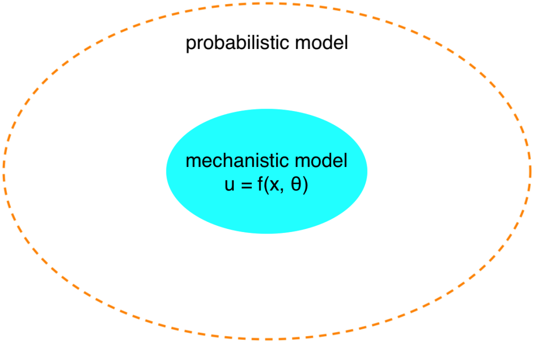
# Solving ODEs in a Bayesian Model

$F_\omega(\alpha + m)!$ 2022

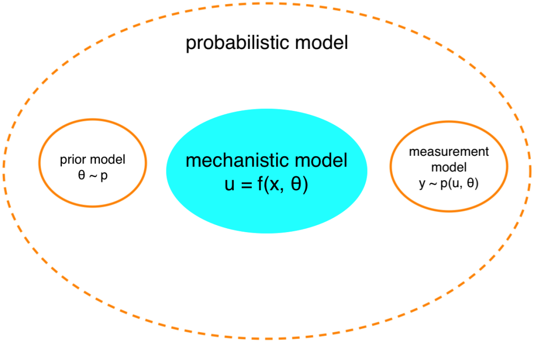
Charles Margossian

- Outline:
  - ODE-based Bayesian models
  - Solving ODEs across the parameter space
  - Propagating derivatives through ODEs

# Integrating mechanistic models inside probabilistic models



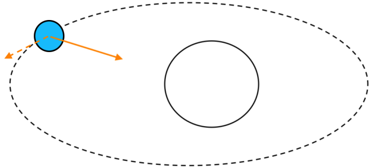
## Integrating mechanistic models inside probabilistic models




- Goal: learn the latent variables via $p(\theta \mid y)$.

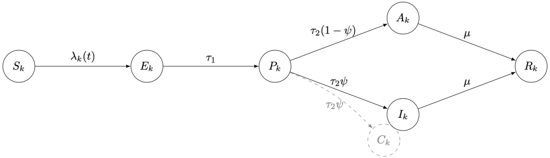
## Example 1: "textbook" model of planetary motion[1]

- $k$: star-planet gravitational interaction
- position $q(t, k)$ and momentum $p(t, k)$ solve Hamilton's equations.
- observations: $y(t) = q(t, k) + \epsilon$, with $\epsilon \sim \text{normal}(0, \sigma^2)$.



1: Gelman et al. (2020) *Bayesian Workflow*, preprint

# Example 2: SEIR model for Covid-19 to estimate mortality rate[2,3]




Mechanistic model:

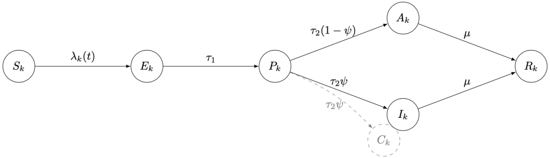
- ODE model for disease transmission, stratified by age.

---

2: Riou et al. (2020) *Estimation of SARS-CoV-2 mortality during the early stages of a pandemic*, PLOS Medicine

3: Grinsztajn et al. (2021) *Bayesian workflow for disease transmission*, Statistics in Medicine

# Example 2: SEIR model for Covid-19 to estimate mortality rate[2,3]




Mechanistic model:

- ODE model for disease transmission, stratified by age.

Probabilistic model:

- Combine multiple (biased) data sources
- Use prior on symptomatic rate for identifiability

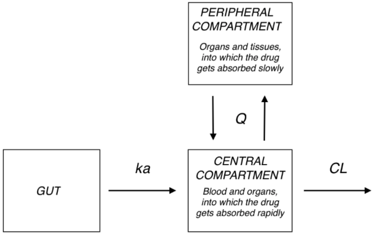
---

2: Riou et al. (2020) *Estimation of SARS-CoV-2 mortality during the early stages of a pandemic*, PLOS Medicine

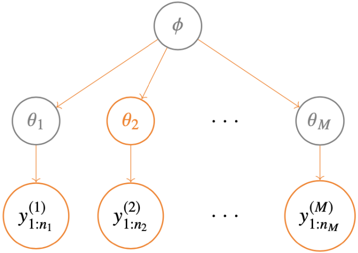
3: Grinsztajn et al. (2021) *Bayesian workflow for disease transmission*, Statistics in Medicine

