# The GSI Minimization Code Structure

Kristen Bathmann

January 11, 2021

## 1 General Conjugate Gradient Code Structure

At each inner loop iteration $i$, the current estimate of the state $\mathbf{x}_i$ is updated as

$$\mathbf{x}_i = \mathbf{x}_{i-1} + \alpha_{i-1}\mathbf{d}_{i-1}. \tag{1}$$

This requires the computation of a search direction, $\mathbf{d}_{i-1}$ (a vector), and step size, $\alpha_{i-1}$ (a scalar). In the GSI, the subroutine `pcgsoi` performs the inner loop iterations. This subroutine will apply preconditioning through the array `vprecond`, an estimate of the Hessian that does not change during the inner loop iterations. Then `pcgsoi` computes the search direction. It accounts for nonlinearity by using the Polak-Ribiere method to compute $\mathbf{d}_{i-1}$. This means that the search direction is updated as

$$\mathbf{d}_i = \mathbf{r}_i + \beta_i\mathbf{d}_{i-1}, \tag{2}$$

$$\beta_i = \max\left\{\frac{\mathbf{r}_i^T(\mathbf{r}_i - \mathbf{r}_{i-1})}{\mathbf{r}_{i-1}^T\mathbf{r}_{i-1}}, 0\right\}, \tag{3}$$

where $\mathbf{r}_i$ is the gradient of the cost function at iteration $i$. This value of $\beta$ is computed in `pcgsoi` as `b=gnorm(2)/gsave`. The values of the gradients are in `gradx%values` and `grady%values`, which ultimately are computed in `stpcalc` and stored in `xhatsave%values` and `yhatsave%values` (the $x$ and $y$ gradients are averaged in `pcgsoi` to give the actual gradient). Gradient $r_i$ is updated in stpcalc as

$$\mathbf{r}_i = \mathbf{r}_{i-1} - \alpha_{i-1}\mathbf{d}_{i-1}. \tag{4}$$

Next, `pcgsoi` calls the subroutine `stpcalc` to calculate the step size. This subroutine estimates $\alpha_i$ iteratively, using a maximum of five iterations (`istp_iter`=5). It will loop though all the terms that contribute to the cost function (background, moisture constraint, dry pressure constraint, all types of observations, etc). For the observation terms, `stpcalc` will call subroutine `stpjo`, which will call the `stp` routine associated with each observation type, for example `stpt` for temperature, `stprad` for radiance, and so on. Once the computation of the step size $\alpha_i$ is complete, `pcgsoi` can move on to the next inner iteration, assuming that there are no minimization issues.

## 2 Step Size Calculation

At each inner loop iteration $i$, the subroutine `stpcalc` will perform a maximum five iterations $j = 1:5$ to calculate the step size $\alpha_i$. For simplicity, drop the subscript $i$. A quadratic function is fitted to the penalty associated with each term of the cost function,

$$q(\alpha) = c\alpha^2 + 2b\alpha + a. \tag{5}$$

The idea is to find the step size that minimizes the cost function along the search direction $\mathbf{d}$. This will occur at $\alpha = -b/c$. The penalty is computed for three different step sizes, $\alpha$, $(1-\delta)\alpha$, and $(1+\delta)\alpha$, where $\delta = 0.1$. In `stpcalc`, these three values are stored in `sges`, $\delta$ is `dels` and $\alpha$ at iteration $j$ is `stp(ii)`. The three values of $q(\alpha)$ are computed in the the appropriate `stp` routines, for example `stpt` for temperature. Subroutine `stpt` will loop over all of the temperature observations passing quality control, and sum up the total temperature penalties with respect to these three step sizes. This penalty is a summation over all observations,

$$\sum \frac{1}{\sigma^2}[H(x+\alpha d)]^2, \tag{6}$$

assuming no correlated error ($x$ is actually OMB). If variational quality control is turned on, it is applied here. The output of the `stp` routines is the array `out`, which stores

$$[q(\alpha), q((1-\delta)\alpha) - q(\alpha), q((1+\delta)\alpha) - q(\alpha)]. \tag{7}$$

In `stpcalc` this ultimately ends up in the array `pbc`. The individual types of observations and terms are kept separate in `pbc`. Read the comments at the beginning of `stpcalc` to know what these components are.

Once the three values of $q(\alpha)$ are computed, the new step size can be computed. Search for `bcoef` to see where this computation is done. Here `pbc(2,i)-pbc(3,i)` simplifies to $-4c\delta\alpha^2 - 4b\delta\alpha$

$$\begin{aligned} q((1-\delta)\alpha) - q(\alpha) - [q((1+\delta)\alpha) - q(\alpha)] &= c(1-\delta)^2\alpha^2 + 2b(1-\delta)\alpha + a - \left[c(1+\delta)^2\alpha^2 + 2b(1+\delta)\alpha + a\right] \\ &= c(1-2\delta+\delta^2)\alpha^2 - c(1+2\delta+\delta^2)\alpha^2 - 4b\delta\alpha \\ &= -4c\delta\alpha^2 - 4b\delta\alpha. \end{aligned}$$

Also, `pbc(2,i)+pbc(3,i)` simplifies to $2c\delta^2\alpha^2$

$$\begin{aligned} q((1-\delta)\alpha) - q(\alpha) + (q((1+\delta)\alpha) - q(\alpha)) &= c[(1-\delta)^2 - 1]\alpha^2 - 2b\delta\alpha + \left(c[(1+\delta)^2 - 1]\alpha^2 + 2b\delta\alpha\right) \\ &= c(-2\delta+\delta^2)\alpha^2 + c(2\delta+\delta^2)\alpha^2 \\ &= 2c\delta^2\alpha^2. \end{aligned}$$

These are multiplied by $1/4\delta\alpha$ and $1/2\alpha^2\delta^2$, respectively (`bcoef` and `ccoef`). Then the step size is updated `stp(ii)=stp(ii)+bx/cx`, which simplifies to $\alpha = -b/c$. This calculation is repeated at most four more times before moving onto the next inner loop iteration. If two consecutive estimates of $\alpha$ are very close, the iterative calculation of $\alpha$ will terminate.

# 3 Minimization Issues

Because the penalty is being approximated by the quadratic $q(\alpha) = c\alpha^2 + 2b\alpha + a$, $c$ is an approximation of the Hessian. The values of $c$ for each term contributing to the cost function are stored in `csum` in `stpcalc`, and their sum is stored in the scalar `cx`. If `cx` is negative, the Hessian is not positive definite, most likely a result of the variational quality control. The minimization will not automatically terminate if this happens; `stpcalc` will look at all estimates of the cost function during the five iterations of $\alpha$ and if there is a step size that does produce a cost function that is smaller than what it was at inner loop $i-1$, it will go with that step size. But if all five iterations increase the cost, the minimization will terminate. The minimization will only terminate if `cx` is negative; an individual component of `csum` can be negative and not cause `cx` to be negative. It is easy to determine which term has caused the termination. Look at `fort.220` or the `gsistat` and search for `cx`. There will be a printout of $c$. Find the negative term, and consult the comments on `pbc` in `stpcalc` to see which type of observation this is.

Sometimes resets can occur. This happens when the cost function gradient norm at iteration $i$ is much larger than it was at iteration $i-1$. Usually this indicates a problem with a forward model. Determining which term causes a reset can be difficult, but it requires running the GSI in verbose mode. This will print out the $c$ terms in `fort.220` and the `gsistat` file. After running the GSI in verbose mode, look at the components of $c$ right before and after the reset. If one term is blowing up right after the reset, this is the likely culprit. Consult the comments on `pbc` in `stpcalc` to see what this term actually is.