Note that the code for the real data set does not estimate μ_p. We also use a single subscript i instead of ij because our algorithm does not make a distinction between time and location. There is no particular 'group' structure related to time or location.

Notation:
$\boldsymbol{\theta}_p = \{\boldsymbol{\beta}_p, \sigma_p^2\}$,
$\boldsymbol{\theta} = \{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_P\}$,
$\boldsymbol{\Lambda}_i = [\lambda_{1,i}, \ldots, \lambda_{P,i}]^T$,
$\boldsymbol{\Sigma}_\epsilon = \text{diag}(\sigma_1^2, \ldots, \sigma_P^2)$,
$\boldsymbol{\Sigma}_{y,i} = \boldsymbol{\Lambda}_i \boldsymbol{\Lambda}_i^T + \boldsymbol{\Sigma}_\epsilon$,
$\mathbf{y}_{.,i} = [y_{1,i}, \ldots, y_{P,i}]^T$,
$\mathbf{y}_{p,.} = [y_{p,1}, \ldots, y_{p,n}]^T$,
$\eta_i^{(r)} = \text{E}_{\boldsymbol{\theta}^{(r-1)}}(\eta_i | \mathbf{y}_{.,i})$,
$(\check{\eta}_i^2)^{(r)} = \text{Var}_{\boldsymbol{\theta}^{(r-1)}}(\eta_i | \mathbf{y}_{.,i})$,
$\boldsymbol{\omega}_p = [\omega_{p,u|v}, \omega_{p,v|u}, \omega_{p,t}]^T$,
$\boldsymbol{\rho}_p = [\rho_{p,u|v}, \rho_{p,v|u}, \rho_{p,t}]^T$,
$\tilde{\mathbf{X}}^{(r)} = \boldsymbol{\eta}^{(r)} \circ \mathbf{X}$, where $\boldsymbol{\eta}^{(r)} = [\eta_1^{(r)}, \ldots, \eta_n^{(r)}]^T$,
$\check{\mathbf{X}}^{(r)} = \check{\boldsymbol{\eta}}^{(r)} \circ \mathbf{X}$, where $\check{\boldsymbol{\eta}}^{(r)} = [(\check{\eta}_1^2)^{(r)}, \ldots, (\check{\eta}_n^2)^{(r)}]^T$,
$\mathbf{B}_{\boldsymbol{\omega}_p}^{(r)} = \begin{bmatrix} \mathbf{0} & \mathbf{B}_{\boldsymbol{\omega}_{p,uv}}^{(r)} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{B}_{\boldsymbol{\omega}_{p,t}}^{(r)} \end{bmatrix}$, where $(\mathbf{B}_{\boldsymbol{\omega}_{p,uv}}^{(r)})^T \mathbf{B}_{\boldsymbol{\omega}_{p,uv}}^{(r)} = \omega_{p,u|v}^{(r)} \mathbf{S}_{u|v} + \omega_{p,v|u}^{(r)} \mathbf{S}_{v|u}$,
$(\mathbf{B}_{\boldsymbol{\omega}_{p,t}}^{(r)})^T \mathbf{B}_{\boldsymbol{\omega}_{p,t}}^{(r)} = \omega_{p,t}^{(r)} \mathbf{S}_t$, and the first column is all zero.

Note that $\mathbf{S}_{u|v}$, $\mathbf{S}_{v|u}$, $\mathbf{S}_t$ are all assumed as transformed already with the Z matrices. So, we should use 'B_uv_z', 'B_vu_z', 'B_t_z' when constructing these penalty matrices.

E():
This module does the E-step. It uses $\boldsymbol{\theta}^{(r-1)}$, and returns $\eta_i^{(r)}$, $(\check{\eta}_i^2)^{(r)}$, and it also evaluates the log-likelihood, $\log \mathcal{L}\left(\boldsymbol{\theta}^{(r-1)} \middle| \mathbf{y}\right)$.

Formulas:

$$\eta_i^{(r)} = (\boldsymbol{\Lambda}_i^T)^{(r-1)} \left(\boldsymbol{\Sigma}_{y,i}^{(r-1)}\right)^{-1} \mathbf{y}_{.,i}$$

$$(\check{\eta}_i^2)^{(r)} = 1 - (\boldsymbol{\Lambda}_i^T)^{(r-1)} \left(\boldsymbol{\Sigma}_{y,i}^{(r-1)}\right)^{-1} \boldsymbol{\Lambda}_i^{(r-1)}$$

$$\log \mathcal{L}\left(\boldsymbol{\theta}^{(r-1)} \middle| \mathbf{y}\right) \propto -\frac{1}{2} \sum_{i=1}^{n} \left[\log \left|\boldsymbol{\Sigma}_{y,i}^{(r-1)}\right| + \mathbf{y}_{.,i}^T \left(\boldsymbol{\Sigma}_{y,i}^{(r-1)}\right)^{-1} \mathbf{y}_{.,i}\right]$$

