Solving Optimal Power Flow on GPUs in Julia

SIAM Conference on Computational Science and Engineering (CSE21)

François Pacaud, Adrian Maldonado, Michel Schanen, Mihai Anitescu

Argonne National Laboratory
Mathematics and Computer Science Division

March, 3rd

Motivation

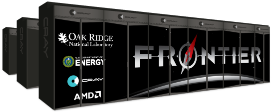
ExaSGD project

- Optimizing Stochastic Grid Dynamics at ExaScale
- Leverage new GPU-centric HPC architectures

Aurora

- Intel's Xe compute architecture
- $>$ 1 exaflops

Frontier



- AMD EPYC processors and Radeon Instinct GPU
- 1.5 exaflops

Today's challenges in power systems

Figure: The last year had been demanding for the grid

How to optimize the grid short-term response, while facing many hazards?

ExaSGD's targets

- Security constrained Optimal Power Flow (SC-OPF)
- Model stochasticity (weather, renewable) and contingencies
- Multiperiod analysis, with ramping constraints

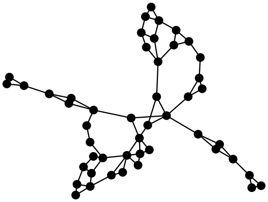
Our target

Leverage GPUs to solve

- large-scale *multiperiod* OPF
- with *contingencies*
- using the Julia language

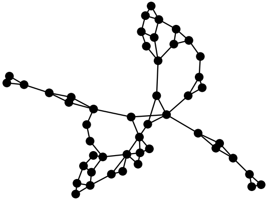
This work is part of a broader package implementing a decomposition solver, ProxAL

Solving Optimal Power Flow on GPU is easy, huh?



- Graphs are the natural abstraction for power networks, but come with *unstructured sparsity*
- OPF formulate as large-scale nonlinear nonconvex optimization problems

Solving Optimal Power Flow on GPU is easy, huh?



- Graphs are the natural abstraction for power networks, but come with *unstructured sparsity*
- OPF formulate as large-scale nonlinear nonconvex optimization problems

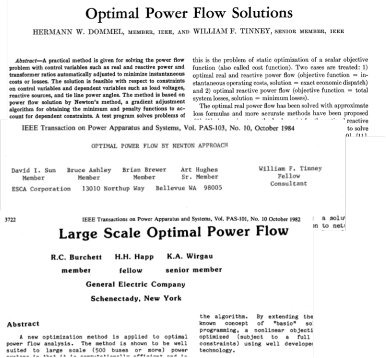
Large-scale optimization solvers rely on sparse solvers!

State-of-the-art for OPF: *Interior Points Method* (IPM)

- Newton method with very ill-conditioned linear systems
- Efficient IPM requires indefinite sparse direct inertia revealing solvers (HSL, Pardiso)...
- Sparse linear libraries on GPU are not mature (yet!)

Back to the future: Revisiting reduced-space method for OPF

A brief history of the resolution of OPF (nonlinear optimization only)


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 problem with control variables such as real and reactive power and transformer ratios automatically adjusted to minimize instantaneous costs or losses. The solution is feasible with respect to constraints on control variables and dependent variables such as load voltages, reactive sources, and tie line power angles. The method is based on power flow solution by Newton's method, a gradient adjustment algorithm for obtaining the minimum and penalty functions to account for dependent constraints. A test program solves problems of

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).

The optimal real power flow has been solved with approximate loss formulas and more accurate methods have been proposed

IEEE Transaction on Power Apparatus and Systems, Vol. PAS-103, No. 10, October 1984

OPTIMAL POWER FLOW BY NEWTON APPROACH

David I. Sun, Member; Bruce Ashley, Member; Brian Brewer, Member; Art Hughes, Sr. Member; William F. Tinney, Fellow, Consultant

ESCA Corporation 13010 Northup Way Bellevue WA 98005

3722 IEEE Transactions on Power Apparatus and Systems, Vol. PAS-101, No. 10 October 1982

Large Scale Optimal Power Flow

R.C. Burchett, member; H.H. Happ, fellow; K.A. Wirgau, senior member

General Electric Company

Schenectady, New York

Abstract

A new optimization method is applied to optimal power flow analysis. The method is shown to be well suited to large scale (500 buses or more) power

the algorithm. By extending the known concept of "basic" so programming, a nonlinear objecti optimized (subject to a full constraints) using well develope technology.

