Dinâmica Hamiltoniana, Stan e modelagem Bayesiana de regressão

Prof. Vinícius D. Mayrink
EST088 - Inferência Bayesiana

Sala: 4073 Email: vdm@est.ufmg.br

1^o semestre de 2025

6.1 - Algorítmos MCMC sob a dinâmica Hamiltoniana

Uma desvantagem do Metropolis-Hastings (MH) é o fato de que um novo candidato é sempre amostrado em uma vizinhança do atual valor da cadeia.

Este ponto implica que o MH, em alguns problemas, pode não ser eficaz para propor valores em regiões do espaço paramétrico distantes da posição atual da cadeia. O custo disso é uma demora (requisição de muitas iterações) para percorrer grande parte do espaço do parâmetro alvo.

O *Hamiltonian Monte Carlo* (HMC) é um método de amostragem indireta, a partir da distribuição *a posteriori*, que foi criado com o objetivo de evitar este ponto fraco do MH.

A inspiração do HMC surgiu de questões estudadas na área de física, dentro de um tema conhecido como dinâmica Hamiltoniana. Alguns princípios básicos são discutidos a seguir.

Um sistema Hamiltoniano pode ser conceitualizado através do comportamento de uma bola movendo-se ao longo do tempo em uma superfície sem atrito (ex.: imagine uma bolinha de gude movimentando dentro de uma tigela).

A bolinha está sob efeito da gravidade (puxando para baixo) e de seu impulso (que faz ela seguir em uma direção).

A trajetória contínua que a bolinha vai fazer pode ser descrita matematicamente por um conjunto de equações diferenciais, as quais envolvem o cálculo de gradientes e derivadas. Quem estuda física, aprende a trabalhar com esse problema. Veremos mais adiante como isto se encaixa no HMC.

Alguns conceitos importantes estão ligados à bolinha de gude:

- Sua posição (um vetor de coordenadas) em um plano (a mesa) imediatamente abaixo da tijela. A posição varia com o tempo na movimentação da bolinha.
- O impulso que a bolinha tem em dada posição.
- A energia potencial (altura da bola dentro da tijela) em dada posição.
- A energia cinética (energia de movimentação), a qual está associada ao impulso da bolinha em dada posição.

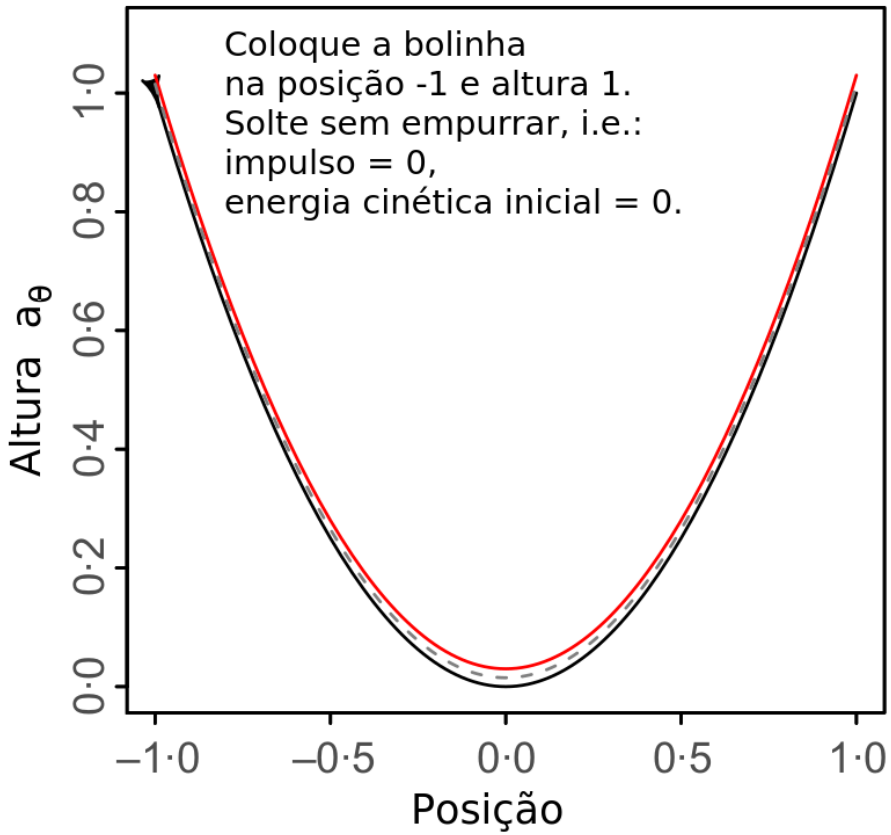
Visto que a superfície é sem atrito, a energia total ($H =$ potencial $+$ cinética) permanece constante ao longo do tempo. Esta energia total H é conhecida na física por **Hamiltonian**.

