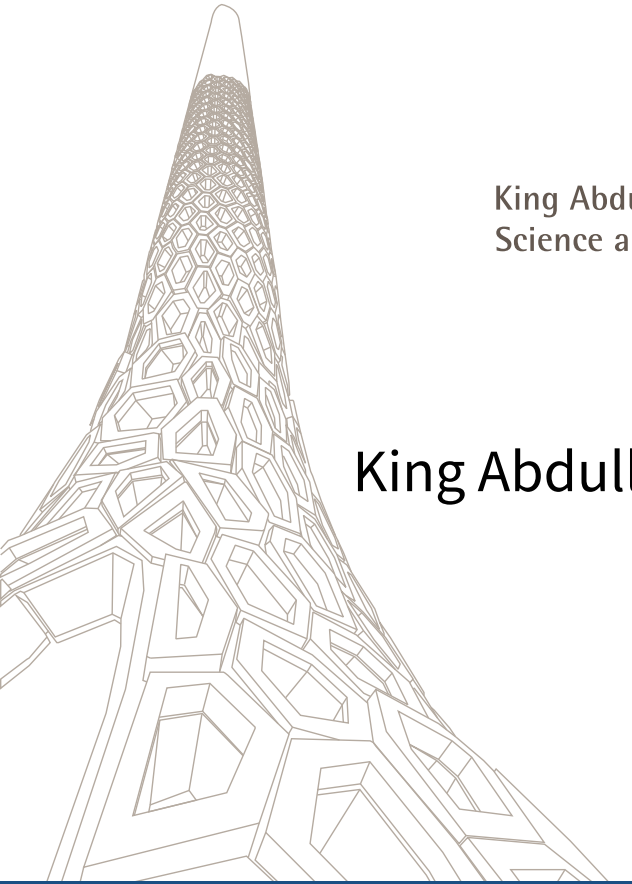
The Role of GMRFs in Statistical Modeling and Scalable Bayesian Inference (part II)



King Abdullah University of Science and Technology

جامعة الملك عبدالله للعلوم والتقنية

Håvard Rue
King Abdullah University of Science and Technology
Saudi Arabia

May 2026

Plan for the talk

- Continous time
- Continous space
- Where do GMRFs come now?
- Generalisation: how simple ideas can solve hard problems
- ...and the solution is still a GMRF
- (Approximate) Bayesian inference
- Variational Bayes
- Numerics for sparse (precision-)matrices
- New sparse solver: *sTiles*

Plan for the talk

- Continous time
- Continous space
- Where do GMRFs come now?
- Generalisation: how simple ideas can solve hard problems
- ...and the solution is still a GMRF
- (Approximate) Bayesian inference
- Variational Bayes
- Numerics for sparse (precision-)matrices
- New sparse solver: *sTiles*

Plan for the talk

- Continous time
- Continous space
- Where do GMRFs come now?
- Generalisation: how simple ideas can solve hard problems
- ...and the solution is still a GMRF
- (Approximate) Bayesian inference
- Variational Bayes
- Numerics for sparse (precision-)matrices
- New sparse solver: *sTiles*

Plan for the talk

- Continous time
- Continous space
- Where do GMRFs come now?
- Generalisation: how simple ideas can solve hard problems
- ...and the solution is still a GMRF
- (Approximate) Bayesian inference
- Variational Bayes
- Numerics for sparse (precision-)matrices
- New sparse solver: *sTiles*

Plan for the talk

- Continous time
- Continous space
- Where do GMRFs come now?
- Generalisation: how simple ideas can solve hard problems
- ...and the solution is still a GMRF
- (Approximate) Bayesian inference
- Variational Bayes
- Numerics for sparse (precision-)matrices
- New sparse solver: *sTiles*

Plan for the talk

- Continous time
- Continous space
- Where do GMRFs come now?
- Generalisation: how simple ideas can solve hard problems
- ...and the solution is still a GMRF
- (Approximate) Bayesian inference
- Variational Bayes
- Numerics for sparse (precision-)matrices
- New sparse solver: *sTiles*

Plan for the talk

- Continous time
- Continous space
- Where do GMRFs come now?
- Generalisation: how simple ideas can solve hard problems
- ...and the solution is still a GMRF
- (Approximate) Bayesian inference
- Variational Bayes
- Numerics for sparse (precision-)matrices
- New sparse solver: *sTiles*

Plan for the talk

- Continous time
- Continous space
- Where do GMRFs come now?
- Generalisation: how simple ideas can solve hard problems
- ...and the solution is still a GMRF
- (Approximate) Bayesian inference
- Variational Bayes
- Numerics for sparse (precision-)matrices
- New sparse solver: *sTiles*

Plan for the talk

- Continous time
- Continous space
- Where do GMRFs come now?
- Generalisation: how simple ideas can solve hard problems
- ...and the solution is still a GMRF
- (Approximate) Bayesian inference
- Variational Bayes
- Numerics for sparse (precision-)matrices
- New sparse solver: *sTiles*

Example: Markov processes in time

- Auto-regressive process of order 1

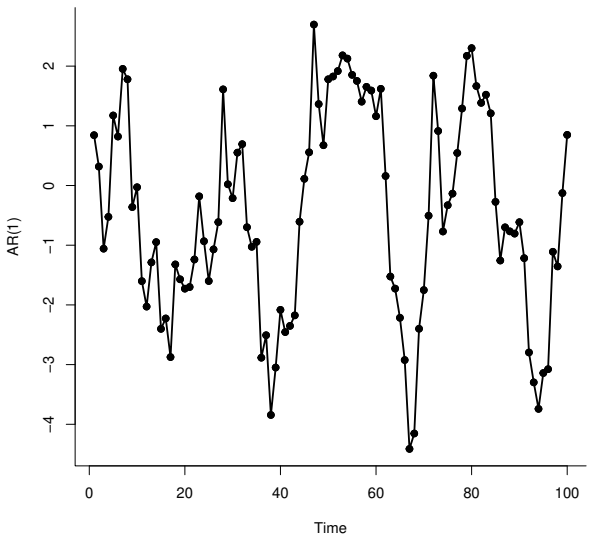
$$x_t = \phi x_{t-1} + \epsilon_t, \quad 0 \le \phi < 1, \quad t = 1, \ldots, n$$

with Gaussian noise ϵ_t

- $\boldsymbol{Q}$ is tridiagonal
- $\boldsymbol{C}$ is $\phi^{-|i-j|}$ (exp-correlation function)
- Cont.time: Ornstein-Uhlenbeck process

$$dx_t = -\theta\, x_t\, dt + \sigma dW_t,$$

Wiener process W_t



Example: Markov processes in time

- Auto-regressive process of order 1

$$x_t = \phi x_{t-1} + \epsilon_t, \quad 0 \le \phi < 1, \quad t = 1, \ldots, n$$

  with Gaussian noise ϵ_t

- $\boldsymbol{Q}$ is tridiagonal

$$\begin{pmatrix} 1 & -\phi & & & & & \\ -\phi & 1+\phi^2 & -\phi & & & & \\ & \ddots & \ddots & \ddots & & & \\ & & & -\phi & 1+\phi^2 & -\phi & \\ & & & & -\phi & 1 \end{pmatrix}$$

- $\boldsymbol{C}$ is $\phi^{-|i-j|}$ (exp-correlation function)
- Cont.time: Ornstein-Uhlenbeck process

$$dx_t = -\theta\, x_t\, dt + \sigma dW_t,$$

  Wiener process W_t

Example: Markov processes in time

- Auto-regressive process of order 1

$$x_t = \phi x_{t-1} + \epsilon_t, \quad 0 \le \phi < 1, \quad t = 1, \ldots, n$$

  with Gaussian noise ϵ_t
- $\boldsymbol{Q}$ is tridiagonal
- $\boldsymbol{C}$ is $\phi^{-|i-j|}$ (exp-correlation function)
- Cont.time: Ornstein-Uhlenbeck process

$$dx_t = -\theta\, x_t\, dt + \sigma dW_t,$$

  Wiener process W_t

$$\begin{pmatrix} 1 & \phi & \phi^2 & \phi^3 & \phi^4 & \phi^5 & \phi^6 \\ \phi & 1 & \phi & \phi^2 & \phi^3 & \phi^4 & \phi^5 \\ \phi^2 & \phi & 1 & \phi & \phi^2 & \phi^3 & \phi^4 \\ \phi^3 & \phi^2 & \phi & 1 & \phi & \phi^2 & \phi^3 \\ \phi^4 & \phi^3 & \phi^2 & \phi & 1 & \phi & \phi^2 \\ \phi^5 & \phi^4 & \phi^3 & \phi^2 & \phi & 1 & \phi \\ \phi^6 & \phi^5 & \phi^4 & \phi^3 & \phi^2 & \phi & 1 \end{pmatrix}$$

Example: Markov processes in time

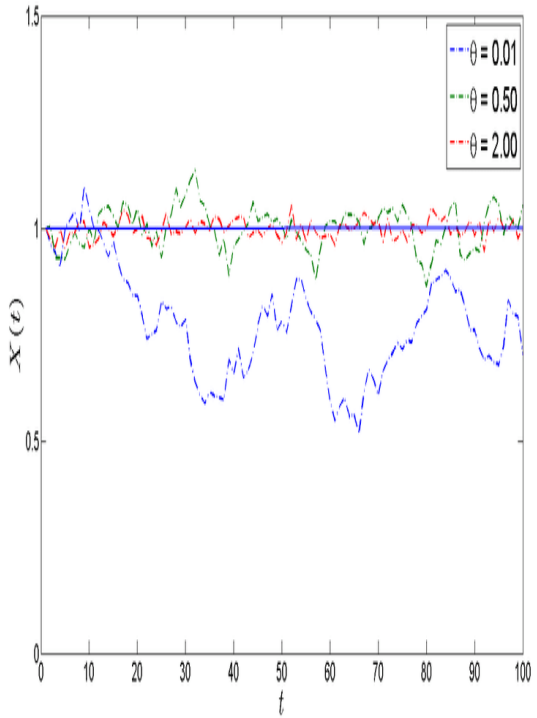
- Auto-regressive process of order 1

$$x_t = \phi x_{t-1} + \epsilon_t, \quad 0 \le \phi < 1, \quad t = 1, \ldots, n$$

  with Gaussian noise ϵ_t
- $\boldsymbol{Q}$ is tridiagonal
- $\boldsymbol{C}$ is $\phi^{-|i-j|}$ (exp-correlation function)
- Cont.time: Ornstein-Uhlenbeck process

$$dx_t = -\theta\, x_t\, dt + \sigma dW_t,$$

  Wiener process W_t



Conditional distribution

$$\mathrm{E}(x_t|\{x_s, s \le 0\}) = x_0 \exp(-\theta t)$$

$$\mathrm{Var}(x_t|\{x_s, s \le 0\}) = \frac{\sigma^2}{2\theta}(1 - \exp(-2\theta t))$$

With timesteps equal to 1, then the OU-process equal to AR1 with

$$\phi = \exp(-\theta)$$

and

$$\mathrm{Var}(\epsilon_t) = \frac{\sigma^2}{2\theta}(1 - \exp(-2\theta))$$

Higher order

Informally, we have

$$x_t' = -\theta x_t + \epsilon_t$$

It should be no surprise, that we can do higher order with

$$\cdots + a_2 x_t'' + a_1 x_t' + a_0 x_t = \epsilon_t$$

or

$$\sum_{k=0}^{p} a_p D^{(p)} x_t = DW_t$$

Conditional distribution

$$\begin{pmatrix} x \\ x' \\ x'' \\ \vdots \end{pmatrix}_t \quad \text{depends on} \quad \begin{pmatrix} x \\ x' \\ x'' \\ \vdots \end{pmatrix}_0 \quad + \quad \text{noise}$$

Vector representation

where the state vector $\mathbf{X}(t) = (X_0(t), \ldots, X_{p-1}(t))^T$ satisfies the Itô equation,

$$d\mathbf{X}(t) = A\mathbf{X}(t)dt + \mathbf{e}dW(t), \tag{2.3}$$

with

$$A = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ -a_p & -a_{p-1} & -a_{p-2} & \cdots & -a_1 \end{bmatrix} \quad \text{and} \quad \mathbf{e} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}.$$

So we're back at the state-space formulation

State-space formulation

$$\boldsymbol{x}_{t+1} = \boldsymbol{A}\boldsymbol{x}_t + \boldsymbol{\epsilon}_t$$

- Kalman recursions
- Var(ϵ_t) is singular, so implementation is annoying and wasteful
- Markov, as $\boldsymbol{x}_{t+1}$ only depends on $\boldsymbol{x}_t$
- In continous time, this include also its derivatives
- Again, Markov in a extended space

Matérn fields and the reference SPDE

The solution of this SPDE (Whittle, 1954/1963)

$$(\kappa^2 - \Delta)^{\alpha/2} \boldsymbol{x}(\boldsymbol{s}) = \boldsymbol{\epsilon}(\boldsymbol{s}), \qquad \alpha = \nu + \text{dim}/2$$

is a Gaussian field with Matérn Covariance function.

The SPDE is harder to work with, but knowing how, then earlier hard problems becomes easy!

- manifolds
- non-stationary
- computational properties
- +++

Matérn fields and the reference SPDE

The solution of this SPDE (Whittle, 1954/1963)

$$(\kappa^2 - \Delta)^{\alpha/2} \boldsymbol{x}(\boldsymbol{s}) = \boldsymbol{\epsilon}(\boldsymbol{s}), \qquad \alpha = \nu + \text{dim}/2$$

is a Gaussian field with Matérn Covariance function.

The SPDE is harder to work with, but knowing how, then earlier hard problems becomes easy!

- manifolds
- non-stationary
- computational properties
- +++

Matérn fields and the reference SPDE

The solution of this SPDE (Whittle, 1954/1963)

$$(\kappa^2 - \Delta)^{\alpha/2} \boldsymbol{x}(\boldsymbol{s}) = \boldsymbol{\epsilon}(\boldsymbol{s}), \qquad \alpha = \nu + \text{dim}/2$$

is a Gaussian field with Matérn Covariance function.

The SPDE is harder to work with, but knowing how, then earlier hard problems becomes easy!

- manifolds
- non-stationary
- computational properties
- +++

Matérn fields and the reference SPDE

The solution of this SPDE (Whittle, 1954/1963)

$$(\kappa^2 - \Delta)^{\alpha/2} \boldsymbol{x}(\boldsymbol{s}) = \boldsymbol{\epsilon}(\boldsymbol{s}), \qquad \alpha = \nu + \text{dim}/2$$

is a Gaussian field with Matérn Covariance function.

The SPDE is harder to work with, but knowing how, then earlier hard problems becomes easy!

- manifolds
- non-stationary
- computational properties
- +++

Matérn fields and the reference SPDE

The solution of this SPDE (Whittle, 1954/1963)

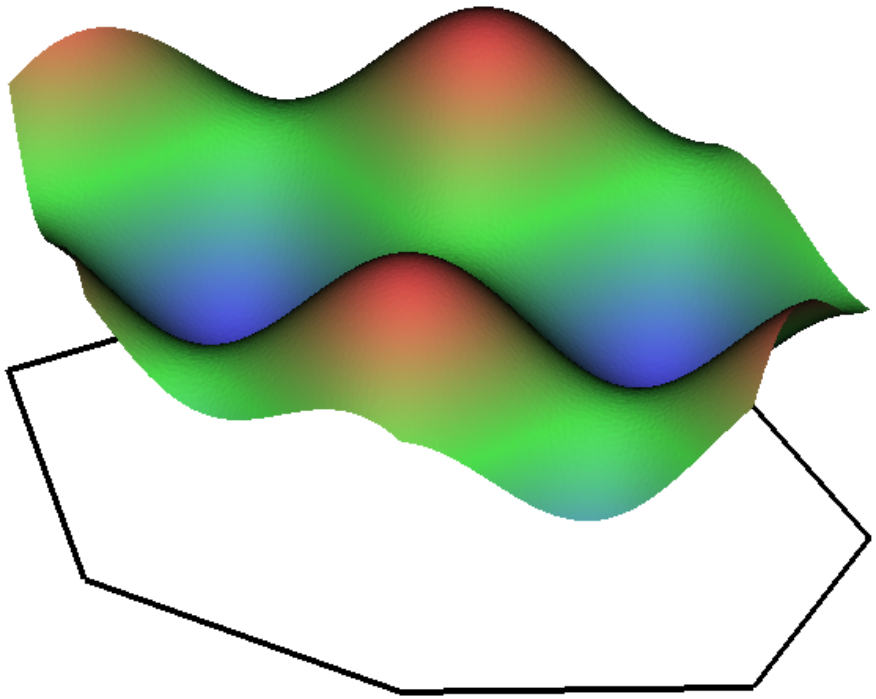
$$(\kappa^2 - \Delta)^{\alpha/2} \boldsymbol{x}(\boldsymbol{s}) = \boldsymbol{\epsilon}(\boldsymbol{s}), \qquad \alpha = \nu + \text{dim}/2$$

is a Gaussian field with Matérn Covariance function.

The SPDE is harder to work with, but knowing how, then earlier hard problems becomes easy!

