ETI Measurement Locations

Craig Bakker and Rohan Punamiya

January 5, 2025

1 Introduction

1.1 Fisher Information Derivation

Let us assume that

- we have measurements at points i coming from sources at locations j
- we have a radiation kernel A_{ij} mapping from j to i
- the signal measured at point i from any given source of strength s_j at point j follows a Poisson distribution with mean $A_{ij}s_j$

It then follows that

- the random variable M_i corresponding to the measurement at point i is Poisson distributed with a mean of $\sum_j A_{ij}s_j$
- the Fisher information matrix corresponding to those measurements is

$$\mathcal{I}_{kl}(K) = \sum_{i \in K} \frac{A_{ik}A_{il}}{\sum_j A_{ij}s_j} \tag{1}$$

The Fisher information matrix corresponding to a single measurement is

$$\begin{aligned} -E\left[\frac{\partial^2}{\partial s_k \partial s_l} f(m_i; s)\right] &= -E\left[-m_i \frac{1}{\left(\sum_j A_{ij}s_j\right)^2} A_{ik}A_{il}\right] \\ &= \frac{1}{\left(\sum_j A_{ij}s_j\right)^2} A_{ik}A_{il}E[m_i] \\ &= \frac{1}{\sum_j A_{ij}s_j} A_{ik}A_{il} \end{aligned} \tag{2}$$

The Fisher information matrix of independent variables is the sum of the Fisher information matrices of the separate variables, and for fixed (deterministic) s_j values, the measurements at different locations are independent.

Note that this matrix will always be symmetric and positive semi-definite (and therefore normal).

1.2 Optimization Formulation

The goal is to choose a subset of measurements points $K \subset \{1, \ldots, N\}, |K| = n < N$ (where N is the number of sources) to maximize the amount of information provided by measurements at those points about the sources s_j (which essentially parameterize the distribution of measurements). This becomes something like

$$\max_K \operatorname{rank}\left(\mathcal{I}\left(K\right)\right) \tag{3}$$

which we can approximate using the nuclear norm $\left\|\mathcal{I}\left(K\right)\right\|_*$ or the stable rank:

$$\frac{\left\|\mathcal{I}\left(K\right)\right\|_F^2}{\left\|\mathcal{I}\left(K\right)\right\|_2^2} \tag{4}$$

I would probably recommend the stable rank, as it encourages the optimization to find $\mathcal{I}$ with approximately equal singular values (as opposed to having one large singular value and many smaller ones). Furthermore, the numerator is easy to calculate while the denominator is easy to approximate (e.g., via a power method, which will benefit from the normality of $\mathcal{I}\left(K\right)$).

Unfortunately, we do not know the values of s_j. For the purpose of the optimization calculations, it might make sense to set $s_j = 1 \; \forall j$ to consider the possibility of a source at any location. In principle, if we were able to set a prior distribution on the s_j values, We could calculate an expectation on $\mathcal{I}\left(K\right)$ based on that prior distribution. Alternatively, with that information, we could use a Bayesian approach or try to maximize the covariance matrix of M; however, without distributions on s_j, the covariance matrix of M is diagonal.

This is essentially a combinatorial optimization problem, so if we are using non-gradient-based optimization methods, we could just use matrix rank directly, but it may be useful to distinguish between matrices with similar singular values and ones with widely-varying singular values. If we instead had a continuous kernel $A_j\left(x\right)$, and assuming that we are using the stable rank objective, this would become a continuous optimization problem:

$$\max_{x_i, i=1,\ldots K} \frac{\left\|\mathcal{I}\left(x\right)\right\|_F^2}{\left\|\mathcal{I}\left(x\right)\right\|_2^2} \tag{5}$$

$$\mathcal{I}_{kl}\left(x\right) = \sum_i \frac{A_k\left(x_i\right) A_l\left(x_i\right)}{\sum_j A_j\left(x_i\right) s_j} \tag{6}$$

