optimal design may not choose candidates in the "interesting" part of the input space, because sampling is high there already. Classic optimal design criteria, in general, are ill-suited partition models wherein "closeness" may not measured homogeneously across the input space. Another disadvantage is computational, namely decomposing and finding the determinant of a large covariance matrix.

One possible solution to both computational and nonstationary modeling issues is to use treed sequential D-optimal design [10]. Separate sequential D-optimal designs can be computed in each of the partitions depicted by the maximum *a posteriori* (MAP) tree $\hat{\mathcal{T}}$. The number of candidates selected from each region can be proportional to the volume of—or proportional to the number of grid locations in—the region. MAP parameters $\hat{\boldsymbol{\theta}}_\nu|\hat{\mathcal{T}}$, or "neutral" or "exploration encouraging" ones, can be used to create the candidate design. Separating design from inference by using custom parameterizations in design steps, rather than inferred ones, is a common practice [20]. Small range parameters, for learning about the wiggliness of the response, and a modest nugget parameter, for numerical stability, tend to work well together.

Finding a local maxima is generally sufficient to get well-spaced candidates. The `dopt.gp` function uses a stochastic ascent algorithm which can find local maxima without calculating too many determinants.

3 Examples using `tgp`

The following subsections take the reader through a series of examples based, mostly, on synthetic data. At least two different `b*` models are fit for each set of data, offering comparisons and contrasts. Duplicating these examples in your own `R` session is highly recommended. The `Stangle` function can help extract executable `R` code from this document. For example, the code for the exponential data of Section 3.3 can be extracted with one command.

```
> Stangle(vignette("exp", package="tgp")$file))
```

This will write a file called "exp.R". Additionally, each of the subsections that follow is available as an `R` demo. Try `demo(package="tgp")` for a listing of available demos. To envoke the demo for the exponential data of Section 3.3 try `demo(exp, package="tgp")`. This is equivalent to `source("exp.R")` because the demos were created using `Stangle` on the source files of this document.

Other successful uses of the methods in this pacakge include applications to the Boston housing data [13], and designing an experiment for a reusable NASA launch vehicle [11, 10] called the Langely glide-back booster (LGBB).

3.1 1-d Linear data

Consider data sampled from a linear model.

$$z_i = 1 + 2x_i + \epsilon, \quad \text{where} \quad \epsilon_i \overset{\text{iid}}{\sim} N(0, 0.25^2) \tag{13}$$

The following R code takes a sample $\{\mathbf{X}, \mathbf{Z}\}$ of size $N = 50$ from (13). It also chooses $N' = 99$ evenly spaced predictive locations $\tilde{\mathbf{X}} =$ XX.

```
> X <- seq(0, 1, length = 50)
> XX <- seq(0, 1, length = 99)
> Z <- 1 + 2 * X + rnorm(length(X), sd = 0.25)
```

Using tgp on this data with a Bayesian hierarchical linear model goes as follows:

```
> lin.blm <- blm(X = X, XX = XX, Z = Z)

tree[alpha,beta,nmin]=[0,0,10]
n=50, d=1, nn=99
BTE=(1000,4000,3), R=1, linburn=0
preds: data
linear prior: flat
s2[a0,g0]=[5,10]
s2 lambda[a0,g0]=[0.2,10]
corr prior: separable power
nug[a,b][0,1]=[1,1],[1,1]
nug prior fixed
gamlin=[-1,0.2,0.7]
d[a,b][0]=[1,20],[10,20]
d prior fixed

burn in:
r=1000 corr=[0] : n = 50

Obtaining samples (nn=99 predictive locations):
r=1000 corr=[0] : mh=1 n = 50
r=2000 corr=[0] : mh=1 n = 50
r=3000 corr=[0] : mh=1 n = 50

finished repetition 1 of 1
removed 0 leaves from the tree
```

The first group of text printed to stdout is a summary of inputs to the C code, and the prior parameterization. Then, MCMC progress indicators are printed every 1,000 rounds. The linear model is indicated by cor=[0]. In terminal versions, e.g. Unix, the progress indicators can give a sense of when the code will finish. GUI versions of R—Windows or MacOS X—usually buffer stdout, rendering this feature essentially useless as a real-time indicator of progress.

