type cv_algorithmic	cV_A%	! ALGORITHMIC CONTROL VARIABLES				
Variable type	*Variable name*	*Description*	*Defaults*	*Block in Input file*	*Name in Input file*	*Note*
double precision	structural_conv	Structural parameter convergence values	0.001	algorithmic_cv	structural_conv	
double precision	phi_conv	Objective function convergence value	0.001	algorithmic_cv	phi_conv	
double precision	bga_conv	Geostatistical method (more external loop) convergence value	0.001	algorithmic_cv	bga_conv	
integer	it_max_structural	Max number of iterations for struct parameters	10	algorithmic_cv	it_max_structural	
integer	it_max_phi	Max number of iterations for objective function	10	algorithmic_cv	it_max_phi	
integer	it_max_bga	Max number of iterations for geostatistical method	10	algorithmic_cv	it_max_bga	
integer	lns_flag	Linesearch procedure flag: [0] not perform [1] perform	0	algorithmic_cv	linesearch	
integer	it_max_lns	Max number of iterations for linesearch procedure	10	algorithmic_cv	it_max_linesearch	
logical	store_Q	TRUE --> Store Q FALSE --> Not store Q --> We need to address this option	TRUE	algorithmic_cv	*Not read*	Right now we always store Q. Store_Q is set to TRUE by default and it is not possible to change the value from outside.
integer	theta_cov_form	Form of theta covariance: [0] none, [1] diag, [2] full matrix	0	algorithmic_cv	theta_cov_form	
integer	deriv_mode	derivatives (Jacobian) calculation method: [0] make PEST files internally, [1] use secondary command line argumern (typically adjoint state)	0	algorithmic_cv	deriv_mode	
integer	Q_compression_flag	[0] none - calculate full Q0, [1] Calculate Q0 for each beta separately and if nugget store just 1, if toep_flag store just a vector	0	algorithmic_cv	Q_compression_flag	
end type cv_algorithmic						

type d_algorithmic	d_A%	! ALGORITHMIC "GLOBALS"				
Variable type	*Variable name*	*Description*	*Defaults*	*Block in Input file*	*Name in Input file*	*Note*
double precision, pointer	H(:,:)	Sensitivity matrix	-	-	-	
double precision, pointer	HX(:,:)	H*X	-	-	-	
double precision, pointer	HQHt(:,:)	H*Qss*Ht	-	-	-	
double precision, pointer	Hsold(:)	H*d_PAR%pars_old	-	-	-	
double precision, pointer	Qsy(:,:)	QHt is the cross covariance between s and y	-	-	-	
double precision, pointer	Qyy(:,:)	HQHt + R (Auto-covariance matrix of the observ y)	-	-	-	
double precision, pointer	beta_hat(:)	Estimated means	-	-	-	
double precision, pointer	ksi(:)	ξ	-	-	-	
end type d_algorithmic						

type cv_prior_mean	cv_PM%	! PRIOR MEANS CONTROL VARIABLES				
Variable type	*Variable name*	*Description*	*Defaults*	*Block in Input file*	*Name in Input file*	*Note*
integer	betas_flag	Have or not prior informations about mean? [0] No - [1] Yes	0	prior_mean_cv	prior_betas	
integer	Qbb_form	Form of Beta covariance: [0] none, [1] diag, [2] full matrix	0	prior_mean_cv	beta_cov_form	
end type cv_prior_mean						

type d_prior_mean	d_PM%	! DATA FOR PRIOR MEANS				
Variable type	*Variable name*	*Description*	*Defaults*	*Block in Input file*	*Name in Input file*	*Note*

double precision, pointer	beta_0(:)	Prior beta values	-	prior_mean_data	beta_0	The value in the table must be entered in ascending order of BetaAssoc. First row --> BetaAssoc=1, Second row --> BetaAssoc=2, ...
double precision, pointer	Qbb(:,:)	Covariance of beta	-	prior_mean_data	beta_cov_i i = 1, p	
double precision, pointer	InvQbb(:,:)	Inverse of covariance of beta	-	-	-	
double precision, pointer	InvQbbB0(:)	Inverse of covariance of beta * beta0	-	-	-	
integer, pointer	Partrans(:)	Vector of parameter transformation [1] Log [0] None - In the input file write NONE or LOG. (No case sensitive)	-	prior_mean_data	Partrans	If Partrans is equal to LOG, the corresponding beta_0 and covariance of beta values, must be entered LOG transformed by the user.
end type d_prior_mean						

