The Signal to Noise problem and Stochastic Automatic Differentiation

[G.C., A. Ramos, EPJC 85 (2025) 9, 1037]

Guilherme Catumba
Alberto Ramos

Università degli Studi di Milano-Bicocca

The Signal to Noise problem

- Physical properties – Euclidean time behavior of 2-point functions

$$C(x_0) = \sum_{\vec{x}} \left\langle O^\dagger(x) O(0) \right\rangle_{\mathrm{c}} = \sum_n |\langle 0|O|n\rangle|^2 e^{-E_n x_0} \xrightarrow[x_0 \to \infty]{} |\langle 0|O|i\rangle|^2 e^{-E_i x_0}$$

  - large time behavior
  - energy levels, matrix elements
  - variance problem for large x_0

The Signal to Noise problem

- Physical properties – Euclidean time behavior of 2-point functions

$$C(x_0) = \sum_{\vec{x}} \left\langle O^\dagger(x) O(0) \right\rangle_{\mathrm{c}} = \sum_n |\langle 0|O|n\rangle|^2 e^{-E_n x_0} \xrightarrow[x_0 \to \infty]{} |\langle 0|O|i\rangle|^2 e^{-E_i x_0}$$

  - large time behavior
  - energy levels, matrix elements
  - variance problem for large x_0

- Parisi/Lepage argument for the variance [Parisi 1984; Lepage 1989]
  - Case of a scalar interpolator

$$\sigma^2(x_0) \propto \left\langle O^2(x_0) O^2(0) \right\rangle - \langle O(x_0) O(0) \rangle^2$$

$$\langle A(x_0) B(0) \rangle \xrightarrow[x_0 \to \infty]{} \langle A(x_0) \rangle \langle B(0) \rangle = \langle A(0) \rangle \langle B(0) \rangle$$

$$\sigma(x_0) \xrightarrow[x_0 \to \infty]{} C(x_0 = 0)/\sqrt{N_{\mathrm{conf}}}$$

The Signal to Noise problem

- Physical properties – Euclidean time behavior of 2-point functions

$$C(x_0) = \sum_{\vec{x}} \left\langle O^\dagger(x) O(0) \right\rangle_c = \sum_n |\langle 0|O|n\rangle|^2 e^{-E_n x_0} \xrightarrow[x_0 \to \infty]{} |\langle 0|O|i\rangle|^2 e^{-E_i x_0}$$

  - large time behavior
  - energy levels, matrix elements
  - variance problem for large x_0

- Parisi/Lepage argument for the variance [Parisi 1984; Lepage 1989]
  - Case of a scalar interpolator

$$\sigma^2(x_0) \propto \left\langle O^2(x_0) O^2(0) \right\rangle - \langle O(x_0) O(0) \rangle^2$$
$$\langle A(x_0) B(0) \rangle \xrightarrow[x_0 \to \infty]{} \langle A(x_0) \rangle \langle B(0) \rangle = \langle A(0) \rangle \langle B(0) \rangle$$
$$\sigma(x_0) \xrightarrow[x_0 \to \infty]{} C(x_0 = 0)/\sqrt{N_{\text{conf}}}$$

  - Signal to Noise

$$\frac{\text{error}[C(x_0)]}{C(x_0)} \xrightarrow[x_0 \to \infty]{} \frac{e^{E_0 x_0}}{\sqrt{N_{\text{conf}}}}$$

The Signal to Noise problem

- More generally: apply the same argument to the variance

$$\left\langle O^\dagger(x)O^\dagger(x)O(0)O(0)\right\rangle = \sum_n \left|\langle 0|O^2|n\rangle\right|^2 e^{-E_n x_0} \xrightarrow[x_0\to\infty]{} |\langle 0|O|j\rangle|^2 e^{-E_j x_0}$$

The Signal to Noise problem

- More generally: apply the same argument to the variance

$$\left\langle O^\dagger(x)O^\dagger(x)O(0)O(0)\right\rangle = \sum_n \left|\langle 0|O^2|n\rangle\right|^2 e^{-E_n x_0} \xrightarrow[x_0\to\infty]{} |\langle 0|O|j\rangle|^2 e^{-E_j x_0}$$

$$\frac{\text{error}[C(x_0)]}{C(x_0)} \xrightarrow[x_0\to\infty]{} \frac{e^{(E_i - E_j/2)x_0}}{\sqrt{N_{\text{conf}}}}\,.$$

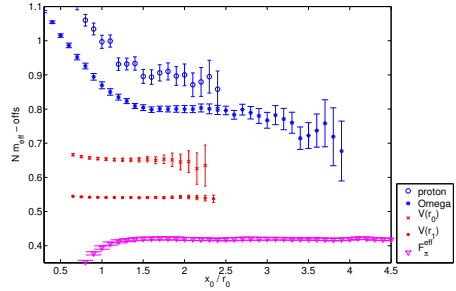
The Signal to Noise problem

- More generally: apply the same argument to the variance

$$\left\langle O^\dagger(x)O^\dagger(x)O(0)O(0)\right\rangle = \sum_n \left|\langle 0|O^2|n\rangle\right|^2 e^{-E_n x_0} \xrightarrow[x_0\to\infty]{} |\langle 0|O|j\rangle|^2 e^{-E_j x_0}$$

$$\frac{\text{error}[C(x_0)]}{C(x_0)} \xrightarrow[x_0\to\infty]{} \frac{e^{(E_i - E_j/2)x_0}}{\sqrt{N_{\text{conf}}}}\,.$$

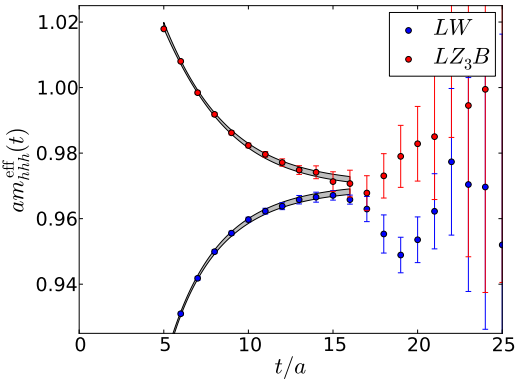
- Scalar: StN $\sim e^{E_0 x_0}$
- Pion: StN $\sim$ const.
- Proton: StN $\sim e^{(m_p - 3m_\pi)x_0}$



[Sommer 2014]

Solutions?

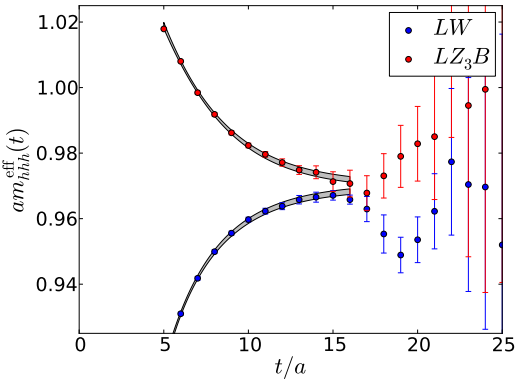
- Longstanding & ubiquitous problem in MC
- Typical solutions (smearing, GEVP, etc.)
  - Allow to extract information at earlier x_0
  - Excited state contamination
  - Do not solve the core of the StN problem



[Blum et al. 2016]

Solutions?

- Longstanding & ubiquitous problem in MC
- Typical solutions (smearing, GEVP, etc.)
  - Allow to extract information at earlier x_0
  - Excited state contamination
  - Do not solve the core of the StN problem



[Blum et al. 2016]

Source:
2-point expectation value

Derivatives & Sources – another perspective

- Do not compute a 2-point function

Derivatives & Sources – another perspective

- Do not compute a 2-point function
- Add source to the action

$$Z[J] = \int D\phi e^{-S[\phi]+JO(x_0=0)} \qquad \longrightarrow \qquad C(t) = \frac{\partial}{\partial J}\bigg|_{J=0} \langle O(x_0)\rangle$$

1-point function!

Derivatives & Sources – another perspective

- Do not compute a 2-point function
- Add source to the action

$$Z[J] = \int D\phi e^{-S[\phi]+JO(x_0=0)} \longrightarrow C(t) = \frac{\partial}{\partial J}\bigg|_{J=0} \langle O(x_0)\rangle$$

1-point function!

- 1-point expectation value – no StN

Derivatives & Sources – another perspective

- Do not compute a 2-point function
- Add source to the action

$$Z[J] = \int D\phi e^{-S[\phi]+JO(x_0=0)} \qquad \longrightarrow \qquad C(t) = \left.\frac{\partial}{\partial J}\right|_{J=0} \langle O(x_0)\rangle$$

1-point function!

- 1-point expectation value – no StN
- How to compute the derivative?
  - Variance of a derivative depends on how it is computed
  - 2-point function – evaluation of the derivative through reweighting

Derivatives & Sources – another perspective

- Do not compute a 2-point function
- Add source to the action

$$Z[J] = \int D\phi e^{-S[\phi]+JO(x_0=0)} \qquad \longrightarrow \qquad C(t) = \left.\frac{\partial}{\partial J}\right|_{J=0} \langle O(x_0)\rangle$$

1-point function!

- 1-point expectation value – no StN
- How to compute the derivative?
  - Variance of a derivative depends on how it is computed
  - 2-point function – evaluation of the derivative through reweighting
- How to compute derivatives efficiently
  - Not with finite differences [Detmold 2005]
  - Stochastic automatic differentiation [G.C., A. Ramos, B. Zaldivar, 2024]

Automatic Differentiation

Automatic Differentiation – truncated polynomials

- Many flavour of AD – we choose truncated polynomials
- Power series $\mathcal{O}(\varepsilon^K)$

$$\tilde{x} \equiv x_0 + x_1\varepsilon + x_2\varepsilon^2 + \cdots + x_K\varepsilon^K$$

$$\tilde{y} \equiv y_0 + y_1\varepsilon + y_2\varepsilon^2 + \cdots + y_K\varepsilon^K$$

Automatic Differentiation – truncated polynomials

- Many flavour of AD – we choose truncated polynomials
- Power series $\mathcal{O}(\varepsilon^K)$

$$\tilde{x} \equiv x_0 + x_1\varepsilon + x_2\varepsilon^2 + \cdots + x_K\varepsilon^K$$

$$\tilde{y} \equiv y_0 + y_1\varepsilon + y_2\varepsilon^2 + \cdots + y_K\varepsilon^K$$

- Define operations & elementary functions
  - Multiplication for $K = 2$

$$\tilde{x}\tilde{y} = x_0y_0 + (x_0y_1 + x_1y_0)\varepsilon + (x_0y_2 + 2x_1y_1 + x_2y_0)\varepsilon^2 + \ldots$$

Automatic Differentiation – truncated polynomials

- Many flavour of AD – we choose truncated polynomials
- Power series $\mathcal{O}(\varepsilon^K)$

$$\tilde{x} \equiv x_0 + x_1\varepsilon + x_2\varepsilon^2 + \cdots + x_K\varepsilon^K$$

$$\tilde{y} \equiv y_0 + y_1\varepsilon + y_2\varepsilon^2 + \cdots + y_K\varepsilon^K$$

- Define operations & elementary functions
  - Multiplication for $K = 2$

$$\tilde{x}\tilde{y} = x_0y_0 + (x_0y_1 + x_1y_0)\varepsilon + (x_0y_2 + 2x_1y_1 + x_2y_0)\varepsilon^2 + \ldots$$

  - Exponential function

$$\exp(\tilde{x}) = e^{x_0} + e^{x_0}x_1\varepsilon + e^{x_0}({x_1}^2/2 + x_2)\varepsilon^2$$

  - ...