- 1962: introduction of the OPF problem by Carpentier
- 1968: Reduced Gradient method Dommel and Tinney (1968)
- 1972: Generalized Reduced Gradient Peschon et al. (1972)
- 1982: SQP method for OPF Burchett et al. (1982)
- 1984: OPF by Newton approach Sun et al. (1984)
- 1994: Primal-Dual interior points Granville (1994)

Our plan of action

The Hessian matrix in (53) is extremely difficult to compute for high-dimensional problems. In the first place, the derivatives $\mathfrak{L}_{uu}$, $\mathfrak{L}_{xx}$, $\mathfrak{L}_{xu}$ involve three-dimensional arrays, e.g., in

$$\mathfrak{L}_{xx} = \left[\frac{\partial^2 f}{\partial x^2}\right] + [\lambda]^T \left[\frac{\partial^2 g}{\partial x^2}\right]$$

where $[\partial^2 g/\partial x^2]$ is a three-dimensional matrix. This in itself is not the main obstacle, however, since these three-dimensional matrices are very sparse. This sparsity could probably be increased by rewriting the power flow equations in the form

$$\sum_{m=1}^{N} (G_{km} + jB_{km}) V_m e^{j\theta m} - \frac{P_{NETk} - jQ_{NETk}}{V_k e^{-j\theta_k}} = 0$$

and applying Newton's method to its real and imaginary part, with rectangular, instead of polar, coordinates. Then most of the first derivatives would be constants [1] and, thus, the respective second derivatives would vanish. The computational difficulty lies in the sensitivity matrix $[S]$. To see the implications for the realistic system of Fig. 6 with 328 nodes, let 50 of the 80 control parameters be voltage magnitudes, and 30 be transformer tap settings. Then the sensitivity matrix would have 48 400 entries $[605 \times 80$, where 605 reflects 327 P-equations (2) and 328 − 50 Q-equations (3)], which is far beyond the capability of our present computer. Aside from the

Figure: Dommel and Tinney (1968)

1. We revisit the reduced space method of Dommel and Tinney (1968) on the GPU
2. We compute the reduced Hessian using an adjoint-adjoint method
3. We solve the OPF problem with an Augmented Lagrangian algorithm

Formulating the OPF as a non-linear problem

We adopt the *polar formulation*

- **Objective**
  - Minimize costs of power generation

$$F(\boldsymbol{z}) = \sum_{g=1}^{n_g} c_2^g (p_g)^2 + c_1^g p_g$$

$$\min_{\boldsymbol{z}} F(\boldsymbol{z})$$

(OPF)

Formulating the OPF as a non-linear problem

We adopt the *polar formulation*

$$\min_{\boldsymbol{z}} F(\boldsymbol{z})$$

$$\boldsymbol{z} = (\boldsymbol{v}, \boldsymbol{\theta}, \boldsymbol{p}_g, \boldsymbol{q}_g) \qquad \text{(OPF)}$$

- **Objective**
  - Minimize costs of power generation

$$F(\boldsymbol{z}) = \sum_{g=1}^{n_g} c_2^g (p_g)^2 + c_1^g p_g$$

- **Variables** $\boldsymbol{z} = (\boldsymbol{v}, \boldsymbol{\theta}, \boldsymbol{p}_g, \boldsymbol{q}_g) \in \mathbb{R}^{2\times(n_b+n_g)}$
  - Voltage magnitude $\boldsymbol{v} \in \mathbb{R}^{n_b}$
  - Voltage angle $\boldsymbol{\theta} \in \mathbb{R}^{n_b}$
  - Active power generation $\boldsymbol{p}_g \in \mathbb{R}^{n_g}$
  - Reactive power generation $\boldsymbol{q}_g \in \mathbb{R}^{n_g}$

