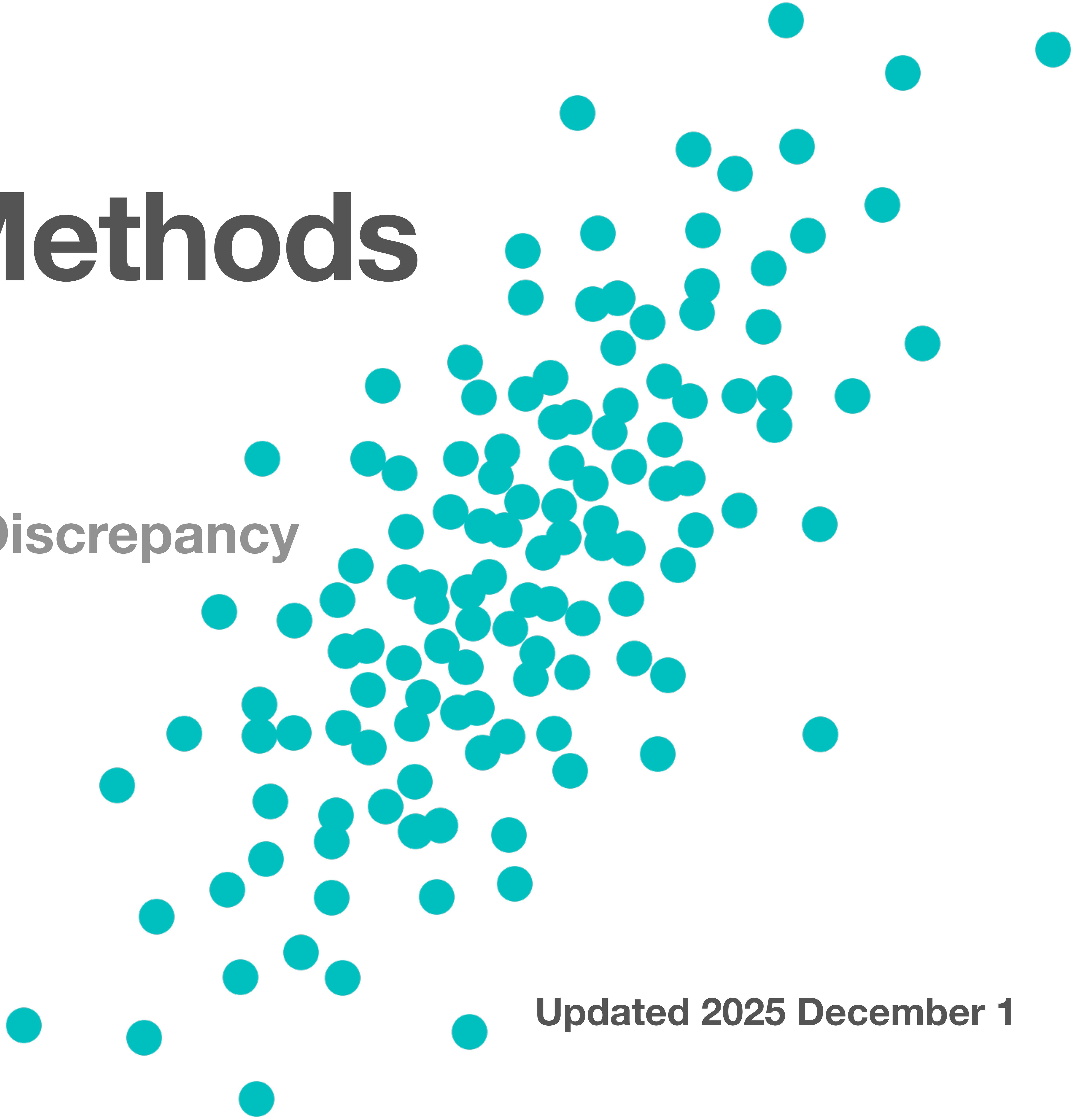
Monte Carlo Methods

Introduction
Generating Samples
Markov Chain Monte Carlo + Discrepancy
Improving Efficiency
Selected Topics

Git [website]() and [repository]()
[Canvas]()



Fred Hickernell, Fall 2025

Updated 2025 December 1

Selected Topics

Owen, Chapters ???