Automatic Differentiation – truncated polynomials

- Many flavour of AD – we choose truncated polynomials
- Power series $\mathcal{O}(\varepsilon^K)$

$$\tilde{x} \equiv x_0 + x_1\varepsilon + x_2\varepsilon^2 + \cdots + x_K\varepsilon^K$$

$$\tilde{y} \equiv y_0 + y_1\varepsilon + y_2\varepsilon^2 + \cdots + y_K\varepsilon^K$$

- Define operations & elementary functions
  - Multiplication for $K = 2$

$$\tilde{x}\tilde{y} = x_0y_0 + (x_0y_1 + x_1y_0)\varepsilon + (x_0y_2 + 2x_1y_1 + x_2y_0)\varepsilon^2 + \ldots$$

  - Exponential function

$$\exp(\tilde{x}) = e^{x_0} + e^{x_0}x_1\varepsilon + e^{x_0}({x_1}^2/2 + x_2)\varepsilon^2$$

  - ...
- All operations are exact at each order

Automatic Differentiation – truncated polynomials

- Any function $f(x)$ is a composition of elementary operations and functions

Automatic Differentiation – truncated polynomials

- Any function $f(x)$ is a composition of elementary operations and functions
- 'Overload' $+,\ -, \times,\ , \ldots,\ \sin(x),\ \log,\ \ldots$

Automatic Differentiation – truncated polynomials

- Any function $f(x)$ is a composition of elementary operations and functions
- 'Overload' $+,\ -, \times,\ , \ldots,\ \sin(x),\ \log,\ \ldots$
- Evaluation of a function using $\tilde{x}$ instead of x

$$f(\tilde{x}) \longrightarrow \tilde{f}$$

Automatic Differentiation – truncated polynomials

- Any function $f(x)$ is a composition of elementary operations and functions
- 'Overload' $+,\ -, \times,\ , \ldots,\ \sin(x),\ \log,\ \ldots$
- Evaluation of a function using $\tilde{x}$ instead of x

$$f(\tilde{x}) \longrightarrow \tilde{f}$$

- How to compute derivatives?
  - Introduce $\tilde{x} = x_0 + 1\varepsilon$
  - Taylor Theorem + Chain rule

Automatic Differentiation – truncated polynomials

- Any function $f(x)$ is a composition of elementary operations and functions
- 'Overload' $+,\ -, \times,\ , \ldots,\ \sin(x),\ \log,\ \ldots$
- Evaluation of a function using $\tilde{x}$ instead of x

$$f(\tilde{x}) \longrightarrow \tilde{f}$$

- How to compute derivatives?
  - Introduce $\tilde{x} = x_0 + 1\varepsilon$
  - Taylor Theorem + Chain rule

Evaluate function at $\tilde{x} = x_0 + 1\varepsilon$

$$f(\tilde{x}) = f(x_0 + \varepsilon) = f(x_0) + f'(x_0)\varepsilon + \frac{1}{2}f''(x_0)\varepsilon^2$$

Automatic Differentiation – truncated polynomials

- Any function $f(x)$ is a composition of elementary operations and functions
- 'Overload' $+,\ -, \times,\ , \ldots,\ \sin(x),\ \log,\ \ldots$
- Evaluation of a function using $\tilde{x}$ instead of x

$$f(\tilde{x}) \longrightarrow \tilde{f}$$

- How to compute derivatives?
  - Introduce $\tilde{x} = x_0 + 1\varepsilon$
  - Taylor Theorem + Chain rule

Evaluate function at $\tilde{x} = x_0 + 1\varepsilon$

$$f(\tilde{x}) = f(x_0 + \varepsilon) = f(x_0) + f'(x_0)\varepsilon + \frac{1}{2}f''(x_0)\varepsilon^2$$

- Compute Taylor expansion of any deterministic function

Numerical Implementation

- “Hard Part”: Teach the computer to operate with truncated polynomials

Numerical Implementation

- "Hard Part": Teach the computer to operate with truncated polynomials
- Julia implementation https://igit.ific.uv.es/alramos/formalseries.jl
  - meta programming – avoid recursive relations

Numerical Implementation

- "Hard Part": Teach the computer to operate with truncated polynomials
- Julia implementation https://igit.ific.uv.es/alramos/formalseries.jl
  - meta programming – avoid recursive relations
- Practical example:

Code (Julia)

```
julia> using FormalSeries
julia> f(x) = 1/(1-x)
julia> xs = Series((0.0,1.0,0.0,0.0,0.0))
julia> f(xs)
Series{Float64, 5}((1.0, 1.0, 1.0, 1.0, 1.0))
```

Numerical Implementation

- "Hard Part": Teach the computer to operate with truncated polynomials
- Julia implementation https://igit.ific.uv.es/alramos/formalseries.jl
  - meta programming – avoid recursive relations
- Practical example:

Code (Julia)

```
julia> using FormalSeries
julia> f(x) = 1/(1-x)
julia> xs = Series((0.0,1.0,0.0,0.0,0.0))
julia> f(xs)
Series{Float64, 5}((1.0, 1.0, 1.0, 1.0, 1.0))
```

- Formally equivalent to:

$$f(0+\varepsilon) = \frac{1}{1-\varepsilon}$$
$$= 1 + 1\varepsilon + 1\varepsilon^2 + 1\varepsilon^3 + 1\varepsilon^4 + \dots$$

Numerical Implementation

- "Hard Part": Teach the computer to operate with truncated polynomials
- Julia implementation https://igit.ific.uv.es/alramos/formalseries.jl
  - meta programming – avoid recursive relations
- Practical example:

Code (Julia)

```
julia> using FormalSeries
julia> f(x) = 1/(1-x)
julia> xs = Series((0.0,1.0,0.0,0.0,0.0))
julia> f(xs)
Series{Float64, 5}((1.0, 1.0, 1.0, 1.0, 1.0))
```

Code (Julia)

```
julia> f(x) = sin(exp(x))/log(tanh(x)*cos(x))
julia> xs = Series((0.5,1.0,0.0,0.0,0.0))
julia> f(xs)
Series{Float64, 5}((-1.1046557358573723, -1.2722621217127463,
 3.4725235570733166, 6.860067727919006, -0.32678479239224445))
```

- Formally equivalent to:

$$f(0+\varepsilon) = \frac{1}{1-\varepsilon}$$
$$= 1 + 1\varepsilon + 1\varepsilon^2 + 1\varepsilon^3 + 1\varepsilon^4 + \dots$$

Numerical Implementation

- "Hard Part": Teach the computer to operate with truncated polynomials
- Julia implementation https://igit.ific.uv.es/alramos/formalseries.jl
  - meta programming – avoid recursive relations
- Practical example:

Code (Julia)

```
julia> using FormalSeries
julia> f(x) = 1/(1-x)
julia> xs = Series((0.0,1.0,0.0,0.0,0.0))
julia> f(xs)
Series{Float64, 5}((1.0, 1.0, 1.0, 1.0, 1.0))
```

Code (Julia)

```
julia> f(x) = sin(exp(x))/log(tanh(x)*cos(x))
julia> xs = Series((0.5,1.0,0.0,0.0,0.0))
julia> f(xs)
Series{Float64, 5}((-1.1046557358573723, -1.2722621217127463,
 3.4725235570733166, 6.860067727919006, -0.32678479239224445))
```

- Formally equivalent to:

$$f(0+\varepsilon) = \frac{1}{1-\varepsilon} = 1 + 1\varepsilon + 1\varepsilon^2 + 1\varepsilon^3 + 1\varepsilon^4 + \dots$$

$$f'(0.5) = 1! \times (-1.1046557358573723)$$
$$f''(0.5) = 2! \times (-1.2722621217127463)$$
$$f'''(0.5) = 3! \times (3.4725235570733166)$$
$$\dots$$

Numerical Implementation

- "Hard Part": Teach the computer to operate with truncated polynomials
- Julia implementation https://igit.ific.uv.es/alramos/formalseries.jl
  - meta programming – avoid recursive relations
- Practical example:

Code (Julia)

```
julia> using FormalSeries
julia> f(x) = 1/(1-x)
julia> xs = Series((0.0,1.0,0.0,0.0,0.0))
julia> f(xs)
Series{Float64, 5}((1.0, 1.0, 1.0, 1.0, 1.0))
```

Code (Julia)

```
julia> f(x) = sin(exp(x))/log(tanh(x)*cos(x))
julia> xs = Series((0.5,1.0,0.0,0.0,0.0))
julia> f(xs)
Series{Float64, 5}((-1.1046557358573723, -1.2722621217127463,
 3.4725235570733166, 6.860067727919006, -0.32678479239224445))
```

- Formally equivalent to:

$$f(0+\varepsilon) = \frac{1}{1-\varepsilon} = 1 + 1\varepsilon + 1\varepsilon^2 + 1\varepsilon^3 + 1\varepsilon^4 + \dots$$

$$f'(0.5) = 1! \times (-1.1046557358573723)$$
$$f''(0.5) = 2! \times (-1.2722621217127463)$$
$$f'''(0.5) = 3! \times (3.4725235570733166)$$
$$\dots$$

- This can be done for any existing program

Numerical Implementation

- ε is just a formal 'symbolic' expansion parameter

Numerical Implementation

- ε is just a formal 'symbolic' expansion parameter
- Multiple expansions $\varepsilon \longrightarrow \varepsilon_i$
  - Multiple derivatives of $f(x, y)$

$$f(x_0+\varepsilon_1, y_0+\varepsilon_2) = f(x_0, y_0) + \varepsilon_1 \left.\frac{\partial f}{\partial x}\right|_{x_0,y_0} + \varepsilon_2 \left.\frac{\partial f}{\partial y}\right|_{x_0,y_0} + \varepsilon_1\varepsilon_2 \left.\frac{\partial^2 f}{\partial x \partial y}\right|_{x_0,y_0} + \dots$$

  - (available in the code)

Numerical Implementation

- ε is just a formal 'symbolic' expansion parameter
- Multiple expansions $\varepsilon \longrightarrow \varepsilon_i$
  - Multiple derivatives of $f(x, y)$

$$f(x_0+\varepsilon_1, y_0+\varepsilon_2) = f(x_0, y_0) + \varepsilon_1 \left.\frac{\partial f}{\partial x}\right|_{x_0,y_0} + \varepsilon_2 \left.\frac{\partial f}{\partial y}\right|_{x_0,y_0} + \varepsilon_1\varepsilon_2 \left.\frac{\partial^2 f}{\partial x \partial y}\right|_{x_0,y_0} + \dots$$

  - (available in the code)
- Key message: differentiate any deterministic computer program w.r.t. input parameters
  - Fitting, Integration, ...
  - Error propagation and data analysis [ADerrors - A. Ramos 2018]
  - Ideal for systematic error estimates in, e.g., machine learning algorithms

[G.C., A. Ramos, B. Zaldivar, 2024]

Numerical Implementation

- ε is just a formal 'symbolic' expansion parameter
- Multiple expansions $\varepsilon \longrightarrow \varepsilon_i$
  - Multiple derivatives of $f(x, y)$

$$f(x_0+\varepsilon_1, y_0+\varepsilon_2) = f(x_0, y_0) + \varepsilon_1 \left.\frac{\partial f}{\partial x}\right|_{x_0,y_0} + \varepsilon_2 \left.\frac{\partial f}{\partial y}\right|_{x_0,y_0} + \varepsilon_1\varepsilon_2 \left.\frac{\partial^2 f}{\partial x \partial y}\right|_{x_0,y_0} + \dots$$

  - (available in the code)
- Key message: differentiate any deterministic computer program w.r.t. input parameters
  - Fitting, Integration, ...
  - Error propagation and data analysis [ADerrors - A. Ramos 2018]
  - Ideal for systematic error estimates in, e.g., machine learning algorithms [G.C., A. Ramos, B. Zaldivar, 2024]
- Basis of Forward Automatic Differentiation

Metropolis & Truncated Polynomials

- Why not just use this directly with lattice simulations?