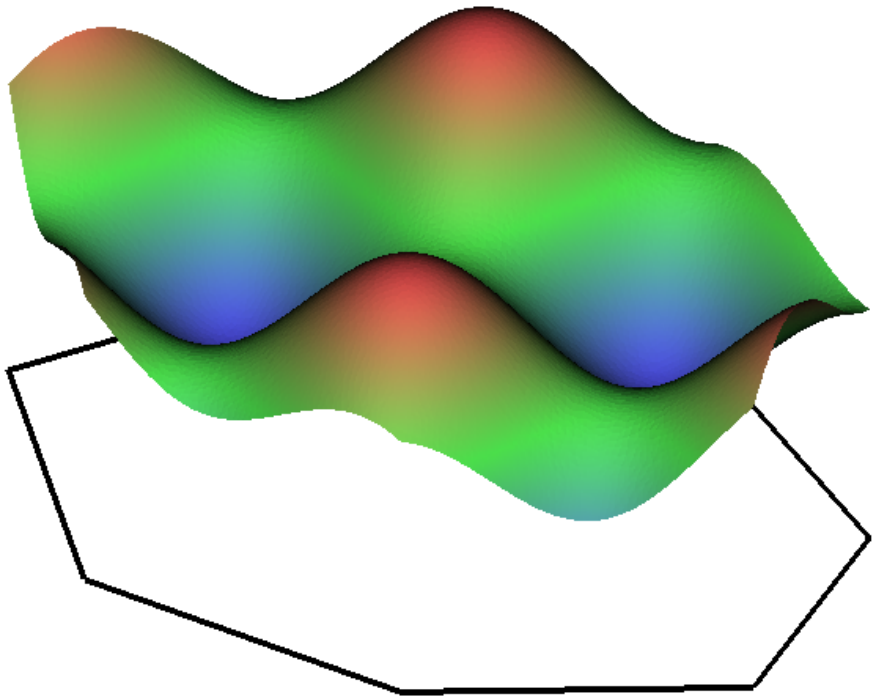
- manifolds
- non-stationary
- computational properties
- +++

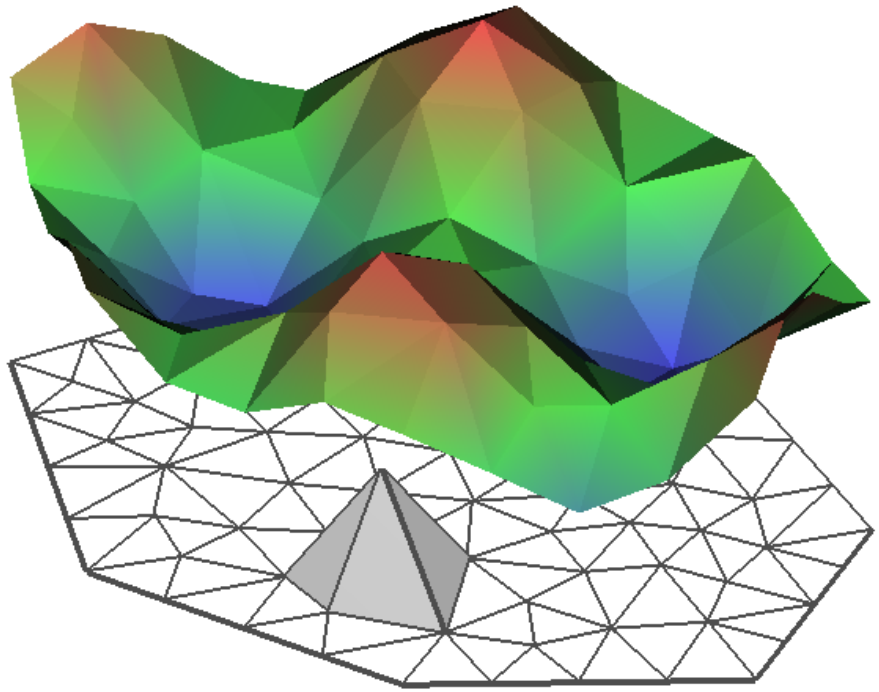
Piece-wise linear representations



Piece-wise linear representations

$$x(\boldsymbol{u}) = \sum_k \psi_k(\boldsymbol{u})\, x_k$$





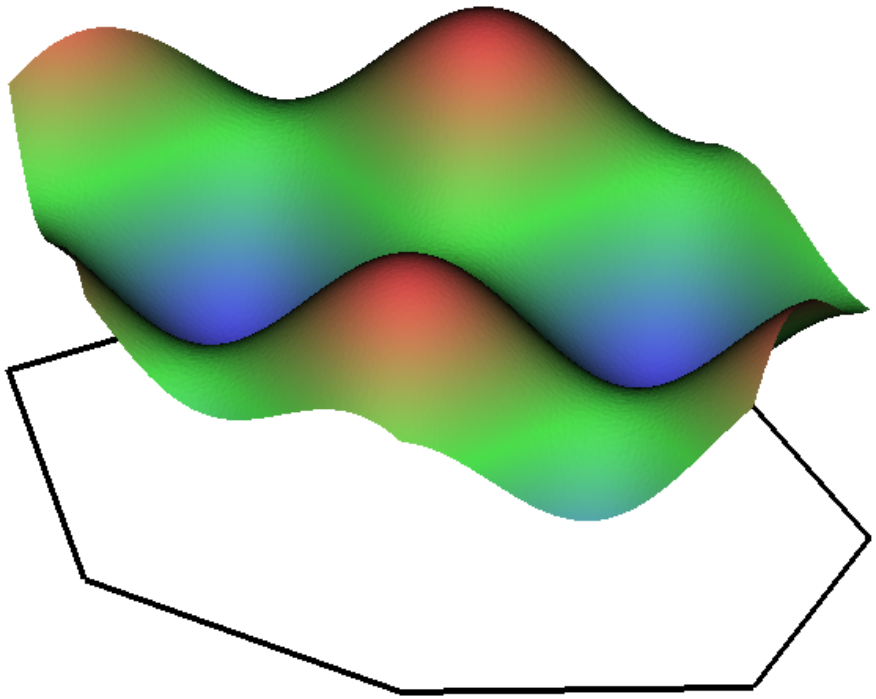
How to “work” with the SPDE?

Hilbert space representation/finite element method

$$\boldsymbol{x}(\boldsymbol{u}) = \sum_{k=1}^{N} \psi_k(\boldsymbol{u}) w_k$$

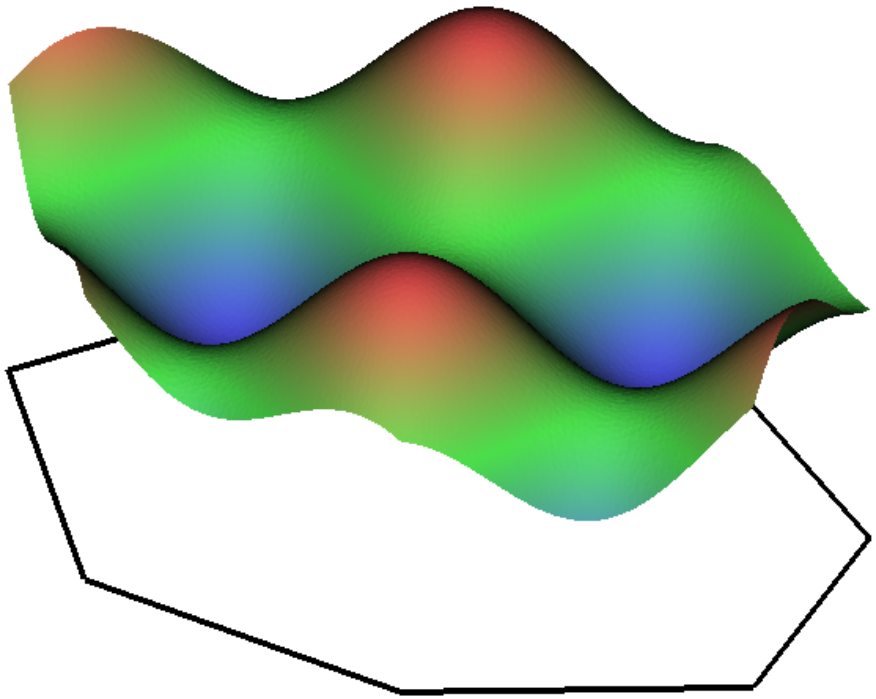
for basis-functions $\{\psi_k\}$ and (Gaussian) weights $\{w_k\}$

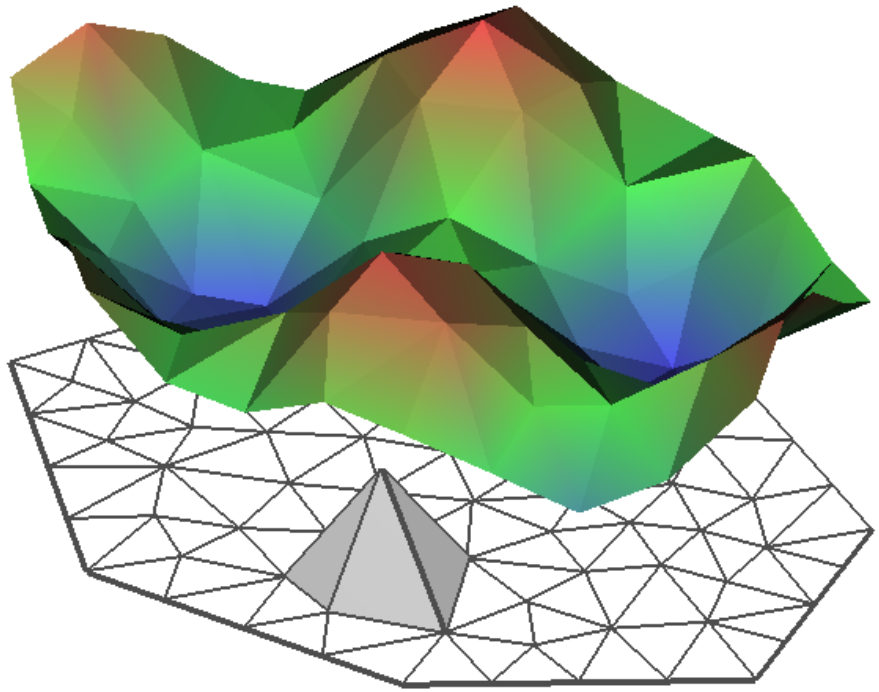
Piecewise linear representations



Piecewise linear representations

$$x(\boldsymbol{u}) = \sum_k \psi_k(\boldsymbol{u})\, x_k$$





(Stochastic) Weak solution

$$(\kappa^2 - \Delta)^{\alpha/2} \boldsymbol{x}(\boldsymbol{s}) = \boldsymbol{\epsilon}(\boldsymbol{s})$$

(Stochastic) Weak solution

$$\left\{ \langle \phi_k, (\kappa^2 - \Delta)^{\alpha/2} \boldsymbol{x} \rangle \right\}_k \stackrel{D}{=} \{ \langle \phi_k, \boldsymbol{\epsilon} \rangle \}_k$$

for all *test functions* $\{\phi_k\}_k$.

Results for $\alpha = 1, 2, \ldots$ (I)

$$\alpha = 1: \qquad \phi_k = (\kappa^2 - \Delta)^{1/2} \psi_k$$
$$\alpha = 2: \qquad \phi_k = \psi_k$$

Define matrices $\boldsymbol{C}$, $\boldsymbol{G}$ and $\boldsymbol{K}$

$$C_{ij} = \langle \phi_i, \phi_j \rangle \quad i \neq j$$
$$G_{ij} = \langle \nabla \phi_i, \nabla \phi_j \rangle$$
$$\boldsymbol{K} = \kappa^2 \boldsymbol{C} + \boldsymbol{G}$$

Results for $\alpha = 1, 2, \ldots$ (II)

The weights are Gaussian with precision matrix $\boldsymbol{Q}_{\alpha,\kappa}$

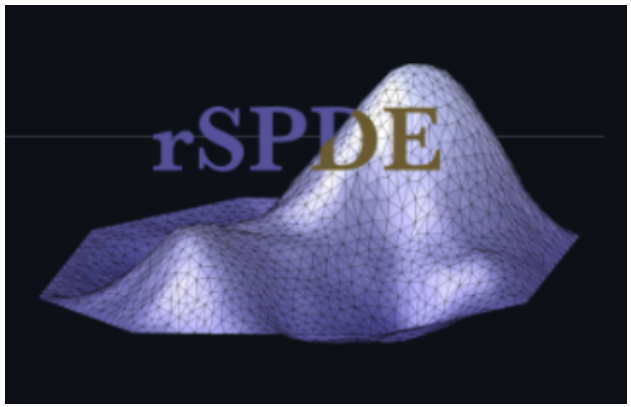
$$
\begin{aligned}
\boldsymbol{Q}_{1,\kappa} &= \kappa^2 \boldsymbol{C} + \boldsymbol{G} \\
\boldsymbol{Q}_{2,\kappa} &= \boldsymbol{K}\boldsymbol{C}^{-1}\boldsymbol{K} \\
&\vdots \\
\boldsymbol{Q}_{\alpha,\kappa} &= \boldsymbol{K}\boldsymbol{C}^{-1}\boldsymbol{Q}_{\alpha-2,\kappa}\boldsymbol{C}^{-1}\boldsymbol{K}
\end{aligned}
$$

Replace $\boldsymbol{C}$ with a appropriate diagonal matrix ("mass lumping"): This is OK

Non-integer values of α

Non-integer values of α

- Not Markov
- Similar to 'long memory' from Part I
- Can be approach in the same way: as a weighted sum of integer-α fields

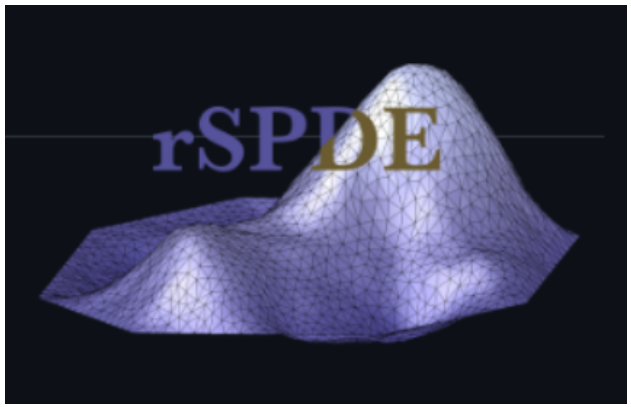


`github.com/davidbolin/rSPDE`

Non-integer values of α

Non-integer values of α

- Not Markov
- Similar to 'long memory' from Part I
- Can be approach in the same way: as a weighted sum of integer-α fields

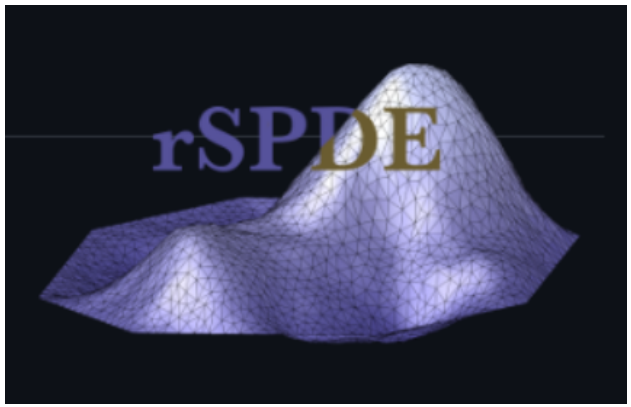


`github.com/davidbolin/rSPDE`

Non-integer values of α

Non-integer values of α

- Not Markov
- Similar to 'long memory' from Part I
- Can be approach in the same way: as a weighted sum of integer-α fields



`github.com/davidbolin/rSPDE`

Markov properties in dim≥ 2

- Similar to time-case
- Iff the Spectrum is of the form

$$S(\boldsymbol{w}) \propto \frac{1}{\text{even.poly}(\boldsymbol{w})}$$

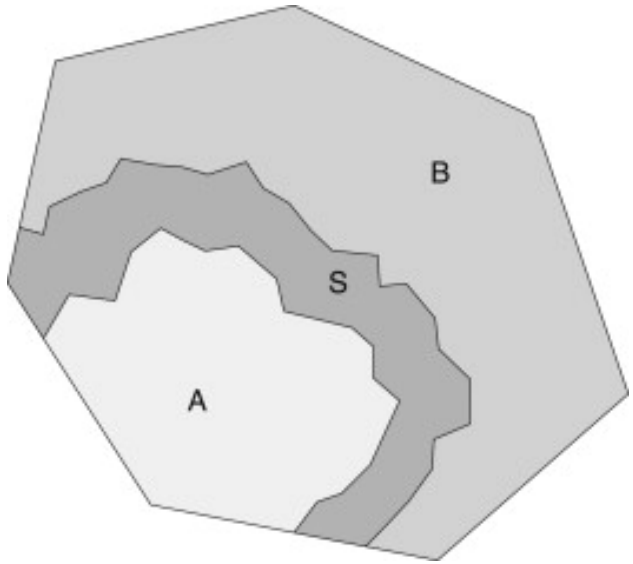
- It is intutitive

$$\sum \text{part.deriv}(\boldsymbol{x}(\boldsymbol{s}))) = \text{noise}$$

then numerical solution is $\boldsymbol{Ax} = \boldsymbol{w}$ for some **local** $\boldsymbol{A}$, and

$$\text{Prec}(\boldsymbol{x}) \propto \boldsymbol{AA}^T$$

which is local as $\boldsymbol{A}$ is local.



Markov properties in dim≥ 2

- Similar to time-case
- Iff the Spectrum is of the form

$$S(\boldsymbol{w}) \propto \frac{1}{\text{even.poly}(\boldsymbol{w})}$$

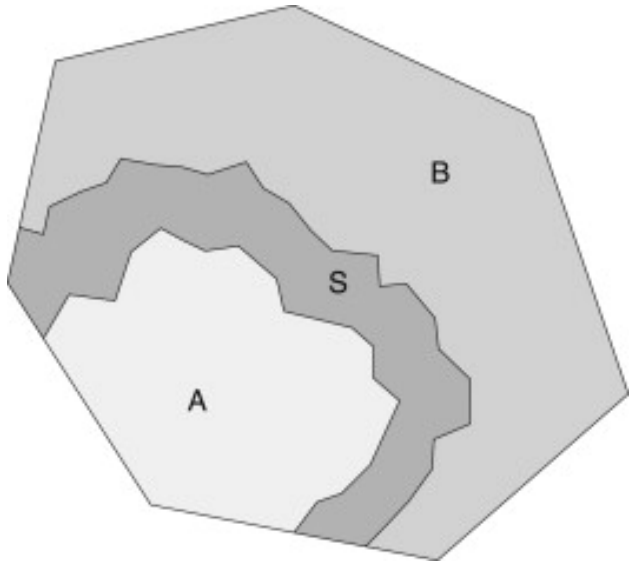
- It is intutitive

$$\sum \text{part.deriv}(\boldsymbol{x}(\boldsymbol{s}))) = \text{noise}$$

then numerical solution is $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{w}$ for some **local** $\boldsymbol{A}$, and

$$\text{Prec}(\boldsymbol{x}) \propto \boldsymbol{A}\boldsymbol{A}^T$$

which is local as $\boldsymbol{A}$ is local.



