ENSEMBLE-BASED METHOD FOR HAWKES-PROCESS NETWORK CONSTRUCTION FROM TIME-SERIES OF COUNT DATA

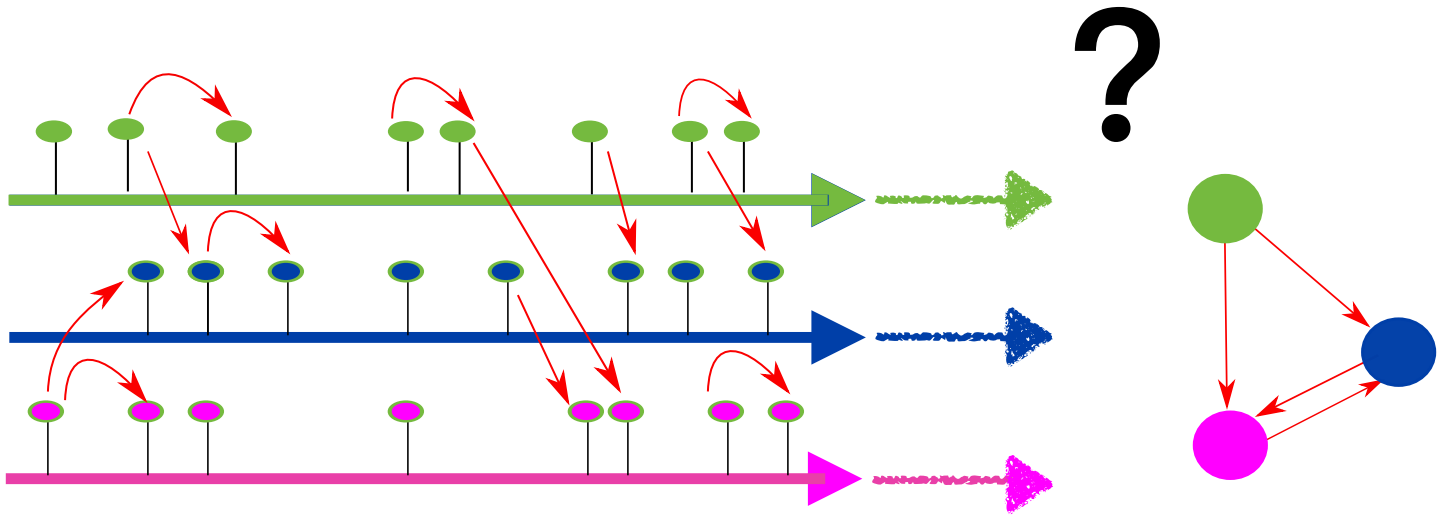
Naratip Santitissadeekorn

School of Physics and Mathematics, University of Surrey

19th International EnKF Workshop, Norway

Hawkes process for excitation/influence network

- The occurrences of an event increases the probability of the occurrences of the subsequent events
- Either through self-excitation or by influencing other nodes in the network, or both



- A Hawkes-process is a conventional model for this type of network

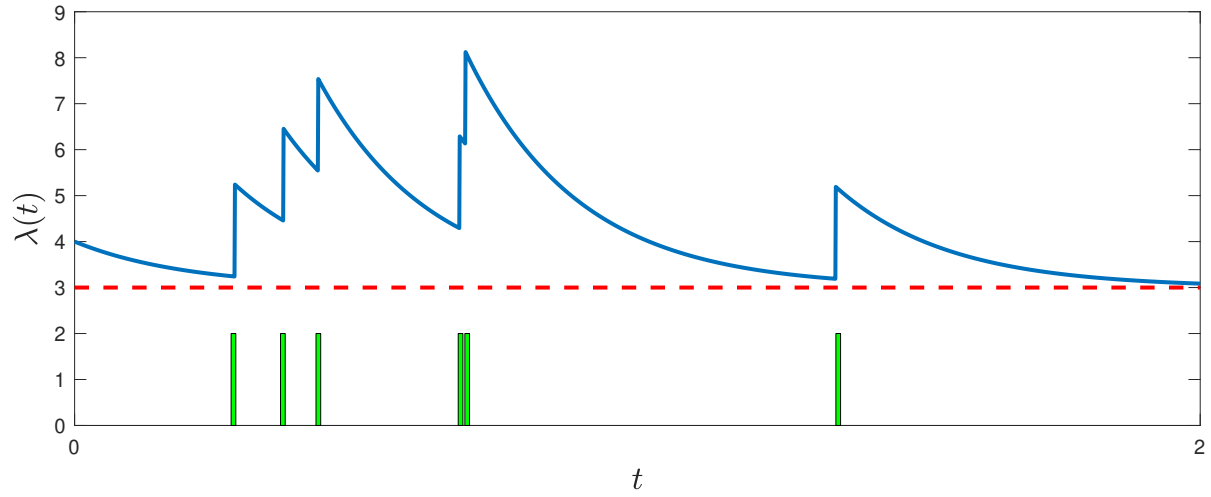
Continuous-time Hawkes process: Time-stamp data

- **Hawkes process:** Conditional intensity process

$$\lambda(t) = \mu + \sum_{j:t_j<t} g(t - t_j; \Theta) \qquad \lim_{h \to 0^+} \frac{1}{h} \mathbb{P}\left(N_{t+h} - N_t = 1 \mid \mathcal{H}_t\right) =: \lambda(t)$$

  - N_t: a counting process
  - t_j: time of event (i.e. timestamp) and $\mathcal{H}_T := \{t_j | t_j < T\}$
  - $\mu > 0$: baseline
  - $g(\tau)$: Excitation kernel parameterized by Θ

- **Poisson process:** $g(\tau) := 0$
- **Exponential decay kernel:** $g(\tau) := \alpha e^{-\beta \tau}$



- **m-dimensional Hawkes process:** binding m Hawke processes together

$$\lambda_k(t) = \mu_k + \sum_{i=1}^{m} \sum_{j:t_j^i<t} \mathcal{K}_{ik}(t - t_j^i; \Theta_{ik}), \quad k = 1, \ldots, m.$$

- **Exponential decay:** $\mathcal{K}_{ik}(\tau) = \alpha_{ik} e^{-\beta_{ik}\tau}$
- Main interest: $\alpha_{ik} =$ influence of the i-node on the k-node.