Metropolis & Truncated Polynomials

- Why not just use this directly with lattice simulations?
- Metropolis as basis for sampling algorithms

Metropolis & Truncated Polynomials

- Why not just use this directly with lattice simulations?
- Metropolis as basis for sampling algorithms
  - Action $S(\theta)$: θ denotes some couplings

Metropolis & Truncated Polynomials

- Why not just use this directly with lattice simulations?
- Metropolis as basis for sampling algorithms
  - Action $S(\theta)$: θ denotes some couplings
  - Sampling proposal (e.g., HMC)

$$S(\theta) \longrightarrow S'(\theta)$$

Metropolis & Truncated Polynomials

- Why not just use this directly with lattice simulations?
- Metropolis as basis for sampling algorithms
  - Action $S(\theta)$: θ denotes some couplings
  - Sampling proposal (e.g., HMC)

$$S(\theta) \longrightarrow S'(\theta)$$

  - Accept using

$$\min\{1, e^{-\Delta S(\tilde{\theta})}\}$$

Metropolis & Truncated Polynomials

- Why not just use this directly with lattice simulations?
- Metropolis as basis for sampling algorithms
  - Action $S(\theta)$: θ denotes some couplings
  - Sampling proposal (e.g., HMC)

$$S(\theta) \longrightarrow S'(\theta)$$

  - Accept using

$$\min\{1, e^{-\Delta S(\tilde{\theta})}\}$$

  - Not possible if $\tilde{\theta}$ is a series element

Metropolis & Truncated Polynomials

- Why not just use this directly with lattice simulations?
- Metropolis as basis for sampling algorithms
  - Action $S(\theta)$: θ denotes some couplings
  - Sampling proposal (e.g., HMC)

$$S(\theta) \longrightarrow S'(\theta)$$

  - Accept using

$$\min\{1, e^{-\Delta S(\tilde{\theta})}\}$$

  - Not possible if $\tilde{\theta}$ is a series element

Automatic Differentiation ruled out due to Stochastic elements

How to overcome this?

Extending Automatic Differentiation for Monte Carlo processes

Reweighting & Automatic Differentiation

[G.C., A. Ramos, B. Zaldivar, 2024]

Samples

$$\{\phi^\alpha\}_{\alpha=1}^N \sim e^{-S(\phi;\theta)}$$

Expectation values w.r.t

$$e^{-S(\phi;\theta')}$$

Reweighting & Automatic Differentiation

[G.C., A. Ramos, B. Zaldivar, 2024]

Samples

$$\{\phi^\alpha\}_{\alpha=1}^N \sim e^{-S(\phi;\theta)}$$

Expectation values w.r.t

$$e^{-S(\phi;\theta')}$$

- Conventional Reweighting

$$\langle f(\phi)\rangle_{S'} = \frac{\left\langle e^{S-S'} f(\phi)\right\rangle_S}{\langle e^{S-S'}\rangle_S}$$

- Usually noisy (disc. contributions)
- Solve this later!

Reweighting & Automatic Differentiation

[G.C., A. Ramos, B. Zaldivar, 2024]

Samples

$$\{\phi^\alpha\}_{\alpha=1}^N \sim e^{-S(\phi;\theta)}$$

Expectation values w.r.t

$$e^{-S(\phi;\theta')}$$

- Conventional Reweighting

$$\langle f(\phi)\rangle_{S'} = \frac{\left\langle e^{S-S'} f(\phi)\right\rangle_S}{\langle e^{S-S'}\rangle_S}$$

  - Usually noisy (disc. contributions)
  - Solve this later!

- Introduce truncated polynomials with $\tilde{\theta} = \theta + \varepsilon$

$$w^\alpha = e^{S(\phi,\theta)-S(\phi,\tilde{\theta})} = 1 + (\ \dots\ )\varepsilon + (\ \dots\ )\varepsilon^2 + \dots$$

Reweighting & Automatic Differentiation

[G.C., A. Ramos, B. Zaldivar, 2024]

Samples

$$\{\phi^\alpha\}_{\alpha=1}^N \sim e^{-S(\phi;\theta)}$$

Expectation values w.r.t

$$e^{-S(\phi;\theta')}$$

- Conventional Reweighting

$$\langle f(\phi)\rangle_{S'} = \frac{\left\langle e^{S-S'} f(\phi)\right\rangle_S}{\langle e^{S-S'}\rangle_S}$$

  - Usually noisy (disc. contributions)
  - Solve this later!

- Introduce truncated polynomials with $\tilde{\theta} = \theta + \varepsilon$

$$w^\alpha = e^{S(\phi,\theta)-S(\phi,\tilde{\theta})} = 1 + (\ \dots\ )\varepsilon + (\ \dots\ )\varepsilon^2 + \dots$$

- Expectation value with Taylor series coefficients

$$\frac{\sum_\alpha e^{S(\phi^\alpha,\theta)-S(\phi^\alpha,\tilde{\theta})} f(\phi)}{\sum_\alpha e^{S(\phi^\alpha,\theta)-S(\phi^\alpha,\tilde{\theta})}} = \sum_{n=0}^{K} f_n \varepsilon^n, \qquad f_n = \frac{1}{n!}\frac{\partial^n}{\partial\theta^n}\,\langle f(\phi)\rangle\Big|_\theta$$

Conventional 2-point functions & Reweighting

Samples

$\{\phi^\alpha\}_{\alpha=1}^N \sim e^{-S(\phi;\theta)}$

Expectation values w.r.t

$e^{-S(\phi;\theta)-JO(0)}$

Conventional 2-point functions & Reweighting

Samples	Expectation values w.r.t
$\{\phi^\alpha\}_{\alpha=1}^N \sim e^{-S(\phi;\theta)}$	$e^{-S(\phi;\theta)-JO(0)}$

- Expansion in $J = 0 + \varepsilon$

$$w^\alpha = e^{S(\phi,\theta)-S(\phi,\tilde{\theta})+JO(0)} = 1 + \varepsilon O(0) + \varepsilon^2 O(0)^2/2 + \dots$$

Conventional 2-point functions & Reweighting

Samples	Expectation values w.r.t
$\{\phi^\alpha\}_{\alpha=1}^N \sim e^{-S(\phi;\theta)}$	$e^{-S(\phi;\theta)-JO(0)}$

- Expansion in $J = 0 + \varepsilon$

$$w^\alpha = e^{S(\phi,\theta)-S(\phi,\tilde{\theta})+JO(0)} = 1 + \varepsilon O(0) + \varepsilon^2 O(0)^2/2 + \dots$$

- Reweighting to compute d/dJ

$$\langle O(x_0)\rangle_{S'} = \frac{\langle w^\alpha O(x_0)\rangle_S}{\langle w^\alpha\rangle_S} = \langle O(x_0)\rangle + \varepsilon\langle O(x_0)O(0)\rangle_c + \mathcal{O}(\varepsilon^2)$$

  - Conventional computation of a 2-point function can be seen as reweighting

Conventional 2-point functions & Reweighting

Samples	Expectation values w.r.t
$\{\phi^\alpha\}_{\alpha=1}^N \sim e^{-S(\phi;\theta)}$	$e^{-S(\phi;\theta)-JO(0)}$

- Expansion in $J = 0 + \varepsilon$

$$w^\alpha = e^{S(\phi,\theta)-S(\phi,\tilde{\theta})+JO(0)} = 1 + \varepsilon O(0) + \varepsilon^2 O(0)^2/2 + \dots$$

- Reweighting to compute d/dJ

$$\langle O(x_0)\rangle_{S'} = \frac{\langle w^\alpha O(x_0)\rangle_S}{\langle w^\alpha\rangle_S} = \langle O(x_0)\rangle + \varepsilon\langle O(x_0)O(0)\rangle_c + \mathcal{O}(\varepsilon^2)$$

  - Conventional computation of a 2-point function can be seen as reweighting
  - StN + usual reweighting variance issue

Conventional 2-point functions & Reweighting

Samples

$$\{\phi^\alpha\}_{\alpha=1}^N \sim e^{-S(\phi;\theta)}$$

Expectation values w.r.t

$$e^{-S(\phi;\theta)-JO(0)}$$

- Expansion in $J = 0 + \varepsilon$

$$w^\alpha = e^{S(\phi,\theta)-S(\phi,\tilde{\theta})+JO(0)} = 1 + \varepsilon O(0) + \varepsilon^2 O(0)^2/2 + \dots$$

- Reweighting to compute d/dJ

$$\langle O(x_0)\rangle_{S'} = \frac{\langle w^\alpha O(x_0)\rangle_S}{\langle w^\alpha\rangle_S} = \langle O(x_0)\rangle + \varepsilon\langle O(x_0)O(0)\rangle_c + \mathcal{O}(\varepsilon^2)$$

  - Conventional computation of a 2-point function can be seen as reweighting
  - StN + usual reweighting variance issue
  - Ignore (for now)

HMC & Automatic Differentiation (NSPT-inspired method)

1. Fictitious momenta π conjugate to ϕ

$$H(\phi, \pi) = \frac{1}{2}\pi^2 + S(\phi; \theta)$$

2. Solve EoM with initial random momenta $\pi(t = 0) \sim N(0, 1)$

$$\dot{\phi} = \frac{\partial H}{\partial \pi} = \pi, \qquad \dot{\pi} = -\frac{\partial H}{\partial \phi}$$

$$(\phi(0), \pi(0)) \longrightarrow (\phi(t), \pi(t))$$

3. Metropolis: Acc./Rej. with probability $e^{-\Delta H}$
4. Repeat

HMC & Automatic Differentiation (NSPT-inspired method)