Mais adiante veremos que no contexto de métodos MCMC:

- vetor de posição $=$ vetor θ de parâmetros do modelo.
- energia potencial $= -\ln[f_{\theta|\boldsymbol{Y}}(\theta|\boldsymbol{y})]$.

Para que os conceitos fiquem mais concretos, vamos simplificar a superfície substituindo a tijela por uma parábola.

Suponha θ escalar (coordenada na bolinha) e associe este valor com a energia potencial (altura a_θ) pela formulação $a_\theta = \theta^2$.



A gravidade irá puxar a bolinha para baixo, acelerando o deslocamento ao longo do tempo.

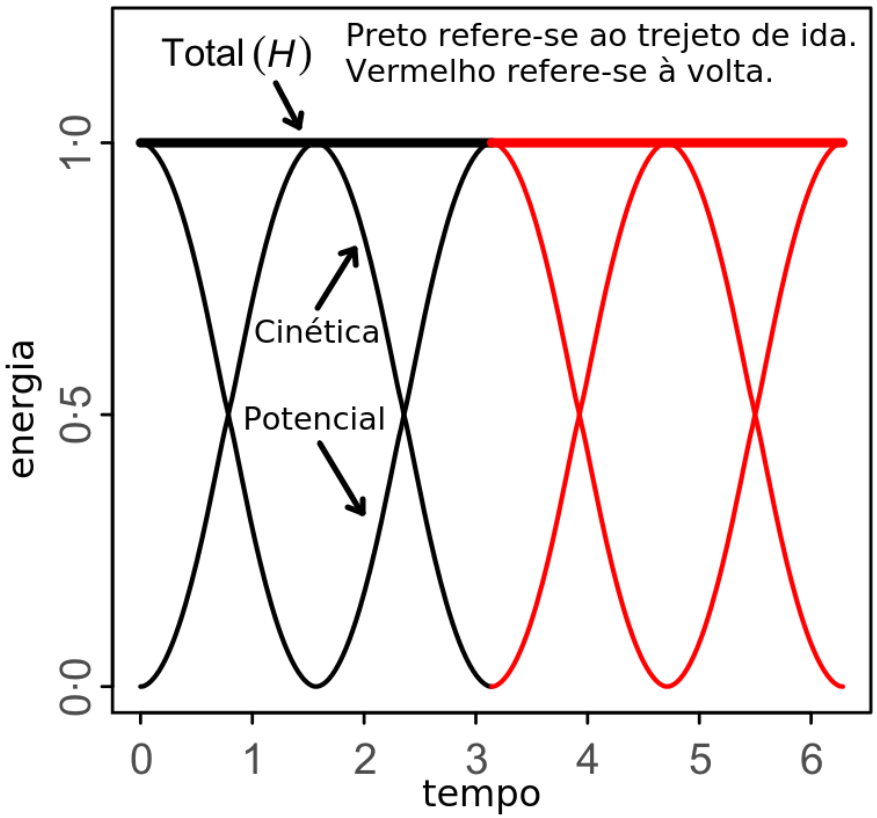
Energia potencial (altura) é convertida em energia cinética.

O impulso fará a bolinha ultrapassar $\theta = 0$, posição na qual toda energia potencial foi convertida em energia cinética.

