Solving large-scale optimal power flow instances

François Pacaud
frapac.github.io

Centre Automatique et Systèmes, Mines Paris - PSL

Guest lecture
Institute of Control Systems (ICS), TU Hamburg
July 10th, 2025

Contents

What have you learned before in this class?

- Lecture II: Power flow
- Lecture III: Optimal power flow
- Lecture IV: Stochastic OPF

What will you learn today?

Solve practical power system problems in the large-scale regime ($>$ 10,000 buses)

What is current state-of-the-art?

- Solving an OPF with 10,000 buses takes a few seconds on a laptop
- Real challenge lies in solving efficiently OPF variants. By order of difficulties:
  - Multiperiod OPF with loose time coupling
    (ramping constraints for the generators, damsvalley, batteries)
  - Security-constrained OPF with $N-1$ security criterion
  - Large-scale unit-commitment ($\rightarrow$ binary variables!)

Outline

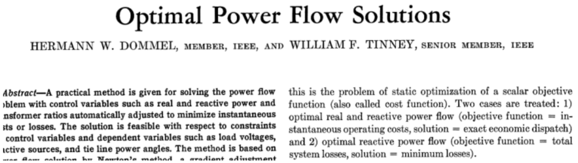
Brief historical perspectives

The importance of sparse linear algebra

Solving large-scale optimal power flow instances

Advanced topics

A brief history



Optimal Power Flow Solutions

HERMANN W. DOMMEL, MEMBER, IEEE, AND WILLIAM F. TINNEY, SENIOR MEMBER, IEEE

Abstract—A practical method is given for solving the power flow blem with control variables such as real and reactive power and nsformer ratios automatically adjusted to minimize instantaneous sts or losses. The solution is feasible with respect to constraints control variables and dependent variables such as load voltages, ıctive sources, and tie line power angles. The method is based on wer flow solution by Newton's method, a gradient adjustment

this is the problem of static optimization of a scalar objective function (also called cost function). Two cases are treated: 1) optimal real and reactive power flow (objective function = instantaneous operating costs, solution = exact economic dispatch) and 2) optimal reactive power flow (objective function = total system losses, solution = minimum losses).

- **1962:** formalization of the OPF problem by Carpentier [1]
  See also the first German reference on this topic [Edelmann, 1965]
- **1967:** solution of the power flow equations by Newton method [Tinney and Hart, 1967]
- **1968:** first practical method to solve OPF using generalized reduced gradient [Dommel and Tinney, 1968]
- **1980s:** most OPFs are solved using sequential linear programming (SLP) or sequential quadratic programming (SQP) [Sun et al., 1984]
- **1990s:** the interior-point method becomes the dominant methods to solve OPF

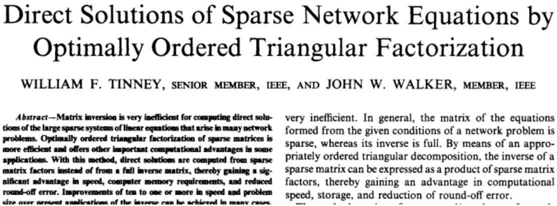
[1] I was not able to find the original paper...

Some historical lessons

Take-away

Sparse linear algebra is key for scalability!

In fact, Tinney was one of the first researcher to work on sparse linear algebra [Tinney and Hart, 1967]!



Direct Solutions of Sparse Network Equations by Optimally Ordered Triangular Factorization

WILLIAM F. TINNEY, SENIOR MEMBER, IEEE, AND JOHN W. WALKER, MEMBER, IEEE

***Abstract*—Matrix inversion is very inefficient for computing direct solutions of the large sparse systems of linear equations that arise in many network problems. Optimally ordered triangular factorization of sparse matrices is more efficient and offers other important computational advantages in some applications. With this method, direct solutions are computed from sparse matrix factors instead of from a full inverse matrix, thereby gaining a significant advantage in speed, computer memory requirements, and reduced round-off error. Improvements of ten to one or more in speed and problem size over present applications of the inverse can be achieved in many cases.**

very inefficient. In general, the matrix of the equations formed from the given conditions of a network problem is sparse, whereas its inverse is full. By means of an appropriately ordered triangular decomposition, the inverse of a sparse matrix can be expressed as a product of sparse matrix factors, thereby gaining an advantage in computational speed, storage, and reduction of round-off error.