Arguments:
(1) **mm_y**: the $n \times P$ data matrix, $\mathbf{y} = [\mathbf{y}_{.,1}, \ldots, \mathbf{y}_{.,n}]^T$.
(2) **mm_l**: the $n \times P$ matrix of factor loadings, $\left[\boldsymbol{\Lambda}_1^{(r-1)}, \ldots, \boldsymbol{\Lambda}_n^{(r-1)}\right]^T$.
(3) **mm_s**: the $P \times 1$ vector of the variance of random errors, $\left[(\sigma_1^2)^{(r-1)}, \ldots, (\sigma_P^2)^{(r-1)}\right]^T$.
(4) **mm_N**: the number of observations, n.

(5) **mm_eta**: the $n \times 2$ matrix that stores $\eta_i^{(r)}$ (the first column), $(\check{\eta}_i^2)^{(r)}$ (the second column).

(6) **mm_lkhd**: $2 \log \mathcal{L}\left(\boldsymbol{\theta}^{(r-1)} \middle| \mathbf{y}\right)$.

Internal Variables:

(1) **l_i**: $\mathbf{\Lambda}_i^{(r-1)}$.

(2) **inv_vy**: $\left(\mathbf{\Sigma}_{y,i}^{(r-1)}\right)^{-1}$.

(3) **y_i**: $\mathbf{y}_{.,i}$.

(4) **eta**: $\eta_i^{(r)}$.

(5) **eta2**: $(\check{\eta}_i^2)^{(r)}$.

M_QR():

This module does the first part of the M-step: QR decomposition.

Formulas:

$$\boldsymbol{\beta}_p^{(r)} = \left[(\tilde{\mathbf{X}}^T)^{(r)}\tilde{\mathbf{X}}^{(r)} + (\check{\mathbf{X}}^T)^{(r)}\check{\mathbf{X}}^{(r)} + (\mathbf{B}_{\boldsymbol{\omega}_p}^T)^{(r)}\mathbf{B}_{\boldsymbol{\omega}_p}^{(r)}\right]^{-1} (\tilde{\mathbf{X}}^T)^{(r)}\mathbf{y}_{p,.}$$

Arguments:

(1) **mm_y**: the $n \times 1$ vector, $\mathbf{y}_{p,.}$.

(2) **mm_X1**, **mm_X2**: $\tilde{\mathbf{X}}^{(r)}$, $\check{\mathbf{X}}^{(r)}$. We carry out QR decomposition on these two matrices.

(3) **mm_nb**: K, the dimension of $\boldsymbol{\beta}_p^{(r)}$.

(4) **mm_r1**, **mm_r2**: $\tilde{\mathbf{R}}^{(r)}$, $\check{\mathbf{R}}^{(r)}$, the two $\mathbf{R}$ matrices from QR decomposition of $\tilde{\mathbf{X}}^{(r)}$, $\check{\mathbf{X}}^{(r)}$.

(5) **mm_qy**: $\tilde{\mathbf{Q}}_1^T\mathbf{y}_{p,.}$, where $\tilde{\mathbf{Q}}_1$ is the first K columns of $\tilde{\mathbf{Q}}$, the $\mathbf{Q}$ matrix from QR decomposition of $\tilde{\mathbf{X}}^{(r)}$. In SAS, $\tilde{\mathbf{Q}}^T\mathbf{y}_{p,.}$ is directly output from 'call qr()', so we obtain the first K rows of $\tilde{\mathbf{Q}}^T\mathbf{y}_{p,.}$.

Note:

When we apply 'call qr()' on $\check{\mathbf{X}}^{(r)}$, we let it output $\check{\mathbf{Q}}_1^T\mathbf{J}_n$, even though we do not need this in our computation. We do this to avoid output the whole $n \times n$ $\check{\mathbf{Q}}$ matrix, which costs time.

M_SVD():

This module does the second part of the M-step: SVD decomposition and obtain $\boldsymbol{\theta}_p^{(r)}$, $\text{GCV}_p^{(r)}$ given $\eta_i^{(r)}$, $(\check{\eta}_i^2)^{(r)}$, $\boldsymbol{\omega}_p^{(r)}$.