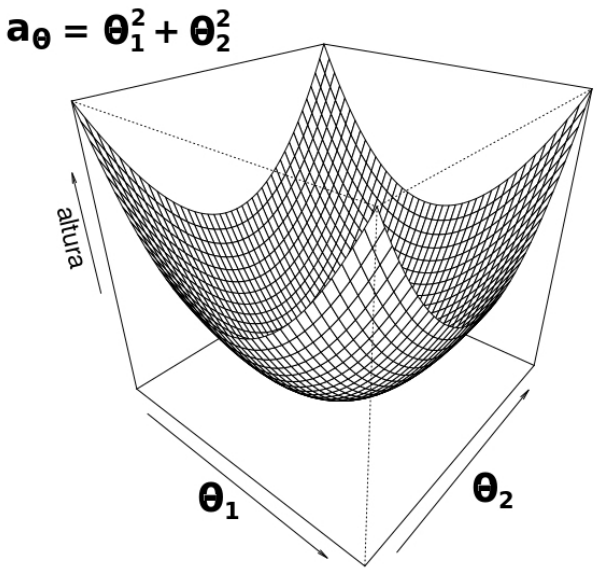
Não há atrito, então bolinha vai parar na posição 1 com altura 1. Aqui a energia cinética volta a 0.

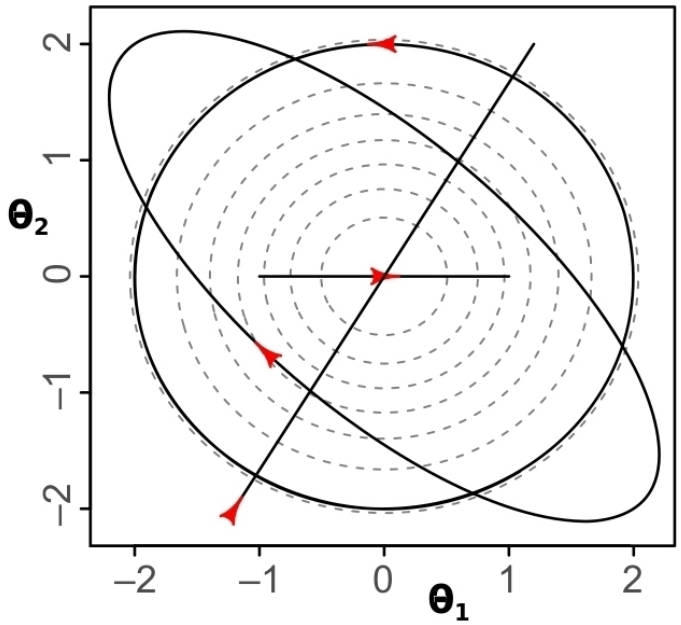
A bolinha vai reverter sua trajetória e oscilar eternamente nesse caminho. Energias potencial e cinética variam ao longo do tempo, mas a soma delas (H) permanecerá constante.

A figura abaixo esquematiza a troca entre energia potencial e cinética ao longo do tempo. Lembre-se que: H = energia potencial + energia cinética.



Considere agora uma parábola construída com 2 coordenadas: $a_\theta = \theta_1^2 + \theta_2^2$.
O formato é ilustrado abaixo.





Empurrando a bolinha para a esquerda (ou direita), ela fará uma trajetória circular com altura constante.

Empurrando para baixo, ela cruza o fundo e sobe o lado oposto da superfície.

Empurrando para a esquerda (ou direita) com ângulo direncionando para baixo, a trajetória será elíptica.

trajeto da bolinha.
curva de nível da superfície.

A posição agora é um vetor bi-dimensional $\theta = (\theta_1, \theta_2)^\top$.

Iremos empurrar, com alguma direção, a bolinha partindo do local θ.