Outline

Brief historical perspectives

The importance of sparse linear algebra

Solving large-scale optimal power flow instances

Advanced topics

Coming back to the power flow problem

For transmission network, it is commonplace to solve the power flow equations in polar form using Newton method (see Lecture II)

Using the same notations as in Lecture II, the Newton iterations write:

$$x^{k+1} = x^k - \left(D_f(x^k)\right)^{-1} f(x^k)$$

with the *square* Jacobian

$$D_f(x) = \begin{pmatrix} J^{p\theta} & J^{pv} \\ J^{q\theta} & J^{qv} \end{pmatrix} \in \mathbb{R}^{n_x \times n_x}$$

and $x \in \mathbb{R}^{n_x}$ a vector storing the values of the dependent variables

Observations

- The Jacobian $D_f(x)$ is super-sparse (and has a graph-structure)
- In practice we never compute the inverse $\left(D_f(x)\right)^{-1}$
  $\rightarrow$ use matrix factorization instead

The LU decomposition

If the Jacobian is non-singular, it decomposes as

$$D_f(x) = L\,U$$

with L lower-triangular, U upper-triangular

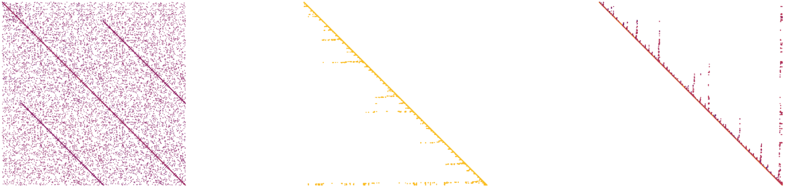
The decomposition LU is called a *matrix factorization*: it is a generalization of Gaussian elimination

Solving the triangular system $L^{-1}x$ (resp. $U^{-1}x$) is straightforward using a *backsolve*

Once the matrix factorized, solving the system $(D_f(x^k))^{-1}f(x^k)$ in the Newton iterations just translates to two successive backsolves:

$$\left(D_f(x^k)\right)^{-1} f(x^k) = U^{-1}v \quad \text{with} \quad v := L^{-1}f(x^k)$$

Illustrations



$D_f(x)$ = L x U

Figure: LU decomposition of the power flow Jacobian for `1354pegase` (computed using KLU)

Pivoting

For a sparse matrix A, we compute the LU decomposition of a permuted matrix

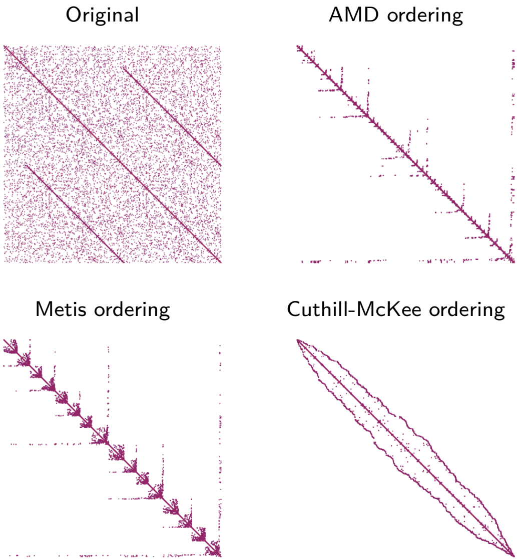
$$A = P\,LU\,Q$$

with P *row permutation* matrix and Q *column permutation* matrix

The goal is to reduce the fill-in (almost dense rows in L and U)

Finding the permutation minimizing the fill-in is NP-hard :
in practice the permutation is found using complicated heuristics
(*approximate minimum ordering* being one of the most common)

Testing different permutations for the power flow Jacobian



Original

AMD ordering

Metis ordering

Cuthill-McKee ordering

Sparse direct linear solver

In practice, the LU decomposition is computed by a sparse direct solver

Most direct solvers use the following operations:

1. *Symbolic factorization:* Analyse the matrix sparsity pattern and find an ordering that reduces the fill-in. Then instantiate the sparse factors in memory
2. *Numerical factorization:* Compute inplace the nonzero values in L and U
3. *Backsolve:* Solve the system $L^{-1}b$ and $U^{-1}d$ to get solution of the linear system

Sparse direct linear solver

In practice, the LU decomposition is computed by a sparse direct solver

Most direct solvers use the following operations:

1. *Symbolic factorization:* Analyse the matrix sparsity pattern and find an ordering that reduces the fill-in. Then instantiate the sparse factors in memory
2. *Numerical factorization:* Compute inplace the nonzero values in L and U
3. *Backsolve:* Solve the system $L^{-1}b$ and $U^{-1}d$ to get solution of the linear system

When solving power flow, the symbolic factorization has to be computed *only once* (the sparsity pattern remains fixed as the network remains the same)

Practical (open-source) sparse LU solvers

- SuperLU
- UMFPack
- KLU (particularly fast for power flow equations)

If you are interested into that topic, a very good reference is [Stewart, 2003]

Is there any alternative to sparse direct solver?

