Practical large-scale computer model calibration to real data

Robert B. Gramacy
The University of Chicago Booth School of Business

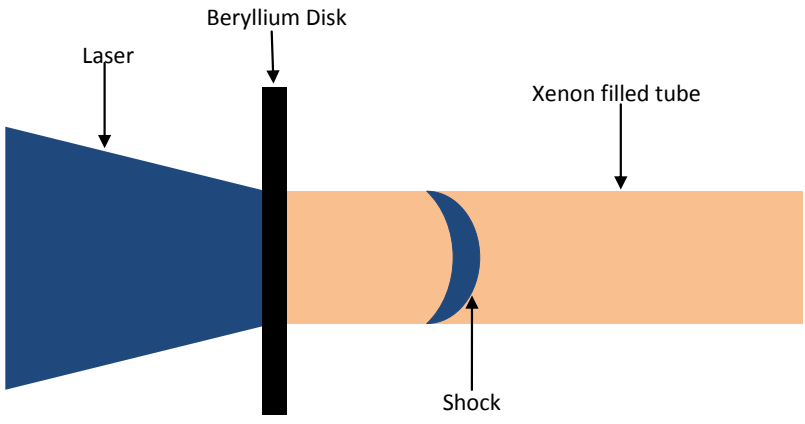
faculty.chicagobooth.edu/robert.gramacy

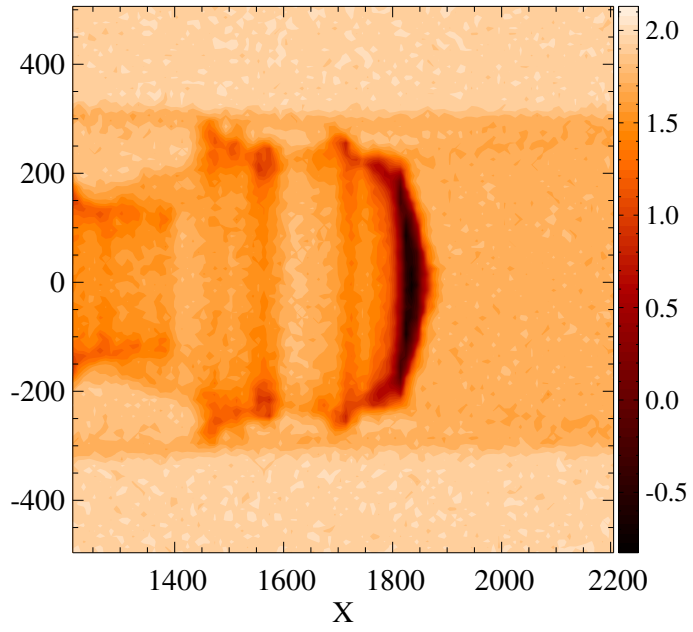
Joint with **Derek Bingham** – Simon Fraser

CASD, Washington DC — October 2014

Radiative Shock Experiment

A high-energy laser irradiates a Be disk at the front of a Xe-filled tube, launching a shock. (Boehy, et al., 1997)





- 9 design variables: describing energy, disk, tube
- response: distance the wave travels in a certain time

Design variables

Input	CE1	CE2	Field Design
Be thick (microns)	[18,22]	21	21
Xe fill press (atm)	[1.100,1.2032]	[0.852,1.46]	[1.032,1.311]
Time (nano-secs)	[5,27]	[5.5,27]	6-values in [13, 28]
Tube diam (microns)	575	[575,1150]	$\{575, 1150\}$
Taper len (microns)	500	[460,540]	500
Nozzle len (microns)	500	[400,600]	500
Aspect ratio (microns)	1	[1,2]	1
Laser energy (J)	[3600,3990]		[3750.0 3889.6]
Eff laser energy (J)		[2156.4,4060]	

Calibration parameters

Input	CE1	CE2	
Electron flux limiter	[0.04, 0.10]	0.06	
Energy scale-factor	[0.40,1.10]	[0.60,1.00]	

Relationship between design variables and output explored via

- a field experiment with 20 observations
- two computer experiments, 26458 runs combined, requiring two extra calibration or tuning parameters

Computer model calibration

The goal is to find a value of the calibration parameter(s), u, relating the possibly biased computer model $Y^M(x, u)$ to the noisy field data $Y^F(x)$.

Kennedy & O'Hagan (2001) modeled the *real*/physical process as $Y^R(x) = Y^M(x, u^*) + b(x)$, implying

$$Y_j^F(x) = Y^R(x) + \varepsilon_{xj}, \quad \varepsilon_{xj} \overset{\text{iid}}{\sim} \mathcal{N}(0, \sigma_\varepsilon^2), \quad j = 1, \ldots, n_x$$

giving $$Y_j^F(x) = Y^M(x, u^*) + b(x) + \varepsilon_{xj},$$

where all unknowns (u^*, Y_M, b, σ_ε) are inferred *jointly* given data from M and F, via Bayesian Monte Carlo inference.

KOH proposed GP priors for $Y^M(\cdot,\cdot)$ and $b(\cdot)$, but that's not going to work for our application.

- $N_M = 26458$ is too big for GP emulation of M;
- and that's compounded with learning u^* and $b(\cdot)$, ...

too much Monte Carlo.

We instead propose the following. (G, Bingham, et al., 2014)

- Modularize emulation: learn $Y_M(\cdot,\cdot)$ from runs of M only, ignoring Y^F. (Liu, et al., 2009)
- Obtain $\hat{Y}_M(\cdot, u)$, for each u via a local approximate GPs.
- Learn $\hat{u}$, by optimizing the integrated likelihood/posterior for $\hat{b}(\cdot)$ fit to residuals $\hat{Y}_M(X_F, u) - Y_F$.

Computer experiments and emulation

An emulator or surrogate model $\hat{Y}_M(x, u)$ is just a fancy regression fit to runs of the model.

So lets talk regression, generically for inputs x and outputs $Y(x)$, and say that $\hat{f}_N$ is just regression or response surface fit to input-output pairs $(x_1, y_1), \ldots, (x_N, y_N)$, where $y_i \sim f(x_i)$.

Gaussian process (GP) regressions are popular emulators.

- As predictors, they are rarely beaten out-of-sample,
- have appropriate coverage, can interpolate,
- and Gaussians offer a degree of analytic tractibility.