type cv_struct	cv_S%	! CONTROL VARIABLES FOR STRUCTURAL PARAMETERS				
Variable type	*Variable name*	*Description*	*Defaults*	*Block in Input file*	*Name in Input file*	*Note*
integer, pointer	prior_cov_mode(:)	Supplied matrix [0] or calculated [1].	1	structural_parameter_cv	prior_cov_mode	Right now, we always calculate the covariance matrix, independently of the value entered here. So 0 or 1 is indifferent. We don't have choice in the present version.
integer, pointer	var_type(:)	Type of variogram [0] pure nugget, [1] linear, [2] exponential	1	structural_parameter_cv	var_type	Linear means the limiting case of the exponential variogram type with fixed integral scale set to 10 times the maximum distance of nodes.
integer, pointer	struct_par_opt(:)	Structural parameters optimization: [0] No optimization, [1] Optimization	1	structural_parameter_cv	struct_par_opt	We can choose, individually, which parameters estimate. The structural parameters estimation is not addressed right now. There is the structural parameter estimation loop in the main program but empty.
integer, pointer	num_theta_type(:)	Number of structural parameters related to the var_type. 1 for pure nugget, 1 for linear, 2 for exponential	1	- Assigned based on var_type	num_theta_type	1 for pure nugget. 1 for linear. 2 for exponential. The pure nugget variogram type can be also used for lumped buffer zones (set the theta_0_1 to 10-4 or a small positive value)
integer, pointer	trans_theta(:)	Transformation of structural parameters in the estimation space (power transform): [0] No, [1] Yes	1	structural_parameter_cv	trans_theta	**The value in the table must be entered in ascending order of BetaAssoc. First row --> BetaAssoc=1, Second row --> BetaAssoc=2, ...**
double precision, pointer	alpha_trans(:)	Exponent of power transformation in case of trans_theta	50	structural_parameter_cv	alpha_trans	
integer	num_theta_opt	Number of structural parameters to be optimized. This can include sigma (epistemic error)	0	-	-	Calculated in bxq_theta_cov_calcs
end type cv_struct						

type d_struct	d_S%	! DATA FOR STRUCTURAL PARAMETERS				
Variable type	*Variable name*	*Description*	*Defaults*	*Block in Input file*	*Name in Input file*	*Note*
double precision, pointer	theta_0(:,:)	Initial value of theta matrix	-	structural_parameters_data	theta_0_i i=1,…,num_theta_type	The max(num_theta_type) defines the column number of the theta_0 matrix. In case max(num_theta_type)=2 and num_theta_type=1, theta_0_2 must be negative otherwise there is a warning.
double precision, pointer	str_par_opt_vec_0(:)	initial values of all structural parameters to optimize. May include sigma -- made into a single vector	-	-	-	
double precision, pointer	str_par_opt_vec(:)	current values of all structural parameters to optimize. May include sigma -- made into a single vector	-	-	-	
double precision, pointer	theta_cov(:,:)	Theta covariance matrix	-	structural_parameters_cov	theta_cov_i i=1,…,max(num_thata_type)	
double precision, pointer	Qtheta(:,:)	Theta covaraince matrix (may include sig) for prior covariance for optimization	-	-	-	Combines active theta parameters from theta_cov with sigma (if requested). Made in bxq_theta_cov_calcs
double precision	sig_0	Initial value of sigma (epistemic uncertainty parameter)	-	epistemic_error_term	sig_0	
double precision	sig_p_var	Variance of sigma (variance of the epistemic error)	-	-	sig_p_var	
integer	sig_opt	Optimization for sig: [0] No, [1] Yes	-	epistemic_error_term	-	
double precision, pointer	theta(:,:)	Structural parameters matrix - Current iteration	-	-	-	
integer	trans_sig	Transformation of epistemic error in the estimation space (power transform): [0] No, [1] Yes	0	epistemic_error_term	trans_sig	
double precision	alpha_trans_sig	Exponent of power transformation in case of trans_sig	50	epistemic_error_term	alpha_trans	
double precision	sig	Epistemic uncertainty parameter - Current iteration	-	-	-	
double precision, pointer	struct_par_opt_vec_0(:)	Initial value of all structural parameter values to be optimized for. May include sigma.	-	-	-	
double precision, pointer	struct_par_opt_vec_(:)	Current value of all structural parameter values to be optimized for. May include sigma.	-	-	-	
double precision, pointer	invQtheta(:,:)	Inverse of prior covariance matrix of all structural parameter values to be optimized for. May include the inverse of variance of sigma.	-	-	-	
end type d_struct						

