High Performance Computing and Aspects of Computing in Lattice Gauge Theories

//

Giannis Koutsou
Computation-Based Science and Technology Research Centre (CaSToRC)
The Cyprus Institute

//

The Cyprus Institute
RESEARCH•TECHNOLOGY•INNOVATION

EuroPLEx School
Lecture 02 – 4th November 2020

Refresh – computational intensity

Recall the definitions of kernel and machine intensities

- Kernel: floating-point rate, I/O rate, kernel intensity I_k

$$\beta_{FP} = \frac{N_{FP}}{\bar{T}} \qquad \beta_{IO} = \frac{N_{IO}}{\bar{T}} \qquad I_k = \frac{N_{FP}}{N_{IO}}$$

Refresh – computational intensity

Recall the definitions of kernel and machine intensities

- Kernel: floating-point rate, I/O rate, kernel intensity I_k

$$\beta_{FP} = \frac{N_{FP}}{\bar{T}} \qquad \beta_{IO} = \frac{N_{IO}}{\bar{T}} \qquad I_k = \frac{N_{FP}}{N_{IO}}$$

- Machine: peak floating-point rate (γ_{FP}), I/O rate (γ_{IO}), and machine intensity I_m

$$I_m = \frac{\gamma_{FP}}{\gamma_{IO}}$$

Refresh – computational intensity

Recall the definitions of kernel and machine intensities

- Kernel: floating-point rate, I/O rate, kernel intensity $\mathrm{I_k}$

$$\beta_{\mathrm{FP}} = \frac{\mathrm{N_{FP}}}{\bar{\mathrm{T}}} \qquad \beta_{\mathrm{IO}} = \frac{\mathrm{N_{IO}}}{\bar{\mathrm{T}}} \qquad \mathrm{I_k} = \frac{\mathrm{N_{FP}}}{\mathrm{N_{IO}}}$$

- Machine: peak floating-point rate (γ_{FP}), I/O rate (γ_{IO}), and machine intensity $\mathrm{I_m}$

$$\mathrm{I_m} = \frac{\gamma_{\mathrm{FP}}}{\gamma_{\mathrm{IO}}}$$

- Compute-bound, bandwidth- (or memory-) bound, balanced
  - $\mathrm{I_k} \gg \mathrm{I_m}$: Compute bound
  - $\mathrm{I_k} \ll \mathrm{I_m}$: Memory bound
  - $\mathrm{I_k} \simeq \mathrm{I_m}$: Balanced

Practical

Matrix-matrix multiplication

- Consider the multiplication:

$$C_{M\times K} = A_{M\times N} B_{N\times K} \rightarrow C_{mk} = \sum_{n=0}^{N-1} A_{mn} B_{nk}$$

- A straight- forward implementation

```
for(int m=0; m<M; m++) {
  for(int k=0; k<K; k++) {
    C[m][k] = 0;
    for(int n=0; n<N; n++) {
      C[m][k] += A[m][n]*B[n][k];
    }
  }
}
```

Practical

Matrix-matrix multiplication

- Consider the multiplication:

$$C_{M\times K} = A_{M\times N} B_{N\times K} \rightarrow C_{mk} = \sum_{n=0}^{N-1} A_{mn} B_{nk}$$

- A straight- forward implementation

```
for(int m=0; m<M; m++) {
  for(int k=0; k<K; k++) {
    C[m][k] = 0;
    for(int n=0; n<N; n++) {
      C[m][k] += A[m][n]*B[n][k];
    }
  }
}
```

$$N_{FP} = 2NKM$$
$$N_{IO} = w(MN + NK + MK)$$

$$I_k = \frac{2}{w} \frac{1}{\frac{1}{N} + \frac{1}{K} + \frac{1}{M}}$$

Practical

Matrix-matrix multiplication

- Consider the multiplication:

$$C_{M\times K} = A_{M\times N} B_{N\times K} \rightarrow C_{mk} = \sum_{n=0}^{N-1} A_{mn} B_{nk}$$

- A straight- forward implementation

```
for(int m=0; m<M; m++) {
  for(int k=0; k<K; k++) {
    C[m][k] = 0;
    for(int n=0; n<N; n++) {
      C[m][k] += A[m][n]*B[n][k];
    }
  }
}
```

$$N_{FP} = 2NKM$$
$$N_{IO} = w(MN + NK + MK)$$

$$I_k = \frac{2}{w} \frac{1}{\frac{1}{N} + \frac{1}{K} + \frac{1}{M}}$$

Kernel intensity grows with dimensions

Practical

Matrix-matrix multiplication

- Consider the multiplication:

$$C_{M\times K} = A_{M\times N} B_{N\times K} \rightarrow C_{mk} = \sum_{n=0}^{N-1} A_{mn} B_{nk}$$

- A straight- forward implementation

```
for(int m=0; m<M; m++) {
  for(int k=0; k<K; k++) {
    C[m][k] = 0;
    for(int n=0; n<N; n++) {
      C[m][k] += A[m][n]*B[n][k];
    }
  }
}
```

$$N_{FP} = 2NKM$$
$$N_{IO} = w(MN + NK + MK)$$

$$I_k = \frac{2}{w}\frac{1}{\frac{1}{N}+\frac{1}{K}+\frac{1}{M}}$$

Kernel intensity grows with dimensions

- For square matrices: M=N=K

$$I_k = \frac{2}{3}\frac{M}{w}$$

Practical

Matrix-matrix multiplication

- Consider the multiplication:

$$C_{M\times K} = A_{M\times N} B_{N\times K} \rightarrow C_{mk} = \sum_{n=0}^{N-1} A_{mn} B_{nk}$$

- A straight- forward implementation

```
for(int m=0; m<M; m++) {
  for(int k=0; k<K; k++) {
    C[m][k] = 0;
    for(int n=0; n<N; n++) {
      C[m][k] += A[m][n]*B[n][k];
    }
  }
}
```

$$N_{FP} = 2NKM$$
$$N_{IO} = w(MN + NK + MK)$$

$$I_k = \frac{2}{w}\frac{1}{\frac{1}{N}+\frac{1}{K}+\frac{1}{M}}$$

Kernel intensity grows with dimensions

- For square matrices: $M{=}N{=}K$

$$I_k = \frac{2}{3}\frac{M}{w}$$

E.g. on Juwels, for double-precision (w=8 bytes), this kernel is memory-bound for $M \lesssim 200$, and compute-bound for $M \gtrsim 200$

Practical

Complex “axpy” operation

- $y_i = ax_i + y_i$, $i=0,\ldots,N-1$, with x, y, complex arrays, a complex scalar

```
_Complex double x[N];
_Complex double y[N];
_Complex double a;
for(int i=0; i<N; i++) {
  y[i] = a*x[i] + y[i];
}
```

Practical

Complex “axpy” operation

- $y_i = ax_i + y_i$, $i=0,\ldots,N-1$, with x, y, complex arrays, a complex scalar

```
_Complex double x[N];
_Complex double y[N];
_Complex double a;
for(int i=0; i<N; i++) {
  y[i] = a*x[i] + y[i];
}
```

- Complex variables are two-component structures
- Write real and imaginary part of left-hand-side explicitly

Practical

Complex "axpy" operation

- $y_i = ax_i + y_i$, $i=0,\ldots,N-1$, with x, y, complex arrays, a complex scalar

```
_Complex double x[N];
_Complex double y[N];
_Complex double a;
for(int i=0; i<N; i++) {
  y[i] = a*x[i] + y[i];
}
```

- Complex variables are two-component structures
- Write real and imaginary part of left-hand-side explicitly:

```
for(int i=0; i<N; i++) {
  y[i].real = a.real*x[i].real - a.imag*x[i].imag + y[i].real;
  y[i].imag = a.imag*x[i].real + a.real*x[i].imag + y[i].imag;
}
```

Practical

Complex "axpy" operation

- $y_i = ax_i + y_i$, $i=0,\ldots,N-1$, with x, y, complex arrays, a complex scalar

```
_Complex double x[N];
_Complex double y[N];
_Complex double a;
for(int i=0; i<N; i++) {
  y[i] = a*x[i] + y[i];
}
```

- Complex variables are two-component structures
- Write real and imaginary part of left-hand-side explicitly:

```
for(int i=0; i<N; i++) {
  y[i].real = a.real*x[i].real - a.imag*x[i].imag + y[i].real;
  y[i].imag = a.imag*x[i].real + a.real*x[i].imag + y[i].imag;
}
```

$$N_{FP} = 8N$$
$$N_{IO} = 2w(N+N+N) = 6wN$$

Practical

Complex “axpy” operation

- $y_i = ax_i + y_i$, $i=0,\ldots,N-1$, with x, y, complex arrays, a complex scalar

```
_Complex double x[N];
_Complex double y[N];
_Complex double a;
for(int i=0; i<N; i++) {
  y[i] = a*x[i] + y[i];
}
```

- Complex variables are two-component structures
- Write real and imaginary part of left-hand-side explicitly:

```
for(int i=0; i<N; i++) {
  y[i].real = a.real*x[i].real - a.imag*x[i].imag + y[i].real;
  y[i].imag = a.imag*x[i].real + a.real*x[i].imag + y[i].imag;
}
```

$$N_{FP} = 8N$$
$$N_{IO} = 2w(N+N+N) = 6wN$$

$$I_k = \frac{4}{3w}$$

Practical

Complex “axpy” operation

- $y_i = ax_i + y_i$, $i=0,\ldots,N-1$, with x, y, complex arrays, a complex scalar

```
_Complex double x[N];
_Complex double y[N];
_Complex double a;
for(int i=0; i<N; i++) {
  y[i] = a*x[i] + y[i];
}
```

- Complex variables are two-component structures
- Write real and imaginary part of left-hand-side explicitly:

```
for(int i=0; i<N; i++) {
  y[i].real = a.real*x[i].real - a.imag*x[i].imag + y[i].real;
  y[i].imag = a.imag*x[i].real + a.real*x[i].imag + y[i].imag;
}
```

$$N_{FP} = 8N$$
$$N_{IO} = 2w(N+N+N) = 6wN$$

$$I_k = \frac{4}{3w}$$ Memory-bound (on most architectures)

Outline – 4th Nov.

Computational aspects of lattice gauge theories

- Gauge theories on the lattice
  - Path integral formulation
  - Typical workflow
- Challenges
  - Errors, systematic uncertainties
  - Computational resource requirements
- Overview of some common methods and their impact
  - Sparse linear systems, multigrid solvers, Hybrid Monte Carlo
  - Impact on current state-of-the-art in lattice QCD
- Practical/technical aspects of implementing LGT
  - Computational intensity of the Schwinger fermion matrix

Gauge Theories on the Lattice

Introduction

- Action and Lagrangian:

$$S = \int d^4x \mathcal{L}(x), \quad \mathcal{L} = \bar{\psi} \underbrace{(i\not{D} - m)}_{M} \psi - \frac{1}{4} F^a_{\mu\nu} F_a^{\mu\nu}$$

where

$$F^a_{\mu\nu} = \partial_\mu G^a_\nu - \partial_\nu G^a_\mu - g f_{abc} G^b_\mu G^c_\nu$$

- Fermion fields: ψ
- Gauge (Gluon) fields: G

Gauge Theories on the Lattice

Introduction

- Action and Lagrangian:

$$S = \int d^4x \mathcal{L}(x), \quad \mathcal{L} = \bar{\psi} \underbrace{(i\not{D} - m)}_{M} \psi - \frac{1}{4} F^a_{\mu\nu} F_a^{\mu\nu}$$

where

$$F^a_{\mu\nu} = \partial_\mu G^a_\nu - \partial_\nu G^a_\mu - g f_{abc} G^b_\mu G^c_\nu$$

- Fermion fields: ψ
- Gauge (Gluon) fields: G
- Feynman path integral formulation:

$$\langle \mathcal{O} \rangle = \frac{1}{Z} \int \mathcal{D}[U]\mathcal{D}[\bar{\psi}]\mathcal{D}[\psi]\mathcal{O}(U, \psi, \bar{\psi}) e^{iS[U,\psi,\bar{\psi}]}$$

- Where we have introduced the *link variables* $U_\mu(x_1;x_0)$:

$$U_\mu(x_1; x_0) \equiv e^{-ig \int_{\mathcal{C}} dx G^c_\mu(x) T_c}$$

Gauge Theories on the Lattice

Introduction

- Feynman path integral formulation:

$$\langle \mathcal{O} \rangle = \frac{1}{Z} \int \mathcal{D}[U]\mathcal{D}[\bar{\psi}]\mathcal{D}[\psi]\mathcal{O}(U,\psi,\bar{\psi})e^{iS[U,\psi,\bar{\psi}]}$$

Gauge Theories on the Lattice

Introduction

- Feynman path integral formulation:

$$\langle \mathcal{O} \rangle = \frac{1}{Z} \int \mathcal{D}[U]\mathcal{D}[\bar{\psi}]\mathcal{D}[\psi]\mathcal{O}(U,\psi,\bar{\psi})e^{iS[U,\psi,\bar{\psi}]}$$

- Integration over field variables U and ψ via Monte Carlo:
  - ✗ Fermion fields are Grassmann-valued
  - ✗ Exponent (probability) is complex

Gauge Theories on the Lattice

Introduction

- Feynman path integral formulation:

$$\langle \mathcal{O} \rangle = \frac{1}{Z} \int \mathcal{D}[U]\mathcal{D}[\bar{\psi}]\mathcal{D}[\psi]\mathcal{O}(U,\psi,\bar{\psi})e^{iS[U,\psi,\bar{\psi}]}$$

- Integration over field variables U and ψ via Monte Carlo:
  - ✗ Fermion fields are Grassmann-valued
  - ✗ Exponent (probability) is complex

- Integration over fermion fields + Wick rotation to Euclidean time:

$$\langle \mathcal{O} \rangle = \frac{1}{Z} \int \mathcal{D}[U]\mathcal{O}(U, M^{-1})e^{-S_G[U]+\ln(\det(M[U]))}$$

Gauge Theories on the Lattice

Introduction

- Feynman path integral formulation:

$$\langle \mathcal{O} \rangle = \frac{1}{Z} \int \mathcal{D}[U]\mathcal{D}[\bar{\psi}]\mathcal{D}[\psi]\mathcal{O}(U,\psi,\bar{\psi})e^{iS[U,\psi,\bar{\psi}]}$$

- Integration over field variables U and ψ via Monte Carlo:
  - ✗ Fermion fields are Grassmann-valued
  - ✗ Exponent (probability) is complex

- Integration over fermion fields + Wick rotation to Euclidean time:

$$\langle \mathcal{O} \rangle = \frac{1}{Z} \int \mathcal{D}[U]\mathcal{O}(U, M^{-1})e^{-S_G[U]+\ln(\det(M[U]))}$$