Formulas:

$$\begin{bmatrix}\tilde{\mathbf{R}}^{(r)} \\ \check{\mathbf{R}}^{(r)} \\ \mathbf{B}_{\boldsymbol{\omega}_p}^{(r)}\end{bmatrix} = \begin{bmatrix}\mathbf{U}_1 \\ \mathbf{U}_2 \\ \mathbf{U}_3\end{bmatrix} \mathbf{D}\mathbf{V}^T$$

$$\mathbf{P}^{-1} = \mathbf{V}\mathbf{D}^{-1}$$

$$\mathbf{y}_p^* = \mathbf{U}_1^T\tilde{\mathbf{Q}}_1^T\mathbf{y}_{p,.}$$

$$\boldsymbol{\beta}_p^{(r)} = \mathbf{P}^{-1}\mathbf{y}_p^*$$

$$(\sigma_p^2)^{(r)} = \frac{1}{n}\left[\mathbf{y}_{p,\cdot}^T\mathbf{y}_{p,\cdot} - 2(\mathbf{y}_p^*)^T\mathbf{y}_p^* + (\mathbf{y}_p^*)^T(\mathbf{U}_1^T\mathbf{U}_1 + \mathbf{U}_2^T\mathbf{U}_2)\mathbf{y}_p^*\right]$$

$$\Delta_p^{(r)} = \|\mathbf{y}_{p,\cdot} - \tilde{\mathbf{X}}^{(r)}\boldsymbol{\beta}_p^{(r)}\|^2$$

$$\delta_p^{(r)} = n - \mathrm{tr}(\mathbf{U}_1^T\mathbf{U}_1)$$

$$\mathrm{GCV}_p^{(r)} = \frac{n\Delta_p^{(r)}}{[\delta_p^{(r)}]^2}$$

Arguments:

(1) **mm_RBm**: the input matrix for SVD decomposition, $\begin{bmatrix}\tilde{\mathbf{R}}^{(r)}\\ \check{\mathbf{R}}^{(r)}\\ \mathbf{B}_{\boldsymbol{\omega}_p}^{(r)}\end{bmatrix}$.

(2) **mm_y**: the $n \times 1$ vector, $\mathbf{y}_{p,\cdot}$.

(3) **mm_X1**: $\tilde{\mathbf{X}}^{(r)}$.

(4) **mm_qy**: $\tilde{\mathbf{Q}}_1^T\mathbf{y}_{p,\cdot}$, which is output from 'M_QR' through the argument 'mm_qy'.

(5) **mm_nb**: K, the dimension of $\boldsymbol{\beta}_p^{(r)}$.

(6) **mm_N**: n, the number of observations.

(7) **mm_Pm_inv**, **mm_UU1**, **mm_uqy**: $\mathbf{P}^{-1}$, $\mathbf{U}^T\mathbf{U}$, $\mathbf{y}_p^*$, respectively.

(8) **mm_dt1**, **mm_dt2**, **mm_GCV**: $\delta_p^{(r)}$, $\Delta_p^{(r)}$, $\mathrm{GCV}_p^{(r)}$, respectively.

(9) **mm_s**: $(\sigma_p^2)^{(r)}$.

(10) **mm_l**, **mm_b**: $\lambda_{p,0}^{(r)}$, $[(\boldsymbol{\beta}_{p,uv}^{(r)})^T, (\boldsymbol{\beta}_{p,t}^{(r)})^T]^T$, respectively.

Internal Variables:

(1) **Um**, **Dm**, **Vm**: are $\begin{bmatrix}\mathbf{U}_1\\ \mathbf{U}_2\\ \mathbf{U}_3\end{bmatrix}$, the $K \times 1$ vector that has the diagonal of $\mathbf{D}$, and the $K \times K$ orthogonal matrix $\mathbf{V}$, respectively.

(2) **UU2**: $\mathbf{U}_2^T\mathbf{U}_2$.

M_drho():

This module adjusts $\boldsymbol{\omega}_p^{(r)}$ until $\mathrm{GCV}_p^{(r)}$ actually decreases.

We use a quadratic approximation of $\mathrm{GCV}_p^{(r)}$ to find $\boldsymbol{\omega}_p^{(r)}$ that supposedly minimizes $\mathrm{GCV}_p^{(r)}$. The approximation is based on the first and second derivatives of $\mathrm{GCV}_p^{(r)}$ with regard to $\boldsymbol{\rho}_p$, the logarithm of $\boldsymbol{\omega}_p$. Sometimes the approximation is not good enough and the $\boldsymbol{\omega}_p$ that minimizes the approximated $\mathrm{GCV}_p^{(r)}$ does not even reduce the actual $\mathrm{GCV}_p^{(r)}$. In this case, we adjust $\boldsymbol{\omega}_p^{(r)}$ and check the $\mathrm{GCV}_p^{(r)}$ again.

This module does not calculate the derivatives, which are output from 'GCVr()'. It also does not evaluate $\mathrm{GCV}_p^{(r)}$ during the process, but it works in tandem with 'M_SVD()' and iteratively checks whether $\mathrm{GCV}_p^{(r)}$ has decreased. The adjustment of $\boldsymbol{\omega}_p^{(r)}$ happens in a 'withdrawing' fashion, which means we always move from the initial $\boldsymbol{\omega}_p$ and depend only on the derivatives of $\mathrm{GCV}_p^{(r)}$ evaluated at that $\boldsymbol{\omega}_p$.