Markov properties in dim$\geq$ 2

- Similar to time-case
- Iff the Spectrum is of the form

$$S(\boldsymbol{w}) \propto \frac{1}{\text{even.poly}(\boldsymbol{w})}$$

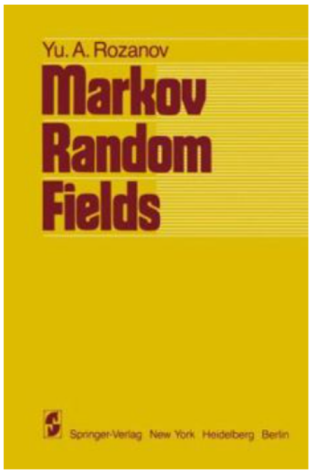
- It is intutitive

$$\sum \text{part.deriv}(\boldsymbol{x}(\boldsymbol{s}))) = \text{noise}$$

then numerical solution is $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{w}$ for some **local** $\boldsymbol{A}$, and

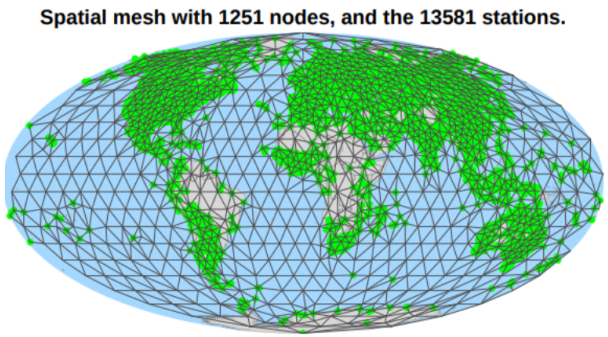
$$\text{Prec}(\boldsymbol{x}) \propto \boldsymbol{A}\boldsymbol{A}^T$$

which is local as $\boldsymbol{A}$ is local.



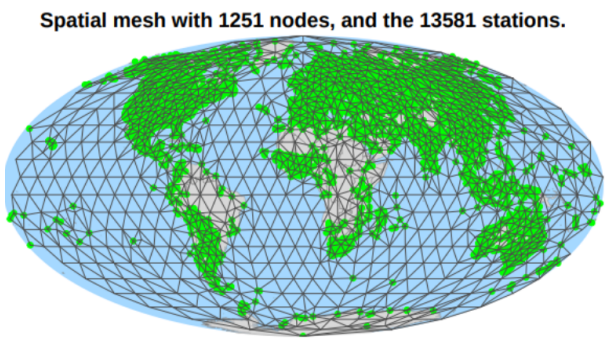
Extentions

- space-time
- non-stationary
- sphere
- and mix thereof



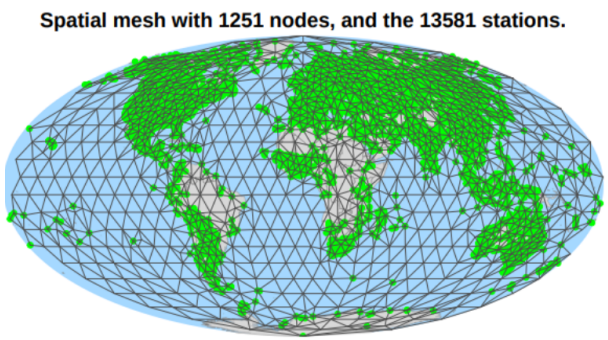
Extentions

- space-time
- non-stationary
- sphere
- and mix thereof



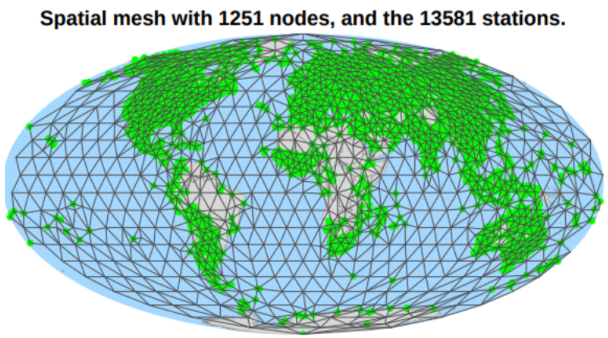
Extentions

- space-time
- non-stationary
- sphere
- and mix thereof

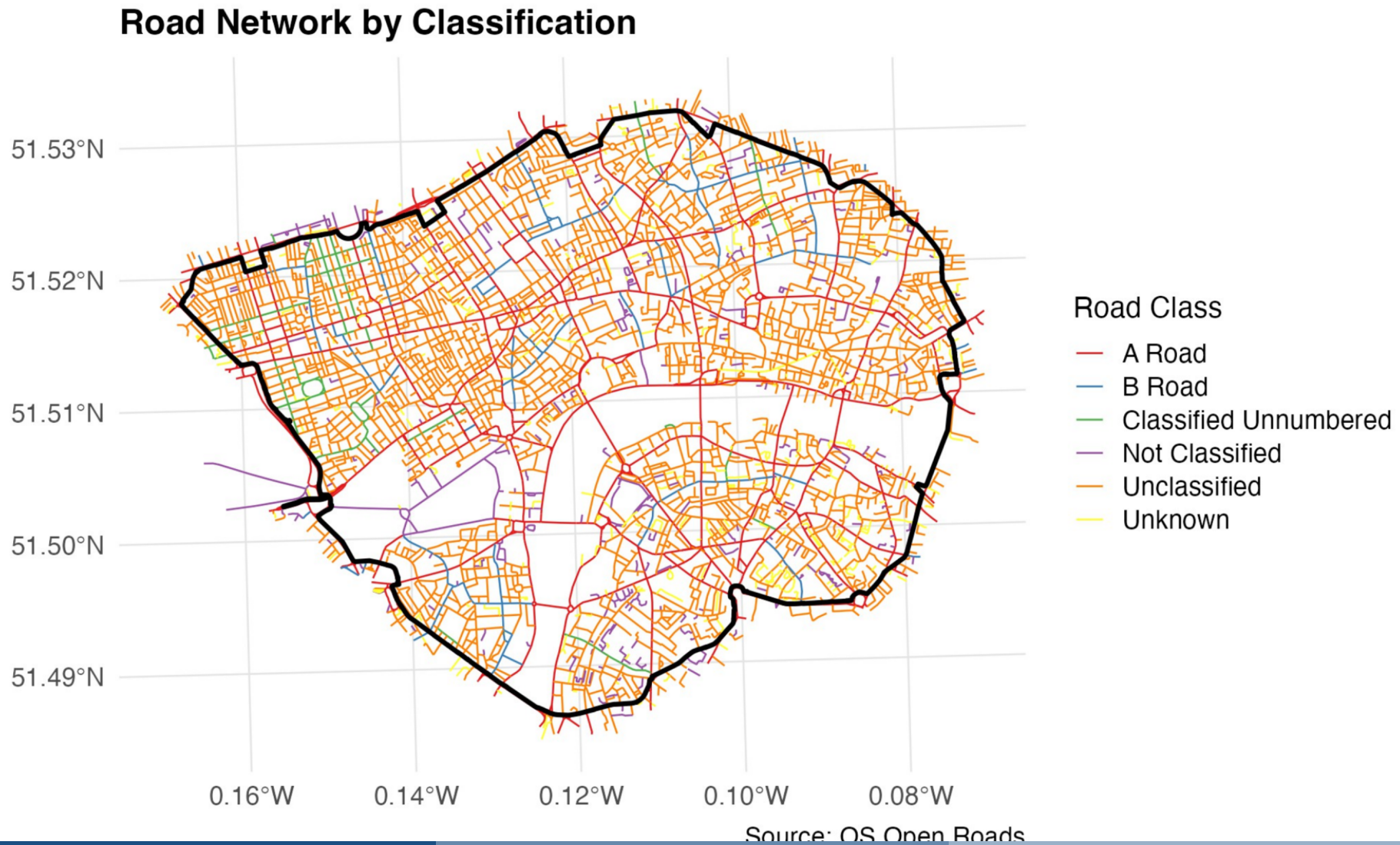


Extentions

- space-time
- non-stationary
- sphere
- and mix thereof



Extentions


Road Network by Classification
51.53°N
51.52°N
51.51°N
51.50°N
51.49°N
0.16°W
0.14°W
0.12°W
0.10°W
0.08°W
Source: OS Open Roads
Road Class
A Road
B Road
Classified Unnumbered
Not Classified
Unclassified
Unknown

Implicite, not explicite!

Extentions

MetricGraph 1.5.1.9000 Get started Functions Vignettes Discussions Changelog

MetricGraph

CRAN 1.6.0 – 2026-05-06 R-CMD-check failing R-CMD-check-windows failing

`MetricGraph` is an R package used for working with data and random fields on metric graphs, such as street or river networks. The main functionality is contained in the `metric_graph` class, which is used for specifying metric graphs, adding data to them, visualization, and other basic functions that are needed for working with data and random fields on metric graphs. The package also implements various Gaussian fields on metric graphs, and in particular the Whittle–Matérn fields introduced in the references below.

Research group of Prof David Bolin, KAUST

Transparent barrier

Transparent barrier

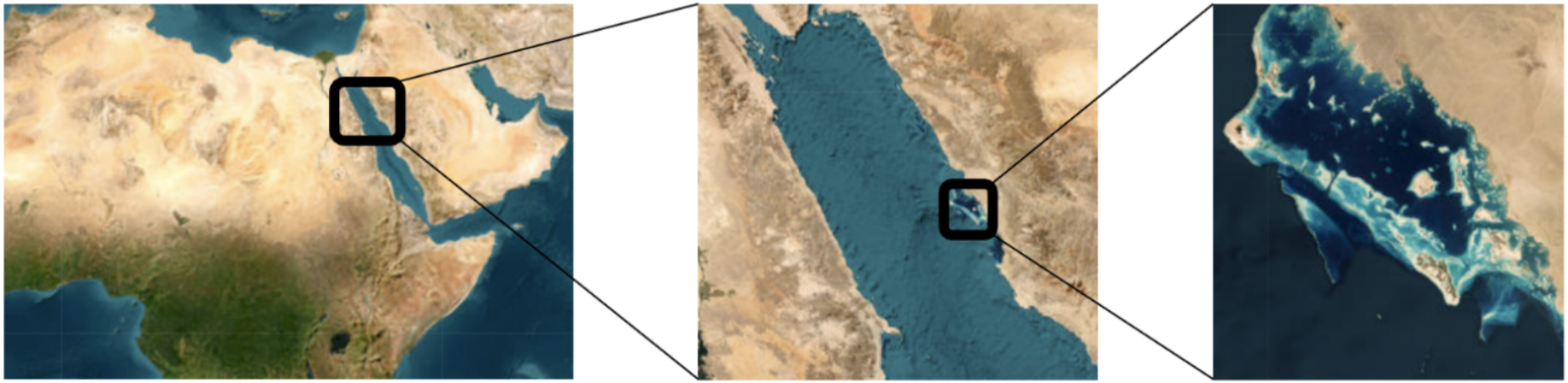
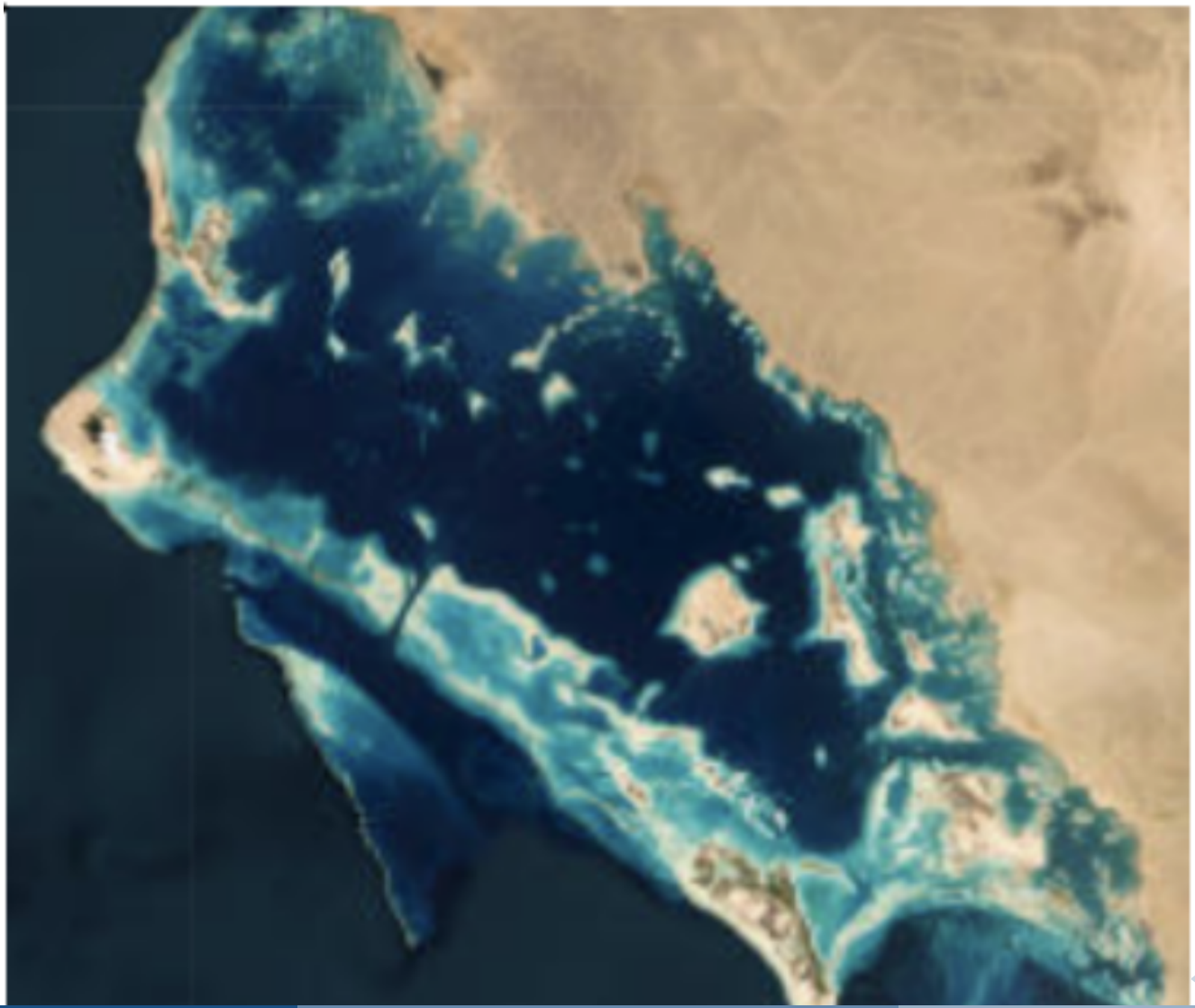
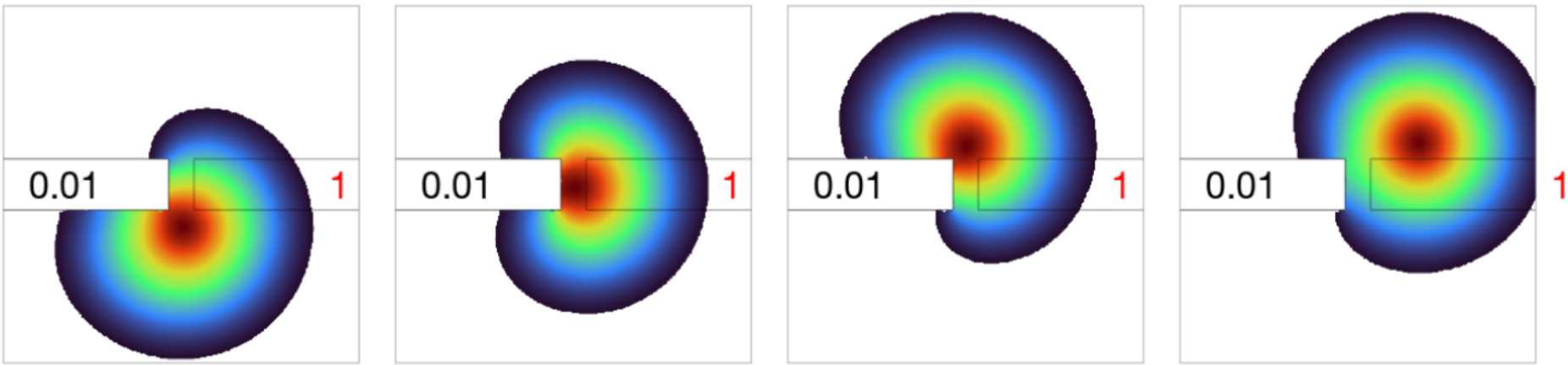