Let us assume that we have an ordering $R\left(K\right)$ on the set K, indicating the order in which the points are traversed, and let us further assume that we can calculate the travel cost associated with that ordering $\mathcal{C}\left(R\left(K\right)\right)$. Then what we have is a multi-objective optimization problem:

$$\max_K \frac{\left\|\mathcal{I}\left(K\right)\right\|_F^2}{\left\|\mathcal{I}\left(K\right)\right\|_2^2} \tag{7}$$

$$\min_{K,R} \mathcal{C}\left(R\left(K\right)\right) \tag{8}$$

$$\mathcal{I}_{kl}\left(K\right) = \sum_{i \in K} \frac{A_{ik} A_{il}}{\sum_j A_{ij} s_j} \tag{9}$$

In principle, this could be solved as a nested optimization problem (with the travel cost on the inner loop), but given the combinatorial nature of both $R\left(K\right)$ and K, it might be more efficient to solve the problem in its natural bi-objective form using a population-based metaheuristic optimization.

1.3 Single-Objective Integer Programming Problem

Let us formulate the measurement point selection problem using binary variables w_i that indicate whether measurement point i is selected, and let us ignore the travel cost objective:

$$\max_{w_i} \frac{\|\mathcal{I}(K)\|_F^2}{\|\mathcal{I}\|_2^2} \tag{10}$$

$$\mathcal{I}_{kl} = \sum_i w_i \frac{A_{ik}A_{il}}{\sum_j A_{ij}s_j} \tag{11}$$

$$\sum_i w_i = n \tag{12}$$

We can turn this into a continuous optimization problem using a few approximations or expansions:

$$\|\mathcal{I}\|_F^2 = \sum_{k,l} \left(\sum_i w_i \frac{A_{ik}A_{il}}{\sum_j A_{ij}s_j} \right)^2 \tag{13}$$

$$\|\mathcal{I}\|_2^2 \approx \frac{\left\|\mathcal{I}^{k+1}\xi\right\|_2^2}{\left\|\mathcal{I}^k\xi\right\|_2^2} \tag{14}$$

where ξ is a random vector, and k is chosen to get a sufficient level of convergence on the power method. If a different ξ vector is chosen at each iteration of the optimization, this may slow down convergence, but it will also help to make the optimization more robust to the approximation. We also make w_i continuous but add a penalty function $\|w\|_1 = \sum_i w_i$ – this encourages the optimization to force entries to 0. The optimization then becomes

$$\max_{w_i} \frac{\left\|\mathcal{I}^k\xi\right\|_2^2}{\left\|\mathcal{I}^{k+1}\xi\right\|_2^2} \sum_{k,l} \left(\sum_i w_i \frac{A_{ik}A_{il}}{\sum_j A_{ij}s_j} \right)^2 - \rho \sum_i w_i \tag{15}$$

$$\mathcal{I}_{kl} = \sum_i w_i \frac{A_{ik}A_{il}}{\sum_j A_{ij}s_j} \tag{16}$$

$$\sum_i w_i = n \tag{17}$$

$$w_i \leq 1 \tag{18}$$

$$w_i \geq 0 \tag{19}$$

Alternatively, if we just want to plug the integer programming problem directly into, say, a branch-and-bound method, we would have

$$\max_{w_i} \frac{\left\|\mathcal{I}^k\xi\right\|_2^2}{\left\|\mathcal{I}^{k+1}\xi\right\|_2^2} \sum_{k,l} \left(\sum_i w_i \frac{A_{ik}A_{il}}{\sum_j A_{ij}s_j} \right)^2 \tag{20}$$

$$\mathcal{I}_{kl} = \sum_i w_i \frac{A_{ik}A_{il}}{\sum_j A_{ij}s_j} \tag{21}$$

$$\sum_i w_i = n \tag{22}$$

$$w_i \in [0, 1] \tag{23}$$

It is also possible to simplify down the matrix powers of $\mathcal{I}$ by using the form provided above:

$$\mathcal{I}_{kl} = \sum_i \frac{w_i}{\sigma_i} A_{ik} A_{il} \tag{24}$$

$$\sigma_i = \sum_j A_{ij} s_j \tag{25}$$

$$\left[\mathcal{I}^2\right]_{kl} = \sum_{i,j} \frac{w_i w_j}{\sigma_i \sigma_j} A_{ik} \left[\sum_m A_{im} A_{jm}\right] A_{jl} \tag{26}$$

