lett. 16, pp. 11 - 12, 1983.

15. Kruskal, J. B. On the shortest spaning subtree and the traveling salesman problem. Proceedings of the American Mathematical Society 7, pp. 48-50, 1956.

16. Rodríguez Tello, E. A. Triangulaciones de Delaunay. CINVESTAV-Tamaulipas, 2013. https://www.tamps.cinvestav.mx/~ertello/gc/sesion18.pdf

17. Pollak, H. O. Some Remarks on the Steiner Problem. Journal of Combinatorial Theory, Series A 24, 278 - 295, 1978.

18. SageMath http://www.sagemath.org/index.html

19. Simson, R. Los seis primeros libros, y el undécimo y duocécimo de los elementos de Euclides. Editorial Maxtor. Madrid 2014.

20. Somoza López, M.C. Estructuras métricas en grafos, Tesis. Departamento de Matemática Aplicada I, Universidade de Vigo, 2015. https://corbacho.webs.uvigo.es

21. Stein, W. SAGE for Power Users, 2012. http://www.wstein.org/books/sagebook/sagebook.pdf

22. Van Laarhoven, J. W. Exact ahd heuristic algorithms for the Euclidean Steiner tree problem. University of Iowa, 2010. https://ir.uiowa.edu/etd/755/

23. Vo$\mathcal{B}$, S. Steiner-Probleme in Graphen.Anton Hain, Frankfurt a.M., 1990.

24. Zsoldos, I. Effect of topological defects on graphene geometry and stability. Nanotechnology, Science and Applications, 3, Dovepress. Dove Medical Press. págs. 101--106, 2010.

Servizo de Publicacións

Universidade Vigo

9 788481 588453