# Example 3: Population pharmacokinetic model[4,5]

## Pharmacokinetic model:




## Hierarchical model:

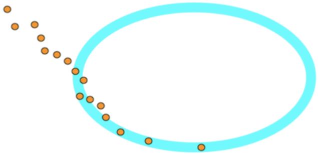



---

4: Wakefield (1996) *The Bayesian analysis of pop. PK models* JASA

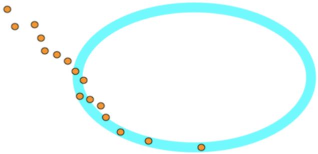
5: M et al. (2022) *Pharmacometrics modeling using Stan and Torsten...* CPT: Pharmacometrics & Systems Pharmacology

Bayesian inference: probing $p(\theta \mid y)$ with MCMC



- Probabilistic programming languages are expressive and allow us to specify ODE-based likelihoods (e.g. `Stan`, `Turing`, `TensorFlow Probability`, ...)

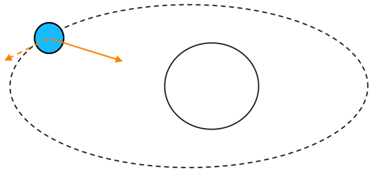
## Bayesian inference: probing $p(\theta \mid y)$ with MCMC



- Probabilistic programming languages are expressive and allow us to specify ODE-based likelihoods (e.g. `Stan`, `Turing`, `TensorFlow Probability`, ...)
- Across the parameter space:
  - Evaluate the likelihood, $\log p(y \mid \theta)$.
  - Evaluate the gradient, $\nabla_\theta \log p(y \mid \theta)$.

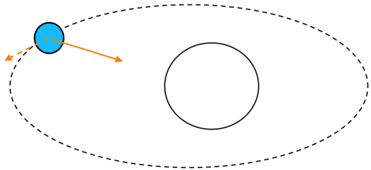
- Outline:
  - ODE-based Bayesian models
  - Solving ODEs across the parameter space
  - Propagating derivatives through ODEs

Model of planetary motion

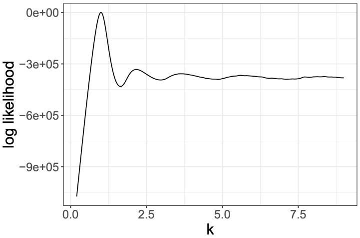


| chain | run time (s) |
|---|---|
| 1 | 10.56 |
| 2 | 3.40 |
| 3 | 4433.93 |
| 4 | 181.98 |

# Model of planetary motion



| chain | run time (s) |
|---|---|
| 1 | 10.56 |
| 2 | 3.40 |
| 3 | 4433.93 |
| 4 | 181.98 |



Model of planetary motion

$$\begin{aligned} q'(t) &= \frac{p}{m} \\ p'(t) &= -\frac{k}{r^3}(q - q_\odot) \end{aligned}$$

## Model of planetary motion

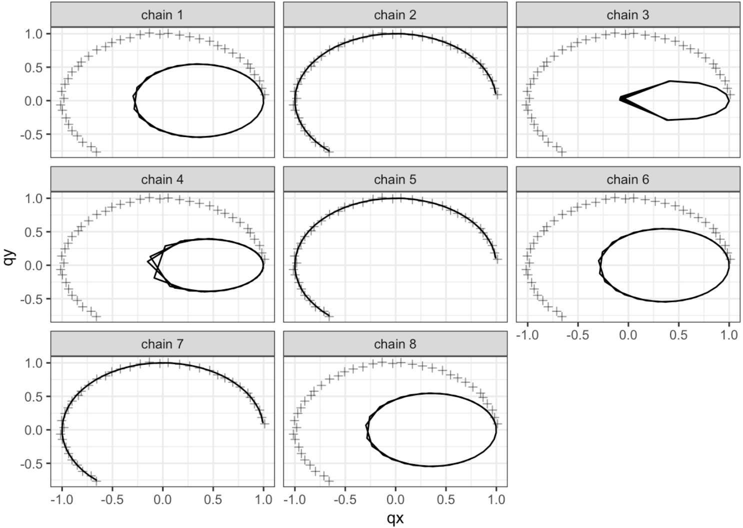
$$\begin{aligned} q'(t) &= \frac{p}{m} \\ p'(t) &= -\frac{k}{r^3}(q - q_\odot) \end{aligned}$$