- **Observables**: N-point correlation functions of fermions → functions of the inverse of the fermion matrix M (Wick's theorem/Wick contractions):

$$\langle \psi_{j_1}\psi_{j_2}...\bar{\psi}_{i_1}\bar{\psi}_{i_2}... \rangle = \langle (-1)^p M^{-1}_{i_1 j_1} M^{-1}_{i_2 j_2} ... + \text{all permutations} \rangle_U$$

Gauge Theories on the Lattice

Introduction

- Feynman path integral:

$$\langle \mathcal{O} \rangle = \frac{1}{Z} \int \mathcal{D}[U] \mathcal{O}(U, M^{-1}) e^{-S_G[U] + \ln(\det(M[U]))}$$

Gauge Theories on the Lattice

Introduction

- Feynman path integral:

$$\langle \mathcal{O} \rangle = \frac{1}{Z} \int \mathcal{D}[U] \mathcal{O}(U, M^{-1}) e^{-S_G[U] + \ln(\det(M[U]))}$$

- Determinant calculation:
  - ✗ Scales at best with: $\mathcal{O}(N^{2.373})$, where N is size of matrix
  - ✗ Non-local

Gauge Theories on the Lattice

Introduction

- Feynman path integral:

$$\langle \mathcal{O} \rangle = \frac{1}{Z} \int \mathcal{D}[U] \mathcal{O}(U, M^{-1}) e^{-S_G[U] + \ln(\det(M[U]))}$$

- Determinant calculation:
  - ✗ Scales at best with: $\mathcal{O}(N^{2.373})$, where N is size of matrix
  - ✗ Non-local
- Integrate over complex-valued pseudo-fermions, e.g. for a fermion doublet:

$$\langle \mathcal{O} \rangle = \frac{1}{Z} \int \mathcal{D}[U] \mathcal{D}[\bar{\phi}] \mathcal{D}[\phi] \mathcal{O} e^{-S_G[U] - \bar{\phi}(M^+ M)^{-1} \phi}$$

Gauge Theories on the Lattice

Introduction

- Feynman path integral:

$$\langle \mathcal{O} \rangle = \frac{1}{Z} \int \mathcal{D}[U] \mathcal{O}(U, M^{-1}) e^{-S_G[U] + \ln(\det(M[U]))}$$

- Determinant calculation:
  - ✗ Scales at best with: $\mathcal{O}(N^{2.373})$, where N is size of matrix
  - ✗ Non-local
- Integrate over complex-valued pseudo-fermions, e.g. for a fermion doublet:

$$\langle \mathcal{O} \rangle = \frac{1}{Z} \int \mathcal{D}[U] \mathcal{D}[\bar{\phi}] \mathcal{D}[\phi] \mathcal{O} e^{-S_G[U] - \bar{\phi}(M^+ M)^{-1} \phi}$$

- Inversion of M
  - ✓ M is sparse $\Rightarrow M\phi$ is $\mathcal{O}(N)$ operations
  - ✓ $M^{-1}\phi$ via iterative methods: $\mathcal{O}(n)$ applications of $M\phi$, $n \ll N$

Gauge Theories on the Lattice

Introduction

- Discretization. *Link variables* and *gauge action*:

$$U_\mu(x_1; x_0) \equiv e^{-ig\int^C dx G^c_\mu(x) T_c} \rightarrow U_\mu(x_n) \equiv e^{-iagG^c_\mu(x_n)T_c}$$

Gauge Theories on the Lattice

Introduction

- Discretization. *Link variables* and *gauge action*:

$$U_\mu(x_1; x_0) \equiv e^{-ig\int^{\mathcal{C}} dx G^c_\mu(x) T_c} \rightarrow U_\mu(x_n) \equiv e^{-iagG^c_\mu(x_n)T_c}$$

  - E.g. Wilson *gauge action*:

$$S_G[U] = -\frac{1}{4}\int d^4x F^c_{\mu\nu}F^{\mu\nu}_c \rightarrow \beta \sum_n \sum_{\mu<\nu} \Re\left\{ \mathrm{Tr}\frac{1}{3}\left[1 - P_{\mu\nu}(x_n)\right] \right\}$$

Gauge Theories on the Lattice

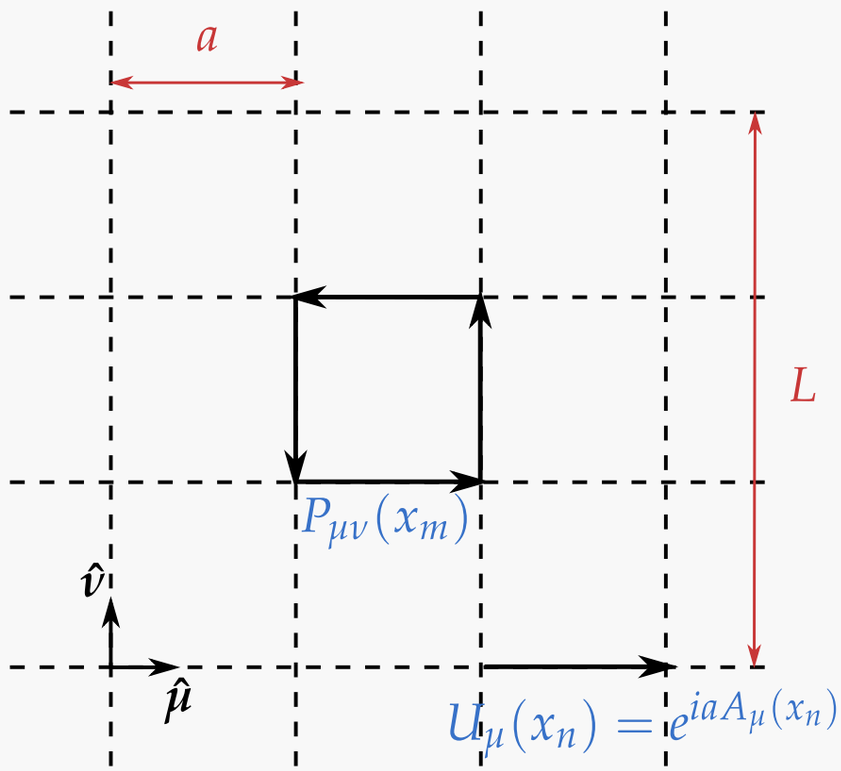
Introduction

- Discretization. *Link variables* and *gauge action*:

$$U_\mu(x_1;x_0) \equiv e^{-ig\int^C dx G^c_\mu(x)T_c} \to U_\mu(x_n) \equiv e^{-iagG^c_\mu(x_n)T_c}$$

  - E.g. Wilson *gauge action*:

$$S_G[U] = -\frac{1}{4}\int d^4x F^c_{\mu\nu}F^{\mu\nu}_c \to \beta \sum_n \sum_{\mu<\nu} \Re\left\{ \mathrm{Tr}\frac{1}{3}\left[1 - P_{\mu\nu}(x_n)\right]\right\}$$



Gauge Theories on the Lattice

Introduction

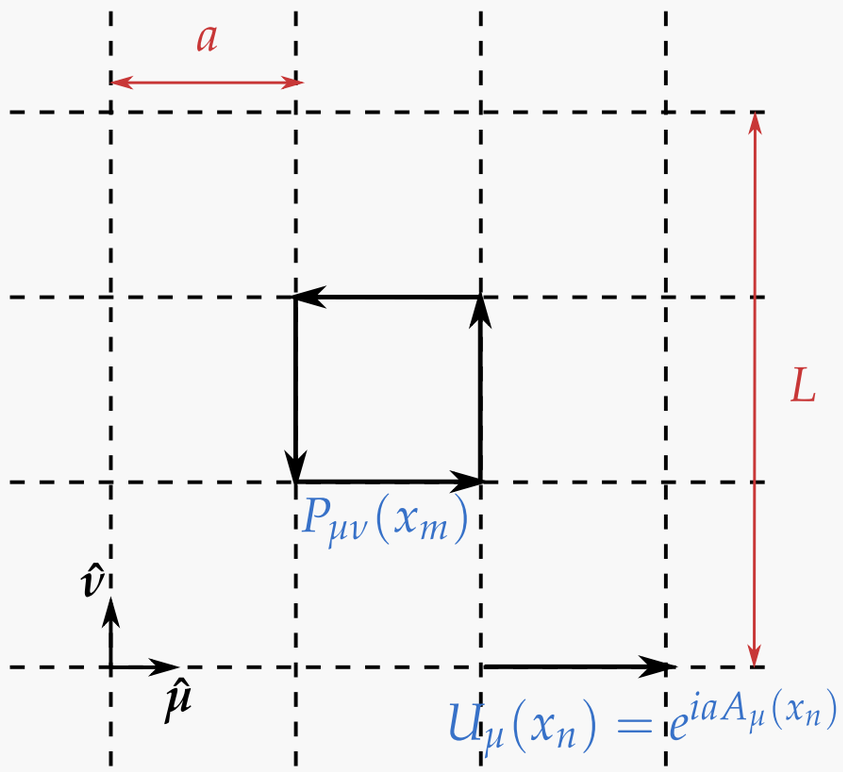
- Discretization. *Link variables* and *gauge action*:

$$U_\mu(x_1; x_0) \equiv e^{-ig\int^C dx G^c_\mu(x) T_c} \to U_\mu(x_n) \equiv e^{-iagG^c_\mu(x_n)T_c}$$

  - E.g. Wilson *gauge action*:

$$S_G[U] = -\frac{1}{4}\int d^4x F^c_{\mu\nu}F^{\mu\nu}_c \to \beta \sum_n \sum_{\mu<\nu} \Re\left\{ \mathrm{Tr}\frac{1}{3}\left[1 - P_{\mu\nu}(x_n)\right]\right\}$$

- Space-time lattice with **lattice spacing** a.



Gauge Theories on the Lattice

Introduction

- Fermion action written in terms of covariant derivatives, e.g. for Wilson fermions

$$S_F = \bar{\psi} M \psi \to -a^4 \sum_{x,\mu} \frac{1}{2a} [\bar{\psi}(x) U_\mu(x)(r-\gamma_\mu)\psi(x+a\hat{\mu}) + \bar{\psi}(x)(r+\gamma_\mu)U_{-\mu}(x)\psi(x-a\hat{\mu})]$$
$$+ a^4 \sum_x (m + \frac{4}{a} r)\bar{\psi}(x)\psi(x)$$

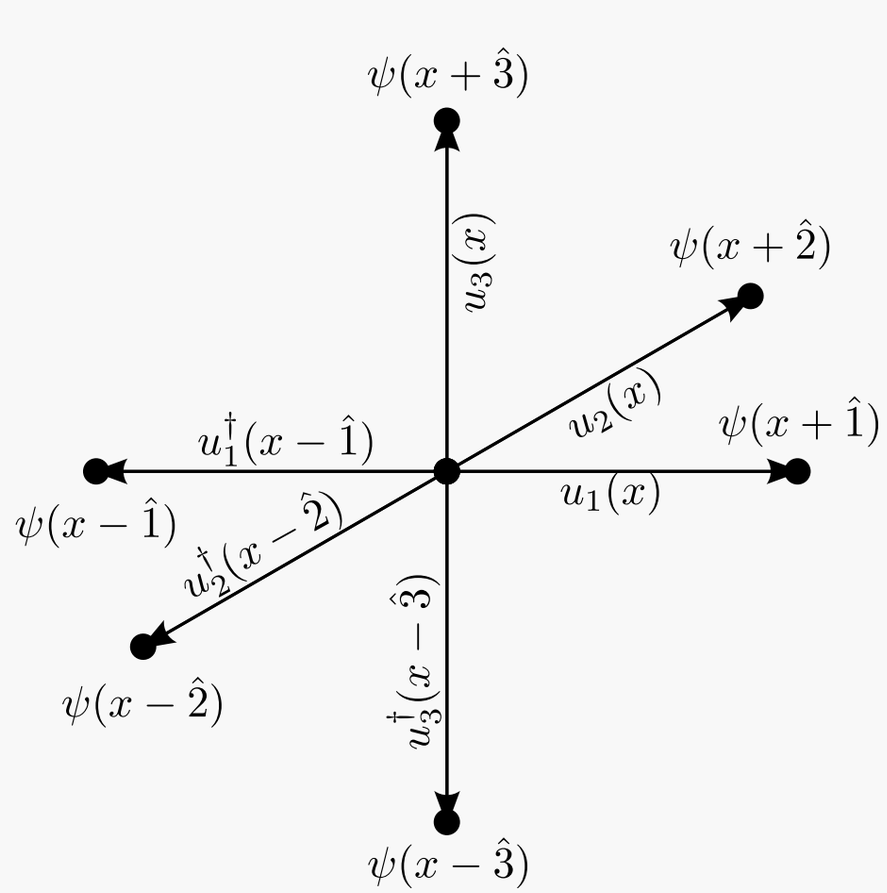
Gauge Theories on the Lattice

Introduction

- Fermion action written in terms of covariant derivatives, e.g. for Wilson fermions

$$S_F = \bar{\psi} M \psi \to -a^4 \sum_{x,\mu} \frac{1}{2a} [\bar{\psi}(x) U_\mu(x)(r-\gamma_\mu)\psi(x+a\hat{\mu}) + \bar{\psi}(x)(r+\gamma_\mu)U_{-\mu}(x)\psi(x-a\hat{\mu})]$$