1. Fictitious momenta π conjugate to ϕ

$$H(\phi, \pi) = \frac{1}{2}\pi^2 + S(\phi; \theta)$$

2. Solve EoM with initial random momenta $\pi(t = 0) \sim N(0, 1)$

$$\dot{\phi} = \frac{\partial H}{\partial \pi} = \pi, \qquad \dot{\pi} = -\frac{\partial H}{\partial \phi}$$

$$(\phi(0), \pi(0)) \longrightarrow (\phi(t), \pi(t))$$

3. Metropolis: Acc./Rej. with probability $e^{-\Delta H}$
4. Repeat

Promote $\theta \longrightarrow \tilde{\theta} = \theta + \varepsilon$
(also π, ϕ)

HMC & Automatic Differentiation (NSPT-inspired method)

1. Fictitious momenta π conjugate to ϕ

$$H(\phi, \pi) = \frac{1}{2}\pi^2 + S(\phi; \theta)$$

2. Solve EoM with initial random momenta $\pi(t = 0) \sim N(0, 1)$

$$\dot{\phi} = \frac{\partial H}{\partial \pi} = \pi, \qquad \dot{\pi} = -\frac{\partial H}{\partial \phi}$$

$$(\phi(0), \pi(0)) \longrightarrow (\phi(t), \pi(t))$$

3. Metropolis: Acc./Rej. with probability $e^{-\Delta H}$
4. Repeat

Promote $\theta \longrightarrow \tilde{\theta} = \theta + \varepsilon$
(also π, ϕ)

Solve EoM at each order

$\phi \longrightarrow \tilde{\phi}$

HMC & Automatic Differentiation (NSPT-inspired method)

1. Fictitious momenta π conjugate to ϕ

$$H(\phi, \pi) = \frac{1}{2}\pi^2 + S(\phi; \theta)$$

2. Solve EoM with initial random momenta $\pi(t = 0) \sim N(0, 1)$

$$\dot{\phi} = \frac{\partial H}{\partial \pi} = \pi, \qquad \dot{\pi} = -\frac{\partial H}{\partial \phi}$$

$$(\phi(0), \pi(0)) \longrightarrow (\phi(t), \pi(t))$$

3. ~~Metropolis: Acc./Rej. with probability $e^{-\Delta H}$~~
4. Repeat

Promote $\theta \longrightarrow \tilde{\theta} = \theta + \varepsilon$
(also π, ϕ)

Solve EoM at each order

$\phi \longrightarrow \tilde{\phi}$

ΔH as Taylor series
Cannot Acc./Rej.

HMC & Automatic Differentiation (NSPT-inspired method)

1. Fictitious momenta π conjugate to ϕ

$$H(\phi, \pi) = \frac{1}{2}\pi^2 + S(\phi; \theta)$$

2. Solve EoM with initial random momenta $\pi(t = 0) \sim N(0, 1)$

$$\dot{\phi} = \frac{\partial H}{\partial \pi} = \pi, \qquad \dot{\pi} = -\frac{\partial H}{\partial \phi}$$

$$(\phi(0), \pi(0)) \longrightarrow (\phi(t), \pi(t))$$

3. ~~Metropolis: Acc./Rej. with probability $e^{-\Delta H}$~~
4. Repeat

Promote $\theta \longrightarrow \tilde{\theta} = \theta + \varepsilon$
(also π, ϕ)

Solve EoM at each order

$\phi \longrightarrow \tilde{\phi}$

ΔH as Taylor series
Cannot Acc./Rej.

MC samples are truncated polynomials
$\tilde{\phi}^0, \tilde{\phi}^1, \ldots, \tilde{\phi}^N$

HMC & Automatic Differentiation

- HMC samples are truncated polynomials

$$\tilde{\phi} = \phi + \varepsilon\phi_{(1)}$$
$$\{\tilde{\phi}^{\alpha}\}_{\alpha=1}^{N_{\text{conf.}}} = \{\phi^{\alpha} + \varepsilon\phi_{(1)}^{\alpha}\}_{\alpha=1}^{N_{\text{conf.}}} \sim e^{-S(\phi,\tilde{\theta})}$$

  - Samples carry information about the dependence on θ

HMC & Automatic Differentiation

- HMC samples are truncated polynomials

$$\tilde{\phi} = \phi + \varepsilon\phi_{(1)}$$
$$\{\tilde{\phi}^{\alpha}\}_{\alpha=1}^{N_{\text{conf.}}} = \{\phi^{\alpha} + \varepsilon\phi_{(1)}^{\alpha}\}_{\alpha=1}^{N_{\text{conf.}}} \sim e^{-S(\phi,\tilde{\theta})}$$

  - Samples carry information about the dependence on θ

- Usual MC averages

$$\left\langle \tilde{O}(\phi) \right\rangle = \frac{1}{N_{\text{conf}}} \sum_{\alpha} O(\tilde{\phi}^{\alpha})$$
$$= \langle O(\phi) \rangle + \frac{\mathrm{d}}{\mathrm{d}\theta} \langle O(\phi) \rangle \varepsilon$$

  - 1-point function with no RW factors (disc. contributions)

HMC & Automatic Differentiation

- HMC samples are truncated polynomials

$$\tilde{\phi} = \phi + \varepsilon\phi_{(1)}$$
$$\{\tilde{\phi}^\alpha\}_{\alpha=1}^{N_{\text{conf.}}} = \{\phi^\alpha + \varepsilon\phi_{(1)}^\alpha\}_{\alpha=1}^{N_{\text{conf.}}} \sim e^{-S(\phi,\tilde{\theta})}$$

  - Samples carry information about the dependence on θ

- Usual MC averages

$$\left\langle \tilde{O}(\phi) \right\rangle = \frac{1}{N_{\text{conf}}} \sum_\alpha O(\tilde{\phi}^\alpha)$$
$$= \langle O(\phi) \rangle + \frac{\mathrm{d}}{\mathrm{d}\theta} \langle O(\phi) \rangle \varepsilon$$

  - 1-point function with no RW factors (disc. contributions)

- Valid for similar sampling algorithm (Langevin, SMD, etc.)

Comparison & Change of variables

Reweighting

- $\{\phi^\alpha\}_{\alpha=1}^N \sim e^{-S(\phi;\theta)}$
- Weights $w^\alpha = e^{S(\phi^\alpha;\theta)-S(\phi^\alpha;\theta')}$ take into account dependence on parameters θ

HMC

- $\{\tilde{\phi}^\alpha\}_{\alpha=1}^N \sim e^{-S(\tilde{\phi};\tilde{\theta})}$
- Samples carry dependence on the parameters $\tilde{\theta}$

Comparison & Change of variables

Reweighting

- $\{\phi^\alpha\}_{\alpha=1}^N \sim e^{-S(\phi;\theta)}$
- Weights $w^\alpha = e^{S(\phi^\alpha;\theta)-S(\phi^\alpha;\theta')}$ take into account dependence on parameters θ

HMC

- $\{\tilde{\phi}^\alpha\}_{\alpha=1}^N \sim e^{-S(\tilde{\phi};\tilde{\theta})}$
- Samples carry dependence on the parameters $\tilde{\theta}$

How are these methods related?

Comparison & Change of variables

Reweighting

- $\{\phi^\alpha\}_{\alpha=1}^N \sim e^{-S(\phi;\theta)}$
- Weights $w^\alpha = e^{S(\phi^\alpha;\theta)-S(\phi^\alpha;\theta')}$ take into account dependence on parameters θ

HMC

- $\{\tilde{\phi}^\alpha\}_{\alpha=1}^N \sim e^{-S(\tilde{\phi};\tilde{\theta})}$
- Samples carry dependence on the parameters $\tilde{\theta}$

How are these methods related?

- By definition the HMC samples the probability distribution $\exp(-H(\phi,\pi))$ (ignoring integration errors)
- Trivially the samples $\{\tilde{\phi}^\alpha\}_{\alpha=1}^N$ have constant RW factors

Comparison & Change of variables

Reweighting

- $\{\phi^\alpha\}_{\alpha=1}^N \sim e^{-S(\phi;\theta)}$
- Weights $w^\alpha = e^{S(\phi^\alpha;\theta)-S(\phi^\alpha;\theta')}$ take into account dependence on parameters θ

HMC

- $\{\tilde{\phi}^\alpha\}_{\alpha=1}^N \sim e^{-S(\tilde{\phi};\tilde{\theta})}$
- Samples carry dependence on the parameters $\tilde{\theta}$

How are these methods related?

- By definition the HMC samples the probability distribution $\exp(-H(\phi,\pi))$ (ignoring integration errors)
- Trivially the samples $\{\tilde{\phi}^\alpha\}_{\alpha=1}^N$ have constant RW factors
- Hamiltonian AD finds the exact transformation that leads to constant RW factors [G.C., A. Ramos, B. Zaldivar, 2024; G.C. 2025 (Thesis)]

Field transformations & Reweighting

- Reweighting generally leads to large variance
  - weights w^{α}

Field transformations & Reweighting

- Reweighting generally leads to large variance
  - weights w^{α}
- Consider a change of variables before reweighting

$$\phi \longrightarrow \tilde{\phi} = \phi + \varepsilon f(\phi)$$

Field transformations & Reweighting

- Reweighting generally leads to large variance
  - weights w^α
- Consider a change of variables before reweighting

$$\phi \longrightarrow \tilde{\phi} = \phi + \varepsilon f(\phi)$$

- Reweighting factors become

$$\tilde{w}[\tilde{\phi}] = \exp\left[-\tilde{S}_J[\tilde{\phi}] + \log\left|\frac{\mathrm{d}\tilde{\phi}}{\mathrm{d}\phi}\right| + S[\phi]\right]$$

Field transformations & Reweighting

- Reweighting generally leads to large variance
  - weights w^α
- Consider a change of variables before reweighting

$$\phi \longrightarrow \tilde{\phi} = \phi + \varepsilon f(\phi)$$

- Reweighting factors become

$$\tilde{w}[\tilde{\phi}] = \exp\left[-\tilde{S}_J[\tilde{\phi}] + \log\left|\frac{\mathrm{d}\tilde{\phi}}{\mathrm{d}\phi}\right| + S[\phi]\right]$$

- Find transformation that eliminates RW factors
  - Exact – RW factors drop from the computation!

$$-\tilde{S}_J[\tilde{\phi}] + \log\left|\frac{\mathrm{d}\tilde{\phi}}{\mathrm{d}\phi}\right| + S[\phi] = \text{const.} \qquad \frac{\langle w^\alpha O(x_0)\rangle_S}{\langle w^\alpha\rangle_S} \longrightarrow \langle O(x_0)\rangle_{S'}$$

Field transformations & Reweighting

- Reweighting generally leads to large variance
  - weights w^α
- Consider a change of variables before reweighting

$$\phi \longrightarrow \tilde{\phi} = \phi + \varepsilon f(\phi)$$

- Reweighting factors become

$$\tilde{w}[\tilde{\phi}] = \exp\left[-\tilde{S}_J[\tilde{\phi}] + \log\left|\frac{\mathrm{d}\tilde{\phi}}{\mathrm{d}\phi}\right| + S[\phi]\right]$$

