Moving a Mountain

Association for Women in Mathematics

Claire Murphy

UC Santa Barbara

February 25, 2026

My Mathematical Journey

1. Started my undergraduate career at Santa Clara University in the computer science department.

Figure: SCU and Intel HQ are both located in the South Bay Area.

My Mathematical Journey

1. Started my undergraduate career at Santa Clara University in the computer science department.
2. Decided to add a math major because I'd always enjoyed math.

Figure: SCU and Intel HQ are both located in the South Bay Area.

My Mathematical Journey

1. Started my undergraduate career at Santa Clara University in the computer science department.
2. Decided to add a math major because I'd always enjoyed math.
3. Interned at Intel $\Rightarrow$ realized I did NOT want to be a software engineer.

Figure: SCU and Intel HQ are both located in the South Bay Area.

My Mathematical Journey

1. Started my undergraduate career at Santa Clara University in the computer science department.
2. Decided to add a math major because I'd always enjoyed math.
3. Interned at Intel $\Rightarrow$ realized I did NOT want to be a software engineer.
4. Took SCU's version of MATH 118 $\Rightarrow$ got more serious about math, realized I wanted to become a mathematician.

Figure: SCU and Intel HQ are both located in the South Bay Area.

Where I Am Now

I'm now in my 4th year at UCSB, researching particle methods with Dr. Katy Craig.

Figure: Dr. Craig and my cats.

Where I Am Now

I'm now in my 4th year at UCSB, researching particle methods with Dr. Katy Craig.

I serve as vice president of the UCSB Scientific, Industrial, and Applied mathematics seminar.

Figure: Dr. Craig and my cats.

Where I Am Now

I'm now in my 4th year at UCSB, researching particle methods with Dr. Katy Craig.

I serve as vice president of the UCSB Scientific, Industrial, and Applied mathematics seminar.

I'm passionate about teaching, and this summer I'm planning to teach at a STEM camp for middle school girls.

Figure: Dr. Craig and my cats.

Where I Am Now

I'm now in my 4th year at UCSB, researching particle methods with Dr. Katy Craig.

I serve as vice president of the UCSB Scientific, Industrial, and Applied mathematics seminar.

I'm passionate about teaching, and this summer I'm planning to teach at a STEM camp for middle school girls.

I have two kittens who are my pride and joy!

Figure: Dr. Craig and my cats.

Motivation

Figure: Mammoth mountain in the Eastern Sierra Nevada mountains of California.

Mammoth mountain will experience an avalanche in two days. How do we predict how the mountain will settle?

Particle Methods

Idea: Split the mountain into N small "chunks," or particles. Predict where each particle will land.

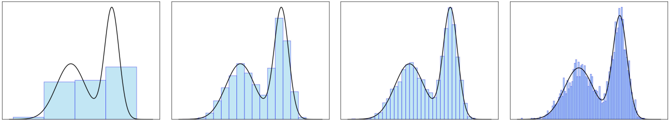

Figure: Modeling a 1-dimensional version of Mammoth mountain with more and more chunks.

Particle Methods

Idea: Split the mountain into N small "chunks," or particles. Predict where each particle will land.

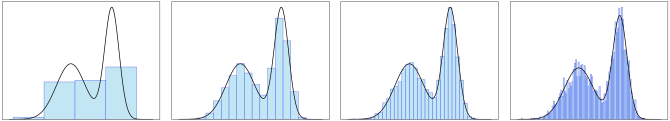

Figure: Modeling a 1-dimensional version of Mammoth mountain with more and more chunks.

Intuitively, as we model the mountain with more and more particles, our prediction will get more and more accurate.

Particle Methods

Idea: Split the mountain into N small "chunks," or particles. Predict where each particle will land.

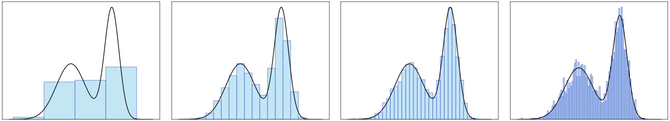

Figure: Modeling a 1-dimensional version of Mammoth mountain with more and more chunks.

Intuitively, as we model the mountain with more and more particles, our prediction will get more and more accurate.

Scientists usually use between 10^3 to 10^{12} particles to model a system.

