

SIGGRAPH 2022
 MONTE CARLO 101: AREA



GRID-FREE MONTE CARLO FOR PDES WITH SPATIALLY VARYING COEFFICIENTS

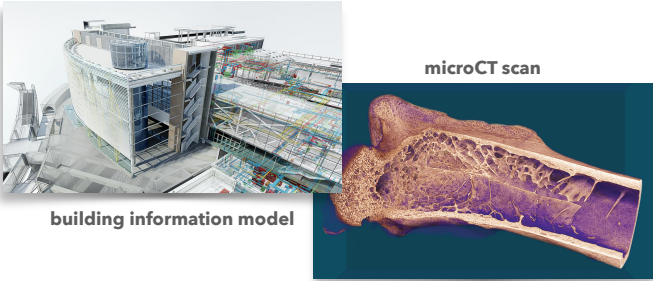
ROHAN SAWHNEY*, CARNEGIE MELLON UNIVERSITY
 DARIO SEYB*, DARTMOUTH COLLEGE
 WOJCIECH JAROSZ†, DARTMOUTH COLLEGE
 KEENAN CRANE†, CARNEGIE MELLON UNIVERSITY

The symbols * and † indicate equal contribution.

© 2022 SIGGRAPH. ALL RIGHTS RESERVED.

- Hi everyone, in this talk we'll present a Monte Carlo method to solve partial differential equations with spatially varying coefficients.

Geometric & material complexity in science & engineering




building information model

microCT scan

2

- Models in engineering & science often have way more complexity in their geometry and materials than what conventional PDE solvers can handle.

Photorealistic rendering of complex geometry & materials



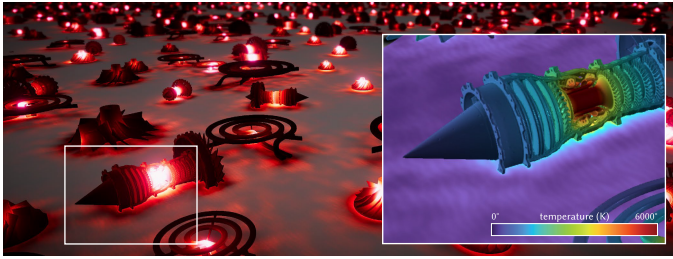
geometry + materials

rendered output

3

- But imagine if simulation was like Monte Carlo rendering: just load up a complex model and hit go without worrying about meshing or basis functions.
- Our paper takes a major step towards this vision by building a bridge between PDEs and volume rendering.

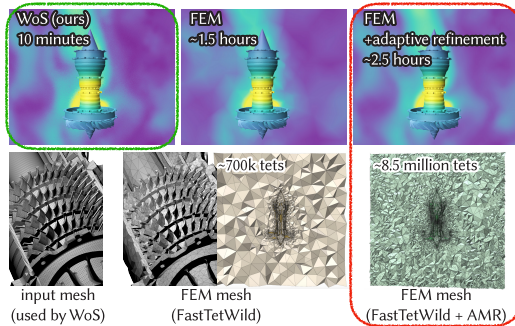
Physical analysis of complex geometry & materials



Heat radiating from infinitely many blackbodies in a heterogeneous medium
About 600 million effective vertices from visible viewpoint

- Here's an example: heat radiating off of infinitely many black body emitters, each with super-detailed geometry and material coefficients.
- From this view alone, the boundary meshes have ~600M vertices.
- To get the same level of detail with a conventional PDE solver such as

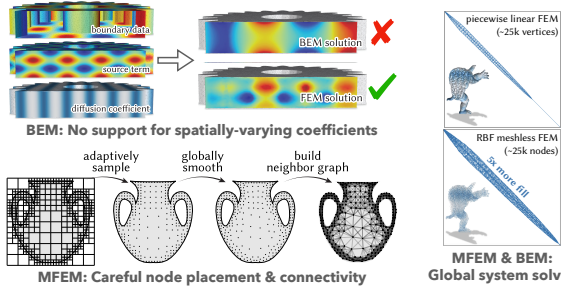
Challenge with conventional PDE solvers: scalability



- the finite element method, we need about 8.5M tetrahedra & 2.5 hours of meshing time even on a tiny piece of the scene.
- But with Monte Carlo we get rapid feedback that can be progressively refined.

Challenge with conventional "mesh free" PDE solvers

Boundary Element Method (BEM) and Meshless Finite Element (MFEM)



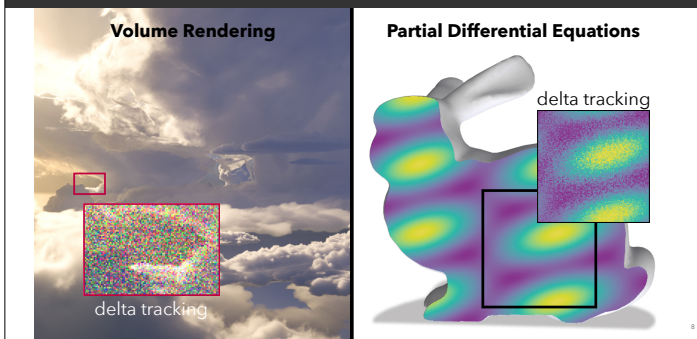
- At this moment, experts might point to the boundary element method and meshless FEM.
- The short story here is that these methods either lack support for variable coefficients, or they must still do expensive & error-prone node placement and global solves.

Monte Carlo Geometry Processing [SIGGRAPH 2020]



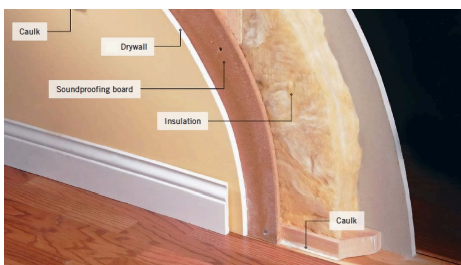
- Our journey with Monte Carlo began a few years ago, when we realized that rendering techniques from computer graphics could be used to turbo charge Muller's "walk on spheres" algorithm for solving PDEs.

Contribution: Bridge between PDEs & Volume Rendering



- In this paper, we make the connection between rendering & simulation even stronger, by linking and applying tools from volume rendering to PDEs with variable coefficients.

