Programa queorbita_G_csv: Descobrindo a órbita entre galáxias em interação

Irapuan Rodrigues*

Abril de 2021

Abstract

No estudo da dinâmica de pares de galáxias em interação, um problema a ser resolvido é o da determinação da órbita entre as galáxias. Os dados observacionais são insuficientes, levando a soluções não únicas, degeneradas. As observações nos fornacem 3, das 6 coordenadas do espaço de fases: a posição projetada no plano do céu (X e Y), e as velocidades na linha de visada das galáxias (Vz). As outras 3 coordenaddas (Z, Vx e Vy) devem, então ser descobertas. Para isso foi desenvolvido programa Q̈ueorbita; que tem como entrada os dados observacionais de posição no plano do céu e velocidade radial relativas das galáxias do par, e tem como saída os parâmetros de um número grande de órbitras que poderiam levao o sistema à sua condição atual. Este artigo apresenta um breve tutorial sobre o uso do programa `Queorbita` e alguns programas auxiliares.

Keywords: galáxias em interação, dinâmica de galáxias, órbitas galácticas

1 Introdução

O estado dinâmico de uma colisão de galáxias é descrito por uma função de distribuição no espaço de fases $f(r, v)$ (hexadimensional), que dá a densidade de massa na posição r e velocidade v. Observações, por sua vez, podem fornecer informações em apenas 3 coordenadas: posição em cada ponto (X, Y) no plano do céu, e uma componente do vetor velocidade, V_z, na direção da linha de visada. $f(r, v)$ depende de 6 variáveis, mas os dados observacionais só fornecem 3: $Fc(X, Y, V)$, o que significa que as observações não oferecem informação suficiente. De forma simples, há infinitas $f(r, v)$ hexadimensionais consistentes com a $Fc(X, Y, V)$ tridimensional fornecida pelos dados observacionais.

Pensando nisso, foi desenvolvido um algoritmo que varre um espaço predefinido de excentricidades e orbitais e distâncias de pericentro q, buscando encontrar órbitas que satisfaçam os vínculos observacionais, fornecendo para

*Universidade do Vale do Paraíba (Univap), Brasil.

cada órbita encontrada as coordenadas relativas Z, V_x e V_y. Além das coordenasas de posição e velocidade, as massas das galáxias também são dados de entrada importantes. Em geral, essa técnica fornece um número muito grande de possíveis órbitas, que devem posteriormente ser analisadas e testadas. A morfologia do par, suas caudas de maré e pontes oferecem indícios que nos permitem eliminar ou escolher certas famílias de órbitas a serem testadas por meio de simulações de colisões de galáxias.

Um problema adicional, em se tratando de galáxias discoidais, é a orientação dos discos (ângulos de inclinação (i) e de posição (P_A) e direção do vetor momento angular, que podem ser estimados observacionalmente. Estas informações também serão importantes no refinamento das soluções encontradas.

2 O algoritmo Queorbita

Calcula todas as órbiras com valores dados de excentricidade e, distância de pericentro q, massas das galáxias M e m que satisfaçam as condições observacionais fornecidas (posição e velocidade na linha de visada de dois objetos).

Sistema de coordenadas e unidades:

- O plano do céu é o plano XY e o eixo Z aponta para o observador.
- A galáxia de massa M está no foco da órbita e é o centro do sistema de coordenadas $(0, 0, 0)$;
- Os dados de entrada devem ser informados no sistema de coordenadas usado nas simulações Gadget, em que $G = 43007.1$;
- Unid de distância = 1 kpc
- Unid de velocidade = 1 km/s
- Unid de massa = 1e10 $M_\odot$
- Unid de tempo = 0.9779 Gyr
- ATENÇÃO: A velocidade na linha de visada (V_sys) deve ser informada no sistema de unidades acima, em que o sinal é invertido com respeito à à convenção observacional. Na convenção observacional o eixo z aponta do observador para o objeto (velocidade positiva para o objeto se afastando do observador). Aqui deve ser o contrário.

O programa foi escrito em linguagem C. Sua primeira versão data de 1998. Diversas modificações foram feitas ao longo dos anos. Ver comentários no código fonte.