Figure 1: Study area in the Red Sea for Dugong conservation

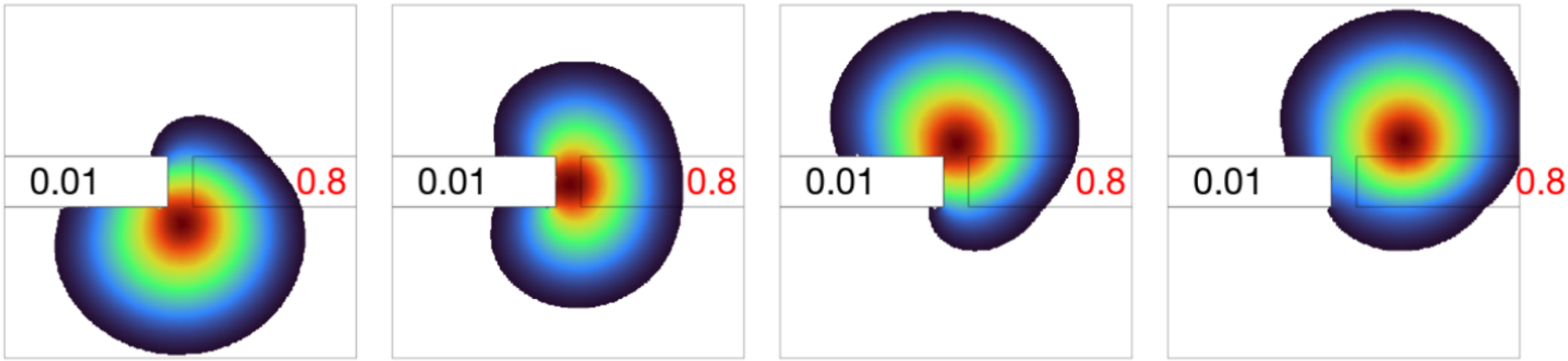
Transparent barrier



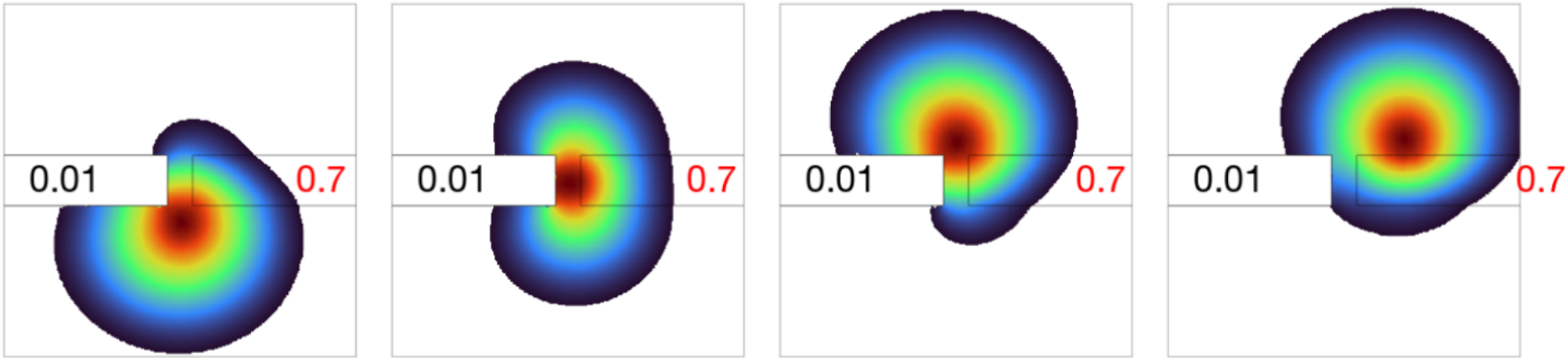
Transparent barrier



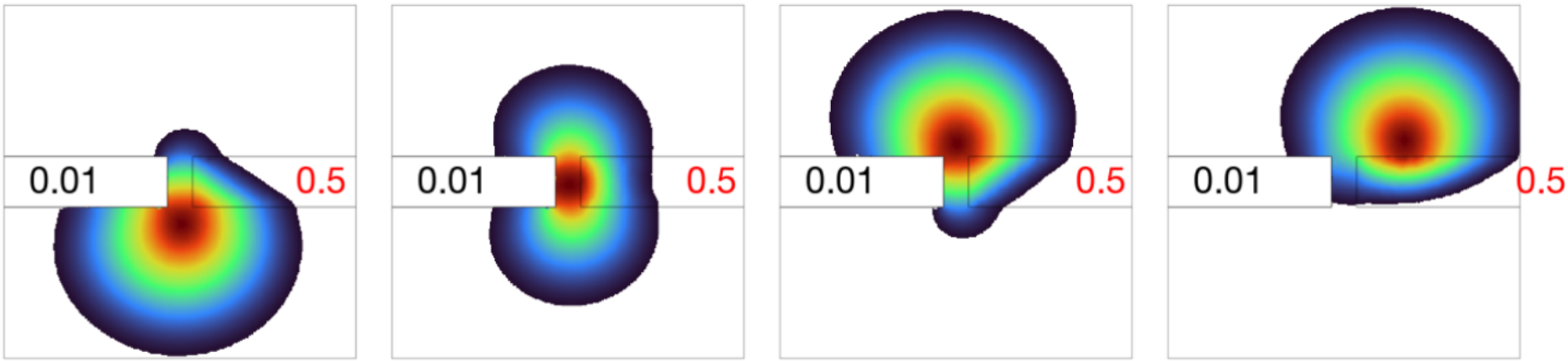
Transparent barrier



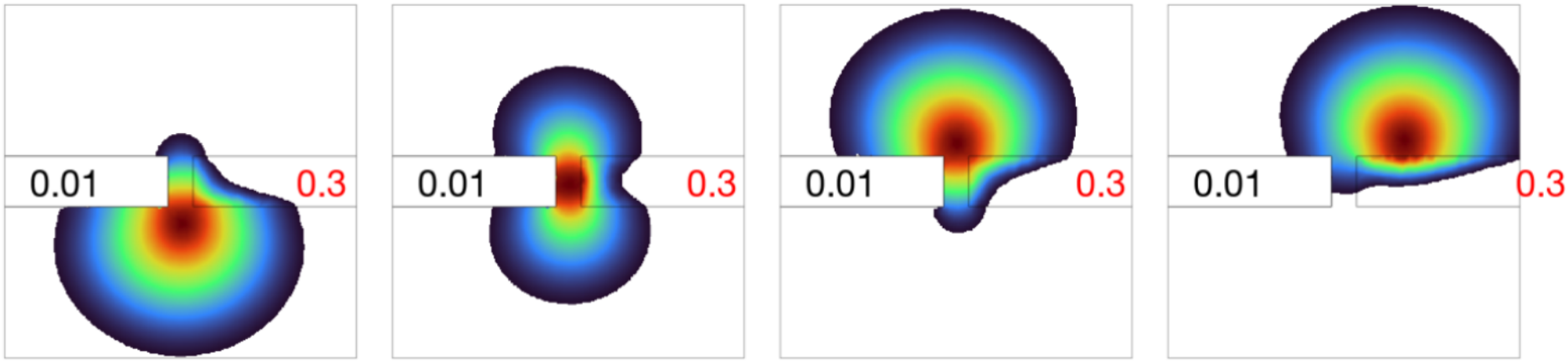
Transparent barrier



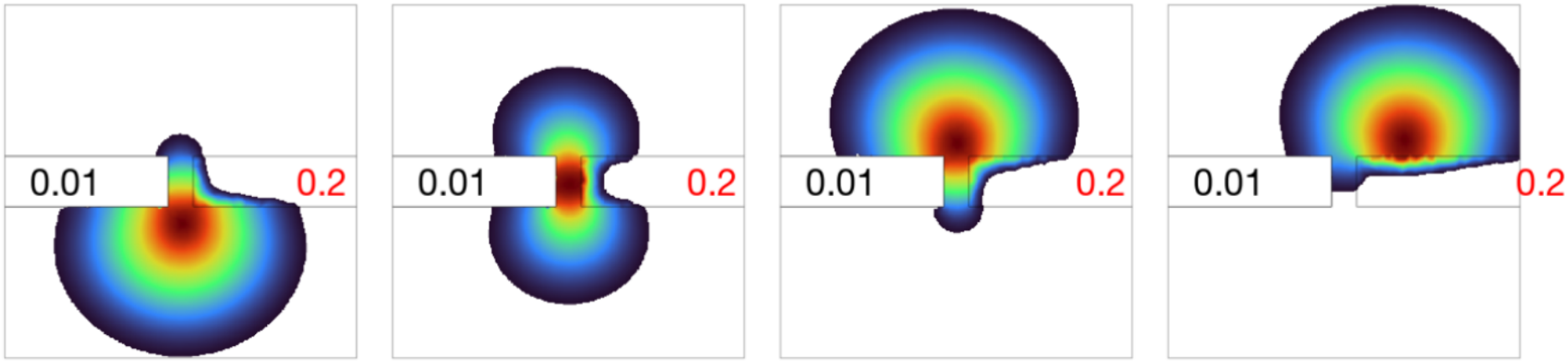
Transparent barrier



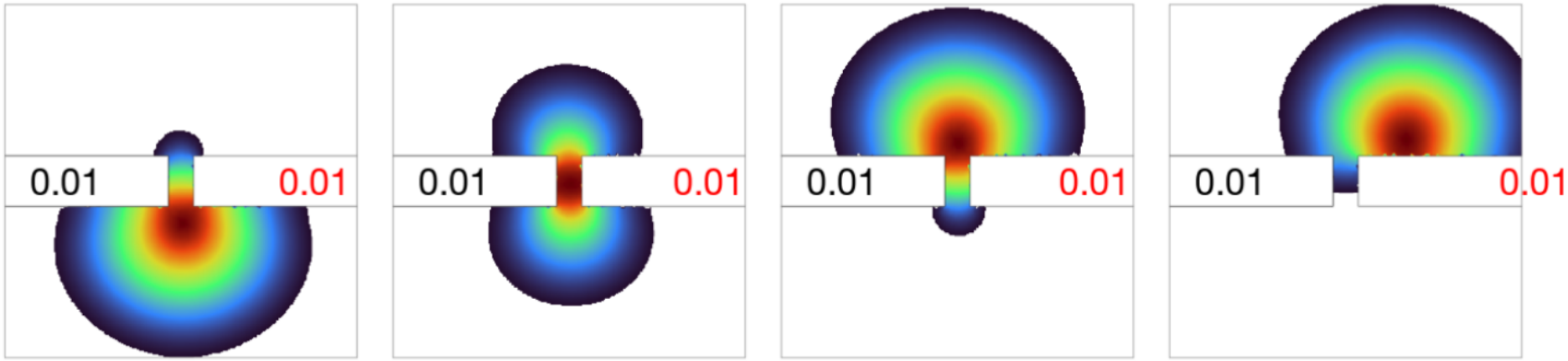
Transparent barrier



Transparent barrier



Transparent barrier



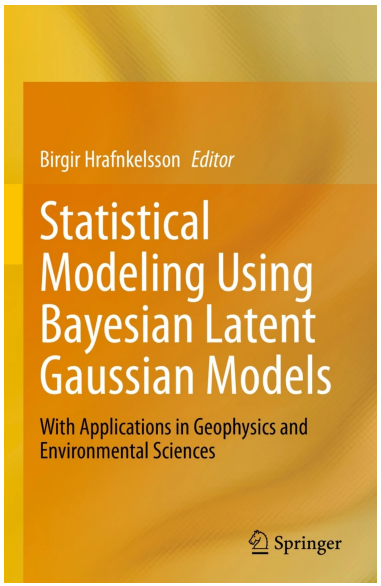
Bayesian inference: Latent Gaussian models (LGMs)

The most important class of models...

- Conditionally independent data $y_i|\eta_i, \theta$
- Latent

$$\eta_i = \sum_j \beta_j c_{ij} + \sum_k f_{ki}$$

- Hyperparameters θ (latent and/or likelihood)
- Each component is a GMRF (or a "small" dense)



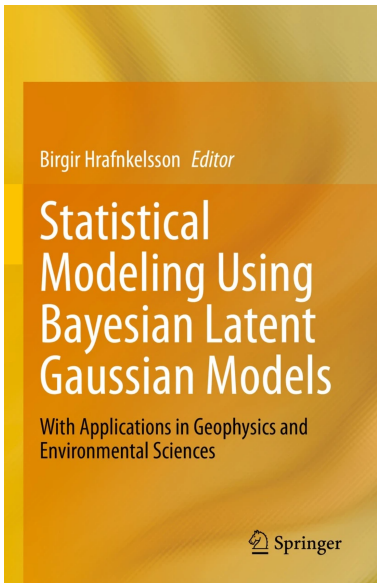
Bayesian inference: Latent Gaussian models (LGMs)

The most important class of models...

- Conditionally independent data $y_i|\eta_i, \boldsymbol{\theta}$
- Latent

$$\eta_i = \sum_j \beta_j c_{ij} + \sum_k f_{ki}$$

- Hyperparameters θ (latent and/or likelihood)
- Each component is a GMRF (or a "small" dense)



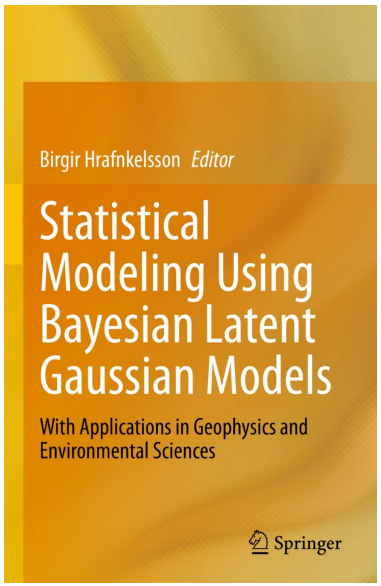
Bayesian inference: Latent Gaussian models (LGMs)

The most important class of models...

- Conditionally independent data $y_i|\eta_i, \boldsymbol{\theta}$
- Latent

$$\eta_i = \sum_j \beta_j c_{ij} + \sum_k f_{ki}$$

- Hyperparameters $\boldsymbol{\theta}$ (latent and/or likelihood)
- Each component is a GMRF (or a "small" dense)



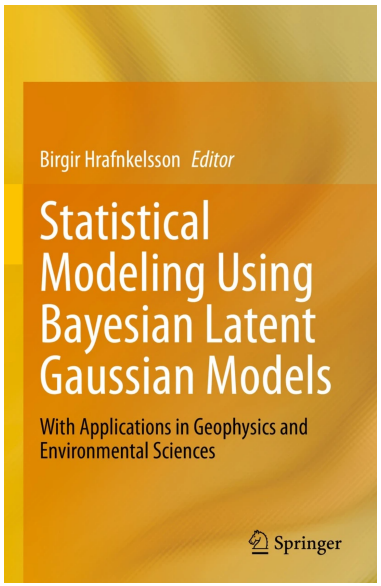
Bayesian inference: Latent Gaussian models (LGMs)

The most important class of models...

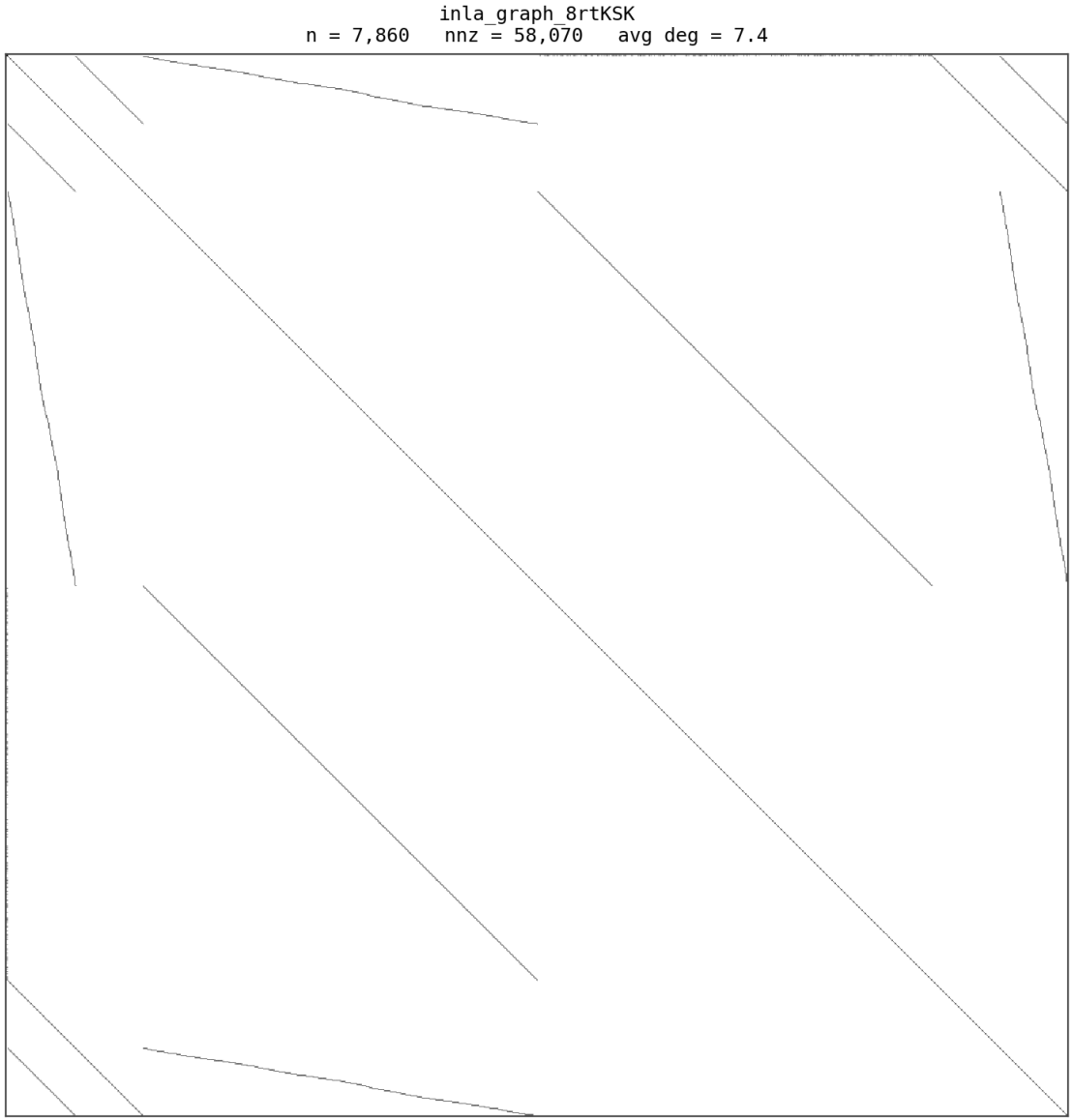
- Conditionally independent data $y_i|\eta_i, \boldsymbol{\theta}$
- Latent