Project Presentations Nov 24–25

Take-Home Final Available Dec 3

Take-Home Final Due and
In-Class Final Dec 11, 2–4PM,
WH 115

Parallel Computing

Many personal computers now have GPUs built in. This allows one to perform computations in parallel using PyTorch. Here are some things to keep in mind

- Monte Carlo calculations are pleasantly parallel because the function values can be obtained independently, and one can even compute multiple expectations independently.
- Calculations on GPUs are typically using 32-bit floating point, not 64-bit floating point. This means that we have about 6 significant digits of accuracy, not 15.
- For small problems, GPUs may not be better than CPUs, but for large numbers of samples or large numbers of similar problems, the speed-up can be impressive.

Gradient Descent

Optimization problems take the form

$$\boldsymbol{\theta}_* = \underset{\boldsymbol{\theta} \in \mathbb{R}^m}{\operatorname{argmin}} \operatorname{Loss}(\boldsymbol{\theta})$$

Gradient descent updates the estimate of $\boldsymbol{\theta}_*$ iteratively by moving downhill:

$$\boldsymbol{\theta}_0 \text{ given}, \quad \boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \eta \nabla \operatorname{Loss}(\boldsymbol{\theta}_k),$$

where η is the step size or learning rate.

- This will often converge since $\nabla \operatorname{Loss}(\boldsymbol{\theta}_k) \to \mathbf{0}$ as $\boldsymbol{\theta}_k \to \boldsymbol{\theta}_*$
- But it may converge to a local minimum
- This may require more steps than methods that use second derivatives, but each step should be cheaper and easier to compute

Stochastic Gradient Descent

Loss function in regression often take the form

$$\mathrm{Loss}(\boldsymbol{\theta}) = \sum_{i=1}^{N} \mathrm{loss}_i(\boldsymbol{\theta}; \text{data}), \qquad \text{e.g., } \mathrm{loss}_i(\boldsymbol{\theta}; \text{data}) = [y_i - \theta_1 \exp(\theta_2 x_i)]^2$$

The loss to be optimized is a sum of many losses. If N is large, then the gradient is expensive. An alternative is stochastic gradient descent

$$\boldsymbol{\theta}_0 \text{ given}, \quad \boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \eta \nabla \mathrm{loss}_{i_k}(\boldsymbol{\theta}_k)$$

where $i_0, i_1, \ldots$ are chosen IID uniform over $1, \ldots, N$, which is cheap.

Stochastic Gradient Descent

Loss function in regression often take the form

$$\mathrm{Loss}(\boldsymbol{\theta}) = \sum_{i=1}^{N} \mathrm{loss}_i(\boldsymbol{\theta}; \text{data}), \qquad \text{e.g., } \mathrm{loss}_i(\boldsymbol{\theta}; \text{data}) = [y_i - \theta_1 \exp(\theta_2 x_i)]^2$$

gradient is expensive. An alternative is stochastic gradient descent

vanilla $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \eta \nabla \mathrm{loss}_{i_k}(\boldsymbol{\theta}_k)$

decaying η $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \eta_k \nabla \mathrm{loss}_{i_k}(\boldsymbol{\theta}_k), \quad \eta_k = \dfrac{\eta_0}{\sqrt{1 + k/100}}$

mini-batch $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \eta \dfrac{1}{B} \sum_{i_{kj}=1}^{B} \nabla \mathrm{loss}_{i_{kj}}(\boldsymbol{\theta}_k), \quad i_{kj} \overset{\text{IID}}{\sim} \mathscr{U}\{1, \ldots, N\}$

mini-batch + decay $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \eta_k \dfrac{1}{B} \sum_{i_{kj}=1}^{B} \nabla \mathrm{loss}_{i_{kj}}(\boldsymbol{\theta}_k), \quad i_{kj} \overset{\text{IID}}{\sim} \mathscr{U}\{1, \ldots, N\}\, \eta_k = \dfrac{\eta_0}{\sqrt{1 + k/100}}$

Two Level Monte Carlo

Let's start with a simple example: $Y = Y_1 + Y_2$ and we want to estimate

$$\mu := \mu_1 + \mu_2, \text{ where } \mu := \mathbb{E}(Y), \mu_1 := \mathbb{E}(Y_1), \mu_2 := \mathbb{E}(Y_2)$$

using IID samples

$$\widehat{\mu} := \widehat{\mu}_1 + \widehat{\mu}_2, \quad \widehat{\mu}_1 := \frac{1}{n_1} \sum_{i=0}^{n_1-1} Y_{i1}, \; \widehat{\mu}_2 := \frac{1}{n_2} \sum_{i=0}^{n_2-1} Y_{i2}, \qquad Y_{0,1}, Y_{1,1}, \ldots, Y_{0,2}, Y_{1,2}, \text{IID}$$

$$\text{mse}(\widehat{\mu}) = \frac{\text{var}(Y_1)}{n_1} + \frac{\text{var}(Y_2)}{n_2}$$

and the cost of one Y_ℓ is $\$_\ell$. How does one choose n_1 and n_2 optimally to minimize $\text{mse}(\widehat{\mu})$?