2.1 O algoritmo

Depois de ler os dados de entrada, o programa realiza vários loops aninhados. O mais externo varre os valores de excentricidade e. Em seguida, varre os valores de distância de pericentro q. Depois vem o loop em gz (a coordenada espacial Z), cujos limites são predefinidos de -10 a 10 vezes a distância entre as galáxias, projetada no plano do céu, em 1000 passos.

Para cada órbita definida dentro desses loops, é calculada a poição espacial do vetor de pericentro. Um novo sistema de coordenadas é definido, como sendo o sistema próprio definido pelo plano orbital, com a direção Y coincidindo com a direção do pericentro orbital e centro do sistema coindidindo com a galáxia principal (a de massa M).

O importante até aqui são os ângulos:

- ϕ, entre o vetor posição atual $rnow$ e o vetor velocidade v;
- ω, entre q e $rnow$

Agora devo girar o vetor velocidade v em torno da direção de $rnow$ e medir sua projecao na linha de visada para ver se em alguma posição ele tem componente Z que satisfaça o vínculo observacional imposto por $V_{sys} \pm \sigma_{V_{sys}}$. Isso só será satisfeito, obviamente, se v for maior que V_{sys}.

O jeito de fazer isso é assim:

- Faço mudanca de coordenadas para um sistema em que o vetor $rnow$ esteja alinhado com um dos eixos. Para isso escrevo $rnow$ em coordenadas esfericas $(sr, stheta, sphi)$, giro um ângulo $-sphi$ em torno de Z, e giro um ângulo $stheta$ em torno de Y;
- Nesse sistema crio um vetor SVV com as coordenadas dos pontos que representam v, que estão sobre um circulo que é o corte de um cone formado pelo giro de v em redor de $rnow$;
- Transformação inversa de coordenadas sobre os pontos de v, para obter os possíveis vetores v no sistema do céu. A coordenada que interessa aqui é a alinhada com a linha de visada.
- As órbitas que satisfizerem as condições impostas por $V_{sys} \pm \sigma_{V_{sys}}$ e posição projetada, serão guardadas na saída.

2.2 Uso do programa

Instrução de compilação:

```
gcc queorbita_G_csv.c -o queorbita_G_csv -lm
```

Depois de compilado, você pode rodar o programa e introduzir os dados de entrada à medida que forem solicitados, que serão:

- Os valores de excentricidade inicial ($eini$), final ($efin$) e o passo ($estep$);
- As distâncias de pericentro inicial ($qini$), final ($qfin$) e o passo ($qstep$);
- A posição relativa da galáxia secundária (gx, gy);
- As massas M e m;
- A velocidade radial relativa V_{sys} e sua incerteza $\sigma_{V_{sys}}$;
- As 3 coordenadas do vetor de spin do disco principal () para ver se o movimento é prograd o ou retrógrado.

Mas é preferível escrever um arquivo (chamaremos aqui de `queorbita.in`) com as condições iniciais, como no exemplo numérico a seguir:

0.8 1.2 0.1
3.5 12.5 0.0.2
86.5874 95.2401
-8.8 -4.2
-335 10
0.616342 0.539567 0.752415

Para rodar o programa, usando o arquivo de entrada `queorbita.in`, e redireiconar a saída de tela (STDOUT) para o arquivo `queorbita.out`:

```
./queorbita_G_csv < queorbita.in > queorbita.out
```

A saída do programa é para o STDOUT, devendo ser redirecionada para um arquivo (`queorbita.out` no exemplo acima). Os dados de entrada, bem como uma série de outras informações, são apresentados em um cabeçalho. Em seguida, para cada órbita são nostradas as quantidades:

- e = excentricidade;
- q = distancia de pericentro;
- $rnow$ = distancia atual;
- vq = velocidade no pericentro;
- gx = coordenada X (sistema CÉU, dado de entrada);
- gy = coordenada Y (sistema CÉU, dados de entrada);
- gz = coordenada Z (sistema CÉU);
- VX_{ceu} = Velocidade coordenada X (sistema CÉU);