$$\eta_i = \sum_j \beta_j c_{ij} + \sum_k f_{ki}$$

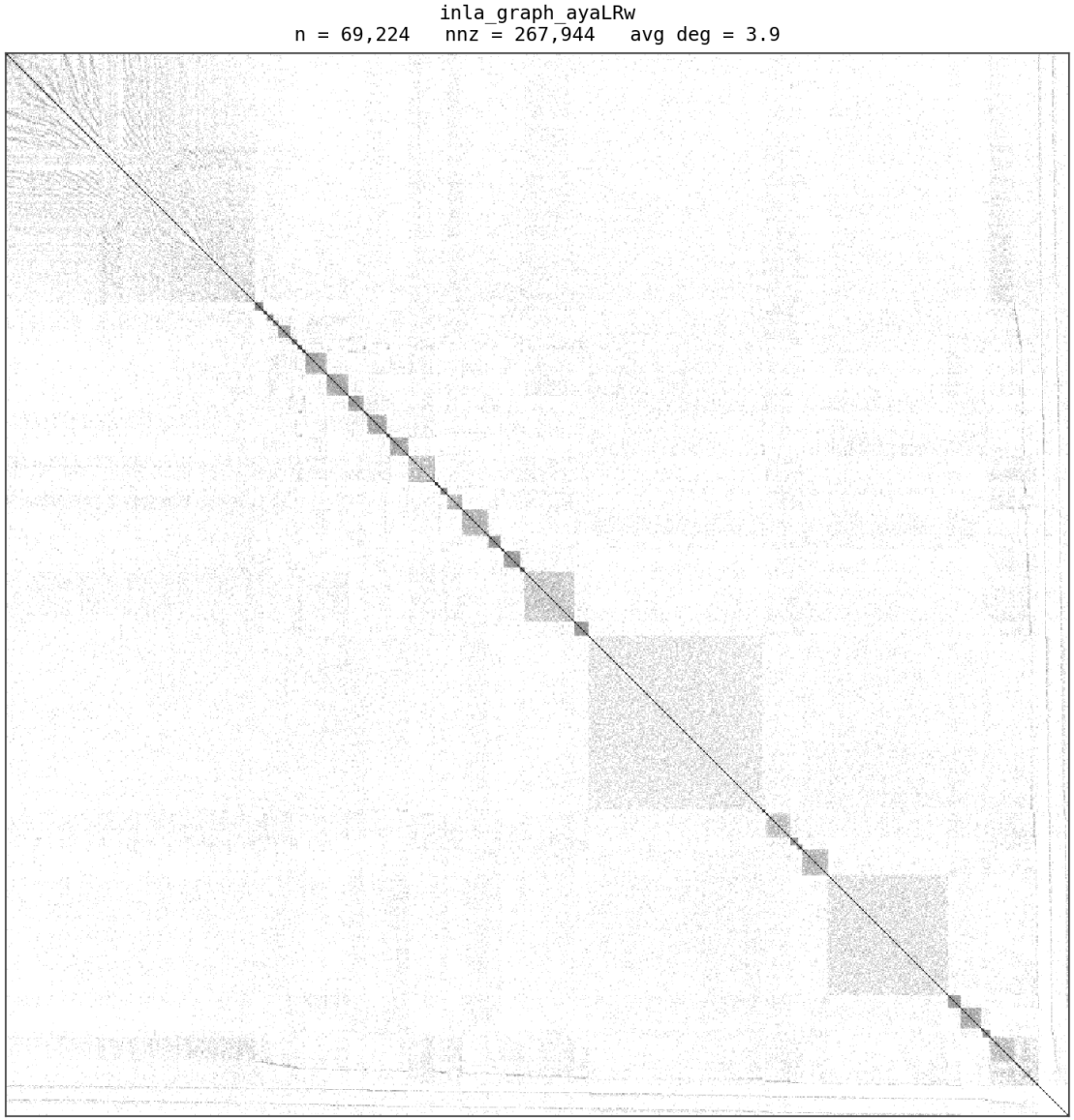
- Hyperparameters $\boldsymbol{\theta}$ (latent and/or likelihood)
- Each component is a GMRF (or a "small" dense)



