

# 2D $\phi^4$ theory: A study in OpenACC with PGI++

Dean Howarth

July 8, 2019

# Table of contents

- 1 Introduction
- 2 2D  $\phi^4$  theory and its discrete analogue (Lecture)
- 3 2D  $\phi^4$  with PGI++ (Code Review)

# Coming up...

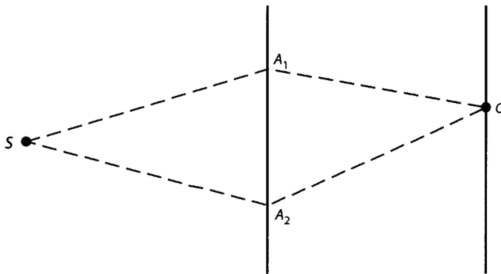
- 1 We need a concrete physical model to use: 2D  $\phi^4$
- 2 Within the 2D  $\phi^4$  code are some important algorithms (HMC, cluster decomposition)
- 3 Identify opportunities for parallelism
- 4 Use OpenACC and pgi++ to exploit those opportunities.
- 5 Inspect our optimisations using profiling tools, gain insights into further progress
- 6 Apply all of the above to an different code.

# Why study 2D $\phi^4$ theory?

- ❶ We know all the answers! Same universality class as 2D Ising model
- ❷ Contains most of the ingredients required for computational physics problems
  - ❶ Boundary conditions (periodic)
  - ❷ A differential operator
  - ❸ Markov chain dynamics
- ❸ We don't actually know all the answers... (Google "2D  $\phi^4$  " for recent results)
  - ❶ Long range coupling still an ongoing debate
  - ❷ Add one more dimension, you're in unknown territory

# Path Integrals: A lightning tour

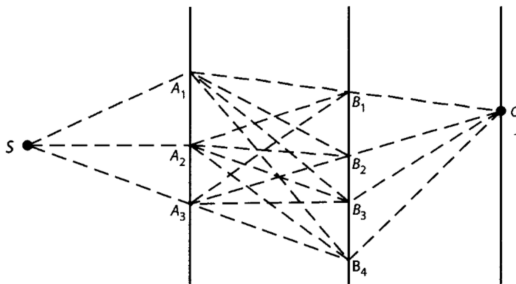
Double slit experiment..



$$P(S \rightarrow O) = |\psi_{A1} + \psi_{A2}|^2 = |\psi_{A1}|^2 + |\psi_{A2}|^2 + 2|\psi_{A1}||\psi_{A2}| \cdot (\text{phase})$$

# Path Integrals: A lightning tour

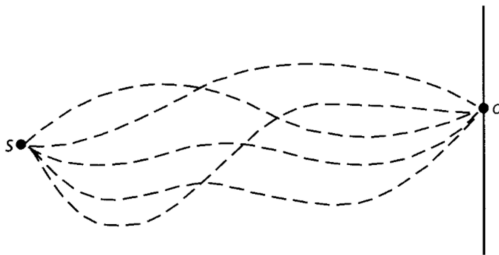
Multiple slits ( $n, m, l \dots$ ) in multiple panels ( $A, B, C, \dots$ ).



$$P(S \rightarrow O) = |(\psi_{A1} + \dots + \psi_{An}) \cdot (\psi_{B1} + \dots + \psi_{Bm}) \dots|^2 = \dots$$

# Path Integrals: A lightning tour

Let the number of slits and panels go to infinity!



$P(S \rightarrow O) = \dots$  we need a path integral formulation.

# Path Integrals: A lightning tour

Each path contributes an amplitude (weight) to the final answer. The further away from the classical path, the less weight is given to the path. These weights are given by a path integral.

$$\langle \phi^2 \rangle = \frac{\int D\phi \exp(iS[\phi]) \phi^2}{\int D\phi \exp(iS[\phi])}$$

where

$$S(\phi) = \int dx^2 \mathcal{L}(x)$$

$$\mathcal{L}(x) = \frac{1}{2}(\nabla \phi(x))^2 - \frac{1}{2}m^2 \phi^2(x) - \frac{1}{4!} \lambda \phi^4(x) \quad x = (x, t)$$



# Markov Chains

What happens when one can't solve the path integral? How does one simulate a quantum field theory? Probabilistically.

Quantum fields are superpositions of classical field configurations. If we can identify a representative sample of those configurations, we will be able to compute a good estimate of an expectation value.

**This is in perfect analogy with statistical sampling: a robust estimate of the mean of a data set can be obtained via an unbiased sample from that data set.**

Markov chains offer us the ability to create such a sample. But first, we must discretise.