- use a numerical integrator.
  e.g. Euler's method (for simplicity):

$$u(t+\epsilon) \leftarrow u(t) + \epsilon u'(t+\epsilon)$$

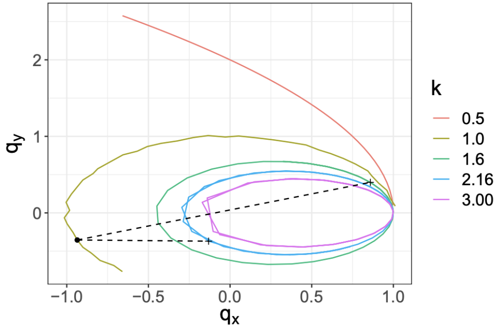
  - Error: $\mathcal{O}(\epsilon^2 |u''(t)|)$
  - In practice, set the error tolerance and tune $\epsilon$.

# Posterior predictive checks



$$\log p(y \mid \theta) = K + \frac{1}{2\sigma^2} \sum_{t=t_0}^{\tau} (u(t) - y_t)^2$$

## Potential Fixes

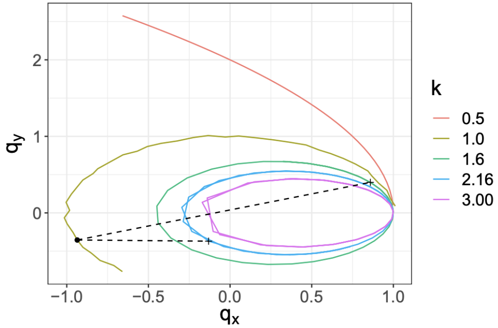



---

[6]: Gabrié et al (2022) *Adaptive Monte Carlo augmented with normalizing flows*, PNAS

[*]: Kaze (2022), *FlowMC*, https://github.com/kazewong/flowMC

$$\log p(y \mid \theta) = K + \frac{1}{2\sigma^2} \sum_{t=t_0}^{\tau} (u(t) - y_t)^2$$




Potential Fixes

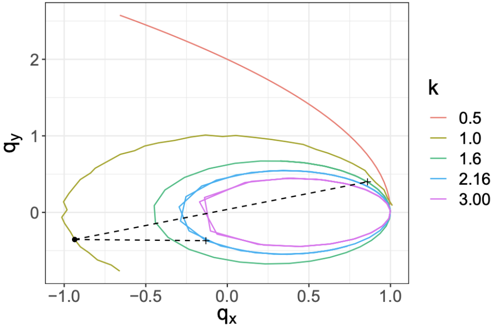
- Use a stronger prior (still multimodal)
- Correct samples in post-processing step
- Don't rely on default initializations! Draw inits from prior.
- Use "global" algorithms:
    - tempering
    - MCMC with normalizing flow[6,*]
    - "momaVI"

---

[6]: Gabrié et al (2022) *Adaptive Monte Carlo augmented with normalizing flows*, PNAS

[*]: Kaze (2022), *FlowMC*, https://github.com/kazewong/flowMC

$$\log p(y \mid \theta) = K + \frac{1}{2\sigma^2} \sum_{t=t_0}^{\tau} (u(t) - y_t)^2$$




## Potential Fixes

- Use a stronger prior (still multimodal)
- Correct samples in post-processing step
- Don't rely on default initializations! Draw inits from prior.
- Use "global" algorithms:
  - tempering
  - MCMC with normalizing flow.
  - "momaVI"

What makes a good initialization for MCMC?

7: Zhang et al (2021) *Pathfinder: quasi-Newton VI*, preprint
8: Gelman et al. (2013) *Bayesian Data Analysis*, textbook
9: M et al. (2022) *Nested $\widehat{R}$: assessing convergence*, preprint

## What makes a good initialization for MCMC?

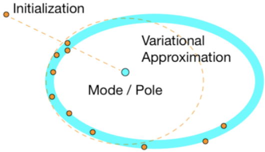
- Draw from the prior, $p(\theta)$.