We have just seen that the system $Ax = b$ can be solved efficiently using a direct solver

Observations

- Sparse direct solver can be slow if the matrix becomes very large
- Sometimes, we cannot find a permutation that reduces the fill-in

Iterative methods (aka Krylov methods) are practical alternative to a direct solver *provided* an efficient preconditioner M is available

Iterative methods have a much lower memory footprints than direct solvers (just require sparse matrix-vector products)

Using a left-preconditioner M, an iterative method solves the preconditioned system

$$M\,A\,x = Mb$$

M can be computed e.g. using incomplete-LU, or with block-Jacobi.
An ideal preconditionner M would satisfy

$$M \approx A^{-1}$$

Practical iterative methods

A is *symmetric definite*

- Conjugate gradient (CG)
- Conjugate residual (CR)

A is *indefinite* (as is the case in power flow)

- Biconjugate gradient (BiCG)
- Biconjugate gradient stabilized (BiCGstab)
- Generalized minimal residual method (GMRES)

Observation

In practice, iterative methods are harder to tune than a direct solver, but can be much more effective

→ for power flow, BiCGstab is a valid alternative to a sparse LU solver

Numerical experiments

Experiment

Solve the power flow equations in 9241pegase (from matpower) with the solver KLU

```
it 0:  1.08645e+02
it 1:  8.84101e+01
it 2:  1.88123e+01
it 3:  3.45636e+00
it 4:  1.75824e-01
it 5:  4.59846e-04
it 6:  3.02185e-09
Power flow has converged:  true
* iterations:  6
* Time Jacobian (s) ........:  0.1044
* Time linear solver (s) ...:  0.0161
* Time total (s) ...........:  0.1216
```

Outline

Brief historical perspectives

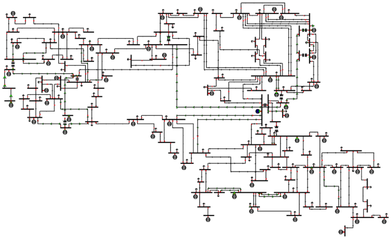
The importance of sparse linear algebra

Solving large-scale optimal power flow instances

Advanced topics

We are now ready to tackle the optimal power flow problem

From now on, we consider the optimal power flow in polar form



$$\begin{aligned} \min_{x,u} \ & f(x,u) \\ \text{subject to} \ & \boxed{g(x,u) = 0} \\ & \boxed{h(x,u) \geq 0} \end{aligned}$$

power flow

line flow constraints

with

- x: dependent variables
  (voltage magnitudes at PQ nodes, voltage angles)
- u: degrees of freedom
  (voltage magnitudes at PV and REF nodes, active power generations)

Observations

- The optimal power flow problem is non-convex
- Its coupling structure is determined by the power flow equations

Karush-Kuhn-Tucker (KKT) conditions

A locally optimal solution of the OPF satisfies the KKT conditions:

$$\begin{aligned}
&\nabla_x f(x,u) + D_{g,x}(x,u)^\top y - D_{h,x}(x,u)^\top z = 0\\
&\nabla_u f(x,u) + D_{g,u}(x,u)^\top y - D_{h,u}(x,u)^\top z = 0\\
&g(x,u) = 0\\
&0 \leq z \perp h(x,u) \geq 0
\end{aligned}$$

where the symbol $\perp$ denotes the complementarity constraints

$$z_i h_i(x,u) = 0 \quad \forall i = 1, \ldots, m_h$$

Observations

- If we find a solution of the KKT conditions with *positive curvature*, then it is a locally optimal solution
- The KKT conditions are nonsmooth equations