Particle Methods Cont'd

$x_i(t) =$ location of the ith particle t seconds after the avalanche.

Particle Methods Cont'd

$x_i(t) =$ location of the ith particle t seconds after the avalanche.

To model $x_i(t)$, we consider its velocity $x_i'(t)$. The velocity depends on:

Particle Methods Cont'd

$x_i(t) =$ location of the ith particle t seconds after the avalanche.

To model $x_i(t)$, we consider its velocity $x_i'(t)$. The velocity depends on:

1. External forces. For example, the wind might be blowing the particle rightward with speed 1.

Particle Methods Cont'd

$x_i(t)$ = location of the ith particle t seconds after the avalanche.

To model $x_i(t)$, we consider its velocity $x_i'(t)$. The velocity depends on:

1. External forces. For example, the wind might be blowing the particle rightward with speed 1.
2. Interactions with other particles. If two particles get too close, they repel each other. We take the average over the interactions with x_i and every other particle.

Particle Methods Cont'd

$x_i(t) =$ location of the ith particle t seconds after the avalanche.

To model $x_i(t)$, we consider its velocity $x_i'(t)$. The velocity depends on:

1. External forces. For example, the wind might be blowing the particle rightward with speed 1.
2. Interactions with other particles. If two particles get too close, they repel each other. We take the average over the interactions with x_i and every other particle.

The total trajectory is

$$x_i'(t) = 1 + \frac{1}{N}\sum_{j=1}^{N} \frac{1}{1 + |x_i(t) - x_j(t)|}.$$

The Forward Euler Method

Intuition:

- Take the current position of the particle, add the current velocity multiplied by a small time step (Δt), to find the new position.
- This allows us to "step" through time and predict where the particle will eventually land.

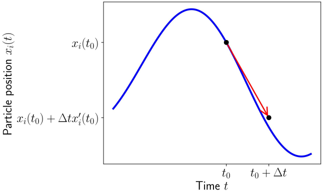

Figure: The forward Euler method.

Equation: $x_i(t_1) = x_i(t_0) + \Delta t x_i'(t)$.

One Big Issue

Recall the trajectory of the *i*th particle at time *t*:

$$x_i'(t) = 1 + \sum_{j=1}^{N} \frac{1}{1 + |x_i(t) - x_j(t)|}.$$

When using the forward Euler method to predict the location of $x_i(t)$, we have to sum over EVERY other particle $\Rightarrow$ extremely computationally costly.

One Big Issue

Recall the trajectory of the *i*th particle at time t:

$$x_i'(t) = 1 + \sum_{j=1}^{N} \frac{1}{1 + |x_i(t) - x_j(t)|}.$$

When using the forward Euler method to predict the location of $x_i(t)$, we have to sum over EVERY other particle $\Rightarrow$ extremely computationally costly.

For example, suppose we split our mountain into 1000 chunks. If it takes .01 second to compute $K(x_i(t), x_j(t))$, then

One Big Issue

Recall the trajectory of the *i*th particle at time *t*:

$$x_i'(t) = 1 + \sum_{j=1}^{N} \frac{1}{1 + |x_i(t) - x_j(t)|}.$$

When using the forward Euler method to predict the location of $x_i(t)$, we have to sum over EVERY other particle $\Rightarrow$ extremely computationally costly.

For example, suppose we split our mountain into 1000 chunks. If it takes .01 second to compute $K(x_i(t), x_j(t))$, then

$\Rightarrow$ It takes $1000 * .01 = 10$ seconds to calculate the trajectory of $x_i(t)$.

One Big Issue

Recall the trajectory of the *i*th particle at time t:

$$x_i'(t) = 1 + \sum_{j=1}^{N} \frac{1}{1 + |x_i(t) - x_j(t)|}.$$

When using the forward Euler method to predict the location of $x_i(t)$, we have to sum over EVERY other particle $\Rightarrow$ extremely computationally costly.

For example, suppose we split our mountain into 1000 chunks. If it takes .01 second to compute $K(x_i(t), x_j(t))$, then

- $\Rightarrow$ It takes $1000 * .01 = 10$ seconds to calculate the trajectory of $x_i(t)$.
- $\Rightarrow$ It takes $1000 * 10 = 10000$ seconds to calculate the trajectory of EVERY particle. This is almost three hours of computation time!

