Parallel sampling for computing rare event probabilities, inverse Bayesian inference and uncertainty quantification with MOOSE

IPAM, Exascale Workshop
Multiphysics Object-Oriented Simulation Environment (MOOSE)

A DoE sponsored finite element / finite volume based modeling framework
Idaho National Laboratory Position Nationally

- One of 9 large DOE multi-program Labs
- DOE’s Lead Lab for Nuclear Energy
- ~ 5,700 employees
- ~ 900 mi\(^2\)

INL is advancing clean, safe, and secure energy for the future
Reactor Modeling requires Multiphysics

• Reactors are inherently nonlinear, multiscale, multiphysics problems
  – Neutron transport, heat conduction, solid-mechanics, fluid flow, material degradation, chemistry, corrosion, etc.

• Predictive reactor simulation requires multiphysics (always has)
  – Doppler broadening, moderator density, etc.

• Licensing of advanced reactors will heavily rely on multiphysics mod/sim
  – Uncertainty analysis, accident scenarios, refueling, material selection, etc.

Pebble bed PLOFC/DLOFC calculations using Pronghorn, Griffin, MOOSE-THM, MOOSE-HC
Credit: Sebastian Schunert, Guillaume Guidicelli, Alexander Lindsay, Paolo Balestra
Materials Microstructure and Properties

- A key objective of materials science is to understand the impact of microstructure on macroscale material behavior.

- An essential part of that is predicting the impact of microstructure evolution.
Material Behavior is Multiphysics

• Material behavior is influenced by many different physics, for example:

**Mechanics**
- Dislocations
- Cracking
- Stress-driven Diffusion

\[ \nabla \cdot (\sigma + \sigma_0) + b = 0 \; \text{in} \; \Omega \\
\nabla \cdot g = g \; \text{in} \; \Gamma_g \\
\sigma \cdot n = t \; \text{in} \; \Gamma_t \]

**Chemistry**
- Corrosion
- Oxidation
- Reactive transport

\[ \frac{\partial u}{\partial t} = D_u \nabla^2 u - w u + F(1 - u), \]
\[ \frac{\partial v}{\partial t} = D_v \nabla^2 v + w v^3 - (F + k)v. \]

**Electricity/Magnetism**
- Electromigration
- Ferroelectricity
- Ferromagnetism

\[ \nabla \cdot D = \rho \]
\[ \nabla \cdot B = 0 \]
\[ \nabla \times E = -\frac{\partial B}{\partial t} \]
\[ \nabla \times H = J + \frac{\partial D}{\partial t} \]

**Heat Conduction**
- Species transport
- Melting
- Precipitation

\[ \frac{\partial U}{\partial t} = \nabla^2 U \]
MOOSE is a finite element, multiphysics framework that simplifies the development of advanced numerical applications.

- It provides a high-level interface to sophisticated nonlinear solvers and massively parallel computational capability.

MOOSE is in use across the world to solve:

- Phase field
- Solid mechanics
- Heat conduction
- Neutronics
- Comp. fluid dynamics
- Stochastic modeling
- Thermal hydraulics
- Geomechanics
- Reactive transport
- Corrosion
- Crystal plasticity
- Fracture
- Porous flow
- Electromagnetics
MOOSE Development and Community

• **Active development on GitHub**
  - Transparent development process
  - >850 git forks, >300 unique cloners / 2wk, 1.1k⭐
  - >9,000 closed issues

• **Active User Community**
  - Hosted on GitHub Discussions
  - 1000s of views / 100-200 unique visitors / day
  - Hundreds of unique users contributing on Discussions each month
MOOSE Testing and Continuous Integration

- Continuous integration and testing
  - Public: https://civet.inl.gov
  - Continuous integration with ~10,000 tests in framework and modules
  - Tested in various...
    - ...parallelization schemes
    - ...mesh modes, AD modes
    - ...compilers, operating systems, CPU architectures
    - ...documentation and code coverage
  - Test on HPC
    - pull requests (before code is merged)
    - master merge (production code) (Valgrind, parallel sweeps)
  - Test >50 internal and external apps
  - Going on 3k cores for CI (Linux/mac, Arm/x64)