Coming back to the origin: Dommel & Tinney method

The power flow equations are easy to solve: for a fixed u, the Newton method finds a state $x(u)$ solution of the power flow

$$g\Big(x(u), u\Big) = 0$$

The *implicit function theorem* tells us that $x(u)$ is at least twice differentiable, its Jacobian satisfying

$$D_x(u) = -\Big(D_{g,x}(x(u), u)\Big)^{-1} D_{g,u}(x(u), u)$$

The OPF problem becomes equivalent to the reduced problem

$$\min_u \; f(x(u), u) \quad \text{subject to} \quad h(x(u), u) \geq 0$$

Observations

- Dommel & Tinney method is a *reduced space* method
  It was the first practical method to solve OPF
- In practice, Hessian is *fully dense* and can be complicated to evaluate
- Original method encountered difficulties in handling the inequality constraints

Current state-of-the-art: the interior-point method (IPM)

N.B.: The inequality constraints are the real difficult part in OPF

First, IPM reformulates the OPF problem with a slack variable s such as

$$\begin{aligned} &\min_{x,u,s} \; f(x,u) \\ \text{subject to} \quad & g(x,u) = 0 \\ & h(x,u) - s = 0 \; , \; s \geq 0 \end{aligned}$$

For a given barrier parameter $\mu > 0$, the primal-dual interior-point method approximates the KKT conditions of this new problem by

$$\begin{aligned} & \nabla_x f(x,u) + D_{g,x}(x,u)^\top y - D_{h,x}(x,u)^\top z = 0 \\ & \nabla_u f(x,u) + D_{g,u}(x,u)^\top y - D_{h,u}(x,u)^\top z = 0 \\ & g(x,u) = 0 \\ & h(x,u) - s = 0 \\ & \boxed{0 \leq z \perp s \geq 0} \end{aligned}$$

Current state-of-the-art: the interior-point method (IPM)

N.B.: The inequality constraints are the real difficult part in OPF

First, IPM reformulates the OPF problem with a slack variable s such as

$$\begin{aligned} &\min_{x,u,s} \; f(x,u) \\ \text{subject to} \quad & g(x,u) = 0 \\ & h(x,u) - s = 0 \; , \; s \geq 0 \end{aligned}$$

For a given barrier parameter $\mu > 0$, the primal-dual interior-point method approximates the KKT conditions of this new problem by

$$\begin{aligned} & \nabla_x f(x,u) + D_{g,x}(x,u)^\top y - D_{h,x}(x,u)^\top z = 0 \\ & \nabla_u f(x,u) + D_{g,u}(x,u)^\top y - D_{h,u}(x,u)^\top z = 0 \\ & g(x,u) = 0 \\ & h(x,u) - s = 0 \\ & \boxed{ZSe = \mu e} \end{aligned}$$

Observation

We obtain a system of nonlinear equations, solvable by Newton method!

Newton step

Starting from a feasible iterate, the Newton direction is computed at each iteration as solution of the linear system

$$\begin{bmatrix} W_{xx} & W_{xu} & 0 & G_x^\top & -H_x^\top \\ W_{ux} & W_{uu} & 0 & G_u^\top & -H_u^\top \\ 0 & 0 & Z & 0 & S \\ G_x & G_u & 0 & 0 & 0 \\ H_x & H_u & -I & 0 & 0 \end{bmatrix} \begin{pmatrix} \Delta x \\ \Delta u \\ \Delta s \\ \Delta y \\ \Delta z \end{pmatrix} = - \begin{pmatrix} kkt_x \\ kkt_u \\ kkt_s \\ kkt_y \\ kkt_z \end{pmatrix}$$

As before, we obtain an indefinite matrix!

Observation

So we use sparse LU, or?

Augmented KKT system

In fact, the linear system is equivalent to the symmetric indefinite system

$$\begin{bmatrix} W_{xx} & W_{xu} & 0 & G_x^\top & H_x^\top \\ W_{ux} & W_{uu} & 0 & G_u^\top & H_u^\top \\ 0 & 0 & S^{-1}Z & 0 & -I \\ G_x & G_u & 0 & 0 & 0 \\ H_x & H_u & -I & 0 & 0 \end{bmatrix} \begin{pmatrix} \Delta x \\ \Delta u \\ \Delta s \\ \Delta y \\ -\Delta z \end{pmatrix} = - \begin{pmatrix} kkt_x \\ kkt_u \\ S^{-1} kkt_s \\ kkt_y \\ kkt_z \end{pmatrix}$$

This linear system is called the *augmented KKT system*

The augmented KKT system can be factorized efficiently using a LBL decomposition (a generalization of the Cholesky factorization)

Observations

- This is the system used in current nonlinear IPM solvers (Ipopt, Knitro, MadNLP)
- The system becomes ill-conditioned as we are reaching convergence (for active constraints $s_i \to 0$)
- In practice, 90% of the time is spent factorizing the augmented KKT system

Inertia revealing solver

The inertia of a symmetric matrix A is the triple (n_+, n_-, n_0) giving the number of positive, negative and zero eigenvalues

Proposition

If in the augmented KKT system

1. The Jacobian is full row-rank
2. The reduced Hessian[2] is positive definite

then the solution Δ is a *descent direction*.