---

7: Zhang et al (2021) *Pathfinder: quasi-Newton VI*, preprint
8: Gelman et al. (2013) *Bayesian Data Analysis*, textbook
9: M et al. (2022) *Nested $\widehat{R}$: assessing convergence*, preprint

# What makes a good initialization for MCMC?

- Draw from the prior, $p(\theta)$.
- Draw from an approximation $q(\theta) \approx p(\theta \mid y)$, e.g. using a



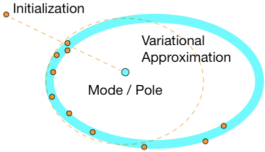

*pathfinder.*[7]

---

[7]: Zhang et al (2021) *Pathfinder: quasi-Newton VI*, preprint
[8]: Gelman et al. (2013) *Bayesian Data Analysis*, textbook
[9]: M et al. (2022) *Nested $\widehat{R}$: assessing convergence*, preprint

# What makes a good initialization for MCMC?

- Draw from the prior, $p(\theta)$.
- Draw from an approximation $q(\theta) \approx p(\theta \mid y)$, e.g. using a *pathfinder*.[7]




- Draw *overdispersed* initializations to make convergence diagnostics such as $\widehat{R}$ reliable.[8]
- How much overdispersion do we need? For unimodal target, the initial variance must be lower-bounded by a linear function of the initial squared bias.[9]

7: Zhang et al (2021) *Pathfinder: quasi-Newton VI*, preprint
8: Gelman et al. (2013) *Bayesian Data Analysis*, textbook
9: M et al. (2022) *Nested $\widehat{R}$: assessing convergence*, preprint

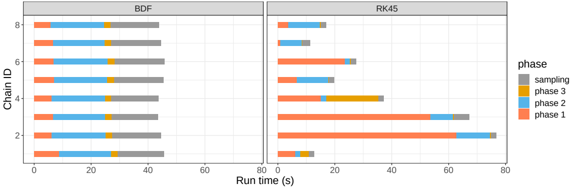
## Study on Michaelis-Menten Pharmacokinetic model[10]

- Which numerical integrator should we use in `Stan`?
  - $RK4^{th}/5^{th}$ (non-stiff solver)
  - BDF (stiff solver)

[10]: M et al. *Solving ODEs in a Bayesian context* poster at PAGE.

# Study on Michaelis-Menten Pharmacokinetic model[10]

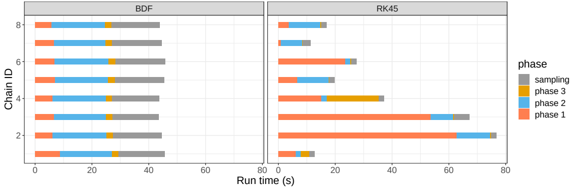
- Which numerical integrator should we use in `Stan`?
  - RK4$^{\text{th}}$/5$^{\text{th}}$ (non-stiff solver)
  - BDF (stiff solver)

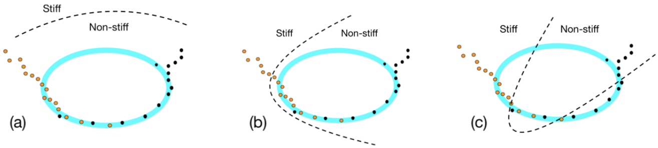



10: M et al. *Solving ODEs in a Bayesian context* poster at PAGE.

# Study on Michaelis-Menten Pharmacokinetic model[10]

- Which numerical integrator should we use in `Stan`?
  - RK4$^{\text{th}}$/5$^{\text{th}}$ (non-stiff solver)
  - BDF (stiff solver)



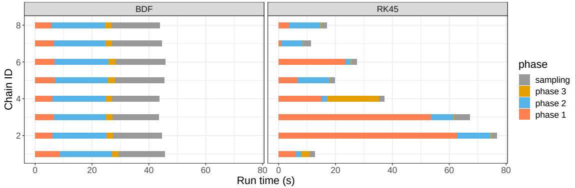



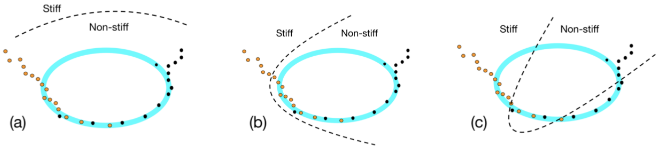

---

10: M et al. *Solving ODEs in a Bayesian context* poster at PAGE.