# Discretising the Lagrangian

The continuum 2D  $\phi^4$  **Minkowski** Lagrangian

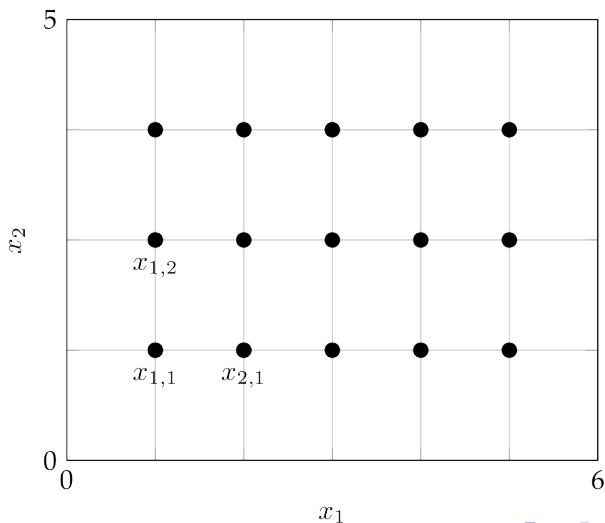
$$\mathcal{L}(x) = \frac{1}{2}(\nabla\phi(x))^2 - \frac{1}{2}m^2\phi^2(x) - \frac{1}{4!}\lambda\phi^4(x) \quad x = (x, t) \quad (1)$$

The discrete 2D  $\phi^4$  **Euclidean** Lagrangian

$$\begin{aligned} \mathcal{L}_E(x) = & \frac{1}{2}[(\phi(x + an_x) - \phi(x))^2 + (\phi(x + an_y) - \phi(x))^2] \\ & + \frac{1}{2}m^2\phi^2(x) + \frac{1}{4!}\lambda\phi^4(x) \quad x = (x, y) \end{aligned} \quad (2)$$

Discretisation of the differential operator introduces errors. Can you work out what those errors are? (see me during the break)

# Discretising the Lagrangian



# Hybrid Monte-Carlo and Importance sampling

If you Google "Hybrid Monte-Carlo and Importance sampling" you will be occupied for weeks! For now, you only need to understand the concepts.

- Wick Rotation: rotate the  $t$ -dimension to  $-it$

$$\langle \phi^2 \rangle_E = \frac{\int D\phi \exp(-S_E[\phi]) \phi^2}{\int D\phi \exp(-S_E[\phi])}$$

- Fictitious momenta: noise with zero mean

$$\langle \phi^2 \rangle_E = \frac{\int D\phi \exp(-\frac{1}{2}\pi^2 - S_E[\phi]) \phi^2}{\int D\phi \exp(-\frac{1}{2}\pi^2 - S_E[\phi])}$$

- Fictitious hamiltonian: field evolution through an extra dimension

$$\mathcal{H}_E(P, \phi) = \frac{1}{2}\pi^2 + S_E[\phi]$$

# Hybrid Monte-Carlo and Importance sampling

We evolve the  $\phi$  field through a fictitious time variable  $\tau$  using the Hamiltonian equations of motion.

$$\frac{\partial \phi}{\partial \tau} = \frac{\partial \mathcal{H}}{\partial \pi}, \quad \frac{\partial \pi}{\partial \tau} = -\frac{\partial \mathcal{H}}{\partial \phi} = -\frac{\partial \mathcal{S}}{\partial \phi} \quad (3)$$

We can solve these equations (evolve the  $\phi$  field) using Leapfrog algorithm (See `phi4Serial.cpp` and Stefan's notes). We can explore the phase space further using cluster algorithms. (See `WolffCluster.pdf` for more details)

# The HMC algorithm for $\phi^4$ theory

Here is the HMC, with annotations (search for 'Step n:') directing you to the relevant code in phi4Serial.cpp

- Step1: choose gaussian distributed momenta  
 $P_i \in \exp(-\pi_i^2/2)$
- Step2: evolve the  $\phi$  and  $\pi$  fields to candidate configurations  
 $\phi'$  and  $\pi'$ :

$$\begin{aligned}\frac{d}{d\tau}\phi_x(\tau) &= \frac{\partial}{\partial\pi_x}\mathcal{H}(\pi(\tau),\phi(\tau)) \\ \frac{d}{d\tau}\pi_x(\tau) &= -\frac{\partial}{\partial\phi_x}\mathcal{H}(\pi(\tau),\phi(\tau))\end{aligned}$$

- Step3: accept candidate configurations with probability

$$P_{\text{accept}} = \min(1, \exp(-(\mathcal{H}' - \mathcal{H})))$$

# The HMC algorithm for $\phi^4$ theory

Step 2 in a bit more detail. (search for 'Step 2.n:')

- Step 2.1:  $\pi_{1/2} = \pi_0 - \frac{d\tau}{2} F(\phi)$
- Step 2.2: For  $k = 1 \dots n - 1$

$$\phi_k = \phi_{k-1} + \pi_{k-1/2} d\tau$$

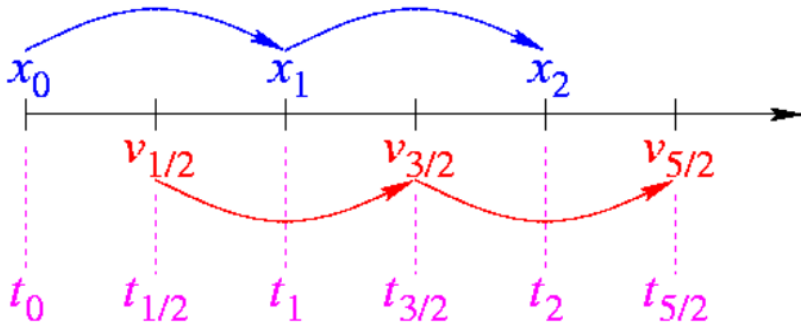
$$\pi_{k+\frac{1}{2}} = \pi_{k-\frac{1}{2}} - F(\phi) d\tau$$

- Step 2.3:  $\phi_n = \phi_{n-1} + \pi_{n-\frac{1}{2}} d\tau$

$F(\phi) = \nabla S(\phi)$  is a fictitious force, driven by the 'action potential'.

