AJUSTE COMPUTACIONAL DE PARÂMETROS DE EXOPLANETAS VIA SIMULAÇÃO DE MONTE CARLO COM CADEIAS DE MARKOV

Julio Cesar Lopes Bertolacini (IC) e Prof. Dr. Luciano Silva (Orientador)

Apoio: PIBIC CNPq

RESUMO

O método de trânsitos planetários é um dos métodos possíveis para detecção de exoplanetas. A análise computacional dos dados coletados por instrumentos astronômicos que buscam exoplanetas por análise de trânsitos como, por exemplo, o instrumento Kepler (NASA), pode ser feita por algoritmos determinísticos. Um dos principais motivos para o uso destes algoritmos é sua simplicidade de implementação. Porém, a sua aplicação em grandes volumes de dados tem demonstrado grande perda de eficiência. Uma maneira alternativa para realizar análises computacionais de trânsitos planetários consiste no uso de algoritmos probabilísticos, que possuem taxas de convergência mais eficientes que algoritmos determinísticos para grandes volumes de dados. Neste contexto, este trabalho apresenta uma integração de um algoritmo probabilístico (MCMC), com estratégia de amostragem via ensemble com invariância afim, para detecção e determinação de parâmetros de trânsitos em dados sintéticos. Foi realizada uma implementação em Python integrando os módulos emcee (MCMC) e batman (gerador sintético de trânsitos planetários), que permitiu disponibilizar um notebook para análise de trânsitos planetários.

Palavras-chave: Trânsitos Planetários, MCMC, Astroinformática.

ABSTRACT

The planetary transit method is frequently used in exoplanets detection. Given the simplicity of implementation, deterministic algorithms are frequently applied for transit detection in data collected by astronomical instruments that look for exoplanets using transit analysis such as Kepler (NASA). Despite the advantage of easy implementation, the application of deterministic algorithms in large amount of data has presented substantial loss of efficiency. Probabilistic algorithms are an alternative way to perform computational analysis of planetary transits, this brand of algorithms have better convergence rates than deterministic algorithms for large data volume. In this context, this work presents an integration the probabilistic algorithm MCMC, using ensemble sampler with affine invariance, for transit detection in synthetic transit data and parameters determination. A Python implementation was made by integrating modules emcee (MCMC) and batman (synthetic planetary transits generator), which allowed to provide a notebook for planetary transits analysis.

Keywords: Planetary Systems, MCMC, Astroinformatics.

1. INTRODUÇÃO

Um exoplaneta é um planeta que orbita qualquer estrela que não seja o Sol, ou seja, um planeta extra-solar. A busca por exoplanetas tem evoluído nos últimos anos, impulsionada por missões como a do satélite Kepler e a busca por planetas que possam abrigar vida. O satélite Kepler utiliza o método dos trânsitos planetários, uma das muitas técnicas de detecção de exoplanetas. Através de telemetria, o Kepler observa variações de brilhos de cerca de 145.000 estrelas na Constelação de Cisne. Os dados gerados por estas observações são chamados curvas de luz e podem utilizadas para obter diversos parâmetros a respeito da estrela e de possíveis planetas que a orbitem. O tamanho da curva de luz, dependendo da cadência de observação, pode gerar muitos pontos (BASILE *et. al*, 2015), o que justifica a utilização de métodos computacionais para colaborar no processamento.

Para identificar a existência de um trânsito planetário em uma curva de luz, os métodos computacionais podem buscar periodicidades ou decaimentos nas curvas de luz. Existem três métodos computacionais determinísticos para análise de curvas de luz: *Box Least Squares* (BLS), Lomb-Scargle e Plavchan. O tempo de processamento para grandes curvas de luz com algoritmos determinísticos pode ser bastante elevado, da ordem de dias (BASILE *et. al*, 2015). Assim, o desenvolvimento de métodos alternativos de análise de curvas de luz que diminuam o tempo de processamento é de grande interesse tanto do ponto de vista computacional quanto para aplicações em Astronomia e Astrofísica.

Dentro deste contexto, este trabalho apresenta e implementa uma integração de uma classe de métodos probabilísticos para ajuste de parâmetros chamados MCMC (Monte Carlo Markov Chain) e geradores de curvas de luz sintéticas, para estimação do parâmetro de período orbital de trânsitos planetários. O uso de geradores de curvas de luz sintéticas tem a vantagem de se poder controlar o número de pontos, além de se poder introduzir efeitos típicos de observação como ruídos.

A integração dependeu do desenvolvimento de geradores de probabilidade *a priori* e *a posteriori*, além de uma função de verossimilhança, necessários ao método MCMC. A implementação do MCMC foi realizada com o módulo Python emcee (FOREMANMACKEY, 2013). Em seguida, foram geradas curvas sintéticas com e sem ruído utilizando o módulo Python batman (KREIDBERG, 2015), de tal modo que a detecção dos trânsitos pudesse ser feita desde o caso mais simples (sem ruído) até um caso mais complexo (com ruído).

Como produto desta integração, o trabalho disponibiliza um *notebook* em Python que permite a sua utilização para detecção de trânsitos tanto em dados de curvas de luz sintéticas quanto naquelas advindas de observações. Além disto, os resultados deste trabalho também foram publicados em um congresso nacional de grande impacto (ROHLING e SILVA, 2016).

2. REFERENCIAL TEÓRICO

A Fotometria (SEAGER, 2010) é o ramo da óptica que se preocupa em medir a luz, em termos de como seu brilho é percebido por um dispositivo de captura como, por exemplo, um satélite de observação astronômica. A Fotometria também é utilizada na Astronomia, na observação de estrelas, pela percepção da diminuição da luz por elas emitida. Através de processos de otimização, é possível descobrir novos planetas e saber informações como rotação, translação, distância da estrela e satélites.

Técnicas de Fotometria têm sido aplicadas com sucesso na detecção e caracterização de exoplanetas (PERRYMAN, 2014), que são planetas orbitando outras estrelas que não sejam o Sol. O Método dos Trânsitos Planetários é uma das técnicas fotométricas de identificação de exoplanetas mais utilizada nos observatórios mundiais.

O trânsito planetário é um fenômeno que consiste na passagem de um planeta, durante sua órbita, exatamente entre a estrela e o observador (PERRYMAN, 2014). Quando isso ocorre, o brilho aparente da estrela diminui um pouco, pois uma pequena fração de sua superfície permanece temporariamente oculta, como ilustrado na Figura 1, alterando a sua curva de luz percebida.