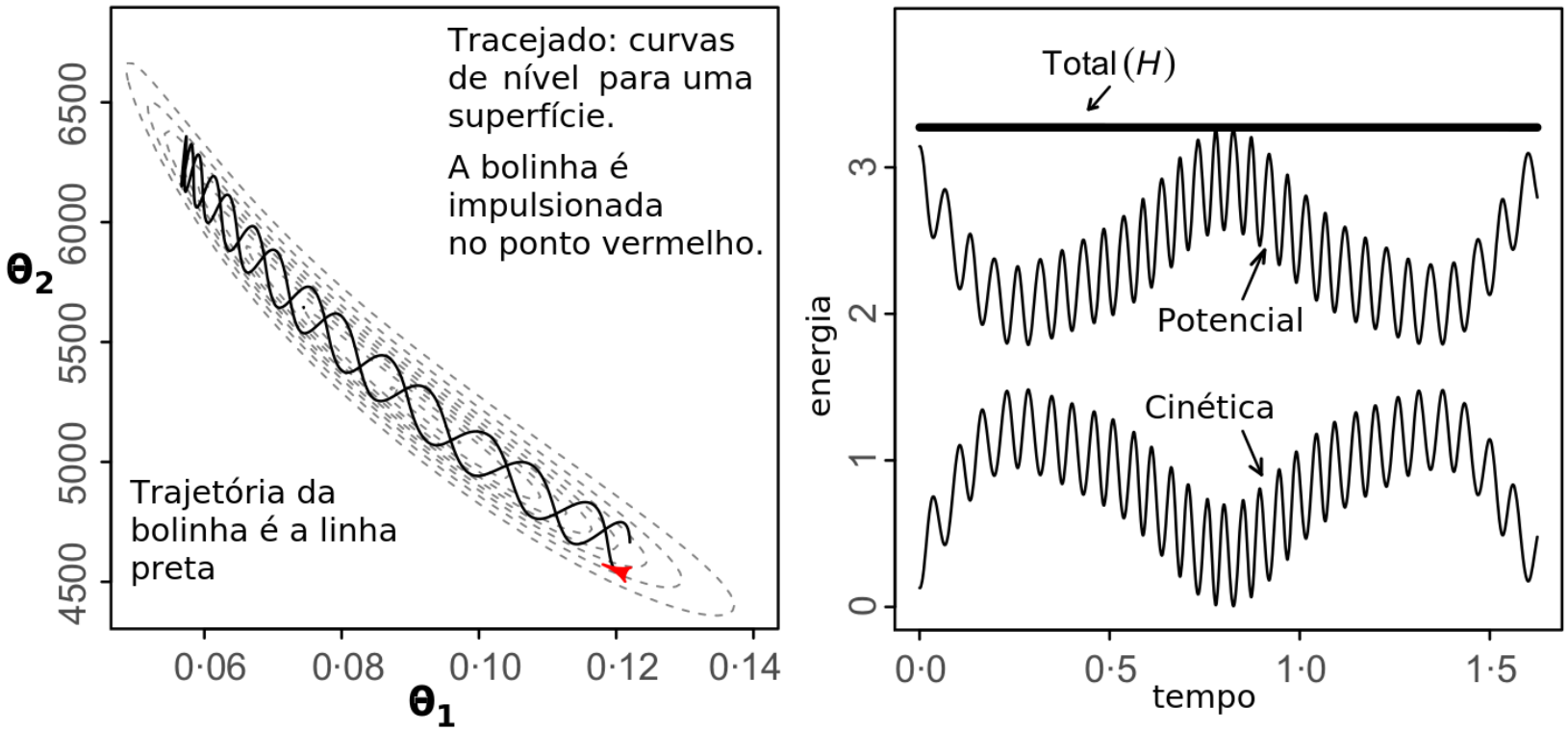
No espaço de 2 coordenadas, o impulso será um vetor bi-dimensional usualmente representado por $\mathbf{p} = (\mathrm{p}_1, \mathrm{p}_2)^\top$.

O vetor $\mathbf{p}$ possui informação sobre direção e magnitude do impulso:

- p_1 representaria a velocidade na direção do eixo θ_1.
- p_2 representaria a velocidade na direção do eixo θ_2.
- $\|\mathbf{p}\|^2 = \mathrm{p}_1^2 + \mathrm{p}_2^2$ representaria a magnitude do impulso.
  Pode-se ver este valor como uma medida de energia cinética da bolinha.

Energias potecial e cinética permanecem sendo escalares.

A lógica discutida até aqui será válida para superfícies mais complicadas.



Os princípios da dinâmica Hamiltoniana proporcionam uma maneira eficiente de realizar a transição de um valor para outro no contexto de um algorítmo MCMC.

A bolinha pode se mover para quase qualquer lugar do espaço paramétrico, dada uma escolha de tempo no qual deixaremos ela movendo e um valor de impulso inicial. O impulso terá uma direção que determinará trajetórias variadas. Isto evita o problema de transição de um valor θ para outro na vizinhança.

Os algorítmos MCMC que utilizam a dinâmica Hamiltoniana são de dois tipos:

- Hamiltonian Monte Carlo estático (SHMC - *Static HMC*).
- No-U-Turn Sampling (NUTS).

Nos próximos slides iremos descrever os aspectos principais destes métodos.

