

# On the role of wind covariance estimation in power grid dispatch

**Cosmin G. Petra**

Mathematics and Computer Science Division

Argonne National Laboratory (ANL)

[petra@mcs.anl.gov](mailto:petra@mcs.anl.gov)

**SIAM Conference on Computational Science and  
Engineering**

**Salt Lake City, UT**

**March 17, 2014**

# Outline

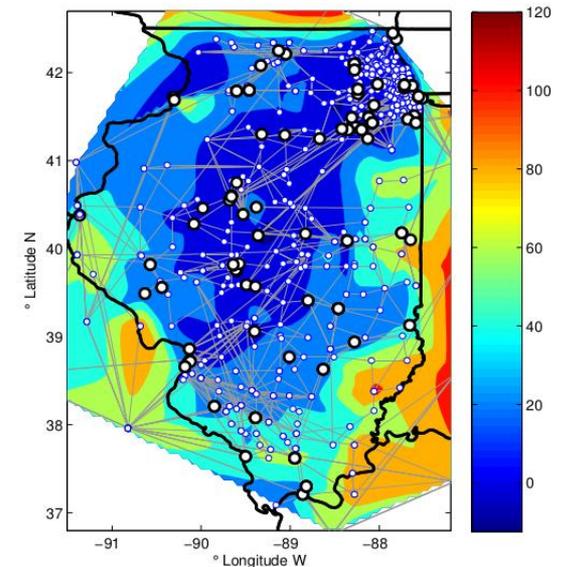
- Economic dispatch models in power grid
- Parallel solver for stochastic optimization – PIPS-IPM
- Modeling framework - StochJuMP
- Covariance estimation and impact on optimal
  - Motivation
  - State of Illinois' power grid simulation



# Economic dispatch models

- Basis for the electricity distribution and electricity market
- Used by all Independent System Operators (ISOs) in the US.
- In the simpler form, for direct currents, is formulated as a linear programming problem

$$\begin{aligned} \min_{x, f} \quad & \sum_{i \in G} c_i x_i \\ \text{subj.to:} \quad & \tau_n(f) + \sum_{i \in T(n)} x_i = d_n, \forall n \in N \\ & f \in U \\ & x_i \in C_i, \forall i \in G \end{aligned}$$



- For alternating currents (AC), it takes the form of power flow, a nonlinear programming problem



# Stochastic dispatch models

- Adoption of highly volatile renewable energy and randomness in demand requires stochastic formulations
- Cost-optimal decision in the presence of uncertain generation/demand
- Two-stage linear stochastic programming with recourse: “energy only” model (Pritchard, Zakeri, Philpott, 2010)

$$\begin{aligned} \min_{x, X(\omega), f, F(\omega)} \quad & \sum_{i \in G} c_i x_i + \mathbb{E}_\omega \sum_{i \in G} p_i |x_i - y_i(\omega)| \\ \text{subj.to:} \quad & \tau_n(f) + \sum_{i \in T(n)} x_i = d_n, \forall n \in N \\ & \tau_n(F(\omega)) + \sum_{i \in T(n)} y_i(\omega) = d_n, \forall n \in N, \omega \in \Omega \\ & f, F(\omega) \in U, \forall \omega \in \Omega \\ & x_i, y_i(\omega) \in U_i, \forall i \in G, \omega \in \Omega \end{aligned}$$



- Two-markets: “ahead”, decisions/prices to be taken-now and “realtime”, scenario-specific adjustments in decisions/prices.
- The model is ISO revenue adequate (no “missing money”).

# PIPS-IPM - parallel solver for stochastic optimization



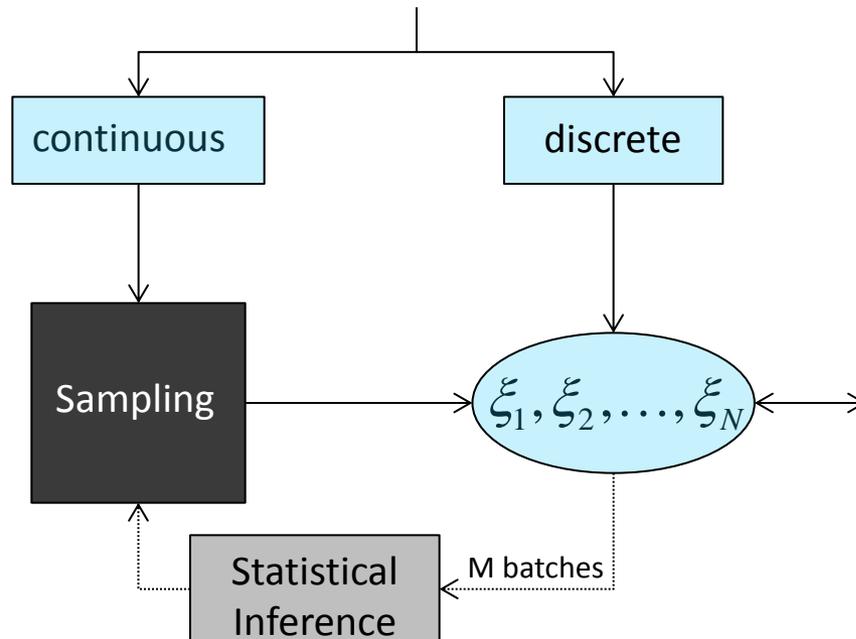
# Optimization under uncertainty

- Two-stage stochastic programming with recourse (“here-and-now”)

$$\begin{aligned} & \underset{x_0}{\text{Min}} \left\{ f_0(x_0) + \mathbb{E} \left[ \underset{x}{\text{Min}} f(x, \omega) \right] \right\} \\ & \text{subj. to. } \mathbf{W}(\omega)x = \mathbf{b}(\omega) - \mathbf{T}(\omega)x_0 \\ & \text{subj. to. } A_0x_0 = b_0 \qquad x \geq 0 \\ & \qquad \qquad \qquad x_0 \geq 0 \end{aligned}$$