Yes, we can test YOUR app!
Some MOOSE-Based Applications

- **Bison**
  - Nuclear Fuel Performance

- **Grizzly**
  - Structural Mechanics for Component Aging

- **Griffin**
  - Radiation Transport

- **Pronghorn**
  - Medium-fidelity CFD

- **Marmot**
  - Mesoscale Materials

- **Sockeye**
  - Heat pipe Simulation

- **Bioinspired Vascular Networks**
  - Frontal Polymerization

- **Solid-state Electrolytes**
  - Electrical Conductivity

The MOOSE “Framework”

- MOOSE is not an “application”, but a “framework” that provides
  - C++ infrastructure to write your own...
    - physics (PDE weak forms)
    - Initial and boundary conditions
    - On-the-fly postprocessing (e.g. max/min concentration, precipitate counting, line scans)
  - An input file parser to combine the MOOSE C++ objects into a simulation at runtime, and add/configure
    - Variables
    - Solver and solver parameters
    - Outputs

```plaintext
[Mesh]
  type = GeneratedMesh
  dim = 2
  nx = 10
  ny = 10
[]

[Variables]
[ ]
[
]
[Kernels]
[diff]
  type = Diffusion
  variable = u
[]
[
]
[BCs]
[left]
  type = DirichletBC
  variable = u
  boundary = left
  value = 0
[]
[right]
  type = DirichletBC
  variable = u
  boundary = right
  value = 1
[]
[
]
[Executioner]
  type = Steady
  solve_type = 'PJFNK'
  petsc_options_iname = '-pc_type -pc_hypre_type'
  petsc_options_value = 'hypre boomeramg'
[]
[Outputs]
  exodus = true
[]
```
Implementing your physics in MOOSE

1. Derive your PDE term weak form
2. Create small boilerplate C++ .C and .h file
3. Type in weak form

\[ \tilde{v} \nabla u - \frac{\partial u}{\partial t} = 0 \]

\[ (\psi, \tilde{v} \nabla u) - (\psi, \frac{\partial u}{\partial t}) = 0 \]

```cpp
registerMooseObject("MyAwesomeApp", ADAdvection);

InputParameters ADAdvection::validParams()
{
  InputParameters params = ADKernel::validParams();
  params.addRequiredParam<MooseObjectPropertyName>("velocity_vector", "Advection velocity vector");
  return params;
}

ADAdvection::ADAdvection(const InputParameters & parameters) 
: ADKernel(parameters),
  _v(getADMaterialProperty<RealVectorValue>("velocity_vector"))
{}

ADReal ADAdvection ::computeQpResidual()
{
  return _test[_i][_qp] * _v[_qp] * _grad_u[_qp];
}
```
Mesoscale Materials Modeling

Brief overview over MOOSE mesoscale modeling capabilities
- Provides the tools necessary for modeling mechanical deformation and stress at the mesoscale.
- Strain (small, finite, incremental)
- Eigenstrains / Eigenstresses
- Elastic stress, Modular inelastic stress system
  - Creep
  - Plasticity (J2, Crystal Plasticity)
  - Fracture
- Strain periodicity (representative volume elements)
- Fully couplable to phase field
Phase Field Module

- Provides all the tools necessary to develop a massively parallel phase field code using FEM.
- Base classes for solving Cahn Hilliard equations
  - Direct solution / Split solution
  - Grand Potential, KKS
- Base classes for Allen-Cahn equations
- Grain growth model
- Grain remapping algorithm for efficient models
- Initial conditions
- Postprocessors for characterizing microstructure
- Nucleation
Phase Field Examples

Grain growth model (6000 grains) with OP remapping 150 million DOFs on ~580 CPU cores

Grain boundaries decorated with growing gas bubbles. (Larry Aagesen)
Void lattice formation due to anisotropic interstitial diffusion. (D. Schwen)

Local laser melting – heat transport, ablation, and Navier-Stokes using ALE. (Alexander Lindsey)
Simulating radiation and microstructure

atomic scale

1 Å

mesoscale

10 nm–1 μm
Nanoscale ballistic mixing

Coupling to FFTW3 to get Fourier space data, such as power spectrum and characteristic length scale.

Patterning, mixing, coarsening in an immiscible alloy under irradiation. Axes are displacement rate and distance.