- VY_{ceu} = Velocidade coordenada Y (sistema CÉU);
- VZ_{ceu} = Velocidade coordenada Z (sistema CÉU);
- x = coordenada X no sistema da órbita (orb no plano XY, pericentro em Y, Z=0, antihorário);
- y = coordenada Y no sistema da órbita;
- vx = componente velocidade X no sistema da órbita;
- vy = componente velocidade Y no sistema da órbita;
- spx, spy, spz = componentes vetor spin da órbita no sistema CÉU;
- qx, qy, qz = componentes vetor direção pericentro no sistema CÉU;
- $vsys$ = componente velocidade Z no sistema CÉU (igual a gz);
- Dir = Direção da órbita: prograd ou retrógrado em rel ao disco M;
- $spin - orb$ = Ângulo entre vetor de spin do disco M e spin da órbita;
- PERIC = Situação PRÉ ou PÓS PERICENTRO;
- $pos - peri$ = Ângulo entre vetor posição (g) e o vetor de pericentro (q);
- APOCENTRO = Distância de apocentro;

A saída do `queorbita_G_csv` é em formato CSV, para facilitar a leitura pelos programas em python, escritos para visualização e análise das órbitas, que serão descritas a seguir.

2.3 Calculando as órbitas

Para calcular as órbitas obtidas pelo `queorbita_G_csv`, foi escrito o programa `pot_hbd_din_fric-2.c` (em linguagem C), que calcula cada uma das órbitas. Nesse programa as órbitas são calculadas de 3 formas diferentes:

1. Como órbitas keplerianas que consideram as galáxias como massas puntuais;
2. Levando em conta os potenciais das galáxias, que são estendidos. Isso afeta a órbita, uma vez que quando elas estão a pequenas distâncias, a massa que entra no cálculo da aceleração, (pelo Teorema de Newton), é a massa contida até a distância que separa as galáxias. Assim, quanto mais próximas as galáxias, ao considerarmos massas puntuais (caso kepleriano) estaremos superestimando a aceleração.
3. Como no método 2, mas incluindo uma estimativa da fricção dinâmica, calculada pela fórmula de Chandrasekar.

Calcular as órbitas pelos métodos 2 e 3 é importante, pois nas simulações autoconsistentes as órbitas se desviam consideravelmente do caso kepleriano. Esse programa fornace as condições iniciais mais consistentes com o que se vai ver nas simulações. Quando usamos a órbita kepleriana para gerar as condições iniciais de uma simulação, a reprodução de um sistema observado fica comprometida. O método 3 pode ser "calibrado" usando simulações.

Instrução de compilação:

```
gcc pot_hbd_din_fric-2.c -o pot_hbd_din_fric-2 -lm
```

Os dados de entrada devem ser informados no seguinte sistema de unidades:

- Distâncias em kpc;
- Velocidades em km/s;
- Massas in Massas Solares;
- Tempo em Myr (t=0 será o tempo correspondente à posição informada nos dados de entrada).

Os parâmetros de entrada do `pot_hbd_din_fric-2.c` são:

- Potential 1 - MAIN GALAXY (Hernquist, 1990):
  - MAIN Galaxy Total mass (M200, MakeNewDisk output) [M_{sun}]
  - MAIN Galaxy R200 (MakeNewDisk output) [kpc]
  - HALO scalelength (aHalo, MakeNewDisk output - RH) [kpc]
  - DISK mass fraction (dmfrac)
  - DISK scalelength (aDisk, end of MakeNewDisk output) [kpc]
  - DISK vertical scalelength (DiskHeight, from MakeNewDisk parameters file) [kpc]
  - BULGE mass fraction (bmfrac, from MakeNewDisk parameters file - MD)
  - BULGE scalelength fraction (aBulge, from MakeNewDisk parameters file - BulgeSize)
  - BLACK HOLE mass fraction (bhfrac, from MakeNewDisk parameters file - MBH)
- Potential 2 - SECONDARY GALAXY (Hernquist, 1990):
  - Pot 2 Position (x,y,z) [kpc]
  - Pot 2 Velocity (vx,vy,vz) [km/s]
  - Pot 2 mass [M_{sun}]
  - Pot 2 radial scalelength [kpc]

- Tempos (use valores positivos para avançar no tempo, negativos para retroceder):
    - Tempo final (tf [Myr])
    - Timestep (dt [Myr])