Connection with Granger causality

- There is a connection between the (multivariate) Hawkes process (or influence) network and the Granger's causality: if the process j "Granger-causes" the process i, then the past events of the process j should contain information that helps to predict the events of the process i beyond the information contained in the past event of the process i alone.
- Node i does not "Granger-cause" Node j if and only if $\alpha_{ij} = 0$ [Eichler et al. 2016]

Maximum likelihood estimate (MLE)

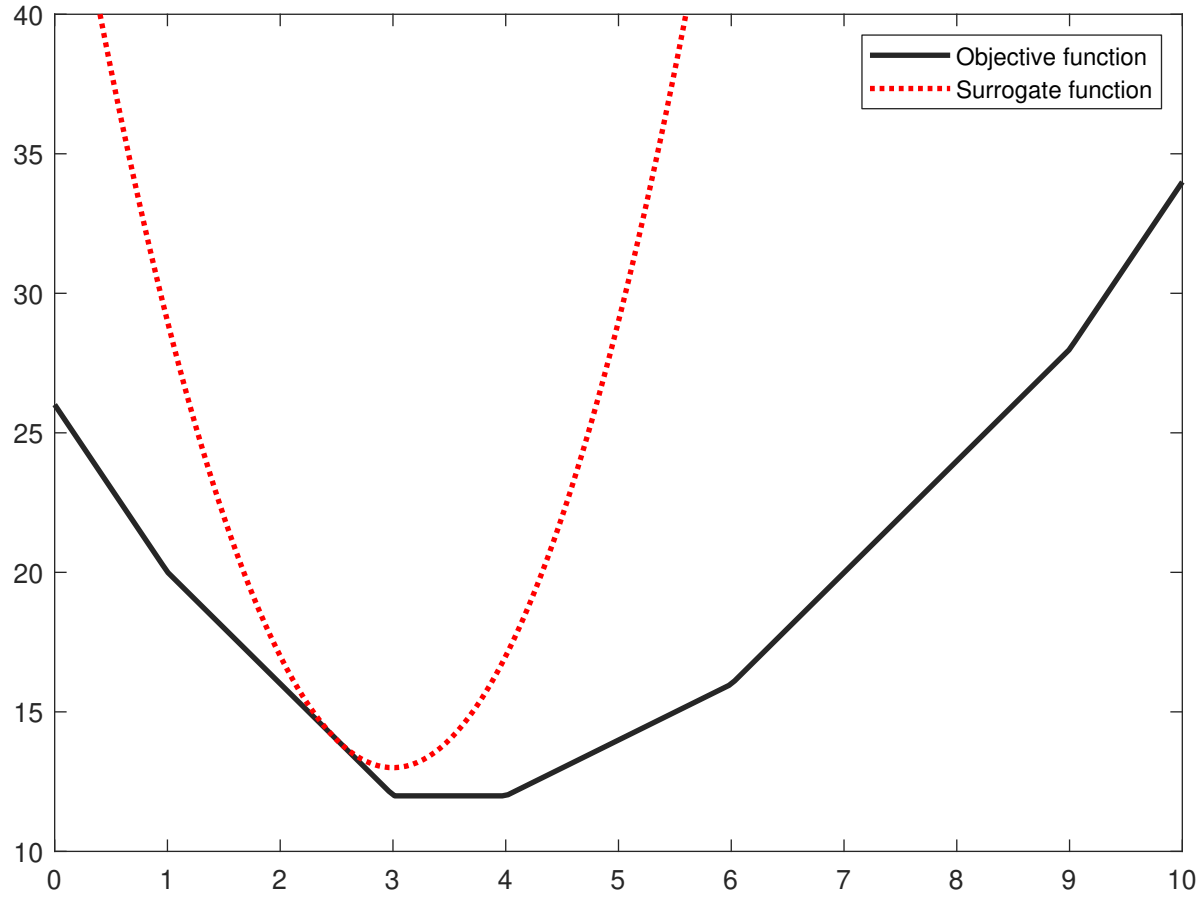
- But evaluating the likelihood can be expensive (for a large batch of data and large m)

$$\mathcal{L}(\Theta) = -\sum_{i=1}^{N} \log \lambda_{s_i}(t_i) + \sum_{k=1}^{m} \int_0^T \lambda_k(t) dt$$

- **MM**: "Majorisation-minimisation" technique
- **Main idea:** Iteratively minimise a "tight upper bound" (surrogate) function that should be easier to solve

$$f(\mathbf{x}_n) \leq Q(\mathbf{x}_n \mid \mathbf{x}_{n-1}) \leq Q(\mathbf{x}_{n-1} \mid \mathbf{x}_{n-1}) = f(\mathbf{x}_{n-1})$$

- **Expectation Maximization (EM)**: usually employed the branching process


Objective function
Surrogate function
40
35
30
25
20
15
10
0
1
2
3
4
5
6
7
8
9
10

Hawkes process driven by count data

- How do we deal with a time-series of count data?

  ΔN_k^i : no. of events in an interval $(\tau_{k-1}, \tau_k]$ of the process i

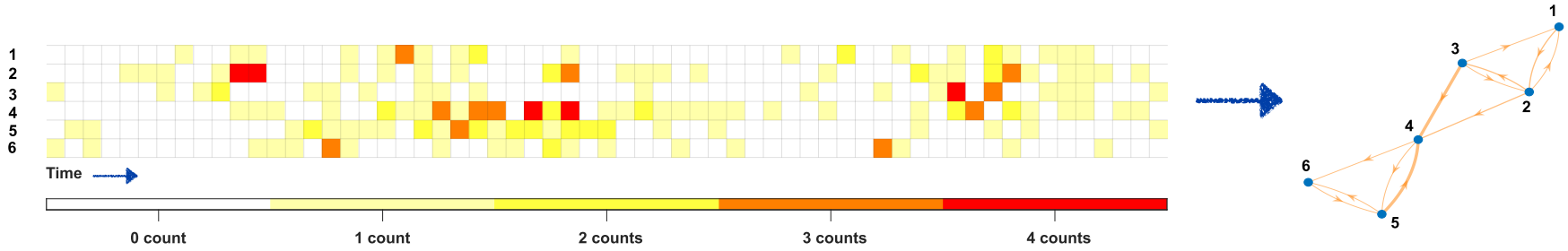
- Use the discrete version of Hawkes: λ_k^i is constant in $(\tau_{k-1}, \tau_k]$
- e.g. **Exponential decay**

$$\lambda_{k+1}^i = \mu^i + (\lambda_k^i - \mu^i)(1 - \beta^i \delta t) + \sum_{j=1}^{m} \alpha_{ij} \Delta N_k^j$$

$$\lambda_{k+1}^i = \mu^i + \sum_{j=1}^{m} \sum_{l=1}^{k-1} \left(\beta^i\right)^{k-l-1} \alpha_{ij} \Delta N_k^j$$

- As $(\tau_{k-1}, \tau_k] =: \delta \to 0$, the equilibrium mean and variance is the same as the continuous version

- Learning the (weighted) network α_{ij} from count data
- α_{ij} has the same interpretation as the time-stamp data



- Aims: Develop a scalable approach for the (approximate) inference of an influence network.