This is equivalent to check that the inertia of the augmented KKT system is $(n + m_h, m_g + m_h, 0)$.

The solver checks the inertia of the matrix at each iteration. If the inertia is not correct, the system is regularized as

$$\begin{bmatrix} W_{xx} + \delta_w I & W_{xu} & 0 & G_x^\top & H_x^\top \\ W_{ux} & W_{uu} + \delta_w I & 0 & G_u^\top & H_u^\top \\ 0 & 0 & S^{-1}Z & 0 & -I \\ G_x & G_u & 0 & -\delta_c I & 0 \\ H_x & H_u & -I & 0 & 0 \end{bmatrix}$$

[2] the projection of the Hessian onto the null-space of the Jacobian

Solving the augmented KKT system

Implementing a stable LBL decomposition requires complicated pivoting operations

Duff-Reid factorization:

$$A = PQ\,LBL^\top\,Q^\top P^\top$$

with

- P: fill-in minimization matrix
- Q: additional pivoting for numerical stability
- L: unit lower-triangular matrix
- B: block diagonal matrix with blocks of dimension 1×1 or 2×2

→ Getting an efficient algorithm for the numerical pivoting Q is highly non trivial

The LBL factorization has become competitive only in the 1990s, using a technique known as matching-based preprocessing [Duff and Koster, 2001]

The progress in sparse linear solvers has directly benefited to nonlinear optimization solvers such as Ipopt or Knitro

Practical linear solvers implementing the LBL factorization

`MUMPS` (Multifrontal massively parallel sparse direct solver)

- Large-scale solver leveraging MPI
- Open-source, but does not offer the best performance for OPF
- Use by default in Ipopt

`HSL` (Harwell Subroutine Library)

- HSL MA27 : multifrontal LBL solver, naive implementation
- HSL MA57 : multifrontal LBL solver, using BLAS as a backend
- HSL MA86 : supernodal LBL solver, using multithreading
- HSL MA97 : same as MA86, but with multiprocessing using MPI

`Panua Pardiso` (supernodal LBL solver)

Observation

The solver HSL MA27 is often offering the best performance for OPF problems

The importance of automatic differentation

IPM has to evaluate the Hessian W and the Jacobians G and H
→ these matrices are *highly sparse*

Observations

- We never evaluate the Hessian W and the Jacobians G and H by hand (except in matpower...)
- Use (sparse) AD!
- AD is usually provided for free if you use an optimization modeler
  - AMPL
  - CasADi
  - JuMP

Using automatic differentiation (AD), we can evaluate G and H using one forward pass (=Jacobian vector-product), and the Hessian W with a forward-over-reverse pass (=Jacobian vector-product of the gradient)

Take into account the sparsity pattern of each matrix to reduce the number of Jacobian-vector products required! → *sparse matrix coloring method*

Practical optimization solvers to solve large-scale OPF

`Ipopt`

- Filter line-search globalization
- Open-source

`Knitro`

- Trust-region steps using the Byrd & Omojokun method
- Commercial

`MadNLP`

- Port of Ipopt in Julia
- Open-source, and supports GPU acceleration

Numerical results

Observation

We analyse the convergence of Ipopt on 9241pegase

Ipopt displays the following information at each iteration:

- inf_pr: primal infeasibility
- inf_du: dual infeasibility
- lg(mu): current barrier parameter
- ||d||: norm of the descent direction
- lg(rg): current primal-dual regularization
- alpha_du: dual line-search step
- alpha_pr: primal line-search step
- ls: number of line-search step

Numerical results

Observation

We analyse the convergence of Ipopt on `9241pegase`

Ipopt displays the following information at each iteration:

- `inf_pr`: primal infeasibility *(should converge to 0)*
- `inf_du`: dual infeasibility *(should converge to 0)*
- `lg(mu)`: current barrier parameter *(should converge to 0)*
- `||d||`: norm of the descent direction *(should converge to 0)*
- `lg(rg)`: current primal-dual regularization *(should stay at 0)*
- `alpha_du`: dual line-search step *(should be close to 1 for Newton step)*
- `alpha_pr`: primal line-search step *(should be close to 1 for Newton step)*
- `ls`: number of line-search step *(should stay at 1)*