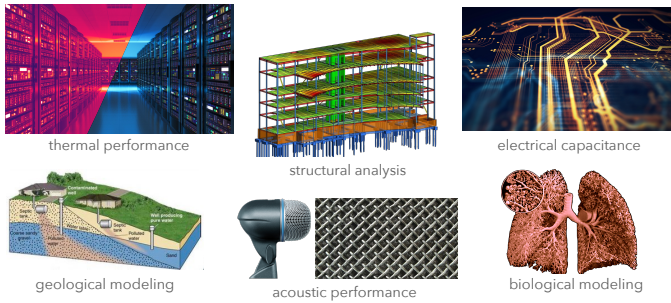
Spatial heterogeneity is everywhere!



Rich material properties e.g., wall with thermal insulation, sound proofing etc.

- Spatially-varying coefficients are essential for capturing rich material properties.
- For instance, to understand the thermal performance of a building, note that even a basic wall isn't just a homogeneous slab—it has many layers of different density.

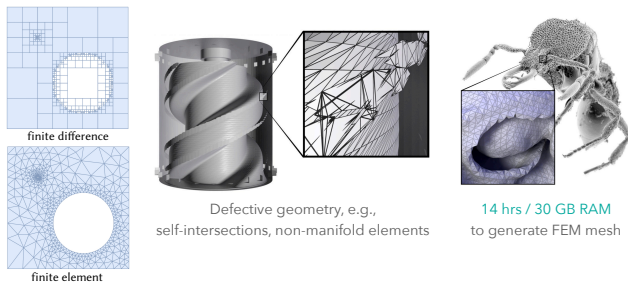
Spatial heterogeneity is everywhere!



- And that's just the tip of the iceberg—PDEs with variable coefficients are everywhere in science & engineering, from thermal and structural analysis, to biomolecular and geological modeling.

Shortcoming of conventional PDE solvers

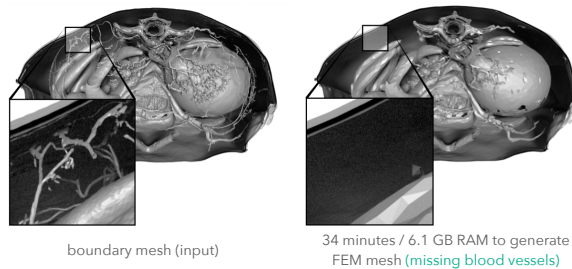
Spatial discretization: expensive and error-prone



- A major challenge with any PDE solver is spatial discretization: this process is expensive and error prone, especially for complex geometric models.

Shortcoming of conventional PDE solvers

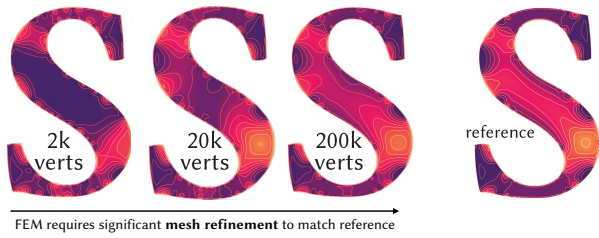
Spatial discretization: destroys geometric features



- Apart from its massive cost, discretization also causes two major headaches for solving PDEs:
 - 1) important geometric features often get destroyed, and

Shortcoming of conventional PDE solvers

Spatial discretization: causes aliasing in the PDE inputs and solution

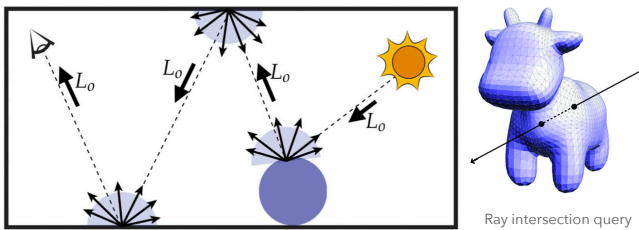


13

- 2) and significant mesh refinement can be needed to remove aliasing artifacts in the PDE solution, boundary conditions and coefficients.

Monte Carlo Rendering

Does not require high quality meshing & solving global systems

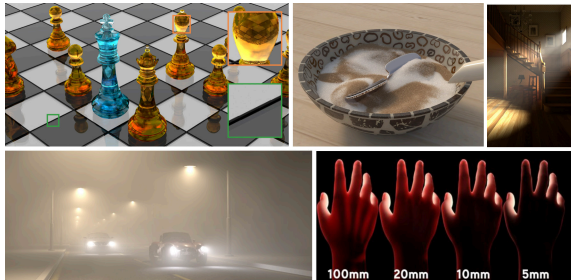


14

- To avoid these problems, photorealistic rendering moved away from meshing to Monte Carlo methods that only need point-wise access to the geometry via ray intersection queries.

Monte Carlo Rendering

Photorealistic image generation of participating and granular media

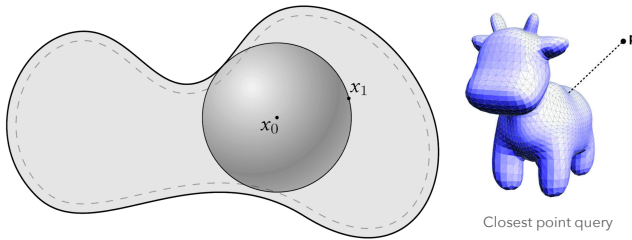


15

- This enabled simulation of intricate light transport phenomena on complex geometric models.
- So what about PDEs?

The walk on spheres (WoS) algorithm [Muller 1956]

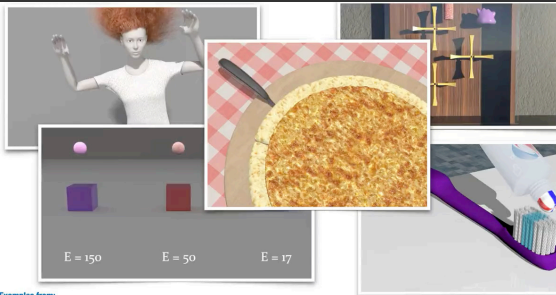
Does not require high quality meshing & solving global systems



16

- Here there's a little known algorithms called walk on spheres, which avoids spatial discretization altogether.
- Much like rendering, it only needs access to a single geometric kernel, namely closely point queries.
- Now, to be candid: WoS is **way** behind mature technology

Multi-material physical simulation in graphics



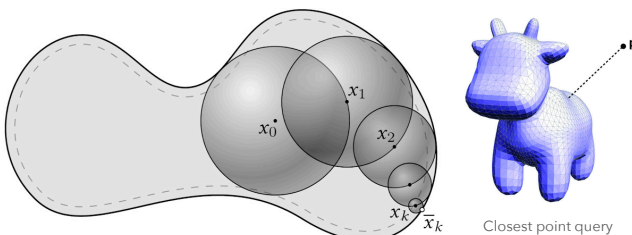
Examples from:
Han et al., "A Hybrid Material Point Method for Frictional Contact with Diverse Materials" (2019)
Zhu et al., "Codimensional Non-Newtonian Fluids" (2015)

17

- like FEM—especially in computer graphics we've seen amazing PDE solvers that handle complex multi-physics scenarios.