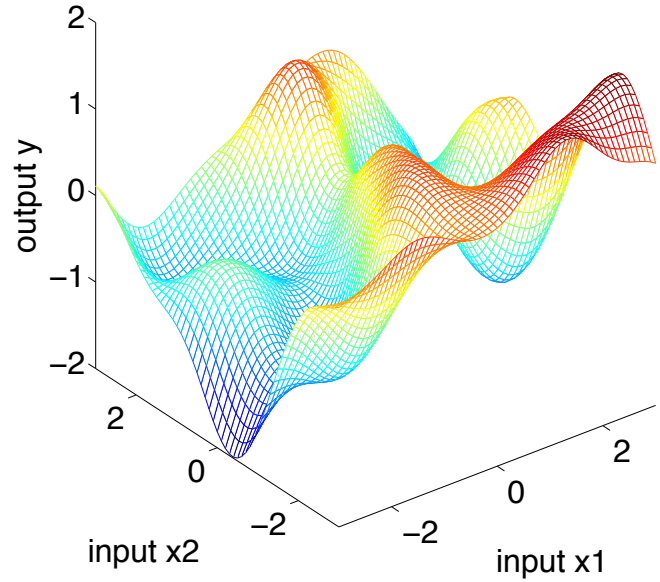
A GP is a prior over functions, $Y : \mathbb{R}^p \to \mathbb{R}$ where

- any finite collection of outputs are jointly Gaussian
- via mean $\mu(x) = \mathbb{E}\{Y(x)\} = 0$, and covariance

$$C(x, x') = \mathbb{E}\{[Y(x) - \mu(x)][Y(x') - \mu(x')]^\top\}$$
$$= \tau^2 K_\theta(x, x'), \text{ usually based on Euclidean distance.}$$

E.g., the isotropic Gaussian:

$$K_\theta(x, x') = \exp\left\{-\frac{||x - x'||^2}{\theta}\right\}$$



Inference

A regression perspective suggests a likelihood interpretation.

Using data $D = (X, Y)$, where X is an $N \times p$ design matrix, the $N \times 1$ response vector Y has MVN likelihood:

$$Y \sim \mathcal{N}_N(0, \tau^2 K), \quad \text{where} \quad K_{ij} = K(x_i, x_j).$$

When $\pi(\tau^2) \propto \tau^{-2}$ (Berger et al., 2001)

$$p(Y|K) = \frac{\Gamma[N/2]}{(2\pi)^{N/2}|K|^{1/2}} \times \left(\frac{\psi}{2}\right)^{-\frac{N}{2}} \quad \text{where} \quad \psi = Y^\top K^{-1} Y.$$

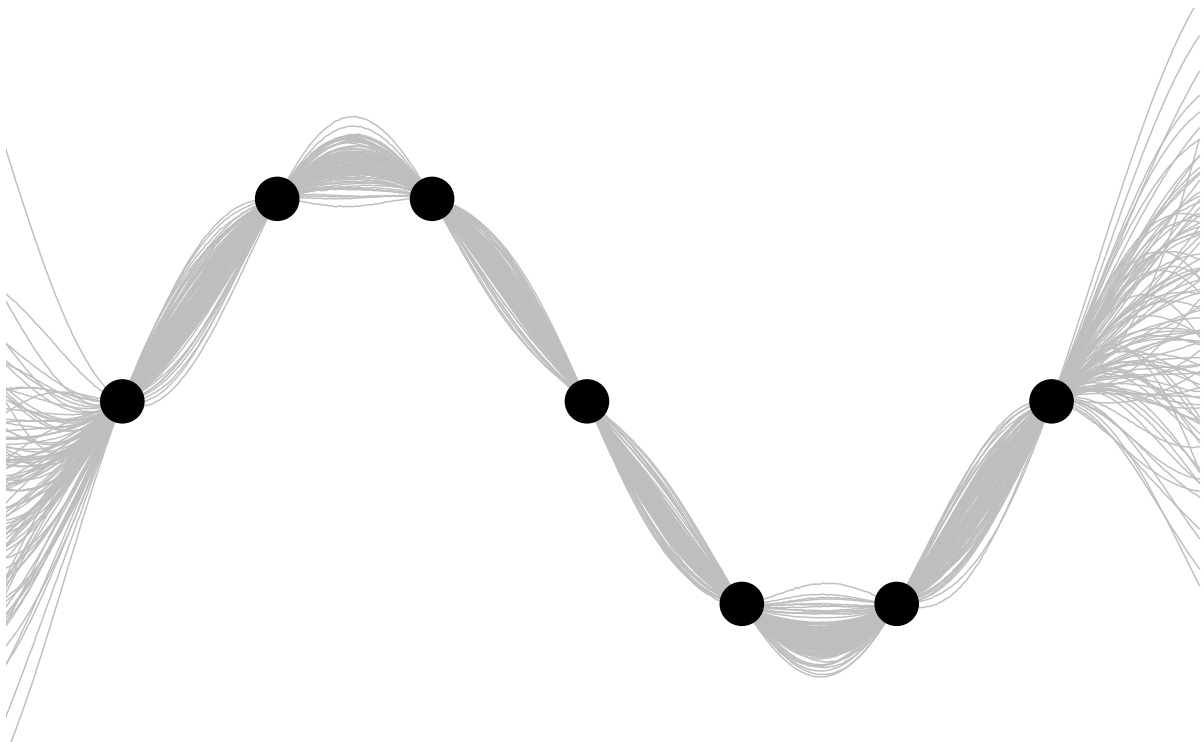
- Analytic derivatives facilitate Newton-like inference for θ.

Kriging

$Y(x)|D, K$ is Student-t with degrees of freedom N,

$$\text{mean} \quad \mu(x|D,K) = k^\top(x)K^{-1}Y, \quad \text{where } k(x)_i = K(x, x_i)$$

$$\text{and scale} \quad \sigma^2(x|D,K) = \frac{\psi[K(x,x) - k^\top(x)K^{-1}k(x)]}{N}.$$



Limitations

The biggest drawback is computational.

- Due to the $O(N^3)$ matrix decompositions, the experiments must be small (usually $N \ll 1000$).

And N is getting big.

- $N = 27$ no longer the prototypical example.
  (Morris, et al., 1993; ... Chen, et al., 2014)

Supercomputers make one run as as cheap as thousands.

- $N = 20$K cosmology/redshift (Kaufman, et al., 2012)
- $N = 60$K cosmology/supernova (Paciorek, et al., 2014)
- $N = 7$M climate/temperature (Pratola, et al., 2014)

Sparsity and local scope

Key themes in fast GPs are approximation and sparsity.
(Snelson & Ghahramani, 2006; Sang & Huang, 2012; Cressie & Johannesson, 2008; Kaufman, et al., 2012)

Our approach (G & Apley, 2014) proceeds similarly, yet is particularly well tailored to calibration:

- where emulation $\hat{Y}_M(x, u)$ is needed only for x's corresponding to N_F field data runs.

It is reminiscent of *ad hoc* methods based on local kriging neighborhoods (e.g., Cressie, 1991, pp. 131–134)

- and tailored to modern parallel computing architectures.

Back to using x generically, rather than (x, u).