Formulating the OPF as a non-linear problem

We adopt the *polar formulation*

$$\min_{\boldsymbol{z}} F(\boldsymbol{z})$$

$$\boldsymbol{z} = (\boldsymbol{v}, \boldsymbol{\theta}, \boldsymbol{p}_g, \boldsymbol{q}_g)$$

subject to (OPF)

$$\boldsymbol{z}^\flat \leq \boldsymbol{z} \leq \boldsymbol{z}^\sharp$$

- **Objective**
  - Minimize costs of power generation

$$F(\boldsymbol{z}) = \sum_{g=1}^{n_g} c_2^g (p_g)^2 + c_1^g p_g$$

- **Variables** $\boldsymbol{z} = (\boldsymbol{v}, \boldsymbol{\theta}, \boldsymbol{p}_g, \boldsymbol{q}_g) \in \mathbb{R}^{2\times(n_b+n_g)}$
  - Voltage magnitude $\boldsymbol{v} \in \mathbb{R}^{n_b}$
  - Voltage angle $\boldsymbol{\theta} \in \mathbb{R}^{n_b}$
  - Active power generation $\boldsymbol{p}_g \in \mathbb{R}^{n_g}$
  - Reactive power generation $\boldsymbol{q}_g \in \mathbb{R}^{n_g}$
- **Constraints**
  - Bounds $\boldsymbol{z}^\flat \leq \boldsymbol{z} \leq \boldsymbol{z}^\sharp$

Formulating the OPF as a non-linear problem

We adopt the *polar formulation*

$$\min_{z} F(z)$$

$$z = (v, \theta, p_g, q_g)$$

subject to (OPF)

$$z^\flat \leq z \leq z^\sharp$$

$$G(z) = 0$$

- **Objective**
  - Minimize costs of power generation

$$F(z) = \sum_{g=1}^{n_g} c_2^g (p_g)^2 + c_1^g p_g$$

- **Variables** $z = (v, \theta, p_g, q_g) \in \mathbb{R}^{2\times(n_b+n_g)}$
  - Voltage magnitude $v \in \mathbb{R}^{n_b}$
  - Voltage angle $\theta \in \mathbb{R}^{n_b}$
  - Active power generation $p_g \in \mathbb{R}^{n_g}$
  - Reactive power generation $q_g \in \mathbb{R}^{n_g}$
- **Constraints**
  - Bounds $z^\flat \leq z \leq z^\sharp$
  - Power-flow constraints, $\forall i = 1, \cdots, n_b$,

$$p^i_{inj} = v_i \sum_j v_j (g_{ij} \cos(\theta_i - \theta_j) + b_{ij} \sin(\theta_i - \theta_j)),$$

$$q^i_{inj} = v_i \sum_j v_j (g_{ij} \sin(\theta_i - \theta_j) - b_{ij} \cos(\theta_i - \theta_j)).$$

Formulating the OPF as a non-linear problem

We adopt the *polar formulation*

$$\min_{z} F(z)$$

$$z = (\boldsymbol{v}, \boldsymbol{\theta}, \boldsymbol{p}_g, \boldsymbol{q}_g)$$

subject to (OPF)