The walk on spheres (WoS) algorithm [Muller 1956]

Goal: extend to broader class of problems \implies PDEs with variable coefficients



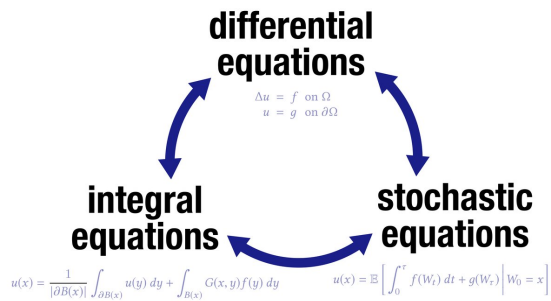
18

- But the WoS idea—and its promise to free us from the bonds for spatial discretization—is so appealing that we want to extend it to a broader classes of PDEs.
- So, let's talk about how

BACKGROUND

- we can extend WoS to variable-coefficient problems.

A tale of three equations...



- This story is really a tale of three kinds of equations:
 - 1) PDEs
 - 2) integral equations and
 - 3) stochastic differential equations.
- Our paper provides a playbook to convert between these different forms.

2nd order linear elliptic PDEs

$$\nabla \cdot (\alpha \nabla u) + \vec{\omega} \cdot \nabla u - \sigma u = -f \text{ on } \Omega$$

diffusion drift absorption source domain
∇ · (α ∇ u) + ω̄ · ∇ u - σ u = -f on Ω

$$u = g \text{ on } \partial\Omega$$

boundary values domain boundary
u = g on ∂Ω

solution → u

- In precise terms, our goal is to develop a Monte Carlo method that solves 2nd order linear elliptic equations with spatially-varying diffusion, drift, and absorption coefficients.
- Let's unpack this equation term by term.

2nd order linear elliptic PDEs

Laplace equation



boundary
values $g(x)$



$$\Delta u = 0$$
$$\Delta := \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}$$

22

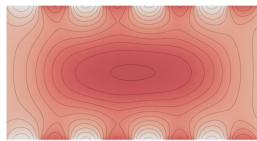
- First, a Laplace equation describes the steady-state temperature inside a domain if heat is fixed to some given function g on the boundary.

2nd order linear elliptic PDEs

Poisson equation



source
term $f(x)$



$$\Delta u = f$$

23

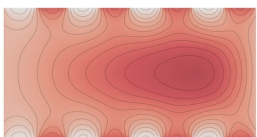
- Adding a source term f yields a Poisson equation, where f describes a background temperature.
- Imagine heat being pumped into the domain at a rate f at each point x .

2nd order linear elliptic PDEs

variable diffusion Poisson equation



diffusion
coeff. $\alpha(x)$



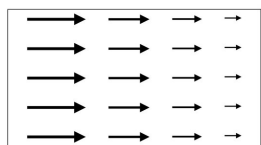
$$\nabla \cdot (\alpha \nabla u) = f$$

24

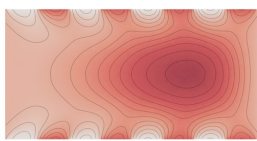
- We can control the rate of heat diffusion by replacing the laplacian with the operator $\nabla \cdot (\alpha \nabla u)$, where α is a scalar function.
- Physically α might describe the thickness or varying composition of a material.

2nd order linear elliptic PDEs

stationary advection-diffusion equation



transport coeff. $\vec{\omega}(x)$



$$\Delta u + \vec{\omega} \cdot \nabla u = f$$

25

- Adding a drift term $\vec{\omega} \cdot \nabla u$ to a Poisson equation indicates that heat is pushed along some vector field $\vec{\omega}$ —imagine a flowing river, which mixes hot water into cold water until it reaches a steady state.

2nd order linear elliptic PDEs

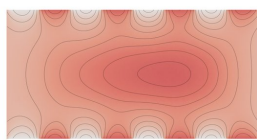
screened Poisson equation

10

-1



absorption coeff. $\sigma(x)$



$$\Delta u - \sigma u = f$$

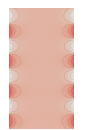
26

- Finally, an absorption term σu acts like a background medium that absorbs heat—think about a heat sink or a cold engine block.
- The function σ describes the strength of absorption at each point x .

2nd order linear elliptic PDEs



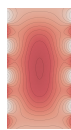
boundary values $g(x)$



$$\Delta u = 0$$



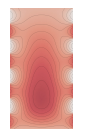
source term $f(x)$



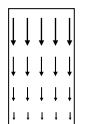
$$\Delta u = f$$



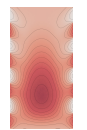
diffusion coeff. $\alpha(x)$



$$\nabla \cdot (\alpha \nabla u) = f$$



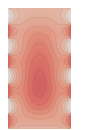
transport coeff. $\vec{\omega}(x)$



$$\Delta u + \vec{\omega} \cdot \nabla u = f$$



absorption coeff. $\sigma(x)$



$$\Delta u - \sigma u = f$$

27

- There are lots of other terms you could add to a PDE, but these already get you pretty far.
- More importantly, these are terms we'll be able to convert into integral representations, and ultimately into Monte Carlo algorithms for PDEs!
- Let's see how...

Integral for Laplace equation

mean value property

basis for WoS!

$$u(x) = \frac{1}{|\partial B(x)|} \int_{\partial B(x)} u(y) dy$$

solution (unknown!) solution (unknown!)
 volume of bounding sphere ball around point x

28

- The solutions to basic PDEs can be expressed via integral equations.
- E.g., the solution to a Laplace equation is given by the “mean value property”, which says that “the solution at a point x equals the average value over any empty ball centered at x”.
- This integral is *recursive*: the unknown value u at x depend on unknown values at y!
- Sounds like a problem, but this *exactly* how WoS works:

Integral for Laplace equation

mean value property

basis for WoS!

$$u(x) = \frac{1}{|\partial B(x)|} \int_{\partial B(x)} u(y) dy$$

solution (unknown!) solution (unknown!)
 volume of bounding sphere ball around point x

29

- recursively estimate the value of u till we reach the boundary and then grab the known boundary value.