# Study on Michaelis-Menten Pharmacokinetic model[10]

- Which numerical integrator should we use in `Stan`?
  - RK$4^{\text{th}}/5^{\text{th}}$ (non-stiff solver)
  - BDF (stiff solver)




- Poses a challenge when running many chains in parallel (ongoing work with Stanislas Du Ché).

---

10: M et al. *Solving ODEs in a Bayesian context* poster at PAGE.

- Outline:
  - ODE-based Bayesian models
  - Solving ODEs across the parameter space
  - Propagating derivatives through ODEs

Need $\nabla_\theta \log p(\theta \mid y) = \nabla_\theta \left[\log p(y \mid \theta) + \log p(\theta)\right]$.

[11] M and Betancourt (2022) *Efficient automatic differentiation of implicit functions*, preprint.

Need $\nabla_\theta \log p(\theta \mid y) = \nabla_\theta \left[\log p(y \mid \theta) + \log p(\theta)\right]$.

$$\log p(y \mid \theta) = f(u(\theta), \theta)$$

[11] M and Betancourt (2022) *Efficient automatic differentiation of implicit functions*, preprint.

Need $\nabla_\theta \log p(\theta \mid y) = \nabla_\theta \left[\log p(y \mid \theta) + \log p(\theta)\right]$.

$$\log p(y \mid \theta) = f(u(\theta), \theta)$$

$$\frac{\mathrm{d}f}{\mathrm{d}\theta} = \frac{\partial f}{\partial \theta} + \sum_{t=t_0}^{\tau} \frac{\partial f}{\partial u(t)} \frac{\mathrm{d}u(t)}{\mathrm{d}\theta}$$

[11] M and Betancourt (2022) *Efficient automatic differentiation of implicit functions*, preprint.

Need $\nabla_\theta \log p(\theta \mid y) = \nabla_\theta \left[\log p(y \mid \theta) + \log p(\theta)\right]$.

$$\log p(y \mid \theta) = f(u(\theta), \theta)$$

$$\frac{\mathrm{d}f}{\mathrm{d}\theta} = \frac{\partial f}{\partial \theta} + \sum_{t=t_0}^{\tau} \frac{\partial f}{\partial u(t)} \frac{\mathrm{d}u(t)}{\mathrm{d}\theta}$$

- *Hidden in this equation is a Fréchet derivative between $f$ and $u$, i.e. between two infinite-dimensional objects. Such a derivative cannot be stored on a finite computer, so we need to make sure we don't compute it.*[11]

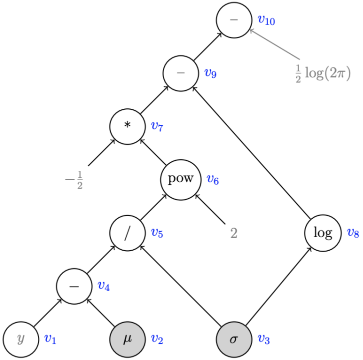
[11] M and Betancourt (2022) *Efficient automatic differentiation of implicit functions*, preprint.

# Automatic differentiation of implicit functions

(1) Direct method

(2) Forward method

(3) Adjoint method



## Automatic differentiation of implicit functions

(1) Direct method

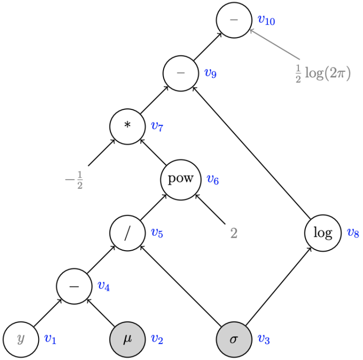
- Treat $u$ as a sequence of explicit functions, $u(t + \epsilon) \approx u(t) + \epsilon u'(t)$.

(2) Forward method

- Write an ODE solved by $\mathrm{d}u/\mathrm{d}\theta(t)$.

(3) Adjoint method

- Write an ODE solved by $\partial f / \partial u(t) \cdot \mathrm{d}u/\mathrm{d}\theta(t)$.