- $\xi(\omega) := (\mathbf{W}(\omega), \mathbf{T}(\omega), \mathbf{b}(\omega), \mathbf{Q}(\omega), \mathbf{c}(\omega))$

$$f(x, \omega) = \frac{1}{2} x^T \mathbf{Q}(\omega)x + x^T \mathbf{c}(\omega)$$



Sample average approximation (SAA)

$$\begin{aligned} & \underset{x_0, x_1, x_2, \dots, x_N}{\text{Min}} f_0(x_0) + \frac{1}{N} \sum_{i=1}^N f_i(x_i) \\ & \text{subj. to. } A_0x_0 = b_0 \\ & \qquad \qquad T_k x_0 + W_k x_k = b_k, \\ & \qquad \qquad x_0 \geq 0, \quad x_k \geq 0, \quad k = 1, \dots, N. \end{aligned}$$

# Large-scale (dual) block-angular LPs

## Extensive form

$$\begin{array}{llllllllll} \min & c_0^T x_0 & + & c_1^T x_1 & + & c_2^T x_2 & + & \dots & + & c_N^T x_N \\ \text{s.t.} & Ax_0 & & & & & & & & = & b_0, \\ & T_1 x_0 & + & W_1 x_1 & & & & & & = & b_1, \\ & T_2 x_0 & & & + & W_2 x_2 & & & & = & b_2, \\ & \vdots & & & & & & \ddots & & & \vdots \\ & T_N x_0 & & & & & & & + & W_N x_N & = & b_N, \\ & x_0 \geq 0, & x_1 \geq 0, & x_2 \geq 0, & \dots, & x_N \geq 0. & & & & & & \end{array}$$

- In terminology of stochastic LPs:
  - First-stage variables (decision now):  $x_0$
  - Second-stage variables (recourse decision):  $x_1, \dots, x_N$
  - Each diagonal block is a realization of a random variable (scenario)



# Computational challenges and difficulties

- May require many scenarios (100s, 1,000s, 10,000s ...) to accurately model uncertainty
- “Large” scenarios ( $W_i$  up to 250,000 x 250,000)
- “Large” 1<sup>st</sup> stage (1,000s, 10,000s of variables)
- Easy to build a practical instance that requires **terabytes** of RAM to solve
  - ➔ Requires distributed memory
- Solution approach: interior-point methods (IPM)



# Linear algebra of primal-dual interior-point methods (IPM)

Convex quadratic problem

$$\begin{aligned} \text{Min } & \frac{1}{2} x^T Q x + c^T x \\ \text{subj. to. } & Ax = b \\ & x \geq 0 \end{aligned}$$



IPM Linear System

$$\begin{bmatrix} Q + \Lambda & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = rhs$$



Multi-stage SP

Two-stage SP

**nested** arrow-shaped linear system  
(modulo a permutation)

$$\begin{bmatrix} Q_1 & W_1^T & & & & & & 0 & 0 \\ W_1 & 0 & & & & & & T_1 & 0 \\ & & Q_2 & B_2^T & & & & 0 & 0 \\ & & W_2 & 0 & & & & T_2 & 0 \\ & & & & \ddots & & & \vdots & \vdots \\ & & & & & Q_S & W_S^T & 0 & 0 \\ & & & & & W_S & 0 & T_S & 0 \\ 0 & T_1^T & 0 & T_2^T & \dots & 0 & T_S^T & Q_0 & A_0^T \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 & A_0 & 0 \end{bmatrix}$$



## Special Structure of KKT System (Arrow-shaped)

$$\begin{bmatrix} K_1 & & & B_1 \\ & \ddots & & \vdots \\ & & K_N & B_N \\ B_1^T & \dots & B_N^T & K_0 \end{bmatrix} \begin{bmatrix} \Delta z_1 \\ \vdots \\ \Delta z_N \\ \Delta z_0 \end{bmatrix} = \begin{bmatrix} r_1 \\ \vdots \\ r_N \\ r_0 \end{bmatrix}$$

where,

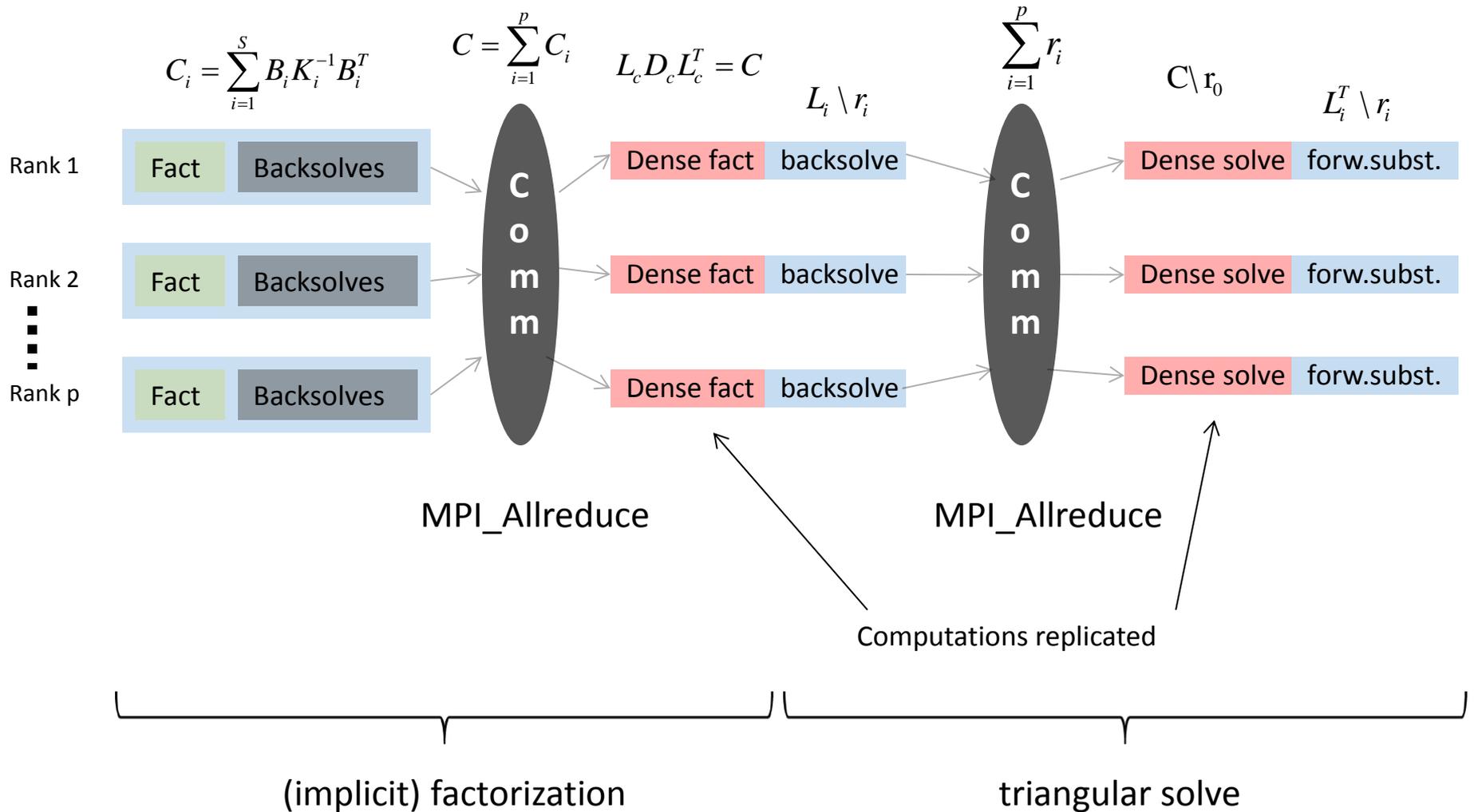
$$K_i := \begin{bmatrix} \bar{Q}_i & W_i^T \\ W_i & 0 \end{bmatrix}, \quad K_0 := \begin{bmatrix} \bar{Q} & A^T \\ A & 0 \end{bmatrix},$$
$$B_i := \begin{bmatrix} 0 & 0 \\ T_i & 0 \end{bmatrix}, \quad i = 1, 2, \dots, N.$$