Hamiltonian Monte Carlo estático - SHMC:

Este foi o 1^o método MCMC a utilizar a dinâmica Hamiltoniana. Inicialmente era chamado de Monte Carlo híbrido (*hybrid Monte Carlo*).

Sua divulgação ocorreu no artigo: *Duane, S., Kennedy, A.D., Pendleton, B.J. and Roweth, D. (1987) Hybrid Monte Carlo. Physics Letters, B, 195, 216-222*.

O algorítmo NUTS, que será comentado mais adiante, é uma versão melhorada do SHMC. Grande parte do que for apresentado aqui servirá para entender o NUTS.

A processo de transição (de um valor para outro no espaço paramétrico) ocorre no SHMC pela simulação de uma bolinha partindo de uma posição atual θ (com um impulso aleatório) e movendo-se por um período finito de tempo t. O novo valor de θ será dado pela posição da bolinha no instante t (fim da trajetória).

Três problemas trazem dificuldade para este processo de transição.
Eles são relatados a seguir.

Problema 1: Como simular o movimento da bolinha considerando superfícies arbitrárias representando a log f.d.p. *a posteriori* ? Em outras palavras, como saberemos o trajeto da bolinha?

Superfícies simples (i.e. modelos simples), como a parábola, possuem solução analítica baseada em equações diferenciais que determinam o trajeto (contínuo) exato da bolinha.

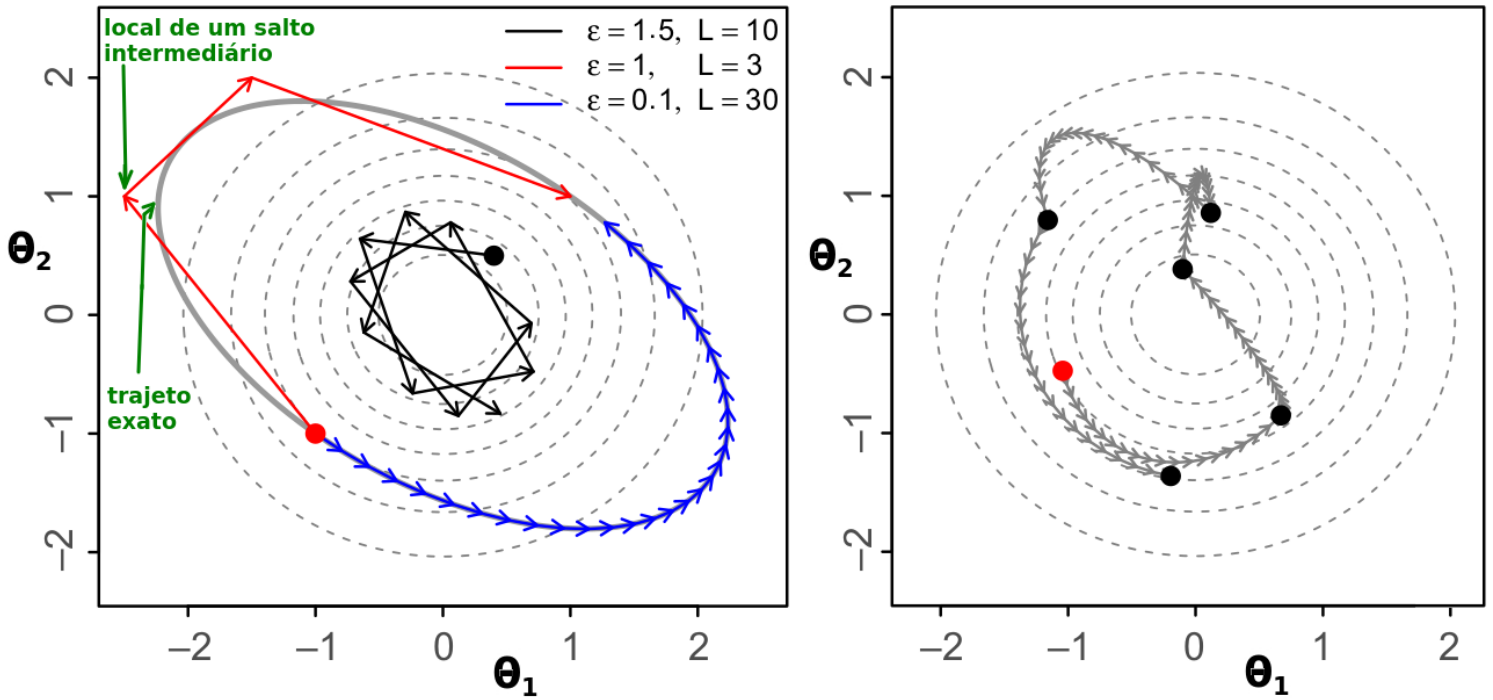
Entretanto, para muitos modelos estatísticos, a superfície em questão é complicada. A solução é aproximar trajetos por um método numérico conhecido como integração *leapfrog*.