$$z^\flat \leq z \leq z^\sharp$$

$$G(z) = 0$$

$$H(z) \leq 0$$

- **Objective**
  - <u>Minimize</u> costs of power generation

$$F(z) = \sum_{g=1}^{n_g} c_2^g (p_g)^2 + c_1^g p_g$$

- **Variables** $z = (\boldsymbol{v}, \boldsymbol{\theta}, \boldsymbol{p}_g, \boldsymbol{q}_g) \in \mathbb{R}^{2\times(n_b+n_g)}$
  - Voltage magnitude $\boldsymbol{v} \in \mathbb{R}^{n_b}$
  - Voltage angle $\boldsymbol{\theta} \in \mathbb{R}^{n_b}$
  - Active power generation $\boldsymbol{p}_g \in \mathbb{R}^{n_g}$
  - Reactive power generation $\boldsymbol{q}_g \in \mathbb{R}^{n_g}$
- **Constraints**
  - Bounds $z^\flat \leq z \leq z^\sharp$
  - Power-flow constraints, $\forall i = 1, \cdots, n_b$,

$$p^i_{inj} = v_i \sum_j v_j (g_{ij} \cos(\theta_i - \theta_j) + b_{ij} \sin(\theta_i - \theta_j)),$$

$$q^i_{inj} = v_i \sum_j v_j (g_{ij} \sin(\theta_i - \theta_j) - b_{ij} \cos(\theta_i - \theta_j)).$$

  - Line-flow constraints: $(|S_f|^2, |S_t|^2) \leq S^2_{max}$

Projecting the problem into the powerflow manifold

We can solve the powerflow $G(\boldsymbol{z}) = 0$ on the GPU (c.f. Adrian's talk), by

- splitting PQ buses apart from PV and slack buses
- defining the state $\boldsymbol{x} = (\boldsymbol{\theta}^{pv}, \boldsymbol{\theta}^{pq}, \boldsymbol{v}^{pq})$, and the control $\boldsymbol{u} = (\boldsymbol{v}^{ref}, \boldsymbol{v}^{pv}, \boldsymbol{p}_g^{pv})$
  Powerflow rewrites as $G(\boldsymbol{x}, \boldsymbol{u}) = 0$
- If $\nabla_x G$ is *non-singular* at $\boldsymbol{u}$, there exists a function $\tilde{x} : \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$ such that $G(\tilde{x}(\boldsymbol{u}), \boldsymbol{u}) = 0$ in a neighborhood of $\boldsymbol{u}$ (Implicit function theorem)

Projecting the problem into the powerflow manifold

We can solve the powerflow $G(\boldsymbol{z}) = 0$ on the GPU (c.f. Adrian's talk), by

- splitting PQ buses apart from PV and slack buses
- defining the state $\boldsymbol{x} = (\boldsymbol{\theta}^{pv}, \boldsymbol{\theta}^{pq}, \boldsymbol{v}^{pq})$, and the control $\boldsymbol{u} = (\boldsymbol{v}^{ref}, \boldsymbol{v}^{pv}, \boldsymbol{p}_g^{pv})$
  Powerflow rewrites as $G(\boldsymbol{x}, \boldsymbol{u}) = 0$
- If $\nabla_x G$ is *non-singular* at $\boldsymbol{u}$, there exists a function $\tilde{x} : \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$ such that $G(\tilde{x}(\boldsymbol{u}), \boldsymbol{u}) = 0$ in a neighborhood of $\boldsymbol{u}$ (Implicit function theorem)

Reduced problem

Let $f(\boldsymbol{u}) := F(\tilde{x}(\boldsymbol{u}), \boldsymbol{u})$ and $h(\boldsymbol{u}) := H(\tilde{x}(\boldsymbol{u}), \boldsymbol{u})$. Problem (OPF) is equivalent to

$$\min_{\boldsymbol{u}^\flat \leq \boldsymbol{u} \leq \boldsymbol{u}^\sharp} f(\boldsymbol{u}) \quad \text{s.t.} \quad \begin{cases} \boldsymbol{x}^\flat \leq x(\boldsymbol{u}) \leq \boldsymbol{x}^\sharp \\ h(\boldsymbol{u}) \leq 0 \end{cases} \tag{ROPF}$$