When we go to $\boldsymbol{\omega}_p$ that does not decrease $\mathrm{GCV}_p^{(r)}$, it means we go too far and we move back accordingly.

Here are the detailed steps of how this module works. **Note that we work with $\boldsymbol{\rho}_p$, the logarithm of $\boldsymbol{\omega}_p$, because $\boldsymbol{\omega}_p$ is a vector of positive smoothing parameters and is thus constrained.**

(1) Let $[\boldsymbol{\rho}_p]_0$ be the initial $\boldsymbol{\rho}_p$.

Before starting, we should have (a) $\mathrm{GCV}_p^{(r)}$ evaluated at $[\boldsymbol{\rho}_p]_0$, denoted by $[\mathrm{GCV}_p^{(r)}]_0$, (b) the derivatives of $\mathrm{GCV}_p^{(r)}$ evaluated at $[\boldsymbol{\rho}_p]_0$, (c) $[\boldsymbol{\rho}_p]_1$, the new $\boldsymbol{\rho}_p$ obtained using the derivatives, (d) $\mathrm{GCV}_p^{(r)}$ evaluated at $[\boldsymbol{\rho}_p]_1$, denoted by $[\mathrm{GCV}_p^{(r)}]_1$.

(2) We compare $[\mathrm{GCV}_p^{(r)}]_1$ with $[\mathrm{GCV}_p^{(r)}]_0$. If $[\mathrm{GCV}_p^{(r)}]_1 < [\mathrm{GCV}_p^{(r)}]_0$, then $\boldsymbol{\rho}_p^{(r)} = [\boldsymbol{\rho}_p]_1$. If not, we use the following two alternative ways to 'move back' and get a new $\boldsymbol{\rho}_p$.

(3) Let $[\boldsymbol{\rho}_p]_2$ be the new $\boldsymbol{\rho}_p$.

Option1: $[\boldsymbol{\rho}_p]_2 = [\boldsymbol{\rho}_p]_0 + \kappa\left([\boldsymbol{\rho}_p]_1 - [\boldsymbol{\rho}_p]_0\right)$.

Option2: $[\boldsymbol{\rho}_p]_2 = [\boldsymbol{\rho}_p]_0 - \kappa d[\nabla\mathrm{GCV}_p^{(r)}]_0$, where $d = \frac{\|[\boldsymbol{\rho}_p]_1 - [\boldsymbol{\rho}_p]_0\|}{\|[\nabla\mathrm{GCV}_p^{(r)}]_0\|}$ and $[\nabla\mathrm{GCV}_p^{(r)}]_0$ is the gradient of $\mathrm{GCV}_p^{(r)}$ $\boldsymbol{\rho}_p$ evaluated at $[\boldsymbol{\rho}_p]_0$.

Both options obtain $[\boldsymbol{\rho}_p]_2$ by moving from $[\boldsymbol{\rho}_p]_0$ in a smaller distance than before when $\kappa < 1$ (in the code we use $\kappa = \frac{1}{2}$). Option1 follows in the direction suggested by the 'quadratic approximation', that is, the direction from $[\boldsymbol{\rho}_p]_0$ to $[\boldsymbol{\rho}_p]_1$. Option2 uses another route by following the 'steepest descent', $-[\nabla\mathrm{GCV}_p^{(r)}]_0$. d is used to adjust the size of the descent and we make $\|d[\nabla\mathrm{GCV}_p^{(r)}]_0\| = \|[\boldsymbol{\rho}_p]_1 - [\boldsymbol{\rho}_p]_0\|$.

(4) Then we go back to Step (2), replacing $[\boldsymbol{\rho}_p]_1$ with $[\boldsymbol{\rho}_p]_2$. But before repeating the whole process, we exit this module and use 'M_SVD()' again, so that we can have the item (d) listed in Step (1). Items (a) and (b) are unchanged. Item (c) is updated with $[\boldsymbol{\rho}_p]_2$, as said before.

We first use Option1 to adjust $\boldsymbol{\rho}_p^{(r)}$. If that does not work after a couple of times, then it suggests the 'quadratic approximation' is poor and we switch to Option2. In the code Option1 is replaced after three times.

Another thing to note is this module only gives us $\boldsymbol{\omega}_p^{(r)}$ that decreases but not necessarily minimizes $\mathrm{GCV}_p^{(r)}$. We find the minimum by using another loop to obtain a series of $\boldsymbol{\omega}_p^{(r)}$ that decreases $\mathrm{GCV}_p^{(r)}$ successively until it reaches the minimum. During each iteration of this loop, we replace the item (a) in Step (1) with $\boldsymbol{\rho}_p^{(r)}$ from the last iteration. Items (b) – (d) are obtained by calling for 'GCVr()', 'M_rho()', 'M_SVD()' before starting the iteration (see below the notes on mm_ir for more explanation).