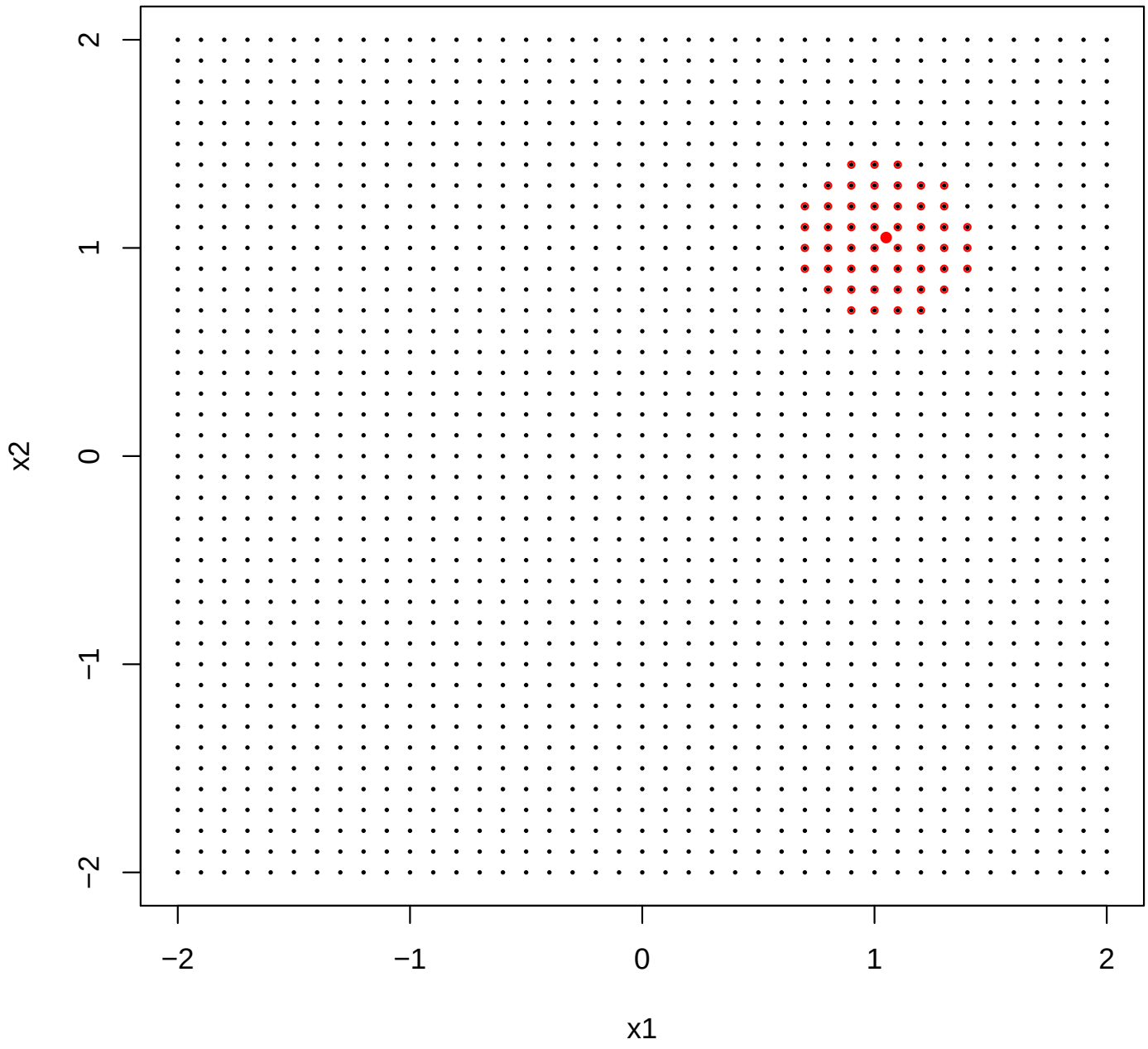
The idea is to concentrate on predicting well, *local* to x.

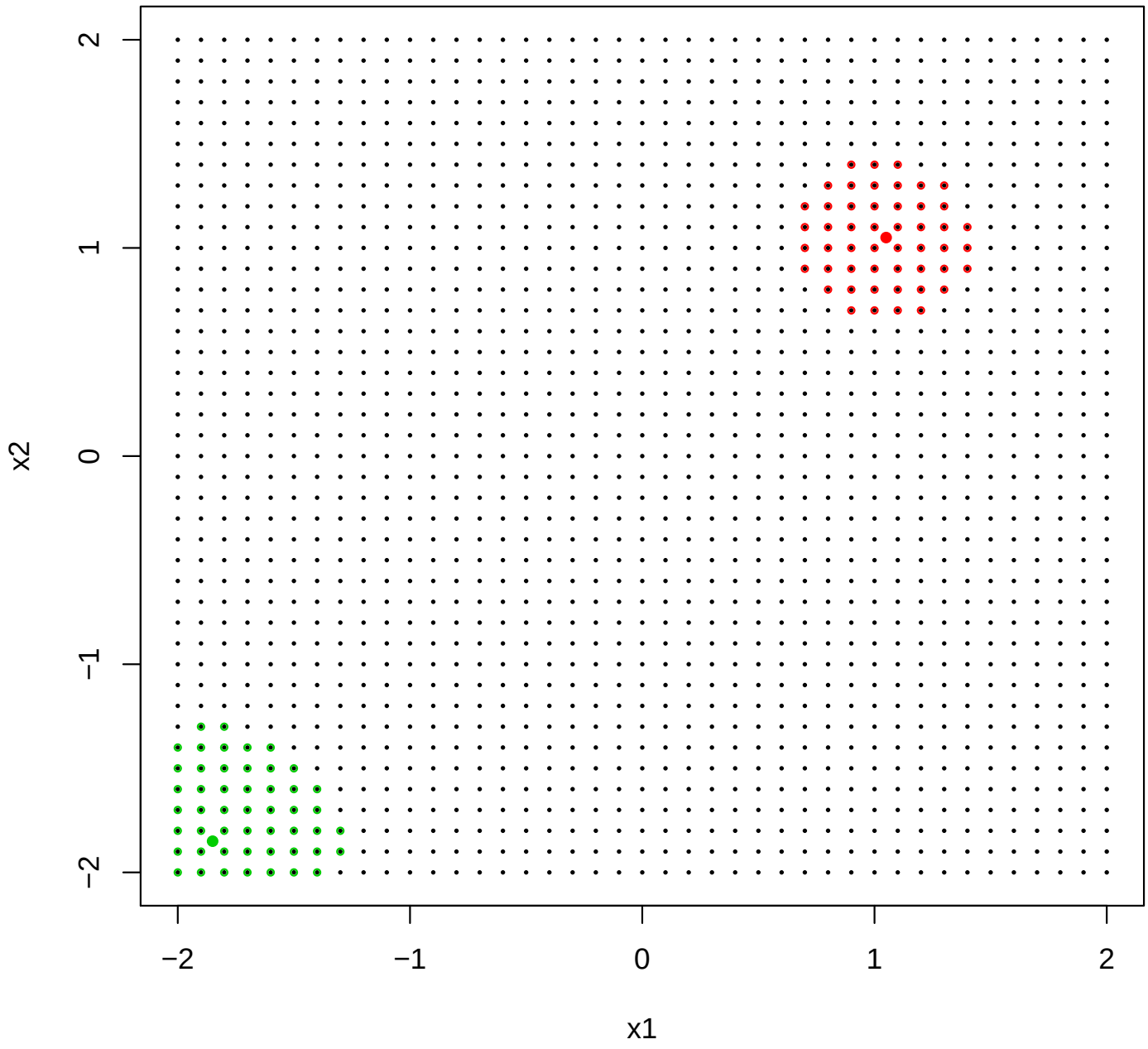
- Data far from x have vanishing influence on prediction.
- So search for the most useful data points (a sub-design relative to x) without considering/handling large matrices.