One Big Issue cont'd

Recall: 1000 particles $\Rightarrow$ 3 hour computation time.

Suppose we have 2000 particles.

One Big Issue cont'd

Recall: 1000 particles $\Rightarrow$ 3 hour computation time.

Suppose we have 2000 particles.

$\Rightarrow$ It takes $2000 * .01 = 20$ seconds to calculate the trajectory of $x_i(t)$.

One Big Issue cont'd

Recall: 1000 particles $\Rightarrow$ 3 hour computation time.

Suppose we have 2000 particles.

$\Rightarrow$ It takes $2000 * .01 = 20$ seconds to calculate the trajectory of $x_i(t)$.

$\Rightarrow$ It takes $2000 * 20 = 40000$ seconds to calculate the trajectory of EVERY particle. This is almost twelve hours of computation time!

One Big Issue cont'd

Recall: 1000 particles $\Rightarrow$ 3 hour computation time.

Suppose we have 2000 particles.

- $\Rightarrow$ It takes $2000 * .01 = 20$ seconds to calculate the trajectory of $x_i(t)$.
- $\Rightarrow$ It takes $2000 * 20 = 40000$ seconds to calculate the trajectory of EVERY particle. This is almost twelve hours of computation time!

When we double the particles, we quadruple the runtime.

One Big Issue cont'd

Recall: 1000 particles $\Rightarrow$ 3 hour computation time.

Suppose we have 2000 particles.

$\Rightarrow$ It takes $2000 * .01 = 20$ seconds to calculate the trajectory of $x_i(t)$.

$\Rightarrow$ It takes $2000 * 20 = 40000$ seconds to calculate the trajectory of EVERY particle. This is almost twelve hours of computation time!

When we double the particles, we quadruple the runtime.

Similarly,

- 3x as many particles $\Rightarrow$

One Big Issue cont'd

Recall: 1000 particles $\Rightarrow$ 3 hour computation time.

Suppose we have 2000 particles.

$\Rightarrow$ It takes $2000 * .01 = 20$ seconds to calculate the trajectory of $x_i(t)$.

$\Rightarrow$ It takes $2000 * 20 = 40000$ seconds to calculate the trajectory of EVERY particle. This is almost twelve hours of computation time!

When we double the particles, we quadruple the runtime.

Similarly,

- 3x as many particles $\Rightarrow$ 9x the runtime.

One Big Issue cont'd

Recall: 1000 particles $\Rightarrow$ 3 hour computation time.

Suppose we have 2000 particles.

$\Rightarrow$ It takes $2000 * .01 = 20$ seconds to calculate the trajectory of $x_i(t)$.

$\Rightarrow$ It takes $2000 * 20 = 40000$ seconds to calculate the trajectory of EVERY particle. This is almost twelve hours of computation time!

When we double the particles, we quadruple the runtime.

Similarly,

- 3x as many particles $\Rightarrow$ 9x the runtime.
- 4x as many particles $\Rightarrow$ 16x the runtime.

One Big Issue cont'd

Recall: 1000 particles $\Rightarrow$ 3 hour computation time.

Suppose we have 2000 particles.

- $\Rightarrow$ It takes $2000 * .01 = 20$ seconds to calculate the trajectory of $x_i(t)$.
- $\Rightarrow$ It takes $2000 * 20 = 40000$ seconds to calculate the trajectory of EVERY particle. This is almost twelve hours of computation time!

When we double the particles, we quadruple the runtime.

Similarly,

- 3x as many particles $\Rightarrow$ 9x the runtime.
- 4x as many particles $\Rightarrow$ 16x the runtime.

Computer scientists say the runtime is $O(N^2)$, because we effectively square the number of particles to get the runtime.

The Random Batch Method

Idea: Suppose we want to predict the location of $x_i(t)$ after a small time jump. We select a **batch** of representative particles. These particles are the only ones that affect the trajectory of our particle:

$$x_i'(t) = \sum_{j=1}^{N} K(x_j(t), x_i(t)) \Rightarrow x_i'(t) = \sum_{j \in I_{\text{batch}}} K(x_j(t), x_i(t)).$$

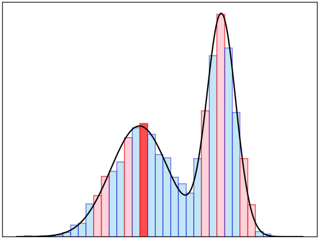

Figure: The location of the red chunk ONLY depends on the batch of randomly-selected pink chunks. We ignore every other chunk on the mountain.