# The HMC algorithm for $\phi^4$ theory

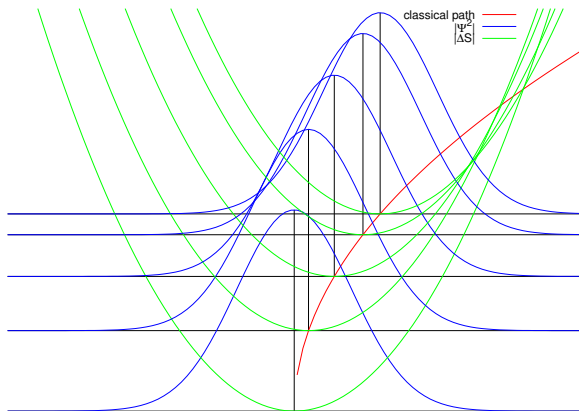
This is why the velocity Verlet algorithm is often called Leapfrog...





# The HMC algorithm for $\phi^4$ theory

This is why we call it the 'action potential'...



# HMC is its own thang



# Cluster Algorithms

Cluster decomposition of  $\phi^4$  theory can lead to some staggeringly efficient evolution algorithms. It was pioneered by Fortuin and Kasteleyn in the late 60s. Omitting the gory details, one can write the partition function of a lattice theory as the sum of all possible clusters. One constructs such clusters with a probability,

$$P[bond(x, y)] = 1 - \exp(-2\phi(x)\phi(y)) \quad (4)$$

Swendsen and Wang used this result to produce an algorithm that, without rejection, moves a lattice from one configuration to another. Ulli Wolff came up with one too.

# Swendsen-Wang And Wolff

Swendsen-Wang:

- Step 1: Test every n.n. bond ONCE to see if they are connected via a bond using
$$P[\text{bond}(x, y)] = 1 - \exp(-2\phi(x)\phi(y))$$
- Step 2: Identify every connected cluster on the lattice.
- Step 3: Flip each cluster with 50% probability.

Wolff:

- Step 1: Choose a lattice site at random.
- Step 2: Identify every site connected to that original site using
$$P[\text{bond}(x, y)] = 1 - \exp(-2\phi(x)\phi(y))$$
- Step 3: Flip the cluster with 100% probability.

You have been provided with notes on the Wolff algorithm, and code for the Swendsen-Wang. Can you write a Wolff cluster algorithm in the activity time?

# The serial implementation

Provided under

RPI-Advanced-

Cyberinfrastructure/phi4AndOpenACC/phi4Serial

is a set of routines that perform HMC on a  $\phi^4$  theory. Let's read through the code and understand it before using PGI++.

Points to consider:

- HMC is performed once for multiple cluster updates... why?
- Can you decipher what the cluster algorithm does?
- The trajectory() function performs the Leapfrog integration. Can you see how it does it?
- Looking ahead, can you intuitively see where the routines can be easily parallelised?

# What are OpenACC and PGI++?

PGI++ is an OpenACC compiler. It is able to compile C++ standard code, along with OpenACC compliant pragmas:

```
// Copy lattice field (Device)
template<typename T> inline void copyField(T phi2[L][L], T phi1[L][L]) {
{
#pragma acc parallel loop collapse(2)
    for(int x=0; x<L;x++)
        for(int y=0; y<L;y++)
            phi2[x][y] = phi1[x][y];
    }
}

// Copy lattice field (Host)
template<typename T> inline void copyField_host(T phi2[L][L], T phi1[L][L]) {

    for(int x=0; x<L;x++)
        for(int y=0; y<L;y++)
            phi2[x][y] = phi1[x][y];
}
```

A pragma is a 'hint' or 'directive' to the compiler. In the above excerpt, we have informed the compiler that the (2) nested loop may be collapsed. I.e., each iteration is independent.

# Using OpenACC and PGI++

We are going to learn OpenACC by example. Having seen and understood some serial code, we shall add OpenACC pragmas where relevant and explain each addition in context.

- v0: The serial code, with timings.
- v1: Lots of OpenACC additions to the HMC.
- v2: More OpenACC in the cluster decomposition.

## ACTIVITY CHALLENGES

- v3: Can you write a Wolff algorithm? Can you make the HMC more efficient? Can you write a two-point correlation function measurement  $\langle \phi(x)\phi(y) \rangle$  and make it run at nosebleed speed?

Ask questions, work in groups, read the code comments,  
happy hacking!