MM/EM framework

- A very generic form of EM for state-space modelling is well-known.
- Maximize (marginal) likelihood function

$$\widehat{\theta} := \arg\max_{\theta\in\Theta} \log \int p\left(\mathbf{x}_{0:K}, \Delta N_{1:K} \mid \theta\right) d\mathbf{x}_{0:K}.$$

- **E-step:** Set up a tight lower-bound (or surrogate) function for maximization

$$\begin{aligned}\mathcal{Q}\left(\theta;\theta^{(\kappa)}\right) &= \int p\left(\mathbf{x}_{0:K} \mid \Delta N_{1:K}, \theta^{(\kappa)}\right) \log p\left(\mathbf{x}_{0:K}, \Delta N_{1:K} \mid \theta\right) d\mathbf{x}_{0:K} \\ &= \mathbb{E}\left[\log p\left(\mathbf{x}_{0:K}, \Delta N_{1:K} \mid \theta\right)\right].\end{aligned}$$

  which represents the E-step of the EM algorithm.

- **M-step:** Solve the maximization problem

$$\theta^{(\kappa+1)} := \arg\max_{\theta\in\Theta} \mathcal{Q}\left(\theta;\theta^{(\kappa)}\right).$$

- Under the (first-order) Markovian assumption, we can decompose the surrogate function $\mathcal{Q}\left(\theta, \theta^{(\kappa)}\right)$ by

$$\mathcal{Q}\left(\theta, \theta^{(\kappa)}\right) = Q_0\left(\theta, \theta^{(\kappa)}\right) + Q_x\left(\theta, \theta^{(\kappa)}\right) + Q_{\Delta N}\left(\theta, \theta^{(\kappa)}\right),$$

$$Q_0\left(\theta, \theta^{(\kappa)}\right) = \mathbb{E}\left[\log p\left(\mathbf{x}_0 \mid \theta\right)\right],$$

$$Q_{\mathbf{x}}\left(\theta, \theta^{(\kappa)}\right) = \sum_{k=1}^{K} \mathbb{E}\left[\log p\left(\mathbf{x}_k \mid \mathbf{x}_{k-1}, \Delta N_{1:K}, \theta\right)\right],$$

$$Q_{\Delta N}\left(\theta, \theta^{(\kappa)}\right) = \sum_{k=1}^{K} \mathbb{E}\left[\log p\left(\Delta N_k \mid \mathbf{x}_k, \theta\right)\right].$$

- If we can sample from $p\left(\mathbf{x}_{0:K} \mid \Delta N_{1:K}, \theta^{(\kappa)}\right)$, we can then estimate all the expectations using the sample paths
- It can be computationally infeasible for a large-scale problem

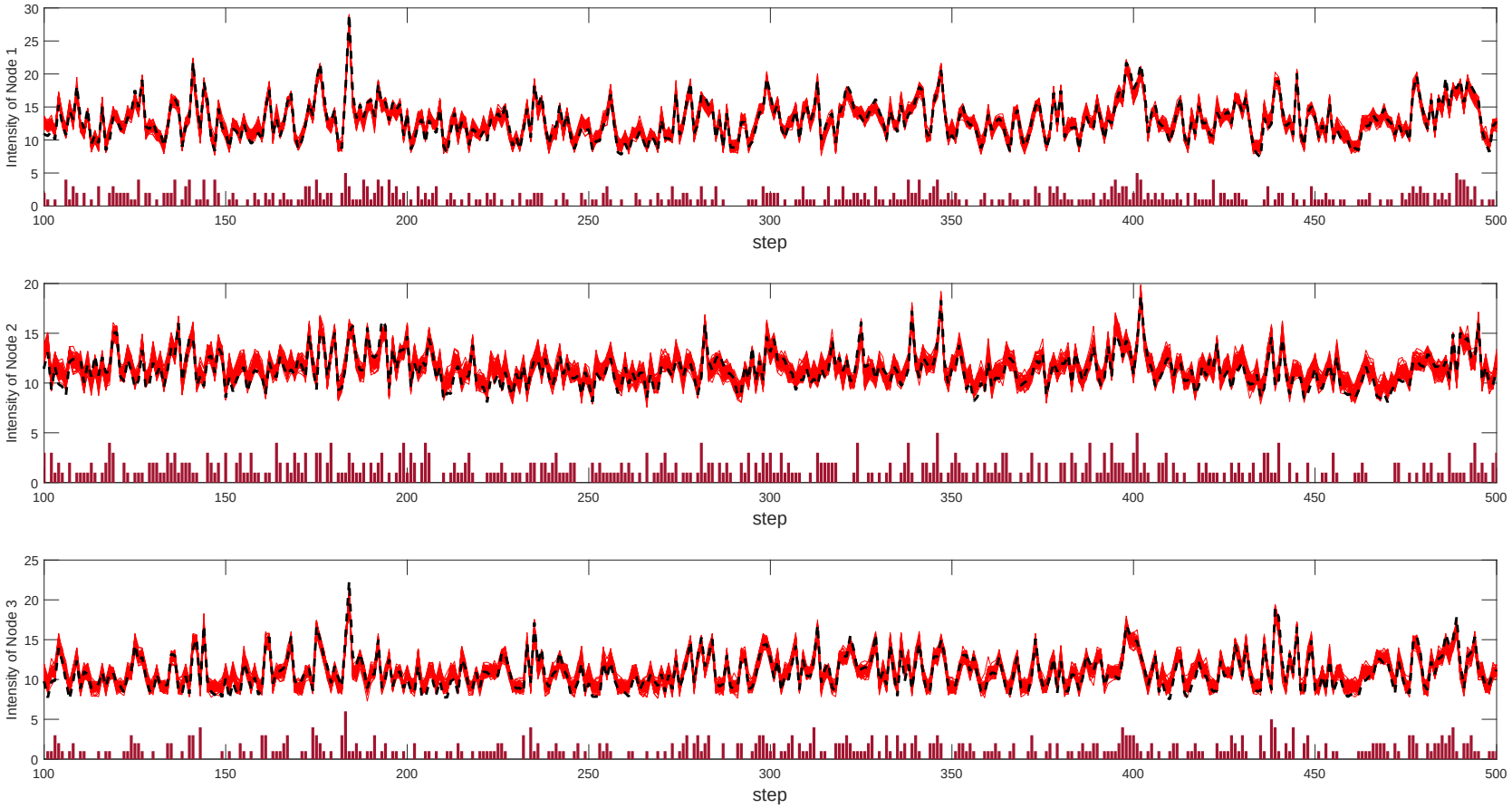
- Example: Log-Gaussian Cox process (LGCP) on a small network