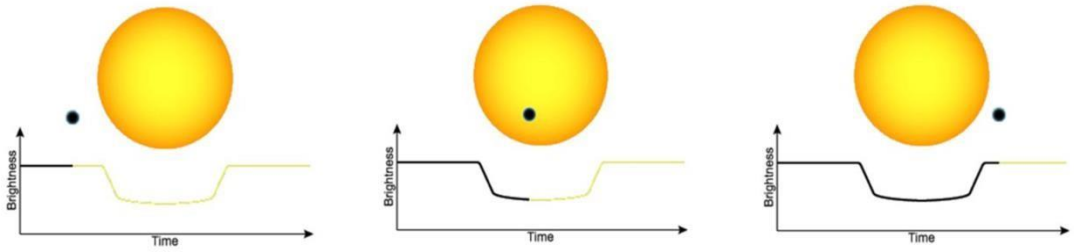

Figura 1: Trânsito planetário e sua respectiva curva de luz.

Fonte: (Perryman, 2014).

Este fenômeno é considerado como um método próprio de investigação de atividade estelar, uma vez que pode ser combinado às análises de curvas de luz. A análise das curvas de luz de uma estrela permite: detectar a ocorrência de exoplanetas orbitando a estrela e a partir da detecção, estimar parâmetros como raio do planeta, massa, fator de impacto, dentre outros.

Um dos grandes observadores de trânsitos planetários é a sonda Kepler (KEPLER, 2015), que consiste em um observatório espacial projetado pela NASA que deverá procurar por planetas extra-solares. Para esta finalidade, a sonda deverá observar as 100.000 estrelas

mais brilhantes do céu por um período de quatro anos, a fim de detectar alguma ocultação periódica de uma estrela por um de seus planetas.

O objetivo da missão é explorar a estrutura e a diversidade dos sistemas planetários. Para atingir este objetivo, um grande número de estrelas deverá ser observado. Esta missão vai procurar:

- Determinar quantos planetas do tipo da Terra e de grandes planetas existem nas proximidades da região habitável (*) de um amplo espectro variável de estrelas.
- Determinar o tamanho das órbitas destes planetas.
- Estimar quantos planetas existem em sistemas de múltiplas estrelas.
- Determinar o tamanho e o tipo da órbita, brilho, tamanho, massa e densidade dos planetas gigantes de período curto.
- Identificar membros adicionais a cada descoberta de um sistema planetário, fazendo o uso de outras técnicas.
- Determinar as propriedades das estrelas que hospedam sistemas planetários.

A análise de uma curva de luz gerada pelo observatório Kepler passa por alguns estágios. O primeiro estágio consiste em eliminar os possíveis (e usuais) ruídos obtidos das medições (PERRYMAN, 2014). A Figura 2 ilustra um exemplo típico de uma fotometria com ruído:

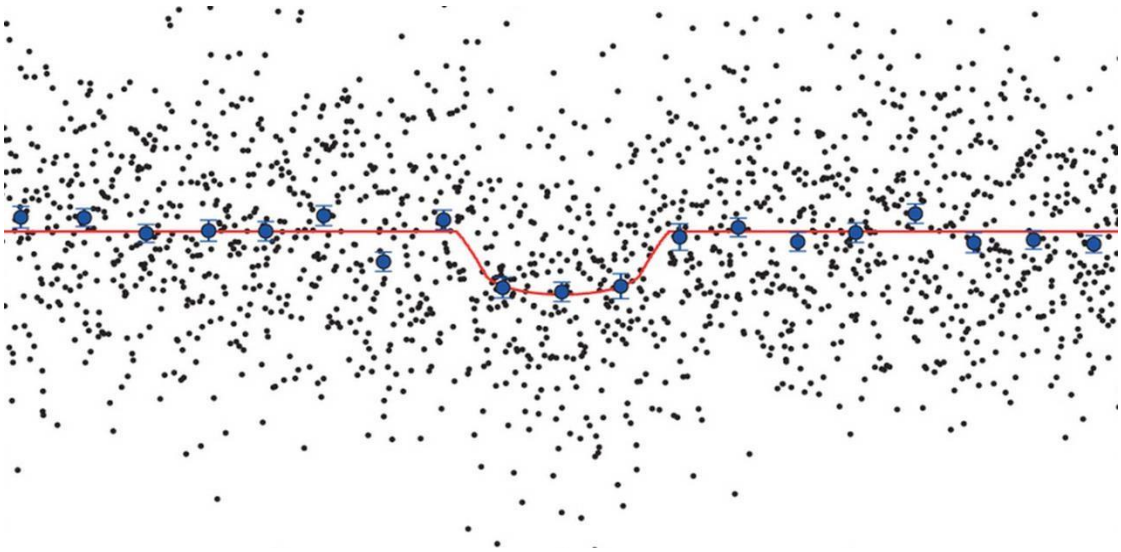

Figura 2: Fotometria da estrela Kepler 37, com alto nível de ruído.

Fonte: (KEPLER, 2015)

Nesta figura, a linha vermelha representa a curva real de luz da estrela Kepler 37. Porém, devido aos potenciais ruídos que podem ocorrer nas medições e recebimento do sinal

na Terra, a curva de luz torna-se um emaranhado de pontos. Assim, técnicas de processamento digital de sinais podem ser necessárias para atenuar estes ruídos.