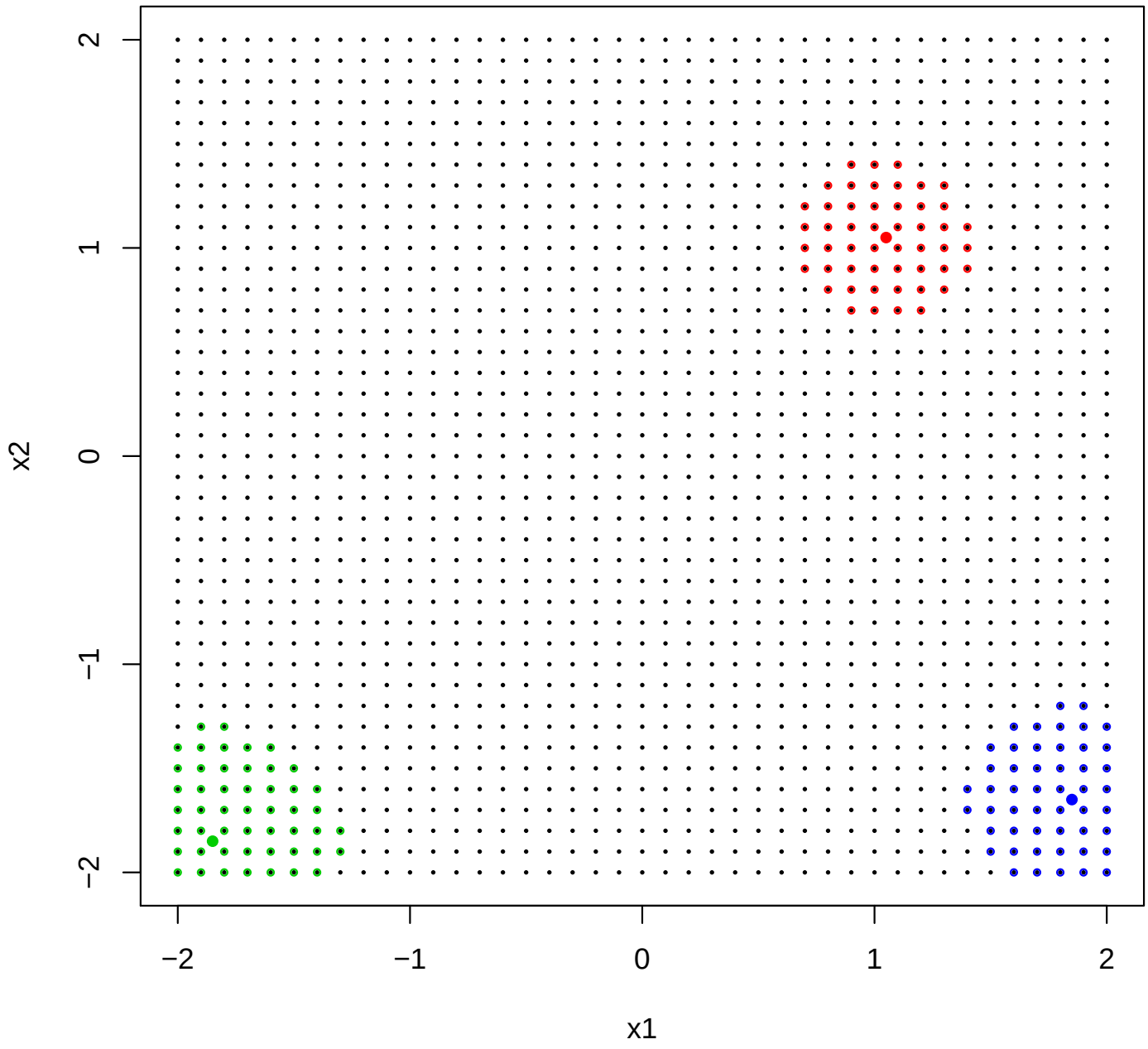
One option is nearest neighbor (NN):

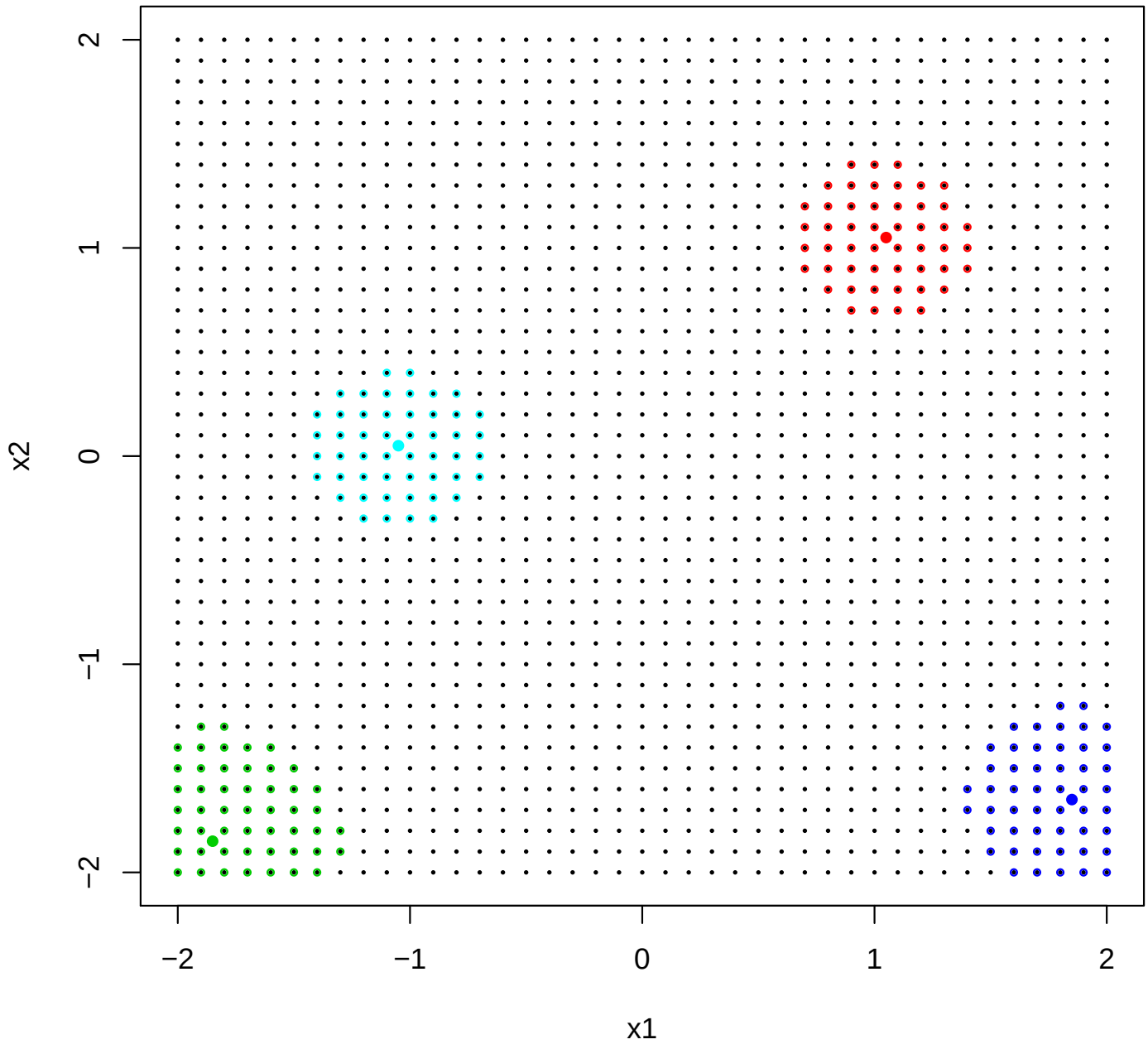
1. fill $X_n(x) \subseteq X$ with the n closest locations to x
2. emulate with $Y(x)|D_n(x)$ where $D_n(x) = (X_n, Y_n)$