# Forward method

$$u'(t) = g(t, u, \vartheta) \in \mathbb{R}^N$$

- Let $K$ be the dimension of input which depend on $\theta$
  - $\vartheta$: explicit ODE parameters
  - $u_0$: initial condition if parameter dependent $\rightarrow$ Can rewrite the ODE as a deviation from a baseline and take $u_0 = \mathbf{0}$.
  - time steps: $t_0, t_1, \cdots$

3: Grinsztajn et al. (2021) *Bayesian workflow for disease transmission*, Statistics in Medicine

# Forward method

$$u'(t) = g(t, u, \vartheta) \in \mathbb{R}^N$$

- Let $K$ be the dimension of input which depend on $\theta$
  - $\vartheta$: explicit ODE parameters
  - $u_0$: initial condition if parameter dependent $\rightarrow$ Can rewrite the ODE as a deviation from a baseline and take $u_0 = \mathbf{0}$.
  - time steps: $t_0, t_1, \cdots$

$$\frac{\mathrm{d}u}{\mathrm{d}\vartheta}(t) = \int_{t_0}^{\tau} \mathrm{d}t \frac{\mathrm{d}u'}{\mathrm{d}\vartheta}(t)$$

- Solve a system of $N + NK$ coupled ODEs.

3: Grinsztajn et al. (2021) *Bayesian workflow for disease transmission*, Statistics in Medicine

## Forward method

$$u'(t) = g(t, u, \vartheta) \in \mathbb{R}^N$$

- Let $K$ be the dimension of input which depend on $\theta$
  - $\vartheta$: explicit ODE parameters
  - $u_0$: initial condition if parameter dependent $\rightarrow$ Can rewrite the ODE as a deviation from a baseline and take $u_0 = \mathbf{0}$.
  - time steps: $t_0, t_1, \cdots$

$$\frac{\mathrm{d}u}{\mathrm{d}\vartheta}(t) = \int_{t_0}^{\tau} \mathrm{d}t \frac{\mathrm{d}u'}{\mathrm{d}\vartheta}(t)$$

- Solve a system of $N + NK$ coupled ODEs.
- Our goal should be to minimize $K$.
  - In Covid-19 model, went from $K = 62$ to $K = 5$, meaning solving 3596 to 290 ODEs[3] $\rightarrow$ 3 days to 2 hours to fit model.

3: Grinsztajn et al. (2021) *Bayesian workflow for disease transmission*, Statistics in Medicine

## Adjoint method

- Construct an ODE which *directly* returns

$$\delta^\dagger \cdot \frac{\mathrm{d}u}{\mathrm{d}\theta}(t).$$

- Requires solving the ODE forward and backward in time.
- Solve a total of $2N + K$ coupled ODEs (rather than $N + NK$).

- Eligibly method is old;[12] popularized in Machine Learning by Neural ODEs.[13]
- Better scaling but not always better performance than forward method.[14]

---

12: Pontryagin et al (1963) *The Mathematical Theory of Point Processes*, textbook
13: Chen et al (2018) *Neural Ordinary Differential Equations*, NeurIPS
14: Rackauckas et al (2021) *A comparison of Autodiff [...] for derivatives of differential equations* IEEE HPEC

## Arsenal of tools

- Prob languages which support ODEs:
  - `Stan`:
    - supports three ODE integrators (RK45, BDF, Adams).
    - supports forward and adjoint differentiation methods.
    - supports matrix exponential for linear ODEs.
    - supports several implicit functions.
  - `Torsten`:
    - extends Stan for pharmacometrics modeling.
    - solve ODEs within a clinical event schedule.
  - `TensorFlow Probability`
    - tip: use it with JAX, rather than TensorFlow.
    - ODE support exists but is limited.
    - support for finite-dim implicit functions seems quite good.
  - `Turing`
    - uses Julia
    - I have not tried it, but I suspect it's good.

## Arsenal of tools

- Prob languages which support ODEs:
  - Stan:
    - supports three ODE integrators (RK45, BDF, Adams).
    - supports forward and adjoint differentiation methods.
    - supports matrix exponential for linear ODEs.
    - supports several implicit functions.
  - Torsten:
    - extends Stan for pharmacometrics modeling.
    - solve ODEs within a clinical event schedule.
  - TensorFlow Probability
    - tip: use it with JAX, rather than TensorFlow.
    - ODE support exists but is limited.
    - support for finite-dim implicit functions seems quite good.
  - Turing
    - uses Julia
    - I have not tried it, but I suspect it's good.
- Questions? Comments?

  cmargossian@flatironinstitute.org

  CCM third floor!