Integral for screened Poisson equation (constant absorption)

screened Poisson – PDE

diffusion absorption source term

$$\Delta u - \sigma u = -f \text{ on } \Omega$$

solution $u = g$ on $\partial\Omega$

screened Poisson – IE

WoS: sample f inside each ball

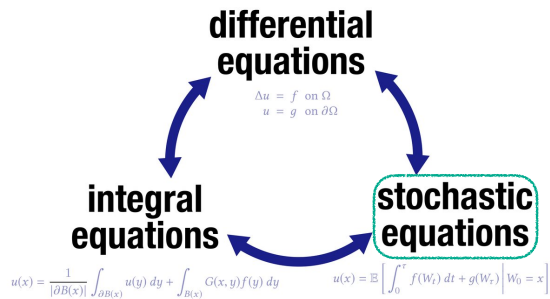
$$u(x) = \int_{B(c)} f(y) G^\sigma(x, y) dy + \int_{\partial B(c)} u(z) P^\sigma(x, z) dz$$

solution ball containing x source term Green's function solution Poisson kernel

30

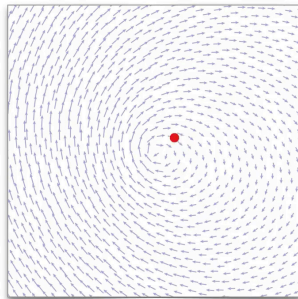
- As with PDEs, we can keep adding terms to integral equations to capture additional behavior.
- For instance there’s a nice integral representation for a screened Poisson equation, as long as the absorption coefficient sigma is constant!
- From an algorithmic perspective, WoS now also picks a random point inside each ball in the walk to sample the source term.

A tale of three equations...



- Finally, we can describe the same phenomena using stochastic differential equations (SDE).
- For us, the stochastic picture is super important because it lets us deal with spatially-varying coefficients.

Ordinary differential equation (ODE)



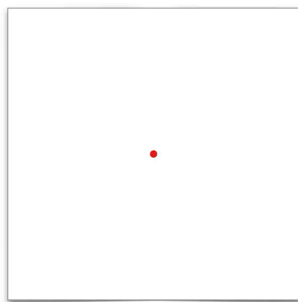
DETERMINISTIC MOTION

$$dX_t = \vec{\omega}(X_t) dt$$

- trajectory (X_t)
- drift direction ($\vec{\omega}$)

- An ODE describes the location of a particle in terms of its derivatives in time.
- For instance, $dX = \omega dt$ says that a particle's velocity is given by some vector field ω —e.g., a speck of dust blowing in the wind.

Stochastic differential equation (SDE)



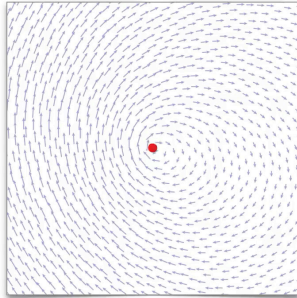
BROWNIAN MOTION

$$dX_t = dW_t$$

- trajectory (X_t)

- A *stochastic* differential equation describes random motions—a key example is a Brownian motion, where changes in position follow a Gaussian distribution, and are independent of past events.
- Brownian motion is often used to model everything from moving molecules to fluctuations in stock prices.

Stochastic differential equation (SDE)



BROWNIAN MOTION WITH DRIFT

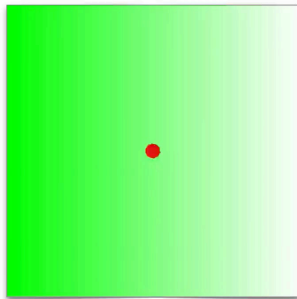
$$dX_t = \vec{\omega}(X_t) dt + dW_t$$

● trajectory (X_t)
→ drift direction ($\vec{\omega}$)

34

- Adding Brownian motion to our earlier ODE gives a more general *diffusion process* which we can think of as either a deterministic particle with noise, or a random walk with drift.

Stochastic differential equation (SDE)



BROWNIAN MOTION WITH VARIABLE SCALE

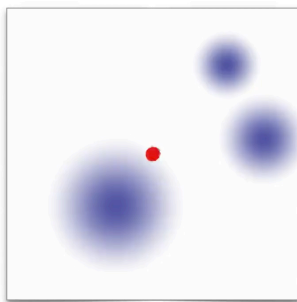
$$dX_t = \sqrt{\alpha(X_t)} dW_t$$

● trajectory (X_t)
■ diffusivity (α)

35

- We can also modulate the strength of the jiggling via a function α . As α increases, things “heat up”, and particles move faster.

Stochastic differential equation (SDE)



BROWNIAN MOTION IN ABSORBING MEDIUM

$$dX_t = dW_t$$

● trajectory (X_t)
■ absorption (σ)

36

- Finally we can think about a random walker possibly getting absorbed in a background medium, like ink getting soaked up in a sponge.
- Here σ denotes the strength of absorption—it doesn't show up in the SDE itself, but will be incorporated in a moment.

Feynman-Kac formula

$$u(x) = \mathbb{E} \left[\int_0^\tau e^{-\int_0^t \sigma(X_s) ds} f(X_t) dt + e^{-\int_0^\tau \sigma(X_t) dt} g(X_\tau) \right]$$

Labels in the formula: solution, time to reach boundary, absorption, source, random walk, boundary values.

$$dX_t = \tilde{\omega}(X_t) dt + \sqrt{\alpha(X_t)} dW_t$$

Labels in the SDE: random walk, velocity, diffusion rate, Brownian motion.

- Now, it's no coincidence that we use the same symbols α , σ , ω for both our PDE and SDE.
- These perspectives are linked by the Feynman-Kac formula, which gives the solution to our main PDE as an expectation over many random walks.

Special case: Kakutani's principle

$$u(x) = \mathbb{E}[u(W_\tau)]$$

$$= \frac{1}{|\partial B(x)|} \int_{\partial B(x)} u(y) dy$$

WoS simulates Brownian motion efficiently!