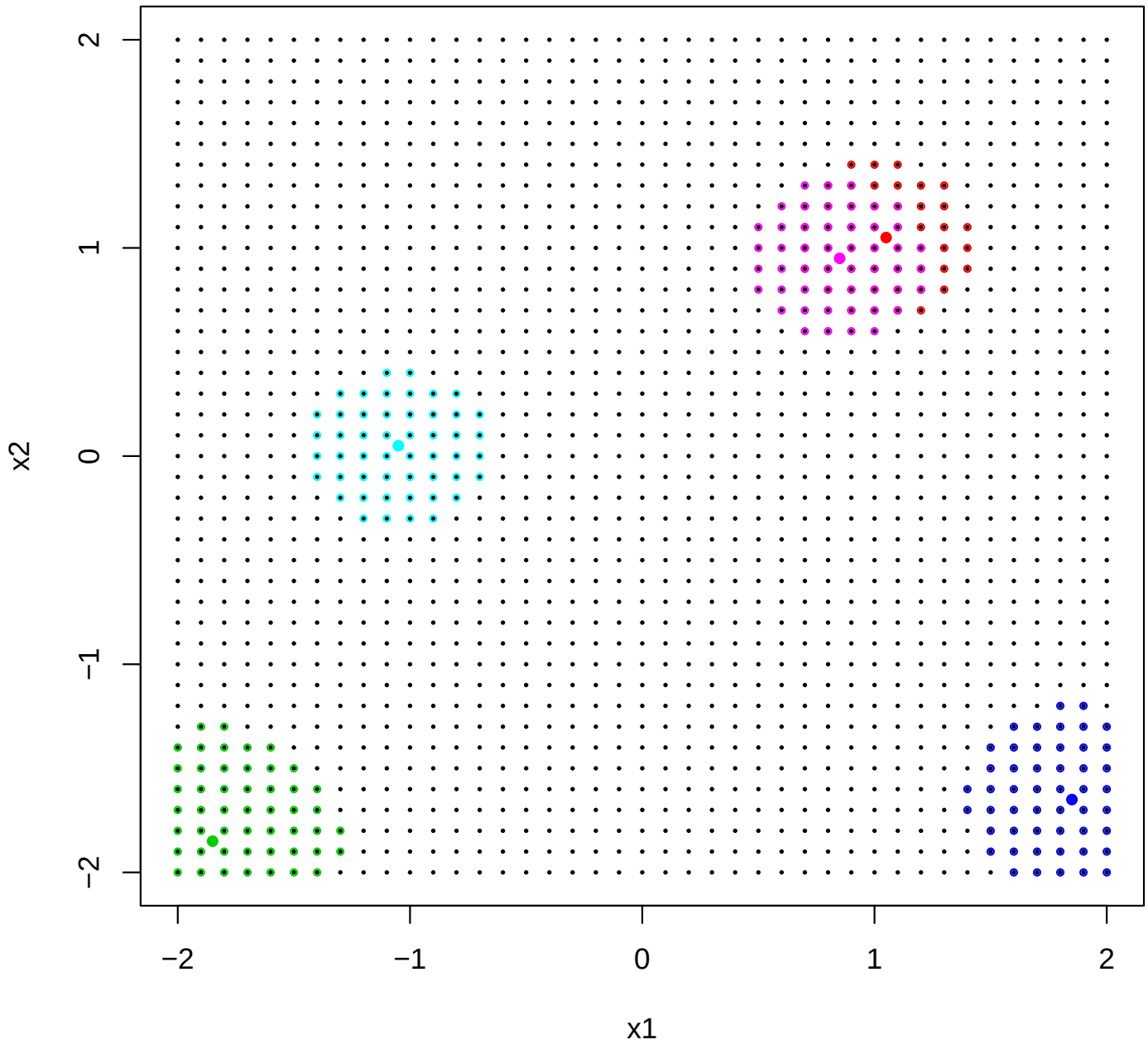
Choose n as large as computational constraints allow.


x2
2
1
0
−1
−2
−2
−1
0
1
2
x1


x2
2
1
0
−1
−2
−2
−1
0
1
2
x1


x2
2
1
0
−1
−2
−2
−1
0
1
2
x1


x2
2
1
0
−1
−2
−2
−1
0
1
2
x1


x2
2
1
0
−1
−2
−2
−1
0
1
2
x1

Sensible?

- as $n \to N$, predictions $Y(x)|D_n \to Y(x)|D$.
- *Usually* $V(x)|D_n \gg V(x)|D$, reflecting uncertainties inflated by the smaller design.

Good?

- It is *not optimal* given computational constraints, n. (Vecchia, 1998; Stein, et al., 2004)
- But, the optimal solution(s) involve a high-dimensional non-convex optimization.

G & Apley showed

- you can do better than NN without much extra effort ...

... with a greedy/forward stepwise scheme.

For a particular x, solve a sequence of *easy* decision problems.

For $j = n_0, \ldots, n$:

1. given $D_j(x)$, choose x_{j+1} to minimize empirical Bayes mean-squared prediction error:

$$J(x_{j+1}, x) = \mathbb{E}\{[Y(x) - \mu_{j+1}(x; \hat{\theta}_{j+1})]^2 | D_j(x)\}$$
$$\approx V_j(x|x_{j+1}; \hat{\theta}_j) + \left(\frac{\partial \mu_j(x;\theta)}{\partial \theta}\Big|_{\theta=\hat{\theta}_j} \right)^2 / \mathcal{G}_{j+1}(\hat{\theta}_j).$$

2. augment the design $D_{j+1}(x) = D_j(x) \cup (x_{j+1}, y(x_{j+1}))$ and update the GP approximation

A special case

Minimizing the reduced variance $V_j(x|x_{j+1};\theta_j)$, as a criteria on its own, presents an attractive option.

- Avoids complicated calculations involving derivatives.

In fact, an important component of this quantity has been used in sequential design of computer experiments before, as part of the active learning Cohn (ALC) heuristic.

