# Meeting Update

## Gaussian Process Covariance Function Selection Using Minimum Energy Designs


Kristyn Pantoja

Department of Statistics
Texas AM University


21 March 2019

# Outline

# Last Time

# Last Meeting Recap

## What happened last time

1. Posterior probabilities & Bayes factors for linear models given by $H_0, H_1$ and fast SMED algorithm for linear model selection (i.e. algorithm inspired by 2nd paper, Joseph et. al. 2018).
2. Large k in One-At-a-Time SMED algorithm $\implies$ computational issues, gets stuck near edge with greatest distance between the two lines (as expected).
3. Issue: Fast SMED and One-At-a-Time SMED algorithms for linear model selection aren't similar in distribution of design points, perhaps optimization issue with fast SMED algorithm.
4. Started looking at asymptotic properties of SMED algorithms for linear model selection and semi-thinking about criterion for evaluating the design.
5. Started thinking about GP model selection.

# Why Fast SMED and One-At-a-Time SMED Aren't Similar

# The Issue, Potential Causes

The Fast SMED and One-At-a-Time SMED algorithms for linear model selection aren't similar in distribution of design points.

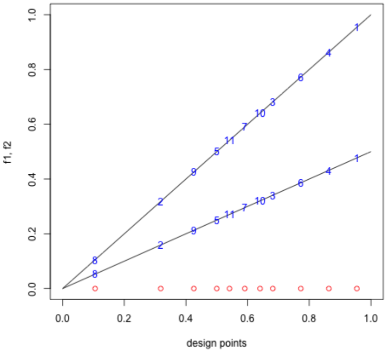
## Potential Causes

1. Optimization issue with fast SMED algorithm.
2. Candidate sets are not defined correctly.

# Compare Results, See Issue

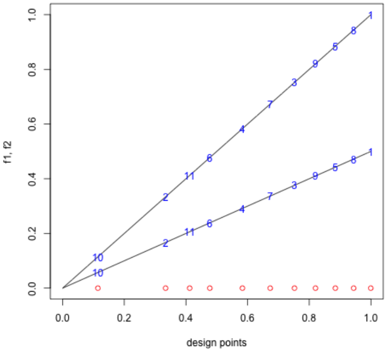
Compare this to the one-at-a-time algorithm (as in Joseph et. al. 2015) with 11 sequentially picked design points, 1000 candidate points, and a power of $k = 4$:

## Fast Algorithm




Points are more concentrated at middle of support.

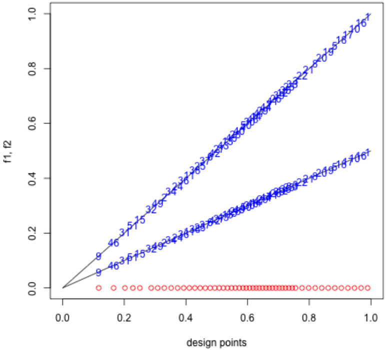
## One-at-a-Time Greedy Algorithm



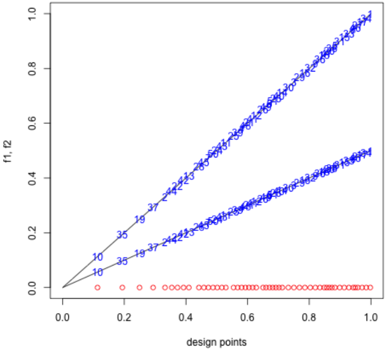
# Compare Results, See Issue, continued

What about for higher $N(= 51)$? Not quite the same...

## Fast Algorithm




## One-at-a-Time Greedy Algorithm



# The Issue, Potential Solutions

## Potential Solutions

1. For suspected optimization issue:
   - Try to do the design method in Fast Algorithm (Joseph et. al. 2018), but with optimization criterion from One-At-A-Time Algorithm (Joseph et. al. 2015): update each of the $N$ design points in designs $k = 1, \ldots, K$ by selecting:

$$\mathbf{x}_j^k = \arg\min_{\mathbf{x} \in \mathbf{C}_k^j} \sum_{i=1}^{j-1} \left\{ \frac{1}{f^{\gamma_k}(\mathbf{x}_i) f^{\gamma_k}(\mathbf{x}) d(\mathbf{x}_i, \mathbf{x})} \right\}^k \tag{1}$$

     instead of the usual

$$\mathbf{x}_j^k = \arg\min_{\mathbf{x} \in \mathbf{C}_k^j} \max_{i=1:(j-1)} \frac{1}{f^{\gamma_k}(\mathbf{x}_i) f^{\gamma_k}(\mathbf{x}) d(\mathbf{x}_i, \mathbf{x})} \tag{2}$$

   - Try with *log* of criterion, like in Joseph et. al. 2018 (even though doesn't seem beneficial with $f = \{Wasserstein\}^{1/(2p)}$, and no NaN problem so far).
   - Don't restrict $L_{jk}$ to be between 0 and 1 in space-filling $D_{k+1}^j$

# The Issue, Potential Solutions, continued...

## Potential Solutions

1. For suspected candidate sets issue:
   - Look at Dr. Raymond Wong's lecture notes for C/C++, try to understand the algorithm in MinEd R package, written in Rccp.
   - Clarify candidate sets: Says $KN$ evaluations of $f$, but we create $N$ new candidate points for each point at each design $k = 1, \ldots, K$, so shouldn't it be $KN^2$ evaluations of $f$?