$$B_{ij} = \sum_m A_{im} A_{jm} \tag{27}$$

$$\left[\mathcal{I}^2\right]_{kl} = \sum_{i,j} \frac{w_i w_j}{\sigma_i \sigma_j} A_{ik} B_{ij} A_{jl} \tag{28}$$

$$\begin{aligned} \mathcal{I}^2 xi &= \sum_l \left[\mathcal{I}^2\right]_{kl} \xi_l \\ &= \sum_{i,j} \frac{w_i w_j}{\sigma_i \sigma_j} A_{ik} B_{ij} \sum_l A_{jl} \xi_l \end{aligned} \tag{29}$$

$$\begin{aligned} \left[\mathcal{I}^3\right]_{kl} &= \sum_{i,j,m} \frac{w_i w_j w_m}{\sigma_i \sigma_j \sigma_m} A_{ik} B_{ij} B_{jm} A_{mp} \\ &= \sum_{i,m} \frac{w_i w_m}{\sigma_i \sigma_m} A_{ik} \left[\sum_j \frac{w_j}{\sigma_j} B_{ij} B_{jm}\right] A_{ml} \end{aligned} \tag{30}$$

$$\begin{aligned} \mathcal{I}^3 xi &= \sum_l \left[\mathcal{I}^2\right]_{kl} \xi_l \\ &= \sum_l \left[\mathcal{I}^3\right]_{kl} \xi_l \\ &= \sum_{i,m} \frac{w_i w_m}{\sigma_i \sigma_m} A_{ik} \left[\sum_j \frac{w_j}{\sigma_j} B_{ij} B_{jm}\right] \sum_l A_{ml} \xi_l \end{aligned} \tag{31}$$

$$\left[\mathcal{I}^4\right]_{kl} = \sum_{i,p} \frac{w_i w_p}{\sigma_i \sigma_p} A_{ik} \left[\sum_{j,m} \frac{w_j w_m}{\sigma_j \sigma_m} B_{ij} B_{jm} B_{mp}\right] A_{pl} \tag{32}$$

$$\begin{aligned} \mathcal{I}^4 xi &= \sum_l \left[\mathcal{I}^2\right]_{kl} \xi_l \\ &= \sum_l \left[\mathcal{I}^4\right]_{kl} \xi \\ &= \sum_{i,p} \frac{w_i w_p}{\sigma_i \sigma_p} A_{ik} \left[\sum_{j,m} \frac{w_j w_m}{\sigma_j \sigma_m} B_{ij} B_{jm} B_{mp}\right] A_{pl} \end{aligned} \tag{33}$$

This should significantly reduce the computational cost, as $\mathcal{I} \in \Re^{N \times N}$, whereas $B \in \Re^{n \times n}$, $n \ll N$.

1.4 Other Options

Let $\tilde{A}(K)$ be the matrix constructed from the rows of A corresponding to the elements of K.

- Condition number of $\tilde{A}(K)$, $\kappa\left(\tilde{A}(K)\right)$.
  - The goal is to minimize $\kappa\left(\tilde{A}(K)\right)$.

- In the pathological case where $|K| = 1$, $\kappa\left(\tilde{A}(K)\right) = 1$ for all K, which is not helpful. Since $\kappa\left(\tilde{A}(K)\right) \geq 1\ \forall K$, this means that a single measurement point is globally optimal (from this perspective).
- $\kappa\left(\tilde{A}(K)\right) = \infty$ if the rows of $\kappa\left(\tilde{A}(K)\right)$ are not linearly independent, which is something we would want.

- Unobservability number of $\tilde{A}(K)$, $1/\sigma_{min}\left(\tilde{A}(K)\right)$
  - The goal is to minimize $1/\sigma_{min}\left(\tilde{A}(K)\right)$.
  - If $A \rightarrow cA, c > 1$, then $1/\sigma_{min}\left(\tilde{A}(K)\right)$ would decrease for all K, which may not be something we want.
  - In the pathological case where $|K| = 1$, a row with one entry of $c + \epsilon$ and the rest being zeros would produce $1/\sigma_{min}\left(\tilde{A}(K)\right) = 1/(c + \epsilon)$, whereas a row with all entries being $c/\sqrt{N}$ would produce $1/\sigma_{min}\left(\tilde{A}(K)\right) = 1/c$.