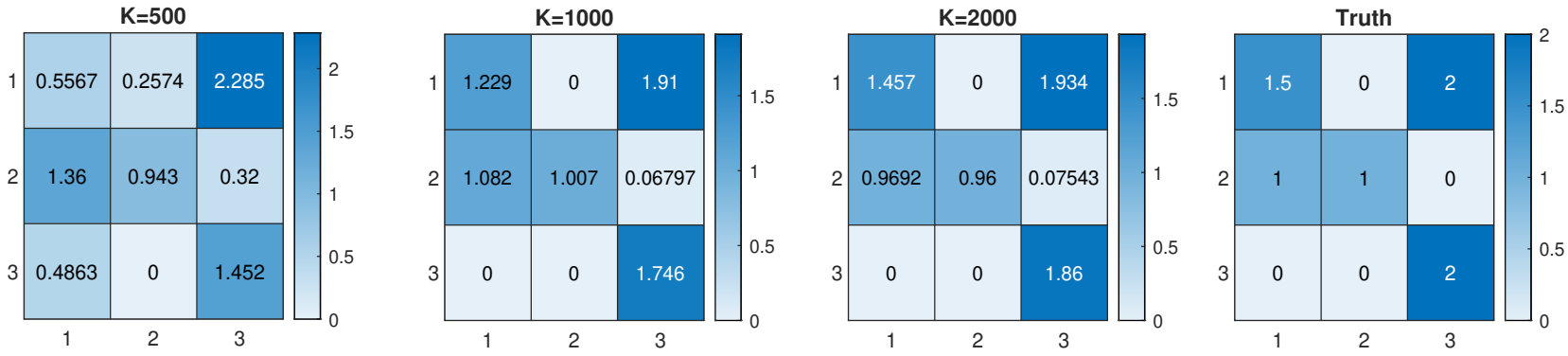
$$x^i_{k+1} = \left[(1-\eta^i)x^i_k + \eta^i \sum_{j \neq i} x^j\right](1-\omega^i_1 \delta t) + \omega^i_1 \mu^j \delta t + \epsilon^i \sqrt{\delta t}\zeta_k,$$

$$g^i_{k+1} = (1-\omega^i_2 \delta t) g^i_k + \sum_{j=1}^{m} \alpha^{ij} \Delta N^j_k,$$

$$\lambda^i_{k+1} = \exp(x^i_{k+1}) + g^i_{k+1},$$

- **E-Step:** Run forward-filter backwards-smoother on x and g; hence λ.
- Bootstrap particle filter with Nakano's resampling scheme
- Backward simulation smoother (BSS)
- **M-step:** Parameter optimisation with constraints


Intensity of Node 1
30
25
20
15
10
5
0
100
150
200
250
300
350
400
450
500
step
Intensity of Node 2
20
15
10
5
0
100
150
200
250
300
350
400
450
500
step
Intensity of Node 3
25
20
15
10
5
0
100
150
200
250
300
350
400
450
500
step



- The algorithm converges to the true network given a long enough sequence of data
- The MM algorithm can be derived for the exponential decay model[NS,DL,MS: in preparation], no need to sample $p\left(\mathbf{x}_{0:K} \mid \Delta N_{1:K}, \theta^{(\kappa)}\right)$. A tight upper bound function (for each node) is

$$Q(\theta \mid \theta^{(n)}) = -\sum_{k=0}^{K} Q_k(\theta \mid \theta^{(n)}) \Delta N_k + N\mu + \sum_{j=1}^{m} H^j \mathcal{N}^j,$$

where

$$H^j = \frac{\left(\alpha^j\right)^{(n)}}{2\left(1 + \left(\gamma^j\right)^{(n)}\right)}(1+\gamma^j)^2 + \frac{2\left(1 + \left(\gamma^j\right)^{(n)}\right)}{\left(\alpha^j\right)^{(n)}}(\alpha^j)^2,$$

$$\mathcal{N}^j = \Delta N_1^j + \ldots + \Delta N_{K-2}^j,$$

$$Q_k(\theta \mid \theta^{(n)}) := -\frac{\mu^{(n)}}{\lambda_k^{(n)}} \log\left(\frac{\lambda_k^{(n)}}{\mu^{(n)}}\mu\right) - \sum_{l=0}^{k-1}\sum_{j=1}^{m} \frac{\phi_{klj}^{(n)}}{\lambda_k^{(n)}} \log\left(\frac{\phi_{klj}^{(n)}}{\lambda_k^{(n)}}\phi_{klj}\right),$$

where $\phi_{klj} := \alpha^j(\gamma^j)^{k-l-1}\Delta N_k^j$.

(Approximate) Filtering/Sequential Monte Carlo

- Using filtering to estimate parameters
- Allowing parameters to be "states" in a state-space model
- Recursively sampling $p(\Theta_t \mid \mathcal{H}_t)$
- The likelihood is a Poisson distribution.
- Approximate filter can be easily developed for the normal prior

- **Extended-Poisson Kalman filter (ExPKF):** second-order approximation [NS et. al. 2019]

$$\mathbf{P}_k^{-1} = \mathbf{P}_{k|k-1}^{-1} + \sum_{j=1}^{m} \left[\left(\frac{\partial \log \lambda_k^j}{\partial \Theta_k} \right) \left(\frac{\partial \log \lambda_k^j}{\partial \Theta_k} \right)^T \lambda_k^j \Delta t_k \right.$$
$$\left. - (\Delta N_k^j - \lambda_k^j \Delta t_k) \frac{\partial^2 \log \lambda_k^j}{\partial \Theta_k^2} \right]$$
$$\bar{\Theta}_k = \bar{\Theta}_{k|k-1} + \mathbf{P}_k \sum_{j=1}^{C} \left[\left(\frac{\partial \log \lambda_k^j}{\partial \Theta_k} \right) (\Delta N_k^j - \lambda_k^j \Delta t_k) \right],$$

- More Efficient with the rank-1 approximation: $\left(\frac{\partial \log \lambda_k^j}{\partial \theta_k} \right) \left(\frac{\partial \log \lambda_k^j}{\partial \theta_k} \right)^T = -\frac{\partial^2 \log \lambda_k^j}{\partial \theta_k^2}$

$$\mathbf{P}_k^{-1} = \mathbf{P}_{k|k-1}^{-1} + \sum_{j=1}^{m} h_j h_j^T \qquad h_j = \sqrt{\Delta N_k^j} \left(\frac{\partial \log \lambda_k^j}{\partial \theta_k} \right)$$

- **Ensemble-based filtering:** motivated by Craig Bishop's GIGG-EnKF (2016). The update has **two stages:**

  1. Update $\lambda_k^{i,(s)}$ for all i: consistent with Poisson-gamma conjugacy
     - Need a new mean $\langle \lambda^a \rangle$ and relative variance $P_r^a = P^a / \langle \lambda^a \rangle^2$

$$\langle \lambda^a \rangle = \langle \lambda \rangle + \frac{\langle \lambda \rangle}{P_r^{-1} + \langle \lambda \rangle \delta t} (\Delta N - \langle \lambda \rangle \delta t)$$

$$(P_r^a)^{-1} = P_r^{-1} + y^o$$

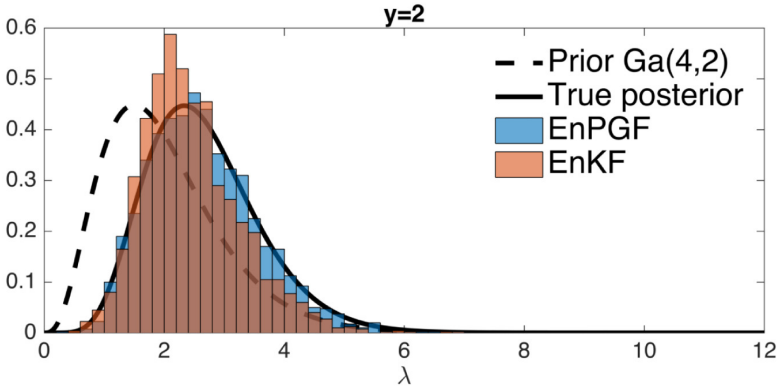
     - Move/update the ensemble of λ using a stochastic equation