- Phase separating immiscible alloy (CuAg)
- 100,000s of recoil cascades run (50 keV Xenon ions)
- Steady-state length scale
- Applications: fission-gas bubbles, FeCrAl precipitates
Compound Energy Formalism

- No sublattices / internal DOFs
  - Directly export free energy expressions
- Solve local equilibrium
  - Concurrently couple thermodynamic software (OpenCalphad, pycalphad, ThermoCalc)
  - Parabolic fits
  - Tabulate free energy (and chemical potentials) in state space
    - Polyadic Tensor decomposition (Moelans, KU Leuven) \(\rightarrow\) MOOSE
    - Neural Networks
Fitting a neural net to TDB data

- Universal Approximation Theorem
  A feed-forward network with a single hidden layer containing a finite number of neurons can approximate continuous functions on compact subsets of $\mathbb{R}^n$

- PyTorch
  - Mini batch learning
  - Cost function includes chemical potentials $dF/dc_1$ etc.
  - Initial input/output weight/bias guess no manual normalization required
  - $g$: SoftSign, Sigmoid, tanh

- Weights & biases → Text file → MOOSE

- Derivative of NN → chemical potential

$$F = \left[ \left( \begin{array}{c} c_1 \\ c_2 \\ T \end{array} \right) \cdot \vec{W}_1 + \vec{b}_1 \right]_g \cdot \left( \begin{array}{c} \vec{W}_2 + \vec{b}_2 \\ \vec{W}_3 + \vec{b}_3 \end{array} \right)$$
Fitting data from a regular solution free energy

\[ f = c(1 - c) + 10^{-3}T(c \log c + (1 - c) \log(1 - c)) \]
Sublattice Kim-Kim-Suzuki Model

Original KKS Model

\[ F = \sum_j h_j F_j \]

Free energy is weighted sum of phase \( F_j \)

\[ c_i = \sum_j h_j c_{ij} \]

Physical concentration is weighted sum of phase \( c_{ij} \)

\[ \frac{\partial F_j}{\partial c_{ij}} = \frac{\partial F_{j'}}{\partial c_{ij'}} = \mu_i \]

Equal chemical potential in phase \( j \) and \( j' \)

New Sublattice KKS Model

\[ F = \sum_j h_j F_j \]

Free energy is weighted sum of phase \( F_j \)

\[ g = -c_{ij} + \sum a_{jk} c_{ijk} \]

\[ \nabla_{c_{ijk}} F = \lambda \nabla_{c_{ijk}} g \]

Physical concentration is weighted sum of sublattice \( c_{ijk} \)

\[ 1 \frac{\partial F_j}{a_{jk} \partial c_{ijk}} = 1 \frac{\partial F_{j'}}{a_{j'k'} \partial c_{ij'k'}} = \frac{\partial F_j}{\partial c_{ij}} = \mu_i \]

Equal chemical potential between sublattices, derived from Lagrange multiplier constrained minimization of \( F_j \)

D. Schwen, et al., A sublattice phase-field model for direct CALPHAD database coupling, Comp. Mat. Sci., 195 (2021) 110466
Internal DOF minimization – Phase free energy

- Solving PDEs for constrained minimization of $\delta$-UZr internal DOFs
- Obtain effective phase free energy for $\delta$-UZr
- Phase chemical potential can be derived as a function of the sublattice chemical potential

$$\frac{\partial F_j}{\partial c_{ij}} = \frac{1}{a_{jk}} \frac{\partial F_j}{\partial c_{ijk}}$$
UZr phase field

- Ready to run the SLKKS phase field model
- Require multi phase formulation for appropriate switching functions $h_j$
- Thermodynamically consistent model for three phases implemented in MOOSE
- Formulations for $n>3$ phases exist, Implementation WIP
- U/Zr bilayer
- Interdiffusion and formation of $\delta$-UZr

(Interfacial free energies 10mJ/m$^2$ and mobilities not based on physical values, all phases have the same atomic volume)
Mo-Ni-Re phase field

- Mo-Ni-Re High temperature Ni-based superalloy
  J.-C. Crivello, R. Souques, A. Breidi, N. Bourgeois, J.-M. Joubert,
  Calphad 51(2015) 233-240

- Ternary system

- Five(!) sublattices in the $\sigma$-phase
  $(\text{Mo,Ni,Re})_2(\text{Mo,Ni,Re})_4(\text{Mo,Ni,Re})_8(\text{Mo,Ni,Re})_8(\text{Mo,Ni,Re})_8$ (Mo,Ni,Re)