Two Level Monte Carlo

$$Y = Y_1 + Y_2,\ \mu := \mu_1 + \mu_2,\ \mu := \mathbb{E}(Y),\ \mu_1 := \mathbb{E}(Y_1),\ \mu_2 := \mathbb{E}(Y_2)$$

$$\widehat{\mu} := \widehat{\mu}_1 + \widehat{\mu}_2, \quad \widehat{\mu}_1 := \frac{1}{n_1}\sum_{i=0}^{n_1-1} Y_{i1},\ \widehat{\mu}_2 := \frac{1}{n_2}\sum_{i=0}^{n_2-1} Y_{i2}, \qquad Y_{0,1}, Y_{1,1}, \ldots, Y_{0,2}, Y_{1,2}, \text{IID}$$

$$\begin{aligned}\operatorname{mse}(\widehat{\mu}) &= \frac{\operatorname{var}(Y_1)}{n_1} + \frac{\operatorname{var}(Y_2)}{n_2} \\ &= \underbrace{n_1\$_1}_{\substack{\text{time spent}\\ \text{on } Y_1}} \underbrace{\frac{\operatorname{var}(Y_1)}{n_1^2\$_1}}_{\substack{\operatorname{mse}(\widehat{\mu}_1) \text{ per}\\ \text{time spent on } Y_1}} + \underbrace{n_2\$_2}_{\substack{\text{time spent}\\ \text{on } Y_2}} \underbrace{\frac{\operatorname{var}(Y_2)}{n_2^2\$_2}}_{\substack{\operatorname{mse}(\widehat{\mu}_2) \text{ per}\\ \text{time spent on } Y_2}}\end{aligned}$$

To optimize make $\dfrac{\operatorname{var}(Y_1)}{n_1^2\$_1} = \dfrac{\operatorname{var}(Y_2)}{n_2^2\$_2}$

Multilevel Monte Carlo

$$Y = Y_1 + \cdots + Y_L,\ \mu := \mu_1 + \cdots + \mu_L,\ \mu := \mathbb{E}(Y),\ \mu_\ell := \mathbb{E}(Y_\ell)$$

$$\widehat{\mu} := \widehat{\mu}_1 + \cdots + \widehat{\mu}_L \quad \widehat{\mu}_\ell := \frac{1}{n_\ell}\sum_{i=0}^{n_\ell - 1} Y_{i1}, \quad Y_{0,1}, Y_{1,1}, \ldots, Y_{0,2}, Y_{1,2}, \ldots, \ldots \ \text{IID}$$

$$\operatorname{mse}(\widehat{\mu}) = \frac{\operatorname{var}(Y_1)}{n_1} + \cdots + \frac{\operatorname{var}(Y_L)}{n_L}, \qquad \$_\ell = \text{cost of one } Y_\ell$$

$$= \underbrace{n_1 \$_1}_{\text{time spent on } Y_1} \underbrace{\frac{\operatorname{var}(Y_1)}{n_1^2 \$_1}}_{\operatorname{mse}(\widehat{\mu}_1) \text{ per time spent on } Y_1} + \cdots + \underbrace{n_L \$_L}_{\text{time spent on } Y_L} \underbrace{\frac{\operatorname{var}(Y_L)}{n_L^2 \$_L}}_{\operatorname{mse}(\widehat{\mu}_L) \text{ per time spent on } Y_L}$$

To optimize make $\dfrac{\operatorname{var}(Y_1)}{n_1^2 \$_1} = \cdots = \dfrac{\operatorname{var}(Y_L)}{n_L^2 \$_L}$, i.e. $n_\ell \propto \sqrt{\dfrac{\operatorname{var}(Y_\ell)}{\$_\ell}}$

