Constraining

We have two functions that both take only variables x_1 and x_2 as inputs:

- $y = f(x_1, x_2)$ is the **constraint** function. This is a function of whose output we are able to observe, and by doing so, at different values of x_1 and x_2, we hope to be able to constrain the output, reducing its a range to only plausible values based on observations.
- $z = h(x_1, x_2)$ is the **forcing** function.

We might hope that, having reduced the range of the output of f, the range of h would also reduce. Below we explore how the alignment of the output surfaces of f and h with each other can affect whether this in practice happens.

> Start with $X_1, X_2 \sim U[0, 1]$

We generate a random sample of size 1000 of each input variable, plotted in Figure 1.

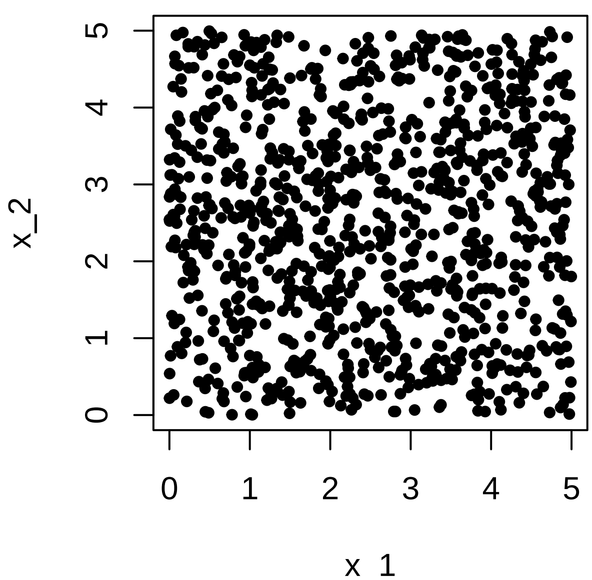


Figure 1: Random input variable settings.

Derive (using MC) $p(Z) \sim h(X_1, X_2)$

We will assume that the forcing function is

$$h(x_1, x_2) = \sqrt{100 - x_1^2 - x_2^2}.$$

This function is evaluated at the points selected by the random sample generated in the first step. Figure 2 illustrates how the value of z is associated to the values of x_1 and x_2.

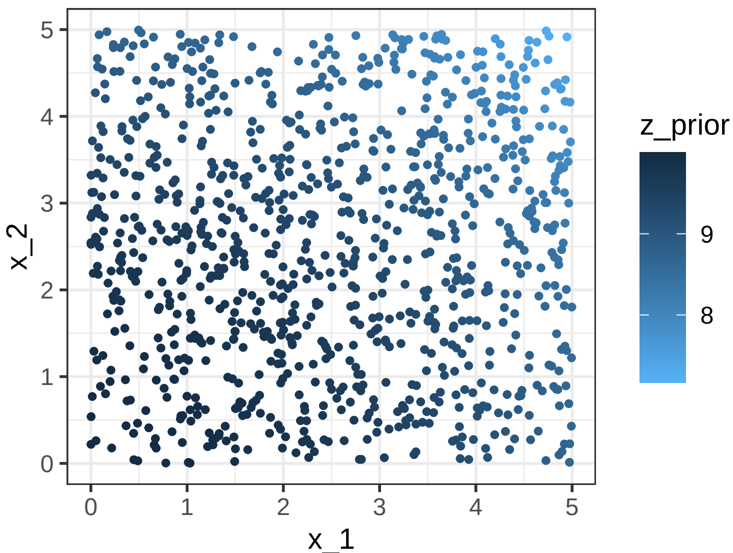


Figure 2: The colour of the points in this plot show the value of z at this point.

A histogram of the values of z resulting from the random sample provides an empirical distribution of $p(Z)$ – see Figure 3.