The generic plot method can be used to visualize the fitted posterior predictive surface (with option layout = 'surf') in terms of means and credible intervals. Figure 3 shows how to do it, and what you get. The default option

```
> plot(lin.blm, main = "Linear Model,", layout = "surf")
> abline(1, 2, lty = 3, col = "blue")
```

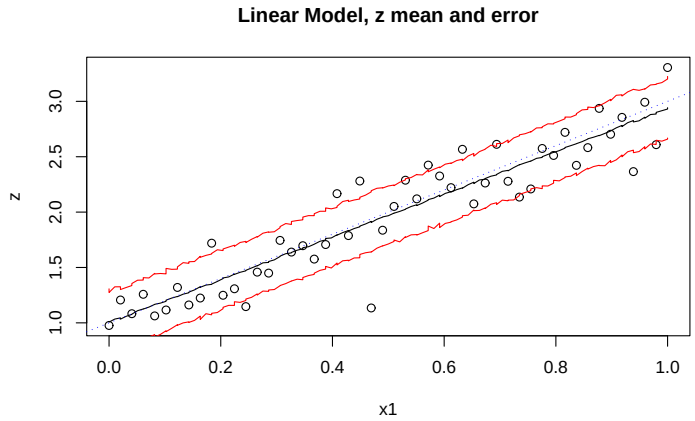

Figure 3: Posterior predictive distribution using `blm` on synthetic linear data: mean and 90% credible interval. The actual generating lines are shown as blue-dotted.

`layout = 'both'` shows both a predictive surface and error (or uncertainty) plot, side by side. The error plot can be obtained alone via `layout = 'as'`. Examples of these layouts appear later.

If, say, you were unsure about the dubious "linearness" of this data, you might try a GP LLM (using `btgpllm`) and let a more flexible model speak as to the linearity of the process.

```
> lin.gpllm <- bgpllm(X = X, XX = XX, Z = Z)

tree[alpha,beta,nmin]=[0,0,10]
n=50, d=1, nn=99
BTE=(1000,4000,2), R=1, linburn=0
preds: data
linear prior: flat
s2[a0,g0]=[5,10]
s2 lambda[a0,g0]=[0.2,10]
corr prior: separable power
nug[a,b][0,1]=[1,1],[1,1]
nug prior fixed
gamlin=[10,0.2,0.7]
d[a,b][0]=[1,20],[10,20]
d prior fixed

burn in:
r=1000 corr=[0] : n = 50
```

```
Obtaining samples (nn=99 predictive locations):
r=1000 corr=[0] : mh=1 n = 50
r=2000 corr=[0] : mh=1 n = 50
r=3000 corr=[0] : mh=1 n = 50

finished repetition 1 of 1
removed 0 leaves from the tree
```

```
> plot(lin.gpllm, main = "GP LLM,", layout = "surf")
> abline(1, 2, lty = 4, col = "blue")
```

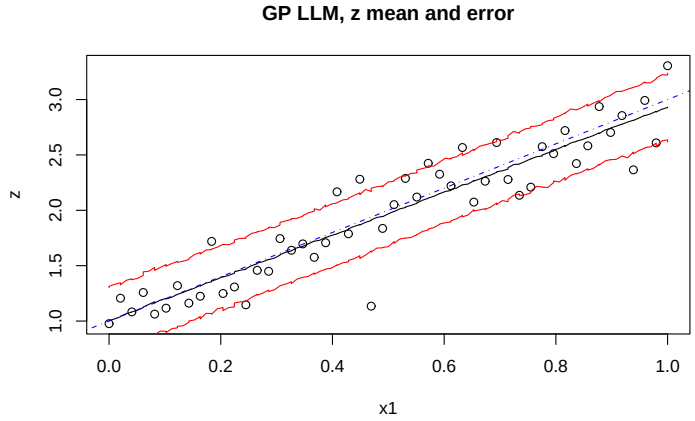

Figure 4: Posterior predictive distribution using `bgpllm` on synthetic linear data: mean and 90% credible interval. The actual generating lines are shown as blue-dotted.

Whenever the progress indicators show `corr[0]` the process is under the LLM in that round, and the GP otherwise. A plot of the resulting surface is shown in Figure 4 for comparison. Since the data is linear, the resulting predictive surfaces should look strikingly similar to one another. On occasion, the GP LLM may find some bendy-ness in the surface. This happens rarely with samples as large as $N = 50$, but is quite a bit more common for $N < 20$.

3.2 1-d Synthetic Sine Data

Consider 1-dimensional simulated data which is partly a mixture of sines and cosines, and partly linear.

$$z(x) = \begin{cases} \sin\left(\frac{\pi x}{5}\right) + \frac{1}{5}\cos\left(\frac{4\pi x}{5}\right) & x < 10 \\ x/10 - 1 & \text{otherwise} \end{cases} \tag{14}$$

The `R` code below obtains $N = 100$ evenly spaced samples from this data in the domain $[0, 20]$, with noise added to keep things interesting. Some evenly spaced predictive locations `XX` are also created.