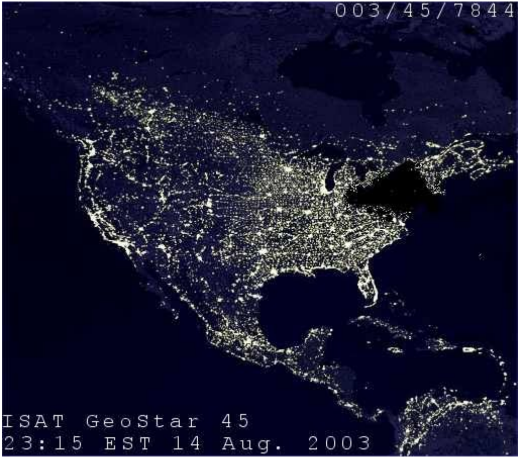
Toward more faithful simulations of cascade failures in power systems

Mihai Anitescu, Chris DeMarco, Charlie Matthews, Bradly Stadie, Jonathan Weare

University of Chicago

August 30, 2016


003/45/7844
ISAT GeoStar 45
23:15 EST 14 Aug. 2003

How do cascade failures occur, how often should we expect them, can we avoid them?

e.g.

1. Can we identify precursor events and use them to predict the occurrence of a cascade failure?

2. By understanding the frequency of cascade failures as a function of environmental parameters, can we design more robust energy grids?

Previous studies of cascade failure involve, e.g.

assigning failure rates to lines based on power across line under steady state conditions which are recomputed after each failure.

or assuming that once a line fails, nearby lines have a specified probability to fail as well.

Previous studies of cascade failure involve, e.g.

assigning failure rates to lines based on power across line under steady state conditions which are recomputed after each failure.

or assuming that once a line fails, nearby lines have a specified probability to fail as well.

Rather than modeling line failures directly, we want to model the dynamics of the power system and study how failures occur within the model.

Our current model problem has $N = 145$ nodes and $M = 453$ lines.

The voltage on each node is

$$v_i(t) = \Re\left\{m_i(t)\, e^{\mathbf{i}\,(\omega_0 t + \theta(t))}\right\}$$

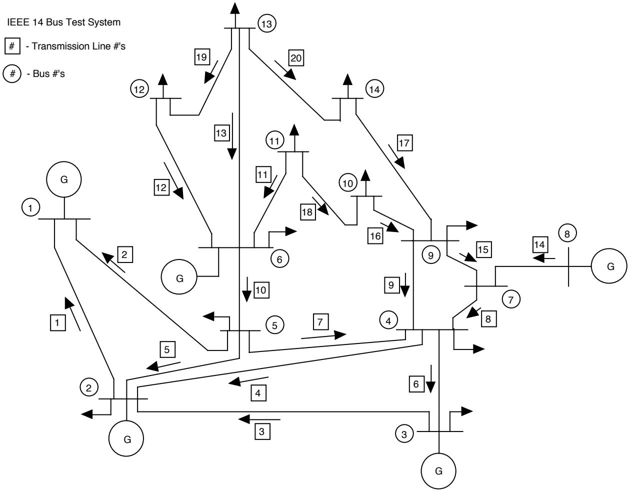
where ω_0 is a prescribed frequency.

We model the evolution of m_i, θ, and $\omega = \dot{\theta}$ which evolve on a much slower time scale than v_i.

Three types of nodes:

1. Load buses (voltage can vary freely)
2. Generator buses (voltage magnitudes fixed)
3. Slack bus (voltage fixed)

A smaller ($N = 14$ nodes, $M = 20$ lines) example:



The starting point for the model is the power flow equations:

$$P_i = \sum_{k=1}^{N} m_i m_k \,(G_{ik}\cos(\theta_i - \theta_k) + B_{ik}\sin(\theta_i - \theta_k))$$

$$Q_i = \sum_{k=1}^{N} m_i m_k \,(G_{ik}\sin(\theta_i - \theta_k) - B_{ik}\cos(\theta_i - \theta_k))$$

where P_i and Q_i are the real and reactive power at node i which we take to be fixed.

G is the conductance matrix. In our model we assume $G = 0$.

The susceptance matrix B specifies the topology of the network.

One can find steady state values for the m_i and θ_i by solving the power flow equations.