$$+ a^4 \sum_x (m + \frac{4}{a} r)\bar{\psi}(x)\psi(x)$$



Gauge Theories on the Lattice

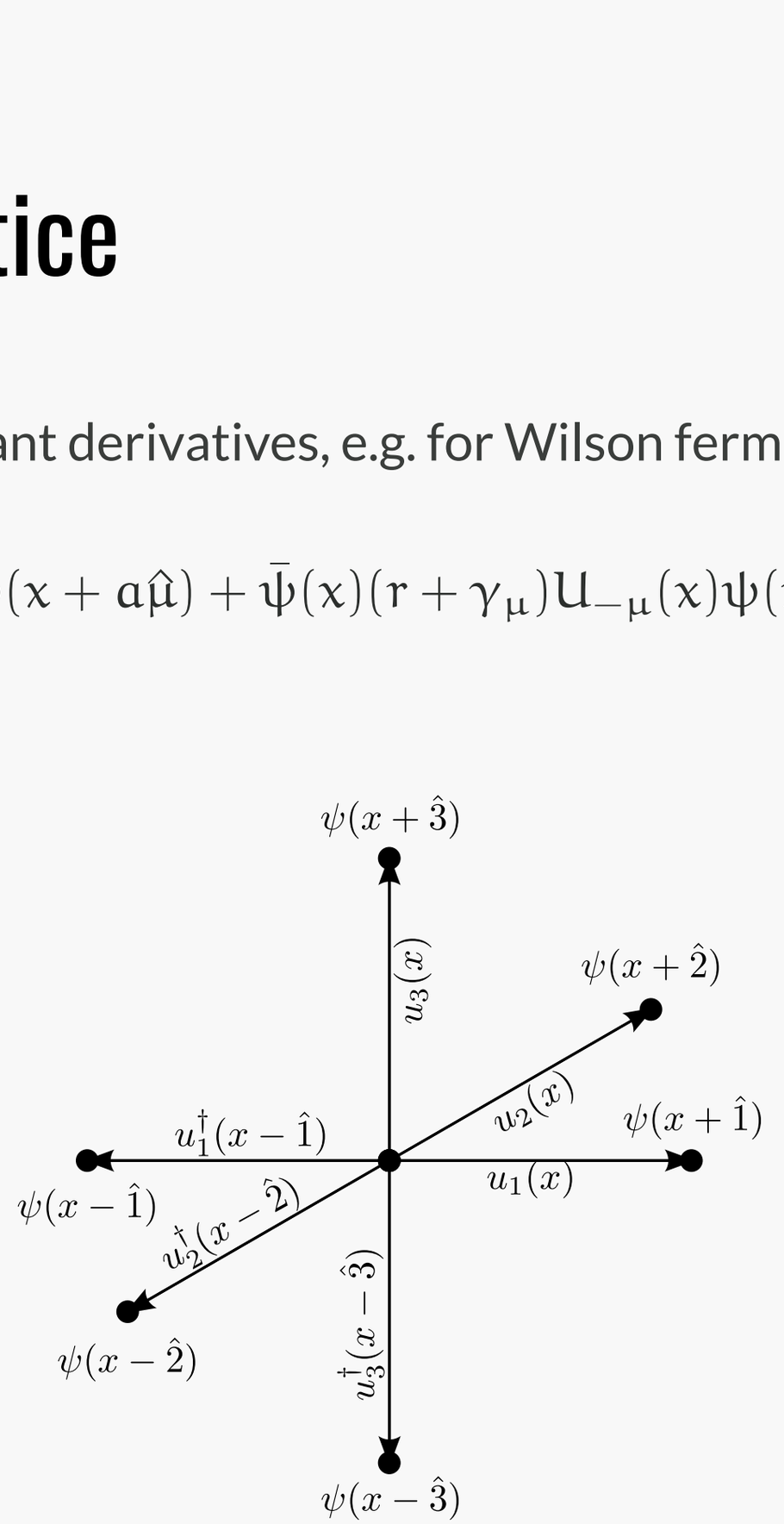
Introduction

- Fermion action written in terms of covariant derivatives, e.g. for Wilson fermions

$$S_F = \bar{\psi} M \psi \to -a^4 \sum_{x,\mu} \frac{1}{2a} [\bar{\psi}(x) U_\mu(x)(r - \gamma_\mu)\psi(x + a\hat{\mu}) + \bar{\psi}(x)(r + \gamma_\mu) U_{-\mu}(x)\psi(x - a\hat{\mu})]$$
$$+ a^4 \sum_x (m + \frac{4}{a} r)\bar{\psi}(x)\psi(x)$$

- 4-dimensional stencil
- Nearest neighbours in ψ
- In terms of covariant derivative:

$$\frac{1}{a}[U_\mu(x)\psi(x + a\hat{\mu}) - \psi(x)]$$



Gauge Theories on the Lattice

Freedom in choice of bare parameters:

- fermion masses (heavier is cheaper)
- lattice spacing/coupling a (larger is cheaper)
- lattice volume $L^3 \times T$ (smaller is cheaper)

Gauge Theories on the Lattice

Freedom in choice of bare parameters:

- fermion masses (heavier is cheaper)
- lattice spacing/coupling α (larger is cheaper)
- lattice volume $L^3 \times T$ (smaller is cheaper)

Choice of discretisation scheme:

Clover, Twisted Mass, Staggered, Overlap, Domain Wall

Trade — offs and advantages for each differ

Gauge Theories on the Lattice

Freedom in choice of bare parameters:

- fermion masses (heavier is cheaper)
- lattice spacing/coupling a (larger is cheaper)
- lattice volume $L^3 \times T$ (smaller is cheaper)

Choice of discretisation scheme:

Clover, Twisted Mass, Staggered, Overlap, Domain Wall

Trade — offs and advantages for each differ

Eventually, all schemes must agree:

Limits: $a \to 0;\ L \to \infty; m \to m_{phys}$

Gauge Theories on the Lattice

The typical workflow

$$\langle \mathcal{O} \rangle = \frac{1}{Z} \int \mathcal{D}[U]\mathcal{D}[\bar{\phi}]\mathcal{D}[\phi]\mathcal{O} e^{-S_G[U] - \bar{\phi}(M^+M)^{-1}\phi}$$

- The typical workflow

Gauge Theories on the Lattice

The typical workflow

$$\langle \mathcal{O} \rangle = \frac{1}{Z} \int \mathcal{D}[U]\mathcal{D}[\bar{\phi}]\mathcal{D}[\phi]\mathcal{O} e^{-S_G[U] - \bar{\phi}(M^+M)^{-1}\phi}$$

- The typical workflow
  - **Markov-chain Monte Carlo (MCMC) process:** Generate representative configurations of the gauge fields U with probability:

$$p(U) = \frac{1}{Z} \int \mathcal{D}[\bar{\phi}]\mathcal{D}[\phi] e^{-S_G[U] - \bar{\phi}(M^+M)^{-1}\phi}$$

Gauge Theories on the Lattice

The typical workflow

$$\langle \mathcal{O} \rangle = \frac{1}{Z} \int \mathcal{D}[U]\mathcal{D}[\bar{\phi}]\mathcal{D}[\phi]\mathcal{O} e^{-S_G[U] - \bar{\phi}(M^+M)^{-1}\phi}$$

- The typical workflow
  - **Markov-chain Monte Carlo (MCMC) process:** Generate representative configurations of the gauge fields U with probability:

$$p(U) = \frac{1}{Z} \int \mathcal{D}[\bar{\phi}]\mathcal{D}[\phi] e^{-S_G[U] - \bar{\phi}(M^+M)^{-1}\phi}$$

  - **Observable "measurements":** Compute expectation values of fermion or gluon correlation functions over the ensemble of Us

Gauge Theories on the Lattice

The typical workflow

$$\langle \mathcal{O} \rangle = \frac{1}{Z} \int \mathcal{D}[U]\mathcal{D}[\bar{\phi}]\mathcal{D}[\phi]\mathcal{O} e^{-S_G[U] - \bar{\phi}(M^+M)^{-1}\phi}$$

- The typical workflow
  - **Markov-chain Monte Carlo (MCMC) process:** Generate representative configurations of the gauge fields U with probability:

$$p(U) = \frac{1}{Z} \int \mathcal{D}[\bar{\phi}]\mathcal{D}[\phi] e^{-S_G[U] - \bar{\phi}(M^+M)^{-1}\phi}$$

  - **Observable "measurements":** Compute expectation values of fermion or gluon correlation functions over the ensemble of Us

- Inversions of the Dirac matrix M
  - Needed during MCMC: $\phi(M^+M)^{-1}\phi$
  - Needed during "measurements"

Gauge Theories on the Lattice

Current state-of-the-art in lattice QCD

- Typical problem sizes
  - Box length L ~ 5 – 7 fm
  - Lattice spacing a ~ 0.06 – 0.1 fm

Gauge Theories on the Lattice

Current state-of-the-art in lattice QCD

- Typical problem sizes
  - Box length L ~ 5 – 7 fm
  - Lattice spacing a ~ 0.06 – 0.1 fm

} **Lattice sizes:** $64^3 \times 128$ – $128^3 \times 256$ grid-points

Gauge Theories on the Lattice

Current state-of-the-art in lattice QCD

- Typical problem sizes
  - Box length L ~ 5 – 7 fm
  - Lattice spacing a ~ 0.06 – 0.1 fm

  } **Lattice sizes:** $64^3 \times 128$ – $128^3 \times 256$ grid-points

- Data types: link variables and *propagators*
  - Link variables are *color matrices*, i.e. $U_\mu(x)$ is 3×3, complex valued, μ=0,…,3 for each of the 4 dimensions ⇒ 36 complex numbers per grid-point

Gauge Theories on the Lattice

Current state-of-the-art in lattice QCD

- Typical problem sizes
  - Box length L ~ 5 – 7 fm
  - Lattice spacing a ~ 0.06 – 0.1 fm

  } **Lattice sizes:** $64^3 \times 128$ – $128^3 \times 256$ grid-points

- Data types: link variables and *propagators*
  - Link variables are *color matrices*, i.e. $U_\mu(x)$ is 3×3, complex valued, μ=0,…,3 for each of the 4 dimensions ⇒ 36 complex numbers per grid-point
  - Propagators are columns (or rows) of M^{-1} or more generally are vectors: $y = M^{-1} b$ ⇒ 3×4 = 12 complex numbers per grid-point

Gauge Theories on the Lattice

Current state-of-the-art in lattice QCD

- Typical problem sizes
  - Box length L ~ 5 – 7 fm
  - Lattice spacing a ~ 0.06 – 0.1 fm

  } **Lattice sizes:** $64^3 \times 128$ – $128^3 \times 256$ grid-points

- Data types: link variables and *propagators*
  - Link variables are *color matrices*, i.e. $U_\mu(x)$ is 3×3, complex valued, μ=0,…,3 for each of the 4 dimensions ⇒ 36 complex numbers per grid-point
  - Propagators are columns (or rows) of M^{-1} or more generally are vectors: $y = M^{-1} b$ ⇒ 3×4 = 12 complex numbers per grid-point

- For a $96^3 \times 192$ lattice
  - A gauge-field is ~100 GBytes
  - A vector is ~30 GBytes

Gauge Theories on the Lattice

Current state-of-the-art in lattice QCD

- Typical problem sizes
  - Box length L ~ 5 – 7 fm
  - Lattice spacing α ~ 0.06 – 0.1 fm

  } **Lattice sizes:** $64^3\times128$ – $128^3\times256$ grid-points

- Data types: link variables and *propagators*
  - Link variables are *color matrices*, i.e. $\mathrm{U}_\mu(\mathrm{x})$ is 3×3, complex valued, μ=0,…,3 for each of the 4 dimensions ⇒ 36 complex numbers per grid-point
  - Propagators are columns (or rows) of M^{-1} or more generally are vectors: $\mathrm{y}=\mathrm{M}^{-1}\mathrm{b}$ ⇒ 3×4 = 12 complex numbers per grid-point

- For a $96^3\times192$ lattice
  - A gauge-field is ~100 GBytes
  - A vector is ~30 GBytes
  - A typical implementation may require two gauge-fields plus $\mathcal{O}(10)$ vectors

    ⇒ Memory (RAM) requirements are $\mathcal{O}(1)$ TBytes

    ⇒For an ensemble of 1000 configurations, storage can reach $\mathcal{O}(1)$ PBytes

Gauge Theories on the Lattice

Computational characteristics

- **Computationally demanding**
  - 4-dimensional theories, problem scales with L^4
  - Markov-chain Monte Carlo; statistical errors scale as: $1/\sqrt{\mathrm{N}_{stat}}$
  - Euclidean-time correlation functions; signal decays as: $e^{-\mathrm{Et}}$

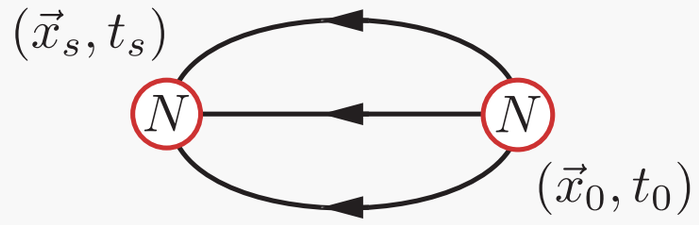
Gauge Theories on the Lattice

Computational characteristics

- **Computationally demanding**
  - 4-dimensional theories, problem scales with L^4
  - Markov-chain Monte Carlo; statistical errors scale as: $1/\sqrt{\mathrm{N}_{stat}}$
  - Euclidean-time correlation functions; signal decays as: $e^{-\mathrm{Et}}$

E.g. Two-point correlation functions

$$\sum_{\vec{x}_s} \Gamma^{\alpha\beta} \langle \bar{\chi}_N^{\beta}(x_s) | \chi_N^{\alpha}(0) \rangle = c_0 e^{-E_0 t_s} + c_1 e^{-E_1 t_s} + \ldots$$



- Correlation function signal decays exponentially with time-separation
- Statistical error constant with time-separation
- Exponentially decaying signal!

Gauge Theories on the Lattice

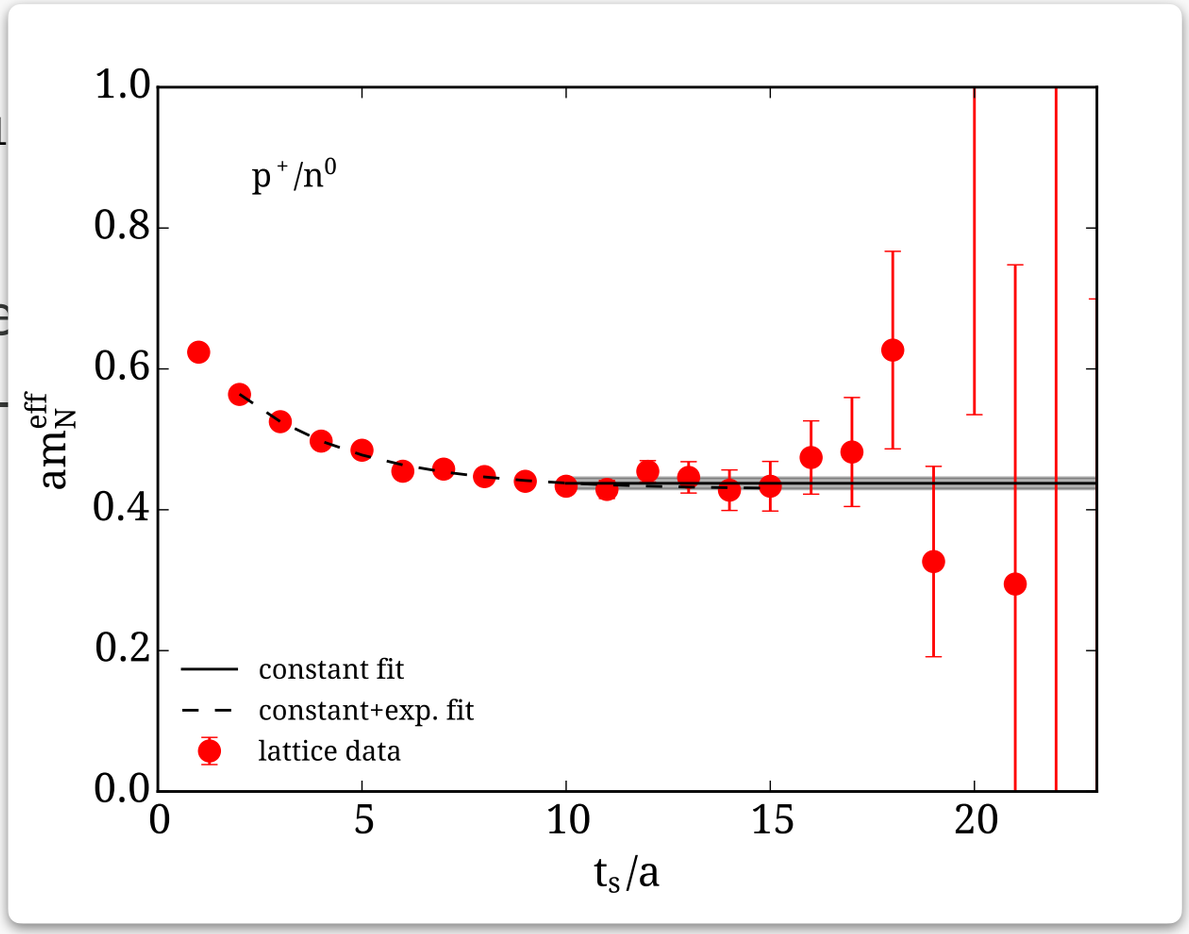
Computational characteristics

- **Computationally demanding**
  - 4-dimensional theories, problem scales with L^4
  - Markov-chain Monte Carlo; statistical errors scale as: $1/\sqrt{\mathrm{N}_{stat}}$
  - Euclidean-time correlation functions; signal decays as: $e^{-\mathrm{E}t}$

E.g. Two-point correlation functions

$$\sum_{\vec{x}_s} \Gamma^{\alpha\beta} \langle \bar{\chi}_N^{\beta}(x_s) | \chi_N^{\alpha}(0) \rangle = c_0 e^{-E_0 t_s} + c_1$$

- Correlation function signal decays e
- Statistical error constant with time-
- Exponentially decaying signal!