- A special case of Feynman Kac is Kakutani's principle, which says the solution to a Laplace equation is the average value seen by a Brownian random walk when it first hits the boundary.
- When restricted to a ball, Kakutani's principle is equivalent to the mean value property due to the rotational symmetry of Brownian motion.
- WoS can therefore be seen as an acceleration strategy for simulating brownian motion.

Feynman-Kac formula

$$u(x) = \mathbb{E} \left[\int_0^\tau e^{-\int_0^t \sigma(X_s) ds} f(X_t) dt + e^{-\int_0^\tau \sigma(X_t) dt} g(X_\tau) \right]$$

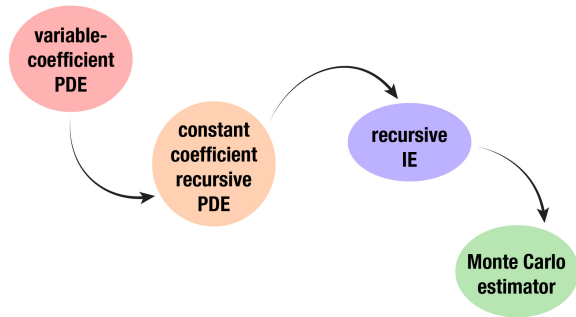
Labels in the formula: solution, time to reach boundary, absorption, source, random walk, boundary values.

$$dX_t = \tilde{\omega}(X_t) dt + \sqrt{\alpha(X_t)} dW_t$$

Labels in the SDE: random walk, velocity, diffusion rate, Brownian motion.

- More importantly, unlike classic integral equations, the Feynman-Kac formula handles spatially-varying coefficients!
- As a result, we can use it to

Next: recursive integral equation for variable coefficients



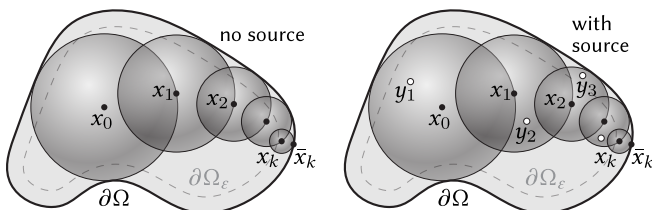
- build a *new* recursive but deterministic integral equation, which in turn leads to modified WoS algorithms for variable coefficient PDEs.

Method

- The key observation behind our method

WoS for PDEs with source terms

E.g., $\Delta u = f(x)$; sample the **spatially-varying source** f inside each ball



- Is that even though WoS cannot directly handle PDEs with variable coefficients, it can still be used to solve problems with spatially-varying source terms.

Transformations

Variable coefficient $\nabla \cdot (\alpha \nabla u) + \vec{\omega} \cdot \nabla u - \sigma u = -f$

↓ Girsanov & delta tracking transformations

Constant coefficient $\Delta u - \bar{\sigma} u = f(x, \alpha, \vec{\omega}, \sigma, u)$ — recursive
(No approximation!)
constant

↓

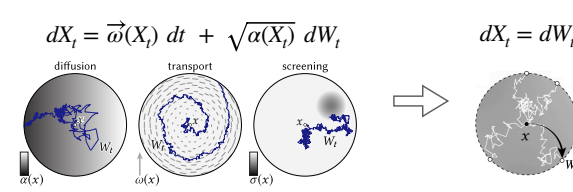
Integral $\int_{B(x)} f(y, \alpha, \vec{\omega}, \sigma, u) G^{\bar{\sigma}}(x, y) dy + \int_{\partial B(x)} u(z) P^{\bar{\sigma}}(x, z) dz$

- We therefore apply a series of transformations that convert our original heterogeneous PDE into a *constant*-coefficient screened Poisson equation with a recursive source term.
- From an FEM perspective, it might feel like we haven't done anything useful: we just shuffled all the hard stuff to the other side of the equals sign.
- Yet from the Monte Carlo perspective we now have a way forward, since we can recursively estimate the resulting deterministic integral.

Transformation 1: Girsanov

Re-express Feynman Kac in terms of Brownian motion

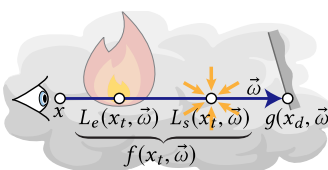
$$u(x) = \mathbb{E} \left[\int_0^\tau e^{-\int_0^s \sigma(W_t) ds} f(W_t) dt + e^{-\int_0^\tau \sigma(W_t) dt} g(W_\tau) \right]$$

$$dX_t = \vec{\omega}(X_t) dt + \sqrt{\alpha(X_t)} dW_t \quad \Rightarrow \quad dX_t = dW_t$$


- On the stochastic front, our first transformation rewrites the Feynman Kac formula purely in terms of brownian motion instead of a diffusion process.
- As part of this transformation, all the original coefficients get converted into a single variable absorption coefficient \sigma.
- To get rid of this \sigma,

Volume Rendering Equation (VRE)

VRE describes the radiance in heterogeneous absorbing & scattering media

$$L(w, \vec{\omega}) = \int_0^d e^{-\int_0^s \sigma(x_t) ds} f(x_t, \vec{\omega}) dt + e^{-\int_0^d \sigma(x_t) dt} g(x_d, \vec{\omega})$$


- we then observe that the Feynman Kac formula actually looks a lot like the volume rendering equation, which in computer graphics describes the radiance L along a ray in a heterogeneous medium that absorbs, scatters and emits radiation.
- But if for a second we put aside the physical meaning of these symbols,

Structural connection between VRE & Feynman-Kac

$L(x, \vec{\omega}) = \int_0^d e^{-\int_0^s \sigma(x_t, \vec{\omega}) dt} f(x_s, \vec{\omega}) ds + e^{-\int_0^d \sigma(x_t, \vec{\omega}) dt} g(x_d, \vec{\omega})$

VRE describes the radiance in heterogeneous absorbing & scattering media

$u(x) = \mathbb{E} \left[\int_0^t e^{-\int_0^s \sigma(W_s) ds} f(W_s) ds + e^{-\int_0^t \sigma(W_s) ds} g(W_t) \right]$

Feynman-Kac for 2nd order variable coefficient PDEs

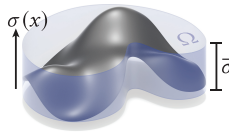
- then structurally the main difference between Feynman Kac and the VRE is that one requires simulation of Brownian random walks, while the other provides the radiance along a ray.
- And as a result, transformations like delta tracking used in graphics to solve the VRE