- Dimension of (ROPF) is $n_u = n_{ref} + 2n_{pv}$ (for (OPF): $2 \times (n_g + n_b)$)
- (ROPF) encompasses only *operational constraints*:
  the physical constraints $G(\boldsymbol{x}, \boldsymbol{u}) = 0$ are *implicitly* satisfied
- (ROPF) requires to solve the powerflow $G(\boldsymbol{x}, \boldsymbol{u}) = 0$ each time a new $\boldsymbol{u}$ is passed
- In practice, $G(\boldsymbol{x}, \boldsymbol{u}) = 0$ is solved using a Newton-Raphson algorithm, directly on the GPU

Computing the reduced gradient with the adjoint method

Reduced gradient

Let $F : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}$ a differentiable function depending both on $\boldsymbol{x}$ and $\boldsymbol{u}$
The function $f(\boldsymbol{u}) := F(\tilde{x}(\boldsymbol{u}), \boldsymbol{u})$ is *differentiable*, and

$$\nabla f(\boldsymbol{u}) = \underbrace{\nabla_u F}_{n_u} + \underbrace{(\nabla_u G)}_{n_x \times n_u}^\top \underbrace{\boldsymbol{\lambda}}_{n_x} \quad \text{with} \quad \underbrace{(\nabla_x G)}_{n_x \times n_x}^\top \boldsymbol{\lambda} = -\underbrace{\nabla_x F}_{n_x}$$

$\boldsymbol{\lambda} \in \mathbb{R}^{n_x}$ is the first-order adjoint

To evaluate ∇f, we need

- the evaluation of two sparse Jacobians $(\nabla_x G, \nabla_u G)$
  (forward mode autodiff on GPU)
- the resolution of *one* sparse linear system, with dimension $n_x \times n_x$
  (Direct QR or BICGSTAB)

Reduced Hessian: dense, dense, dense!

Can we extract second-order information as well? Yes!

- The first-order counterpart of the powerflow equation $G(\boldsymbol{x}, \boldsymbol{u}) = 0$ is

$$\widehat{G}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\lambda}) = \nabla_x F(\boldsymbol{x}, \boldsymbol{u}) + \nabla_x G(\boldsymbol{x}, \boldsymbol{u})^\top \boldsymbol{\lambda} = 0$$

- We derive two first-order adjoints ψ and $\boldsymbol{z}$, using the *adjoint-adjoint* method (Wang et al., 1992)

Reduced Hessian: dense, dense, dense!

Can we extract second-order information as well? Yes!

- The first-order counterpart of the powerflow equation $G(\boldsymbol{x}, \boldsymbol{u}) = 0$ is

$$\widehat{G}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\lambda}) = \nabla_x F(\boldsymbol{x}, \boldsymbol{u}) + \nabla_x G(\boldsymbol{x}, \boldsymbol{u})^\top \boldsymbol{\lambda} = 0$$

- We derive two first-order adjoints ψ and $\boldsymbol{z}$, using the *adjoint-adjoint* method (Wang et al., 1992)

Reduced Hessian

Let $\boldsymbol{w} \in \mathbb{R}^{n_u}$ be a vector. The Hessian-vector product $(\nabla^2 f)\boldsymbol{w}$ is equal to

$$(\nabla^2 f)\boldsymbol{w} = (\nabla^2_{\boldsymbol{uu}} F)\,\boldsymbol{w} + \boldsymbol{\lambda}^\top (\nabla^2_{\boldsymbol{uu}} G)\,\boldsymbol{w} + (\nabla_{\boldsymbol{u}} G)^\top \psi + (\nabla^2_{\boldsymbol{ux}} F)^\top \boldsymbol{z} + \boldsymbol{\lambda}^\top (\nabla^2_{\boldsymbol{ux}} G)^\top \boldsymbol{z}$$

with

$$\begin{cases} (\nabla_{\boldsymbol{x}} G) \quad \boldsymbol{z} = -(\nabla_u G)\ \boldsymbol{w} \\ (\nabla_{\boldsymbol{x}} G)^\top \psi = -(\nabla_u \widehat{G})\boldsymbol{w} - (\nabla_{\boldsymbol{x}} \widehat{G})\boldsymbol{z} \,, \end{cases}$$