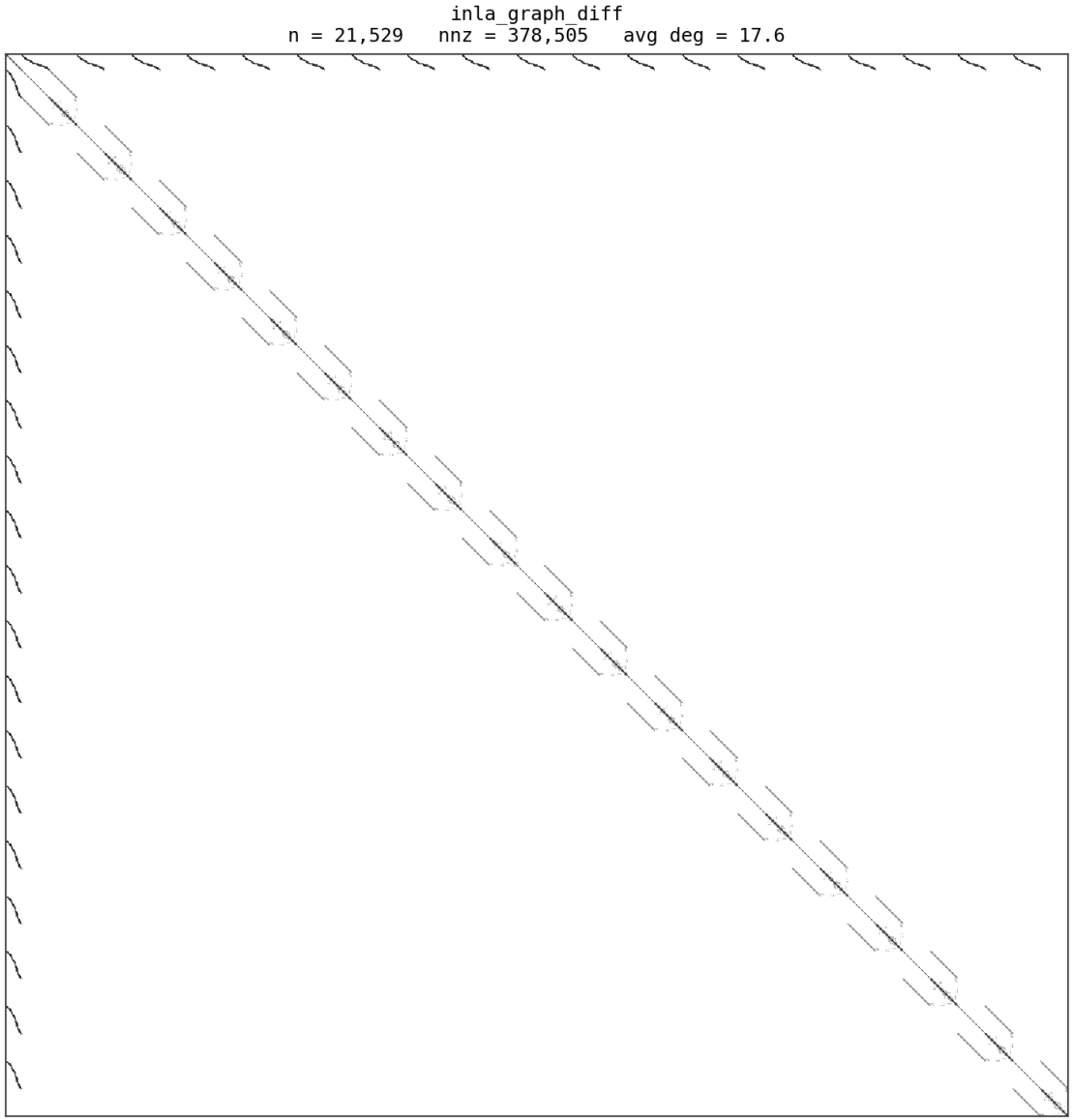
Examples



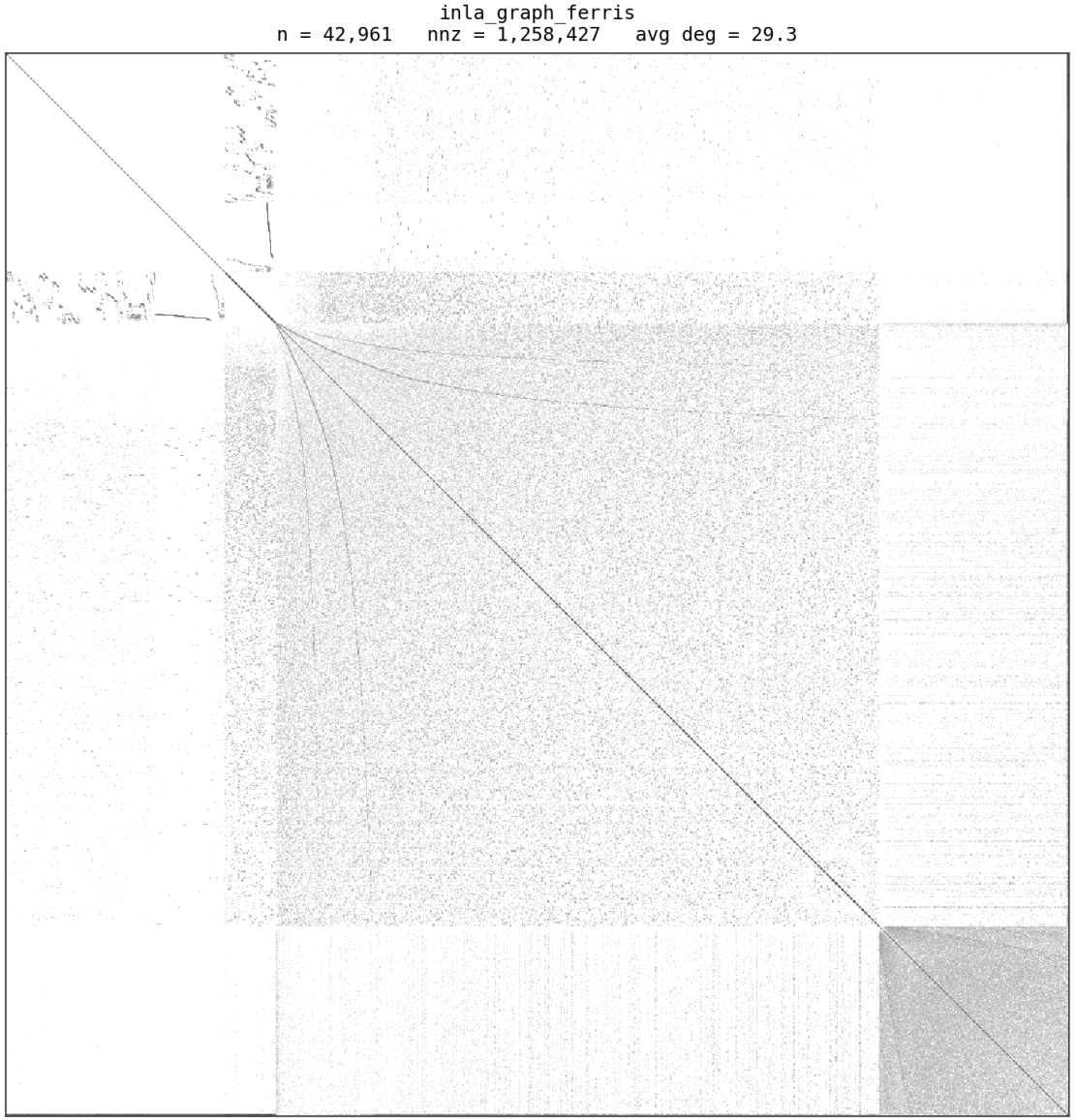
Examples



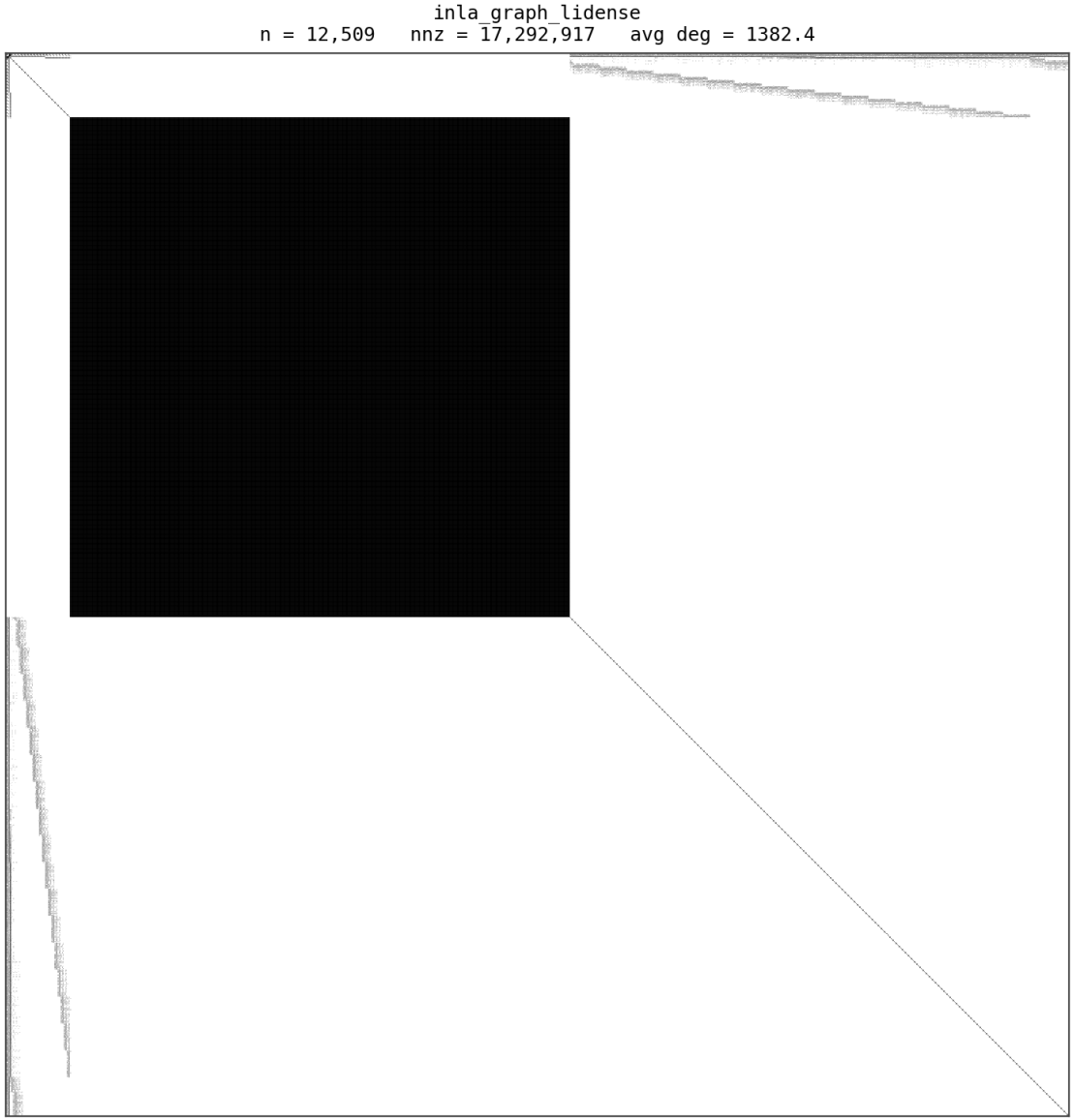
Examples



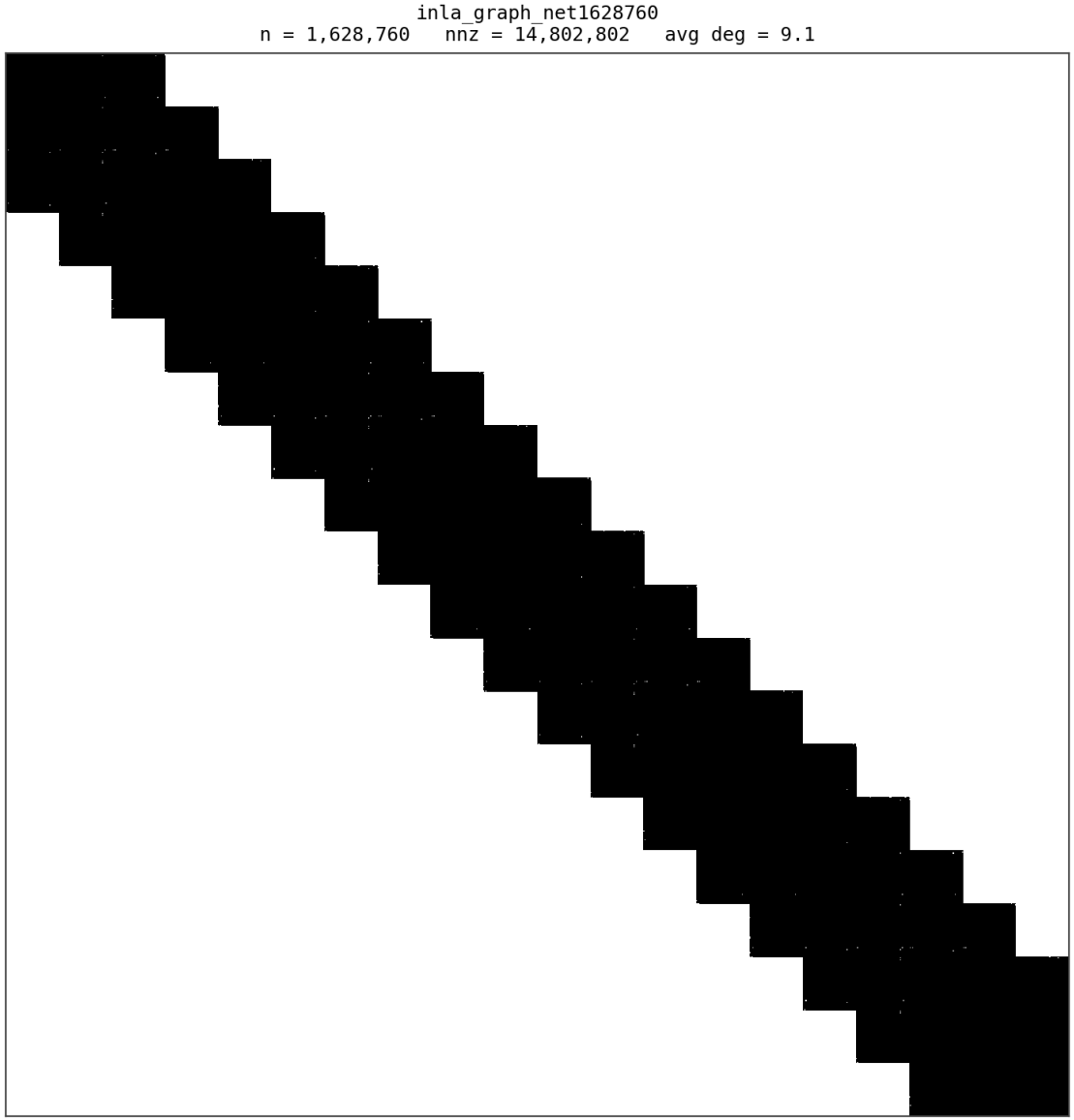
Examples



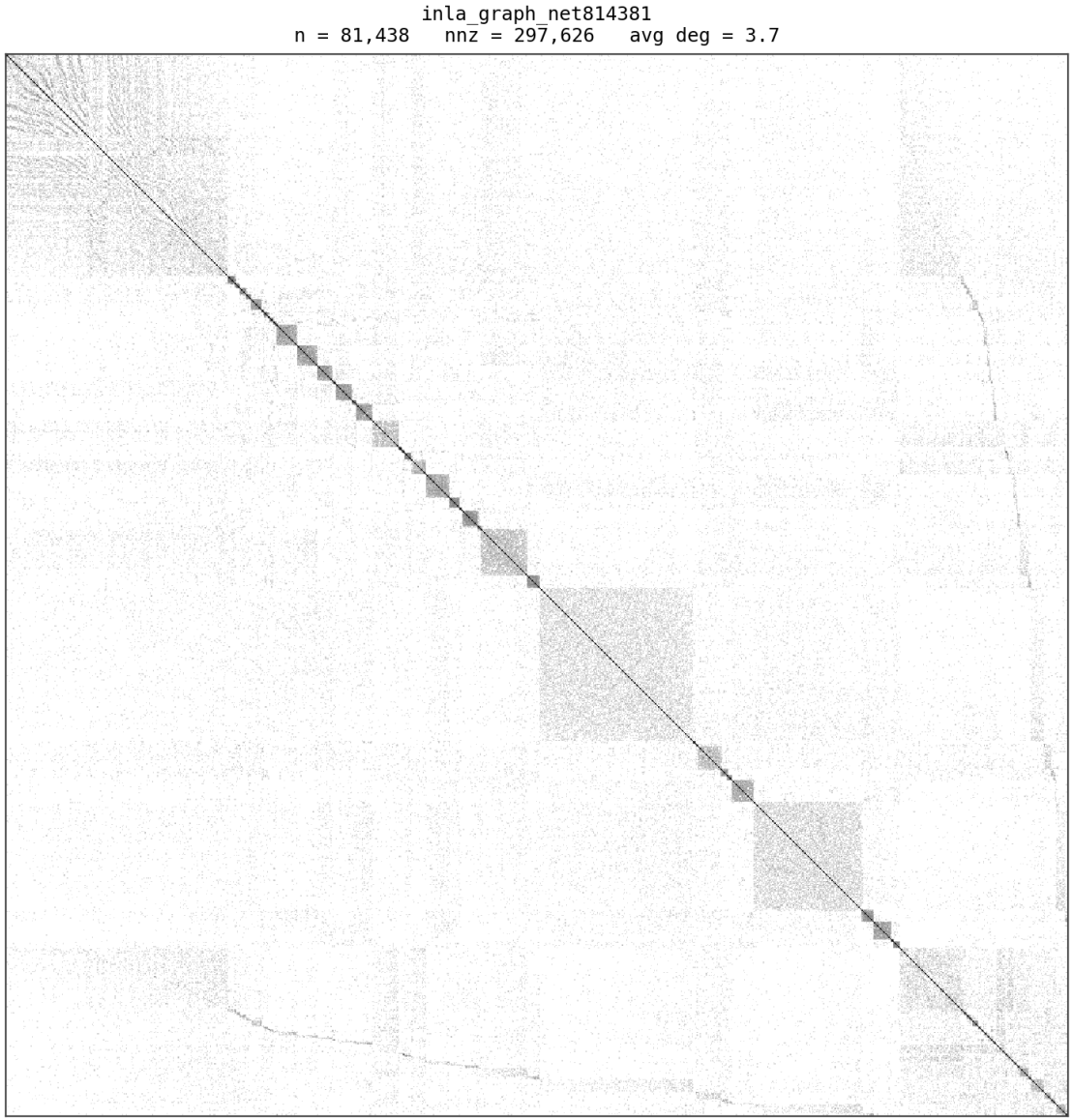
Examples



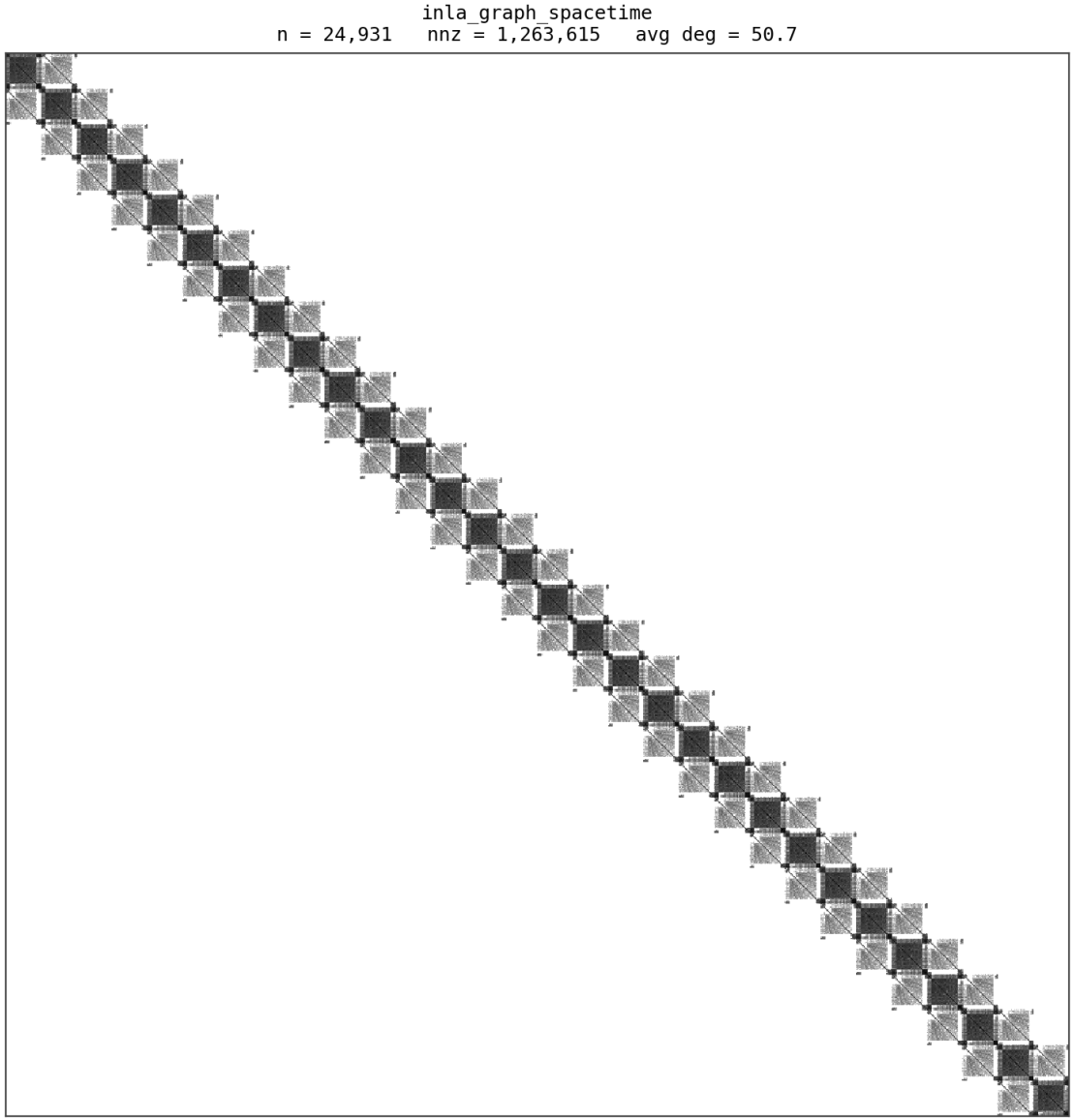
Examples



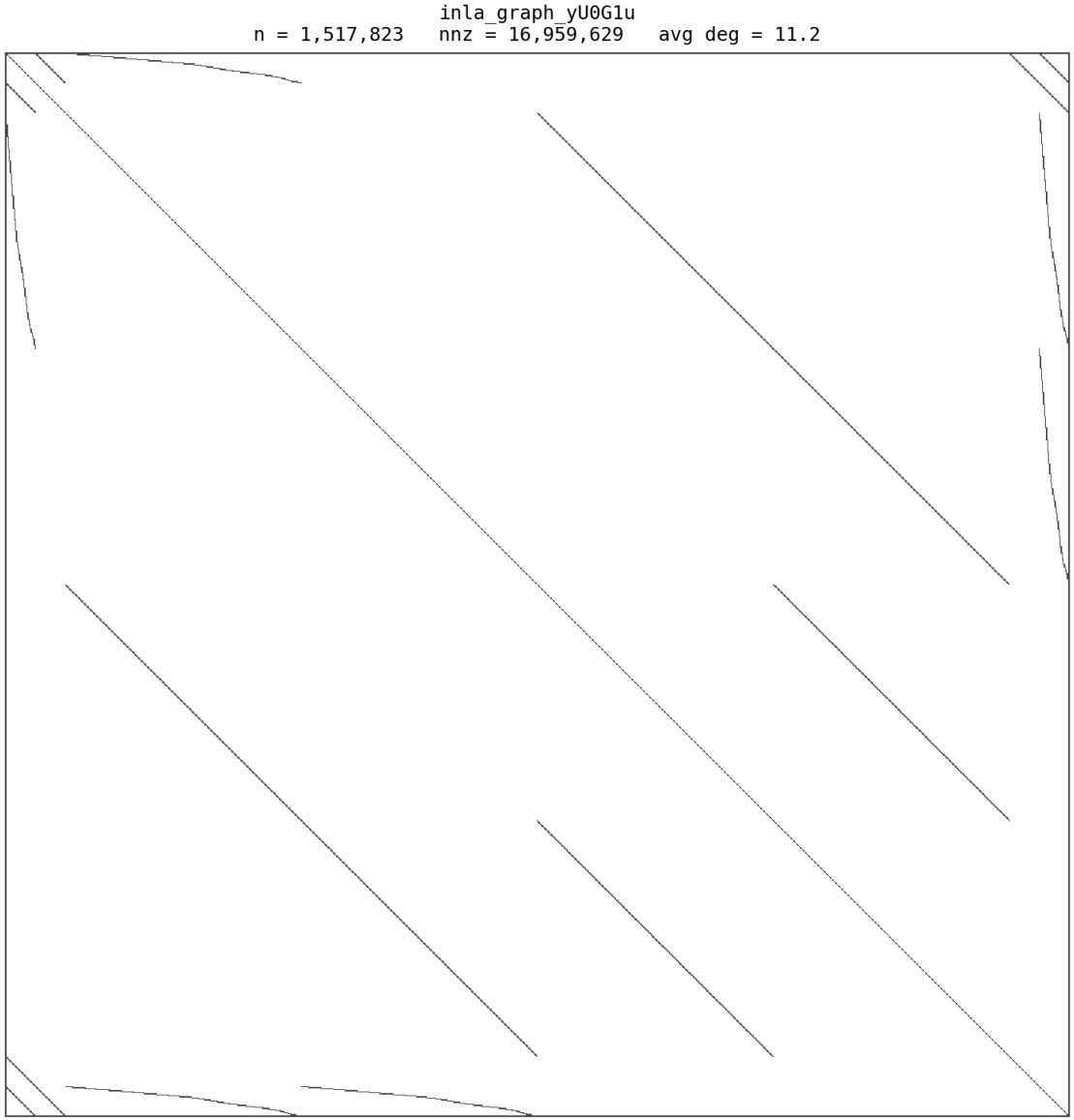
Examples



Examples



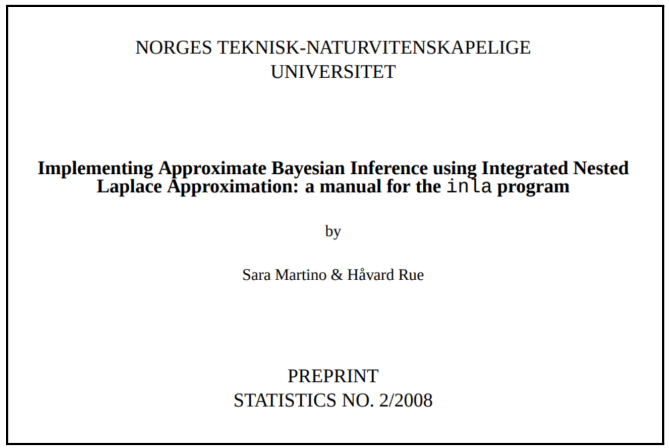
Examples



Inference: simple ideas

$$\pi(\boldsymbol{\theta}|\boldsymbol{y}) \propto \int \pi(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{\theta})\,\pi(\boldsymbol{x}|\boldsymbol{\theta})\,d\boldsymbol{x}$$

$$\pi(x_i|\boldsymbol{y}) \propto \iint \pi(\boldsymbol{x}|\boldsymbol{\theta})\pi(\boldsymbol{\theta}|\boldsymbol{y})\,d\boldsymbol{x}_{-i}\,d\boldsymbol{\theta}$$



NORGES TEKNISK-NATURVITENSKAPELIGE UNIVERSITET

Implementing Approximate Bayesian Inference using Integrated Nested Laplace Approximation: a manual for the `inla` program

by

Sara Martino & Håvard Rue

PREPRINT
STATISTICS NO. 2/2008

Poisson example

- Count-data $y_j \sim \text{Poisson}(\exp(x_j))$
- Latent GMRF $\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{Q})$

Need to approximate

$$\pi(\boldsymbol{x}|\boldsymbol{y}) \propto \exp\left(-\frac{1}{2}\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} + \sum_j y_j x_j - \exp(x_j)\right)$$

with something "manageable", and also approximate marginals

$$\pi(x_i|\boldsymbol{y}) \propto \int \exp\left(-\frac{1}{2}\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} + \sum_j y_j x_j - \exp(x_j)\right) d\boldsymbol{x}_{-i}$$

Poisson example

- Count-data $y_j \sim \text{Poisson}(\exp(x_j))$
- Latent GMRF $\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{Q})$

Need to approximate

$$\pi(\boldsymbol{x}|\boldsymbol{y}) \propto \exp\left(-\frac{1}{2}\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} + \sum_j y_j x_j - \exp(x_j)\right)$$

with something "manageable", and also approximate marginals

$$\pi(x_i|\boldsymbol{y}) \propto \int \exp\left(-\frac{1}{2}\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} + \sum_j y_j x_j - \exp(x_j)\right) d\boldsymbol{x}_{-i}$$

Poisson example

- Count-data $y_j \sim \text{Poisson}(\exp(x_j))$
- Latent GMRF $\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{Q})$

Need to approximate

$$\pi(\boldsymbol{x}|\boldsymbol{y}) \propto \exp\left(-\frac{1}{2}\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} + \sum_j y_j x_j - \exp(x_j)\right)$$

with something "manageable", and also approximate marginals

$$\pi(x_i|\boldsymbol{y}) \propto \int \exp\left(-\frac{1}{2}\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} + \sum_j y_j x_j - \exp(x_j)\right) d\boldsymbol{x}_{-i}$$

Poisson example

- Count-data $y_j \sim \text{Poisson}(\exp(x_j))$
- Latent GMRF $\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{Q})$

Need to approximate

$$\pi(\boldsymbol{x}|\boldsymbol{y}) \propto \exp\left(-\frac{1}{2}\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} + \sum_j y_j x_j - \exp(x_j)\right)$$

with something "manageable", and also approximate marginals

$$\pi(x_i|\boldsymbol{y}) \propto \int \exp\left(-\frac{1}{2}\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} + \sum_j y_j x_j - \exp(x_j)\right) d\boldsymbol{x}_{-i}$$

2nd order approximations

The starting point is that

$$\exp\left(-\frac{1}{2}\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} + \sum_j y_j x_j - \exp(x_j)\right)$$

$$\approx \text{constant} \times \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T(\boldsymbol{Q}_{\text{prior}} + \boldsymbol{Q}_{\text{loglike}})(\boldsymbol{x}-\boldsymbol{\mu})\right)$$

- We need many of those
- Constant graph; that is good
- We can do better than Taylor-expansion

2nd order approximations

The starting point is that

$$\exp\left(-\frac{1}{2}\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} + \sum_j y_j x_j - \exp(x_j)\right)$$

$$\approx \text{constant} \times \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T(\boldsymbol{Q}_{\text{prior}} + \boldsymbol{Q}_{\text{loglike}})(\boldsymbol{x}-\boldsymbol{\mu})\right)$$

- We need many of those
- Constant graph; that is good
- We can do better than Taylor-expansion

2nd order approximations

The starting point is that

$$\exp\left(-\frac{1}{2}\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} + \sum_j y_j x_j - \exp(x_j)\right)$$

$$\approx \text{constant} \times \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T(\boldsymbol{Q}_{\text{prior}} + \boldsymbol{Q}_{\text{loglike}})(\boldsymbol{x}-\boldsymbol{\mu})\right)$$

- We need many of those
- Constant graph; that is good
- We can do better than Taylor-expansion

2nd order approximations

The starting point is that

$$\exp\left(-\frac{1}{2}\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} + \sum_j y_j x_j - \exp(x_j)\right)$$

$$\approx \text{constant} \times \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T(\boldsymbol{Q}_{\text{prior}} + \boldsymbol{Q}_{\text{loglike}})(\boldsymbol{x}-\boldsymbol{\mu})\right)$$

- We need many of those
- Constant graph; that is good
- We can do better than Taylor-expansion

Bayes thm from optimal information processing

From the *American Statistician, 1988* (with discussion)

Optimal Information Processing and Bayes's Theorem

ARNOLD ZELLNER*

In this article statistical inference is viewed as information processing involving input information and output information. After introducing information measures for the input and output information, an information criterion functional is formulated and optimized to obtain an optimal information processing rule (IPR). For the particular information measures and criterion functional adopted, it is shown that Bayes's theorem is the optimal IPR. This optimal IPR is shown to be 100% efficient in the sense that its use leads to the output information being exactly equal to the given input information. Also, the analysis links Bayes's theorem to maximum-entropy considerations.