Arguments:

(1) **mm_rho**, **mm_rho0**: $[\boldsymbol{\rho}_p]_1$, $[\boldsymbol{\rho}_p]_0$, respectively.

(2) **mm_GCV**, **mm_GCV0**: $[\mathrm{GCV}_p^{(r)}]_1$, $[\mathrm{GCV}_p^{(r)}]_0$, respectively.

(3) **mm_GCVr0**: $[\nabla\mathrm{GCV}_p^{(r)}]_0$, which is the same as the first derivative of $\mathrm{GCV}_p^{(r)}$ with regard to $\boldsymbol{\rho}_p$, output from 'GCVr()'.

(4) **mm_GCVm**: is an indicator that indicates if $[\mathrm{GCV}_p^{(r)}]_1 > [\mathrm{GCV}_p^{(r)}]_0$. When mm_GCVm = 1, the adjusting stops.

(5) **mm_drho**: is d, used in Option2.

(6) **mm_opt**: is an indicator that indicates whether Option1 or Option2 is used.

(7) **mm_ir**: is the index of the 'outer' loop that successively decreases $\text{GCV}_p^{(r)}$, as stated before. During each iteration of this loop, we obtain a $\boldsymbol{\omega}_p^{(r)}$ that reduces the $\text{GCV}_p^{(r)}$ further. **Note that no adjusting occurs during the first iteration of this loop, that is, mm_GCVm is automatically set to 1.** This is because before adjusting starts, we need all the items (a) – (d) listed in Step 1, and this requires calling for 'GCVr()', 'M_rho()', 'M_SVD()'. However, for the first iteration, we only have (a), which is $\boldsymbol{\omega}_p^{(r-1)}$. So, we pass this module immediately in order to obtain items (b), (c) from 'GCVr()', 'M_rho()'. Item (d) is created when we go to the second iteration and reach 'M_SVD()' first before using this module to start adjusting. This problem only occurs at the first iteration. For subsequent iterations, we will always have run 'GCVr()', 'M_rho()' at the end of the last iteration. (We put 'GCVr()', 'M_rho()' after 'M_SVD()' in the algorithm because 'GCVr()' depends on the output items from 'M_SVD()'.)

(8) **mm_ir2**: is the index of the 'inner' loop that iterates between 'M_SVD' and this module to carry out the adjusting. When mm_ir2 = 3 we set mm_opt = 2 so that in the fourth iteration we will use Option2.

GCVr():

This module computes the first and second derivatives of $\text{GCV}_p^{(r)}$ with regard to $\boldsymbol{\rho}_p$.

We use $\mathcal{V}$ as $\text{GCV}_p^{(r)}$ in the formulas below and we use ρ_a, ρ_b (omitting the subscript p) as the a-th and b-th entries of $\boldsymbol{\rho}_p$ (the same notation is applied to $\boldsymbol{\omega}_p$). We also let $\mathbf{m}$ be a column vector whose a-th entry is $\frac{\partial \mathcal{V}}{\partial \rho_a}$ and $\mathbf{M}$ be a matrix whose (a,b)-th entry is $\frac{\partial^2 \mathcal{V}}{\partial \rho_a \partial \rho_b}$. Δ and δ refer to $\Delta_p^{(r)}$ and $\delta_p^{(r)}$.

Since $\boldsymbol{\rho}_p$ is only associated with the penalty, We now look at the derivatives of the composite penalty matrix (the following part is mainly to explain how we derive the matrix $\mathbf{P}_a$ used in the formulas),

$$\mathbf{S}_{\boldsymbol{\omega}_p}^{(r)} = (\mathbf{B}_{\boldsymbol{\omega}_p}^{(r)})^T \mathbf{B}_{\boldsymbol{\omega}_p}^{(r)} = \begin{bmatrix} 0 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \omega_{p,u|v}\mathbf{S}_{u|v} + \omega_{p,v|u}\mathbf{S}_{v|u} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \omega_{p,t}\mathbf{S}_t \end{bmatrix}.$$

Listed below are the three first derivatives. **Note that the penalty matrices in these derivatives are in the same position as they are in the composite penalty matrix.**

$$\frac{\partial \mathbf{S}_{\boldsymbol{\omega}_p}^{(r)}}{\partial \rho_{p,u|v}} = \omega_{p,u|v} \begin{bmatrix} 0 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{S}_{u|v} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}, \qquad \frac{\partial \mathbf{S}_{\boldsymbol{\omega}_p}^{(r)}}{\partial \rho_{p,v|u}} = \omega_{p,v|u} \begin{bmatrix} 0 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{S}_{v|u} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix},$$

$$\frac{\partial \mathbf{S}_{\boldsymbol{\omega}_p}^{(r)}}{\partial \rho_{p,t}} = \omega_{p,t} \begin{bmatrix} 0 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{S}_t \end{bmatrix}.$$