The Random Batch Method: Algorithm

Input: Initial positions $x_i(0)$, time step Δt, batch size p.

Pseudocode

for each time step $k = 0, 1, \ldots$ **do**

end for

The Random Batch Method: Algorithm

Input: Initial positions $x_i(0)$, time step Δt, batch size p.

Pseudocode

for each time step $k = 0, 1, \ldots$ **do**

  Let $B_1, \ldots, B_M$ be randomly chosen batches of $\{1, \ldots, N\}$, each of size p

end for

The Random Batch Method: Algorithm

Input: Initial positions $x_i(0)$, time step Δt, batch size p.

Pseudocode

```
for each time step k = 0, 1, ... do
    Let B_1, ..., B_M be randomly chosen batches of {1, ..., N}, each of size p
    for each batch B_ℓ of size p do
    end for
end for
```

The Random Batch Method: Algorithm

Input: Initial positions $x_i(0)$, time step Δt, batch size p.

Pseudocode

for each time step $k = 0, 1, \ldots$ **do**
  Let $B_1, \ldots, B_M$ be randomly chosen batches of $\{1, \ldots, N\}$, each of size p
  for each batch B_ℓ of size p **do**
    for each particle $i \in B_\ell$ **do**
    end for
  end for
end for

The Random Batch Method: Algorithm

Input: Initial positions $x_i(0)$, time step Δt, batch size p.

Pseudocode

for each time step $k = 0, 1, \ldots$ **do**
  Let $B_1, \ldots, B_M$ be randomly chosen batches of $\{1, \ldots, N\}$, each of size p
  for each batch B_ℓ of size p **do**
    for each particle $i \in B_\ell$ **do**
      // Update using only local batch neighbors
      $x_i^{k+1} = x_i^k + \Delta t \left(1 + \frac{1}{p-1} \sum_{j \in B_\ell} \frac{1}{1+|x_i^k - x_j^k|}\right)$
    end for
  end for
end for

Time Savings of the Random Batch Method

Ex. Suppose we split our mountain into 1000 chunks and implement the random batch method with batches of size 5. If it takes .01 second to compute $K(x_i(t), x_j(t))$, then

Time Savings of the Random Batch Method

Ex. Suppose we split our mountain into 1000 chunks and implement the random batch method with batches of size 5. If it takes .01 second to compute $K(x_i(t), x_j(t))$, then

$\Rightarrow$ It takes $5 * .01 = .05$ seconds to calculate the trajectory of $x_i(t)$.

Time Savings of the Random Batch Method

Ex. Suppose we split our mountain into 1000 chunks and implement the random batch method with batches of size 5. If it takes .01 second to compute $K(x_i(t), x_j(t))$, then

$\Rightarrow$ It takes $5 * .01 = .05$ seconds to calculate the trajectory of $x_i(t)$.

$\Rightarrow$ It takes $1000 * .05 = 50$ seconds to calculate the trajectory of EVERY particle.

Time Savings of the Random Batch Method

Ex. Suppose we split our mountain into 1000 chunks and implement the random batch method with batches of size 5. If it takes .01 second to compute $K(x_i(t), x_j(t))$, then

$\Rightarrow$ It takes $5 * .01 = .05$ seconds to calculate the trajectory of $x_i(t)$.

$\Rightarrow$ It takes $1000 * .05 = 50$ seconds to calculate the trajectory of EVERY particle.

We have shortened our computation time from 3 hours to 1 minute!

Time Savings of the Random Batch Method

Ex. Suppose we split our mountain into 1000 chunks and implement the random batch method with batches of size 5. If it takes .01 second to compute $K(x_i(t), x_j(t))$, then

$\Rightarrow$ It takes $5 * .01 = .05$ seconds to calculate the trajectory of $x_i(t)$.

$\Rightarrow$ It takes $1000 * .05 = 50$ seconds to calculate the trajectory of EVERY particle.