Transformation 2: Delta tracking

Variable coefficient $u(x) = \mathbb{E} \left[\int_0^t e^{-\int_0^s \sigma(W_s) ds} f(W_s) ds + e^{-\int_0^t \sigma(W_s) ds} g(W_t) \right]$

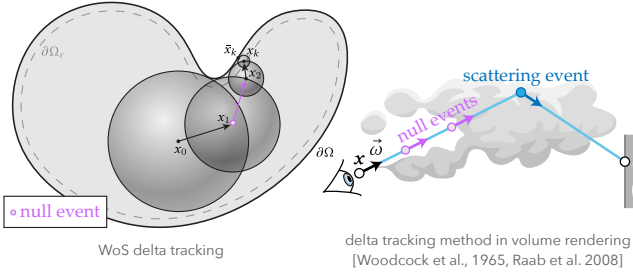
Constant coefficient $u(x) = \mathbb{E} \left[\int_0^t e^{-\bar{\sigma}t} \underbrace{f(W_t, \sigma, u)}_{\text{recursive}} dt + e^{-\bar{\sigma}t} \underbrace{g(W_t)}_{\text{constant}} \right]$

(No approximation!)



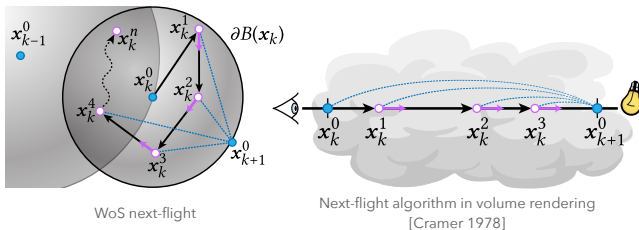
- can be applied to the Feynman Kac formula as well.
- Here we basically move the variable coefficient σ to a recursively defined source term f as in the PDE setting.
- $\bar{\sigma}$ is a free parameter, which we set to the difference between the maximal values of σ over the entire domain.
- Just as in volume rendering, we've essentially turned our original heterogeneous medium into an equivalent homogeneous one.
- Now algorithmically, this is all really interesting because PDEs can suddenly benefit from decades worth of rendering research!

Delta tracking variant of WoS



- In particular, these transformations allow us to develop modified versions of WoS with direct counterparts in volume rendering.
- For instance, the delta tracking version, shown here on the left, uses the concept of null events from volume rendering to sample points either inside or on the boundary of a ball.

Next-flight variant of WoS

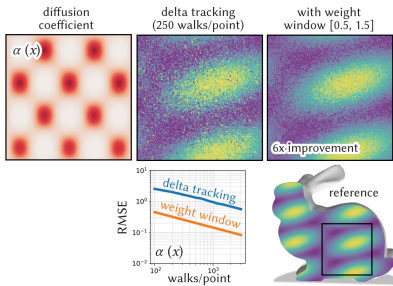


Fewer distance queries higher correlation compared to delta tracking WoS
 Similar variance & run-time characteristics as volume rendering counterparts

- Similarly, the next flight version always jumps to a random point on the largest sphere using off-centered walks, and conceptually it looks a lot like the next-flight algorithm from volume rendering.
- In practice, we find that while the next flight version requires fewer distance queries, it usually suffers from higher correlation compared to delta tracking.
- Both algorithms also share the variance and run-time characteristics of their volume rendering counterparts.

Weight window [Hoogenboom and Légrády 2005]

Uses **splitting** and **Russian roulette** to reduce noise



- Finally, on problems with high frequency coefficients, standard variance reduction techniques in Monte Carlo rendering, like Russian roulette and splitting, can provide similarly dramatic improvements to our algorithms, here providing a 6x speedup.
- Run-time performance also improves in this case, since walks are often terminated early.

Implementation & Results

- From an implementation perspective,

PDE inputs

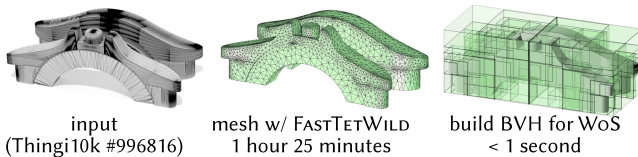
$$\nabla \cdot (\alpha \nabla u) + \vec{\omega} \cdot \nabla u - \sigma u = -f \text{ on } \Omega$$

diffusion drift absorption source domain
 $u = g$ on $\partial\Omega$
 solution boundary values domain boundary

- A PDE is encoded by the description of the scene geometry, boundary conditions, source term and coefficients.
- In our implementation, this data is provided via callback routines that return a value for any query point in the domain.

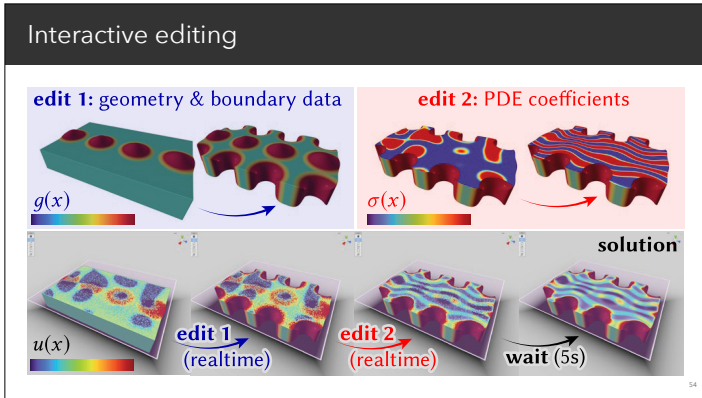
Acceleration of closest point queries

Accelerate closest point queries using BVH



Unlike bad meshes, BVHs do not impact correctness/accuracy of PDE solution!