In the following computation, these derivatives are all in the form of $(\mathbf{P}^{-1})^T \frac{\partial \mathbf{S}_{\boldsymbol{\omega}_p}^{(r)}}{\partial \rho_a} \mathbf{P}^{-1}$. Since $\frac{\partial \mathbf{S}_{\boldsymbol{\omega}_p}^{(r)}}{\partial \rho_a}$ is a block diagonal matrix with only one non-zero block, we only need to use those rows in $\mathbf{P}^{-1}$ that match the non-zero diagonal block in $\frac{\partial \mathbf{S}_{\boldsymbol{\omega}_p}^{(r)}}{\partial \rho_a}$. For

example, in the case of $\frac{\partial \mathbf{S}_{\boldsymbol{\omega}_p}^{(r)}}{\partial \rho_{p,t}}$, we extract the last K_t rows of $\mathbf{P}^{-1}$ (K_t being the number of columns in the square matrix $\mathbf{S}_t$) as a new $K_t \times K$ matrix, $\tilde{\mathbf{P}}_t$. Then we have $(\mathbf{P}^{-1})^T \frac{\partial \mathbf{S}_{\boldsymbol{\omega}_p}^{(r)}}{\partial \rho_{p,t}} \mathbf{P}^{-1} = \omega_{p,t} \tilde{\mathbf{P}}_t^T \mathbf{S}_t \tilde{\mathbf{P}}_t$.

In general, we partition $\mathbf{P}^{-1}$ according to the block diagonal structure of the composite penalty matrix, which in our case becomes

$$\mathbf{P}^{-1} = \begin{bmatrix} \tilde{\mathbf{P}}_0 \\ \tilde{\mathbf{P}}_{uv} \\ \tilde{\mathbf{P}}_t \end{bmatrix}.$$

The three partitions are $1 \times K$, $K_{uv} \times K$, $K_t \times K$ matrices, respectively (K_{uv} being the number of columns in $\mathbf{S}_{u|v}$ and $\mathbf{S}_{v|u}$). Then we calculate $(\mathbf{P}^{-1})^T \frac{\partial \mathbf{S}_{\boldsymbol{\omega}_p}^{(r)}}{\partial \rho_a} \mathbf{P}^{-1}$ as $\omega_a \tilde{\mathbf{P}}_a^T \mathbf{S}_a \tilde{\mathbf{P}}_a$, where $\mathbf{S}_a$ is the penalty matrix related to ω_a and $\tilde{\mathbf{P}}_a$ is the corresponding partition from $\mathbf{P}^{-1}$. According to the composite penalty matrix we are using,

$$\begin{cases} \omega_1 = \omega_{p,u|v}, & \mathbf{S}_1 = \mathbf{S}_{u|v}, & \tilde{\mathbf{P}}_1 = \tilde{\mathbf{P}}_{uv}, \\ \omega_2 = \omega_{p,v|u}, & \mathbf{S}_2 = \mathbf{S}_{v|u}, & \tilde{\mathbf{P}}_2 = \tilde{\mathbf{P}}_{uv}, \\ \omega_3 = \omega_{p,t}, & \mathbf{S}_3 = \mathbf{S}_t, & \tilde{\mathbf{P}}_3 = \tilde{\mathbf{P}}_t. \end{cases}$$

In the formulas below, we let $\mathbf{P}_a = \tilde{\mathbf{P}}_a^T \mathbf{S}_a \tilde{\mathbf{P}}_a$.

Formulas:

$$\frac{\partial \mathcal{V}}{\partial \rho_a} = \frac{n}{\delta^2} \left(\frac{\partial \Delta}{\partial \rho_a} - \frac{2\Delta}{\delta} \frac{\partial \delta}{\partial \rho_a} \right)$$

$$\frac{\partial^2 \mathcal{V}}{\partial \rho_a \partial \rho_b} = \frac{n}{\delta^2} \left(-\frac{2}{\delta} \frac{\partial \Delta}{\partial \rho_a} \frac{\partial \delta}{\partial \rho_b} + \frac{\partial^2 \Delta}{\partial \rho_a \partial \rho_b} - \frac{2}{\delta} \frac{\partial \delta}{\partial \rho_a} \frac{\partial \Delta}{\partial \rho_b} + \frac{6\Delta}{\delta^2} \frac{\partial \delta}{\partial \rho_a} \frac{\partial \delta}{\partial \rho_b} - \frac{2\Delta}{\delta} \frac{\partial^2 \delta}{\partial \rho_a \partial \rho_b} \right)$$