Gauge Theories on the Lattice

Computational characteristics

- **Computationally demanding**
  - 4-dimensional theories, problem scales with L^4
  - Markov-chain Monte Carlo; statistical errors scale as: $1/\sqrt{\mathrm{N}_{stat}}$
  - Euclidean-time correlation functions; signal decays as: $e^{-\mathrm{E}t}$
- **Well-suited for parallel computers**
  - Nearest neighbour communications; predictable and regular communication patterns; potential for *good strong-scaling*
  - Relatively simple compute kernels. Deterministic memory access patterns; potential for *highly-efficient* kernel implementations

Gauge Theories on the Lattice

Computational characteristics

- **Computationally demanding**
  - 4-dimensional theories, problem scales with L^4
  - Markov-chain Monte Carlo; statistical errors scale as: $1/\sqrt{\mathrm{N}_{stat}}$
  - Euclidean-time correlation functions; signal decays as: $e^{-\mathrm{E}t}$

- **Well-suited for parallel computers**
  - Nearest neighbour communications; predictable and regular communication patterns; potential for *good strong-scaling*
  - Relatively simple compute kernels. Deterministic memory access patterns; potential for *highly-efficient* kernel implementations

- **Typical computational requirements**
  - **Simulation:** generation of 500 configurations of a $96^3 \times 192$ lattice with $\mathrm{a} = 0.08$ fm and $\mathrm{m}_\pi \approx 135$ MeV (*physical point*): **~100 Mi core-hours**

Gauge Theories on the Lattice

Computational characteristics

- **Computationally demanding**
  - 4-dimensional theories, problem scales with L^4
  - Markov-chain Monte Carlo; statistical errors scale as: $1/\sqrt{\mathrm{N}_{stat}}$
  - Euclidean-time correlation functions; signal decays as: $e^{-\mathrm{E}t}$

- **Well-suited for parallel computers**
  - Nearest neighbour communications; predictable and regular communication patterns; potential for *good strong-scaling*
  - Relatively simple compute kernels. Deterministic memory access patterns; potential for *highly-efficient* kernel implementations

- **Typical computational requirements**
  - **Simulation:** generation of 500 configurations of a $96^3 \times 192$ lattice with $\mathrm{a} = 0.08$ fm and $\mathrm{m}_\pi \approx 135$ MeV (*physical point*): **~100 Mi core-hours**
  - **Analysis:** 10,000 matrix-inversions per configuration (typical requirements for hadron structure): **~10 Mi GPU-hours**

Iterative solvers

Sparse matrix inversion via iterative solvers

- **Inversion of the fermion matrix is the most common computational task**
  - **During simulation**: for fermion determinant calculation
  - **During analysis**: for obtaining correlation functions

Iterative solvers

Sparse matrix inversion via iterative solvers

- **Inversion of the fermion matrix is the most common computational task**
  - **During simulation**: for fermion determinant calculation
  - **During analysis**: for obtaining correlation functions

- **Solve for y in:**

$$My = b \Rightarrow y = M^{-1}b$$

Iterative solvers

Sparse matrix inversion via iterative solvers

- **Inversion of the fermion matrix is the most common computational task**
  - **During simulation**: for fermion determinant calculation
  - **During analysis**: for obtaining correlation functions

- **Solve for y in:**

$$My = b \Rightarrow y = M^{-1}b$$

- **Krylov sub-space methods**
  - Build y from Mb, M^2b, M^3b, ...
  - Minimise iteratively *the error*: $|y - M^{-1}b|$
  - Since $M^{-1}b$ is unknown, in practice minimise *the residual*: $|My - b|$

Iterative solvers

Sparse matrix inversion via iterative solvers

- **Inversion of the fermion matrix is the most common computational task**
  - **During simulation**: for fermion determinant calculation
  - **During analysis**: for obtaining correlation functions

- **Solve for y in:**

$$My = b \Rightarrow y = M^{-1}b$$

- **Krylov sub-space methods**
  - Build y from Mb, M^2b, M^3b, ...
  - Minimise iteratively *the error*: $|y - M^{-1}b|$
  - Since $M^{-1}b$ is unknown, in practice minimise *the residual*: $|My - b|$
  - **Examples**: Conjugate Gradient (CG), Generalised Minimum Residual (GMRES), Stabilised Bi-Conjugate Gradient (BiCGStab), etc.

Iterative solvers

Sparse matrix inversion via iterative solvers

- **Example:** Conjugate Gradient
  - Requires a Hermitian matrix:

$$\mathrm{M}\mathrm{y} = \mathrm{b} \Rightarrow \mathrm{M}^\dagger\mathrm{M}\mathrm{y} = \mathrm{M}^\dagger\mathrm{b} \Rightarrow \mathrm{y} = (\mathrm{M}^\dagger\mathrm{M})^{-1}\mathrm{M}^\dagger\mathrm{b}$$

  - Monotonic convergence: $\mathrm{r}^\dagger\mathrm{M}^\dagger\mathrm{M}\mathrm{r}$, is guaranteed to decrease with every iteration

Iterative solvers

Sparse matrix inversion via iterative solvers

- **Example:** Conjugate Gradient
  - Requires a Hermitian matrix:

$$\mathrm{M}\mathrm{y} = \mathrm{b} \Rightarrow \mathrm{M}^\dagger\mathrm{M}\mathrm{y} = \mathrm{M}^\dagger\mathrm{b} \Rightarrow \mathrm{y} = (\mathrm{M}^\dagger\mathrm{M})^{-1}\mathrm{M}^\dagger\mathrm{b}$$

  - Monotonic convergence: $\mathrm{r}^\dagger\mathrm{M}^\dagger\mathrm{M}\mathrm{r}$, is guaranteed to decrease with every iteration

$\mathrm{x} = \mathrm{x}_0$ ◁ Initial guess
$\mathrm{r} = \mathrm{b} - A\mathrm{x}$ ◁ $A = \mathrm{M}^\dagger\mathrm{M}$
$\mathrm{p} = \mathrm{r}$
$\rho = \|\mathrm{r}\|$
while $\rho > \rho_{min}$ **do**
    $\mathrm{p}' = A\mathrm{p}$
    $\alpha = \rho/(\mathrm{p}^\dagger\mathrm{p}')$
    $\mathrm{x} \leftarrow \mathrm{x} + \alpha\mathrm{p}$
    $\mathrm{r} \leftarrow \mathrm{x} - \alpha\mathrm{p}'$
    $\beta = \|\mathrm{r}\|/\rho$
    $\mathrm{r} \leftarrow \mathrm{r} + \beta\mathrm{p}$
    $\rho \leftarrow \beta\rho$
end while

Iterative solvers

Sparse matrix inversion via iterative solvers

- **Example:** Conjugate Gradient
  - Requires a Hermitian matrix:

$$\mathrm{M}\mathrm{y} = \mathrm{b} \Rightarrow \mathrm{M}^\dagger\mathrm{M}\mathrm{y} = \mathrm{M}^\dagger\mathrm{b} \Rightarrow \mathrm{y} = (\mathrm{M}^\dagger\mathrm{M})^{-1}\mathrm{M}^\dagger\mathrm{b}$$

  - Monotonic convergence: $\mathrm{r}^\dagger\mathrm{M}^\dagger\mathrm{M}\mathrm{r}$, is guaranteed to decrease with every iteration

$\mathrm{x} = \mathrm{x}_0$ ◁ Initial guess
$\mathrm{r} = \mathrm{b} - A\mathrm{x}$ ◁ $A = \mathrm{M}^\dagger\mathrm{M}$
$\mathrm{p} = \mathrm{r}$
$\rho = \|\mathrm{r}\|$
while $\rho > \rho_{min}$ **do**
    $\mathrm{p}' = A\mathrm{p}$ ◁ mat-vec
    $\alpha = \rho/(\mathrm{p}^\dagger\mathrm{p}')$ ◁ dot-prod
    $\mathrm{x} \leftarrow \mathrm{x} + \alpha\mathrm{p}$ ◁ axpy
    $\mathrm{r} \leftarrow \mathrm{x} - \alpha\mathrm{p}'$ ◁ axpy
    $\beta = \|\mathrm{r}\|/\rho$ ◁ dot-prod
    $\mathrm{r} \leftarrow \mathrm{r} + \beta\mathrm{p}$ ◁ axpy
    $\rho \leftarrow \beta\rho$
end while

Iterative solvers

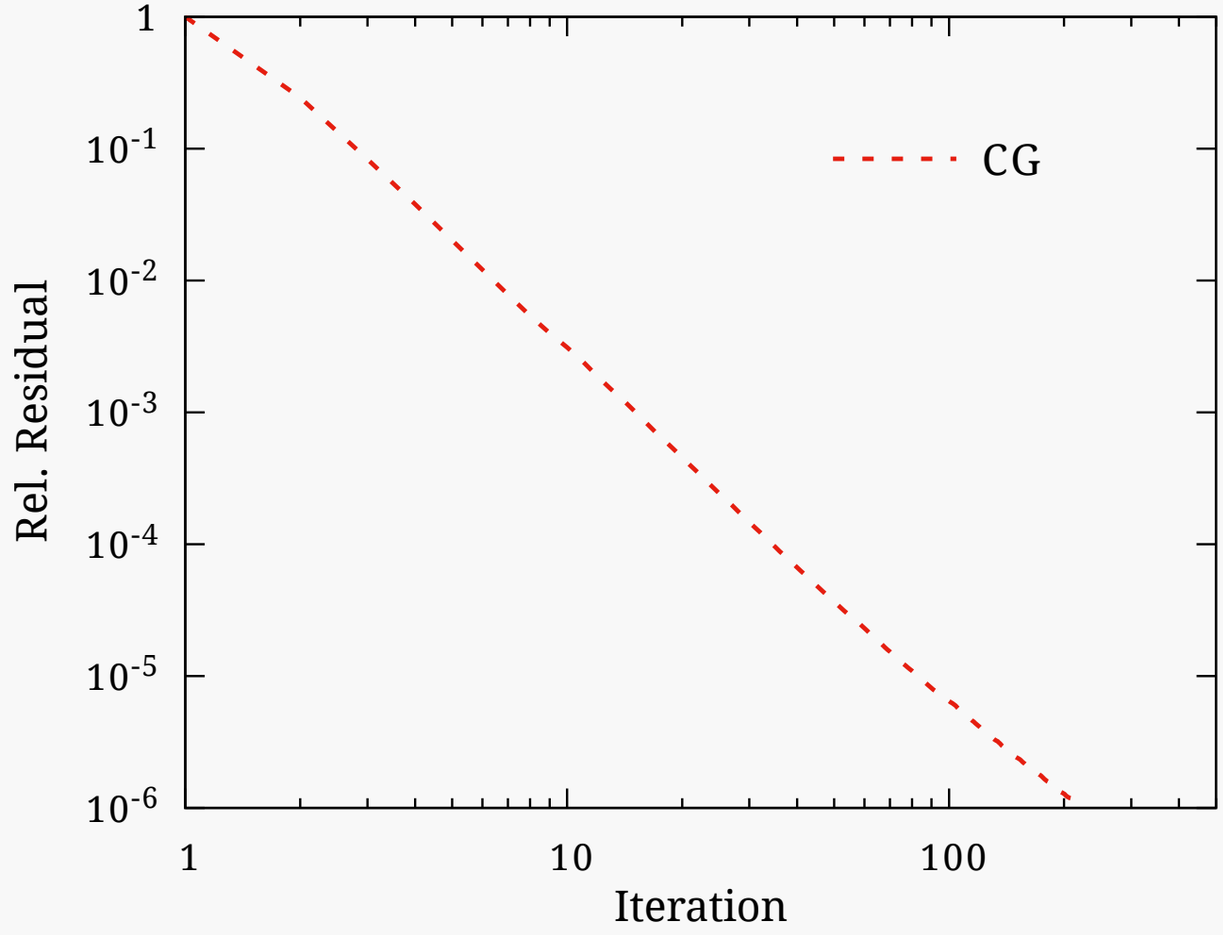
Sparse matrix inversion via iterative solvers

- **Example:** Conjugate Gradient
  - Requires a Hermitian matrix:

$$\mathrm{M}\mathrm{y} = \mathrm{b} \Rightarrow \mathrm{M}^\dagger\mathrm{M}\mathrm{y} = \mathrm{M}^\dagger\mathrm{b} \Rightarrow \mathrm{y} = (\mathrm{M}^\dagger\mathrm{M})^{-1}\mathrm{M}^\dagger\mathrm{b}$$

  - Monotonic convergence: $\mathrm{r}^\dagger\mathrm{M}^\dagger\mathrm{M}\mathrm{r}$, is guaranteed to decrease with every iteration

$\mathrm{x} = \mathrm{x}_0$ ◁ Initial guess
$\mathrm{r} = \mathrm{b} - \mathrm{A}\mathrm{x}$ ◁ $\mathrm{A} = \mathrm{M}^\dagger\mathrm{M}$
$\mathrm{p} = \mathrm{r}$
$\rho = \|\mathrm{r}\|$
while $\rho > \rho_{min}$ **do**
  $\mathrm{p}' = \mathrm{A}\mathrm{p}$ ◁ mat-vec
  $\alpha = \rho/(\mathrm{p}^\dagger\mathrm{p}')$ ◁ dot-prod
  $\mathrm{x} \leftarrow \mathrm{x} + \alpha\mathrm{p}$ ◁ axpy
  $\mathrm{r} \leftarrow \mathrm{x} - \alpha\mathrm{p}'$ ◁ axpy
  $\beta = \|\mathrm{r}\|/\rho$ ◁ dot-prod
  $\mathrm{r} \leftarrow \mathrm{r} + \beta\mathrm{p}$ ◁ axpy
  $\rho \leftarrow \beta\rho$
end while