- Find transformation that eliminates RW factors
  - Exact – RW factors drop from the computation!

$$-\tilde{S}_J[\tilde{\phi}] + \log\left|\frac{\mathrm{d}\tilde{\phi}}{\mathrm{d}\phi}\right| + S[\phi] = \text{const.} \qquad \frac{\langle w^\alpha O(x_0)\rangle_S}{\langle w^\alpha\rangle_S} \longrightarrow \langle O(x_0)\rangle_{S'}$$

  - Approximate – partially eliminate RW factors

4D Scalar field theory

$$S^{\text{4D}}_{\text{latt}}(\phi; m, \lambda) = \sum_x \left\{ \frac{1}{2} \sum_\mu [\phi(x+\mu) - \phi(x)]^2 + \frac{m^2}{2}\phi^2(x) + \lambda\phi^4(x) \right\}$$

Operator: $O(x_0) = \sum_{\vec{x}} \phi(x_0, \vec{x})$

- Good testbed – No excited state contamination

4D Scalar field theory

$$S^{\text{4D}}_{\text{latt}}(\phi; m, \lambda) = \sum_x \left\{ \frac{1}{2} \sum_\mu [\phi(x+\mu) - \phi(x)]^2 + \frac{m^2}{2}\phi^2(x) + \lambda\phi^4(x) \right\}$$

Operator: $O(x_0) = \sum_{\vec{x}} \phi(x_0, \vec{x})$

- Good testbed – No excited state contamination

- HAD (HMC + AD):

$$S(\tilde{\phi}) = S_{\text{latt}}(\tilde{\phi}) + JO(0)$$
$$J = 0 + \varepsilon$$

4D Scalar field theory

$$S^{\text{4D}}_{\text{latt}}(\phi; m, \lambda) = \sum_x \left\{ \frac{1}{2} \sum_\mu [\phi(x+\mu) - \phi(x)]^2 + \frac{m^2}{2}\phi^2(x) + \lambda\phi^4(x) \right\}$$

Operator: $O(x_0) = \sum_{\vec{x}} \phi(x_0, \vec{x})$

- Good testbed – No excited state contamination

- HAD (HMC + AD):

$$S(\tilde{\phi}) = S_{\text{latt}}(\tilde{\phi}) + JO(0)$$
$$J = 0 + \varepsilon$$

$$\dot{\tilde{\phi}}(x) = \tilde{\pi}(x)\,,$$
$$\dot{\tilde{\pi}}(x) = \frac{1}{2}\sum_\mu \left[\tilde{\phi}(x+\mu) + \tilde{\phi}(x-\mu)\right] - (4+\hat{m}^2)\tilde{\phi}(x) - 4\lambda\tilde{\phi}^3(x) + \epsilon\delta_{x_0,0}$$

HAD solution for the Signal to Noise problem

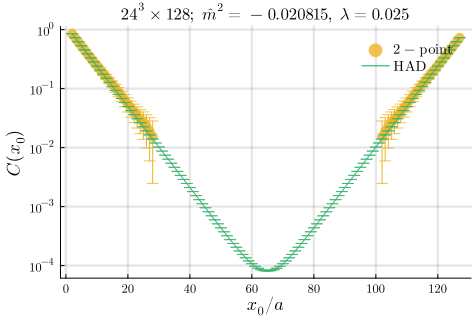
$$C(x_0)^{2-\text{point}} = \langle O(x_0) O(0) \rangle$$

$$C(x_0)^{\text{HAD}} = \left\langle \tilde{O}(x_0) \right\rangle_{(1)}$$

HAD solution for the Signal to Noise problem

$$C(x_0)^{2-\text{point}} = \langle O(x_0) O(0) \rangle$$

$$C(x_0)^{\text{HAD}} = \left\langle \tilde{O}(x_0) \right\rangle_{(1)}$$

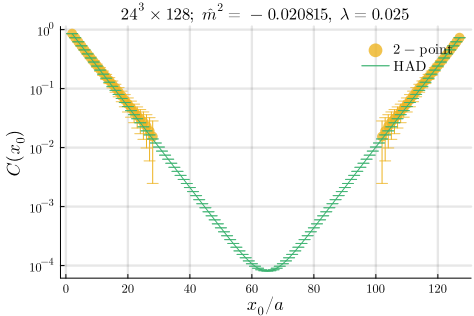


HAD solution for the Signal to Noise problem

$$C(x_0)^{2-\text{point}} = \langle O(x_0) O(0) \rangle$$

$$C(x_0)^{\text{HAD}} = \left\langle \tilde{O}(x_0) \right\rangle_{(1)}$$

$$\hat{m}_{\text{eff}}(x_0) = -\log \frac{C(x_0 + a)}{C(x_0)} \,.$$

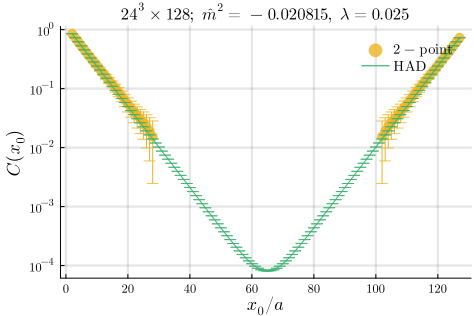


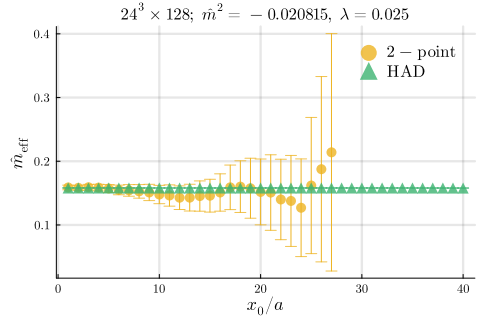
HAD solution for the Signal to Noise problem

$$C(x_0)^{\text{2-point}} = \langle O(x_0) O(0) \rangle$$

$$C(x_0)^{\text{HAD}} = \left\langle \tilde{O}(x_0) \right\rangle_{(1)}$$

$$\hat{m}_{\text{eff}}(x_0) = -\log \frac{C(x_0 + a)}{C(x_0)} \,.$$



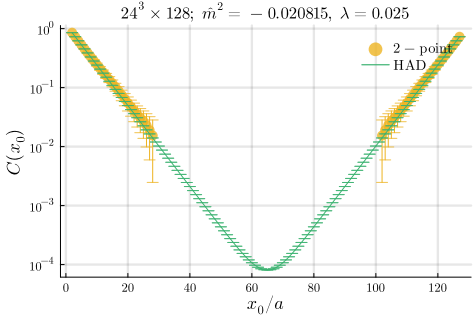


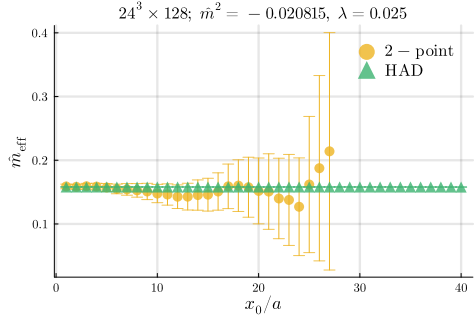
HAD solution for the Signal to Noise problem

$$C(x_0)^{\text{2-point}} = \langle O(x_0) O(0) \rangle$$

$$C(x_0)^{\text{HAD}} = \left\langle \tilde{O}(x_0) \right\rangle_{(1)}$$

$$\hat{m}_{\text{eff}}(x_0) = -\log \frac{C(x_0 + a)}{C(x_0)} \,.$$





- HAD solves StN problem completely
  - convergence
  - compact variables, complex interpolators

Transformed Reweighting (TRW)

- Transformation?

$$S_{\text{latt}}(\phi_p; m) = \sum_p \phi_p^* \left[\sum_\mu \hat{p}_\mu^2 + m^2 \right] \phi_p + \mathcal{O}(\lambda), \qquad \hat{p} = 2\sin(ap/2)$$

Transformed Reweighting (TRW)

- Transformation?

$$S_{\text{latt}}(\phi_p; m) = \sum_p \phi_p^* \left[\sum_\mu \hat{p}_\mu^2 + m^2 \right] \phi_p + \mathcal{O}(\lambda), \qquad \hat{p} = 2\sin(ap/2)$$

- Free Theory case – exact transformation

$$\sum_{\vec{x}} \phi(0, \vec{x}), \qquad \phi_p \longrightarrow \tilde{\phi}_p = \phi_p + \frac{1}{2} \frac{\delta(\vec{p})}{\hat{p}^2 + m^2_{\text{transf}}} \varepsilon$$

  - $\hat{m}_{\text{transf.}} = \hat{m}$
  - Transforms $S^{\text{free}}(\phi)$ into $\tilde{S}^{\text{free}}_{\tilde{J}}(\tilde{\phi})$
  - Constant reweighting factors

Transformed Reweighting (TRW)

- Transformation?

$$S_{\text{latt}}(\phi_p; m) = \sum_p \phi_p^* \left[\sum_\mu \hat{p}_\mu^2 + m^2 \right] \phi_p + \mathcal{O}(\lambda), \qquad \hat{p} = 2\sin(ap/2)$$

- Free Theory case – exact transformation

$$\sum_{\vec{x}} \phi(0, \vec{x}), \qquad \phi_p \longrightarrow \tilde{\phi}_p = \phi_p + \frac{1}{2} \frac{\delta(\vec{p})}{\hat{p}^2 + m^2_{\text{transf}}} \varepsilon$$

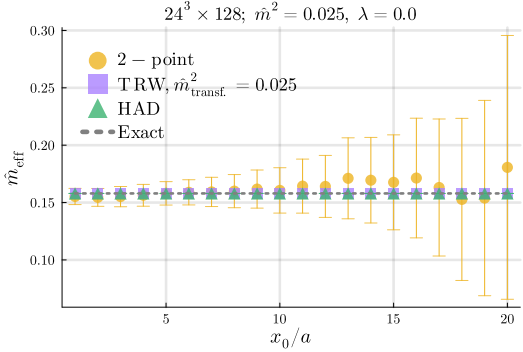
  - $\hat{m}_{\text{transf.}} = \hat{m}$
  - Transforms $S^{\text{free}}(\phi)$ into $\tilde{S}^{\text{free}}_{\tilde{J}}(\tilde{\phi})$
  - Constant reweighting factors
- $\lambda \neq 0$: use the same (approximate) transformation
  - $\hat{m}_{\text{transf.}} = \hat{m}_R$