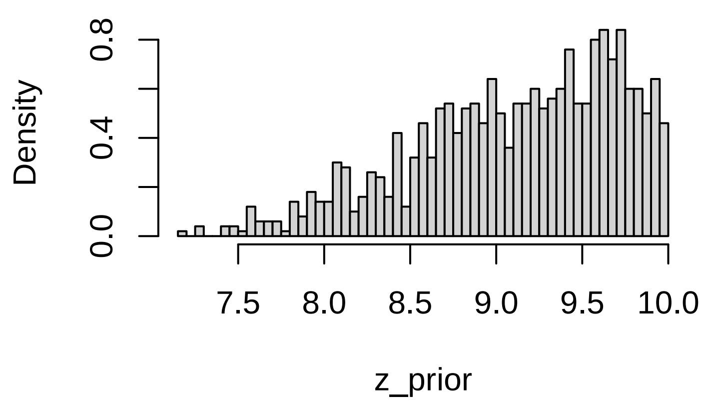


Figure 3: Approximate distribution for $p(Z)$.

Define $Y = f(X_1, X_2)$

To start with we'll look at perfect alignment between the two output surfaces, and so

$$f(x_1, x_2) = \sqrt{100 - x_1^2 - x_2^2}.$$

Observe $\tilde{y} = Y + \epsilon$

Let's start by observing one value of Y, at $(x_{1,1}, x_{2,1})$. We'll choose $(x_{1,1}, x_{2,1}) = (0.2, 0.2)$. Let $\epsilon \sim N(0, 0.1^2)$.

```
y_tilde_1 <- sqrt(100 - (x_1_1-5)^2 - x_2_1^2) +  rnorm(1, 0, 0.1)
y_tilde_1
```

```
## [1] 9.37615
```