- Require the resolution of $2n_u + 1$ linear systems to compute reduced Hessian $\nabla^2 f$
- Involve only *Hessian-vector products*!
- Reduced Hessian $\nabla^2 f$ is *dense*, with dimension $n_u \times n_u$

Augmented Lagrangian formulation

Original reduced problem

$$\min_{\boldsymbol{u}^\flat \leq \boldsymbol{u} \leq \boldsymbol{u}^\sharp} \quad f(\boldsymbol{u}) \quad \text{s.t.} \;\; h(\boldsymbol{u}) \leq 0$$

- 1 objective, $m = 2n_l + n_x$ constraints
- Computing the reduced gradient ∇f and reduced Jacobian ∇h: $1 + m$ adjoint solves
- Computing the reduced Hessian $\nabla^2 f$ and reduced Hessian $y^\top \nabla^2 h$: $(2n_u + 1)(m + 1)$ adjoint solves

Augmented Lagrangian formulation

Original reduced problem

$$\min_{u^\flat \le u \le u^\sharp} f(\boldsymbol{u}) \quad \text{s.t.} \;\; h(\boldsymbol{u}) \le 0$$

- 1 objective, $m = 2n_l + n_x$ constraints
- Computing the reduced gradient ∇f and reduced Jacobian ∇h: $1 + m$ adjoint solves
- Computing the reduced Hessian $\nabla^2 f$ and reduced Hessian $y^\top \nabla^2 h$: $(2n_u + 1)(m + 1)$ adjoint solves

Augmented Lagrangian

$$\min_{u^\flat \le u \le u^\sharp} f(\boldsymbol{u}) + \boldsymbol{y}^\top (h(\boldsymbol{u}) - \boldsymbol{s}) + \frac{\rho}{2} \|h(\boldsymbol{u}) - \boldsymbol{s}\|^2$$

$$\text{s.t.} \;\; \boldsymbol{s} \le 0$$

- 1 objective, only box constraints
- Computing the gradient involves only *transpose-Jacobian vector product* in the full-space and 1 adjoint solve
- Reduced Hessian computed with $2n_u + m + 1$ adjoint solves

Implementation

We have implemented the reduced space method in Julia

https://github.com/exanauts/ExaPF.jl

using the excellent CUDA.jl (Besard et al., 2018)

Powerflow $G(\boldsymbol{x}, \boldsymbol{u}) = 0$

- Newton-Raphson algorithm, implemented fully on the GPU
- Inversion of Newton-Step $(\nabla_x G_k)\boldsymbol{d}_k = -\boldsymbol{G}_k$ using either
  - Sparse QR (`CUSOLVER`)
  - Iterative BICGSTAB with `Krylov.jl` (Montoison et al., 2020)
- AutoDiff implemented with `ForwardDiff.jl` (runs on GPU thanks to (Revels et al., 2018))

Optimal powerflow in the reduced-space (ROPF)

- Augmented Lagrangian algorithm, following Conn et al. (1991); Arreckx et al. (2016)
- Subproblems solved either with:
  - Trust-region conjugate gradient (Tron)
  - Interior-point, using the inertia-free solver MadNLP (Shin et al., 2020) (`https://github.com/sshin23/MadNLP.jl`)
- Factorization of dense KKT matrix deported on the GPU, using Lapack-CUDA.

Results: two take-aways

1. **Inner iterations**

   10x speed-up when factorizing the (dense) Hessian matrix on the GPU

Opt. Solver	Linear Algebra	#it	linear solver (s)	callbacks (s)
MadNLP	Lapack (CPU)	62	1946.	2705.
MadNLP	Lapack (GPU)	62	195.	2688.