Exact & approximate transformations

- Conventional samples tuned to have $(am_R)^2 \sim 0.025$

Exact & approximate transformations

- Conventional samples tuned to have $(am_R)^2 \sim 0.025$

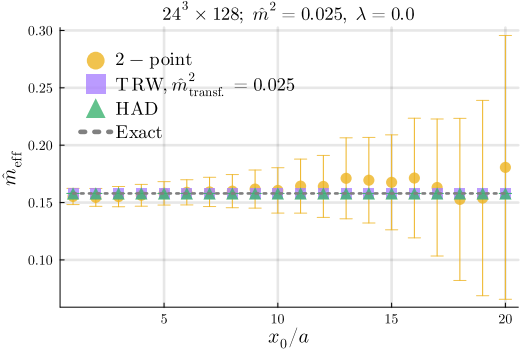


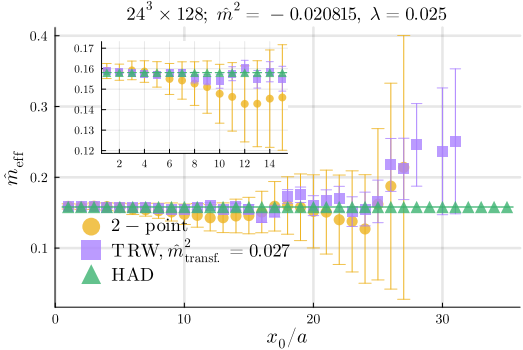
- Exact for $\lambda = 0$
- HAD = TRW
  - Confirms reparametrization argument

Exact & approximate transformations

- Conventional samples tuned to have $(am_R)^2 \sim 0.025$

$$\phi_p \to \tilde{\phi}_p = \phi_p + \frac{1}{2}\frac{\delta(\vec{p})}{\hat{p}^2 + m^2_{\text{transf}}}\varepsilon$$



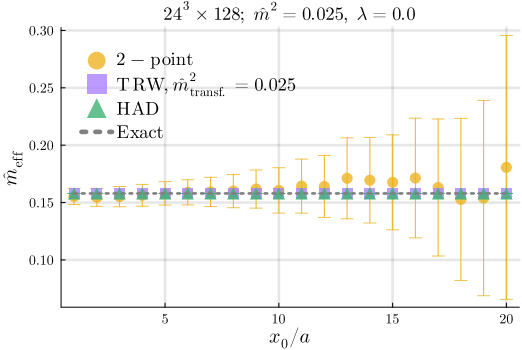


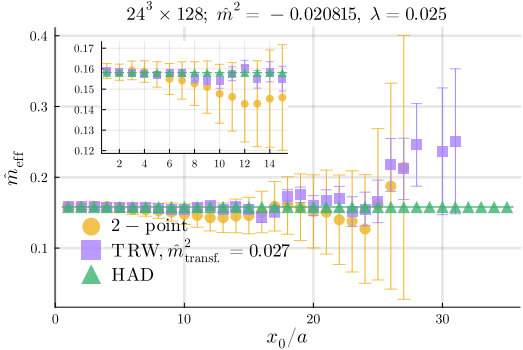
- Exact for $\lambda = 0$
- HAD = TRW
  - Confirms reparametrization argument

Exact & approximate transformations

- Conventional samples tuned to have $(am_R)^2 \sim 0.025$

$$\phi_p \to \tilde{\phi}_p = \phi_p + \frac{1}{2}\frac{\delta(\vec{p})}{\hat{p}^2 + m^2_{\text{transf}}}\varepsilon$$





- Exact for $\lambda = 0$
- HAD = TRW
  - Confirms reparametrization argument

- Approximate for $\lambda \neq 0$
- Still provides an improvement
  - But how do we choose $\hat{m}_{\text{transf.}}$?

Interacting case – transformation dependence

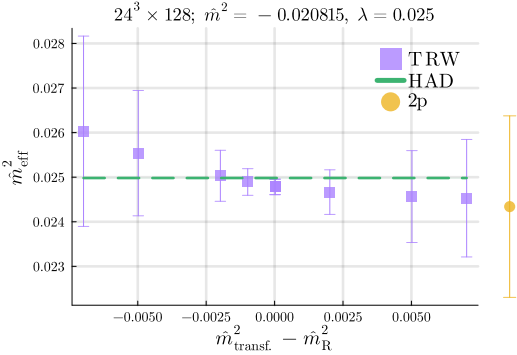
- Scan over $\hat{m}_{\text{trans.}}$ (cheap)

$$\phi_p \to \tilde{\phi}_p = \phi_p + \frac{1}{2}\frac{\delta(\vec{p})}{\hat{p}^2 + m^2_{\text{transf}}}\varepsilon$$

Interacting case – transformation dependence

- Scan over $\hat{m}_{\text{trans.}}$ (cheap)

$$\phi_p \to \tilde{\phi}_p = \phi_p + \frac{1}{2}\frac{\delta(\vec{p})}{\hat{p}^2 + m^2_{\text{transf}}}\varepsilon$$

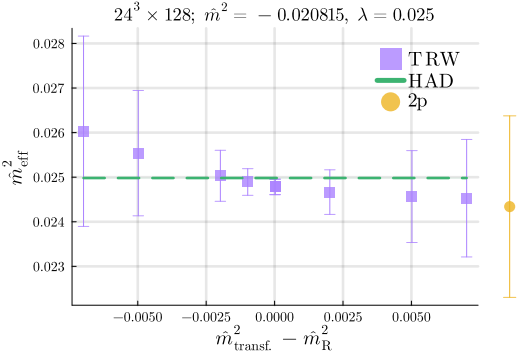


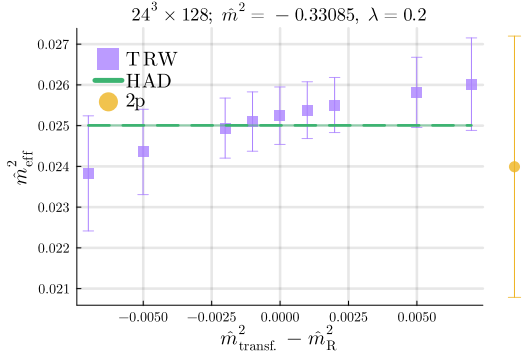
- Optimal $\hat{m}_{\text{trans.}} = \hat{m}_R$

Interacting case – transformation dependence

- Scan over $\hat{m}_{\text{trans.}}$ (cheap)

$$\phi_p \to \tilde{\phi}_p = \phi_p + \frac{1}{2}\frac{\delta(\vec{p})}{\hat{p}^2 + m^2_{\text{transf}}}\varepsilon$$

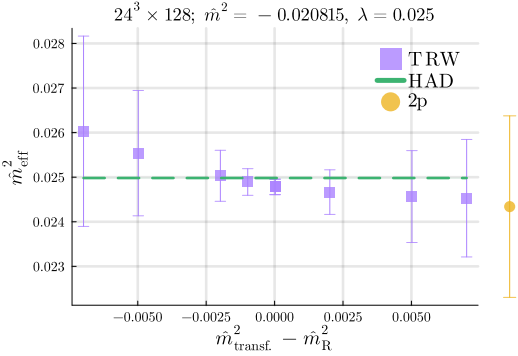




- Optimal $\hat{m}_{\text{trans.}} = \hat{m}_R$
- Nonetheless, large range of $\hat{m}_{\text{trans.}}$

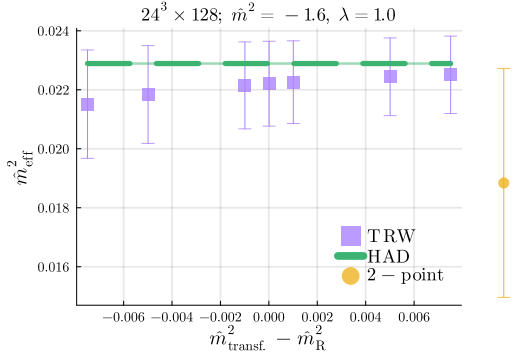
Interacting case – transformation dependence

- Scan over $\hat{m}_{\text{trans.}}$ (cheap)



- Optimal $\hat{m}_{\text{trans.}} = \hat{m}_R$
- Nonetheless, large range of $\hat{m}_{\text{trans.}}$

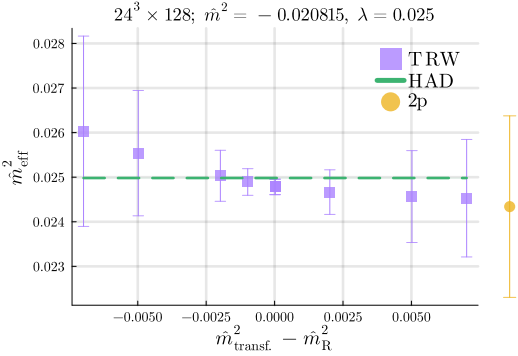
$$\phi_p \to \tilde{\phi}_p = \phi_p + \frac{1}{2}\frac{\delta(\vec{p})}{\hat{p}^2 + m^2_{\text{transf}}}\varepsilon$$



- Precision degrades with λ

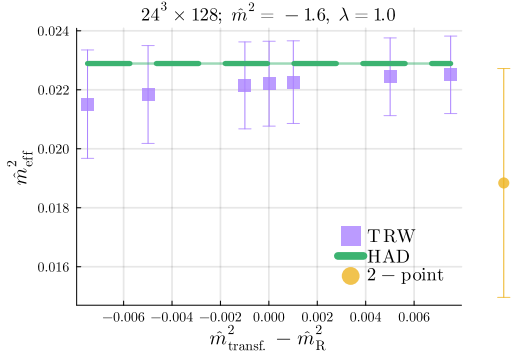
Interacting case – transformation dependence

- Scan over $\hat{m}_{\text{trans.}}$ (cheap)



- Optimal $\hat{m}_{\text{trans.}} = \hat{m}_R$
- Nonetheless, large range of $\hat{m}_{\text{trans.}}$

$$\phi_p \to \tilde{\phi}_p = \phi_p + \frac{1}{2}\frac{\delta(\vec{p})}{\hat{p}^2 + m^2_{\text{transf}}}\varepsilon$$



- Precision degrades with λ
- $\hat{m}_{\text{trans.}} \neq \hat{m}_R$ still provides an improvement w.r.t. 2-point

Does TRW solve the Signal to Noise problem?

- Approximate TRW: $JO(0)$ partially removed

$$-\tilde{S}_J[\tilde{\phi}] + \log\left|\frac{\mathrm{d}\tilde{\phi}}{\mathrm{d}\phi}\right| + S[\phi] \sim \lambda O(0)$$

Does TRW solve the Signal to Noise problem?

- Approximate TRW: $JO(0)$ partially removed

$$-\tilde{S}_J[\tilde{\phi}] + \log\left|\frac{\mathrm{d}\tilde{\phi}}{\mathrm{d}\phi}\right| + S[\phi] \sim \lambda O(0)$$

$$\langle O(x_0)\rangle_{S'} \sim \frac{\left\langle e^{\lambda O(0)} O(x_0)\right\rangle_S}{\langle w^\alpha\rangle_S}$$

Does TRW solve the Signal to Noise problem?

- Approximate TRW: $JO(0)$ partially removed

$$-\tilde{S}_J[\tilde{\phi}] + \log\left|\frac{\mathrm{d}\tilde{\phi}}{\mathrm{d}\phi}\right| + S[\phi] \sim \lambda O(0)$$

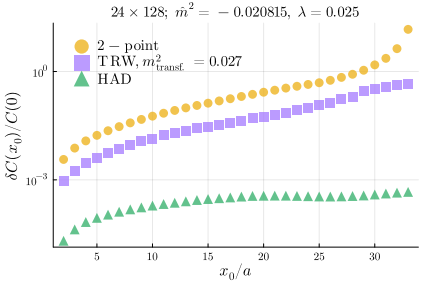
$$\langle O(x_0)\rangle_{S'} \sim \frac{\left\langle e^{\lambda O(0)} O(x_0)\right\rangle_S}{\langle w^\alpha\rangle_S} = \frac{\text{2-point function}}{\text{RW factor w/ reduced variance}}$$

Does TRW solve the Signal to Noise problem?