KEY WORDS: Information theory; Maximum entropy; Statistical inference.

$\pi_a(\theta|I)$ $\equiv$ given prior or antedata probability density function (pdf) for $\theta \subset \Theta$, based on prior information I,

$\pi_p(\theta|D)$ $\equiv$ postdata pdf for $\theta \subset \Theta$, where $D = (y, I)$, the given sample, y, and prior information, I,

$p(y|I)$ $\equiv$ pdf for y, given by

$$p(y|I) \equiv \int_\theta \pi_a(\theta|I) f(y|\theta) d\theta, \tag{2.1}$$

where $f(y|\theta)$ is the pdf for y given θ. [The term *postdata pdf* is employed instead of *posterior pdf* to emphasize that the optimal form of $\pi_p(\theta|D)$ is to be derived.]

Note that (2.1) is a definition that does not involve the assumption that the product rule of probability theory necessarily holds. See Jeffreys (1967, pp. 25, 52) for a discussion of assumptions needed for the product rule to be valid.

The inputs and outputs of any information processing rule (IPR) are depicted graphically in Figure 1, where $l(\theta|y)$,

Bayes thm from optimal information processing

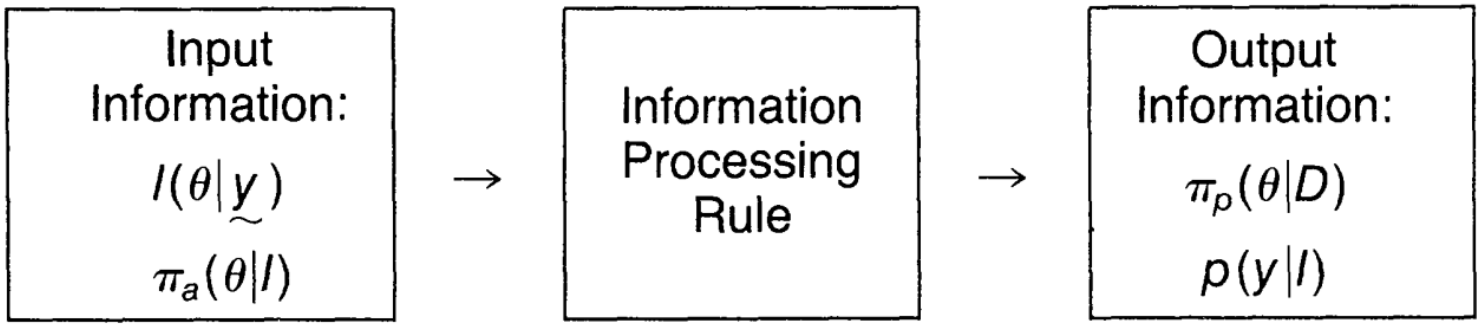


Figure 1. Inputs and Outputs of Any Information Processing Rule.

Bayes thm from optimal information processing (VB)

$$q^*(\boldsymbol{\theta}) = \arg\min_{q \in \mathcal{P}(\boldsymbol{\Theta})} \left\{ -\mathrm{E}_{q(\boldsymbol{\theta})} \log \pi(\boldsymbol{y}|\boldsymbol{\theta}) + \mathrm{KLD}(q(\boldsymbol{\theta}) \| \pi(\boldsymbol{\theta})) \right\}$$

where

- $\mathcal{P}(\boldsymbol{\Theta})$ is the set of all probability distributions on $\boldsymbol{\Theta}$
- $\mathrm{KLD}(q(\boldsymbol{\theta}) \| \pi(\boldsymbol{\theta}))$ is the Kullback-Leibler divergence $\mathrm{E}_{q(\boldsymbol{\theta})} \log \frac{q(\boldsymbol{\theta})}{\pi(\boldsymbol{\theta})}$

Bayes thm gives

$$\pi(\boldsymbol{\theta}|\boldsymbol{y}) \propto \pi(\boldsymbol{y}|\boldsymbol{\theta})\pi(\boldsymbol{\theta})$$

and of course

$$q^*(\boldsymbol{\theta}) = \pi(\boldsymbol{\theta}|\boldsymbol{y})$$

Bayes thm from optimal information processing (VB)

$$q^*(\boldsymbol{\theta}) = \arg\min_{q \in \mathcal{P}(\boldsymbol{\Theta})} \left\{ -\mathrm{E}_{q(\boldsymbol{\theta})} \log \pi(\boldsymbol{y}|\boldsymbol{\theta}) + \mathrm{KLD}(q(\boldsymbol{\theta}) \| \pi(\boldsymbol{\theta})) \right\}$$

where

- $\mathcal{P}(\boldsymbol{\Theta})$ is the set of all probability distributions on $\boldsymbol{\Theta}$
- $\mathrm{KLD}(q(\boldsymbol{\theta}) \| \pi(\boldsymbol{\theta}))$ is the Kullback-Leibler divergence $\mathrm{E}_{q(\boldsymbol{\theta})} \log \frac{q(\boldsymbol{\theta})}{\pi(\boldsymbol{\theta})}$

Bayes thm gives

$$\pi(\boldsymbol{\theta}|\boldsymbol{y}) \propto \pi(\boldsymbol{y}|\boldsymbol{\theta})\pi(\boldsymbol{\theta})$$

and of course

$$q^*(\boldsymbol{\theta}) = \pi(\boldsymbol{\theta}|\boldsymbol{y})$$

Bayes thm from optimal information processing (VB)

$$q^*(\boldsymbol{\theta}) = \arg\min_{q \in \mathcal{P}(\boldsymbol{\Theta})} \left\{ -\mathrm{E}_{q(\boldsymbol{\theta})} \log \pi(\boldsymbol{y}|\boldsymbol{\theta}) + \mathrm{KLD}(q(\boldsymbol{\theta}) \| \pi(\boldsymbol{\theta})) \right\}$$

where

- $\mathcal{P}(\boldsymbol{\Theta})$ is the set of all probability distributions on $\boldsymbol{\Theta}$
- $\mathrm{KLD}(q(\boldsymbol{\theta}) \| \pi(\boldsymbol{\theta}))$ is the Kullback-Leibler divergence $\mathrm{E}_{q(\boldsymbol{\theta})} \log \frac{q(\boldsymbol{\theta})}{\pi(\boldsymbol{\theta})}$

Bayes thm gives

$$\pi(\boldsymbol{\theta}|\boldsymbol{y}) \propto \pi(\boldsymbol{y}|\boldsymbol{\theta})\pi(\boldsymbol{\theta})$$

and of course

$$q^*(\boldsymbol{\theta}) = \pi(\boldsymbol{\theta}|\boldsymbol{y})$$

Variational inference

$$\widehat{q}^*(\boldsymbol{\theta}) = \arg \min_{q \in \widehat{\mathcal{P}}(\boldsymbol{\Theta})} \left\{ -\mathrm{E}_{q(\boldsymbol{\theta})} \log \pi(\boldsymbol{y}|\boldsymbol{\theta}) + \mathrm{KLD}(q(\boldsymbol{\theta}) \| \pi(\boldsymbol{\theta})) \right\}$$

where

- $\widehat{\mathcal{P}}(\boldsymbol{\Theta})$ is the set of all approximate probability distributions on $\boldsymbol{\Theta}$
- same objective as optimising ELBO/VB (in ML)
- Zellner(1988) provides the justification

Variational inference

$$\widehat{q}^*(\boldsymbol{\theta}) = \arg \min_{q \in \widehat{\mathcal{P}}(\boldsymbol{\Theta})} \left\{ -\mathrm{E}_{q(\boldsymbol{\theta})} \log \pi(\boldsymbol{y}|\boldsymbol{\theta}) + \mathrm{KLD}(q(\boldsymbol{\theta}) \| \pi(\boldsymbol{\theta})) \right\}$$

where

- $\widehat{\mathcal{P}}(\boldsymbol{\Theta})$ is the set of all approximate probability distributions on $\boldsymbol{\Theta}$
- same objective as optimising ELBO/VB (in ML)
- Zellner(1988) provides the justification

Variational inference

$$\widehat{q}^*(\boldsymbol{\theta}) = \arg \min_{q \in \widehat{\mathcal{P}}(\boldsymbol{\Theta})} \left\{ -\mathrm{E}_{q(\boldsymbol{\theta})} \log \pi(\boldsymbol{y}|\boldsymbol{\theta}) + \mathrm{KLD}(q(\boldsymbol{\theta}) \| \pi(\boldsymbol{\theta})) \right\}$$

where

- $\widehat{\mathcal{P}}(\boldsymbol{\Theta})$ is the set of all approximate probability distributions on $\boldsymbol{\Theta}$
- same objective as optimising ELBO/VB (in ML)
- Zellner(1988) provides the justification

VB *is* valid Bayesian inference

Forrest Gump 1994

Example: VB correcting Poisson regression

$$y_i \sim \text{Poisson}(\exp(\beta x_i)), \qquad \beta \sim \mathcal{N}(0, 1)$$

log-posterior with constant prior

$$U(\beta) = -\frac{1}{2}\beta^2 + \beta \sum_i x_i y_i - \sum_i exp(\beta x_i) + \text{constant}$$

Gaussian approxmation/"Laplace method" found using MLE/`glm`

$$\beta|\boldsymbol{y} \approx \mathcal{N}(\hat{\beta}, \hat{\sigma}^2)$$

$$\hat{\beta} = \arg\max U(\beta), \qquad \frac{1}{\hat{\sigma}^2} = -\text{Hessian}(U(\beta))\,|_{\beta=\hat{\beta}}$$

Example: VB correcting Poisson regression

$$y_i \sim \text{Poisson}(\exp(\beta x_i)), \qquad \beta \sim \mathcal{N}(0, 1)$$

log-posterior with constant prior

$$U(\beta) = -\frac{1}{2}\beta^2 + \beta \sum_i x_i y_i - \sum_i exp(\beta x_i) + \text{constant}$$

Gaussian approxmation/"Laplace method" found using MLE/`glm`

$$\beta|\boldsymbol{y} \approx \mathcal{N}(\hat{\beta}, \hat{\sigma}^2)$$

$$\hat{\beta} = \arg\max U(\beta), \qquad \frac{1}{\hat{\sigma}^2} = -\text{Hessian}(U(\beta))\,|_{\beta=\hat{\beta}}$$

Example: VB correcting Poisson regression

$$y_i \sim \text{Poisson}(\exp(\beta x_i)), \qquad \beta \sim \mathcal{N}(0, 1)$$

log-posterior with constant prior

$$U(\beta) = -\frac{1}{2}\beta^2 + \beta \sum_i x_i y_i - \sum_i exp(\beta x_i) + \text{constant}$$

Gaussian approxmation/"Laplace method" found using MLE/`glm`

$$\beta|\boldsymbol{y} \approx \mathcal{N}(\hat{\beta}, \hat{\sigma}^2)$$

$$\hat{\beta} = \arg\max U(\beta), \qquad \frac{1}{\hat{\sigma}^2} = -\text{Hessian}(U(\beta))\,|_{\beta=\hat{\beta}}$$

Example: VB correcting Poisson regression (II)

Mean correction with a $\mathcal{N}(0, 1)$ prior

$$\beta \approx \mathcal{N}(\hat{\beta} + \delta, \hat{\sigma}^2)$$

Since the MLE-approxmation is already useful, then

- $|\delta|$ should be small
- optimization should be fast and numerical stable

Example: VB correcting Poisson regression (II)

Mean correction with a $\mathcal{N}(0, 1)$ prior

$$\beta \approx \mathcal{N}(\hat{\beta} + \delta, \hat{\sigma}^2)$$

Since the MLE-approxmation is already useful, then

- $|\delta|$ should be small
- optimization should be fast and numerical stable

Example: VB correcting Poisson regression (III)

The VB correction will minimise

$$-\mathrm{E}_{q(\delta)}\text{log-likelihood} + \mathrm{KLD}(q(\delta)||\text{prior})$$

Example: VB correcting Poisson regression (IV)

The KLD between two multivariate Gaussians is

$$D_{\mathrm{KL}}\left(\mathcal{N}_0 \parallel \mathcal{N}_1\right) = \frac{1}{2}\left(\operatorname{tr}\left(\Sigma_1^{-1}\Sigma_0\right) - k + \left(\mu_1 - \mu_0\right)^{\mathsf{T}}\Sigma_1^{-1}\left(\mu_1 - \mu_0\right) + \ln\left(\frac{\det\Sigma_1}{\det\Sigma_0}\right)\right)$$

Here, $()_1$ is the "prior" and $()_0$ is the "posterior" (the notation is backwards in the Wikipedia page...)

Since we keep the variance constant, the KLD simplifies into

$$\mathrm{KLD}(\delta) = \frac{1}{2}(\hat{\beta} + \delta)^2 + \text{constant}$$

Example: VB correcting Poisson regression (IV)

The KLD between two multivariate Gaussians is

$$D_{\mathrm{KL}}\left(\mathcal{N}_0 \parallel \mathcal{N}_1\right) = \frac{1}{2}\left(\operatorname{tr}\left(\Sigma_1^{-1}\Sigma_0\right) - k + \left(\mu_1 - \mu_0\right)^{\mathsf{T}}\Sigma_1^{-1}\left(\mu_1 - \mu_0\right) + \ln\left(\frac{\det\Sigma_1}{\det\Sigma_0}\right)\right)$$

Here, $()_1$ is the "prior" and $()_0$ is the "posterior" (the notation is backwards in the Wikipedia page...)

Since we keep the variance constant, the KLD simplifies into

$$\mathrm{KLD}(\delta) = \frac{1}{2}(\hat{\beta} + \delta)^2 + \text{constant}$$

Example: VB correcting Poisson regression (V)

The negative expected likelihood is

$$-E_{q(\delta)}\left[\beta \sum_i x_i y_i - \sum_i \exp(\beta x_i)\right] + \text{constant}$$

Since $q(\delta)$ is Normal, we can do this directly

$$-(\hat{\beta}+\delta)\sum_i x_i y_i + \sum_i \exp\left((\hat{\beta}+\delta)x_i + \frac{1}{2}x_i^2\hat{\sigma}^2\right)$$

Example: VB correcting Poisson regression (V)

The negative expected likelihood is

$$-E_{q(\delta)}\left[\beta \sum_i x_i y_i - \sum_i \exp(\beta x_i)\right] + \text{constant}$$

Since $q(\delta)$ is Normal, we can do this directly

$$-(\hat{\beta}+\delta)\sum_i x_i y_i + \sum_i \exp\left((\hat{\beta}+\delta)x_i + \frac{1}{2}x_i^2\hat{\sigma}^2\right)$$

Example: VB correcting Poisson regression (V)

The negative expected likelihood is

$$-E_{q(\delta)}\left[\beta \sum_i x_i y_i - \sum_i \exp(\beta x_i)\right] + \text{constant}$$

Since $q(\delta)$ is Normal, we can do this directly

$$-(\hat{\beta}+\delta)\sum_i x_i y_i + \sum_i \exp\left((\hat{\beta}+\delta)x_i + \frac{1}{2}x_i^2\hat{\sigma}^2\right)$$

Example: VB correcting Poisson regression (VI)

Adding the two terms, we get

$$\hat{\delta} = \arg\min \left(A\delta + \frac{1}{2} B\delta^2 + \sum_i C_i \exp(D_i \delta) \right) = \arg\min V(\delta)$$

for some $A, B, \{C_i\}, \{D_i\}$.

Since we are already pretty close to the solution, then

$$\hat{\delta} \approx -\frac{V'(0)}{V''(0)} = -\frac{A + \sum_i C_i D_i}{B + \sum_i C_i D_i^2}$$

(can iterate if needed)

Example: VB correcting Poisson regression (VI)

Adding the two terms, we get

$$\hat{\delta} = \arg\min \left(A\delta + \frac{1}{2} B\delta^2 + \sum_i C_i \exp(D_i \delta) \right) = \arg\min V(\delta)$$

for some $A, B, \{C_i\}, \{D_i\}$.