But we're interested in the evolution of m_i and θ_i.

Our dynamical model is of the form:

$$\begin{bmatrix} \dot{\omega}(t) \\ \dot{\theta}(t) \\ \dot{m}(t) \end{bmatrix} = (J - \Gamma)\, \nabla\phi(\omega(t), \theta(t), m(t)) + \tau\, \Sigma\, \eta(t)$$

where J an antisymmetric matrix, Γ a SPSD matrix, and

$$\phi(\omega, \theta, m) = \frac{1}{2}\|w\|^2 + \frac{1}{2}V^{\mathsf{H}}BV - P\cdot\theta - Q\cdot\log(m)$$

with $V_j = m_j\, e^{\mathbf{i}\,\theta_j}$.

$\tau > 0$ is small, $\eta \in \mathbb{R}^{3N}$ is white noise, and $\Sigma \in \mathbb{R}^{3N\times 3N}$.

$\tau\,\Sigma\,\eta$ represents small perturbations in demand and generator output.

When $\tau = 0$, if we initialize the process near a solution to the power flow equations, the process will converge to that solution.

If $\tau = 0$ and $\Gamma = 0$, then the value of ϕ is a constant of integration.

If $\Sigma = \sqrt{2\Gamma}$ then the invariant probability density for the process is

$$\pi(\omega, \theta, m) \propto e^{-\varphi(\omega,\theta,m)/\tau}$$

When $\tau = 0$, if we initialize the process near a solution to the power flow equations, the process will converge to that solution.

If $\tau = 0$ and $\Gamma = 0$, then the value of ϕ is a constant of integration.

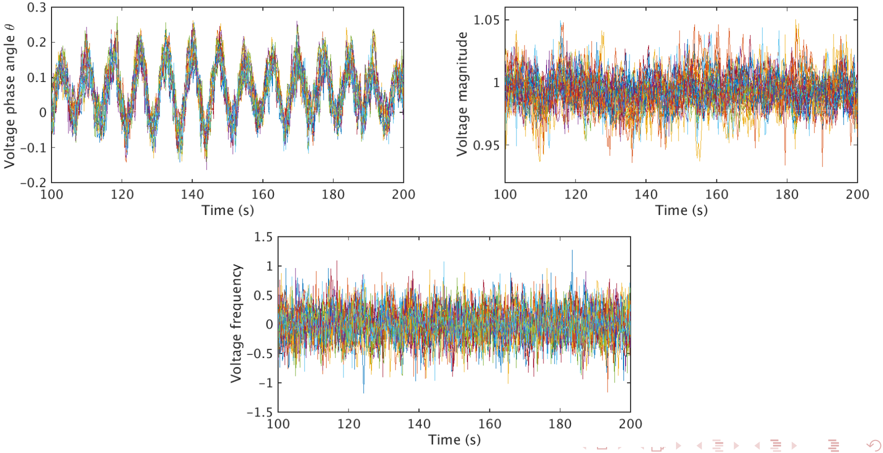
If $\Sigma = \sqrt{2\Gamma}$ then the invariant probability density for the process is

$$\pi(\omega, \theta, m) \propto e^{-\varphi(\omega,\theta,m)/\tau}$$

When the noise is small we expect the system to oscillate around a solution to the power flow equations.

The stochastic differential equation is a version of the so-called underdamped Langevin equation appearing in molecular simulation.

We borrow ideas from molecular simulation to derive efficient numerical discretization schemes for these equations (timestep $=$ 10ms).



What about line failures?

The network topology is encoded in the node-to-branch matrix $A \in \mathbb{R}^{N \times M}$ with entries

$$A_{ij} = \begin{cases} -1 & \text{if line } i \text{ enters node } j \\ 1 & \text{if line } i \text{ exits node } j \\ 0 & \text{otherwise} \end{cases}$$

and enters the equations through

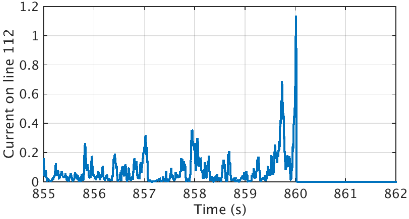
$$B = A^{\mathsf{T}} \text{diag}(s_1, s_2, \ldots, s_M) A$$

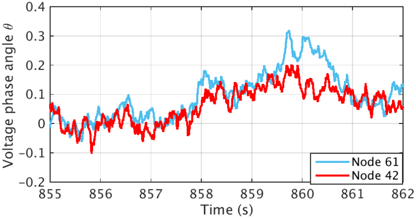
$s_k \geq 0$ is the susceptance value of line k.