- Approximate TRW: $JO(0)$ partially removed

$$-\tilde{S}_J[\tilde{\phi}] + \log\left|\frac{\mathrm{d}\tilde{\phi}}{\mathrm{d}\phi}\right| + S[\phi] \sim \lambda O(0)$$

$$\langle O(x_0)\rangle_{S'} \sim \frac{\langle e^{\lambda O(0)} O(x_0)\rangle_S}{\langle w^\alpha\rangle_S} = \frac{\text{2-point function}}{\text{RW factor w/ reduced variance}}$$

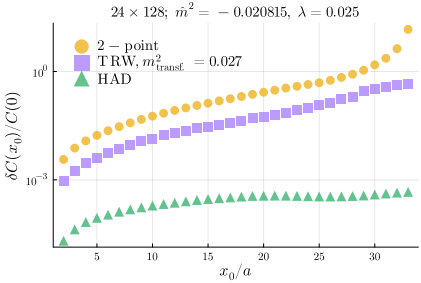


Does TRW solve the Signal to Noise problem?

- Approximate TRW: $JO(0)$ partially removed

$$-\tilde{S}_J[\tilde{\phi}] + \log\left|\frac{\mathrm{d}\tilde{\phi}}{\mathrm{d}\phi}\right| + S[\phi] \sim \lambda O(0)$$

$$\langle O(x_0)\rangle_{S'} \sim \frac{\langle e^{\lambda O(0)} O(x_0)\rangle_S}{\langle w^\alpha\rangle_S} = \frac{\text{2-point function}}{\text{RW factor w/ reduced variance}}$$



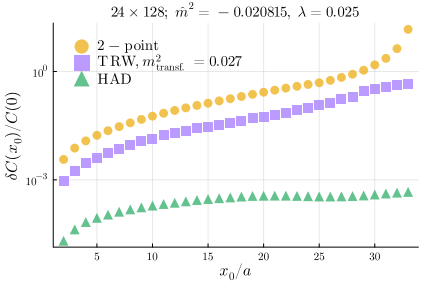
- Approx. TRW still includes a 2-point function (StN)
- Compare 2PF w/ TRW

Does TRW solve the Signal to Noise problem?

- Approximate TRW: $JO(0)$ partially removed

$$-\tilde{S}_J[\tilde{\phi}] + \log\left|\frac{\mathrm{d}\tilde{\phi}}{\mathrm{d}\phi}\right| + S[\phi] \sim \lambda O(0)$$

$$\langle O(x_0)\rangle_{S'} \sim \frac{\langle e^{\lambda O(0)} O(x_0)\rangle_S}{\langle w^\alpha\rangle_S} = \frac{\text{2-point function}}{\text{RW factor w/ reduced variance}}$$



- Approx. TRW still includes a 2-point function (StN)
- Compare 2PF w/ TRW
  - Same exponential scaling
  - Reduced starting variance (same statistics)

Conclusions

- StN is an important problem for lattice field theory
- A change of perspective of the StN is possible:
  - The usual 2-point computation is a reweighting computation of a derivative
  - StN + RW leads to worse precision
  - Correlator as derivative of a 1PF – variance depends on how it is computed
- Key ingredient: Stochastic AD
  - Computes derivatives exactly
  - Single simulation, no need for finite-differences
- Hamiltonian method solves the StN exactly
  - not generally applicable (convergence, complex interpolators, etc.)
- Field transformations can reduce (or eliminate) the variance – solves the StN problem
  - Re-utilize samples – TRW is cheap
  - General (no problems with convergence, etc)

Future ideas

- Machine Learning to find changes of variables
  - P. Butti, G. C., A. Nada, L. Spatscheck
    Normalizing flows – attack the StN and not the sampling
    Parametrize transformation $f(\phi)$ as a norm. flow
    Reweighting factors $\longrightarrow$ KL divergence
  - Extensions for gauge theory
- Scalar-gauge theories
  - Scalar masses known to be noisy
  - Propagators are gauge-invariant (no gauge fixing needed)
  - Study the triviality problem

Numerical differentiation

Finite Differences

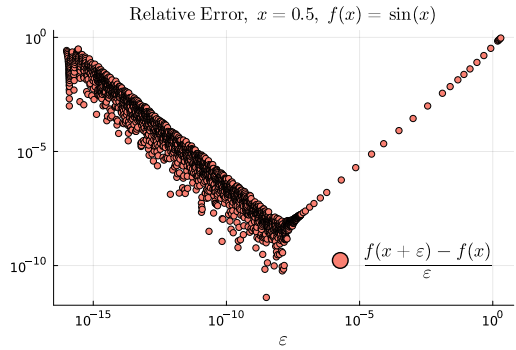
$$f'(x) = \lim_{\varepsilon \to 0} \frac{f(x+\varepsilon) - f(x)}{\varepsilon}$$

Numerical differentiation

Finite Differences

$$f'(x) = \lim_{\varepsilon \to 0} \frac{f(x+\varepsilon) - f(x)}{\varepsilon}$$

- Ill conditioned definition:
  - ε too large $\longrightarrow$ higher order contamination
  - ε too small $\longrightarrow$ computer precision, rounding errors

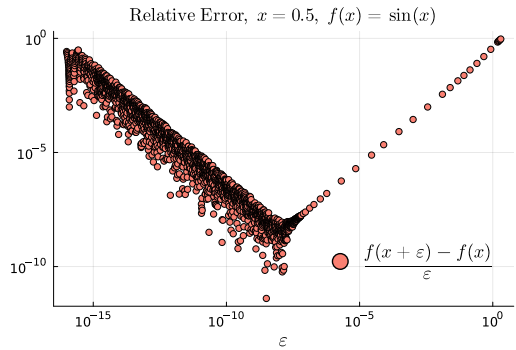


Numerical differentiation

Finite Differences

$$f'(x) = \lim_{\varepsilon \to 0} \frac{f(x+\varepsilon) - f(x)}{\varepsilon}$$

- Ill conditioned definition:
  - ε too large $\longrightarrow$ higher order contamination
  - ε too small $\longrightarrow$ computer precision, rounding errors
- Poor control over systematic error

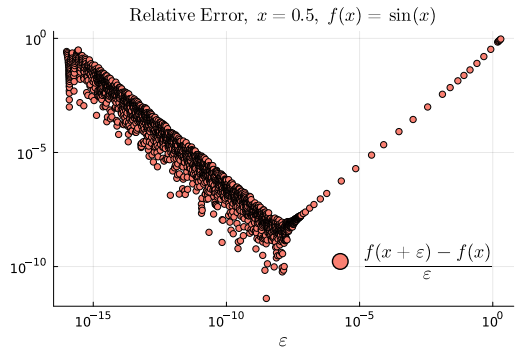


Numerical differentiation

Finite Differences

$$f'(x) = \lim_{\varepsilon \to 0} \frac{f(x+\varepsilon) - f(x)}{\varepsilon}$$

- Ill conditioned definition:
  - ε too large $\longrightarrow$ higher order contamination
  - ε too small $\longrightarrow$ computer precision, rounding errors
- Poor control over systematic error
- Requires multiple simulations
  - $J,\ J + \varepsilon$



Comparison & Change of variables

Reweighting

- $\{\phi^\alpha\}_{\alpha=1}^N \sim e^{-S(\phi;\theta)}$
- Weights $w^\alpha = e^{S(\phi^\alpha;\theta)-S(\phi^\alpha;\theta')}$ take into account dependence on parameters θ

HMC

- $\{\tilde{\phi}^\alpha\}_{\alpha=1}^N \sim e^{-S(\tilde{\phi};\tilde{\theta})}$
- Samples carry dependence on the parameters $\tilde{\theta}$

How are these methods related?

Comparison & Change of variables

Reweighting

- $\{\phi^\alpha\}_{\alpha=1}^N \sim e^{-S(\phi;\theta)}$
- Weights $w^\alpha = e^{S(\phi^\alpha;\theta)-S(\phi^\alpha;\theta')}$ take into account dependence on parameters θ

HMC

- $\{\tilde{\phi}^\alpha\}_{\alpha=1}^N \sim e^{-S(\tilde{\phi};\tilde{\theta})}$
- Samples carry dependence on the parameters $\tilde{\theta}$

How are these methods related?

Toy model: $p_\sigma(\phi) = \dfrac{1}{\sigma\sqrt{2\pi}} e^{-\frac{\phi^2}{2\sigma^2}}$, $\{\phi^\alpha\} \sim p_{\sigma^*=1}$,

Comparison & Change of variables

Reweighting

- $\{\phi^\alpha\}_{\alpha=1}^N \sim e^{-S(\phi;\theta)}$
- Weights $w^\alpha = e^{S(\phi^\alpha;\theta)-S(\phi^\alpha;\theta')}$ take into account dependence on parameters θ

HMC

- $\{\tilde{\phi}^\alpha\}_{\alpha=1}^N \sim e^{-S(\tilde{\phi};\tilde{\theta})}$
- Samples carry dependence on the parameters $\tilde{\theta}$

How are these methods related?

Toy model: $p_\sigma(\phi) = \dfrac{1}{\sigma\sqrt{2\pi}} e^{-\frac{\phi^2}{2\sigma^2}}$, $\{\phi^\alpha\} \sim p_{\sigma^*=1}$,

$\sigma^* = 1 \longrightarrow \sigma$ $w^\alpha(\phi) = \phi -$ dependent, i.e. sample dependent

Comparison & Change of variables

Reweighting

- $\{\phi^\alpha\}_{\alpha=1}^N \sim e^{-S(\phi;\theta)}$
- Weights $w^\alpha = e^{S(\phi^\alpha;\theta)-S(\phi^\alpha;\theta')}$ take into account dependence on parameters θ

HMC

- $\{\tilde{\phi}^\alpha\}_{\alpha=1}^N \sim e^{-S(\tilde{\phi};\tilde{\theta})}$
- Samples carry dependence on the parameters $\tilde{\theta}$

How are these methods related?

Toy model: $p_\sigma(\phi) = \dfrac{1}{\sigma\sqrt{2\pi}} e^{-\frac{\phi^2}{2\sigma^2}}$, $\{\phi^\alpha\} \sim p_{\sigma^*=1}$,

$\sigma^* = 1 \longrightarrow \sigma$ $w^\alpha(\phi) = \phi$ – dependent, i.e. sample dependent

Transformation: $y^\alpha = \sigma\phi^\alpha$, $\{y^\alpha\} \sim p_\sigma$

Comparison & Change of variables

Reweighting

- $\{\phi^\alpha\}_{\alpha=1}^N \sim e^{-S(\phi;\theta)}$
- Weights $w^\alpha = e^{S(\phi^\alpha;\theta)-S(\phi^\alpha;\theta')}$ take into account dependence on parameters θ

HMC

- $\{\tilde{\phi}^\alpha\}_{\alpha=1}^N \sim e^{-S(\tilde{\phi};\tilde{\theta})}$
- Samples carry dependence on the parameters $\tilde{\theta}$

How are these methods related?