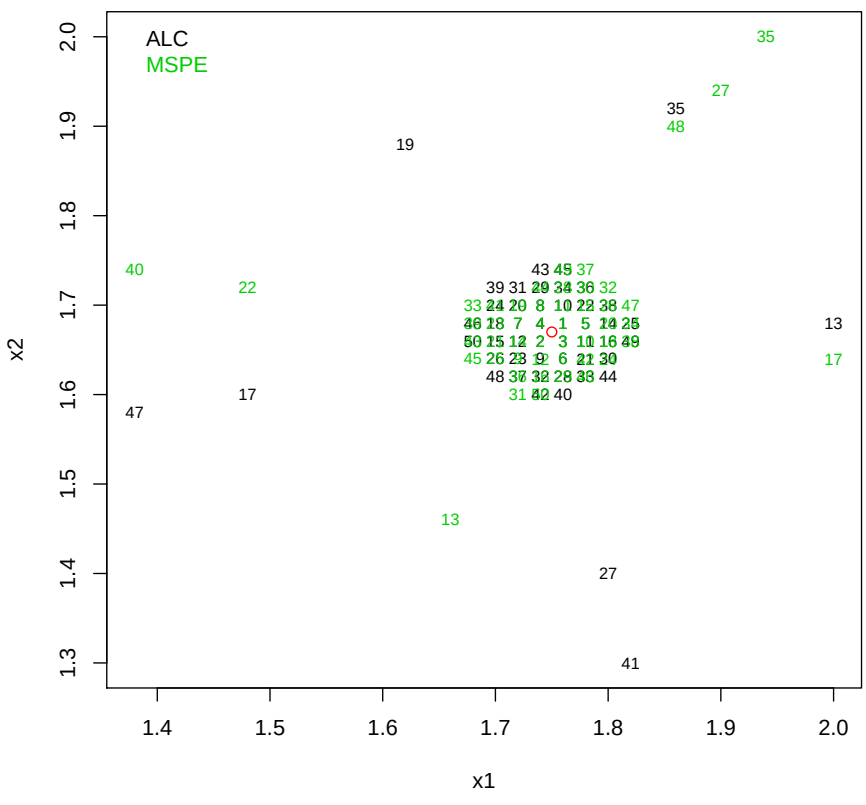
(G & Lee, 2009; Seo, et al., 2000; Cohn, 1996)

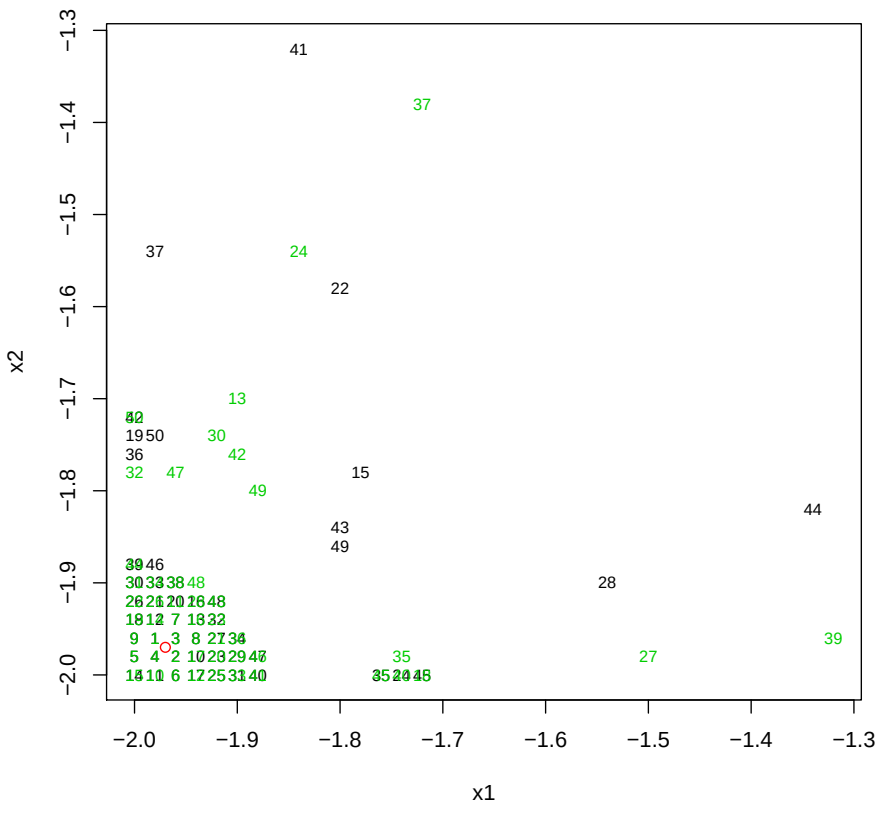
- ... making a connection to *approximate maximum information designs*.

Comparing greedy heuristics

$$f(x_1, x_2) = -w(x_1)w(x_2), \quad \text{where}$$
$$w(x) = \exp\left(-(x-1)^2\right) + \exp\left(-0.8(x+1)^2\right) - 0.05\sin\left(8(x+0.1)\right)$$

with X on a 201×201 ($= 40401$ point) regular grid in $[-2, 2]$.





Global emulation

One option is to serialize:

- loop over each $x \in \mathcal{X}$, or X_F for calibration, collecting approximate predictions

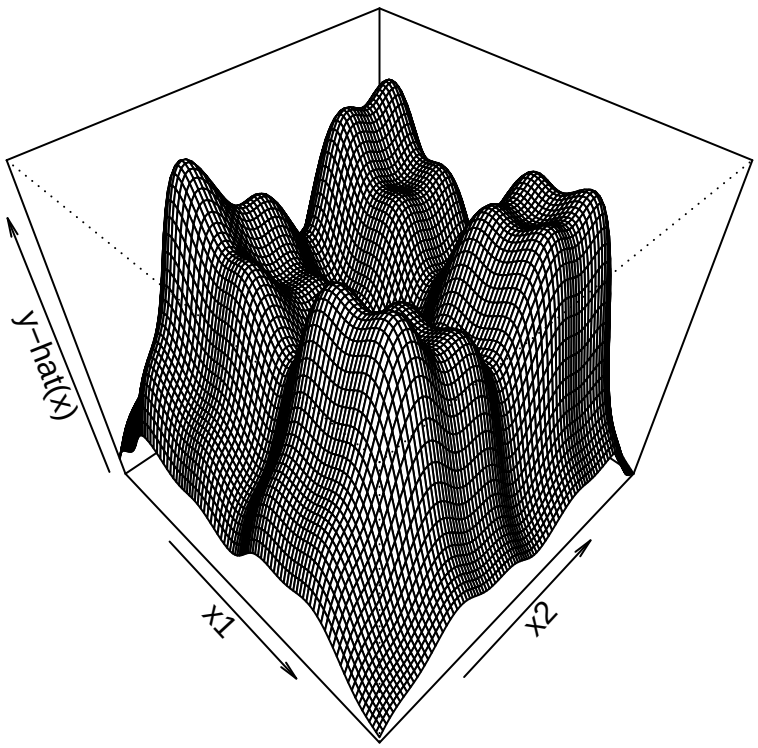
But why serialize when you can parallelize?

