```
> fr.btlm <- btlm(X = X, Z = Z, XX = XX, tree = c(0.95,
+     2, 10), m0r1 = TRUE)
> fr.btlm.mse <- sqrt(mean((fr.btlm$ZZ.mean - ff$Ytrue)^2))
> fr.btlm.mse
```

Next, fit the GP LLM, and obtain its RMSE.

```
> fr.bgpllm <- bgpllm(X = X, Z = Z, XX = XX, m0r1 = TRUE)
> fr.bgpllm.mse <- sqrt(mean((fr.bgpllm$ZZ.mean - ff$Ytrue)^2))
> fr.bgpllm.mse
```

So, the GP LLM is 4.834 times better than Bayesian linear CART on this data, in terms of RMSE (in terms of MSE the GP LLM is 2.199 times better). Watching the evolution of the Markov chain for the GP LLM (via the progress statements written to `stdout`, not shown because the would fit on the page), it is easy to see how the GP LLM quickly learns that $\mathbf{b} = (1, 1, 1, 0, 0, 0, 0, 0, 0, 0)$, and that $\beta_4 \approx 4$ and $\beta_5 \approx 10$—basically that only the first three inputs contribute nonlinearly, the fourth and fifth contribute linearly, and the remaining five not at all [10].

3.6 Adaptive Sampling

In this section, sequential design of experiments, a.k.a. *adaptive sampling*, is demonstrated on the exponential data of Section 3.3. Gathering, again, the data:

```
> exp2d.data <- exp2d.rand()
> X <- exp2d.data$X
> Z <- exp2d.data$Z
> Xcand <- exp2d.data$XX
```

Start by fitting a treed GP LLM model to the data, without prediction, in order to infer the MAP tree $\hat{\mathcal{T}}$.

```
> exp1 <- btgpllm(X = X, Z = Z, pred.n = FALSE, corr = "exp",
+     R = 2)
```

The trees are shown in Figure 15. Then, use the `tgp.design` function to create D-optimal candidate designs in each region of $\hat{\mathcal{T}}$.

```
> XX <- tgp.design(10, Xcand, exp1)

sequential treed D-Optimal design in 3 partitions
dopt.gp (1) choosing 2 new inputs from 63 candidates
dopt.gp (2) choosing 3 new inputs from 107 candidates
dopt.gp (3) choosing 6 new inputs from 191 candidates
```

Figure 16 shows the sampled `XX` locations (circles) amongst the input locations `X` (dots) and MAP partition ($\hat{\mathcal{T}}$). Notice how the candidates `XX` are spaced

```
> tgp.trees(exp1)
```

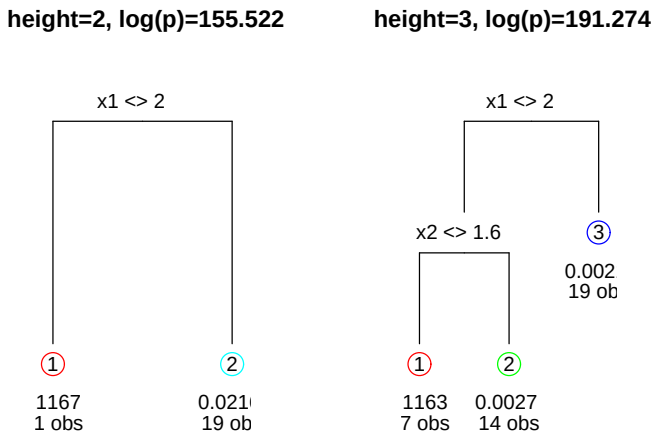

Figure 15: MAP trees of each height encountered in the Markov chain for the exponential data. $\hat{\mathcal{T}}$ is the one with the maximum $\log(p)$ above.

out relative to themselves, and relative to the inputs `X`, unless they are near partition boundaries. The placing of configurations near region boundaries is a symptom particular to D-optimal designs. This is desirable for experiments with `tgp` models, as model uncertainty is usually high there [2].