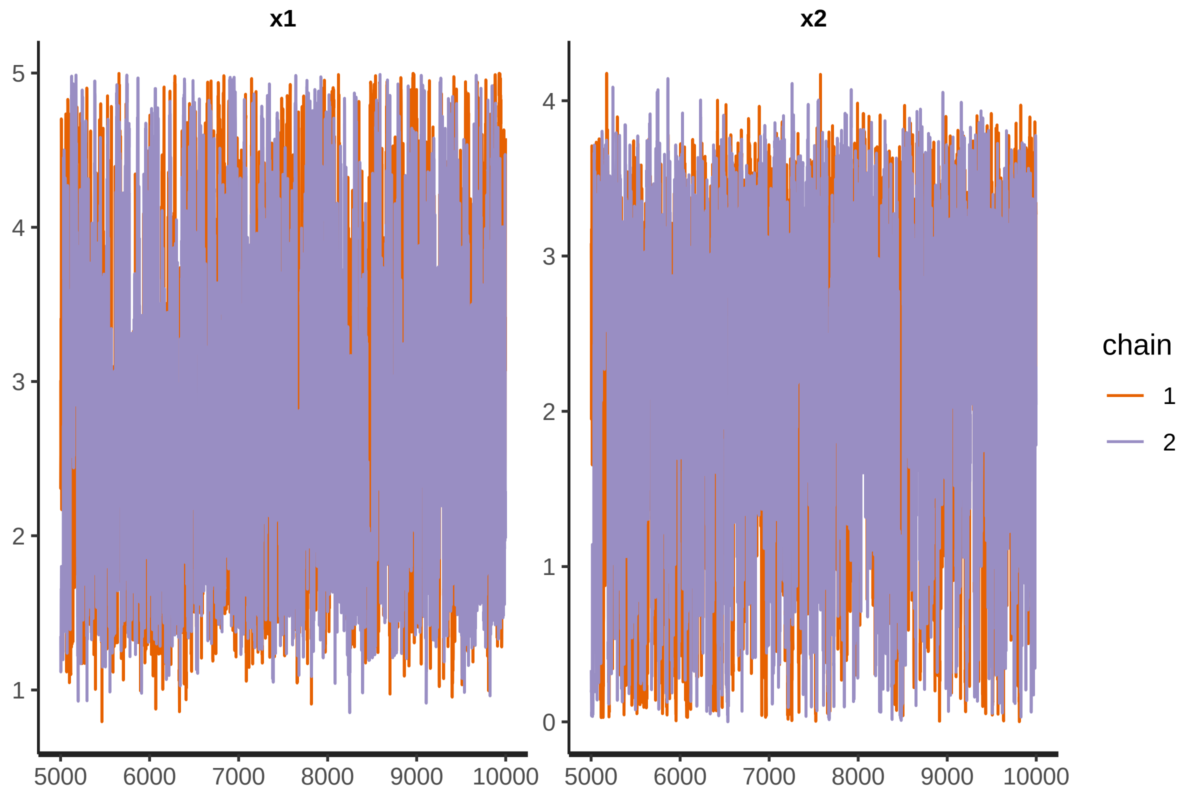
Derive (using MCMC) $p(X_1, X_2 \mid \tilde{y})$

```
data {
  real y_tilde ; // observations
}
parameters {
  real x1 ;
  real x2 ;
}
model {
  y_tilde ~ normal((100 - (x1-5)^2 - x2^2)^0.5, 0.1) ; // likelihood
  x1 ~ uniform(0,5) ;
  x2 ~ uniform(0,5) ;
}
generated quantities {
  real z = (100 - x1^2 - x2^2)^0.5 ;
}
```

```
fit
```

```
## Inference for Stan model: anon_model.
## 2 chains, each with iter=10000; warmup=5000; thin=1;
## post-warmup draws per chain=5000, total post-warmup draws=10000.
##
##      mean se_mean   sd  2.5%   25%   50%   75% 97.5% n_eff Rhat
## x1    2.8    0.03 1.08  1.29  1.86  2.59  3.69  4.84   959 1.01
## x2    2.2    0.04 1.08  0.14  1.31  2.41  3.13  3.71   946 1.01
## z     9.2    0.02 0.59  8.02  8.74  9.34  9.74  9.90   943 1.01
## lp__ -0.5    0.01 0.72 -2.55 -0.66 -0.23 -0.05  0.00  3834 1.00
```

```
##
## Samples were drawn using NUTS(diag_e) at Thu Sep 19 14:40:22 2024.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```



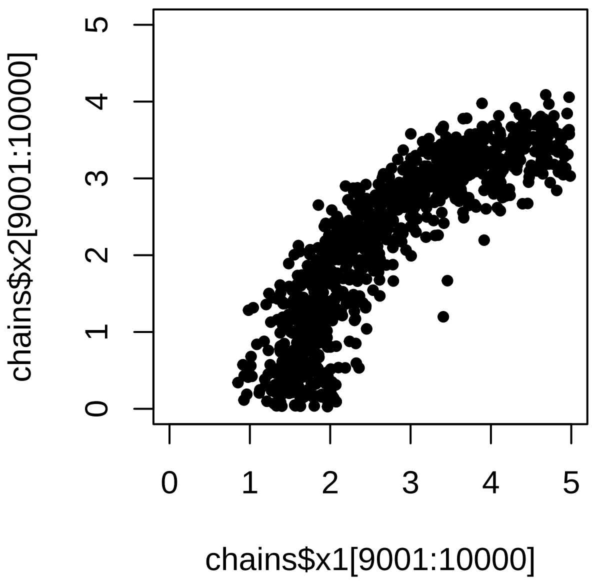


Figure 4: Random input variable settings.

Generate $p(Z \mid \tilde{y})$ and compare with $p(Z)$

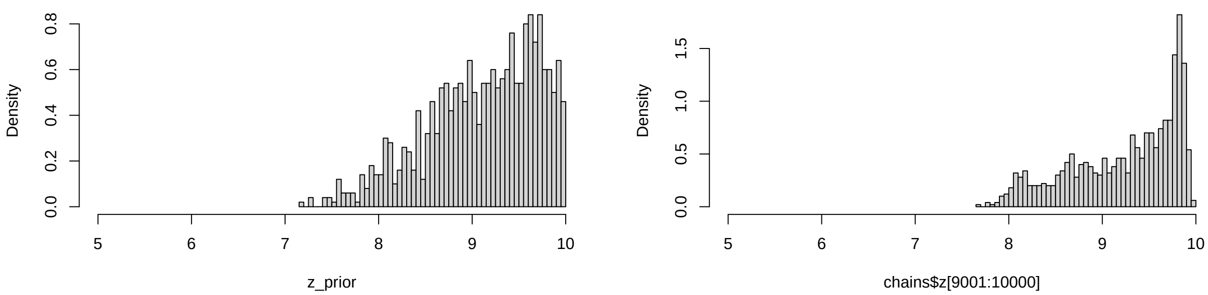


Figure 5: Prior (reprinted from Figure 3) and posterior draws of z, where $z = h(x_1, x_2) = \sqrt{100 - x_1^2 - x_2^2}$.

1 Citations