A trajetória aproximada será estabelecida com base em:

- ε = tamanho de salto.
- L = número de saltos.

O vetor de posição θ correspondendo ao local do salto final L será a observação *a posteriori* gerada na iteração do MCMC.

Os saltos intermediários (anteriores a L) serão descartados.



Erros de aproximação ocorrem principalmente quando ε é grande. Isto implica que a bolinha desvia um pouco da trajetória exata contínua. Consequência: $H =$ energia potecial + cinética não é constante ao longo do tempo.

Problema 2: Determinação do comprimento de trajeto (produto: εL) ideal.

Se o comprimento de trajeto for muito curto, a transição para valores longe da vizinhança será impossível (estaremos na situação ineficiente dos passeios aleatórios).

Se o comprimento de trajeto for muito longo, a bolinha irá retornar ao ponto de partida (isto significa esforço computacional jogado fora).

Um SHMC eficiente depende da escolha do comprimento εL. Esta escolha vai variar de modelo para modelo. A recomendação é que o usuário teste diferentes valores εL (*tunagem*) na análise de seus dados e selecione aquela que forneceu transições explorando melhor o espaço paramétrico.

Problema 3: Determinar o tamanho de salto ε, dado um comprimento de trajeto pré-especificado.

Um comprimento de trajeto fixado pode ser atingido por meio de:

- Poucos saltos de tamanho grande.

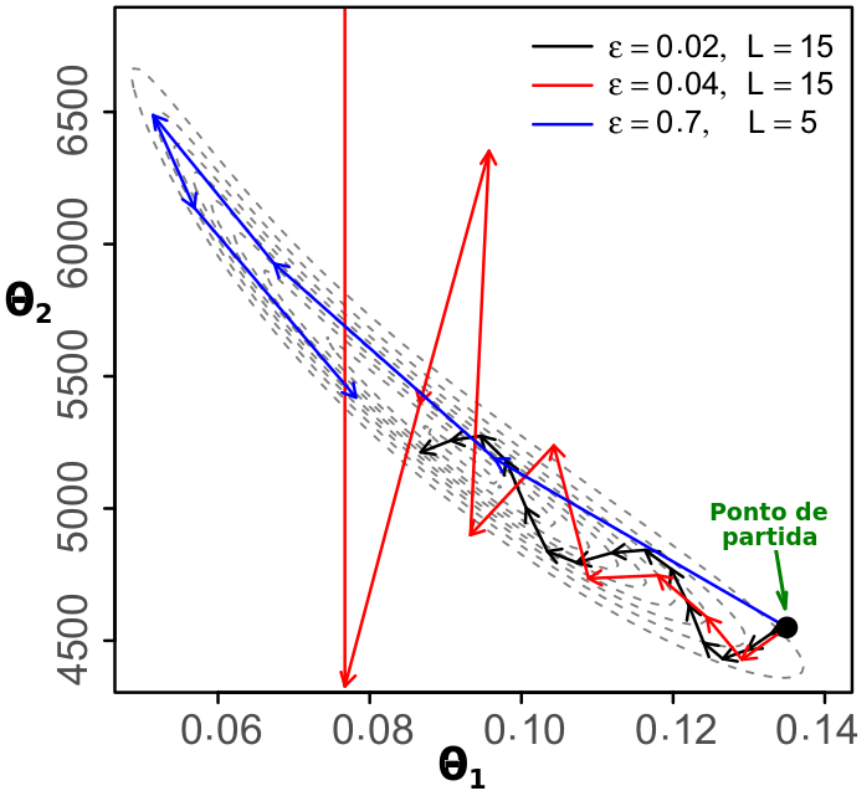
  Vantagem: cada salto tem custo computacional, então
  poucos saltos $\Rightarrow$ transição rápida para o novo θ.