## Solution Procedure for KKT System – a compact view

1. Calculate  $B_i^T K_i^{-1} B_i$ ,  $i = 1, \dots, N$  (“Compute S.C.”)
2. Form  $C := K_0 - \sum_{i=1}^N B_i^T K_i^{-1} B_i$  (“Form S.C.”)
3. Factorize  $C = L_0 D_0 L_0^T$  (“Factor S.C.”)
4. Solve  $\Delta z_0 = C^{-1} (r_0 - \sum_{i=1}^N B_i^T K_i^{-1} r_i)$
5. Solve  $\Delta z_i = K_i^{-1} (B_i \Delta z_0 - r_i)$ ,  $i = 1, \dots, N$



# Parallel computational pattern



# Implementation considerations

- Codename: **PIPS-IPM**
- C++ code based on OOQP optimization solver (S. Wright & M. Gertz, ANL 2003)
- Hybrid parallel: MPI+OpenMP/GPU.
  
- Data matrices are **sparse**
  
- Direct (sparse and dense) factorizations are needed
  - **saddle-point** linear systems: symmetric but indefinite
  - increasingly **ill-conditioned** as the optimality is approached.
  
- Second-stage linear systems handled with off-the-shelf **sparse** linear solvers (MA27/57/86 or PARDISO, other can be adopted as well)
  
- The **dense** Schur complement is solved using LAPACK/MAGMA.



# Incomplete augmented factorization technique

"Compute S.C." (Step 1):  $L_{22}U_{22} = -B_i^T K_i^{-1} B_i$

$$\begin{bmatrix} K_i & B_i^T \\ B_i & 0 \end{bmatrix} = \begin{bmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{bmatrix} \begin{bmatrix} U_{11} & U_{12} \\ 0 & U_{22} \end{bmatrix}$$