Uma vez que os ruídos estejam atenuados, o processo de estimação da curva de luz experimental fica mais preciso, conforme mostrado na Figura 3 (SIRCA, 2012):

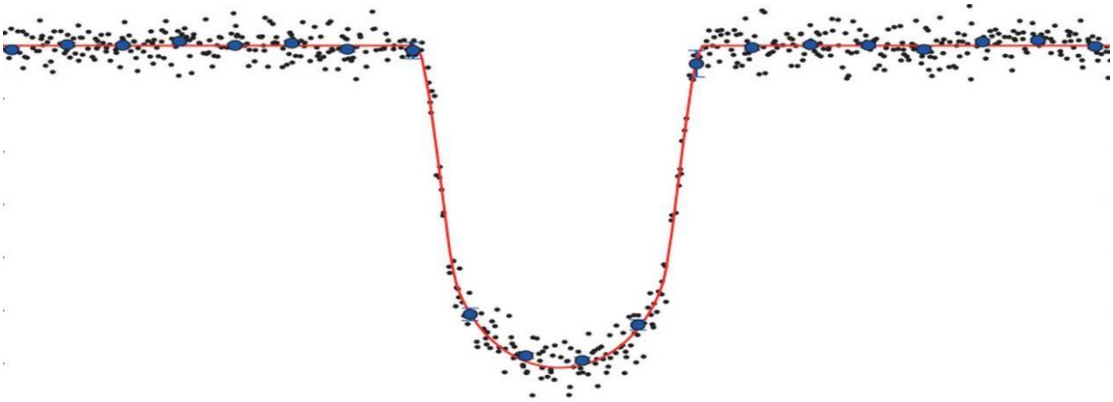

Figura 3: Fotometria com atenuação de ruído da estrela Kepler 37, com curva de luz (em vermelho) indicada.

Fonte: (KEPLER, 2015)

A obtenção das curvas de luz a partir dos dados de Fotometria é conhecida como ajuste da curva de luz e configura-se como um problema de otimização computacional. Os principais métodos numéricos para se resolver este problema de otimização estão relacionados abaixo (CHONG, 2013; SIRCA, 2012):

- Método de Gauss-Newton
- Método de Levenberg
- Método de Marquardt

Todos estes métodos são determinísticos, isto é, não utilizam nenhum tipo de passo aleatório em suas computações. Mesmo que tais métodos sejam amplamente usados, suas taxas de convergência são bastante lentas quando aplicados a grandes curvas de luz. Uma das alternativas para isto é o uso de técnicas de simulação probabilística (PFFEFER, 2016).

O Método de Monte Carlo com Cadeias de Markov (MCMC) já tem sido aplicado em outros contextos de otimização, produzindo resultados muito promissores (GAMERMAN e LOPES, 2015). A essência do método MCMC é realizar passos aleatórios de tamanho controlado para se atingir uma aproximação de um valor ótimo em processos de otimização.

A cada iteração do método MCMC, estima-se a probabilidade do passo (a posteri) melhor o valor atual da otimização a partir de probabilidades a priori e de uma função de verossimilhança (likelihood), conforme mostrado na Figura 4:

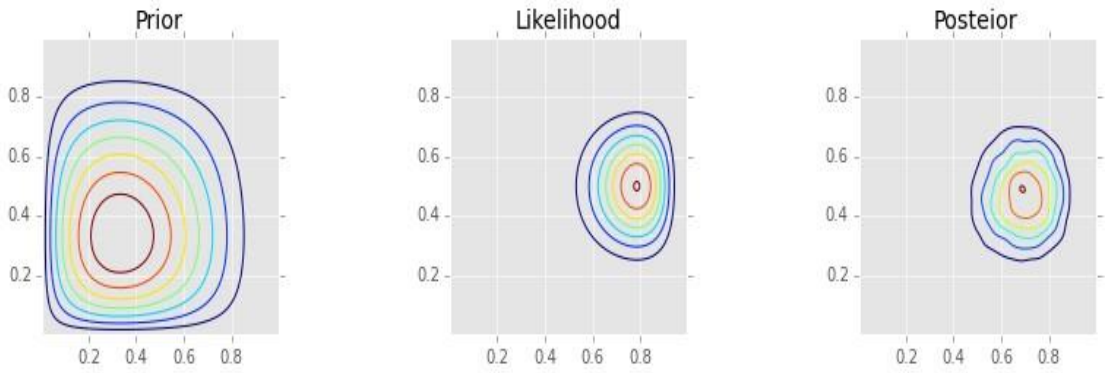

Figura 4: Estimação de uma probabilidade a posteriori a partir de uma probabilidade a priori e uma função de verossimilhança.

Fonte: (GAMERMAN e LOPES, 2015)

Se as distribuições de probabilidade a priori e da função de verossimilhança forem bem escolhidas, o método MCMC apresenta taxas de convergência mais rápidas para o ponto de ótimo quando comparadas àquelas dos algoritmos determinísticos.

Em particular, para grandes curvas de luz na prospecção de exoplanetas, taxas de convergências mais eficientes podem representar ganhos de desempenho no processamento de grandes volumes de dados.

3. METODOLOGIA

Dentro da necessidade de alto desempenho dos métodos computacionais para processamento de grandes curvas de luz apresentada no final da seção anterior, o objetivo principal deste projeto foi trabalhar em um ambiente computacional para ajustes de parâmetros de exoplanetas via Método de Monte Carlo com Cadeias de Markov. As implementações foram realizadas em Python, linguagem bastante comum para implementação sistemas astronômicos e astrofísicos, que foram testadas com dados observacionais do Observatório Kepler (NASA).

No começo do projeto foi perseguido um estudo geral de temas astrofísicos focado em exoplanetas, curvas de luz e algoritmos de detecção de trânsitos planetários, através de pesquisas e estudo da qualificação de mestrado de um dos membros do Centro de Rádio Astronomia e Astrofísica Mackenzie (CRAAM), (BASILE, 2016).

O trabalho seguiu com a montagem de ambientes para execução de aplicações de análise de curvas de luz: Astroconda, IRAF, PyRAF. Envolvendo configurações de máquinas virtuais para execução de versões do sistema operacional Linux.

Posteriormente foi dado ênfase a criação de Python Notebooks para a simulação de curvas de luz sintéticas utilizando o módulo "Basic Transit Model Calculation in Python" (KREIDBERG, 2015), e testes de diferentes exemplos de possíveis curvas de luz.

Abaixo mostro os testes de geração de curva de luz usando o módulo citado anteriormente com diferentes parâmetros de *limb-darkening*. *Limb-darkening* refere-se à queda de intensidade na imagem de uma estrela à medida que um se move do centro da imagem para as "bordas" da imagem. Isto ocorre devido à queda de densidade e temperatura à medida que um afasta-se do centro.

O primeiro teste foi realizado simulando uma curva de luz com um parâmetro de *limbdarkening* quadrático:

Figura 5: teste utilizando Python Notebooks com limb-darkening quadrático.