We have shortened our computation time from 3 hours to 1 minute!

Suppose we have 2000 particles.

Time Savings of the Random Batch Method

Ex. Suppose we split our mountain into 1000 chunks and implement the random batch method with batches of size 5. If it takes .01 second to compute $K(x_i(t), x_j(t))$, then

$\Rightarrow$ It takes $5 * .01 = .05$ seconds to calculate the trajectory of $x_i(t)$.

$\Rightarrow$ It takes $1000 * .05 = 50$ seconds to calculate the trajectory of EVERY particle.

We have shortened our computation time from 3 hours to 1 minute!

Suppose we have 2000 particles.

$\Rightarrow$ It takes $5 * .01 = .05$ seconds to calculate the trajectory of $x_i(t)$.

Time Savings of the Random Batch Method

Ex. Suppose we split our mountain into 1000 chunks and implement the random batch method with batches of size 5. If it takes .01 second to compute $K(x_i(t), x_j(t))$, then

$\Rightarrow$ It takes $5 * .01 = .05$ seconds to calculate the trajectory of $x_i(t)$.

$\Rightarrow$ It takes $1000 * .05 = 50$ seconds to calculate the trajectory of EVERY particle.

We have shortened our computation time from 3 hours to 1 minute!

Suppose we have 2000 particles.

$\Rightarrow$ It takes $5 * .01 = .05$ seconds to calculate the trajectory of $x_i(t)$.

$\Rightarrow$ It takes $2000 * .05 = 100$ seconds to calculate the trajectory of EVERY particle.

Time Savings of the Random Batch Method

Ex. Suppose we split our mountain into 1000 chunks and implement the random batch method with batches of size 5. If it takes .01 second to compute $K(x_i(t), x_j(t))$, then

$\Rightarrow$ It takes $5 * .01 = .05$ seconds to calculate the trajectory of $x_i(t)$.

$\Rightarrow$ It takes $1000 * .05 = 50$ seconds to calculate the trajectory of EVERY particle.

We have shortened our computation time from 3 hours to 1 minute!

Suppose we have 2000 particles.

$\Rightarrow$ It takes $5 * .01 = .05$ seconds to calculate the trajectory of $x_i(t)$.

$\Rightarrow$ It takes $2000 * .05 = 100$ seconds to calculate the trajectory of EVERY particle.

Similarly,

- 3x as many particles $\Rightarrow$ 3x the runtime.
- 4x as many particles $\Rightarrow$ 4x the runtime.

Time Savings of the Random Batch Method

Ex. Suppose we split our mountain into 1000 chunks and implement the random batch method with batches of size 5. If it takes .01 second to compute $K(x_i(t), x_j(t))$, then

$\Rightarrow$ It takes $5 * .01 = .05$ seconds to calculate the trajectory of $x_i(t)$.

$\Rightarrow$ It takes $1000 * .05 = 50$ seconds to calculate the trajectory of EVERY particle.

We have shortened our computation time from 3 hours to 1 minute!

Suppose we have 2000 particles.

$\Rightarrow$ It takes $5 * .01 = .05$ seconds to calculate the trajectory of $x_i(t)$.

$\Rightarrow$ It takes $2000 * .05 = 100$ seconds to calculate the trajectory of EVERY particle.

Similarly,

- 3x as many particles $\Rightarrow$ 3x the runtime.
- 4x as many particles $\Rightarrow$ 4x the runtime.

Computer scientists say the runtime is $O(N)$, because the runtime is proportional to the number of particles.

Applications of the random batch method

The Random Batch Method is used to solve many different interactive particle systems:

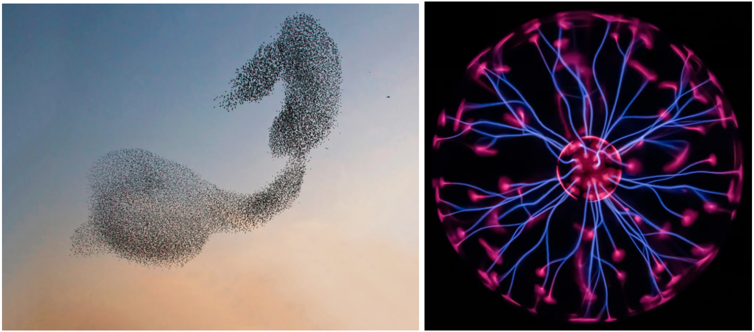

Figure: A flock of birds and a diagram of a plasma particle.