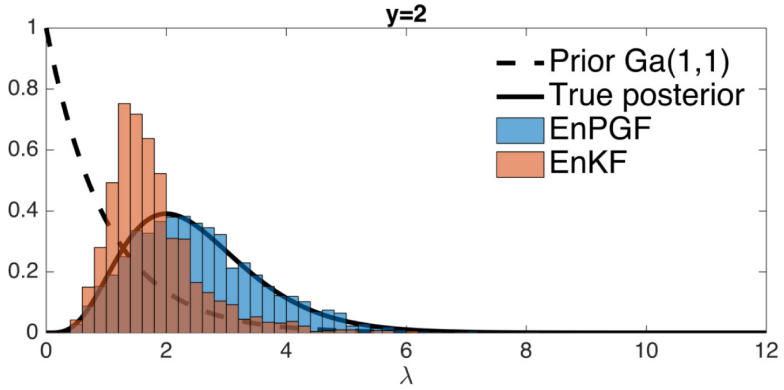
$$\frac{\lambda^{(s),a} - \bar{\lambda^a}}{\bar{\lambda^a}} = \frac{\lambda^{(s)} - \bar{\lambda}}{\bar{\lambda}} + \frac{P_r}{P_r + (\Delta N)^{-1}} \left[\frac{\Delta N_e^{(s)} - \Delta \bar{N}_e}{\Delta \bar{N}_e} - \frac{\lambda^{(s)} - \bar{\lambda}}{\bar{\lambda}} \right],$$

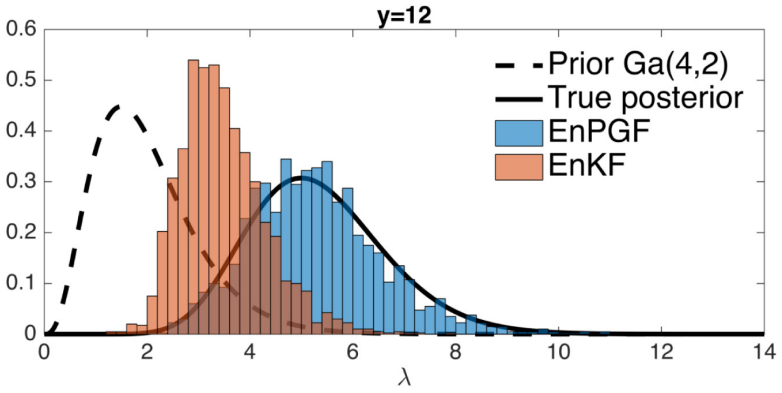
       where $\Delta N_e^{(s)}$ is independently drawn from a gamma distribution with mean ΔN and variance ΔN^2 for $i = 1 \ldots, M$ and its ensemble mean is denoted by $\Delta \bar{N}_e$. [NS,DL,MS: CSDA 2019,2022]

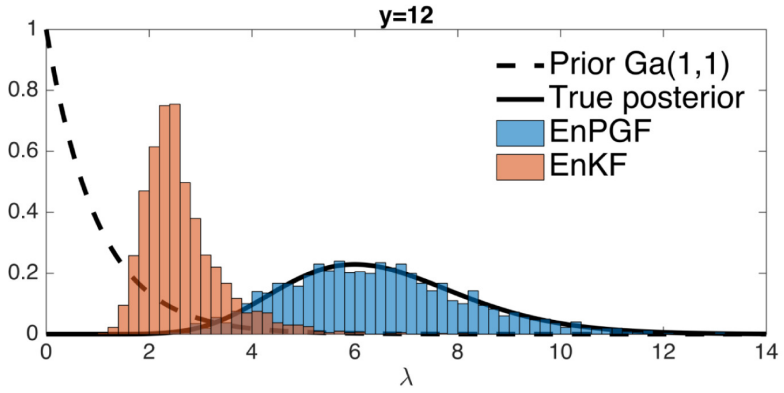
  2. Ensemble Kalman Filter (EnKF): update $\Theta_k^{j,(s)}$ taking $\lambda_k^{i,(s)}$ as "observation"