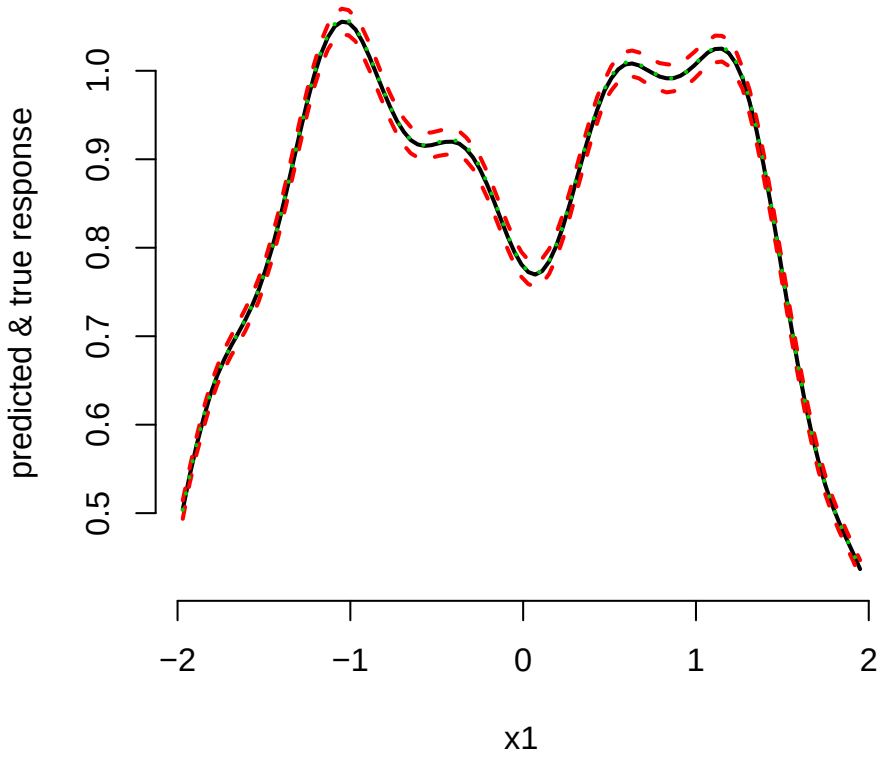
- each $D_n(x)$ is obtained independently of other x's

Predicting at 10K locations (40K design) on an 4-core iMac, takes a couple minutes and requires token programmer effort:

```
#pragma omp parallel for private(i)
for(i=0; i<npred; i++) { ...
```

Here is what the estimated surface looks like.





- No continuity enforced, but looks pretty continuous.

Speeding up search

The most important subroutine for ALC-based local design,

$$\underset{x_{j+1} \in X - X_j(x)}{\operatorname{argmin}} \; V_j(x|x_{j+1}; \theta_j),$$

—already parallelized for independent x—can be

- off-loaded to a GPU: each $x_{j+1} \in X - X_j(x)$ on a separate block with $O(j^2)$ linear algebra on j threads in the block (G, Niemi, Weiss, 2014)
- approximated with a 1-d continuous line search along rays eminating from x. (G & Haaland, 2014)

Speed and accuracy with increasing fidelity (N and n).

		exhaustive				via rays			
		Intel Sandy Bridge/Nvidia Tesla				iMac		Intel SB	
		96x CPU		5x 2 GPUs		1x(4-core) CPU		96x CPU	
N	n	secs	mse	secs	mse	secs	mse	secs	mse
1K	40	0.5	4.88	1.9	4.63	8.0	6.30	0.4	6.38
2K	42	0.7	3.67	2.9	3.93	17.8	4.47	0.5	4.10
4K	44	0.9	2.35	6.0	2.31	40.6	3.49	0.6	2.72
8K	46	1.8	1.73	13.1	1.74	96.9	2.24	1.3	1.94
16K	48	4.0	1.25	29.5	1.28	222.4	1.58	2.3	1.38
32K	50	10.0	1.01	67.1	1.00	490.9	1.14	4.7	1.01
64K	52	28.2	0.78	164.3	0.76	1076.2	0.85	9.9	0.73
128K	54	84.0	0.60	443.7	0.60	3017.8	0.62	18.0	0.55
256K	56	261.9	0.46	1254.6	0.46	5430.7	0.47	40.2	0.43
512K	58	836.0	0.35	4015.1	0.36	12931.9	0.35	80.9	0.33
1.2M	60	2789.8	0.26	13694.5	0.27	32867.0	0.27	188.9	0.26
2.5M	62	-	-	-	-	-	-	466.4	0.21
5.0M	64	-	-	-	-	-	-	1215.3	0.19
8.2M	66	-	-	-	-	-	-	4397.3	0.17

Calibration as optimization

Now we have the ability to emulate quickly, where we need it:

- $\hat{Y}_M(X_F, u)$, for "plausible" u-values.

Residuals $R_F^u \equiv \hat{Y}_M(X_F, u) - Y_f$ can be used to estimate the bias $b(\cdot)$

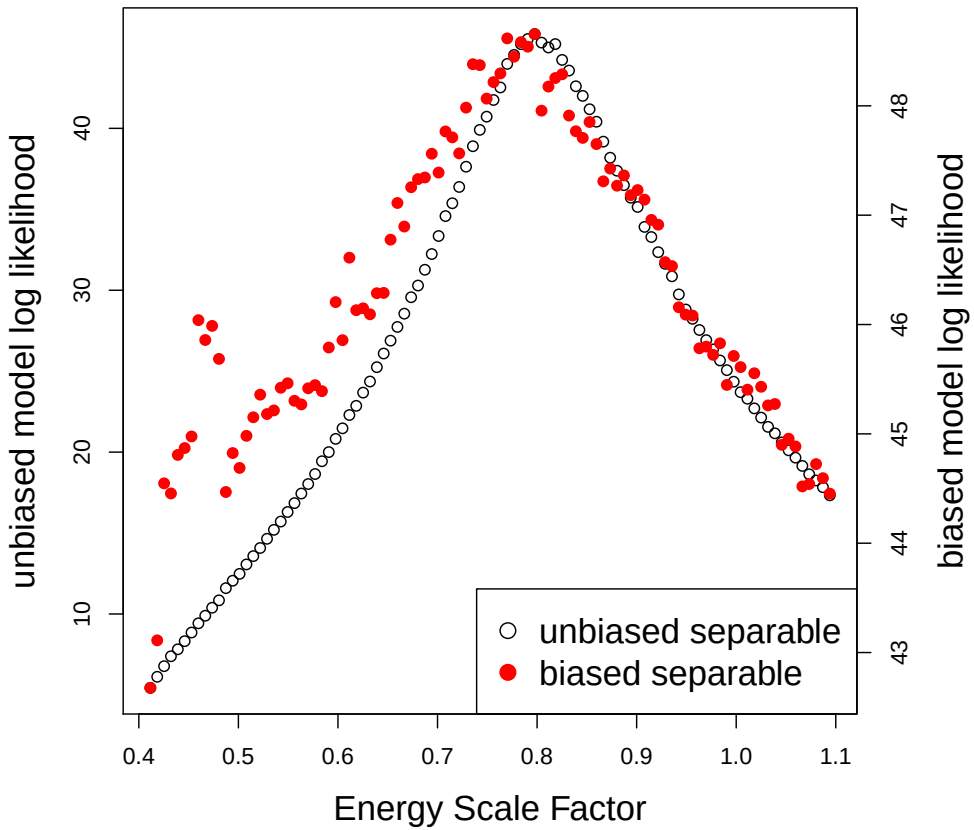
- A GP prior for b is reasonable when N_F is small
- The likelihood/posterior maximizing GP

$$\mathrm{Obj}(u) \equiv \max_{\theta} p_b(\theta | R_F^u), \quad \text{via Newton for } \theta$$

defines an objective in u which can be maximized.