## UC12 – BG/P

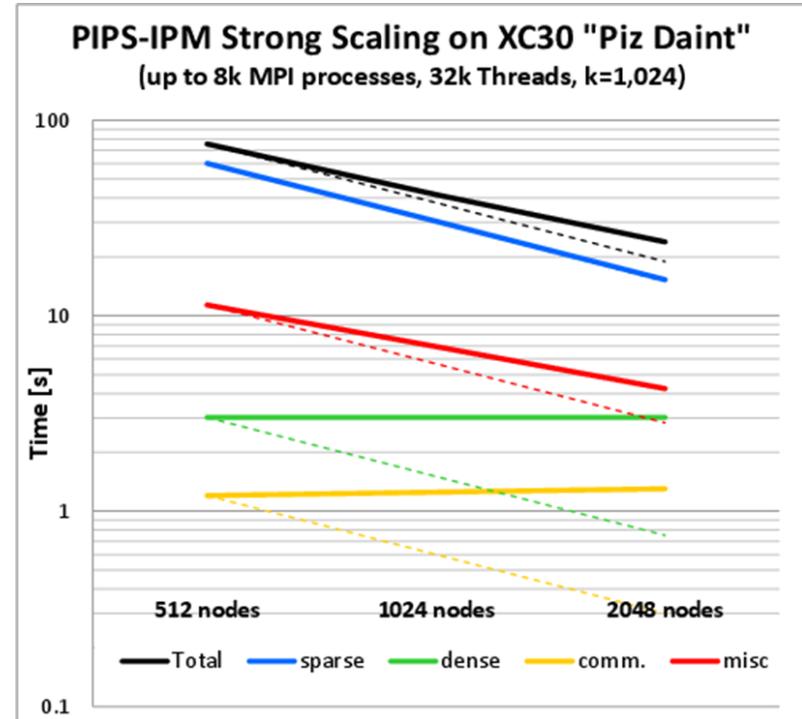
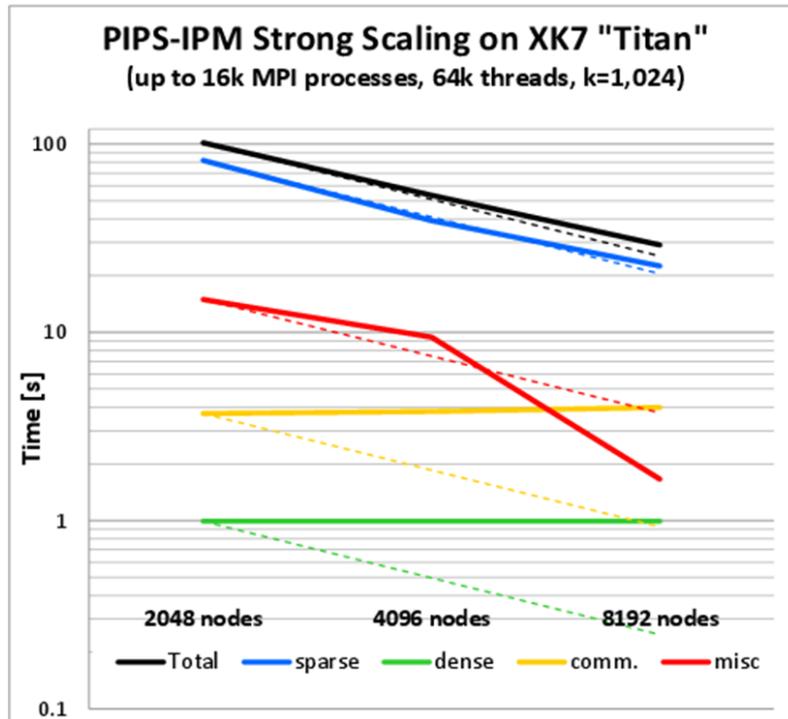
| Test Problem | Solver     | Threads/MPI | $B_i^T K_i^{-1} B_i$ Time (Sec.) |
|--------------|------------|-------------|----------------------------------|
| UC4          | MA57       | 1           | 29.65                            |
|              | PARDISO-SC | 1           | 2.45                             |
|              | PARDISO-SC | 2           | 1.32                             |
|              | PARDISO-SC | 4           | 0.79                             |
| UC12         | MA57       | 1           | 292.02                           |
|              | PARDISO-SC | 1           | 50.39                            |
|              | PARDISO-SC | 2           | 26.09                            |
|              | PARDISO-SC | 4           | 13.81                            |
| UC24         | MA57       | 1           | >3600                            |
|              | PARDISO-SC | 1           | 308.193                          |
|              | PARDISO-SC | 2           | 157.63                           |
|              | PARDISO-SC | 4           | 81.11                            |

## UC24 – XC30

| # of MPI ranks | (b) Number of threads per process on XC30 |      |      |            |     |     |  |
|----------------|---|------|------|------------|-----|-----|--|
|                | 1   | 1    | 2    | 4          | 8   | 16  |  |
|                | PARDISO                                   |      |      | PARDISO-SC |     |     |  |
| 1              | 168.3                                     | 19.0 | 12.9 | 8.4        | 6.2 | 5.9 |  |
| 2              | 175.3                                     | 18.9 | 12.4 | 8.5        | 7.4 |     |  |
| 4              | 194.6                                     | 19.5 | 13.8 | 12.1       |     |     |  |
| 8              | 284.2                                     | 22.1 | 21.8 |            |     |     |  |
| 16             | 281.6                                     | 38.5 |      |            |     |     |  |

Petra et al., "An augmented incomplete factorization approach for computing the Schur complement in stochastic optimization," SIAM Journal on Scientific Computing, 2014

# Strong scaling



The instance used in the XK7 runs has 4.08 billion decision variables and 4.12 billion constraints.

Petra et al., "Real-time Stochastic Optimization of Complex Energy Systems on High Performance Computers," accepted to IEEE Computing in Science & Engineering (CiSE), 2014

## Algorithmic scalability

- In addition to implementation, **the IPM algorithm is also scalable**

| Nodes/scens | Wall time (sec) | IPM Iterations | Time per IPM iteration (sec) |
|-------------|-----------------|----------------|------------------------------|
| 4096        | 3548.5          | 103            | 33.57                        |
| 8192        | 3883.7          | 112            | 34.67                        |
| 16384       | 4208.8          | 123            | 34.80                        |
| 32768       | 4781.7          | 133            | 35.95                        |

# StochJuMP - parallel algebraic modelling for stochastic optimization

J. Huchette, M. Lubin, C. Petra , *“Parallel algebraic modeling for stochastic optimization,”* High Performance Technical Computing in Dynamic Languages (HPTCDL), SC’14.

# Expressing and constructing the stochastic optimization problem

- Express the problem in a human-readable, mathematical format
- Automatic transformation to the low-level format of the solver(s)
  - **efficient and distributed-memory generation of the large models**

$$\begin{array}{llllllllll} \min & c_0^T x_0 & + & c_1^T x_1 & + & c_2^T x_2 & + & \dots & + & c_N^T x_N \\ \text{s.t.} & Ax_0 & & & & & & & & = & b_0, \\ & T_1 x_0 & + & W_1 x_1 & & & & & & = & b_1, \\ & T_2 x_0 & & & + & W_2 x_2 & & & & = & b_2, \\ & \vdots & & & & & & \ddots & & & \vdots \\ & T_N x_0 & & & & & & & + & W_N x_N & = & b_N, \\ & x_0 \geq 0, & x_1 \geq 0, & x_2 \geq 0, & \dots, & x_N \geq 0. & & & & & & \end{array}$$

- The problem's structure is passed **transparently** to the solver



# JuMP – modeling language for Mathematical Programming in Julia

- Miles Lubin (MIT), Iain Dunning (MIT)
- Julia – a fresh approach to scientific and technical computing
  - high-level, high-performance, open-source dynamic language for technical computing
  - keeps productivity of dynamic languages without giving up speed (2x of C/C++/Fortran)
- JuMP - compact, easy-to-use AML in Julia for modelling LP/QP/MILP/MIQCQP

```
m = Model(:Max)
@defVar(m, 0 <= x[j=1:N] <= 1)
@setObjective(m, sum{profit[j] * x[j], j=1:N})
@addConstraint(m, sum{weight[j] * x[j], j = 1:N} <= C)
```