- **Checking sampling performance:**

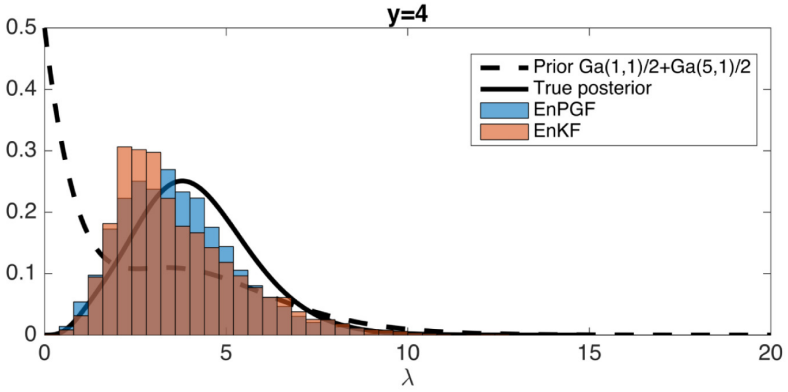


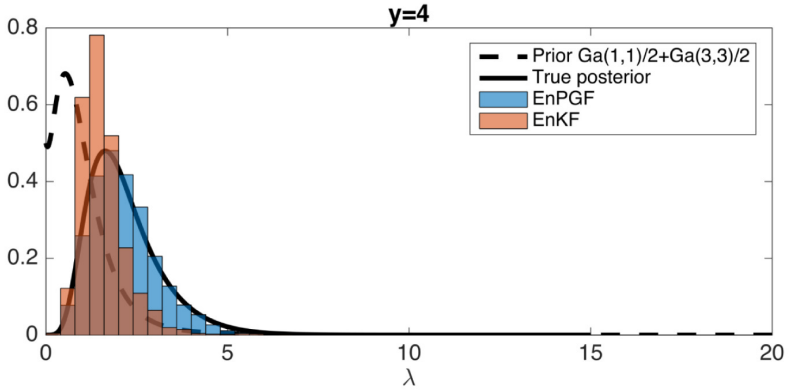


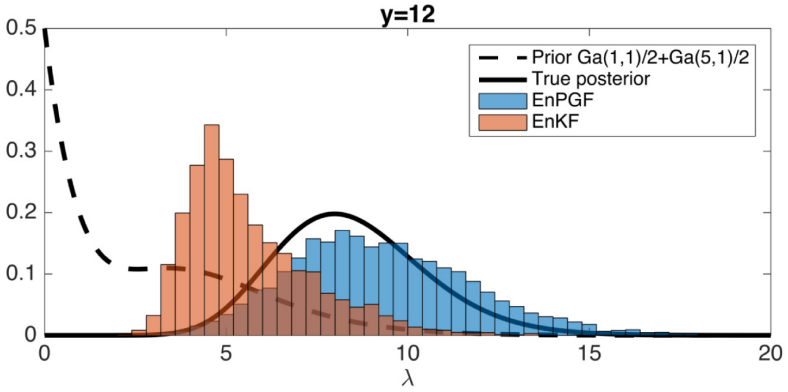


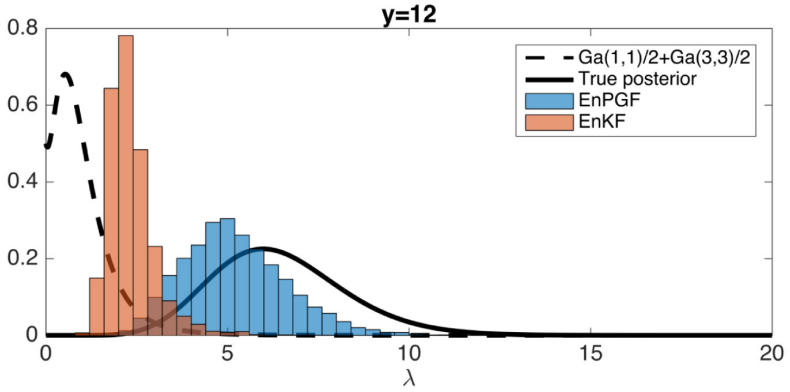


- **Checking sampling performance:**





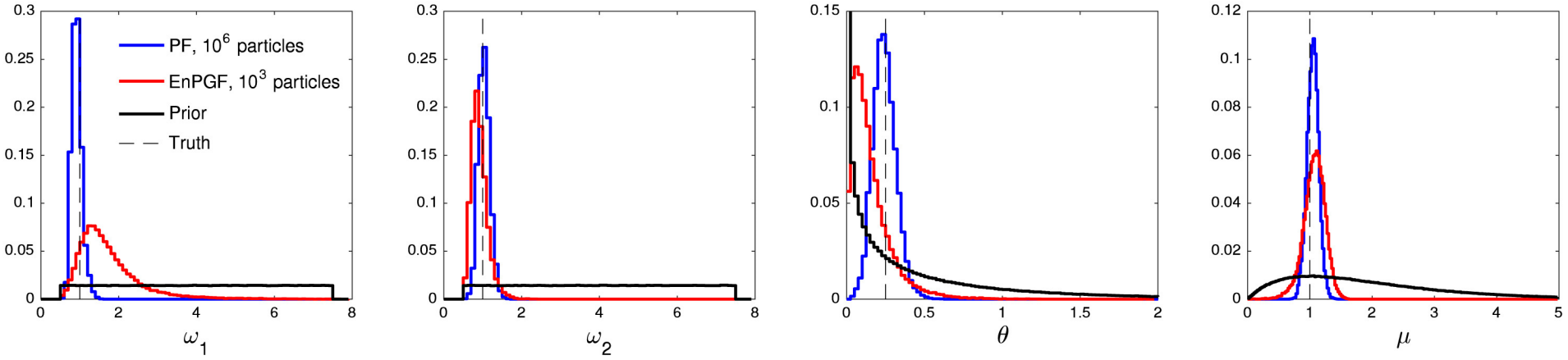




- **Experiment:** Log-Gaussian Cox process (LGCP)

$$x_{k+1} = x_k - \omega_1(x_k - \mu)\delta t + \sigma\sqrt{\delta t}Z_k \qquad Z_k \sim N(0,1)$$
$$\lambda_{k+1} = \exp(x_{k+1}) + (1 - \omega_2\delta t)(\lambda_k - \exp(x_k)) + \theta y_k,$$

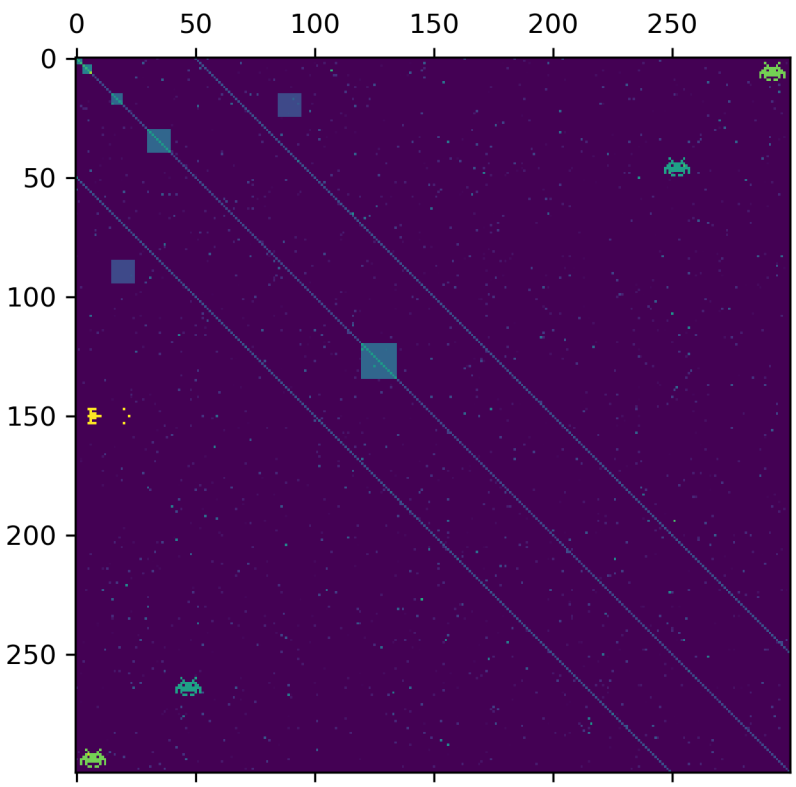
- $\Theta_k = [\mu, \omega_1, \omega_2, \theta]$
- Difficult to estimate/track σ (same issue as EnKF?)
- Metropolis Adjusted Langevin algorithm (MALA) was used for this problem in [Mohler 14]