Iterative solvers

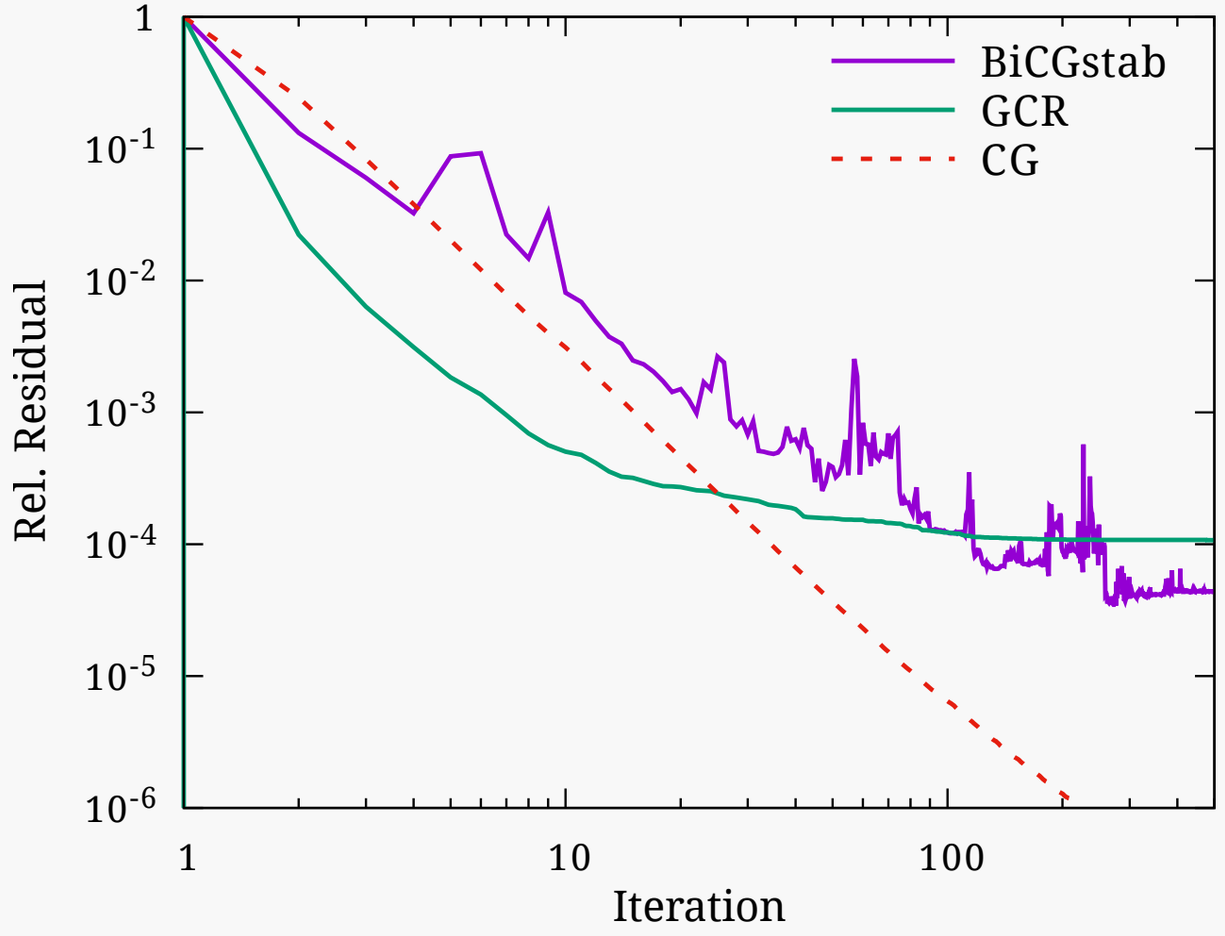
Sparse matrix inversion via iterative solvers

- **Example:** Conjugate Gradient
  - Requires a Hermitian matrix:

$$\mathrm{M}\mathrm{y} = \mathrm{b} \Rightarrow \mathrm{M}^\dagger\mathrm{M}\mathrm{y} = \mathrm{M}^\dagger\mathrm{b} \Rightarrow \mathrm{y} = (\mathrm{M}^\dagger\mathrm{M})^{-1}\mathrm{M}^\dagger\mathrm{b}$$

  - Monotonic convergence: $\mathrm{r}^\dagger\mathrm{M}^\dagger\mathrm{M}\mathrm{r}$, is guaranteed to decrease with every iteration

$\mathrm{x} = \mathrm{x}_0$ ◁ Initial guess
$\mathrm{r} = \mathrm{b} - A\mathrm{x}$ ◁ $A = \mathrm{M}^\dagger\mathrm{M}$
$\mathrm{p} = \mathrm{r}$
$\rho = \|\mathrm{r}\|$
while $\rho > \rho_{min}$ **do**
  $\mathrm{p}' = A\mathrm{p}$ ◁ mat-vec
  $\alpha = \rho/(\mathrm{p}^\dagger\mathrm{p}')$ ◁ dot-prod
  $\mathrm{x} \leftarrow \mathrm{x} + \alpha\mathrm{p}$ ◁ axpy
  $\mathrm{r} \leftarrow \mathrm{x} - \alpha\mathrm{p}'$ ◁ axpy
  $\beta = \|\mathrm{r}\|/\rho$ ◁ dot-prod
  $\mathrm{r} \leftarrow \mathrm{r} + \beta\mathrm{p}$ ◁ axpy
  $\rho \leftarrow \beta\rho$
end while



Iterative solvers

Sparse matrix inversion via iterative solvers

- **Krylov sub-space methods**
  - Residual for a given number of iterations related to *condition number*:

$$\rho = \frac{||r||}{||b||} \leq 2(\frac{\sqrt{K(A)}-1}{\sqrt{K(A)}+1})^n, \quad K(A) = \frac{|\lambda_{max}|}{|\lambda_{min}|}$$

Iterative solvers

Sparse matrix inversion via iterative solvers

- **Krylov sub-space methods**
  - Residual for a given number of iterations related to *condition number*:

$$\rho = \frac{||r||}{||b||} \leq 2(\frac{\sqrt{K(A)}-1}{\sqrt{K(A)}+1})^n, \quad K(A) = \frac{|\lambda_{max}|}{|\lambda_{min}|}$$

  - For large condition number:

$$\rho \xrightarrow{K(A)\gg 1} (1 - \frac{2}{\sqrt{K(A)}})^n$$

Iterative solvers

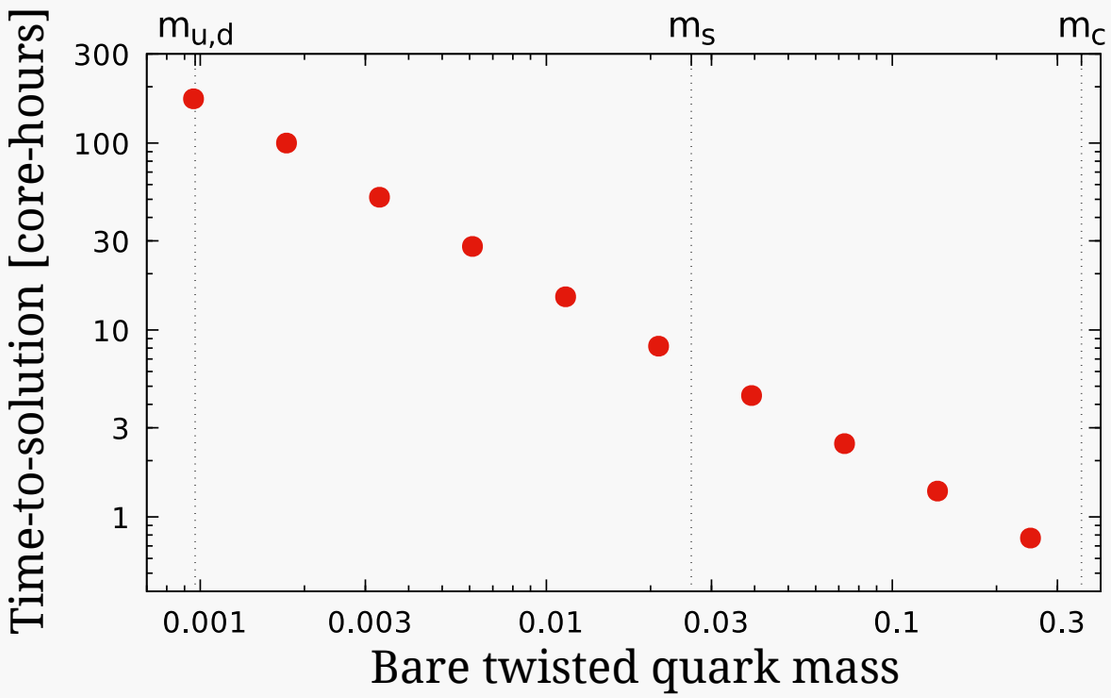
Sparse matrix inversion via iterative solvers

- **Krylov sub-space methods**
  - Residual for a given number of iterations related to *condition number*:

$$\rho = \frac{||r||}{||b||} \leq 2(\frac{\sqrt{K(A)}-1}{\sqrt{K(A)}+1})^n, \quad K(A) = \frac{|\lambda_{max}|}{|\lambda_{min}|}$$

  - For large condition number:

$$\rho \xrightarrow{K(A)\gg 1} (1 - \frac{2}{\sqrt{K(A)}})^n$$



Iterative solvers

Sparse matrix inversion via iterative solvers

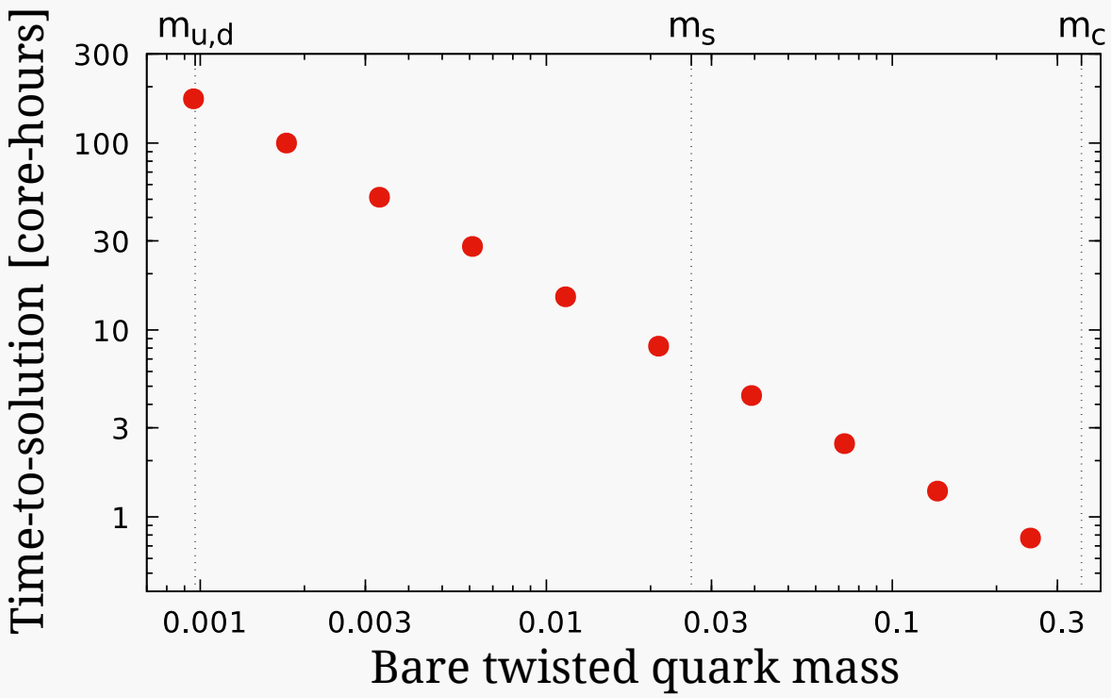
- **Krylov sub-space methods**
  - Residual for a given number of iterations related to *condition number*:

$$\rho = \frac{||r||}{||b||} \leq 2\left(\frac{\sqrt{K(A)}-1}{\sqrt{K(A)}+1}\right)^n, \quad K(A) = \frac{|\lambda_{max}|}{|\lambda_{min}|}$$

  - For large condition number:

$$\rho \xrightarrow{K(A)\gg 1} \left(1 - \frac{2}{\sqrt{K(A)}}\right)^n$$

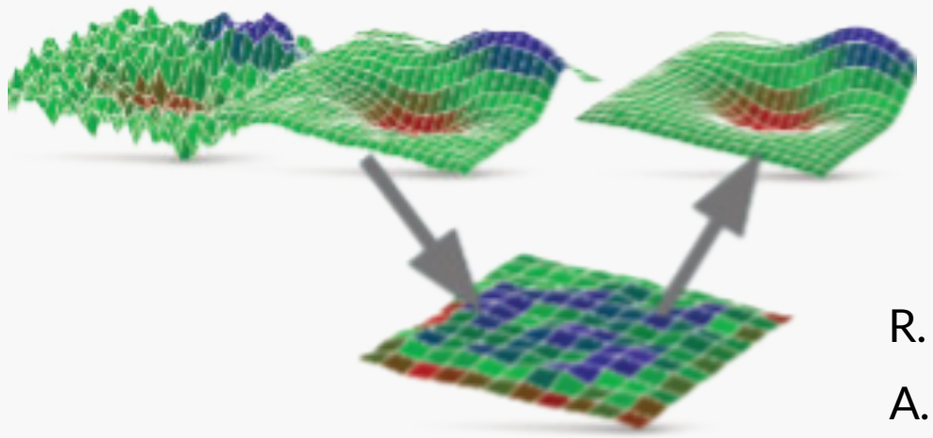
  - λ_{min} proportional to fermion mass
    → **critical slowing down**



Iterative solvers

Sparse matrix inversion via iterative solvers

- **Multigrid methods**
  - Precondition a Krylov subspace solver using a *coarse grid operator*



  - $\mathrm{M_c}=\mathrm{RMP}$: Coarse grid operator
  - R, P: *Restriction* & *Prolongation* ops