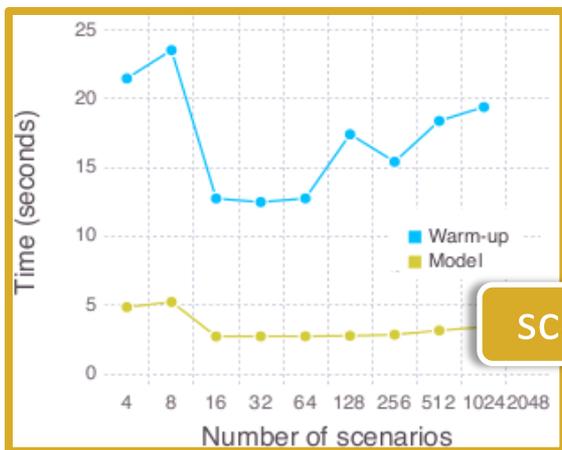
- Macros instead of operator overloading (known to have poor performance)
- Efficient sparse internal generation and representation of the data
- Nonlinear programming fully functional

**Table:** Linear-quadratic control benchmark results.  $N=M$  is the grid size. Total time (in seconds) to process the model definition and produce the output file in LP and MPS formats (as available).

| N     | JuMP/Julia |      | AMPL | Gurobi/C++ |      | Pulp/PyPy |      | Pyomo |
|-------|------------|------|------|------------|------|-----------|------|-------|
|       | LP         | MPS  | MPS  | LP         | MPS  | LP        | MPS  | LP    |
| 250   | 0.5        | 0.9  | 0.8  | 1.2        | 1.1  | 8.3       | 7.2  | 13.3  |
| 500   | 2.0        | 3.6  | 3.0  | 4.5        | 4.4  | 27.6      | 24.4 | 53.4  |
| 750   | 5.0        | 8.4  | 6.7  | 10.2       | 10.1 | 61.0      | 54.5 | 121.0 |
| 1,000 | 9.2        | 15.5 | 11.6 | 17.6       | 17.3 | 108.2     | 97.5 | 214.7 |

# StochJuMP - parallel algebraic modelling for stochastic optimization (Huchette, Lubin, Petra -2014)

- Extension of JuMP for stochastic LP/QP/MILP/ MIQCQP
- Interfaced with PIPS, runs efficiently on “Blues” LCRC cluster
- Parallel, memory-distributed generation of the model



scalable

compact

StochJuMP

efficient

| N    | Load Module | Generate Model |
|------|-------------|----------------|
| 4    | 3.528       | 1.409          |
| 8    | 3.694       | 1.415          |
| 16   | 3.334       | 1.169          |
| 32   | 2.541       | 0.917          |
| 64   | 2.863       | 0.996          |
| 128  | 3.462       | 0.768          |
| 256  | 2.620       | 0.723          |
| 512  | 2.871       | 0.689          |
| 1024 | 1.470       | 0.384          |
| 2048 | -           | 0.500          |

RATIO OF STOCHJUMP TIMINGS OVER SOLVE TIME (×100).

```

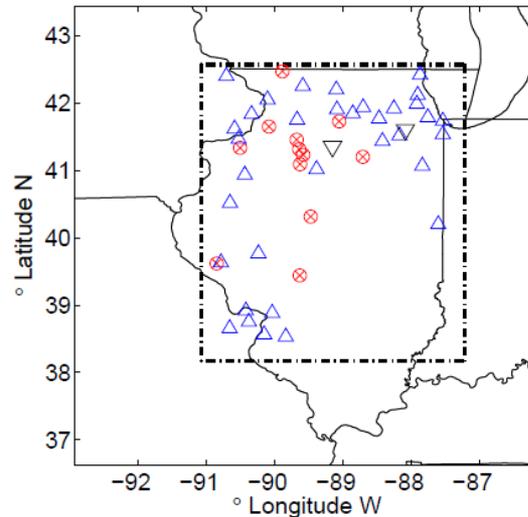
1 m = StochasticModel(NS)
2
3 # Stage 0
4 @defVar(m, 0 <= Pgen_f[i=GENTHE] <= np_capThe[i])
5 @defVar(m, 0 <= PgenWin_f[i=GENWIN] <= np_capWin[i])
6 @defVar(m, -lineCutoff*Pmax[i] <= P_f[i=LIN] <= lineCutoff*Pmax[i])
7
8 # (forward) power flow equations
9 @addConstraint(m, pfeq_f[j=BUS],
10 +sum{P_f[i], i=LIN; j=rec_bus[i]}
11 -sum{P_f[i], i=LIN; j=snd_bus[i]}
12 +sum{Pgen_f[i], i=GENTHE; j=bus_genThe[i]}
13 +sum{PgenWin_f[i], i=GENWIN; j=bus_genWin[i]}
14 -sum{loads[i], i=LOAD; j=bus_load[i]} >= 0)
15
16 @second_stage m node begin
17   bl = StochasticBlock(m)
18   # variables
19   @defVar(bl, 0 <= Pgen[i=GENTHE] <= np_capThe[i])
20   @defVar(bl, 0 <= PgenWin[i=GENWIN] <= windPower[node,i])
21   @defVar(bl, -lineCutoff*Pmax[i] <= P[i=LIN] <= lineCutoff*Pmax[i])
22   @addConstraint(bl, rampUpDown[g=GENTHE],
23     -0.1np_capThe[g] <= Pgen[g] - Pgen_f[g] <= 0.1np_capThe[g])
24   # (spot) power flow equations
25   @addConstraint(bl, pfeq[j=BUS],
26     +sum{P[i]-P_f[i], i=LIN; j=rec_bus[i]}
27     -sum{P[i]-P_f[i], i=LIN; j=snd_bus[i]}
28     +sum{Pgen[i]-Pgen_f[i], i=GENTHE; j=bus_genThe[i]}
29     +sum{PgenWin[i]-PgenWin_f[i], i=GENWIN; j=bus_genWin[i]} >= 0)
30   @defVar(bl, t[GENTHE] >= 0)
31   @addConstraint(bl, t_con1[g=GENTHE],
32     t[g] >= gen_cost_the[g]*Pgen_f[g] +
33     1.2gen_cost_the[g]*(Pgen[g]-Pgen_f[g]))
34   @addConstraint(bl, t_con2[g=GENTHE],
35     t[g] >= gen_cost_the[g]*Pgen_f[g])
36   @defVar(bl, tw[GENWIN] >= 0)
37   @addConstraint(bl, t_w_con1[g=GENWIN],
38     tw[g] >= gen_cost_win[g]*PgenWin_f[g] +
39     1.2gen_cost_win[g]*(PgenWin[g]-PgenWin_f[g]))
40   @addConstraint(bl, t_w_con2[g=GENWIN],
41     tw[g] >= gen_cost_win[g]*PgenWin_f[g])
42
43   @setObjective(bl, Min, sum{t[g], g=GENTHE} + sum{tw[g], g=GENWIN})
44 end
  