The topology of the network evolves when the power across the line

$$s_k|V_{k_1} - V_{k_2}|^2$$

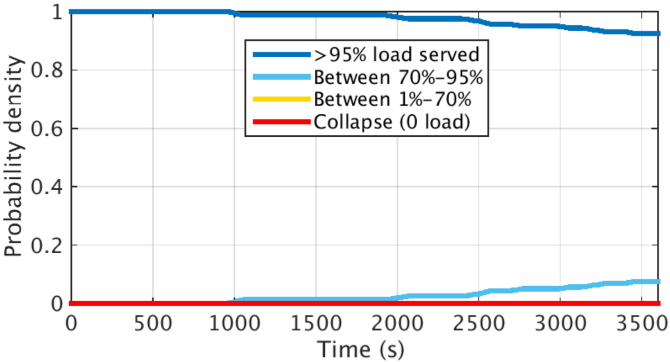
exceeds a fixed threshold and the line is dropped (s_k is set to zero).





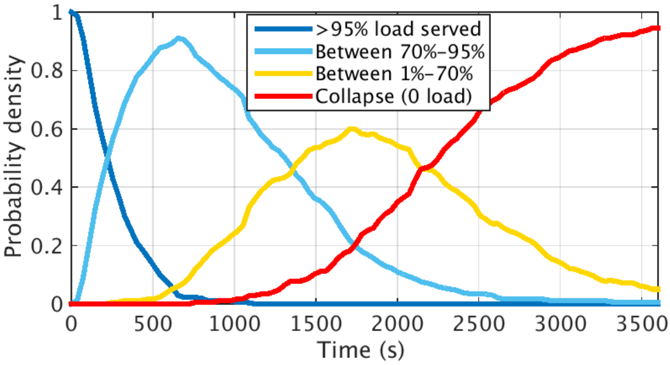
In this simulation, the line power across line 112 (connecting nodes 61 and 42) exceeds the threshold at 860s and the line is dropped.

Now that we have reasonable grid model we can simulate many trajectories of the network and gather failure statistics.



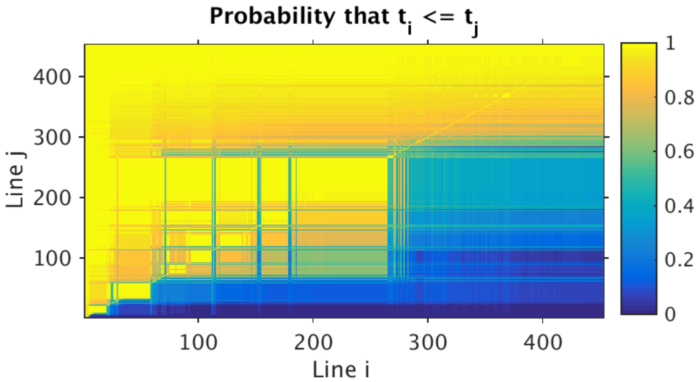
$$\text{load served} = \frac{\sum_{\substack{\text{nodes connected}\\ \text{to slack node}}} P_i}{\sum_{\text{all nodes}} P_i}$$

We can also probe for vulnerabilities in the network by weakening it and gathering failure statistics

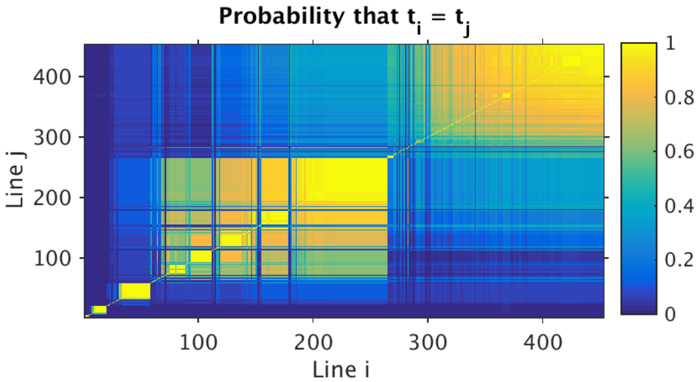


Here we've lowered the line relay threshold.