Toy model: $p_\sigma(\phi) = \dfrac{1}{\sigma\sqrt{2\pi}} e^{-\frac{\phi^2}{2\sigma^2}}$, $\{\phi^\alpha\} \sim p_{\sigma^*=1}$,

$\sigma^* = 1 \longrightarrow \sigma$ $w^\alpha(\phi) = \phi$ – dependent, i.e. sample dependent

Transformation: $y^\alpha = \sigma\phi^\alpha$, $\{y^\alpha\} \sim p_\sigma$

$\sigma^* = 1 \longrightarrow \sigma$ $w^\alpha(y) = y$ – independent ‘No Reweighting’

Comparison & Change of variables

$$y^{\alpha} = \sigma \phi^{\alpha}$$

$$\varepsilon = \sigma - 1$$

Comparison & Change of variables

$$y^\alpha = \sigma\phi^\alpha \qquad \tilde{y}^\alpha = \phi^\alpha + \phi^\alpha\varepsilon \qquad \varepsilon = \sigma - 1$$

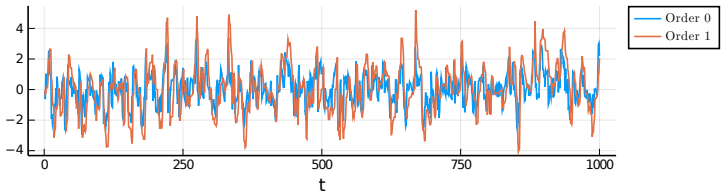
Comparison & Change of variables

$$y^\alpha = \sigma\phi^\alpha \qquad \tilde{y}^\alpha = \phi^\alpha + \phi^\alpha\varepsilon \qquad \varepsilon = \sigma - 1$$

What about HMC?
Equations of motion

$$\ddot{\phi}_0 = -\frac{\phi_0}{\sigma^2}, \qquad \langle\phi_0\rangle$$

$$\ddot{\phi}_1 = -\frac{\phi_1}{\sigma^2} + 2\frac{\phi_0}{\sigma^3}, \qquad \langle\phi_1\rangle = \langle\phi_0\rangle$$



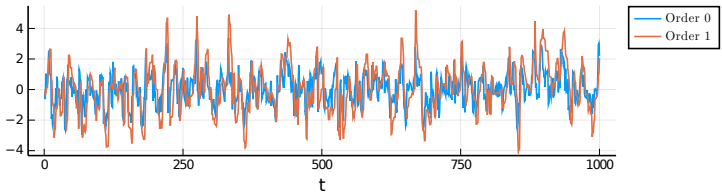
Comparison & Change of variables

$$y^\alpha = \sigma\phi^\alpha \qquad \tilde{y}^\alpha = \phi^\alpha + \phi^\alpha\varepsilon \qquad \varepsilon = \sigma - 1$$

What about HMC?
Equations of motion

$$\ddot{\phi}_0 = -\frac{\phi_0}{\sigma^2}, \qquad \langle\phi_0\rangle$$

$$\ddot{\phi}_1 = -\frac{\phi_1}{\sigma^2} + 2\frac{\phi_0}{\sigma^3}, \qquad \langle\phi_1\rangle = \langle\phi_0\rangle$$



- Hamiltonian method finds the change of variables $\phi \to \tilde{y}$ that lead to constant reweighting factors

Does TRW solve the Signal to Noise problem?

- Key idea: cancel $JO(0)$ in the RW factor

$$-\tilde{S}_J[\tilde{\phi}] + \log\left|\frac{\mathrm{d}\tilde{\phi}}{\mathrm{d}\phi}\right| + S[\phi] = \text{const.}$$

Does TRW solve the Signal to Noise problem?

- Key idea: cancel $JO(0)$ in the RW factor

$$-\tilde{S}_J[\tilde{\phi}] + \log\left|\frac{\mathrm{d}\tilde{\phi}}{\mathrm{d}\phi}\right| + S[\phi] = \text{const.}$$

- Approximate TRW: $JO(0)$ partially removed

$$-\tilde{S}_J[\tilde{\phi}] + \log\left|\frac{\mathrm{d}\tilde{\phi}}{\mathrm{d}\phi}\right| + S[\phi] \sim \lambda O(0)$$

Does TRW solve the Signal to Noise problem?

- Key idea: cancel $JO(0)$ in the RW factor

$$-\tilde{S}_J[\tilde{\phi}] + \log\left|\frac{\mathrm{d}\tilde{\phi}}{\mathrm{d}\phi}\right| + S[\phi] = \text{const.}$$

- Approximate TRW: $JO(0)$ partially removed

$$-\tilde{S}_J[\tilde{\phi}] + \log\left|\frac{\mathrm{d}\tilde{\phi}}{\mathrm{d}\phi}\right| + S[\phi] \sim \lambda O(0)$$

$$\langle O(x_0)\rangle_{S'} \sim \frac{\langle e^{\lambda O(0)} O(x_0)\rangle_S}{\langle w^\alpha\rangle_S}$$

Does TRW solve the Signal to Noise problem?

- Key idea: cancel $JO(0)$ in the RW factor

$$-\tilde{S}_J[\tilde{\phi}] + \log\left|\frac{\mathrm{d}\tilde{\phi}}{\mathrm{d}\phi}\right| + S[\phi] = \text{const.}$$

- Approximate TRW: $JO(0)$ partially removed

$$-\tilde{S}_J[\tilde{\phi}] + \log\left|\frac{\mathrm{d}\tilde{\phi}}{\mathrm{d}\phi}\right| + S[\phi] \sim \lambda O(0)$$

$$\langle O(x_0)\rangle_{S'} \sim \frac{\left\langle e^{\lambda O(0)} O(x_0)\right\rangle_S}{\langle w^\alpha\rangle_S} = \frac{\text{2-point function}}{\text{RW factor w/ reduced variance}}$$

Does TRW solve the Signal to Noise problem?

- Key idea: cancel $JO(0)$ in the RW factor

$$-\tilde{S}_J[\tilde{\phi}] + \log\left|\frac{\mathrm{d}\tilde{\phi}}{\mathrm{d}\phi}\right| + S[\phi] = \text{const.}$$

- Approximate TRW: $JO(0)$ partially removed

$$-\tilde{S}_J[\tilde{\phi}] + \log\left|\frac{\mathrm{d}\tilde{\phi}}{\mathrm{d}\phi}\right| + S[\phi] \sim \lambda O(0)$$

$$\langle O(x_0)\rangle_{S'} \sim \frac{\left\langle e^{\lambda O(0)} O(x_0)\right\rangle_S}{\langle w^\alpha\rangle_S} = \frac{\text{2-point function}}{\text{RW factor w/ reduced variance}}$$

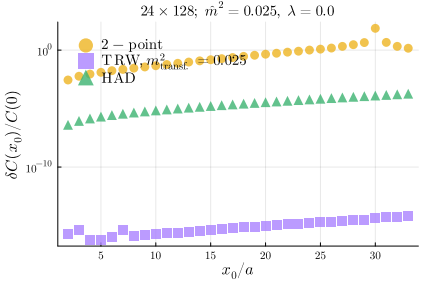
  - Approx. TRW still includes a 2-point function (StN)
  - Reduced starting variance

Does TRW solve the Signal to Noise problem?

- Compute StN explicitly: $\delta C(x_0)/C(0)$

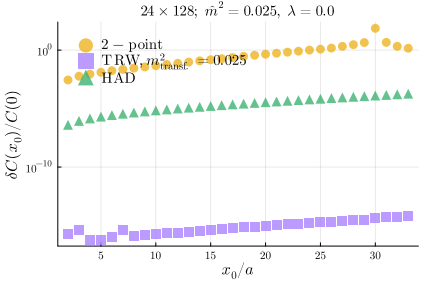
Does TRW solve the Signal to Noise problem?

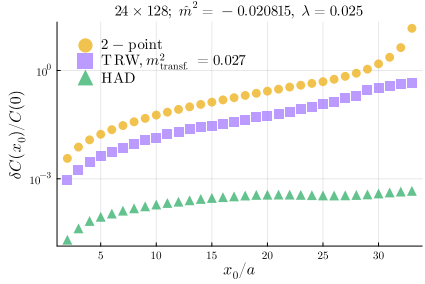
- Compute StN explicitly: $\delta C(x_0)/C(0)$



Does TRW solve the Signal to Noise problem?

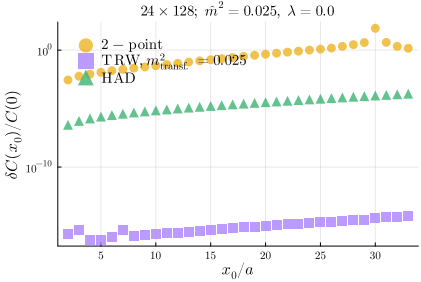
- Compute StN explicitly: $\delta C(x_0)/C(0)$

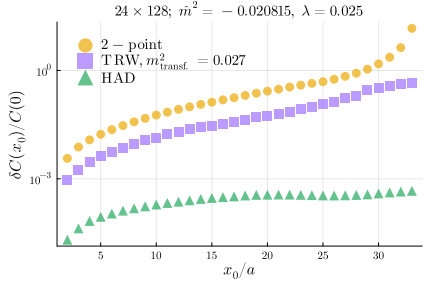




Does TRW solve the Signal to Noise problem?

- Compute StN explicitly: $\delta C(x_0)/C(0)$

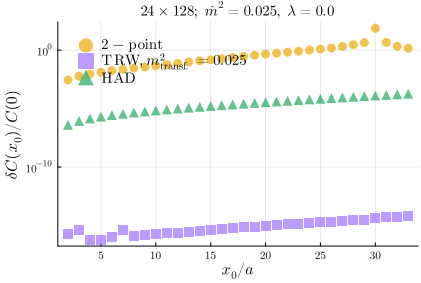


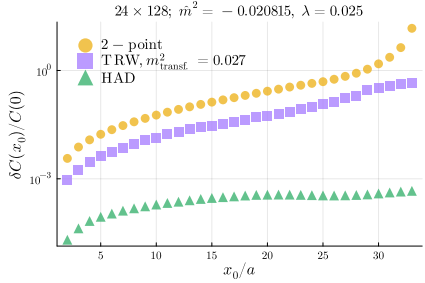


- Rough idea: look at x_0/a-dependence
- Missing error bars (error of the error!)

Does TRW solve the Signal to Noise problem?

- Compute StN explicitly: $\delta C(x_0)/C(0)$





- Rough idea: look at x_0/a-dependence
- Missing error bars (error of the error!)
- Approx. TRW VS 2-point computation
  - Same exponential scaling
  - Improved starting variance for the same statistics