```

# On the role of wind covariance estimation in power grid dispatch – a case study using PIPS-IPM



# Integrating wind samples in the economic dispatch model

- The probability distributions are usually not known, and sampling is used
- Numerical weather forecasting is needed to obtain wind samples.
- **Approach 1:** Wind farms bid energy based on their own, independent forecasts. The ISO then considers all the scenarios in the ED model.
  - Correlation among wind farms is lost
  - An exhaustive list of scenarios leads to a gigantic ED problem. Not clear how to bundle scenarios to reduce dimensionality.
- **Approach 2:** Centralized forecast at the ISO level



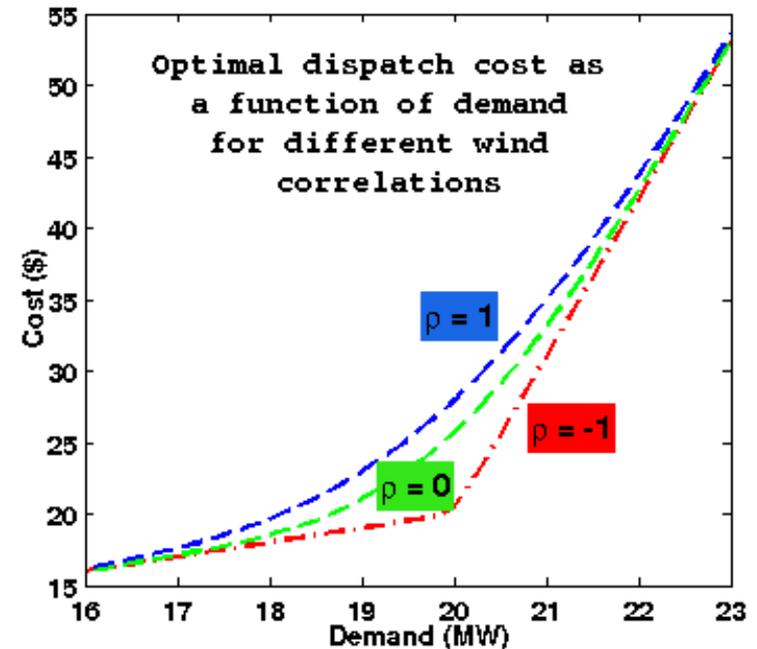
- Here we show that Approach 2 should be considered: ignoring or missing correlation information leads to inefficient dispatch and pricing.

# Motivating example – role of correlation in dispatch

- A very simplistic model: 3 generators (of which 2 wind farms and 1 thermal), 1 demand node, no line constraints
- Power outputs of the wind farms are  $W_1 \sim \mathcal{N}(w_1, \sigma_1)$  and  $W_2 \sim \mathcal{N}(w_2, \sigma_2)$ , and the correlation is  $\rho$  ( $\rho = \mathbb{E}[(W_1 - w_1)(W_2 - w_2)] / (\sigma_1 \sigma_2)$ ).
- **How does correlation affect the optimal dispatch cost?**
- The optimization problem can be solved analytically, and the (expected) optimal dispatch cost is:
$$c_d(\rho) = c_w d + (c_{th} - c_w) \left( (d - w_1 - w_2) \Phi(d, \sigma_1^2 + 2\rho\sigma_1\sigma_2 + \sigma_2^2) + \sigma^2 \phi(d, \sigma_1^2 + 2\rho\sigma_1\sigma_2 + \sigma_2^2) \right)$$
- Here  $\Phi$  and  $\phi$  are the cumulative distribution and probability distribution functions of  $W = W_1 + W_2$
- The optimal dispatch cost is an **increasing function** of the correlation  $\rho$  !

## Motivating example - continued

- Not accounting for positive correlation leads to an “optimistic dispatch” (more wind power is thought to be available).
- Ignoring negative correlations results in an “pessimistic dispatch” (foreseen wind is low, therefore, more thermal generation is dispatched).
- In both cases higher operating costs are obtained over time:
  - “optimistic”: predicted  $>$  realized, which requires expensive reserves
  - “pessimistic”: predicted  $<$  realized, cheap wind was not used and more (expensive) thermal generation than necessary was dispatched.
- Also leads to arbitrage opportunities in the power market for participants that account or have better approximation of the correlation.



# Case study for the economic dispatch for Illinois grid

- The network consists of 2522 lines, 1908 buses, 870 demand buses, 225 generators, of which 32 are wind farms.
- Wind “installed” capacity is 17%. Adoption in around 15%.