- Transient simulation
  Not at equilibrium

- Interfacial energy impacts bulk concentration
  (Interfacial free energies 10mJ/m$^2$ and mobilities not based on physical values, all phases have the same atomic volume)

Phase separated microstructure evolved from a homogeneous initial composition of Mo$_3$Ni$_{10}$Re$_7$, which is located in the hcp, fcc, and $\sigma$-phase coexistence region.
Explicit Nucleation - Modifying free energy density

- Insert nucleation sites base on rate density
- Modulate free energy at nucleation sites (additive contribution)
- Driving force $\frac{\partial F_n}{\partial c}$ coerces precipitate to form
  - Can be applied to conserved or non-conserved order parameters

Notice the vanishing nucleation energy penalty $F_n$ as the nucleus forms. In this example $c_{Cu}$ is controlled and $c_{Ni}$ follows automatically.
Nucleation example

Preemptive mesh refinement, timestep limited by total nucleation probability, and cut-back at each nucleation event.
How could MOOSE utilize Exascale Computing Resources?
MOOSE Scalability

- MOOSE currently runs on CPUs
  - Hypre non-linear solver has GPU support, but that comes with lots of data transfers CPU↔GPU – for now)
- We are currently working with the MFEM team to define a path forward that will allow MOOSE-based applications to more directly utilize GPUs
**MultiApps: Enabling Multiscale Simulation**

- MOOSE-based solves can be nested to achieve Multiscale-Multiphysics simulations
  - Macroscale simulations can be coupled to embedded microstructure simulations
- Arbitrary levels of solves
- Each solve is spread out in parallel to make the most efficient use of computing resources
- Efficiently ties together multiple teams’ codes
- MOOSE-wrapped apps (wrap arbitrary third party codes, e.g. NEK5000 -> Cardinal)
MOOSE design goals vs. Exascale

- User (Developer, Researcher, and Analyst) time is precious
  - Need versatile simulation tools
  - Easy to implement new physics
  - Hardware agnostic

- Virtually nobody is doing “Exascale computing” yet
  - neither are we
  - but we should be forward looking

- Do we even know what Exascale computing looks like?
  - Frontier ~1.1 EFLOPS
    heterogeneous: half a million CPU cores…
  - Fugaku <0.5 EFLOPS
    7 million CPU cores, no accelerators
  - El Capitan
    AMD MI300, zero-copy

Figure: Apositakis et al., doi: 10.3389/fphy.2022.913510
Utilizing Exascale resources

- Parallelizing to extend scales
  - Hard to get good *strong scaling* important for extended time scales
  - Not always easy to get good *weak scaling* important for extended length scales

- How about improving accuracy and *confidence*?

- Improve accuracy using multiscale modeling (MultiApps)
  - Expensive to run many HF/LLS models

- Improve confidence through UQ with stochastic methods (MultiApps)
  - MCMC as the gold standard for UQ is expensive

- Detect rare events / failure probability
Failure probability analysis
TRISO particles for Advanced Nuclear Reactor

TRISO stands for Tri-structural ISOtropic particle fuel. Reactor vendors such as Kairos Power, X-energy, BWXT, USNC, Westinghouse, Radiant are planning to use TRISO fuel for their small modular and micro-reactor designs.

Each pebble contains ~10,000 TRISO particles.

Reactor core contains ~360,000 pebbles.
TRISO Failure Analysis using Monte Carlo Simulation

Perform Monte Carlo simulation
Sampling parameters

Run 1D simulation
At each time step

Check IPyC cracking

Check SiC pressure vessel failure
(adjust stress to account for asphericity)

Check SiC failure due to IPyC cracking

Determine SiC failure

\[ \sigma_{\text{correlation}} > \text{strength sampled from Weibull} (\sigma_{\text{m}}, m) \]

YES

Last sampling?