$$\frac{\partial \delta}{\partial \rho_a} = \omega_a \mathrm{tr}(\mathbf{P}_a \mathbf{U}_1^T \mathbf{U}_1)$$

$$\frac{\partial^2 \delta}{\partial \rho_a \partial \rho_b} = -2\omega_a \omega_b \mathrm{tr}(\mathbf{P}_b \mathbf{P}_a \mathbf{U}_1^T \mathbf{U}_1) + I(a=b) \frac{\partial \delta}{\partial \rho_a}$$

$$\frac{\partial \Delta}{\partial \rho_a} = 2\omega_a (\mathbf{y}_p^*)^T (\mathbf{P}_a - \mathbf{P}_a \mathbf{U}_1^T \mathbf{U}_1) \mathbf{y}_p^*$$

$$\begin{aligned} \frac{\partial^2 \Delta}{\partial \rho_a \partial \rho_b} &= -2\omega_a \omega_b (\mathbf{y}_p^*)^T \left[\mathbf{P}_a \mathbf{P}_b - \mathbf{P}_a \mathbf{P}_b \mathbf{U}_1^T \mathbf{U}_1 + \mathbf{P}_b \mathbf{P}_a - \mathbf{P}_b \mathbf{P}_a \mathbf{U}_1^T \mathbf{U}_1 \right. \\ &\quad \left. - \mathbf{P}_a \mathbf{U}_1^T \mathbf{U}_1 \mathbf{P}_b \right] \mathbf{y}_p^* + I(a=b) \frac{\partial \Delta}{\partial \rho_a} \end{aligned}$$

Arguments:

(1) **mm_nb**: K, the dimension of $\boldsymbol{\beta}_p^{(r)}$, also the number of columns in $\mathbf{P}^{-1}$.

(2) **mm_Bvec**, **mm_Bidx**: 'mm_Bvec' is a column vector that has all the entries of the 'half' penalty matrices (see the 'Matrix' documentation of Code 01b2 on 'B_vec' and 'B_vec_idx'). 'mm_Bidx' is a matrix of indices that tells the module how to restore the penalty matrices from 'mm_Bvec' and how to partition $\mathbf{P}^{-1}$.

We use these when we construct each $\mathbf{P}_a$ and then calculate the derivatives in a loop. Let N_ω be the number of smoothing parameters in $\boldsymbol{\omega}_p$. 'mm_Bidx' is then a $N_\omega \times 5$ matrix, and its a-th row contains five indices that are related to $\mathbf{P}_a$.

For the current model, we have $a = 1, 2, 3$, and the respective $\mathbf{S}_a$ matrices are $\mathbf{S}_1 = \mathbf{S}_{u|v}$, $\mathbf{S}_2 = \mathbf{S}_{v|u}$, $\mathbf{S}_3 = \mathbf{S}_t$. We retrieve these from 'mm_Bvec' using the first three indices from 'mm_Bidx', namely, the $(a, 1)$, $(a, 2)$, $(a, 3)$-th entries of 'mm_Bidx'. Then we obtain $\tilde{\mathbf{P}}_a$ from $\mathbf{P}^{-1}$ with the last two indices, the $(a, 4)$, $(a, 5)$-th entries of 'mm_Bidx'. The row indices of $\mathbf{P}^{-1}$ from 2 to $K_{uv} + 1$ give us the block of $\tilde{\mathbf{P}}_{uv}$, and the $\tilde{\mathbf{P}}_t$ block is from the $(K_{uv} + 2)$-th to the $(K_{uv} + K_t + 1)$-th rows of $\mathbf{P}^{-1}$. Thus, the last two columns of 'mm_Bidx' should be $\begin{bmatrix} 2 & K_{uv}+1 \\ 2 & K_{uv}+1 \\ K_{uv}+2 & K_{uv}+K_t+1 \end{bmatrix}$.

(3) **mm_Pm_inv**, **mm_UU1**, **mm_uqy**, **mm_dt1**, **mm_dt2**:

are $\mathbf{P}^{-1}$, $\mathbf{U}^T\mathbf{U}$, $\mathbf{y}_p^*$, $\delta_p^{(r)}$, $\Delta_p^{(r)}$, respectively. These are output from 'M_SVD()'.

(4) **mm_rho**: $\boldsymbol{\rho}_p$.

(5) **mm_GCVr**, **mm_GCVrr**: are $\mathbf{m}$ and $\mathbf{M}$, respectively. Compared with the formulas, a factor of $\frac{n}{\delta^2}$ is taken out from both 'mm_GCVr' and 'mm_GCVrr' because eventually it will be cancelled out when we compute $\boldsymbol{\omega}_p^{(r)}$.

Internal Variables:

(1) **om**: is $\boldsymbol{\omega}_p = \exp(\boldsymbol{\rho}_p)$.

(2) **Nf**: N_ω, the number of smoothing parameters in $\boldsymbol{\omega}_p$.