Table: We compare the time to solve *one* AugLag subproblem for case9241pegase (Hessian with dimension $12,131 \times 12,131$)

Results: two take-aways

1. **Inner iterations**
   10x speed-up when factorizing the (dense) Hessian matrix on the GPU

Opt. Solver	Linear Algebra	#it	linear solver (s)	callbacks (s)
MadNLP	Lapack (CPU)	62	1946.	2705.
MadNLP	Lapack (GPU)	62	195.	2688.

Table: We compare the time to solve *one* AugLag subproblem for case9241pegase (Hessian with dimension $12,131 \times 12,131$)

2. **Outer iterations**
   Augmented Lagrangian algorithm is not (yet?) competitive with full-space IPM

Case	# outer it	# Hess. eval	tot. time (s)	time / Hessian
case118ieee	10	271	3.0	0.011
case300ieee	7	167	6.7	0.040
case1354pegase	20♭	666	334.4	0.50

Table: Resolution time of (ROPF) with AugLag, using MadNLP+LapackGPU for the subproblems
Time to evaluate one Hessian $\approx O(n_u^2)$

Conclusion

- **Achievements**
  - We have revisited the reduced gradient method of Dommel and Tinney, with second-order
  - We have developed a custom Augmented Lagrangian algorithm

- **Perspective**
  At the moment, only the computation of the Newton step is deported on the GPU
  $\rightarrow$ TODO: Move all the algorithm on the GPU
  - Move the evaluation of the reduced Hessian fully on the GPU, with AD
  - Adapt the Augmented Lagrangian to GPU architectures

Reduced space's wager:
Would you bet 10$ on reduced space/GPU, versus full space/CPU?

Slides available at: https://frapac.github.io/pdf/SIAM_CSE21.pdf

References I

Arreckx, S., Lambe, A., Martins, J. R., and Orban, D. (2016). A matrix-free augmented lagrangian algorithm with application to large-scale structural design optimization. Optimization and Engineering, 17(2):359–384.

Besard, T., Foket, C., and De Sutter, B. (2018). Effective extensible programming: unleashing julia on gpus. IEEE Transactions on Parallel and Distributed Systems, 30(4):827–841.

Burchett, R., Happ, H., and Wirgau, K. (1982). Large scale optimal power flow. IEEE Transactions on Power Apparatus and Systems, (10):3722–3732.

Conn, A. R., Gould, N. I., and Toint, P. (1991). A globally convergent augmented lagrangian algorithm for optimization with general constraints and simple bounds. SIAM Journal on Numerical Analysis, 28(2):545–572.

Dommel, H. and Tinney, W. (1968). Optimal Power Flow Solutions. IEEE Transactions on Power Apparatus and Systems, PAS-87(10):1866–1876.

Granville, S. (1994). Optimal reactive dispatch through interior point methods. IEEE Transactions on power systems, 9(1):136–146.

Montoison, A., Orban, D., and contributors (2020). Krylov.jl: A Julia basket of hand-picked Krylov methods. https://github.com/JuliaSmoothOptimizers/Krylov.jl.

Peschon, J., Bree, D. W., and Hajdu, L. P. (1972). Optimal power-flow solutions for power system planning. Proceedings of the IEEE, 60(1):64–70.

Revels, J., Besard, T., Churavy, V., De Sutter, B., and Vielma, J. P. (2018). Dynamic automatic differentiation of gpu broadcast kernels. arXiv preprint arXiv:1810.08297.

Shin, S., Coffrin, C., Sundar, K., and Zavala, V. M. (2020). Graph-based modeling and decomposition of energy infrastructures. arXiv preprint arXiv:2010.02404.

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.

Wang, Z., Navon, I. M., Le Dimet, F.-X., and Zou, X. (1992). The second order adjoint analysis: theory and applications. Meteorology and atmospheric physics, 50(1):3–20.