Para rodar programa `pot_hbd_din_fric-2.c` também é conveniente criar um arquivo com os dados de entrada na ordem apresentada acima. Do contrário, ao executar ele vai pedir que cada dado seja informado na linha de comando. Se você nomear o arquivo de parâmetros de entrada de `pot_hbd_din_fric-2.in`, a linha de comando seria:

Para rodar programa `pot_hbd_din_fric-2.c` também é escrever um arquivo (chamaremos aqui de `BACKWARDS_pot_hbd_din_fric.in`) com as condições iniciais, como no exemplo numérico a seguir, para integração da órbita a tempos negativos:

$86.5874E10$
155.0
12.9167
0.001
1.61809
0.2
0.17
1.5
0.0001
05.00.0
$-1614.50.00.0$
$95.2401E10$
160.0
-300.0
-0.1

Para rodar o programa, usando o arquivo de entrada `BACKWARDS_pot_hbd_din_fric.in`, e redireiconar a saída de tela (STDOUT) para o arquivo `BACKWARDS_pot_hbd_din_fric.out`:

```
./pot_hbd_din_fric-2 < BACKWARDS_pot_hbd_din_fric.in > BACKWARDS_pot_hbd_din_fric.out
```

A saída para STDOUT será redirecionada para `BACKWARDS_pot_hbd_din_fric.out`, e a órbita vai ficar em arquivo chamado `orbits_pot_hbd_din_fric.dat`

3 Rotinas em python

Foram escritas duas rotinas em Python3 para auxiliar na análise dos dados gerado pelo `queorbita_G_csv`. Essas rotinas são:

1. `calcula_orbitas_queorbita.py`
2. `plota_orbitas_queorbita.py`

3.1 calcula_orbitas_queorbita.py

Para rodar este programa, você deve previamente ter rodado o `queorbita_G_csv` para gerar o arquivo com a listagem das órbitas. O programa lê esse arquivo (cuidado que tem que acertar o nome do arquivo gerado pelo `queorbita_G_csv`. Procure a linha:

```
queorb = pd.read_csv('queorbita_anima.out' ,  header=50, sep=",")
```

e acerte o nome do arquivo, que neste exemplo é `queorbita_anima.out`

Depois de ler o arquivo acima, o programa vai criar o diretório `orbits_temp`, onde vão aparecer os arquivos de todas as órbitas. Para cada órbita, o programa vai fazer duas chamadas ao programa `pot_hbd_din_fric-2`, uma pra calcular a órbita pra frente e outra para trás no tempo. Vai conectar as duas metades da órbita e colocar num arquivo único. Nesse processo, vários arquivos auxiliares serão criados e depois apagados ao final, deixando no diretório `orbits_temp` apênas os arquivos completos de órbitas.

Dependendo do número de órbitas a serem calculadas, o programa pode demorar bastante.

3.2 plota_orbitas_queorbita.py

Esta rotina lê as órbitas criadas pela `calcula_orbitas_queorbita.py` e plota as órbitas desejadas. Ela também precisa ler o arquivo de saída do `queorbita_G_csv` (no exemplo acima `queorbita_anima.out`). Depois testa se o diretório `orbits_temp`, dando uma mensagem de erro, caso não exista. Para plotar, ela usa a biblioteca `PLOTLY` do python, que você deverá instalar previamente.

Por volta da linha 100 dessa rotina, é definida a variável `orb_N`, que lista as órbitas a serem plotadas. Há lá alguns exemplos de como definir essa variável. Os números das órbitas podem ser vistos na no Dataframe `queorb`, que é criado com Pandas logo no início, na linha:

```
queorb = pd.read_csv('queorbita_anima.out' ,  header=50, sep=",")
```

Exemplos de plots da uma única órbita são mostrados na Figura 1. Quando plotar multiplas órbitas, é aconselhável plotar somente um dos 3 tipos, escolhendo qual órbita plotar por volta da linha 112, para dimimuir a confusão.
.