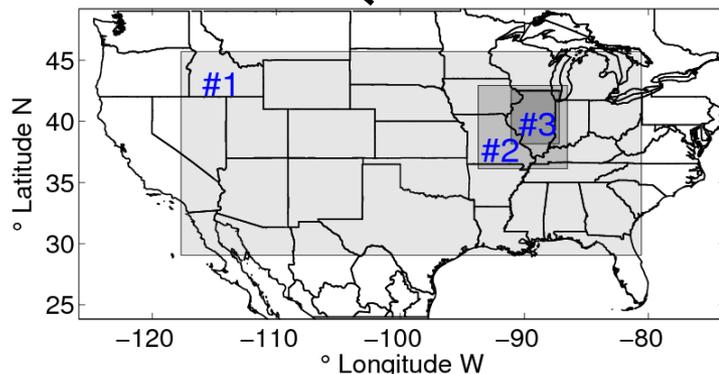
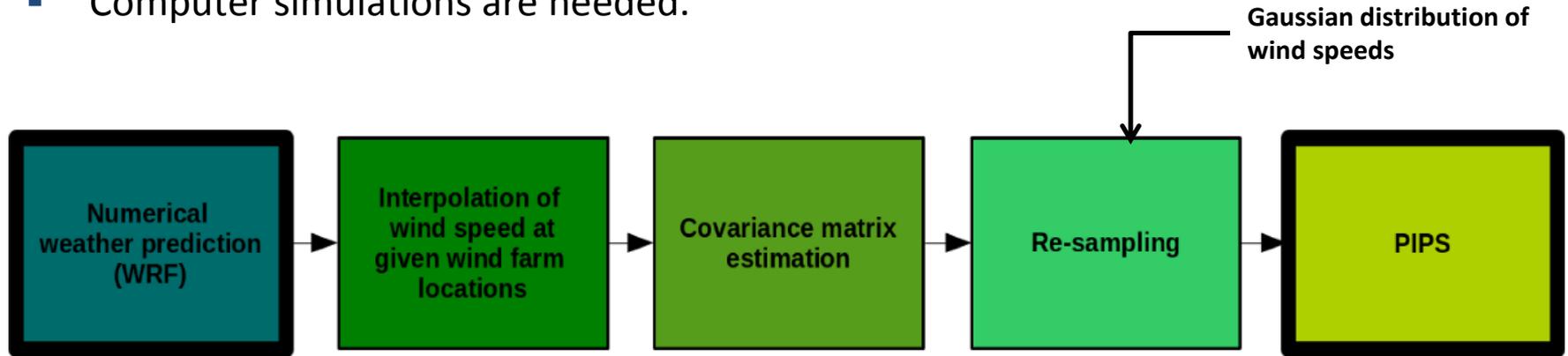
$$\begin{aligned} \min_{x, X(\omega), f, F(\omega)} \quad & \sum_{i \in G} c_i x_i + \mathbb{E}_\omega \sum_{i \in G} p_i |x_i - y_i(\omega)| \\ \text{subj.to:} \quad & \tau_n(f) + \sum_{i \in T(n)} x_i = d_n, \forall n \in N \\ & \tau_n(F(\omega)) + \sum_{i \in T(n)} y_i(\omega) = d_n, \forall n \in N, \omega \in \Omega \\ & f, F(\omega) \in U, \forall \omega \in \Omega \\ & x_i, y_i(\omega) \in U_i, \forall i \in G, \omega \in \Omega \end{aligned}$$

- RBLW covariance matrix (“corr.”) vs diagonal covariance matrix (“indep.”)
- Argonne’s BG/P and BG/Q platforms used in numerical simulations.



# What about real-world large-scale power grid systems?

- Analytical analysis of such complex systems is virtually impossible.
- Computer simulations are needed.



Weather forecasting @ Argonne  
HPC simulation – 30 samples in RT  
(E. Constantinescu)

$$S_{\Theta} = \rho_1 \cdot I + \rho_2 \cdot S$$

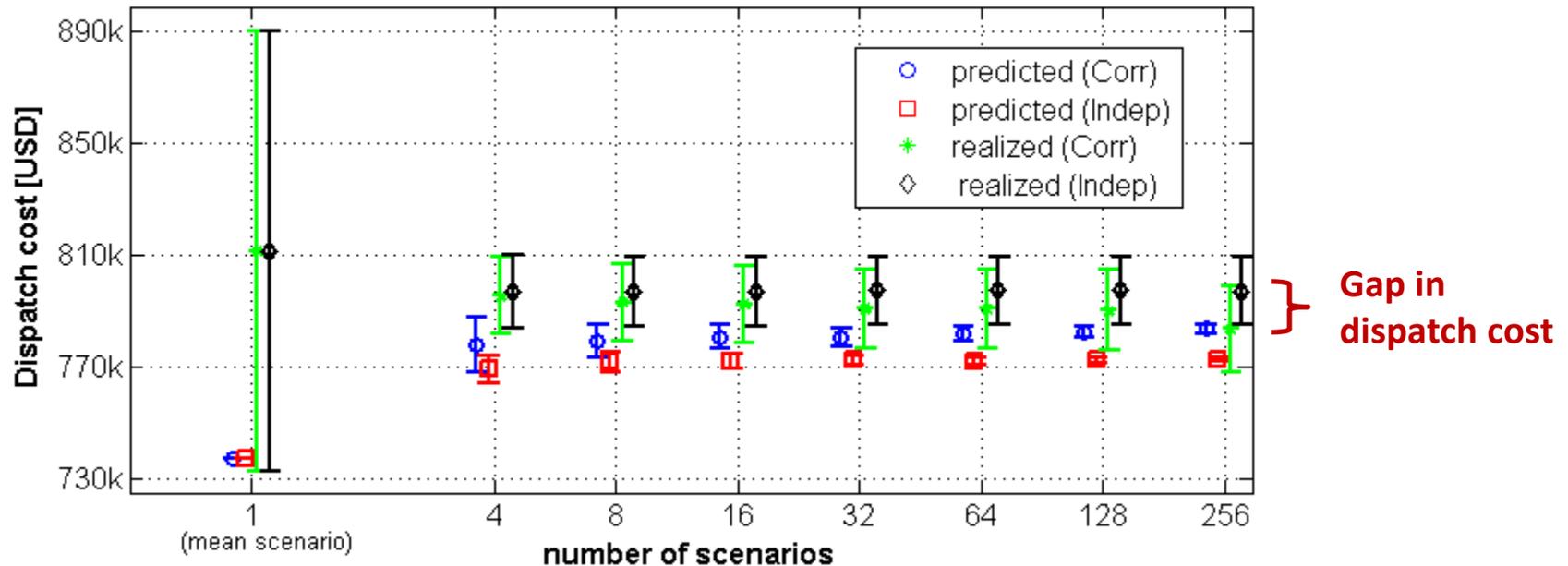
$$\min_{\rho_1, \rho_2} \mathbb{E} [\|Q - S_{\Theta}\|]$$

$$\hat{\Sigma}_{RBLW} = \rho_{RBLW} \cdot I_{p \times p} + (1 - \rho_{RBLW}) \cdot S, \text{ where}$$

$$\rho_{RBLW} = \min \left( \frac{\frac{n-2}{n} \cdot \text{tr}(S^2) + \text{tr}^2(S)}{(n+2) \cdot [\text{tr}(S^2) - \frac{\text{tr}^2(S)}{p}]}, 1 \right)$$