Applications of the random batch method

The Random Batch Method is used to solve many different interactive particle systems:

1. How does a flock of birds (particles) move?


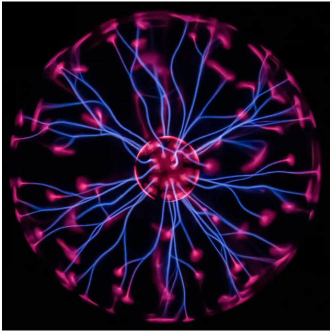

Figure: A flock of birds and a diagram of a plasma particle.

Applications of the random batch method

The Random Batch Method is used to solve many different interactive particle systems:

1. How does a flock of birds (particles) move?
2. How do the opinions of a group of individuals (particles) change?

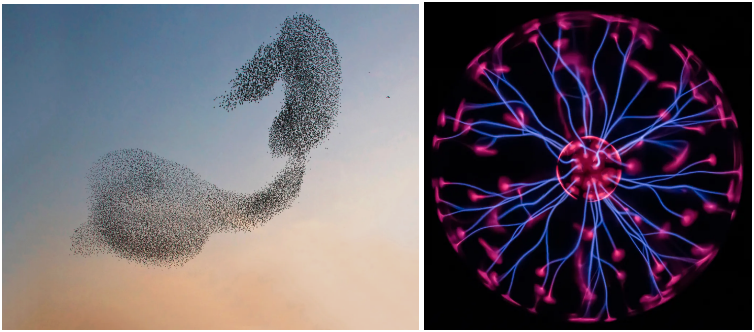

Figure: A flock of birds and a diagram of a plasma particle.

Applications of the random batch method

The Random Batch Method is used to solve many different interactive particle systems:

1. How does a flock of birds (particles) move?
2. How do the opinions of a group of individuals (particles) change?
3. How does a collection of ions and electrons (particles) evolve in time?

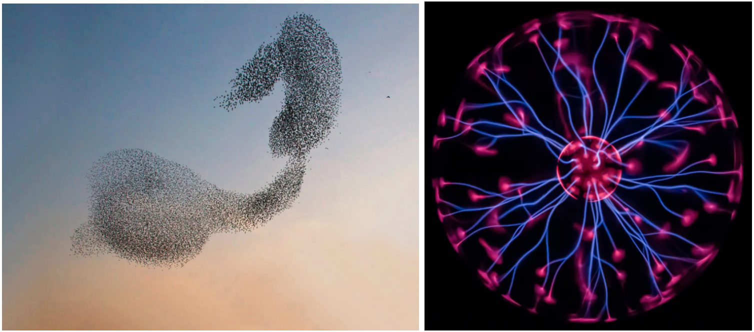

Figure: A flock of birds and a diagram of a plasma particle.

Problems with the Random Batch Method

Just because the random batch method is fast, doesn't mean it's accurate!

Problems with the Random Batch Method

Just because the random batch method is fast, doesn't mean it's accurate!

Ex. The Porous Medium Equation

Problems with the Random Batch Method

Just because the random batch method is fast, doesn't mean it's accurate!

Ex. The Porous Medium Equation

- Models the way a liquid diffuses through a porous surface; think of pouring a sauce over cheesecloth.

Problems with the Random Batch Method

Just because the random batch method is fast, doesn't mean it's accurate!

Ex. The Porous Medium Equation

- Models the way a liquid diffuses through a porous surface; think of pouring a sauce over cheesecloth.
- Predicts how groundwater settles after floods or dam breaks $\Rightarrow$ helps engineers design more effective remediation schemes to clean up water supplies.

Problems with the Random Batch Method

Just because the random batch method is fast, doesn't mean it's accurate!

Ex. The Porous Medium Equation

- Models the way a liquid diffuses through a porous surface; think of pouring a sauce over cheesecloth.
- Predicts how groundwater settles after floods or dam breaks $\Rightarrow$ helps engineers design more effective remediation schemes to clean up water supplies.
- Predicts how biological populations, such as cells in a tumor, invade surrounding tissue $\Rightarrow$ helps doctors determine the growth and spread of tumors.