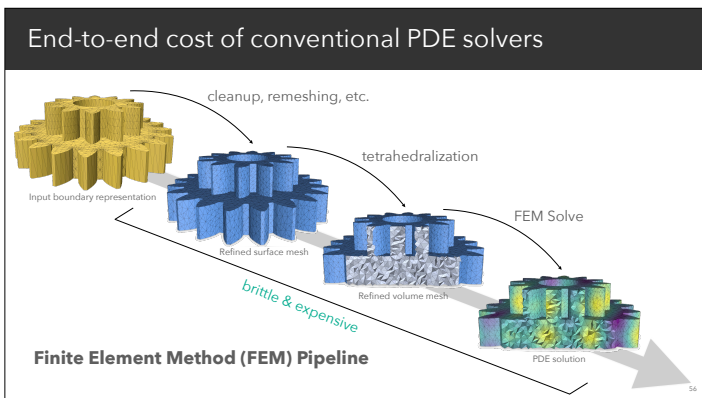
- Closest point queries can be accelerated via standard spatial hierarchies such as a BVH for a wide variety of scene representations.
- Unlike mesh generation, a BVH uses little memory and can be built quickly even for detailed models.
- Also, unlike a bad mesh, a poorly-constructed BVH only harms performance—not correctness or accuracy.



- Our approach is ideal for interactive editing since it operates directly on the original scene representation, and provides instant feedback after updates to the geometry, boundary conditions and PDE coefficients.



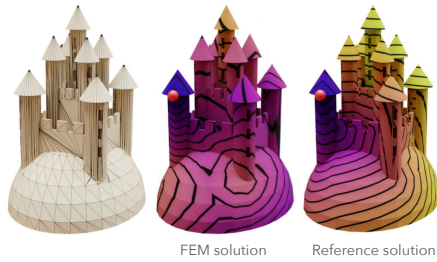
- Unlike conventional solvers, our method also doesn't require any geometric pre-processing, which allows it to scale to extremely large scenes.



- In contrast, the significant issue with traditional numerical methods such as FEM is the end-to-end cost of the pipeline: even if the FEM solve is fast, one has to first convert the boundary description into a high quality simulation mesh.
- This process can be brittle and slow, and requires careful consideration since

Conventional PDE solvers can be brittle

Poor mesh quality completely throws off FEM solution



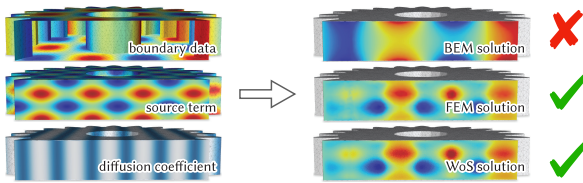
Source: Sharp and Crane, A Laplacian for Nonmanifold Triangle Meshes

57

- even a single bad quality element can throw off the the accuracy of an FEM solution completely.

Comparison with conventional solvers

The **boundary element method (BEM)** does not require volumetric meshing



BEM does not support problems with source terms or variable coefficients

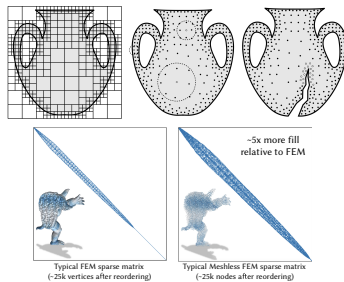
58

- Some conventional solvers such as the boundary element method don't need to mesh the domain.
- However, BEM can't handle problems with source terms or spatially-varying coefficients on the domain interior.
- To include these terms, it has to be coupled with a second solver such as FEM which requires volumetric meshing.

Comparison with conventional solvers

Meshless FEM solvers also do not require a volume mesh

- Require **dense** sampling of the domain
- Require solving large linear systems

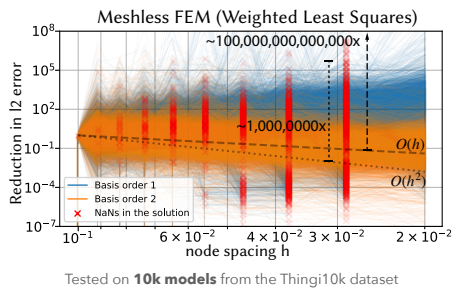


59

- Meshless FEM methods such as moving least squares also don't require meshing either.
- However, unlike Monte Carlo, these methods still require a dense and careful sampling of the entire domain.
- They also need to solve global systems of equations which are typically a lot larger in size compared to FEM.

Meshless FEM is unreliable

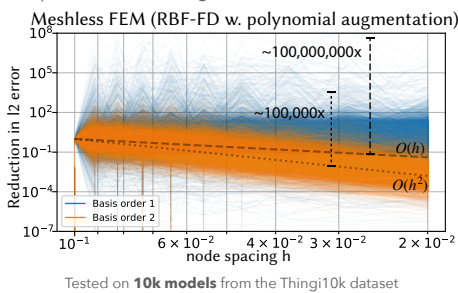
Solvers have unpredictable convergence under refinement



- The bigger problem with meshless methods is that they are unreliable.
- Here we solve a standard PDE on all models from the Thingi10k dataset, and plot the error under refinement.
- As indicated by the crosses in the plot, common meshless schemes often fail to converge under refinement, while

Meshless FEM is unreliable

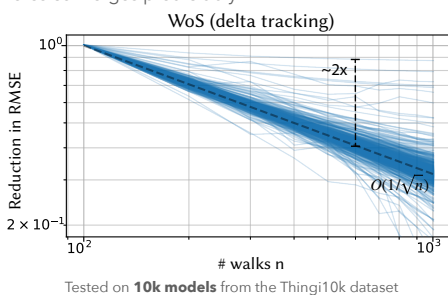
Solvers have unpredictable convergence under refinement



- even state of the art approaches show extremely large variation in error.

Meshless FEM is unreliable

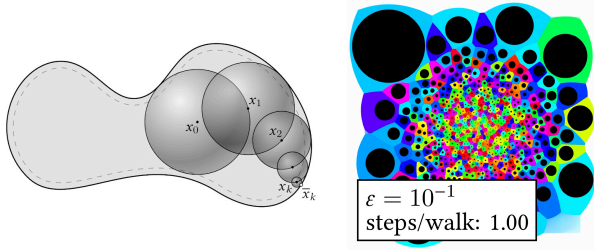
Walk on spheres converges predictably



- In contrast, WoS demonstrates very predictable convergence on all 10000 models in the same dataset.

Stopping tolerance ϵ

Introduces minimal bias and has little impact on performance

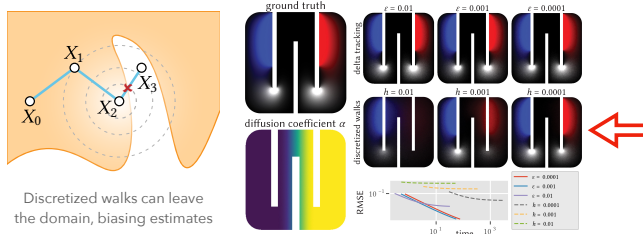


43

- Like standard WoS, the only parameter in our algorithms is an epsilon tolerance that specifies how close to the boundary we have to be before we can grab the known boundary value.
- This tolerance introduces minimal bias and has little impact on performance unlike tolerances in meshing algorithms.