Multilevel Monte Carlo

- Larger $\mathrm{var}(Y_\ell)$ implies larger n_ℓ
- Larger $\$_\ell$ implies smaller n_ℓ
- $\mathrm{var}(Y_\ell)$ and $\$_\ell$ may need to be estimated by pilot samples
- Often $\mu := \mu_1 + \cdots + \mu_L + \Delta$, where Δ is small but cannot be sampled
- For the optimal sample sizes $n_\ell = c\sqrt{\mathrm{var}(Y_\ell)/\$_\ell}$

$$\mathrm{total\,cost}(\widehat{\mu}) = c\left[\sqrt{\mathrm{var}(Y_1)\,\$_1} + \cdots + \sqrt{\mathrm{var}(Y_L)\,\$_L}\right]$$

$$\mathrm{mse}(\widehat{\mu}) = \frac{1}{\mathrm{total\,cost}(\widehat{\mu})}\left[\sqrt{\mathrm{var}(Y_1)\$_1} + \cdots + \sqrt{\mathrm{var}(Y_L)\$_L}\right]^2$$

Multilevel Monte Carlo for ∞-D Expectations

$\mu = \lim_{d\to\infty} \mathbb{E}[f_d(\boldsymbol{X}_{1:d})]$, e.g., f_d is an option payoff using d time steps

Let $Y_\ell = f_{d_\ell}(\boldsymbol{X}) - f_{d_{\ell-1}}(\boldsymbol{X}_{1:d_{\ell-1}})$, for $\boldsymbol{X} \sim \mathcal{U}[0,1]^{d_\ell}$, where $f_{d_0} = 0$. Then

$$\mu = \mathbb{E}[\underbrace{f_{d_1}(\boldsymbol{X}_{1:d_1}) - f_{d_0}}_{Y_1} + \underbrace{f_{d_2}(\boldsymbol{X}_{1:d_2}) - f_{d_1}(\boldsymbol{X}_{1:d_1})}_{Y_2} + \cdots$$

$$+ \underbrace{f_{d_L}(\boldsymbol{X}_{1:d_L}) - f_{d_{L-1}}(\boldsymbol{X}_{1:d_{L-1}})]}_{Y_L} + \lim_{d\to\infty} \mathbb{E}[f_d(\boldsymbol{X}_{1:d}) - f_{d_L}(\boldsymbol{X}_{1:d_L})]$$

$$\hat{\mu} := \hat{\mu}_1 + \cdots + \hat{\mu}_L, \quad \hat{\mu}_\ell := \frac{1}{n_\ell} \sum_{i=0}^{n_\ell - 1} Y_{i\ell}, \qquad Y_{0,1}, Y_{1,1}, \ldots, Y_{0,2}, Y_{1,2}, \text{IID}$$

Typically $\$_\ell \propto d_\ell$. If $\mathrm{var}(Y_{d_\ell})$ decays, then MLMC works well by choosing $d_1 < d_2 < \cdots < d_L$ with d_L large enough.

Multilevel Monte Carlo for ∞-D Expectations

$\mu = \lim_{d\to\infty} \mathbb{E}[f_d(\boldsymbol{X}_{1:d})]$, $Y_\ell = f_{d_\ell}(\boldsymbol{X}) - f_{d_{\ell-1}}(\boldsymbol{X}_{1:d_{\ell-1}})$, for $\boldsymbol{X} \sim \mathcal{U}[0,1]^{d_\ell}$, $f_{d_0} = 0$.

$$\widehat{\mu} := \widehat{\mu}_1 + \cdots + \widehat{\mu}_L, \quad \widehat{\mu}_\ell := \frac{1}{n_\ell}\sum_{i=0}^{n_\ell-1} Y_{i\ell}, \qquad Y_{0,1}, Y_{1,1}, \ldots, Y_{0,2}, Y_{1,2}, \text{IID}$$

If $d_\ell = m^\ell$, $\mathrm{var}(Y_\ell) \le \beta v^\ell$, and $\$_\ell \le \gamma d_\ell = \gamma m^\ell$, then

$$\begin{aligned}
\text{total cost}(\widehat{\mu}) &\le c\left[\sqrt{\beta\gamma\, vm} + \cdots + \sqrt{\beta\gamma\,(vm)^L}\right] \\
&= c\sqrt{\beta\gamma}\left[(vm)^{1/2} + \cdots + (vm)^{L/2}\right] \\
&\le \begin{cases} \frac{c\sqrt{\beta\gamma\, vm}}{1-\sqrt{vm}} \propto \text{cost of } Y_1 \text{ part}, & vm < 1 \\ \frac{c\sqrt{\beta\gamma\,(vm)^L}}{1-1/\sqrt{vm}} \propto \text{cost of } Y_L \text{ part}, & vm > 1 \end{cases}
\end{aligned}$$