Figure 16 uses the `tgp.plot.parts.2d` function. Unfortunately, this function is not well documented in the current version of the `tgp` package. This should change in future versions.

Now, the idea is to fit the treed GP LLM model, again, in order to assess uncertainty in the predictive surface at those new candidate design points.

```
> exp1.btgpllm <- btgpllm(X = X, Z = Z, XX = XX, corr = "exp",
+     R = 2)
```

Figure 17 shows the posterior predictive surface. The error surface, on the *right*, summarizes posterior predictive uncertainty by a norm of quantiles. In accordance with the ALM algorithm, candidate locations `XX` with largest predictive error would be sampled (added into the design) next. These are most likely to be in the interesting region, i.e., the first quadrant. however, due to the random nature of this `Sweave` document, this is not always the case. Results depend heavily on the clumping of the original design in the un-interesting areas, and on the estimate of $\hat{\mathcal{T}}$.

Adaptive sampling via the ALC, or ego (or both) algorithms could proceed by setting any/all of the `Ds2x` or `ego` parameters to the `b*` and `tgp` to `TRUE`.

```
> plot(exp1$X, pch = 19, cex = 0.5)
> points(XX)
> tgp.plot.parts.2d(exp1$parts)
```

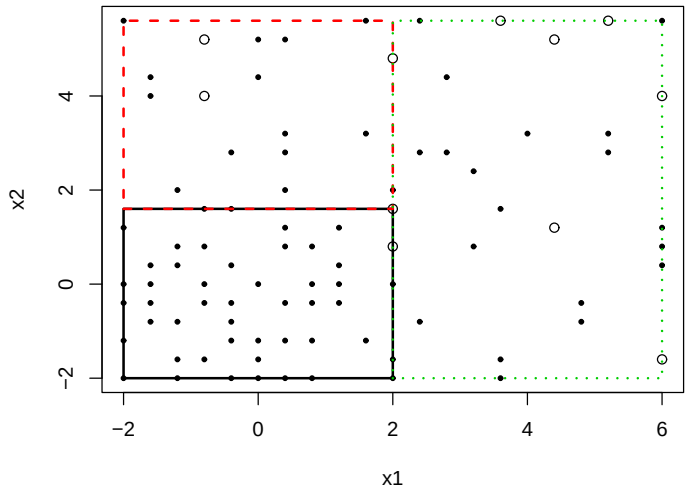

Figure 16: Treed D-optimal candidate locations `XX` (circles), input locations `X` (dots), and MAP tree $\hat{\mathcal{T}}$

```
> plot(exp1.btgpllm, main = "ALM for treed GP LLM,")
```

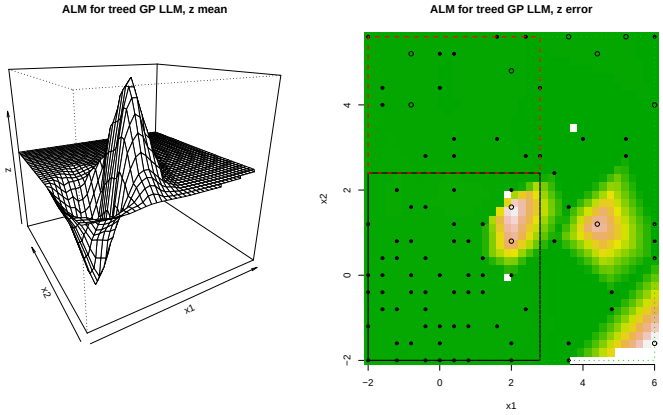

Figure 17: Treed D-optimal candidate locations `XX` (circles), input locations `X` (dots), and MAP tree $\hat{\mathcal{T}}$

A Linking to ATLAS

`ATLAS` is supported as an alternative to standard BLAS and LAPACK for fast, automatically tuned, linear algebra routines. There are three easy steps to