Discretized random walks

Explicit time stepping of diffusion process: $X_{k+1} = X_k + \bar{\omega}(X_k) h + \sqrt{\alpha(X_k)} (W_{k+1} - W_k)$

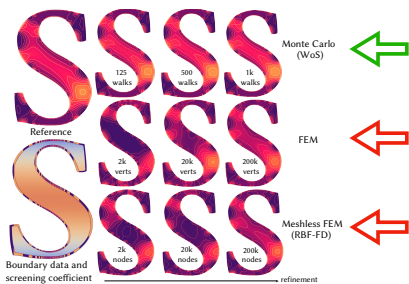


44

- Like ray marching, Feynman Kac can be directly approximated by simulating a diffusion process with explicit time stepping.
- Unlike WoS however, discretized walks can leave the domain, which biases the solution estimates.
- Smaller time steps help reduce this bias, but at significant detriment to run-time performance.

No spatial aliasing

Monte Carlo decouples boundary conditions/coefficients from geometry



45

- An extra benefit of Monte Carlo is that it decouples the boundary conditions and coefficients from the geometry.
- As a result, there is never any spatial aliasing, and WoS is able to capture the global profile of the solution with just a few walks.
- In contrast, conventional methods have to heavily refine the discretization to capture high frequency inputs.
- In general, it's very difficult to predict an adequate mesh size ahead of time.

Physical analysis of complex geometry & materials

No homogenization of PDE coefficients!

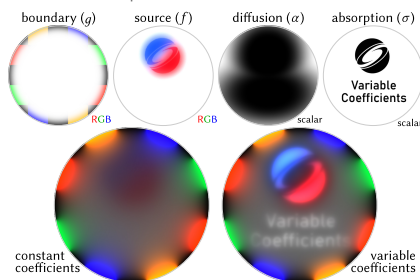


46

- Though our main goal here is to develop core knowledge, there are some cool things we can immediately do.
- One is to simply solve physical PDEs with complex geometry and coefficients.

Example application: variable coefficient diffusion curves

Additional control over sharp details



47

- A graphics example is to generalize so-called "diffusion curves" to variable coefficients, giving more control over how sharp or fuzzy details look.

Example application: diffusion curves on surfaces

Use variable coeffs on flat domains to model constant coeffs on curved domains

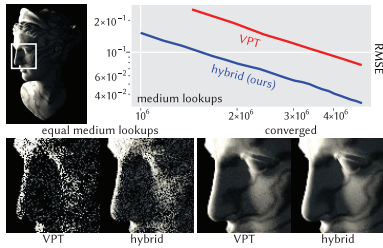


48

- Another nice point of view is that variable coefficients on a flat domain can actually be used to model constant coefficients on a curved domain!
- This way, we can solve PDEs with intricate boundary data on smooth surfaces, without any meshing at all.

Example application: subsurface scattering

Easy to mix volumetric path tracing (VPT) and walk on spheres (WoS)



Hybrid strategy : VPT near boundary, WoS deeper inside volume

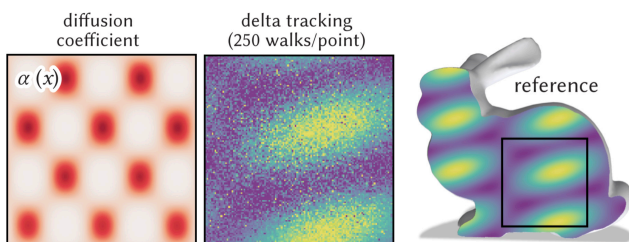
69

- A Monte Carlo method also makes it easy to integrate PDE solvers with physically-based renderers.
- For instance, we can get a way more accurate diffusion approximation of heterogeneous subsurface scattering, without having to painfully hook up to a FEM or grid-based solver.

Limitations & Future Work

- Our method is not without limitations.

High variance due to large spatial variation



Future: local coefficient bounds, low-variance VRE estimators, adaptive weight window

71

- As in rendering, coefficients with large spatial variation can lead to increased variance.
- Adapting further techniques from volume rendering such as local coefficient bounds, low-variance VRE estimators, and adaptive weight windows should help address this issue.

Future: support for important features

Neumann & Robin boundary conditions

Anisotropic diffusion coefficients

Non-linear PDEs

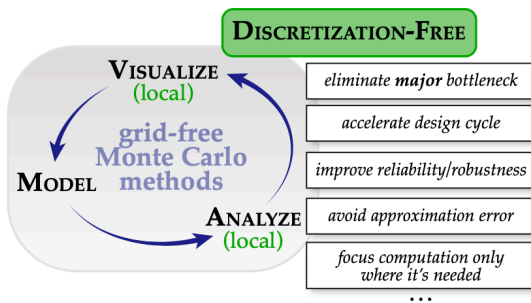
High performance distance queries

Differentiable implementation

72

- More broadly, the WoS framework still lacks support for many basic features of schemes like FEM, such as Neumann boundary conditions and anisotropic diffusion coefficients.
- That said, this framework is still a very interesting [fairly new] way to solve PDEs, with deep-but-unexplored connections to rendering.

The promise of grid-free Monte Carlo



73

- Moreover, since Monte Carlo methods are free from the bonds of spatial discretization, they open the door to new classes of PDE solvers that are robust to bad geometry, scalable to extremely large scenes, and progressive in their solution evaluation.

SIGGRAPH 2022
VANCOUVER • 8-11 AUG



GRID-FREE MONTE CARLO FOR PDES WITH SPATIALLY VARYING COEFFICIENTS

ROHAN SAWHNEY*, CARNEGIE MELLON UNIVERSITY
DARIO SEYB*, DARTMOUTH COLLEGE
WOJCIECH JAROSZ†, DARTMOUTH COLLEGE
KEENAN CRANE†, CARNEGIE MELLON UNIVERSITY

The symbols * and † indicate equal contribution.

© 2022 SIGGRAPH. ALL RIGHTS RESERVED.

- Thank you.

BACKUP

35

I don't know...

D'oh!



36