  Desvantagem: saltos grandes $\Rightarrow$ maior variação na energia total H.
  O erro de aproximação acumula-se de forma que $H \to \infty$.
  Este problema é chamado de transição divergente (*divergent transition*).

- Muitos saltos de tamanho pequeno.

  Vantagem: saltos pequenos $\Rightarrow$ menor variação em H.
  A transição divergente é evitada.

  Desvantagem: alto custo computacional, transição demorada e MCMC lento.



Trajetórias partindo do mesmo ponto e com impulsos iguais:

Trajetória preta: lenta para percorrer a superfície.

Trajetória vermelha: acumula erros de aproximação, causando divergência.

Trajetória azul: construída com base em uma matriz massa que torna a superfície mais fácil de percorrer.

A matriz massa, mencionada acima, é melhor definida a seguir.

Uma vez definido o tamanho de salto ε e o n^o de saltos L, a etapa final é indicar a energia cinética em função do impulso $\mathbf{p}$.

No SHMC, isto é feito por meio da log-densidade de uma Normal Multivariada:

- A dimensão q é a mesma do vetor de coordenadas $\theta = (\theta_1, \cdots, \theta_q)^\top$ e do vetor de impulso $\mathbf{p} = (\mathrm{p}_1, \cdots, \mathrm{p}_q)^\top$.
- A média é zero, i.e., vetor $\mathbf{0}_q = (0, \cdots, 0)^\top$.
- A matriz de covariâncias (chamada matriz massa - *mass matrix*) é $q \times q$. Uma escolha simples aqui é a matriz identidade $\boldsymbol{I}_q$. Outras configurações podem ser usadas.

No caso simples, a energia cinética será escrita por:

$$E_C = -\ln[f_{\mathbf{p}}(\mathbf{p})] = -\left[-\tfrac{q}{2}\ln(2\pi) - \tfrac{1}{2}|\boldsymbol{I}_q| - \tfrac{1}{2}\,\mathbf{p}^\top \boldsymbol{I}_q^{-1}\mathbf{p}\right]$$

$$= \tfrac{q}{2}\ln(2\pi) + \tfrac{1}{2}\,(1) + \tfrac{1}{2}\mathbf{p}^\top\mathbf{p}.$$

Temos uma constante somada a um termo proporcional à magnitude do impulso $\mathrm{p}_1^2 + \mathrm{p}_2^2 + \cdots + \mathrm{p}_q^2$.

O efeito da matriz massa é uma transformação global na f.d.p. *a posteriori* para proporcionar uma geometria mais simples favorecendo a amostragem:

- As variâncias (diagonal da matriz massa) esticam/comprimem a f.d.p. *a posteriori* de forma que todos os parâmetros em θ possuam a mesma escala 1.
- As covariâncias rotacionam a f.d.p. *a posteriori* na direção de tornar os parâmetros em θ independentes.

Após a transformação, teremos média 0, escala 1 e nenhuma correlação.
Isto remete a variáveis i.i.d. $N(0, 1)$.

A matriz massa do SHMC tem papel similar ao da matriz de covariâncias especificada na distribuição geradora de proposta do Metropolis-Hastings. Sua especificação tem impacto substancial na eficiência da amostragem.

Dependendo do modelo, o SHMC pode funcionar bem com a matriz massa $\boldsymbol{I}_q$, mas este caso simples requer mais saltos L por transição $\Rightarrow$ maior tempo computacional.

Uma amostragem eficiente e rápida no SHMC depende de uma boa especificação do número de saltos L, tamanho dos saltos ε e da matriz massa.

No SHMC, o usuário terá um trabalho de *tunagem*, testando diferentes configurações destes elementos, para conseguir que o algorítmo seja eficiente na geração de valores *a posteriori*.

Felizmente, o algorítmo NUTS automatiza este processo, chegando ao ponto de não precisar ou precisar de pouquíssimas adaptações de *tunagem*.

O NUTS é um HMC melhorado com algumas particularidades visando eficiência de amostragem. Tudo o que foi explicado anteriormente sobre o SHMC vale para o NUTS. A discussão a seguir relata as diferenças que o NUTS possui.