- Minimal observability of $\tilde{A}(K)$, $o_{min}\left(\tilde{A}(K)\right) = \min_j \sum_j \tilde{A}(K)_{ij}$.
  - The goal is to maximize $o_{min}\left(\tilde{A}(K)\right)$
  - This metric will be identically 0 for any $\tilde{A}(K)$ with zero columns, so it may be flat for large parts of the K space – it will not be able to distinguish between K and K' if both have a zero column.

- Stable Rank of $\tilde{A}(K)$
  - This is going to be very similar to the stable rank of the Fisher information matrix. If $\tilde{A}(K)$ has singular values σ_i, then the stable rank is

$$\frac{\sum_i \sigma_i^2}{\max_i \sigma_i^2} \tag{34}$$

  - The Fisher information matrix is very similar to $\tilde{A}^T(K)\tilde{A}(K)$, and the stable rank of $\tilde{A}^T(K)\tilde{A}(K)$ is

$$\frac{\sum_i \sigma_i^4}{\max_i \sigma_i^4} \tag{35}$$

    because this matrix is symmetric and positive definite, so its singular values are equal to its eigenvalues, and its eigenvalues λ_i are equal to σ_i^2. So this will likely produce similar results to the stable rank of the Fisher information matrix.

- Quasi-Determinant of $\tilde{A}(K)$ - $\prod_i \sigma_i$
  - Multiplying a vector by a matrix is analogous to taking a hypersphere and mapping it to a hyperellipsoid with axes that have lengths corresponding to the singular values of the matrix. The volume of that hyperellipsoid is proportional to the product of the singular values. In a square matrix, the determinant is equal to the product of the eigenvalues.
  - This means that any $\tilde{A}(K)$ with less than full rank will produce a value of 0.
  - This metric will generally encourage $\tilde{A}(K)$ matrices with approximately equal σ_i values over $\tilde{A}(K)$ matrices with σ_i values that differ widely (as the stable rank approaches do) but in a different way – perhaps more strongly.

2 Simultaneous Optimization – Updated Formulation with Stable Rank

The basic optimization problem is

$$\max_{K,N} \frac{\|A(K)\|_F^2}{\|A(K)\|_2^2} \tag{36}$$

$$K \subset \{1, \ldots, n\} \tag{37}$$

$$|K| = N < n \tag{38}$$

where there are n total possible measurement points and $A(K)$ is the submatrix of A formed by selecting rows $i \in K$.

2.1 Option 1: Continuous Relaxation with Branch-and-Bound and Matrix Norm Approximation

$$A(K) = \text{diag}(x) A \tag{39}$$

$$\max_{x \in \Re^n} \frac{\|\text{diag}(x) A\|_F^2}{\sigma_{max}^2} \tag{40}$$

$$\sigma_{max}^2 = \frac{\|\text{diag}(x) A\xi\|_2^2}{\|\xi\|^2} \tag{41}$$

$$N \geq \sum_i x_i \tag{42}$$

$$0 \leq x_i \leq 1 \tag{43}$$

In this case, the x variables are a continuous relaxation of the binary measurement point selection, ξ are random variables, and we approximate $\|\text{diag}(x) A\|$ using a power method. There might be some computational motivation to using the logarithm of this objective.

2.2 Option 2: Discrete Metaheuristic Optimization

Keep everything discrete and evaluate $\|A(K)\|$ exactly (i.e., not via the power method approximation). Represent the measurement points as nodes on a graph such that geographically neighboring points have edges connecting them. That way, when a particular measurement point is perturbed, it is perturbed to a proximate point. This kind of population based approach can then be used very naturally for identifying the Pareto front in a multi-objective context.

$$\max_{x \in \Re^n} \frac{\|\text{diag}(x) A\|_F^2}{\|\text{diag}(x) A\|_2^2} \tag{44}$$

$$N = \sum_i x_i \tag{45}$$

$$x \in \{0, 1\} \tag{46}$$

The python package `pymoo` (`https://pymoo.org/index.html`) might be suitable for this, as it has a wide range of metaheuristic optimization algorithms to choose from (e.g., NSGA-II, which is specifically designed for multi-objective optimization problems), and it seems possible to customize the evolution process (e.g., crossover and mutation) for the agent population.