We can try to understand the mechanism by which cascade failures occur (e.g. which lines tend to fail early relative to others).



We can try to understand the mechanism by which cascade failures occur (e.g. which lines tend to fail early relative to others).



But if our model is any good, cascade failures will be very rare.

For a large network we won't be able to directly simulate enough cascade failure events to gather reliable statistics.

How do we study cascade failure at reasonable cost?

Large Deviations suggests that, when the noise is small, the stochastic forcing generating a rare event is predictable:

$$\hat{\eta} = \arg\min_{\eta \in \mathcal{A}} \int_0^T \frac{1}{2} \|\eta(s)\|^2 ds$$

where $\mathcal{A}$ (roughly) includes realizations of the stochastic forcing compatible with the event (e.g. forcings leading to a large number of failures in a short time).

Moreover, given $\hat{\eta}$ we can compute

$$P(\text{event}) = e^{-\frac{\int_0^T \frac{1}{2} \|\hat{\eta}(s)\|^2 ds + o(1)}{\tau}}$$

(One can use sampling schemes such as importance sampling to get the $o(1)$ term)

Solving the optimization problem for 1 line failure is relatively easy.

When constraining for multiple line failures many sequences of line failures become local minima.

There are so many local minima that failure probabilities estimated by LD using only one path will be very inaccurate.

Solving the optimization problem for 1 line failure is relatively easy.

When constraining for multiple line failures many sequences of line failures become local minima.

There are so many local minima that failure probabilities estimated by LD using only one path will be very inaccurate.

We have to find a way to sample the failure trajectories more efficiently.... **We need rare event simulation.**

A particularly simple method can be applied if the system locally equilibrates between failures.

If B_0 is the initial susceptibility matrix, then

$$\nu(A) = \lim_{T \to \infty} P((\omega(t), \theta(t), m(t)) \in A, \mid B(t) = B_0)$$

is the **quasi-stationary** distribution for the network topology specified by B_0.

If, between line failures, the quasi-stationary distribution is reached, then

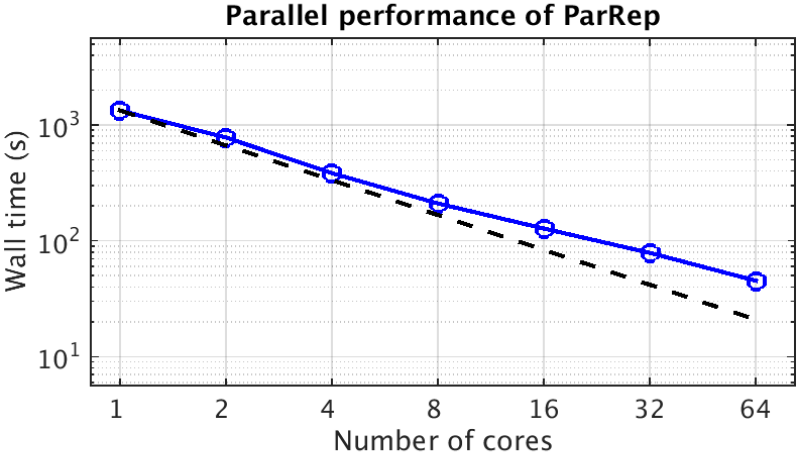
1. The time at which the next failure occurs is an exponential random variable.
2. That time is independent of the values of ω, θ, and m, at the exit time.

Voter's Parallel-Replica Dynamics scheme used in atomistic simulations of solids takes advantage of this assumption:

If $\{t_i\}^n$ are a set of independent exponential random variables with rate λ, then $\min t_i$ is exponential with rate $n\lambda$.

So (very roughly):

1. Check for quasi-stationarity.
2. Run n parallel copies of the system in parallel with the same network topology until first network transition event
3. Scale time by n and move to next topology.



Parallel performance of ParRep
Wall time (s)
10^3
10^2
10^1
1
2
4
8
16
32
64
Number of cores