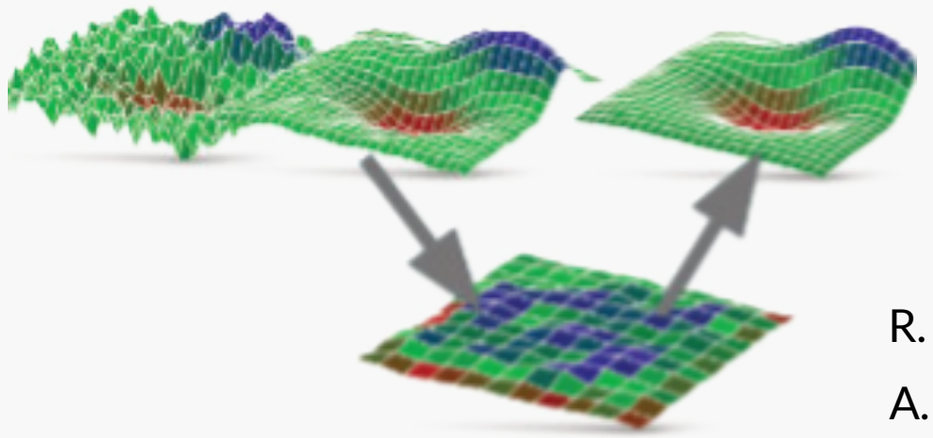
R. Babich *et al.* Phys. Rev. Lett. 105 (2010) 201602

A. Frommer *et al.* SIAM J. Sci. Comput. 36 (2014) A1581-A1608

Iterative solvers

Sparse matrix inversion via iterative solvers

- **Multigrid methods**
  - Precondition a Krylov subspace solver using a *coarse grid operator*



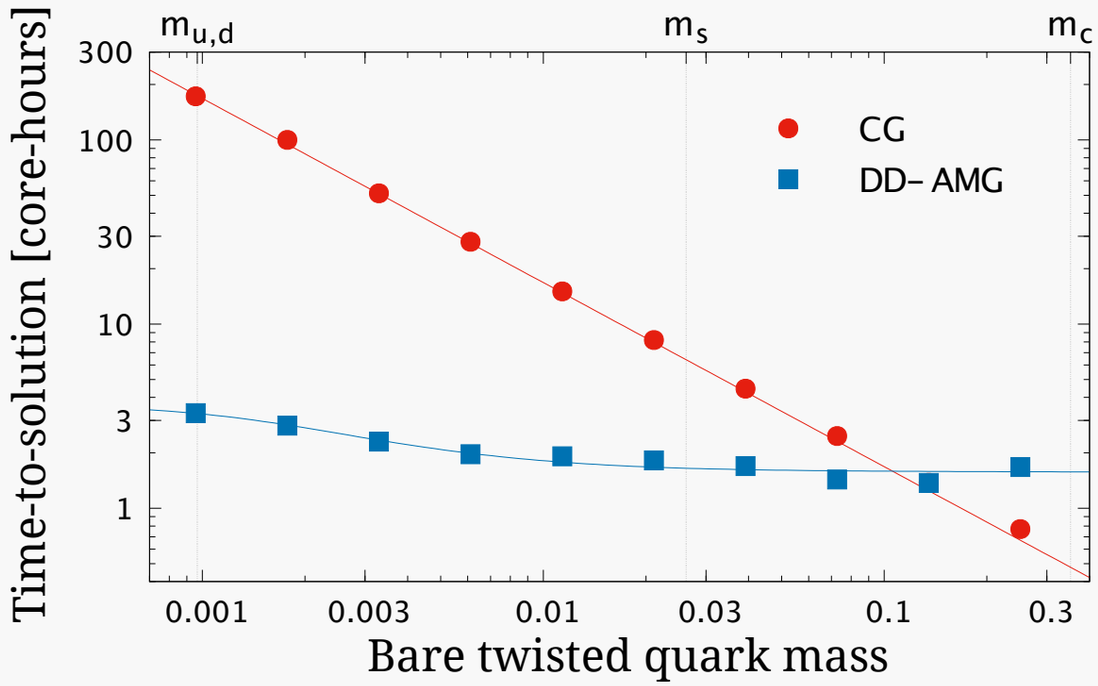
  - $M_c = RMP$: Coarse grid operator
  - R, P: *Restriction* & *Prolongation* ops

R. Babich *et al.* Phys. Rev. Lett. 105 (2010) 201602

A. Frommer *et al.* SIAM J. Sci. Comput. 36 (2014) A1581-A1608

  - Allows simulations directly at physical quark masses
  - ×100 improvement in time-to-solution
  - Requires set-up: building-up R and P. Minor for $m \rightarrow m_{u,d}$

- **Right:** for twisted mass fermions



C. Alexandrou *et al.* Phys. Rev. D 94 (2016) 11, 114509

Markov chain Monte Carlo

Monte Carlo integration

- Path integral formulation

$$\langle \mathcal{O} \rangle = \frac{1}{Z} \int \mathcal{D}[U]\mathcal{D}[\bar{\phi}]\mathcal{D}[\phi]\mathcal{O}e^{-S_G[U]-\bar{\phi}(M^+M)^{-1}\phi}$$

- Generate an ensemble of U with probability distribution $p(U)$:

$$p(U) = \frac{1}{Z} \int \mathcal{D}[\bar{\phi}]\mathcal{D}[\phi]e^{-S_G[U]-\bar{\phi}(M^+M)^{-1}\phi}$$

Markov chain Monte Carlo

Monte Carlo integration

- Path integral formulation

$$\langle \mathcal{O} \rangle = \frac{1}{Z} \int \mathcal{D}[U]\mathcal{D}[\bar{\phi}]\mathcal{D}[\phi]\mathcal{O}e^{-S_G[U]-\bar{\phi}(M^+M)^{-1}\phi}$$

- Generate an ensemble of U with probability distribution $p(U)$:

$$p(U) = \frac{1}{Z} \int \mathcal{D}[\bar{\phi}]\mathcal{D}[\phi]e^{-S_G[U]-\bar{\phi}(M^+M)^{-1}\phi}$$

- Markov chain Monte Carlo:
  - Update fields U according to some *transition probability* $T(U_k, U') : U_k \to U'$

Markov chain Monte Carlo

Monte Carlo integration

- Path integral formulation

$$\langle \mathcal{O} \rangle = \frac{1}{Z} \int \mathcal{D}[U]\mathcal{D}[\bar{\phi}]\mathcal{D}[\phi]\mathcal{O}e^{-S_G[U]-\bar{\phi}(M^+M)^{-1}\phi}$$

- Generate an ensemble of U with probability distribution $p(U)$:

$$p(U) = \frac{1}{Z} \int \mathcal{D}[\bar{\phi}]\mathcal{D}[\phi]e^{-S_G[U]-\bar{\phi}(M^+M)^{-1}\phi}$$

- Markov chain Monte Carlo:
  - Update fields U according to some *transition probability* $T(U_k, U') : U_k \to U'$
  - Our ensemble of fields $\{U\}$ will converge to $p(U)$ if:
    - $T^n(U,U') > 0$ for any U, U' for a finite n, i.e. starting from an arbitrary configuration U we can reach any configuration U' after a finite number n of steps: **Ergodicity**

Markov chain Monte Carlo

Monte Carlo integration

- Path integral formulation

$$\langle \mathcal{O} \rangle = \frac{1}{Z} \int \mathcal{D}[U]\mathcal{D}[\bar{\phi}]\mathcal{D}[\phi]\mathcal{O}e^{-S_G[U]-\bar{\phi}(M^+M)^{-1}\phi}$$

- Generate an ensemble of U with probability distribution $p(U)$:

$$p(U) = \frac{1}{Z} \int \mathcal{D}[\bar{\phi}]\mathcal{D}[\phi]e^{-S_G[U]-\bar{\phi}(M^+M)^{-1}\phi}$$

- Markov chain Monte Carlo:
  - Update fields U according to some *transition probability* $T(U_k, U') : U_k \to U'$
  - Our ensemble of fields $\{U\}$ will converge to $p(U)$ if:
    - $T^n(U,U') > 0$ for any U, U' for a finite n, i.e. starting from an arbitrary configuration U we can reach any configuration U' after a finite number n of steps: **Ergodicity**
    - $\int \mathcal{D}(U)p(U)T(U,U') = p(U')$, i.e. the transition probability is symmetric: **Balance**

Markov chain Monte Carlo

Metropolis sampling

1. Draw an *update proposal* u' from an arbitrary distribution $\mathrm{p}'(\mathrm{u})$

Markov chain Monte Carlo

Metropolis sampling

1. Draw an *update proposal* U' from an arbitrary distribution $p'(U)$
2. Accept U' as the next configuration in the Markov chain (U^{k+1}) with probability:

$$\min\left(1, \frac{p'(U^k)p(U')}{p(U^k)p'(U')}\right)$$

3. Otherwise: $U^{k+1}=U^k$

Markov chain Monte Carlo

Metropolis sampling

1. Draw an *update proposal* U' from an arbitrary distribution $p'(U)$
2. Accept U' as the next configuration in the Markov chain (U^{k+1}) with probability:

$$\min\left(1, \frac{p'(U^k)p(U')}{p(U^k)p'(U')}\right)$$

3. Otherwise: $U^{k+1}=U^k$

- Satisfies ergodicity and balance
- Allows drawing from an arbitrary distribution $p'(U)$, e.g. normal or uniform
- Requires calculating: $p(U_k)/p(U')$, i.e. Z cancels

Markov chain Monte Carlo

Metropolis sampling

1. Draw an *update proposal* U' from an arbitrary distribution $p'(U)$
2. Accept U' as the next configuration in the Markov chain (U^{k+1}) with probability:

$$\min\left(1, \frac{p'(U^k)p(U')}{p(U^k)p'(U')}\right)$$

3. Otherwise: $U^{k+1}=U^k$

- Satisfies ergodicity and balance
- Allows drawing from an arbitrary distribution $p'(U)$, e.g. normal or uniform
- Requires calculating: $p(U_k)/p(U')$, i.e. Z cancels

- Since the configurations U are distributed according to the desired $p(U)$:

$$\langle \mathcal{O} \rangle = \frac{1}{N_{stat}} \sum_{i=1}^{N_{stat}} \mathcal{O}(U^i)$$

and statistical errors scale like $\frac{1}{\sqrt{N_{stat}}}$

Markov chain Monte Carlo

Metropolis sampling in Markov chain Monte Carlo

- *Acceptance rate*:
  - Ratio of accepted trials over total number of configurations
  - Usually can be tuneable

Markov chain Monte Carlo

Metropolis sampling in Markov chain Monte Carlo

- *Acceptance rate*:
  - Ratio of accepted trials over total number of configurations
  - Usually can be tuneable
- *Autocorrelation* $\rho(\tau)$:
  - Probability of having τ rejections in a row, with $\rho(0) = 1$
  - *Autocorrelation time* is loosely the value of τ for which $\rho(\tau) = 0$
  - More formally: $\tau_{\text{int}} = \frac{1}{2} + \sum_{\tau=1}^{\infty} \rho(\tau)$

Markov chain Monte Carlo

Metropolis sampling in Markov chain Monte Carlo

- *Acceptance rate*:
  - Ratio of accepted trials over total number of configurations
  - Usually can be tuneable
- *Autocorrelation* $\rho(\tau)$:
  - Probability of having τ rejections in a row, with $\rho(0) = 1$
  - *Autocorrelation time* is loosely the value of τ for which $\rho(\tau) = 0$
  - More formally: $\tau_{\text{int}} = \frac{1}{2} + \sum_{\tau=1}^{\infty} \rho(\tau)$
- *Critical slowing down*:
  - The divergence of τ_{int} as some parameters of the theory approach their critical value, e.g. as we approach a phase transition

Markov chain Monte Carlo

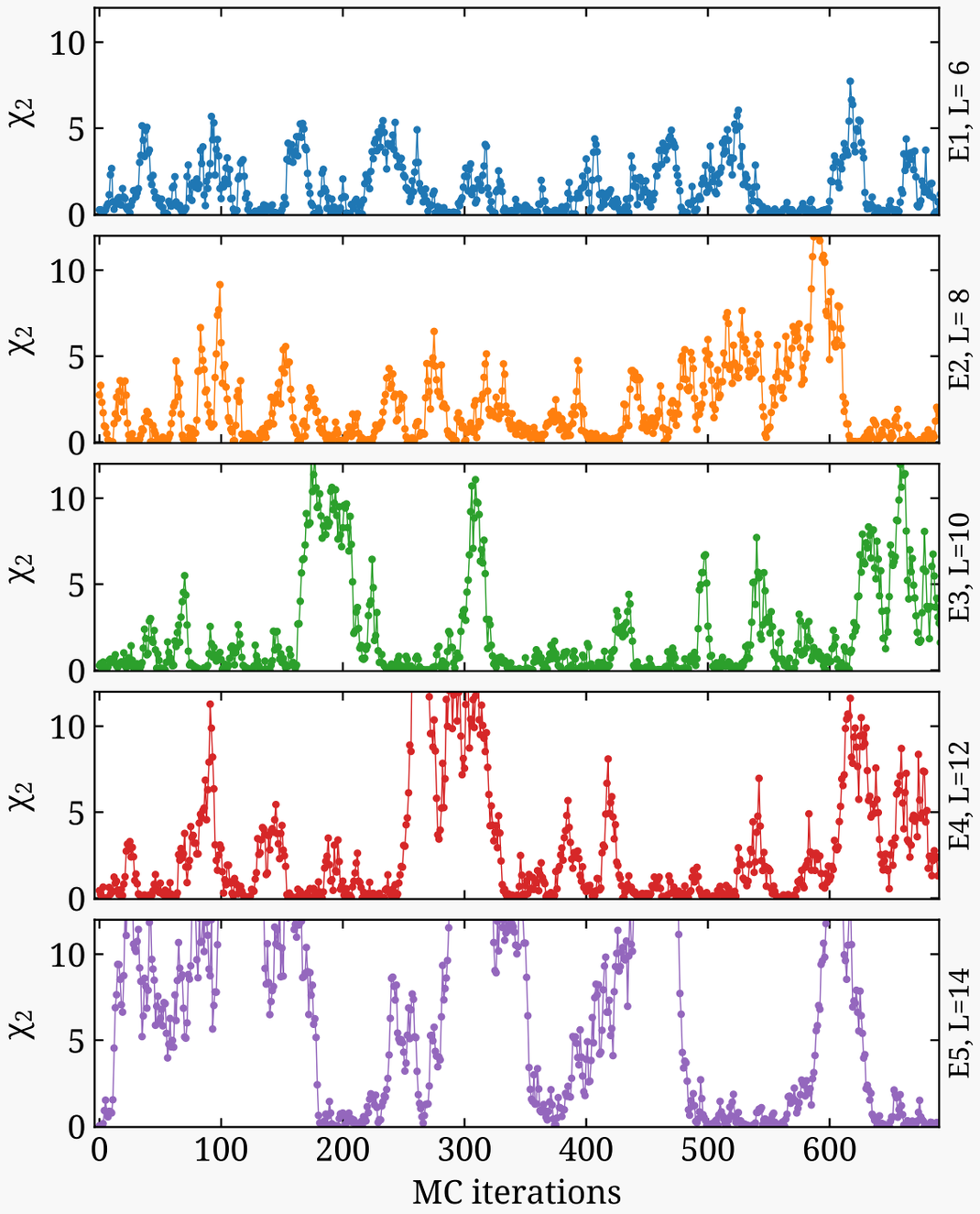
Metropolis sampling in Markov chain Monte Carlo