While the random batch method can quickly solve the porous medium equation, these solutions are not very accurate.

The Batch Forward Euler Method

Recall the forward Euler equation: $x_i(t_1) = x_i(t_0) + \Delta t x_i'(t)$.

Idea:

1. We randomly select a batch of fast-moving particles.
2. These particles take a big step forward in time according to the forward Euler scheme.
3. The remaining particles take lots of little steps until they "catch up" to the fast particles.

Ideally, the fast particles make the computation faster, while the slow particles preserve the accuracy of the system.

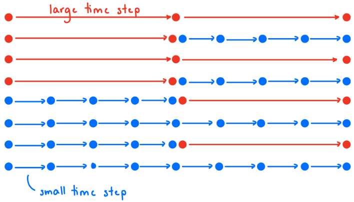

Figure: The batch forward Euler method. Red particles move fast, while blue particle move slow.

Height Constraint

Imagine trying to build Mammoth mountain inside a room with a low ceiling: the “chunks” of the mountain concentrate, until they hit this ceiling.

Height Constraint

Imagine trying to build Mammoth mountain inside a room with a low ceiling: the “chunks” of the mountain concentrate, until they hit this ceiling.

Between regular forward Euler method, the random batch method, and the batch forward Euler method, which method “wins" at simulating height constraint?

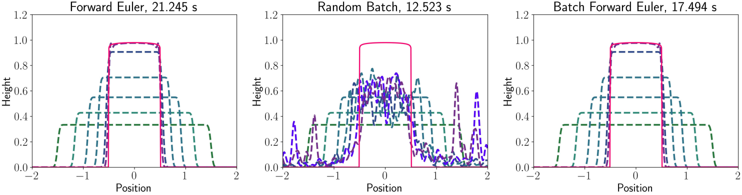

Figure: Building a mountain in a room with a low ceiling.

The Heat Equation

Suppose a rod is heated up. As it cools, how does the heat spread? We can visualize this in two dimensions, where:

1. The x-axis is the rod.
2. The y-axis is the temperature of the rod.

The Heat Equation

Suppose a rod is heated up. As it cools, how does the heat spread? We can visualize this in two dimensions, where:

1. The x-axis is the rod.
2. The y-axis is the temperature of the rod.

Between regular forward Euler method, the random batch method, and the batch forward Euler method, which method “wins" at simulating the heat equation?

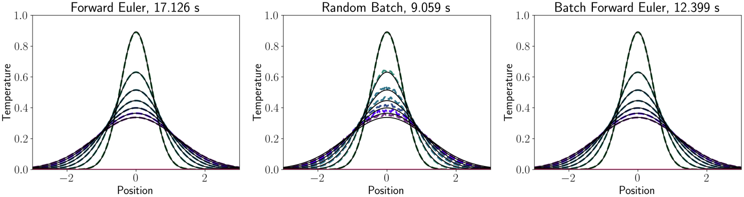

Figure: How does the temperature of a heated rod change in time?

Other Applications of the Batch Forward Euler Method

We can also apply the batch forward Euler method in the following cases:

Figure: The diffusion of a gas and the behavior of an avalanche can both be simulated using the batch forward Euler method.

Other Applications of the Batch Forward Euler Method

We can also apply the batch forward Euler method in the following cases:

1. Fast and slow diffusion (related to the porous medium equation).

Figure: The diffusion of a gas and the behavior of an avalanche can both be simulated using the batch forward Euler method.

Other Applications of the Batch Forward Euler Method

We can also apply the batch forward Euler method in the following cases:

1. Fast and slow diffusion (related to the porous medium equation).
2. Avalanche dynamics (related to our Mammoth mountain example).

Figure: The diffusion of a gas and the behavior of an avalanche can both be simulated using the batch forward Euler method.

Other Applications of the Batch Forward Euler Method

We can also apply the batch forward Euler method in the following cases:

1. Fast and slow diffusion (related to the porous medium equation).
2. Avalanche dynamics (related to our Mammoth mountain example).
3. The Navier-Stokes equation: Simulates the movement of air over a plane's wing, or the turbulence of water in a pipe.

Figure: The diffusion of a gas and the behavior of an avalanche can both be simulated using the batch forward Euler method.