4 Lista dos programas

- `queorbita_G_csv.c`
- `pot_hbd_din_fric-2.c`
- `calcula_orbitas_queorbita.py`
- `plota_orbitas_queorbita.py`

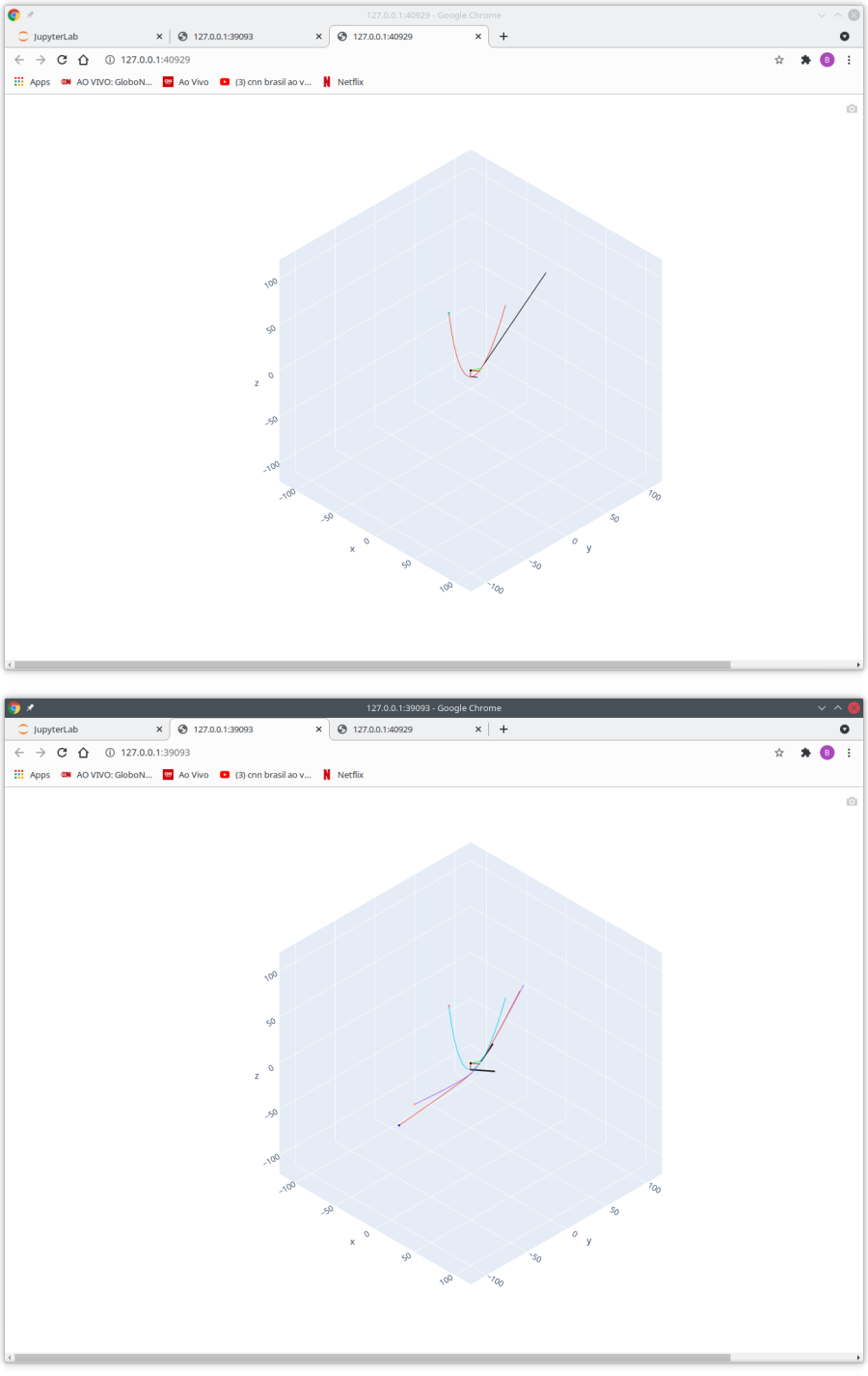

Figure 1: (a) Plot de uma órbita kepleriana; (b) Plot de uma órbita. São mostradas as 3 órbitas calculadas para uma mesma condição inicial: A kepleriana (azul), considerando os potenciais estendidos das galáxias (roxo) e com fricção dinâmica (vermelho)