# Trying Solution 2 to Potential Cause 2

## Potential Solutions

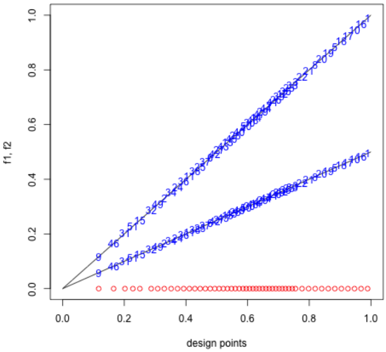
1. For suspected candidate sets issue:
   - Clarify candidate sets: Says $KN$ evaluations of $f$, but we create $N$ new candidate points for each point at each design $k = 1, \ldots, K$, so shouldn't it be $KN^2$ evaluations of $f$?

It turns out, I had a bug in my code that was causing the candidate sets for each $\mathbf{x}_j^{k+1}$ to be wrong. This, combined with my horrendous attempt to re-index $k$, led to the whole algorithm being wrong.

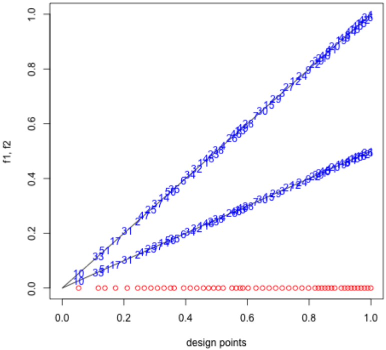
# Old (Wrong) Fast Algorithm vs. Fixed Fast Algorithm, $N = 51$

We can definitely see a difference! Although the "fixed" version seems somewhat choppy.

## Old (Wrong) Fast Algorithm



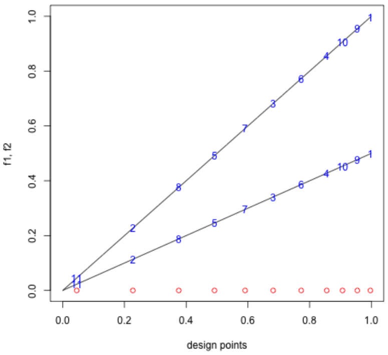

## Fixed Fast Algorithm



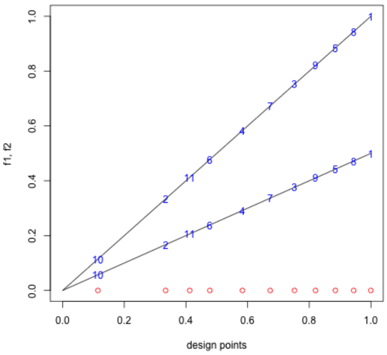
# Fixed Fast Algorithm vs. One-at-a-Time Algorithm, $N = 11$

The new fast algorithm seems to have less of a bisection pattern.

## Fixed Fast Algorithm



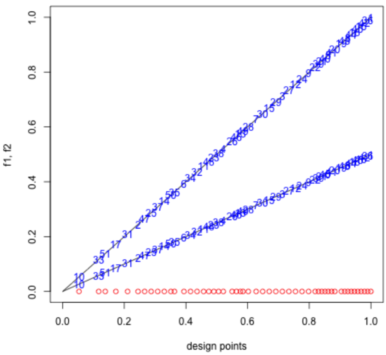

## One-at-a-Time Algorithm



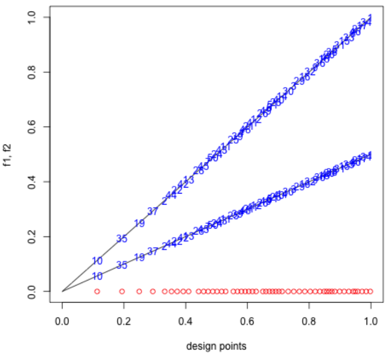
# Fixed Fast Algorithm vs. One-at-a-Time Algorithm, $N = 51$

Distribution of design points in the new fast algorithm is definitely closer to that of the one-at-a-time algorithm, compared to the wrong/old fast algorithm.

## Fixed Fast Algorithm



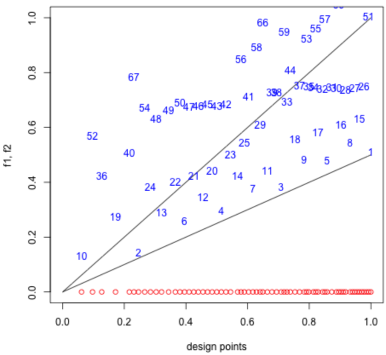

## One-at-a-Time Algorithm



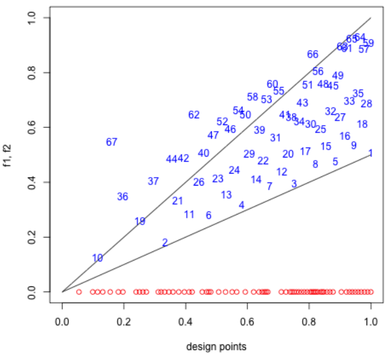
# Seeing Order of Design Points $N = 67$

It seems there is still the difference in order of design points, though.

## Fixed Fast Algorithm



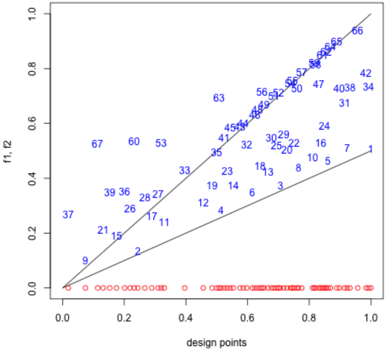

## One-at-a-Time Algorithm



# Seeing Order of Design Points $N = 67$

This is what happens when we draw the candidate points using a Uniform distribution (over $L_{jk}$), rather than using the Lattice function that was used in the 2nd paper (even though this is how candidates were picked in the 1st paper).

## Fixed Fast Algorithm




## One-at-a-Time Algorithm