Very rough explanation of how I added parameter transformations to the TMB MCMC algorithms. Please read through and then address my questions below.

Let be the unconstrained parameter, and be the transformed variable. The log density () and gradient of x are known (fn and gr in TMB object). We wish to calculate these for the unconstrained parameters since that is what the MCMC algorithm will work on. Specifically, we need to adjust for the change in volume due to the transformation used. This leads to uniform priors on the constrained space. In the case of infinite support, the prior will be improper. This is how Stan is setup and makes sense for Bayesian modeling in particular.

Here I only include the possibility for univariate constraints, and as a result we can work with a single variable since the Jacobian of the transformation reduces to a triangular matrix. This is not true for more complex transformations like unit simplexes or constrained covariance and correlation matrices. The log density for a single variable is thus:

And the gradient is



Thus three terms (functions) are needed to calculate the log density and gradient for y. Following the approach taken by Stan, we have four cases for transformations. (1) None; (2) bounded below by a but unbounded above; (3) bounded above by b but not below; and (4) bounded below and above. The latter is all that ADMB is capable of, precluding the functionality that Stan has where a parameter such as a variance can be declared only with a lower bound of 0. In ADMB this would need to be done with a log transform manually, and the Jacobian added manually.

The table below indicates the necessary analytical terms needed to calculate the log density and gradient of log density of y from the TMB object. The implementation of these functions in R can be found [here](https://github.com/colemonnahan/adcomp/blob/mcmc/TMB/R/mcmc.R#L132). When the objective function is multivariate, the new objective function sums over the second term (log of absolute derivative of transformation), implemented [here](https://github.com/colemonnahan/adcomp/blob/mcmc/TMB/R/mcmc.R#L68).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Case | Bounds |  |  |  |
| 1 |  | y | 1 | 0 |
| 2 |  |  |  | 1 |
| 3 |  |  |  | 1 |
| 4 |  | a+ |  |  |

The user would thus specify any combination of these cases, with some parameters unbounded, some bounded below only (e.g., variance) and some with box constraints (e.g., probability parameter). It would look something like:

run\_mcmc(obj, …, lower=c(-Inf, 0, -5), upper=c(Inf, Inf, 5),…)

Additionally, the rotation via the Cholesky decomposition of the [mass matrix](https://github.com/colemonnahan/adcomp/blob/mcmc/TMB/R/mcmc.R#L505) also seems to work without any further adjustments if bounds are used. It is my understanding that these transformations are independent and nothing else is needed.

I’ve tested this with simple models and it seems to behave as I would expect. But I have some **Questions/Thoughts:**

1. Is my math right here? I originally left off the term and it was clearly wrong. It seems Stan (and ADMB) do the Jacobian calculation in the C++ source via AD, so the last term is not explicitly stated anywhere, including the [Stan manual](https://github.com/stan-dev/stan/releases/download/v2.12.0/stan-reference-2.12.0.pdf) (section 58).
2. When the Jacobian adjustment is done, and the bounds are not symmetrical (a=-3, b=2) then the mode changes. That doesn’t seem right to me but I checked it by putting the Jacobian in by hand in the template. Is that supposed to happen?
3. Because of the multiple cases (which will be very common in practice) and mixture of infinite bounds, it was not obvious how to vectorize the new adjusted fn and gr functions, other than the sapply calls. However, these are really slow in R. Any advice on alternative approaches?
4. I’ll send along more info on the HUGE improvements I’ve made to these algorithms and the interface soon, but I want to get these right.