2.3 Option 3: Continuous Relaxation with MPEC solution approach

$$A(K) = \text{diag}(x) A \tag{47}$$

$$\max_{x \in \Re^n} \frac{\|\text{diag}(x) A\|_F^2}{\|\text{diag}(x) A\|_2^2} \tag{48}$$

$$N \geq \sum_i x_i \tag{49}$$

$$0 \leq x_i \perp 1 - x_i \geq 0 \tag{50}$$

In this case, the x variables are a continuous relaxation of the binary measurement point selection, and their binary status is enforced by the complementarity constraints

$$0 \leq x_i \perp 1 - x_i \geq 0 \tag{51}$$

MPECs can be solved via successive NLP relaxations, and there are built-in methods for doing this in pyomo: `https://www.osti.gov/servlets/purl/1195764`. The documentation about how to actually use those built-in methods is not great. Basically, you set up the MPEC like a normal optimization (with complementarity constraints), you use the following commands

```
from pyomo.mpec import *
from pyomo.opt import SolverFactory
opt = SolverFactory('mpec_nlp',executable='ipopt.exe')
opt.options['epsilon_initial'] = 1e0
opt.options['epsilon_final'] = 1e-6
```

and solve use `results = opt.solve(mpec_temp)` where `mpec_temp` is the pyomo MPEC model. You can change the initial and final ϵ values as needed. Because this is a nonlinear optimization problem, it would probably be a good idea to solve the problem several different times from different initial conditions.

The other difficulty has to do with calculating $\|\text{diag}(x) A\|_2^2$. Because we are already solving an MPEC, we can incorporate the norm calculation as additional constraints on the problem. The norm calculation can be described as

$$\max_{\sigma, y} \sigma \tag{52}$$

$$0 = \left(B^T B - \sigma I\right) y \tag{53}$$

$$1 = y^T y \tag{54}$$

The KKT conditions are

$$0 = 1 - \lambda^T y \tag{55}$$

$$0 = \lambda^T \left(B^T B - \sigma I\right) + 2\sigma y \tag{56}$$

$$0 = \left(B^T B - \sigma I\right) y \tag{57}$$

$$1 = y^T y \tag{58}$$

which we can simplify to

$$0 = \sigma + \frac{1}{2} \lambda^T \left(B^T B - \sigma I\right) \lambda \tag{59}$$

$$0 = \sigma \left(B^T B - \sigma I\right)^2 \lambda \tag{60}$$

So we would have

$$\max_{x\in\Re^n,\sigma\in\Re,\lambda\in\Re^n} \frac{\|\mathrm{diag}\,(x)\,A\|_F^2}{\sigma^2} \tag{61}$$

$$N \geq \sum_i x_i \tag{62}$$

$$0 \leq x_i \leq 1 \tag{63}$$

$$0 \leq x_i \perp 1 - x_i \geq 0 \tag{64}$$

$$0 = \sigma + \frac{1}{2}\lambda^T \left(B^T B - \sigma I\right) \lambda \tag{65}$$

$$0 = \sigma \left(B^T B - \sigma I\right)^2 \lambda \tag{66}$$

$$B = \mathrm{diag}\,(x)\,A \tag{67}$$

The KKT conditions for the norm calculation add more nonlinearity to the problem, but the norm calculation *should* have a unique solution (at least with respect to σ), and thus the KKT conditions *should* be necessary and sufficient. It may be worthwhile to test this out, though.

Alternatively, we could use tensorflow and embed the norm calculation directly into the objective. Manually implementing a version of the NLP relaxation (in tensorflow) is sufficiently simple to make this a possibility. The difficulty here is that tensorflow does not explicitly handle constraints.

A third alternative is to use Julia, which can handle the matrix norm calculation in the objective and pass the necessary derivatives through to the optimizer. Manually implementing the successive NLP relaxations of the MPEC is then straightforward and easy to do.