A challenge, however, is that the local GP approximation for $\hat{Y}(X_F, u)$ is not continuous in u, making $\mathrm{Obj}(u)$ look "noisy".

- Due to the discrete nature of search for the local design(s).
- Optimizing with conventional (e.g., Newton-based) methods is hopeless.
- We go derivitive-free instead (Conn, et al., 2009).



NOMAD

For derivative-free optimization we use mesh adaptive direct search (MADS) (Audet and Dennis, Jr., 2006)

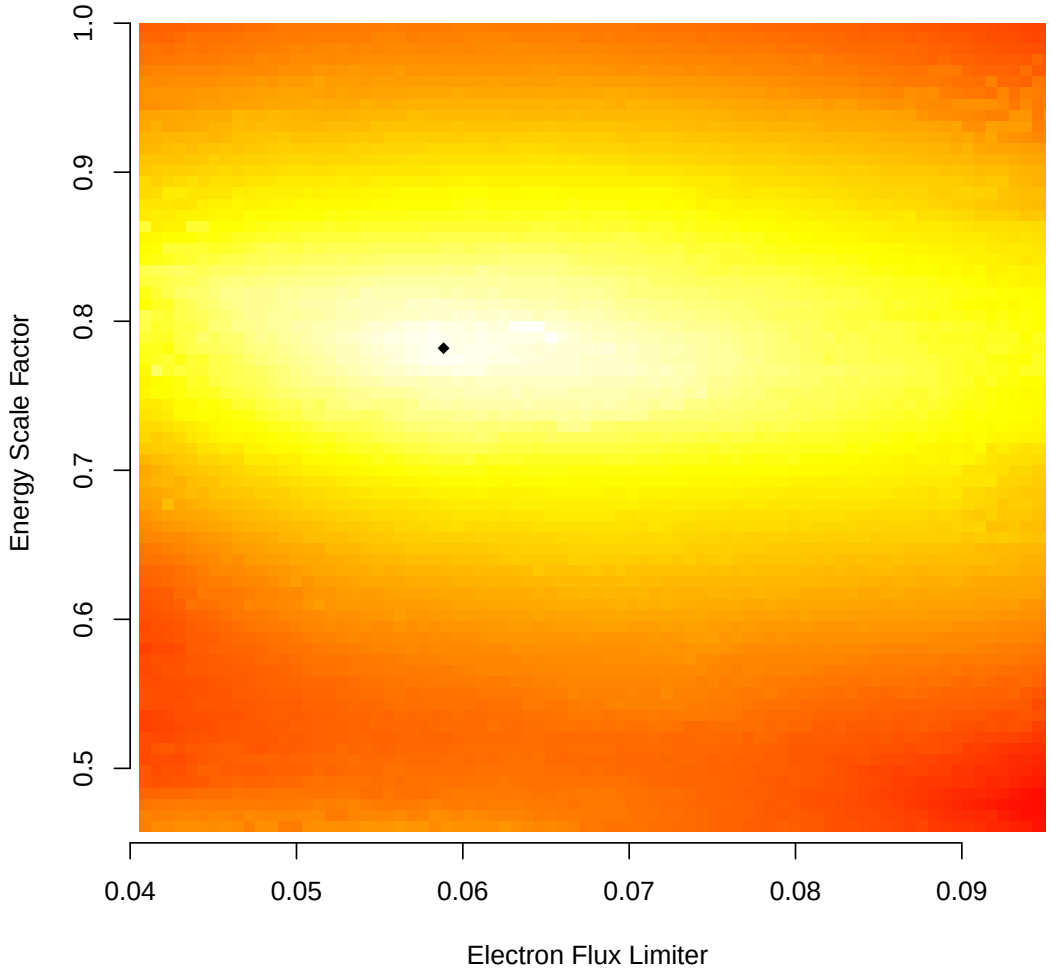
- via the NOMAD library (Le Digabel, 2011)
- wrapped by R package `crs` (Racine and Nie, 2012)

Our implementation used

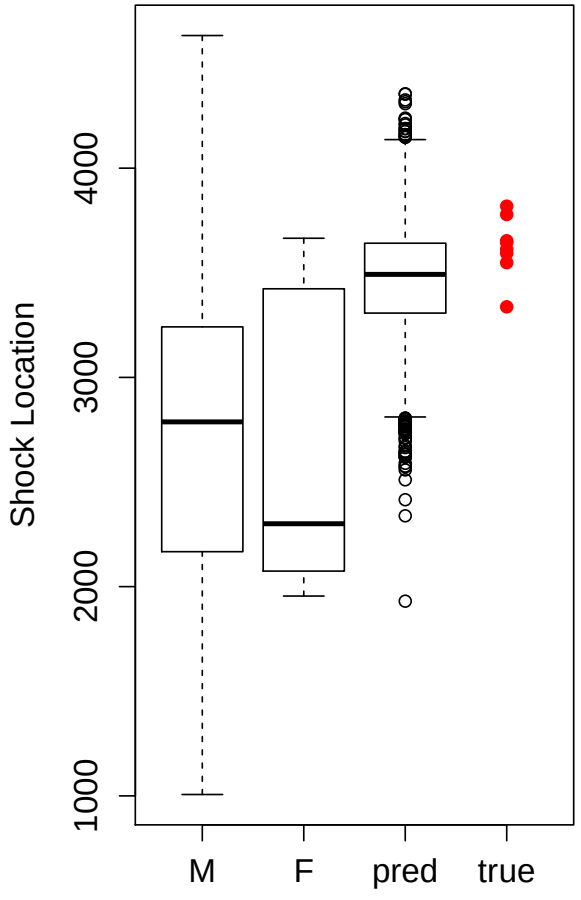
- a 200-element space filling design to search for good initial u-values.
- Then `NOMAD` required a further ~ 500 evaluations

taking a total of fifteen minutes on a 4-core iMac.

$\mathrm{Obj}(u)$, all $\sim$ 700 evals;



field predictions OOS.



Summarizing

Computer experiments are getting too big for conventional methods,

- like GPs and fully Bayesian calibration.

We illustrated a thrifty calibration approach combining

- local approximate GP modeling,
- modularized calibration,
- and derivative-free optimization.

Everything in the `laGP` package on CRAN!