Outline

Brief historical perspectives

The importance of sparse linear algebra

Solving large-scale optimal power flow instances

Advanced topics

Security-constrained OPF with $N-1$ criterion

In practice, system operators solve more complicated variants of the OPF, with multiple time periods and security constraints

In this last section, we look at the security-constrained OPF (SC-OPF), with K a given number of contingencies (line or generator trippings). Denote

- x_0: state variable in the base nominal case
- x_k: state variable in the contingency $k = 1, \cdots, K$

What is the best control that maintains feasibility in all the contingencies?

$$\min_{x_0, x_1, \cdots, x_K, u} f(x_0, u)$$

power flow in contingency k

$$\begin{aligned} &\boxed{g(x_k, u) = 0} \quad \forall k = 0, \cdots, K \\ &\boxed{h(x_k, u) \geq 0} \quad \forall k = 0, \cdots, K \end{aligned}$$

line flow constraints in contingency k

Observe that u is the only *coupling* variable!

Block linear system with arrowhead structure

As before, we can solve the SCOPF using interior-point... but the linear system gets much larger!

We reach the limit of the current generation of linear solvers even with a small number of contingencies

Use structure exploiting method instead: after careful reordering, we can prove that the linear system has a block arrowhead structure

For $w_k := (x_k, y_k, z_k)$, we solve

$$\begin{bmatrix} A_1 & & & B_1^\top \\ & \ddots & & \vdots \\ & & A_K & B_K^\top \\ B_1 & \dots & B_K & A_0 \end{bmatrix} \begin{bmatrix} \Delta w_0 \\ \vdots \\ \Delta w_K \\ \Delta u \end{bmatrix} = \begin{bmatrix} r_1 \\ \vdots \\ r_K \\ r_0 \end{bmatrix}$$

The last column corresponds to the coupling variable u

Solution with the Schur-complement

Exploit the arrowhead structure in the pivots

1. Solve the Schur-complement system

$$\left(A_0 - \sum_{k=1}^{K} B_k A_k^{-1} B_k^\top \right) \Delta u = r_0 - \sum_{k=1}^{K} B_k A_k^{-1} r_k$$

2. Recover the steps Δw_k using

$$\Delta w_k = A_k^{-1} \left(r_k - B_k^\top \Delta u \right)$$

The bottleneck lies in assembling the Schur complement:

- Require solving $A_k^{-1} B_k^\top$, a linear system with multiple sparse right-hand-sides
- Number of right-hand-sides is n_u, the number of coupling variables: the greater n_u gets, the harder the solution is
- Fortunately, the Schur-complement can be assembled efficiently using the *incomplete LU* trick [Petra et al., 2014]
- The solution of the linear system can be distributed using MPI
- The Schur-complement is more ill-conditionned than the original system

Observations

- A similar procedure exists for multiperiod OPF

Conclusion

Take away message

For performance, always leverage sparse linear algebra!

If you are interested into solving large-scale OPF, check this recent tutorial:

https://frapac.github.io/tutorials/powertech/

References I

Dommel, H. W. and Tinney, W. F. (1968).
Optimal power flow solutions.
IEEE Transactions on power apparatus and systems, (10):1866–1876.

Duff, I. S. and Koster, J. (2001).
On algorithms for permuting large entries to the diagonal of a sparse matrix.
SIAM Journal on Matrix Analysis and Applications, 22(4):973–996.

Edelmann, H. (1965).
Die berücksichtigung von ungleichungsbedingungen und konvergenzverbesserungen bei der digitalen berechnung optimaler lastverteilungen.
Archiv für Elektrotechnik, 49:320–330.

Petra, C. G., Schenk, O., Lubin, M., and Gärtner, K. (2014).
An augmented incomplete factorization approach for computing the Schur complement in stochastic optimization.
SIAM Journal on Scientific Computing, 36(2):C139–C162.

Stewart, G. (2003).
Building an old-fashioned sparse solver.

Sun, D. I., Ashley, B., Brewer, B., Hughes, A., and Tinney, W. F. (1984).
Optimal power flow by newton approach.
IEEE Transactions on Power Apparatus and systems, (10):2864–2880.

References II

Tinney, W. F. and Hart, C. E. (1967).
Power flow solution by newton's method.
IEEE Transactions on Power Apparatus and systems, (11):1449–1460.