Since we are already pretty close to the solution, then

$$\hat{\delta} \approx -\frac{V'(0)}{V''(0)} = -\frac{A + \sum_i C_i D_i}{B + \sum_i C_i D_i^2}$$

(can iterate if needed)

Computational issues

Numerical methods for sparse matrices

- Factorise $\boldsymbol{Q} = \boldsymbol{L}\boldsymbol{L}^T$
- Partial inverse $\boldsymbol{Q}^{-1}$ for all $i \sim j$ or $i = j$.
- Determinant $\log |\boldsymbol{Q}|$
- Solve $\boldsymbol{Q}\boldsymbol{x} = \boldsymbol{b}$ and $\boldsymbol{Q}\boldsymbol{X} = \boldsymbol{B}$

```
for (i = 0; i < dimensionSize; i++) {
    for (j = 0; j <= i; j++) {
        float sum = 0;
        for (k = 0; k < j; k++)
            sum += L[i][k] * L[j][k];

        if (i == j)
            L[i][j] = sqrt(A[i][i] - sum);
        else
            L[i][j] = (1.0 / L[j][j] * (A[i][j] - sum));
    }
}
```

Most important ingredient to scalable inference, is to handle sparse matrices well!

Computational issues

Numerical methods for sparse matrices

- Factorise $\boldsymbol{Q} = \boldsymbol{L}\boldsymbol{L}^T$
- Partial inverse $\boldsymbol{Q}^{-1}$ for all $i \sim j$ or $i = j$.
- Determinant log $|\boldsymbol{Q}|$
- Solve $\boldsymbol{Q}\boldsymbol{x} = \boldsymbol{b}$ and $\boldsymbol{Q}\boldsymbol{X} = \boldsymbol{B}$

```
for (i = 0; i < dimensionSize; i++) {
    for (j = 0; j <= i; j++) {
        float sum = 0;
        for (k = 0; k < j; k++)
            sum += L[i][k] * L[j][k];

        if (i == j)
            L[i][j] = sqrt(A[i][i] - sum);
        else
            L[i][j] = (1.0 / L[j][j] * (A[i][j] - sum));
    }
}
```

Most important ingredient to scalable inference, is to handle sparse matrices well!

Computational issues

Numerical methods for sparse matrices

- Factorise $\boldsymbol{Q} = \boldsymbol{L}\boldsymbol{L}^T$
- Partial inverse $\boldsymbol{Q}^{-1}$ for all $i \sim j$ or $i = j$.
- Determinant $\log |\boldsymbol{Q}|$
- Solve $\boldsymbol{Q}\boldsymbol{x} = \boldsymbol{b}$ and $\boldsymbol{Q}\boldsymbol{X} = \boldsymbol{B}$

```
for (i = 0; i < dimensionSize; i++) {
    for (j = 0; j <= i; j++) {
        float sum = 0;
        for (k = 0; k < j; k++)
            sum += L[i][k] * L[j][k];

        if (i == j)
            L[i][j] = sqrt(A[i][i] - sum);
        else
            L[i][j] = (1.0 / L[j][j] * (A[i][j] - sum));
    }
}
```

Most important ingredient to scalable inference, is to handle sparse matrices well!

Computational issues

Numerical methods for sparse matrices

- Factorise $\boldsymbol{Q} = \boldsymbol{L}\boldsymbol{L}^T$
- Partial inverse $\boldsymbol{Q}^{-1}$ for all $i \sim j$ or $i = j$.
- Determinant $\log |\boldsymbol{Q}|$
- Solve $\boldsymbol{Q}\boldsymbol{x} = \boldsymbol{b}$ and $\boldsymbol{Q}\boldsymbol{X} = \boldsymbol{B}$

```
for (i = 0; i < dimensionSize; i++) {
    for (j = 0; j <= i; j++) {
        float sum = 0;
        for (k = 0; k < j; k++)
            sum += L[i][k] * L[j][k];

        if (i == j)
            L[i][j] = sqrt(A[i][i] - sum);
        else
            L[i][j] = (1.0 / L[j][j] * (A[i][j] - sum));
    }
}
```

Most important ingredient to scalable inference, is to handle sparse matrices well!

Computational issues

Numerical methods for sparse matrices

- Factorise $\boldsymbol{Q} = \boldsymbol{L}\boldsymbol{L}^T$
- Partial inverse $\boldsymbol{Q}^{-1}$ for all $i \sim j$ or $i = j$.
- Determinant $\log |\boldsymbol{Q}|$
- Solve $\boldsymbol{Q}\boldsymbol{x} = \boldsymbol{b}$ and $\boldsymbol{Q}\boldsymbol{X} = \boldsymbol{B}$

```
for (i = 0; i < dimensionSize; i++) {
    for (j = 0; j <= i; j++) {
        float sum = 0;
        for (k = 0; k < j; k++)
            sum += L[i][k] * L[j][k];

        if (i == j)
            L[i][j] = sqrt(A[i][i] - sum);
        else
            L[i][j] = (1.0 / L[j][j] * (A[i][j] - sum));
    }
}
```

Most important ingredient to scalable inference, is to handle sparse matrices well!

Computational issues

Numerical methods for sparse matrices

- Factorise $\boldsymbol{Q} = \boldsymbol{L}\boldsymbol{L}^T$
- Partial inverse $\boldsymbol{Q}^{-1}$ for all $i \sim j$ or $i = j$.
- Determinant $\log|\boldsymbol{Q}|$
- Solve $\boldsymbol{Q}\boldsymbol{x} = \boldsymbol{b}$ and $\boldsymbol{Q}\boldsymbol{X} = \boldsymbol{B}$

```
for (i = 0; i < dimensionSize; i++) {
    for (j = 0; j <= i; j++) {
        float sum = 0;
        for (k = 0; k < j; k++)
            sum += L[i][k] * L[j][k];

        if (i == j)
            L[i][j] = sqrt(A[i][i] - sum);
        else
            L[i][j] = (1.0 / L[j][j] * (A[i][j] - sum));
    }
}
```

Most important ingredient to scalable inference, is to handle sparse matrices well!

Not that much around...

- HPC community like “dense” more than “sparse”
- TAUCS (outdated $>$ 20 years ago)
- CHOLMOD (what is used in R, MATLAB, does not scale well in parallel do not handle inverse)
- PARDISO (good, not open-source and but turned more commercial, so...)
- etc...
- With the right person around, we can “just write our own”, right?

Esmail Abdul Fattah

Not that much around...

- HPC community like “dense” more than “sparse”
- TAUCS (outdated $>$ 20 years ago)
- CHOLMOD (what is used in R, MATLAB, does not scale well in parallel do not handle inverse)
- PARDISO (good, not open-source and but turned more commercial, so...)
- etc...
- With the right person around, we can “just write our own”, right?

Esmail Abdul Fattah

Not that much around...

- HPC community like “dense” more than “sparse”
- TAUCS (outdated $>$ 20 years ago)
- CHOLMOD (what is used in R, MATLAB, does not scale well in parallel do not handle inverse)
- PARDISO (good, not open-source and but turned more commercial, so...)
- etc...
- With the right person around, we can “just write our own”, right?

Esmail Abdul Fattah

Not that much around...

- HPC community like “dense” more than “sparse”
- TAUCS (outdated $>$ 20 years ago)
- CHOLMOD (what is used in R, MATLAB, does not scale well in parallel do not handle inverse)
- PARDISO (good, not open-source and but turned more commercial, so...)
- etc...
- With the right person around, we can “just write our own”, right?

Esmail Abdul Fattah

Not that much around...

- HPC community like “dense” more than “sparse”
- TAUCS (outdated $>$ 20 years ago)
- CHOLMOD (what is used in R, MATLAB, does not scale well in parallel do not handle inverse)
- PARDISO (good, not open-source and but turned more commercial, so...)
- etc...
- With the right person around, we can “just write our own”, right?

Esmail Abdul Fattah

Not that much around...

- HPC community like “dense” more than “sparse”
- TAUCS (outdated $>$ 20 years ago)
- CHOLMOD (what is used in R, MATLAB, does not scale well in parallel do not handle inverse)
- PARDISO (good, not open-source and but turned more commercial, so...)
- etc...
- With the right person around, we can “just write our own”, right?

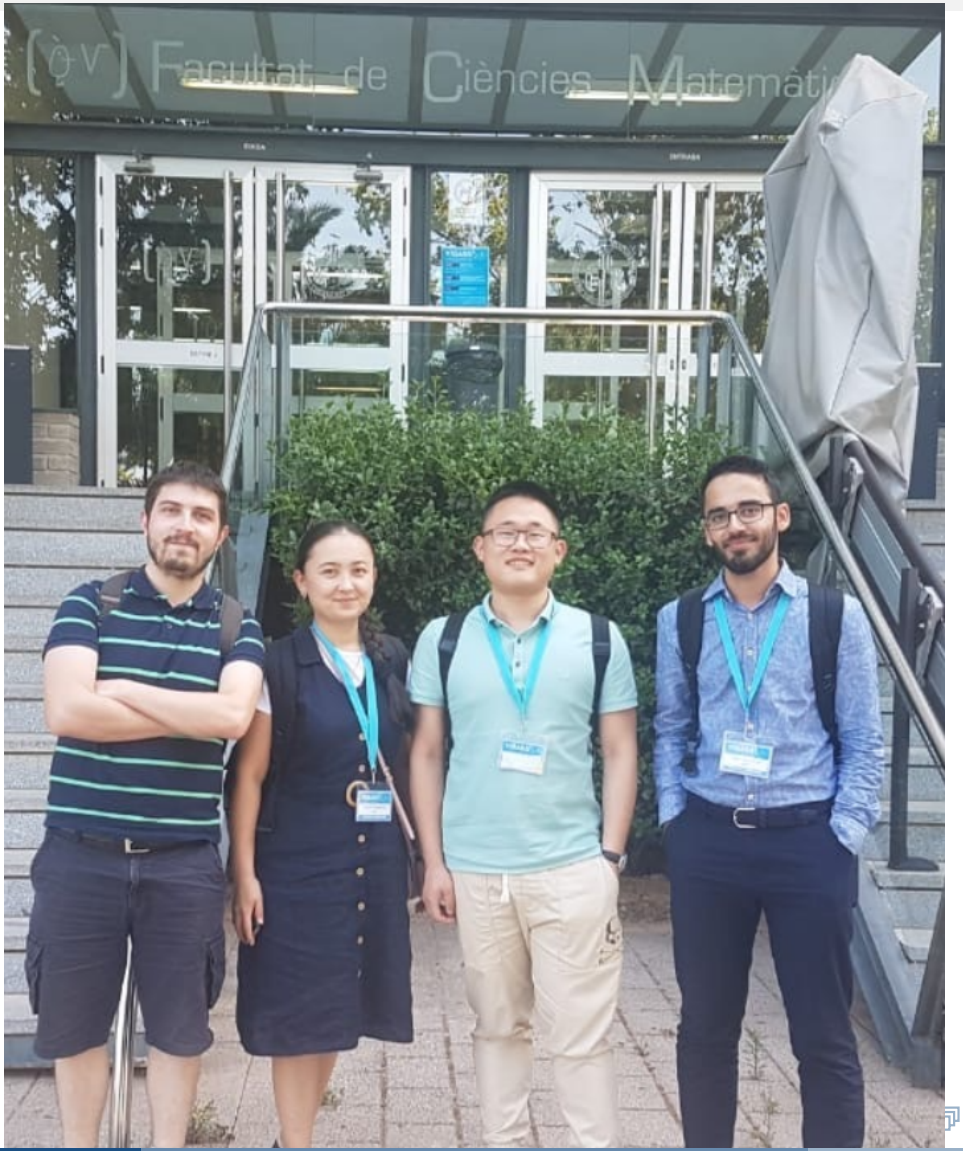
Esmail Abdul Fattah

Cholesky factorisation

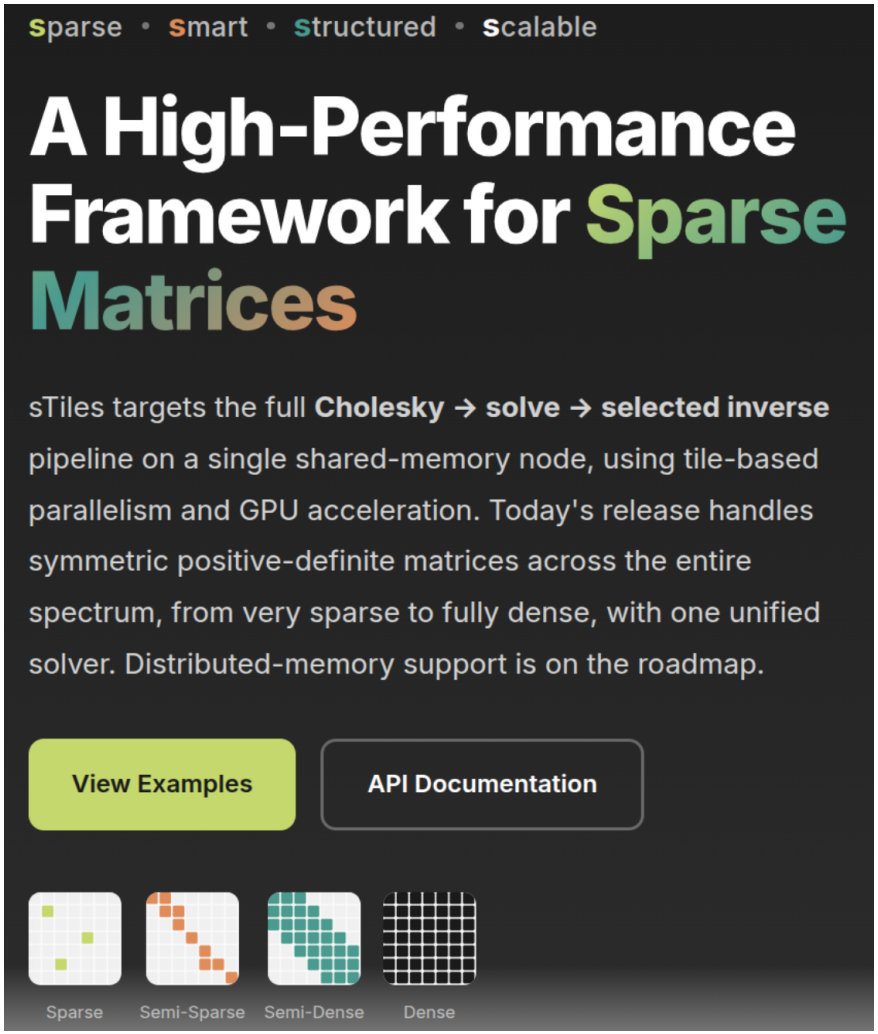
```
for (i = 0; i < dimensionSize; i++) {
    for (j = 0; j <= i; j++) {
        float sum = 0;
        for (k = 0; k < j; k++)
            sum += L[i][k] * L[j][k];

        if (i == j)
            L[i][j] = sqrt(A[i][i] - sum);
        else
            L[i][j] = (1.0 / L[j][j] * (A[i][j] - sum));
    }
}
```

We can, because that person participated in the Valencia summer school!



3 years later...



3 years later...

- We're getting there
- Announced in Glasgow'25 in May
- ...the we found some new problematic cases
- that took another year to fix
- Written for modern CPU and will have GPU acceleration (for hard cases)
- Results are *very* good, with excellent parallel scaling
- Handle hard cases particulary well
- Open source when done
- Stay tuned!

3 years later...

- We're getting there
- Announced in Glasgow'25 in May
- ...the we found some new problematic cases
- that took another year to fix
- Written for modern CPU and will have GPU acceleration (for hard cases)
- Results are *very* good, with excellent parallel scaling
- Handle hard cases particulary well
- Open source when done
- Stay tuned!

sTiles

3 years later...

- We're getting there
- Announced in Glasgow'25 in May
- ...the we found some new problematic cases
- that took another year to fix
- Written for modern CPU and will have GPU acceleration (for hard cases)
- Results are *very* good, with excellent parallel scaling
- Handle hard cases particulary well
- Open source when done
- Stay tuned!

3 years later...

- We're getting there
- Announced in Glasgow'25 in May
- ...the we found some new problematic cases
- that took another year to fix
- Written for modern CPU and will have GPU acceleration (for hard cases)
- Results are *very* good, with excellent parallel scaling
- Handle hard cases particulary well
- Open source when done
- Stay tuned!

3 years later...

- We're getting there
- Announced in Glasgow'25 in May
- ...the we found some new problematic cases
- that took another year to fix
- Written for modern CPU and will have GPU acceleration (for hard cases)
- Results are *very* good, with excellent parallel scaling
- Handle hard cases particulary well
- Open source when done
- Stay tuned!

3 years later...

- We're getting there
- Announced in Glasgow'25 in May
- ...the we found some new problematic cases
- that took another year to fix
- Written for modern CPU and will have GPU acceleration (for hard cases)
- Results are *very* good, with excellent parallel scaling
- Handle hard cases particulary well
- Open source when done
- Stay tuned!

3 years later...

- We're getting there
- Announced in Glasgow'25 in May
- ...the we found some new problematic cases
- that took another year to fix
- Written for modern CPU and will have GPU acceleration (for hard cases)
- Results are *very* good, with excellent parallel scaling
- Handle hard cases particulary well
- Open source when done
- Stay tuned!

3 years later...

- We're getting there
- Announced in Glasgow'25 in May
- ...the we found some new problematic cases
- that took another year to fix
- Written for modern CPU and will have GPU acceleration (for hard cases)
- Results are *very* good, with excellent parallel scaling
- Handle hard cases particulary well
- Open source when done
- Stay tuned!

3 years later...

- We're getting there
- Announced in Glasgow'25 in May
- ...the we found some new problematic cases
- that took another year to fix
- Written for modern CPU and will have GPU acceleration (for hard cases)
- Results are *very* good, with excellent parallel scaling
- Handle hard cases particulary well
- Open source when done
- Stay tuned!


Thank you • شكرا
جامعة الملك عبدالله
للعلوم والتقنية
King Abdullah University of
Science and Technology