No-U-Turn Sampling - NUTS:

Na automatização da tunagem, o NUTS determina o número de saltos L por meio de um sofisticado algorítmo de construção de árvore.

Uma trajetória da bolinha é construída no NUTS por meio de um acúmulo iterativo de saltos:

1. Um salto é realizado (do atual θ para outro).
   Até aqui, o trajeto envolve 2 pontos do espaço paramétrico.
2. Dois saltos adicionais são realizados $\Rightarrow$ trajeto formado por 4 pontos.
3. Quatro saltos adicionais são incluídos $\Rightarrow$ trajeto formado por 8 pontos.
4. Este procedimento, de dobrar o n^o de pontos do trajeto, avança até que uma das seguintes situações ocorra:
   - a trajetória comece a retornar (*U-Turn*) em direção ao ponto de partida da etapa (1).
   - a trajetória indique divergência ($H \to \infty$).

O n^o de iterações, dobrando a quantidade de pontos do trajeto, é chamado de profundidade da árvore (*tree depth*).

Este algorítmo cria trajetórias que não são muito curtas ou muito longas.

O comprimento do trajeto pode variar para diferentes transições dentro do espaço paramétrico. Por exemplo, uma transição pode requerer 8 saltos e outra 128 saltos. Isso depende dos vetores de posição θ e de impulso $\mathbf{p}$.

O NUTS determina o tamanho de salto ε através de um procedimento de adaptação que envolve uma estatística para teste de aceitação de propostas. Tal procedimento é executado durante o período de *burn-in* (ou *warm-up*) do MCMC.

O valor de ε será fixado naquele que fornecer uma taxa de aceitação ideal. O *software* `Stan`, que aplica o algorítmo NUTS, permite ao usuário escolher o valor alvo da taxa de aceitação através do argumento `adapt_delta` (*default* é 0.95).

Aumentar `adapt_delta` $\Rightarrow$ diminuir ε.

Desvantagem do HMC: apenas parâmetros contínuos (com distribuições *a priori* contínuas) podem ser tratados na modelagem.

Parâmetros discretos, com uma f.m.p. *a priori*, não possuem gradiente e assim não possibilitam a solução de equações diferenciais para determinar as trajetórias da bolinha.

A atual versão do `Stan` não corrige este problema, mas espera-se que isso seja resolvido em futuras versões. Uma possível solução é alternar entre passos do Gibbs Sampling (para parâmetros discretos) e do HMC (para parâmetros contínuos).

Vídeos ilustrativos sobre Metropolis-Hastings, HMC e NUTS.

`Stan` é uma plataforma moderna e livre para modelagem estatística com alta performance computacional.

Milhares de usuários usam o `Stan` para análise de dados (seja do ponto de vista da inferência Clássica ou Bayesiana).

No caso Bayesiano, a plataforma aplica o algorítmo NUTS para amostragem indireta a partir de distribuições *a posteriori* sem forma fechada.

O `Stan` está integrado a diversos ambientes computacionais: `RStan` (`R`), `PyStan` (`Python`), `CmdStan` (`shell`), `MatlabStan` (`MATLAB`), `Stan.jl` (`Julia`), `StataStan` (`Stata`), `MathematicaStan` (`Mathematica`) e `ScalaStan` (`Scala`).

Neste curso, utilizaremos o pacote `rstan` para trabalhar com o Stan dentro do R.

Manuais, artigos e outras documentações dando suporte para aprendizagem e esclarecimentos sobre o Stan estão disponíveis na própria homepage do programa:

https://mc-stan.org/users/documentation/

O manual de referência com detalhes sobre a linguagem de programação Stan é bastante completo (possui mais de 600 páginas). Nele é possível tirar praticamente qualquer dúvida sobre implementação de um modelo Bayesiano.

As próximas etapas deste curso envolvem a utilização do `Stan` para ajustar alguns modelos de regressão sob a visão Bayesiana da inferência.

Os modelos escolhidos foram:

1. Modelo de regressão linear múltiplo para resposta normal.
2. Modelo de regressão logística para resposta binária.
3. Modelo de regressão Poisson para dados de contagem.

Na implementação de cada caso, serão apresentados e discutidos diversos detalhes sobre a programação relacionada ao `Stan` e sua interface com o `R`.