O segundo teste foi realizado simulando uma curva de luz com um parâmetro de *limbdarkening* exponencial:

```
In [136]: import batman
          import numpy as np
          import matplotlib.pyplot as plt

In [137]: params = batman.TransitParams()
          params.t0 = 0.                       #time of inferior conjunction
          params.per = 2.                      #orbital period
          params.rp = 0.5                      #planet radius (in units of stellar radii)
          params.a = 20.                       #semi-major axis (in units of stellar radii)
          params.inc = 91.                     #orbital inclination (in degrees)
          params.ecc = 0.                      #eccentricity
          params.w = 90.                       #longitude of periastron (in degrees)
          params.u = [0.1, 0.3]                #limb darkening coefficients
          params.limb_dark = "exponential"       #limb darkening model

In [138]: t = np.linspace(-0.05, 0.05, 100)

In [139]: m = batman.TransitModel(params, t)    #initializes model
          flux = m.light_curve(params)          #calculates light curve

In [140]: plt.plot(t, flux)
          plt.xlabel("Time from central transit")
          plt.ylabel("Relative flux")
          plt.show()
```

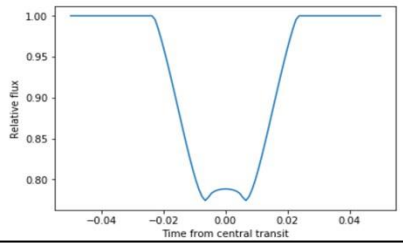

Figura 6: teste utilizando Python Notebooks com limb-darkening exponencial.

Em seguida são explicados os modelos utilizados para aplicar o método MCMC usando uma implementação em Python chamada Emcee (GOODMAN, WEARE, 2010).

O modelo probabilístico generativo

Quando abordamos um novo problema, o primeiro passo é escrever a função de verossimilhança, ou seja, a função dos parâmetros de um modelo estatístico que permite inferir sobre o seu valor a partir de um conjunto de observações. Isso equivale a descrever o procedimento generativo para os dados. Isso equivale a descrever o procedimento generativo para os dados. Neste caso, vamos considerar um modelo linear, onde as incertezas citadas são subestimadas por uma quantidade fracionada constante. Você pode gerar um conjunto de dados sintético desse modelo:

```python
import numpy as np

# Choose the "true" parameters.
m_true = -0.9594
b_true = 4.294
f_true = 0.534

# Generate some synthetic data from the model.
N = 50
x = np.sort(10*np.random.rand(N))
yerr = 0.1+0.5*np.random.rand(N)
y = m_true*x+b_true
y += np.abs(f_true*y) * np.random.randn(N)
y += yerr * np.random.randn(N)
```

Figura 7: trecho de código em Python para gerar um conjunto de dados sintético do modelo probabilístico.

Fonte: (FOREMAN-MACKEY, HOOG, LANG, GOODMAN, 2012)

Este conjunto de dados sintéticos (com as barras de erro subestimados) será algo parecido com:

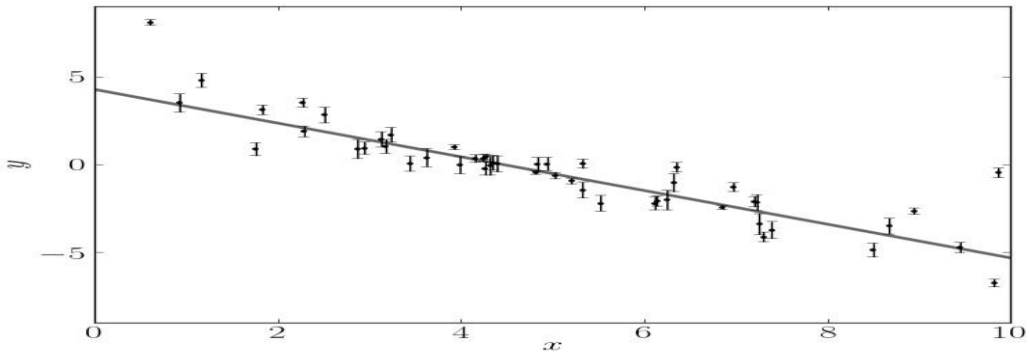

Figura 8: conjunto de dados sintéticos com as barras de erro subestimados.

Fonte: (FOREMAN-MACKEY, HOOG, LANG, GOODMAN, 2012)

O modelo verdadeiro é mostrado como a linha cinza grossa e o efeito das incertezas subestimadas é óbvio quando você olha essa figura. A maneira padrão de ajustar uma linha a esses dados (assumindo barras de erro gaussianas independentes) é pelo método dos mínimos quadrados. O método dos mínimos quadrados é atrativo porque a resolução dos parâmetros - e suas incertezas associadas - é simplesmente uma operação algébrica linear. Após a notação em (HOGG, BOVY & LANG 2010), a solução linear de mínimos quadrados para esses dados é:

```
A = np.vstack((np.ones_like(x), x)).T
C = np.diag(yerr * yerr)
cov = np.linalg.inv(np.dot(A.T, np.linalg.solve(C, A)))
b_ls, m_ls = np.dot(cov, np.dot(A.T, np.linalg.solve(C, y)))
```

Para o conjunto de dados gerado acima, o resultado corresponde a linha tracejada plotada no gráfico abaixo:

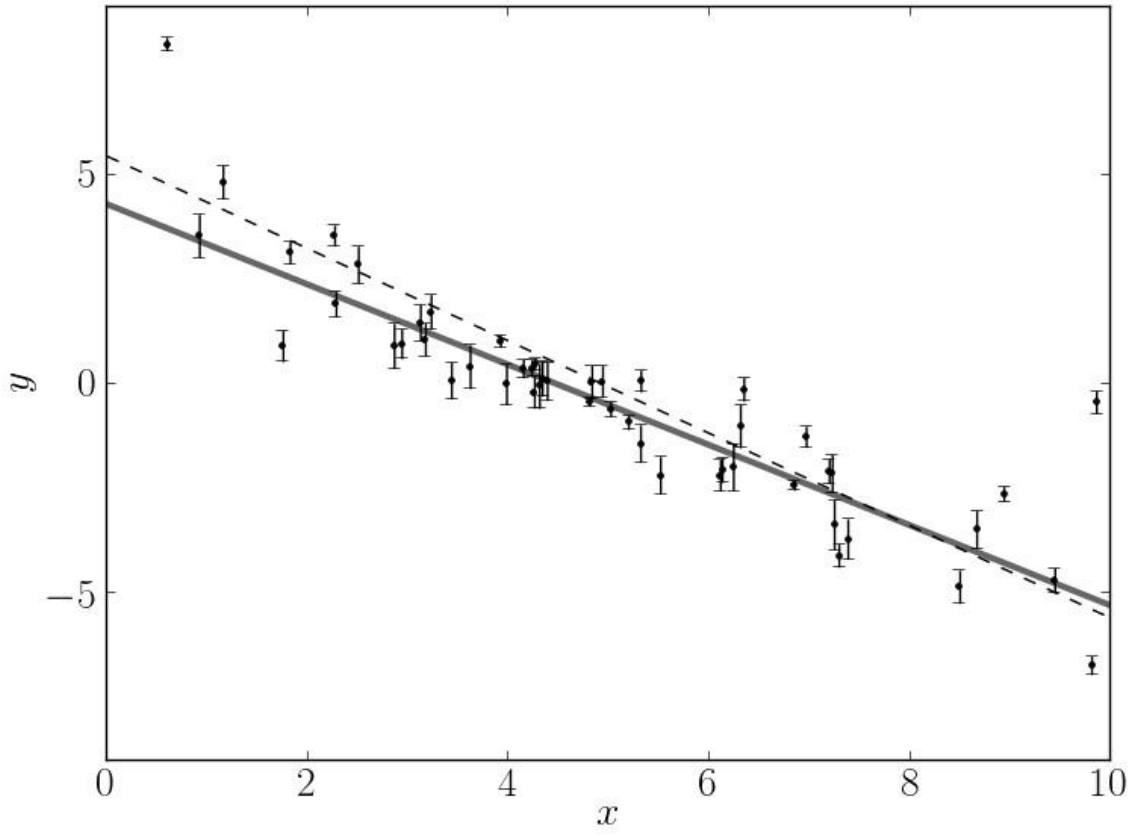

Figura 10: conjunto de dados sintéticos com as barras de erro subestimados.

Fonte: (FOREMAN-MACKEY, HOOG, LANG, GOODMAN, 2012)

Este não é um resultado exorbitante, mas as incertezas na inclinação e na interceptação parecem um pouco pequenas (por causa das pequenas barras de erro na maioria dos pontos de dados).

O método Monte Carlo com Cadeias de Markov (MCMC)

É preciso começar escrevendo a função de probabilidade posterior (até uma constante):

$$p(m, b, f \mid x, y, \sigma) \propto p(m, b, f)\, p(y \mid x, \sigma, m, b, f)$$

Escrever a função de verossimilhança, mostrada anteriormente:

$$p(y \mid x, \sigma, m, b, f)$$

E então o componente anterior a função:

$$p(m, b, f)$$

Esta função codifica qualquer conhecimento prévio que temos sobre os parâmetros: resultados de outras experiências, faixas fisicamente aceitáveis, etc. É necessário que você anote os priores se você estiver usando o MCMC. Porque tudo o que MCMC faz é tirar amostras de uma distribuição de probabilidade e você quer que seja uma distribuição de probabilidade para os seus parâmetros. Não devemos desenhar amostras de parâmetros de sua função de verossimilhança. Isso ocorre porque uma função de verossimilhança é uma distribuição de probabilidade sobre conjuntos de dados, portanto, condicionada aos parâmetros do modelo, você pode desenhar conjuntos de dados representativos (como demonstrado no início deste exercício), mas você não pode desenhar amostras de parâmetros.

Neste exemplo, usaremos os priores uniformes em m, b e o logaritmo de f. Por exemplo, usaremos o seguinte conservador anterior em m:

$$p(m) = \begin{cases} 1/5.5\,, & \text{if } -5 < m < 1/2 \\ 0\,, & \text{otherwise} \end{cases}$$

Após toda esta configuração, é mais fácil amostrar esta distribuição usando Emcee**.** Emcee é uma implementação em Python da amostragem do MCMC. Começaremos por inicializar os andadores (*walkers*) em uma pequena bola gaussiana em torno do resultado da máxima verossimilhança:

```
ndim, nwalkers = 3, 100
pos = [result["x"] + 1e-4*np.random.randn(ndim) for i in range(nwalkers)]
```

Então, podemos configurar o modelo:

```
import emcee
sampler = emcee.EnsembleSampler(nwalkers, ndim, lnprob, args=(x, y, yerr))
```

E executar o MCMC por 500 passos:

```
sampler.run_mcmc(pos, 500)
```

Vamos dar uma olhada no que o modelo fez. A melhor maneira de ver isso é olhar para a série temporal dos parâmetros na cadeia. O objeto do modelo agora tem um atributo chamado cadeia que é uma matriz com a forma (100, 500, 3) dando os valores dos parâmetros para cada *walker* em cada etapa da cadeia. A figura abaixo mostra as posições de cada *walker* em função do número de etapas na cadeia:

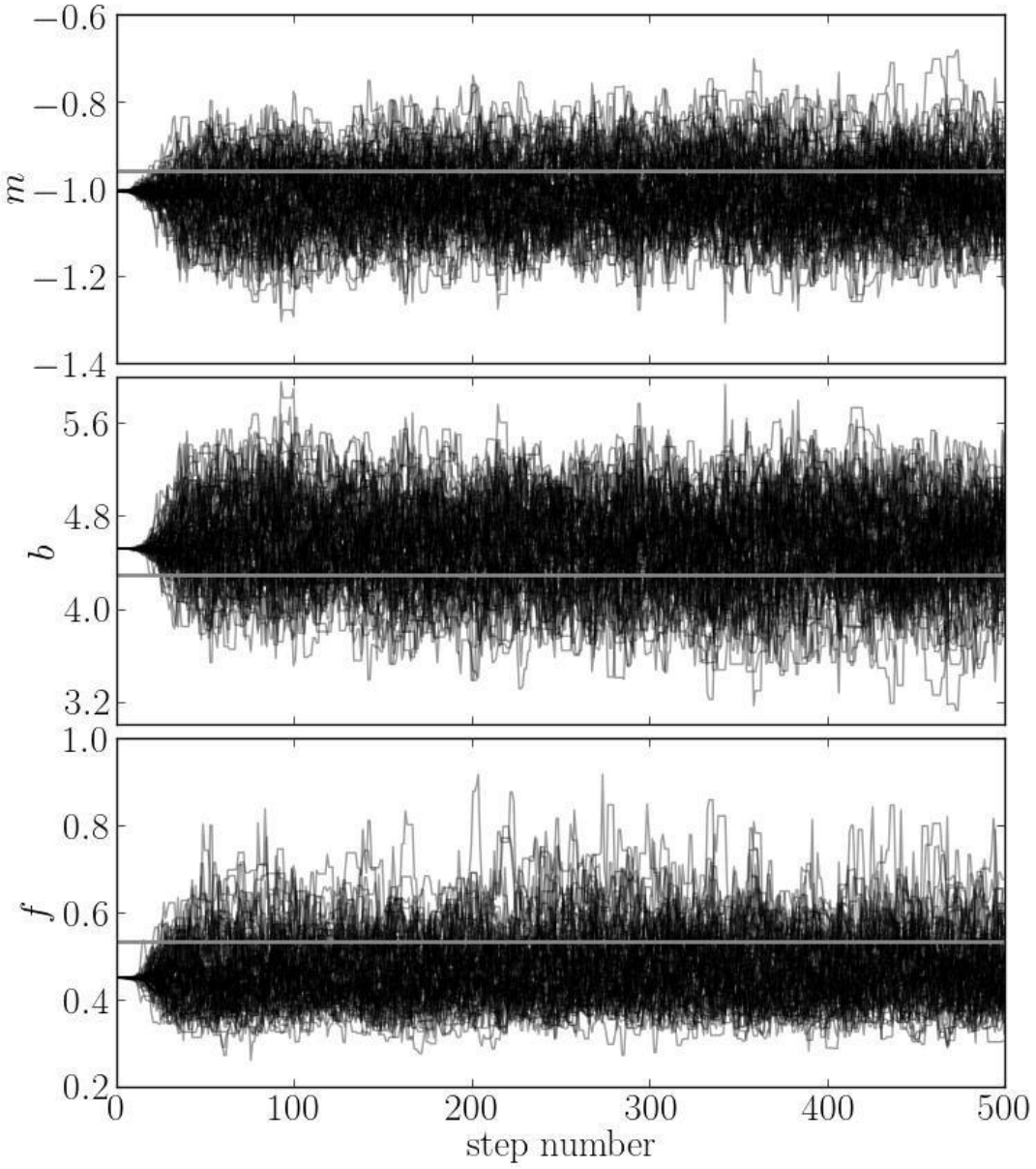

Figura 11: posições de parâmetros *walker* em função do número de etapas na cadeia.

Fonte: (FOREMAN-MACKEY, HOOG, LANG, GOODMAN, 2012)

E assim os valores reais dos parâmetros são indicados como linhas cinza em cima das amostras.

4. RESULTADOS E DISCUSSÃO

O primeiro teste feito foi a aproximação do valor de período orbital em uma curva sintética com erro muito pequeno, o valor real do período orbital foi definido como 1 e a função de verossimilhança recebeu um valor de erro de $\sigma^2 = 0.00001$ para representar um erro ínfimo.

Foram gerados 1000 pontos na curva de luz sintética e 2500 iterações foram feitas no amostrador de Ensemble, onde as 500 primeiras são descartadas no processo de *burn-in*.

O resultado da aproximação pode ser visto na Figura abaixo, onde as linhas (aproximações) sobrepõem os pontos (amostras da curva de luz sintética). O teste durou cerca de 1h e o valor médio encontrado pelos walkers para o parâmetro de período orbital foi de 1.01.

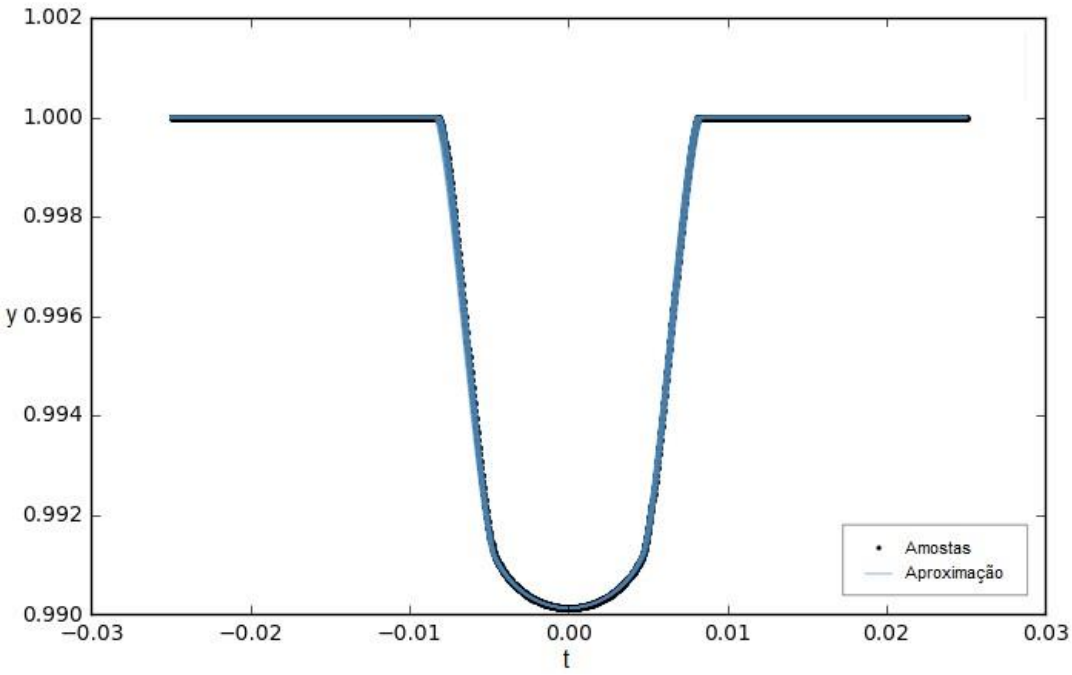

Figura 12: Gráfico da aproximação do amostrador de Ensemble para a curva de luz sintética com 1000 pontos sem ruído.

No teste seguinte, a mesma configuração foi utilizada, porém, dessa vez foi adicionado ruído na ordem de milésimos aos dados. Este teste, tendo a mesma quantidade de pontos do teste anterior, também demorou cerca de 1h e o valor médio encontrado pelos *walkers* foi 0.99.

O resultado está ilustrado na Figura 13, abaixo.