NO: next time step

YES

Compute Statistics

Metropolis Monte Carlo Sampling

<table>
<thead>
<tr>
<th>Category</th>
<th>Parameter</th>
<th>Nominal values ± Standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kernel diameter (µm)</td>
<td></td>
<td>425±10</td>
</tr>
<tr>
<td>Buffer thickness (µm)</td>
<td></td>
<td>100±10</td>
</tr>
<tr>
<td>Particle geometry</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IPyC/OPYC thickness (µm)</td>
<td></td>
<td>40±3</td>
</tr>
<tr>
<td>SiC thickness (µm)</td>
<td></td>
<td>35±2</td>
</tr>
<tr>
<td>Particle asphericity (SiC aspect ratio)</td>
<td></td>
<td>1.04</td>
</tr>
<tr>
<td>Fuel properties</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IPyC density (g/cm³)</td>
<td></td>
<td>1.90±0.02</td>
</tr>
<tr>
<td>OPyC density (g/cm³)</td>
<td></td>
<td>1.90±0.02</td>
</tr>
<tr>
<td>IPyC/OPYC BAF</td>
<td></td>
<td>1.05±0.005</td>
</tr>
</tbody>
</table>

Wen Jiang, INL
TRISO Failure Analysis using Monte Carlo Simulation

Aspherical (2D)

Layer cracking

Correlate 2d to 1d (stress factors)

2D simulations are used to create a 1D surrogate based on stress correlation factors

100 million 1D samples with 7600 cores on INL HPC

Wen Jiang, INL
Fuel elements modeling with MOOSE’s Multi-App system

Material property homogenization

Main App is 3D homogenized matrix

Each TRISO particle is solved individually as a sub-app

Efficient two-way data transfer

Wen Jiang, Yifeng Che, INL
Fission product diffusion in a pebble

Radius (cm)  2.000
Shell layer thickness (cm)  0.200
Fuel layer thickness (cm)  0.420
(AGR-5/6/7) TRISOs  9022
U-235 Enrichment (% wt)  19.55

Shell (graphite)
Fuel region
Low density graphite

Heat point source
Cs point source

9012 intact particles
10 failed particles
Exascale opportunities

- Failure probability calculation
  - Higher fidelity lower scale models 1D → 2D → 3D
    - Capture asphericity
    - Capture failure modes
      - Crack formation
    - More advanced sampling methods
      - Parallel Subset Sampling (in MOOSE)
      - Surrogate models with active learning (LDRD)

- Multiscale coupling
  - Higher fidelity lower scale models 1D → 2D → 3D
  - On the fly particle failure
Bayesian Inference
State of practice of computational model validation

Step ①: model and inputs

Input parameters:
- Fabrication
- Material
- System

MOOSE

Step ②: validation and calibration

Experiment
Calibrated model

Step ③: forward prediction

Key questions:

Input parameter uncertainties (UO₂ material)

Model prediction uncertainties (TRISO fuel Ag release)

Limited experimental data (Heat pipe reactor)

Bayesian calibration and UQ of computational models

- Identifies and propagates parameter uncertainties
- Reflects model mismatches with experiments during forward simulations
- Permits Bayesian optimal experimental design

(References: Wu et al. 2021, Hoff 2009, Che et al. 2021)
Bayesian inference with computational models

\[ f(\theta|x, y) \propto L(y|x, \theta) f(\theta) \]

\[ f(\theta): \text{Prior of the uncertain model parameters} \]

\[ L(y|x, \theta): \text{Likelihood of observing the data given model inputs, parameters, and predictions} \]

\[ f(\theta|x, y): \text{Posterior of the uncertain model parameters given model inputs and predictions} \]

\[ f(\theta): \text{Any user-specified distribution (uniform, bounded Normal)} \]

\[ L(y|x, \theta) \propto \frac{1}{\sigma} \prod_{k=1}^{K} \exp \left( - \frac{(y_k(x) - \hat{y}_k(x, \theta))^2}{2\sigma^2} \right) \]

Experimental value
Model prediction
Model deviation + experimental noise

Inferred params include \( \theta + \sigma \)
Computational aspects: serial and parallel MCMC

Standard Metropolis-Hastings

- Propose a new input
- Evaluate model and likelihood
- Compute the TPM, $A(i, j)$
- Draw an integer b/w [0, 1] with weights $A(i, j)$
- Assign new input

$M$ model evaluations in serial (impractical for us)

Parallel Metropolis-Hastings

- Propose $N$ new inputs
- Evaluate models and likelihoods (parallelization)
- Compute the TPM, $A(i, j)$
- Sub-sampling (generate $N$ new points):
  - Draw an integer b/w [0, N] with weights $A(i, j)$
  - Assign new input

$M/N$ model evaluations in serial

Computational aspects: parallel MCMC

- $N$ can be between 1 and 1000 (massive parallelization)
- $N=1$, standard serial MCMC
- Proposal kernel is flexible: random-walk (adaptive, delayed rejection), Langevin, Hamiltonian
- Improved performance compared to serial M-H
- Limitations??