- *Acceptance rate*:
  - Ratio of accepted trials over total number of configurations
  - Usually can be tuneable
- *Autocorrelation* $\rho(\tau)$:
  - Probability of having τ rejections in a row, with $\rho(0) = 1$
  - *Autocorrelation time* is loosely the value of τ for which $\rho(\tau) = 0$
  - More formally: $\tau_{\text{int}} = \frac{1}{2} + \sum_{\tau=1}^{\infty} \rho(\tau)$
- *Critical slowing down*:
  - The divergence of τ_{int} as some parameters of the theory approach their critical value, e.g. as we approach a phase transition

Largely depend on the choice of $p'(U)$, and how closely it approximates $p(U)$

Markov chain Monte Carlo

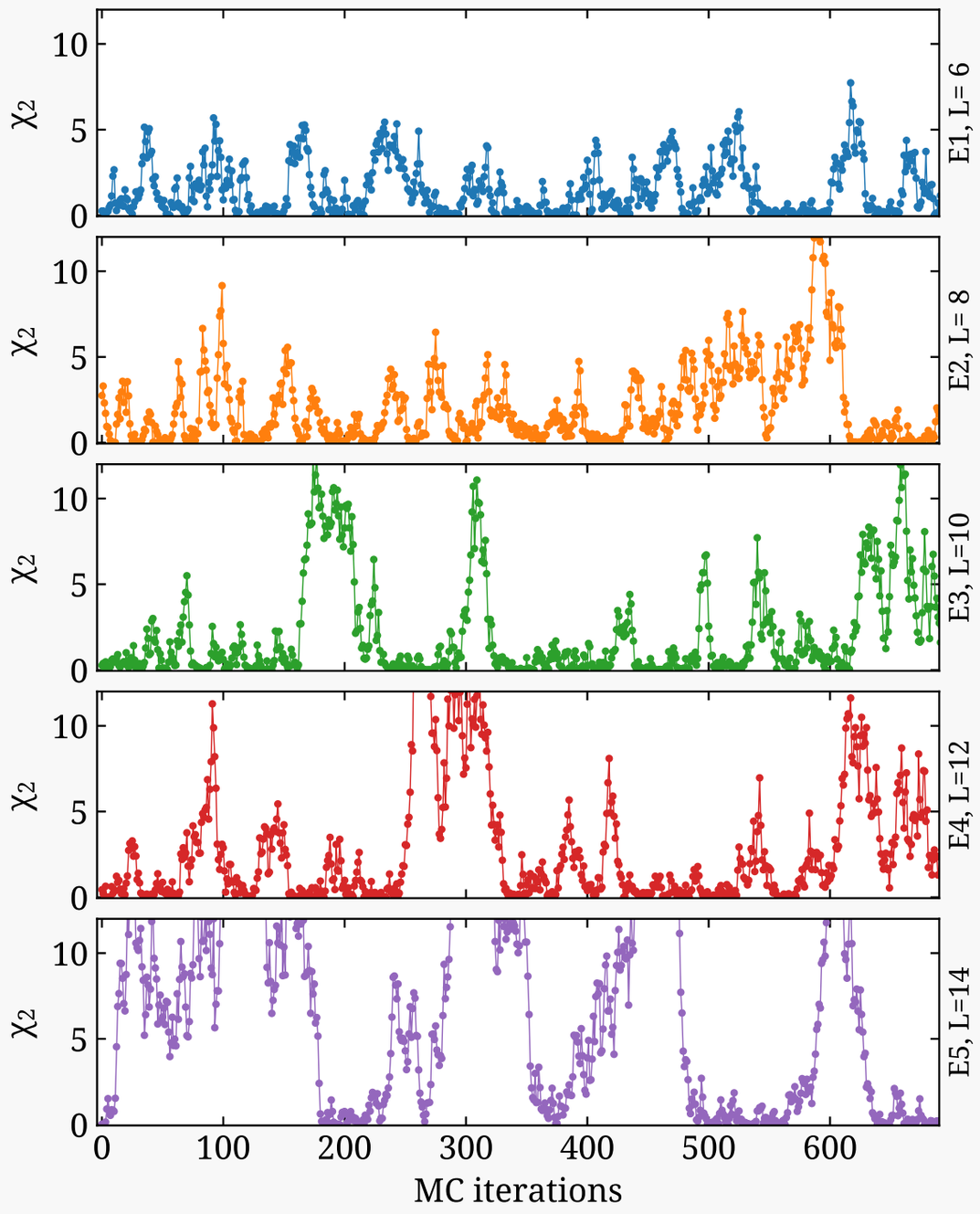
Example of critical slowing down



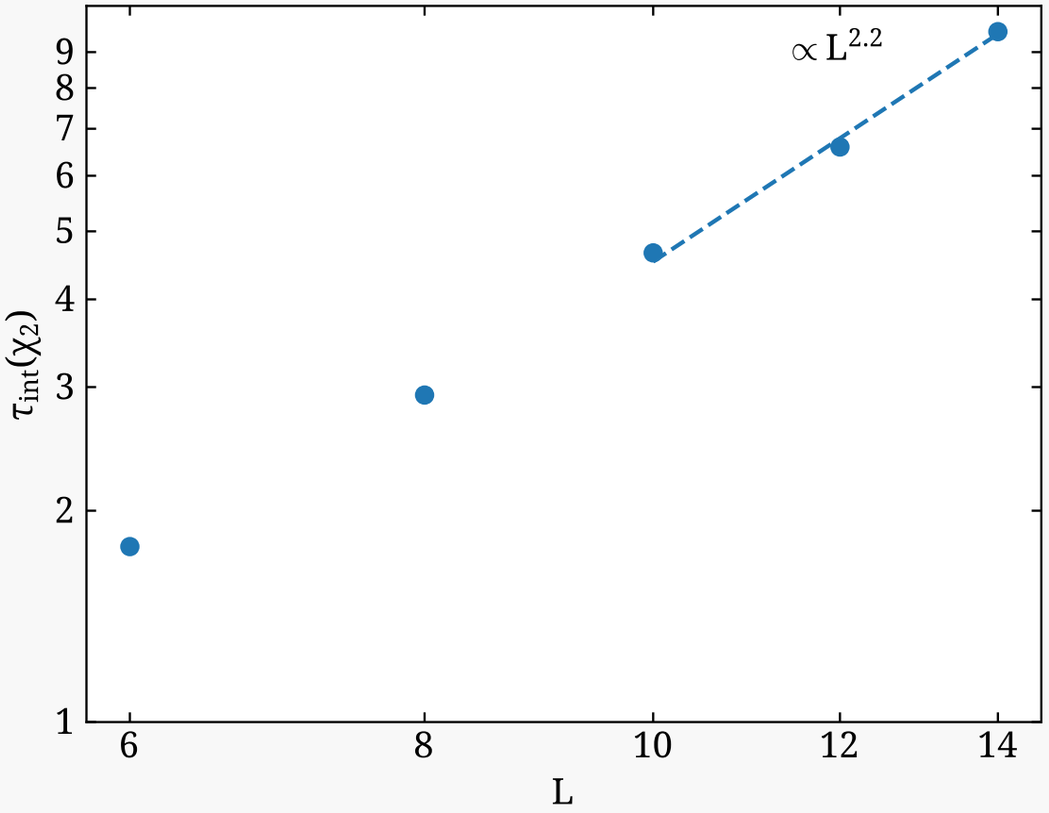
- Observable is the *two-point susceptibility* in the ϕ^4 model
- Criticality is approached from top to bottom, as the volume is increased

Markov chain Monte Carlo

Example of critical slowing down



- Observable is the *two-point susceptibility* in the ϕ^4 model
- Criticality is approached from top to bottom, as the volume is increased



Hybrid Monte Carlo

- *Hybrid Monte Carlo (HMC):* algorithm for updating U motivated by molecular dynamics

$$\langle \mathcal{O} \rangle = \frac{1}{Z} \int \mathcal{D}[U, \bar{\phi}, \phi] \mathcal{O} e^{-S} = \frac{1}{Z} \int \mathcal{D}[U, \bar{\phi}, \phi, \bar{\pi}, \pi] \mathcal{O} e^{-\frac{\pi^2}{2} - S}$$

Hybrid Monte Carlo

- *Hybrid Monte Carlo (HMC):* algorithm for updating U motivated by molecular dynamics

$$\langle \mathcal{O} \rangle = \frac{1}{Z} \int \mathcal{D}[U, \bar{\phi}, \phi] \mathcal{O} e^{-S} = \frac{1}{Z} \int \mathcal{D}[U, \bar{\phi}, \phi, \bar{\pi}, \pi] \mathcal{O} e^{-\frac{\pi^2}{2} - S}$$

- π: pseudo-momenta *conjugate* to the gauge-fields U

$$H = \frac{\pi^2}{2} + S(U)$$

Hybrid Monte Carlo

- *Hybrid Monte Carlo (HMC):* algorithm for updating U motivated by molecular dynamics

$$\langle \mathcal{O} \rangle = \frac{1}{Z} \int \mathcal{D}[U, \bar{\phi}, \phi] \mathcal{O} e^{-S} = \frac{1}{Z} \int \mathcal{D}[U, \bar{\phi}, \phi, \bar{\pi}, \pi] \mathcal{O} e^{-\frac{\pi^2}{2} - S}$$

- π: pseudo-momenta *conjugate* to the gauge-fields U

$$H = \frac{\pi^2}{2} + S(U)$$

— Evolve U according to Hamilton's equation to propose a new U'

$$\frac{d}{d\tau}\pi = -\frac{\partial}{\partial U} H = -\frac{\partial}{\partial U} S$$

$$\frac{d}{d\tau} U = \frac{\partial}{\partial \pi} H = \pi$$

Hybrid Monte Carlo

- *Hybrid Monte Carlo (HMC):* algorithm for updating U motivated by molecular dynamics

$$\langle \mathcal{O} \rangle = \frac{1}{Z} \int \mathcal{D}[U, \bar{\phi}, \phi] \mathcal{O} e^{-S} = \frac{1}{Z} \int \mathcal{D}[U, \bar{\phi}, \phi, \bar{\pi}, \pi] \mathcal{O} e^{-\frac{\pi^2}{2} - S}$$

- π: pseudo-momenta *conjugate* to the gauge-fields U

$$H = \frac{\pi^2}{2} + S(U)$$

— Evolve U according to Hamilton's equation to propose a new U'

$$\frac{d}{d\tau}\pi = -\frac{\partial}{\partial U} H = -\frac{\partial}{\partial U} S$$

$$\frac{d}{d\tau} U = \frac{\partial}{\partial \pi} H = \pi$$

E.g. Leap-frog integration

$$\begin{aligned}
\pi(\delta\tau/2) &= \pi(0) - \frac{\delta\tau}{2} \frac{\partial}{\partial U} S|_{U(\tau=0)} \\
U(\delta\tau) &= U(0) + \delta\tau \pi(\delta\tau/2) \\
\pi(3\delta\tau/2) &= \pi(\delta\tau/2) - \delta\tau \frac{\partial}{\partial U} S|_{U(\tau=\delta\tau)} \\
&\vdots \\
\pi(T) &= \pi(T - \delta\tau/2) - \frac{\delta\tau}{2} \frac{\partial}{\partial U} S|_{U(\tau=T-\delta\tau)}
\end{aligned}$$

S. Duane, Kennedy, Pendleton, Roweth, Phys. Lett. B195, 216 (1987)

Hybrid Monte Carlo

- An example of a *global updating* algorithm
- Introduction of fictitious time τ and time-step $\delta\tau = T/N$
- Calculation of "forces" requires inversion for each time-step:

$$\pi(n\delta\tau/2) = \pi((n-1)\delta\tau/2) - \delta\tau \frac{\partial}{\partial U} S|_{U(\tau = n\delta\tau)}$$

Hybrid Monte Carlo

- An example of a *global updating* algorithm
- Introduction of fictitious time τ and time-step $\delta\tau = T/N$
- Calculation of "forces" requires inversion for each time-step:

$$\pi(n\delta\tau/2) = \pi((n-1)\delta\tau/2) - \delta\tau \frac{\partial}{\partial U} S|_{U(\tau = n\delta\tau)}$$

- Numerical integration → error in Hamiltonian $H(\tau=0) - H(\tau=T)$

Hybrid Monte Carlo

- An example of a *global updating* algorithm
- Introduction of fictitious time τ and time-step $\delta\tau = T/N$
- Calculation of "forces" requires inversion for each time-step:

$$\pi(n\delta\tau/2) = \pi((n-1)\delta\tau/2) - \delta\tau \frac{\partial}{\partial U} S|_{U(\tau = n\delta\tau)}$$

- Numerical integration → error in Hamiltonian $H(\tau=0) - H(\tau=T)$

Correct with Metropolis accept/reject:

$$\min\left(1, \frac{p(U(T))}{p(U(0))}\right)$$

Hybrid Monte Carlo

- An example of a *global updating* algorithm
- Introduction of fictitious time τ and time-step $\delta\tau = T/N$
- Calculation of "forces" requires inversion for each time-step:

$$\pi(n\delta\tau/2) = \pi((n-1)\delta\tau/2) - \delta\tau \frac{\partial}{\partial U} S|_{U(\tau = n\delta\tau)}$$

- Numerical integration → error in Hamiltonian $H(\tau = 0) - H(\tau = T)$

  Correct with Metropolis accept/reject:

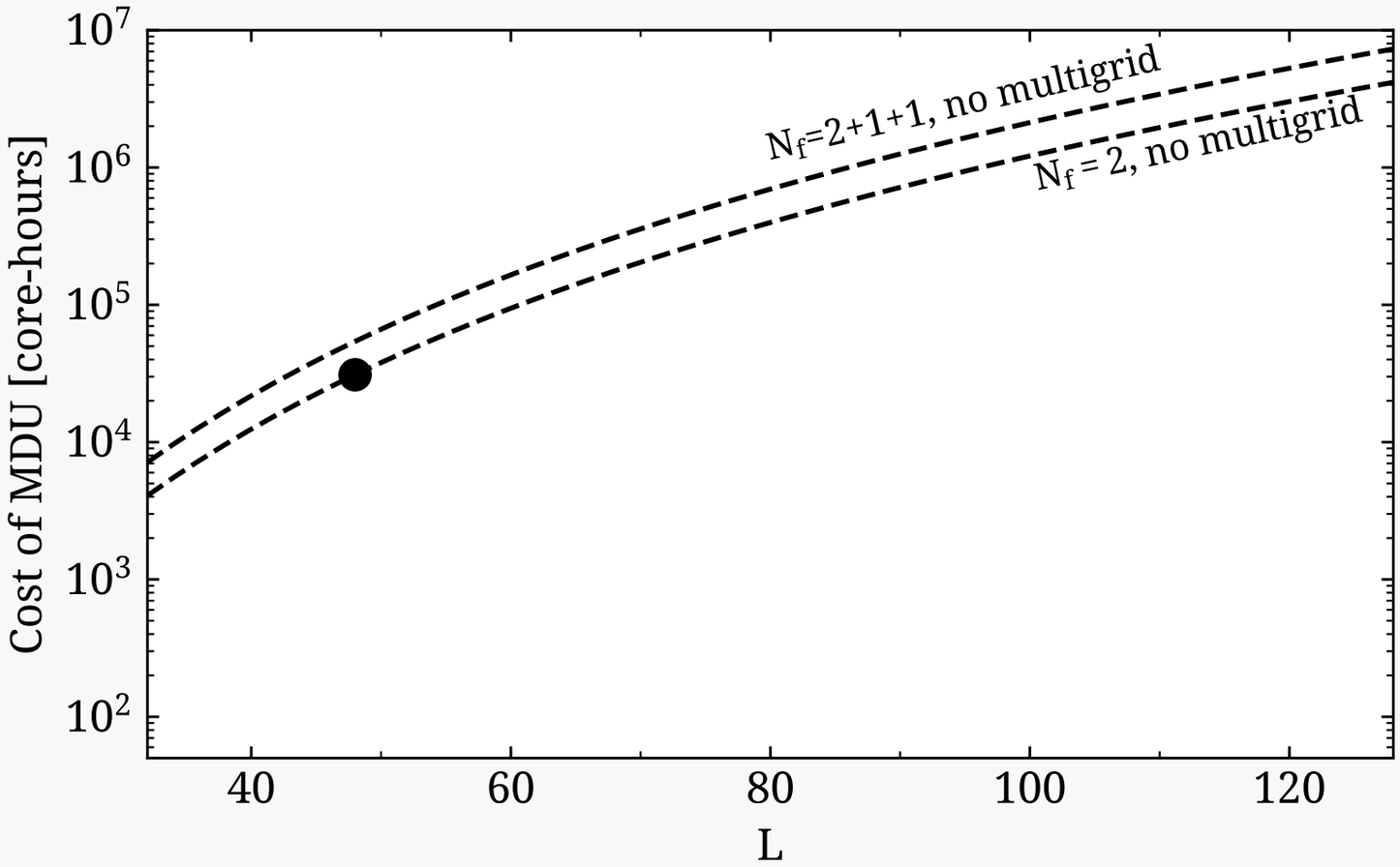
$$\min\left(1, \frac{p(U(T))}{p(U(0))}\right)$$

- In practice:
  - Multiple time-step integration (e.g.: `hep-lat/0506011`)
  - Higher order integration schemes (e.g.: `arXiv:1210.6600 [hep-lat]`)
  - Hasenbusch mass preconditioning (`hep-lat/0211042`)

Lattice QCD — Large-scale simulations

A combination of

- Multi-grid solver improvements
- Integration schemes
- Computers becoming faster

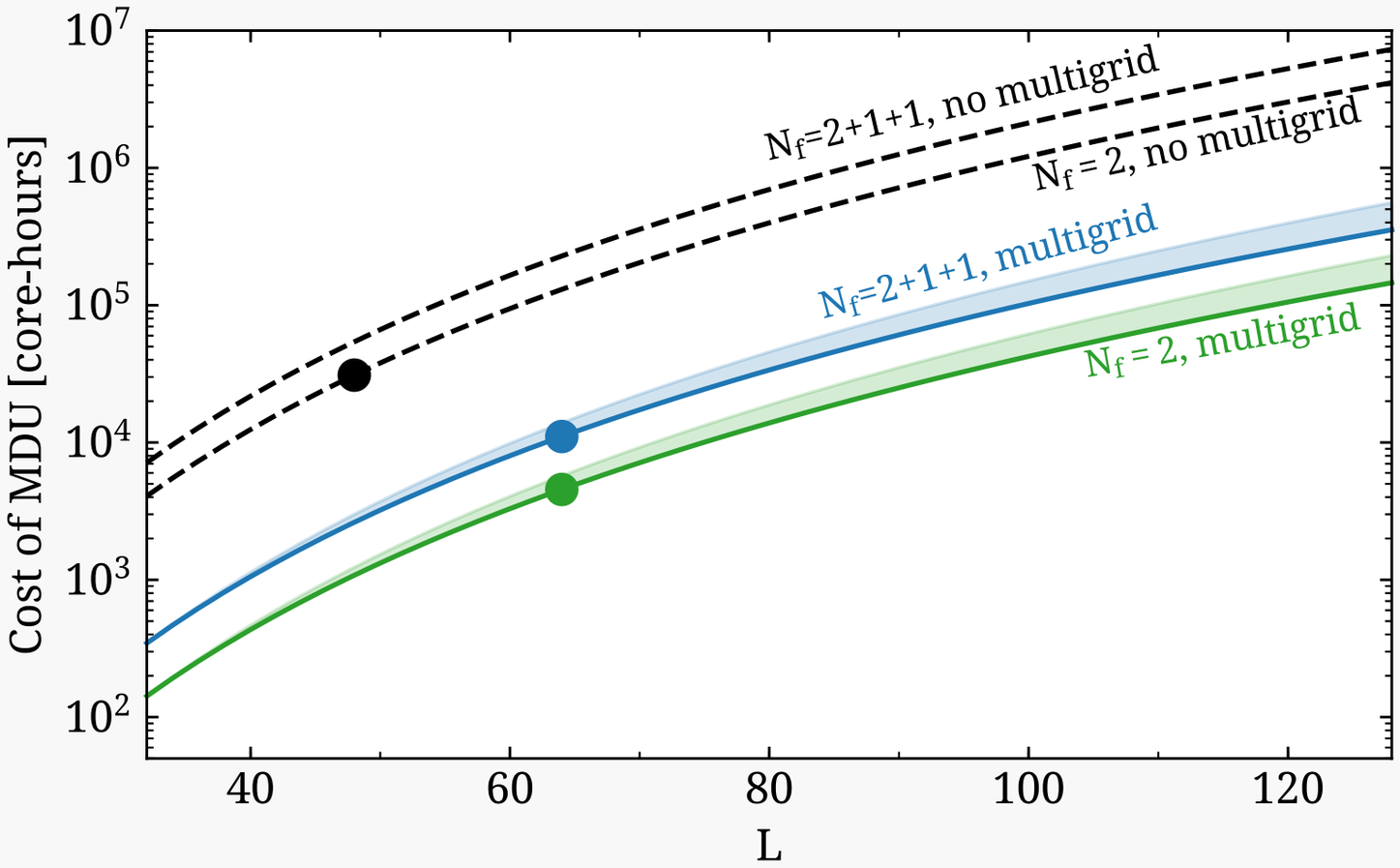


- 2015: baseline, JuQueen

Lattice QCD — Large-scale simulations

A combination of

- Multi-grid solver improvements
- Integration schemes
- Computers becoming faster

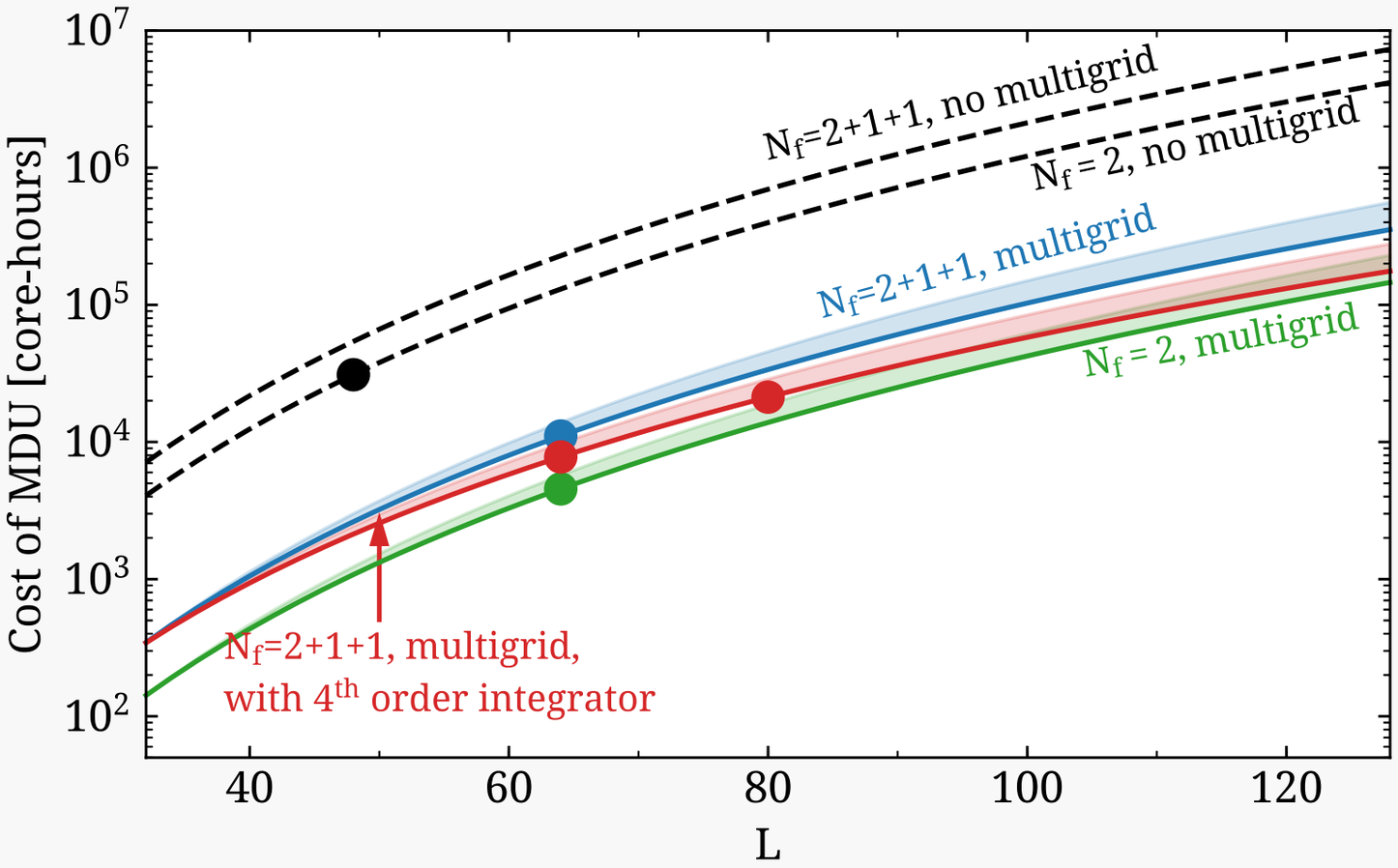


- 2015: baseline, JuQueen
- 2016: Added multi-grid to N_f=2
- 2017: Added multi-grid to N_f=2+1+1

Lattice QCD — Large-scale simulations

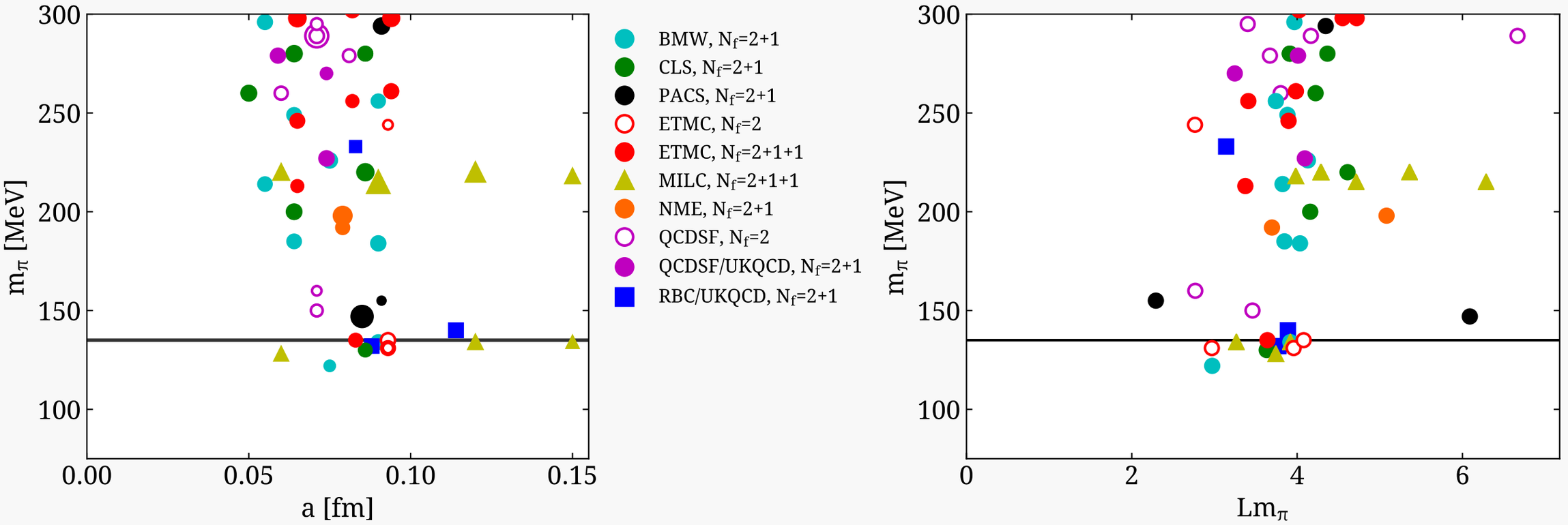
A combination of

- Multi-grid solver improvements
- Integration schemes
- Computers becoming faster



- 2015: baseline, JuQueen
- 2016: Added multi-grid to N_f=2
- 2017: Added multi-grid to N_f=2+1+1
- 2018: Added 4th-order integration scheme

Simulation landscape



Selected lattice simulation points from various collaborations

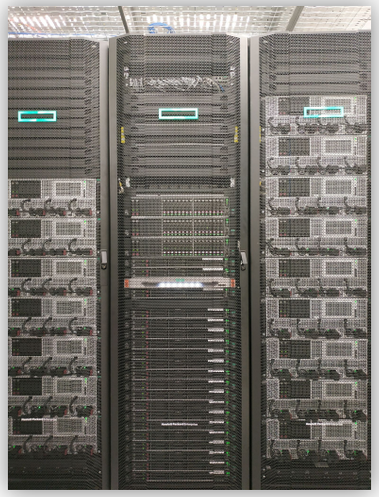
- Multiple collaborations simulating at physical pion mass
- Size of points indicates $m_\pi L$

Acknowledgements

Computer time used to produce some of the data shown

Marconi100, CINECA
PRACE Tier-0 project

SuperMUC-NG, LRZ
Gauss Large Scale project



Cyclone, CyI
Local project

Funding

Project “NextQCD”, Cyprus
Research and Innovation Foundation