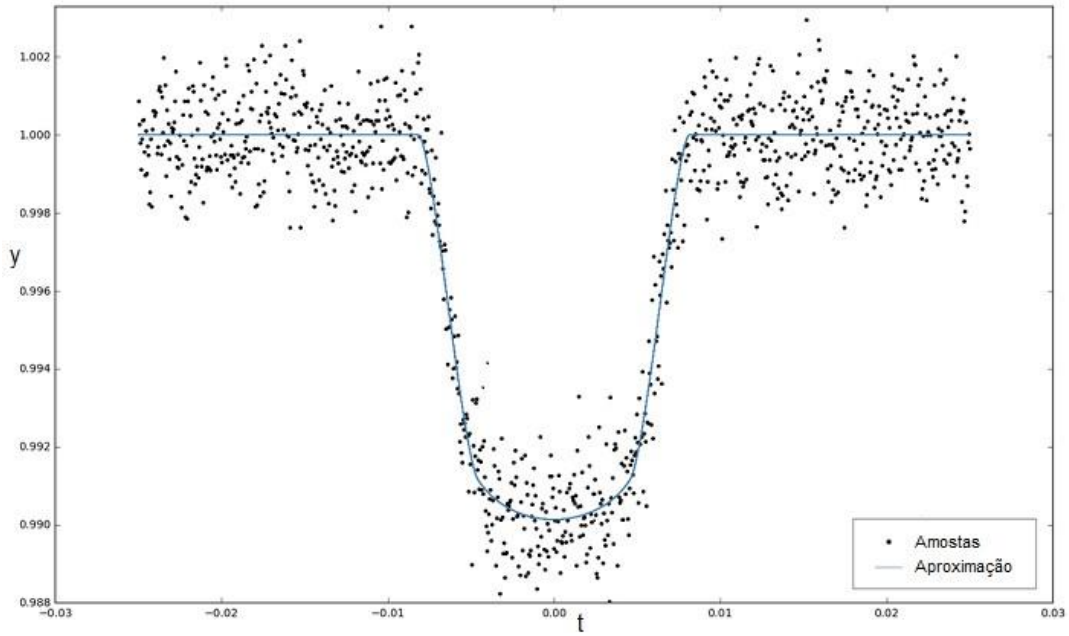

Figura 13: Gráfico da aproximação do amostrador de Ensemble para a curva de luz sintética com 1000 pontos e ruído na ordem de milésimos.

No terceiro teste um cenário mais próximo de uma situação real foi gerado, foram gerados 5000 pontos e o ruído adicionado aos dados é da ordem de centésimos. É possível notar na Figura 14 que os dados reais chegam a perder seu formato curvo original. O teste demorou cerca de 1h30min (decorrente da maior quantidade de pontos) e o valor médio obtido pelos walkers foi de 0.998.

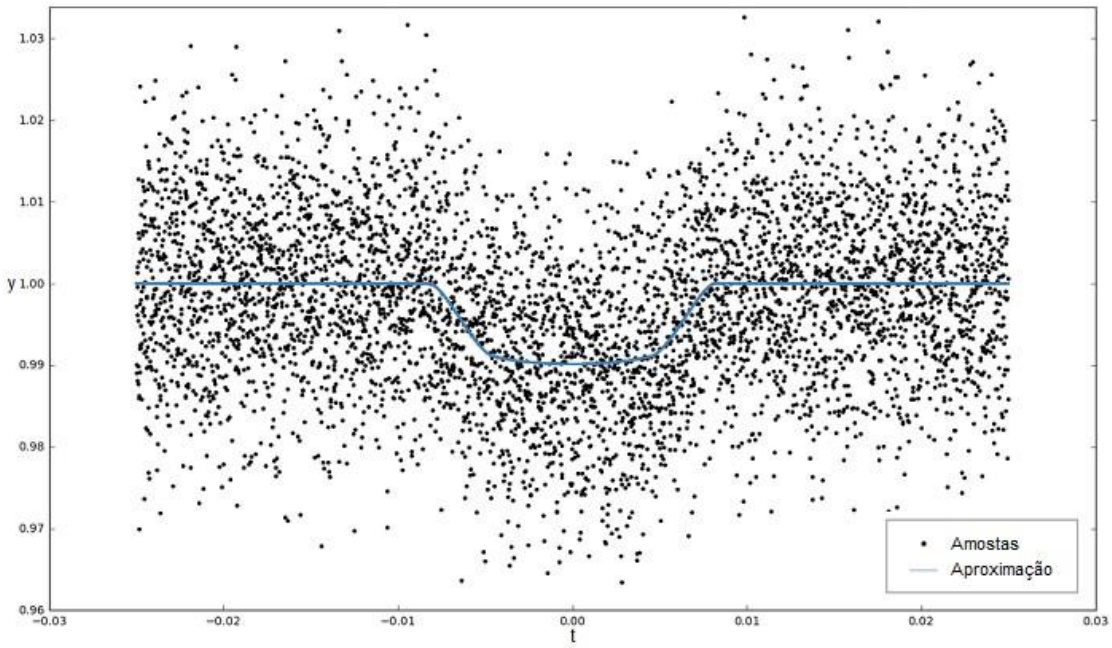

Figura 14: Gráfico da aproximação do amostrador de Ensemble para a curva de luz sintética com 5000 pontos e ruído na ordem de centésimos.

É importante ressaltar que, no processo de burn-in, os walkers já alcançavam os valores de aproximação dados no final do processo, portanto, os mesmos resultados obtidos nos testes feitos podem ser alcançados em cerca de 20 min. Além disso, para problemas como o que foi trabalhado, onde há apenas um parâmetro a ser aproximado, notou-se que existe uma quantidade máxima de iterações que alcançam o resultado de forma satisfatória. Para o teste com 5000 pontos, por exemplo, cerca de 500 iterações são suficientes, ultrapassar essa quantidade não surtiu efeito significativo no resultado final.

Todos os testes forem executados em uma máquina Ubuntu, com processador quadcore de 3.5 GHz e 2GB de memória RAM.