Algorithm by Calderhead 2014

MOOSE implementation

MCMCBase → SpecificMCMC

SpecificMCMC → MultiApp

MultiApp → SubApp 1

SubApp 1 → SubApp 2

SubApp 2 → SubApp N

SubApp N → SpecificMCMCDecision

SpecificMCMCDecision → MCMCDiagnostics

MCMCDiagnostics → MCMCDecisionBase

(batch of input samples)

(outputs)

Executed in parallel

Effective sample size (FFT), R-hat
Case Study: TRISO AGR-2 experiments

- Advanced Gas Reactor Bison model prediction: Ag release
- 36 Post Irradiation Examination (PIE) data for UCO fuel kernel
- Considerable uncertainties in model predictions compared to experiments
- Model params: Pre-factor A and activation energy $E_a$ for SiC, PyC, and fuel kernel (6 params)
- Experimental configurations: T, fluence, heat rate (time varying)
Case Study: TRISO AGR-2 experiments

Inverse analysis

\[ f(\theta|x, y) \propto L(y|x, \theta) f(\theta) \]

Infer 6 params (pre-factor/activation E)

Sigma fixed to 0.1

- 50 parallel proposals
- Each proposal requires 36 model evals: 36 experimental configurations
- Analyzed using 1800 procs on Sawtooth
- Lower bounds: \(5 \times 10^{-10}; 165 \times 10^2\)
- Upper bounds: \(5 \times 10^{-8}; 165 \times 10^4\)
- Proposal stds: \(3 \times 10^{-9}; 5 \times 10^3\)
Case Study: TRISO AGR-2 experiments

Inverse analysis

Deterministic
(Bison report FY 22)
Case Study: TRISO AGR-2 experiments

Forward analysis

Sigma fixed to 0.1
Uncertainty in the uncertainty estimation

Sampling quality

Total samples: 65,000
Acceptance rate: 12%
Effective sample size (ESS): 6.99857546, 11.38694062, 4.91145346, 9.7290921, 2.95958647, 3.36655714

Parallelization of MCMC is still better than serial MCMC
**Affine Invariant Differential Evolution Sampler (AIDES)**

- Uses two states other than the current state from the ensemble of walkers to propose a differential step.

- This differential step is added to the current state to become the new proposal for the walker.

- Procedure repeated across all ensemble of walkers.

- Mathematically, the differential component of the proposal is ($\gamma, \xi$ are the internal params)

\[
\delta X = \gamma (X^a - X^b) + N(0, \xi)
\]

(Cajo and Braak 2006; Vrugt 2016)
AIDES applied to TRISO FGR AGR-2

Independent Metropolis-Hasting

Effective sample sizes:
3.89, 4.44,
24.41, 4.97,
3.79, 4.40

Differential Evolution

Effective sample sizes:
54668.42, 49552.78,
53355.39, 46674.88,
54900., 49619.29

• 55,000 total samples with 50 parallel proposals
Effective parallelization
• 1D Ag diffusion through the particles
• Better exploration of parameter space with Affine Invariant Differential Evolution Sampler

Som Dhulipala, INL
Exascale opportunities

• Parallel sampling
  – Parallelize over experimental data points
  – Parallelize individual model evaluations
  – Parallel Markov Chains

• Infer model parameters with uncertainties from complex experiments (e.g. Taylor impact test)

• ToDo: Gradient based methods
  – Gradient computation requires software retooling and comes with a substantial cost
  – No established benchmarks for parallel gradient based MCMC
Summary

Materials Modeling

- Rapid model development
- Multiphysics coupling
- Effortless parallelism

Parallel multiscale architecture

- Sub-application system
- All data in memory
- Wrap external codes
- Failure probability / rare events sampling

Stochastic Tools

- Parallel MCMC
- Inverse and forward Bayesian inference
Thank you! Questions?