(3) **Bm**, **BP_i**, **P_i**:

'Bm' is $\mathbf{B}_a$, the 'half' penalty matrix of $\mathbf{S}_a$, restored using 'btran()'. We first extract from 'mm_Bvec' the section related to $\mathbf{B}_a$ through the first two indices of 'mm_B_idx', that is, its $(a, 1)$, $(a, 2)$-th entries. The $(a, 3)$-th entry has the number of columns in $\mathbf{B}_a$.

'BP_i' is $\mathbf{B}_a\tilde{\mathbf{P}}_a$, and we obtain $\tilde{\mathbf{P}}_a$ from 'mm_Pm_inv' ($\mathbf{P}^{-1}$) using the $(a, 4)$, $(a, 5)$-th entries of 'mm_B_idx' (described above in the documentation of 'mm_B_idx'). 'P_i' is $\mathbf{P}_a = \tilde{\mathbf{P}}_a^T\mathbf{S}_a\tilde{\mathbf{P}}_a = (\mathbf{B}_a\tilde{\mathbf{P}}_a)^T(\mathbf{B}_a\tilde{\mathbf{P}}_a)$.

(4) **PS**, **pxi**, **P_y**, **PUU**, **PUU_y**, **y_PUU**:

'PS' is a $K \times N_\omega K$ matrix that collects all the $\mathbf{P}_a$ matrices side by side as the loop moves along. ('P_i' is freed from the memory after one iteration ends and is created anew in the next iteration.) We keep them all because they are needed in the computation of the second derivatives. 'pxi' is the index that lets us keep track of each 'P_i' in 'PS'.

'P_y', 'PUU_y' are both $K \times N_\omega$ matrices that respectively collect $\mathbf{P}_a\mathbf{y}_p^*$, $\mathbf{P}_a\mathbf{U}_1^T\mathbf{U}_1\mathbf{y}_p^*$ as their a-th columns during the a-th iteration. All of them, like all the $\mathbf{P}_a$, will later be used in the computation of the second derivatives.

We also calculate 'PUU' and 'y_PUU', which are $\mathbf{P}_a\mathbf{U}_1^T\mathbf{U}_1$ and $(\mathbf{y}_p^*)^T\mathbf{P}_a\mathbf{U}_1^T\mathbf{U}_1$. 'PUU' is used as an intermediate variable in the calculations of both 'PUU_y' and 'y_PUU'. 'y_PUU' is needed only in the current iteration, so unlike 'P_y', 'PUU_y', it does not have to be collected in a matrix.

(5) **dt1r**, **dt2r**, **dt1rr**, **dt2rr**:

'dt1r' and 'dt2r' are $N_\omega \times 1$ column vectors whose a-th entries are $\frac{\partial\delta}{\partial\rho_a}$ and $\frac{\partial\Delta}{\partial\rho_a}$, respectively. 'dt1rr' and 'dt2rr' are $N_\omega \times N_\omega$ matrices whose (a, b)-th entries are $\frac{\partial\delta}{\partial\rho_a\partial\rho_b}$ and $\frac{\partial\Delta}{\partial\rho_a\partial\rho_b}$, respectively.

Since the second derivative matrices are symmetric, we only compute the lower triangular part. So within the a-th iteration, we use a second loop that fills in the (a, b)-th entry ($b \leq a$) of the second derivative matrices.

(6) **GCVr**, **GCVrr**: are basically the same as 'mm_GCVr' and 'mm_GCVrr' ('mm_GCVr' is just 'GCVr'). But sometimes the second derivatives $\frac{\partial^2 \mathcal{V}}{\partial \rho_a^2}$ can be negative. In this case the 'quadratic approximation' would lead us to the maximum of the approximated $\text{GCV}_p^{(r)}$. We 'fix' this problem by simply changing the negative sign to positive.

M_rho:

This module outputs the new $\boldsymbol{\rho}_p^{(r)}$ based on the last adjusted $\boldsymbol{\rho}_p^{(r)}$ (denoted by $\boldsymbol{\rho}_p^{(r')}$), the $\mathbf{m}$ and $\mathbf{M}$ matrices (see the documentation of 'M_drho()' for details about how $\boldsymbol{\rho}_p^{(r)}$ is updated).

Formulas:

$$\boldsymbol{\rho}_p^{(r)} = \boldsymbol{\rho}_p^{(r')} - \mathbf{M}^{-1}\mathbf{m}$$

Arguments:

(1) **mm_rho**: $\boldsymbol{\rho}_p^{(r')}$.

(2) **mm_GCVr**, **mm_GCVrr**: are $\mathbf{m}$ and $\mathbf{M}$ matrices, respectively.

(3) **mm_min**, **mm_max**: set the lower and upper bounds of $\boldsymbol{\rho}_p^{(r)}$.