5. CONSIDERAÇÕES FINAIS

A busca por exoplanetas através do método dos trânsitos planetários consiste na análise de amostras de curvas de luz, coletadas através de fotometria, por missões como a do satélite Kepler. Uma curva de luz é composta de amostras do fluxo de luz de uma estrela observada, no qual ocorre uma queda quando um planeta passa em frente a estrela. A identificação de exoplanetas possui grande relevância para diversos estudos, tal como a busca por planetas semelhantes a terra ou que possam abrigar vida. Para tal, é interessante o estudo de diversos aspectos dos exoplanetas, como, por exemplo, tamanho e massa do planeta e da estrela, distância entre estrela e planeta, composição da atmosfera e etc. Dados como composição da atmosfera e distância entre o planeta e a estrela são utilizados para determinar se o exoplaneta pode abrigar vida como a conhecemos. Estudos como estes podem, ainda, proporcionar melhor entendimento para o processo de formação de um planeta.

Os métodos determinísticos utilizados no processamento das amostras de trânsito planetário, como, por exemplo, o *Box Least Squares,* demandam grande tempo de processamento, podendo levar até 33 dias para processar amostras de uma curva de luz que por sua vez podem conter até 200 mil pontos (BASILE, A. L.; VALIO, A. R. A.; SILVA, L.). Tendo em vista a grande demanda por tempo de processamento dos algoritmos determinísticos, neste trabalho, foi implementada uma aplicação de abordagem probabilística, uma vez que algoritmos probabilísticos possuem tempo de convergência, em geral, menor. A aplicação foi feita em Python, utilizando os módulos emcee e batman. O módulo emcee implementa o método MCMC e o amostrador de Ensemble com invariância afim. Já o módulo batman fornece um meio para geração de amostras sintéticas de curvas de luz de trânsitos planetários.

Nos testes feitos foram disponibilizados ao módulo emcee as amostras da curva de luz sintética (criada utilizando o módulo batman), uma função de probabilidade *a priori* que diz

respeito aos valores dos dados que se quer aproximar e uma função para o cálculo de verossimilhança entre as amostras fornecidas e as aproximações geradas.

Em seguida, são executadas várias iterações, nas quais, novas aproximações da curva de luz são geradas. Para cada curva aproximada gerada é feito o cálculo da verossimilhança entre esta curva e as amostras da curva de luz fornecida e o valor obtido é utilizada pelo módulo emcee para avaliar o quão próximo a aproximação está das amostras. Este processo é repetido uma quantidade estabelecida de vezes e ao final é obtido o valor aproximado do período orbital nas amostras utilizadas.

Os testes executados na aplicação alcançaram resultados satisfatórios em tempo médio de 1 hora para casos de curvas com 5000 pontos, executando 2500 iterações. É importante ressaltar que no processo de testes foi notado que a quantidade de pontos não necessariamente influencia na quantidade de iterações necessárias para obter um bom resultado. Ainda, para o caso da curva com 5000 pontos, notou-se que o resultado obtido com as 2500 iterações pode ser alcançado com apenas 500, o que levaria em torno de 20 minutos.

Além de obter resultados satisfatórios utilizando pouco tempo, a aplicação é um exemplo do potencial existente na união de componentes que implementam algoritmos probabilísticos e componentes que especificam modelos de geração de dados de alguma natureza. Neste trabalho, o módulo batman utilizado, que fornece a geração de curvas de luz sintéticas de trânsitos planetários, foi implantado como modelo de dados na implementação do código, o que foi essencial para que o módulo emcee pudesse gerar as curvas aproximadas e fazer o ajuste do parâmetro do período orbital.

Apesar do potencial demonstrado pela aplicação implementada neste trabalho, existem certas limitações em relação ao modelo de dados. A aplicação não comtempla amostras de curvas de luz onde exista trânsito de mais de um planeta. Além disso, também não são contemplados modelos para estrelas que possuem manchas.

Uma possível extensão do trabalho poderia ser a expansão da aplicação para comtemplar as limitações citadas anteriormente. Tal extensão poderia ser feita, por exemplo, no próprio módulo batman, adaptando-o para fornecer meios para gerar amostras de curvas de luz com trânsito de mais de um planeta, bem como estender os parâmetros relacionados a estrela para especificação de manchas. Além disso, é possível a paralelização da função de verossimilhança para redução do tempo de processamento e adaptação da execução das iterações do amostrador de Ensemble para que, ao invés de especificar uma quantidade de iterações, a convergência do período orbital seja identificada através de uma métrica e apenas as iterações necessárias sejam executadas.

Outro trabalho futuro importante é a aplicação do código implementado em amostras reais de curva de luz, como as coletadas pelo satélite Kepler, onde a princípio poderiam ser testadas curvas as quais já se conhecem os resultados ou que sejam similares aos testes feitos. De tal forma que, seja descoberta a confiabilidade desta aplicação e ela possa ser implantá-la como um método viável para a busca e identificação de exoplanetas.

6. REFERÊNCIAS

BASILE, A.L. Serviço Local de Periodograma com Esquemas de Map-Reduce em Arquiteturas Paralelas para Detecção de Trânsitos Planetários. São Paulo: Universidade Presbiteriana Mackenzie, 2016.

CHONG, E.K.P. An Introduction to Optimization. New York: Wiley, 2013.

CONNOLY, A.D. Statistics, Data Mining, and Machine Learning in Astronomy: A Practical Python Guide for the Analysis. Princeton: Princeton University Press, 2014.

FOREMAN-MACKEY D., HOOG D.W., LANG D., GOODMAN J. Emcee: The MCMC Hammer. Ithaca: Cornell University, 2012.

GAMERMAN, D., LOPES, H.F. Markov Chain Monte Carlo: Stochastic Simulation for Bayesian Inference. New York: Chapman & Hall, 2015.

GOODMAN J., WEARE J. Ensemble samplers with affine invariance. New York: New York University, 2010.

HART, W.E. Optimization Modeling in Python. New York: Springer, 2012.

KEPLER. Kepler: A Search for Habitable Planets. Disponível em: http://kepler.nasa.gov. Acesso em: 02/05/2015.

KREIDBERG, L. Batman: Basic Transit Model Calculation in Python. Chicago: University of Chicago Press, 2015.

PERRYMAN, M. The Exoplanet Handbook. Cambridge: New York, 2014.

PFEFFER, A. Practical Probabilistic Programming. New York: Manning Publication, 2016.

SEAGER, S. Exoplanets. Phoenix: University of Arizona Press, 2010.

SIRCA, S. Computational Methods for Physicists. New York: Springer, 2012.

Contatos: juliolbertolacini@gmail.com e luciano.silva@mackenzie.br