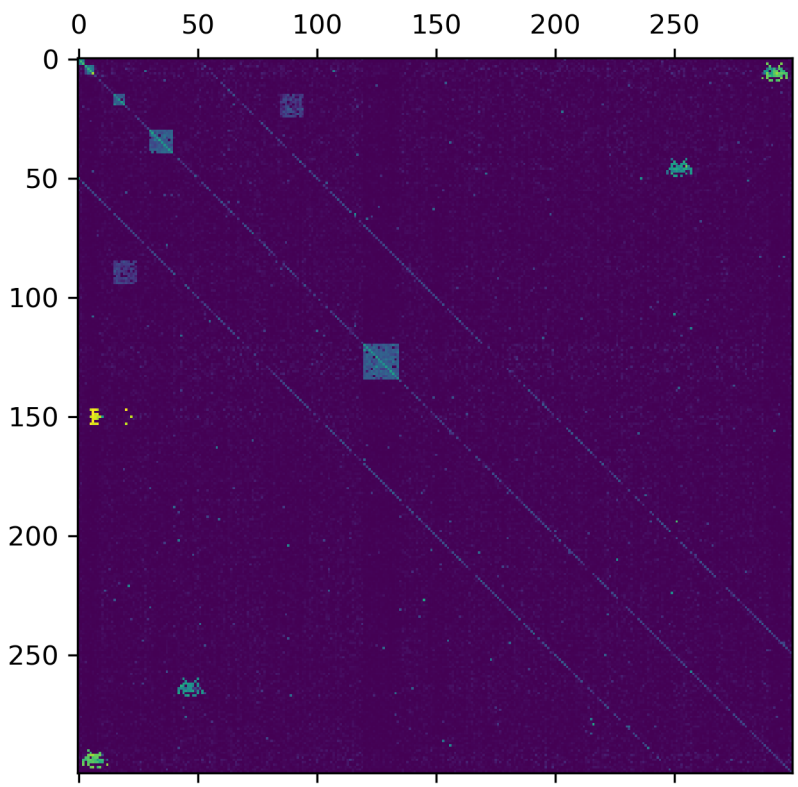


- **Network detection:** Perfect model test
- Network of multiple Hawkes processes connected through mutual excitation
- Discrete-time dynamic model

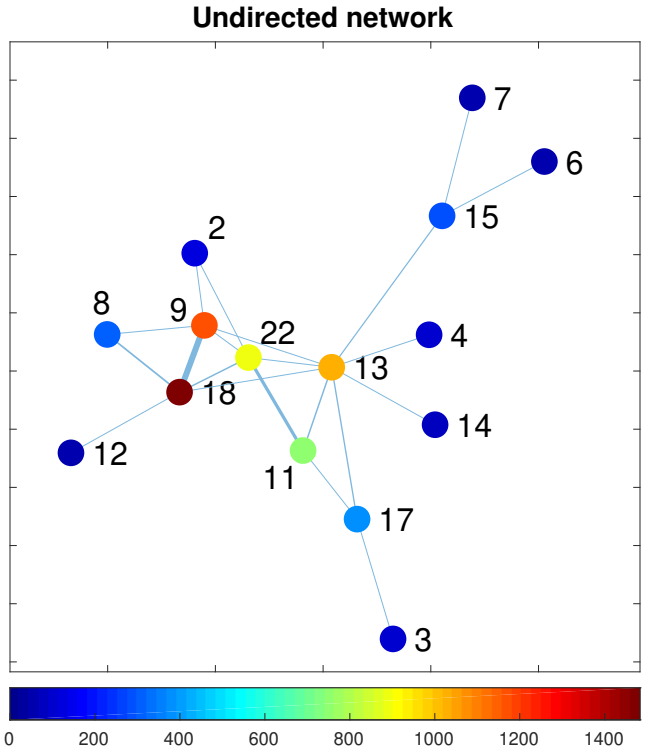
$$\lambda_{k+1}^{j} = \mu^{j} + (\lambda_{k}^{j} - \mu^{j})(1 - \beta^{j}\delta t) + \sum_{i=1}^{m} \alpha_{ij} \Delta N_{k}^{j}.$$

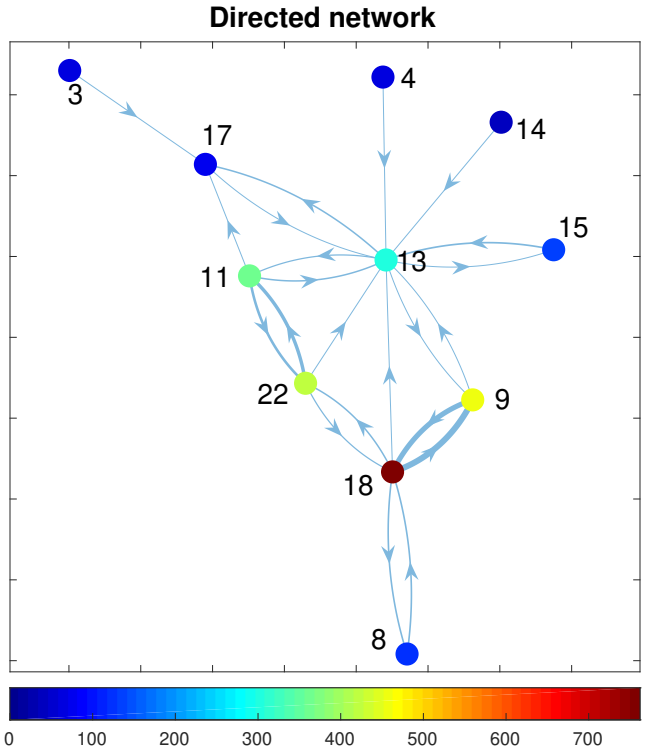
- "Large" network of 300 nodes
- Assume no prior knowledge of underlying structure; hence estimating 300^2 links!
- Apply EnPGF


0
50
100
150
200
250
0
50
100
150
200
250

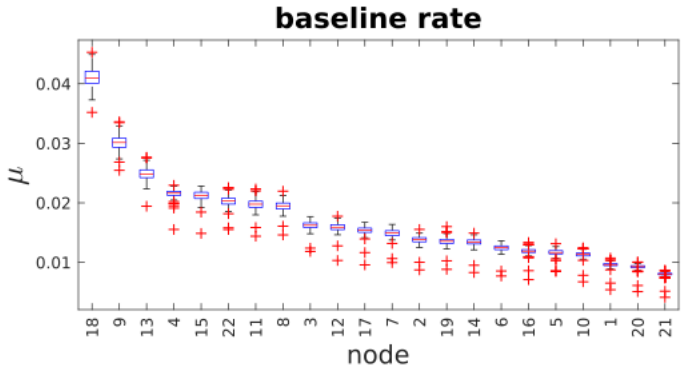

0
50
100
150
200
250
0
50
100
150
200
250

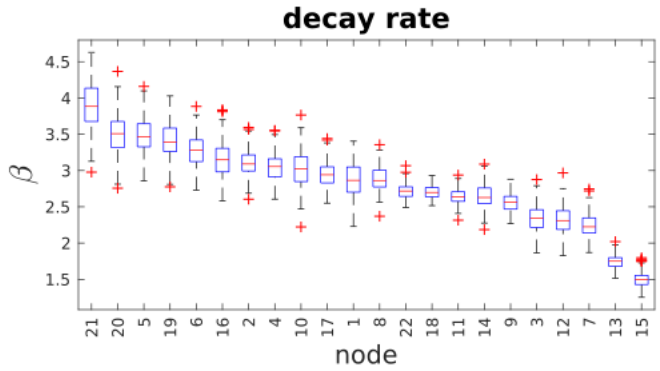
- **Real-world Email data:**
- Email communicated by 22 anonymous volunteers May 2010 to June 2011 (7988 emails in total)