# Dispatch cost – correlation vs independent resampling

95% confidence intervals for the dispatch cost for predicted and realized costs, each with (Corr) and without (Indep) correlation information

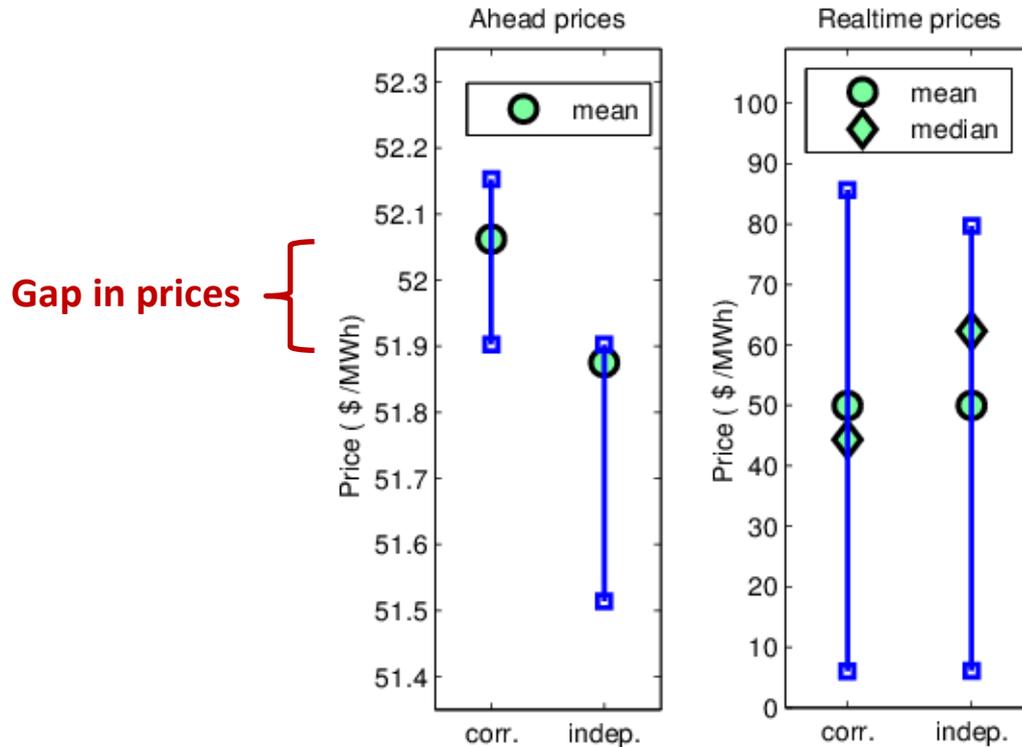


- 1.42% gap or \$10,967 (256 scenarios)
- Gap can potentially add up to approx. \$100 million over a year.



# Prices - correlation vs independent resampling

95% confidence intervals for prices at a typical bus



- Gap also present in the ahead prices.
- Opportunities for market arbitrage for players with better covariance information.

**Thank you for your attention!**

**Any questions?**



Additional material



# Empirical Bayesian estimator

- Again,  $n \ll p$

$$S_{EB} = \frac{p \cdot n - 2 \cdot n - 2}{p \cdot n^2} \cdot m_{EB} \cdot I + \frac{n}{n + 1} \cdot S \in \mathbb{R}^{p \times p}, \text{ where}$$
$$m_{EB} = \text{tr}(S \cdot I) / p$$

# Stein's SVD decomposition-based estimator

$$\mathbf{S} = \mathbf{U} \cdot \Sigma \cdot \mathbf{U}^t,$$

$$\Sigma = \mathbf{diag}(\sigma_1, \sigma_2, \dots, \sigma_p) \in \mathfrak{R}^{p \times p}.$$

$$\hat{\sigma}_i = \frac{n \cdot \sigma_i}{n - p + 1 + 2 \cdot \sigma_i \cdot \sum_{k=1, k \neq i}^p \frac{1}{\sigma_i - \sigma_k}}, i = 1, 2, \dots, p,$$

$$\Lambda = \mathbf{diag}(\hat{\sigma}_1, \hat{\sigma}_2, \dots, \hat{\sigma}_p) \in \mathfrak{R}^{p \times p},$$

$$\mathbf{S}_{\text{SH}} = \mathbf{U} \cdot \Lambda \cdot \mathbf{U}^t \in \mathfrak{R}^{p \times p}.$$



# Shrinkage estimators

- Estimators of the form

$$S_{\Theta} = \rho_1 \cdot I + \rho_2 \cdot S$$

- Where the parameters are chosen so that

$$\min_{\rho_1, \rho_2} \mathbb{E} [\|Q - S_{\Theta}\|]$$

- Rao Blackwell Ledoit Wolf (2004) estimator

$$\hat{\Sigma}_{RBLW} = \rho_{RBLW} \cdot I_{p \times p} + (1 - \rho_{RBLW}) \cdot S, \text{ where}$$
$$\rho_{RBLW} = \min \left( \frac{\frac{n-2}{n} \cdot \text{tr}(S^2) + \text{tr}^2(S)}{(n+2) \cdot \left[ \text{tr}(S^2) - \frac{\text{tr}^2(S)}{p} \right]}, 1 \right)$$