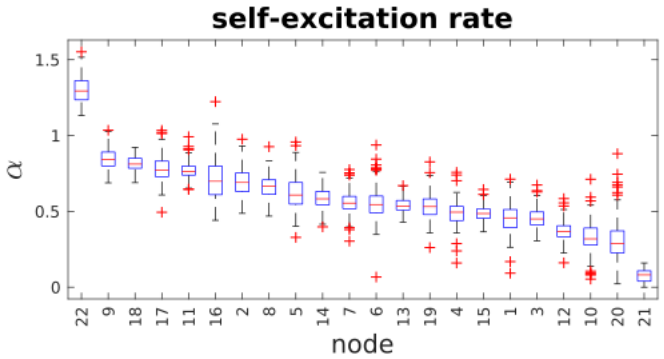


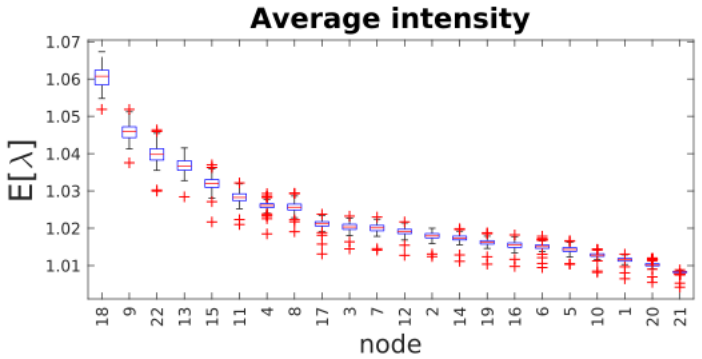


- Test data: use only the number of sending emails per 1 minute
- Results: uncertainty for parameters

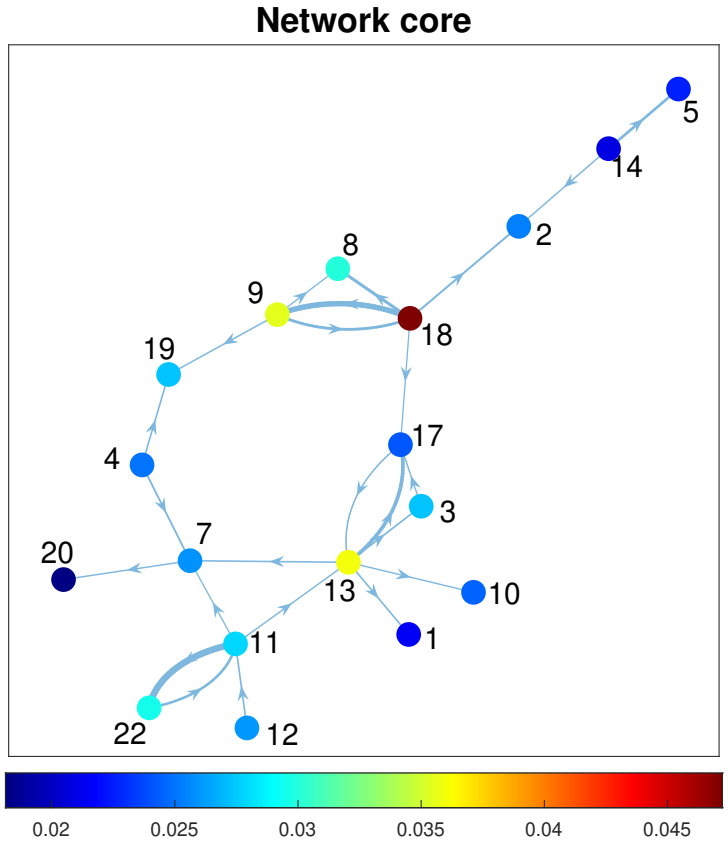




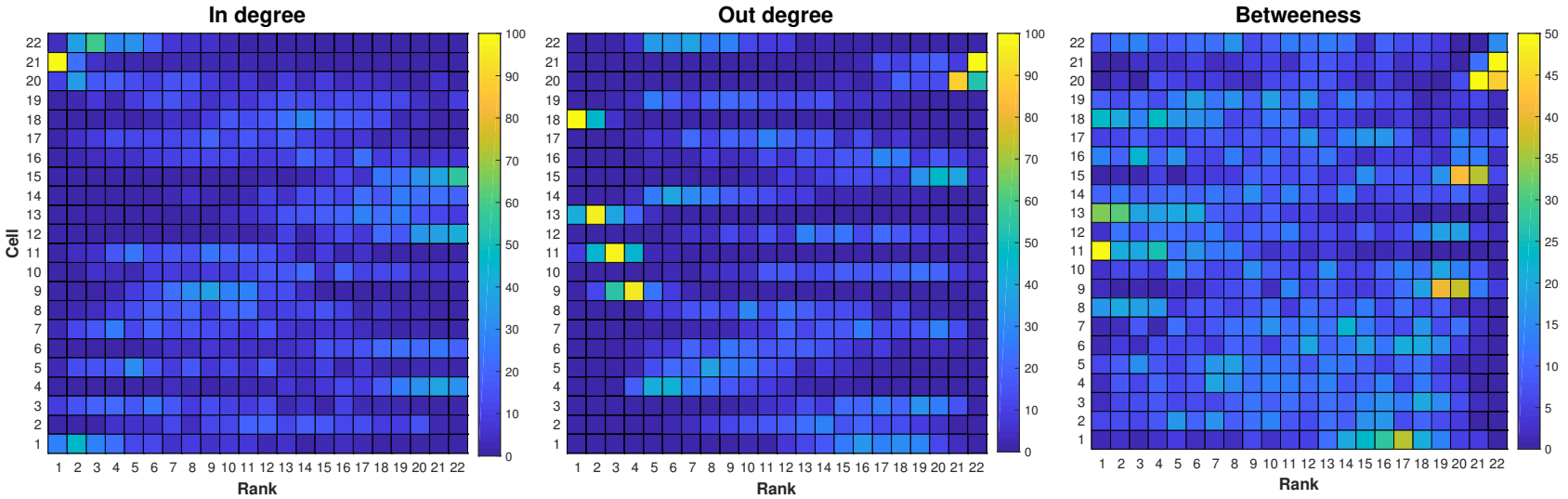




• Ensemble mean of the network



- Uncertainty of ranking



Conclusion

- **Batch DA:**
- EM is powerful but require sampling from smoother distribution
- MM algorithm can be developed for the exponential decay kernel and does not require a smoothed path.
- **Sequential DA:**
- ExPKF requires Hessian and probably the rank-one approximation for efficiency
- EnPGF is more “convenient” (no Hessian required)