type cv_param	cv_PAR%	! CONTROL VARIABLES FOR PARAMETERS				
Variable type	*Variable name*	*Description*	*Defaults*	*Block in Input file*	*Name in Input file*	*Note*
integer	npargp	Number of parameter groups	-	-	-	npargp is equal to the row number of the *parameter_groups* block
integer	npar	Total number of parameters	-	-	-	npar is equal to the row number of the *parameter_data* block
character (len=50), pointer	grp_name(:)	Name of the parameter groups	-	parameter_groups	groupname	
integer, pointer	grp_type(:)	Type of groups	-	parameter_groups	grouptype	
integer	ndim	Spatial dimensions	-	parameter_cv	ndim	

integer	p	Number of means (is also the number of BetaAssociations)	-	-	-	p is equal to the row number of the *prior_mean_data* block
end type cv_param						

type Q0_compr	Q0_All(:)%	! CONTROL VARIABLE (TYPE) FOR COMPRESSION FORM OF Q (TOEPLITZ OR NOT) AND Q0		**Q0_All(:) it is a pointer (one for each BetaAssoc)**		This block is read only if cv_A%Q_compression_flag is not zero
Variable type	*Variable name*	*Description*	*Defaults*	*Block in Input file*	*Name in Input file*	*Note*
integer	BetaAss	Association of variables with the corresponding mean value according to BetaAss	-	Q_compression_cv	BetaAssoc	
integer	Toep_flag	Using Toeplitz matrix for Qss. [0] No, [1] Yes	-	Q_compression_cv	Toep_flag	
integer	Nrow	Number of model rows	-	Q_compression_cv	Nrow	Read only if Toep_flag = 1
integer	Ncol	Number of model columns	-	Q_compression_cv	Ncol	Read only if Toep_flag = 1
integer	Nlay	Number of model layers	-	Q_compression_cv	Nlay	Read only if Toep_flag = 1
integer	Npar	Number of parameters with same BetaAss. (one value for each BetaAss)	-	-	-	Calculated when the parameters are read from the *parameter_data* block
integer	Beta_Start	Identifies where in the parameter list, starts the value with the p-th beta association	-	-	-	Evaluated when the parameters are read from the *parameter_data* block
double precision, pointer	Q0_C(:,:)	Matrix Q0, one for each beta. Just if Q_compression_flag = 1 - A vector if Toep_flag = 0	-	-	-	Prior covariance is in block as a matrix for each beta or a vector for each beta if toepl_flag is 1 and 1 value for nugget.
end type Q0_compr						

type d_param	d_PAR%	! DATA FOR PARAMETERS				
Variable type	*Variable name*	*Description*	*Defaults*	*Block in Input file*	*Name in Input file*	*Note*
character (len=50), pointer	group(:)	Name of groups	-	parameter_data	GroupName	Name of the group where each parameter belongs. There is a control to avoid that the same beta corresponds to parameters of different type.
double precision, pointer	pars(:)	Current vector of parameters. At the beginning contains the StartValue.	-	parameter_data	StartValue	At the beginning pars is the vector of the initial values of the parameters as read in the *parameter_data* block. Then became the current best estimate.
character (len=50), pointer	parnme(:)	Name of parameter	-	parameter_data	ParamName	Names of parameters
double precision, pointer	pars_old(:)	Previous values of parameters - Sold	-	-	-	Parameter values at iteration (current-1)
double precision, pointer	pars_lns(:)	Vector of parameters used in the linesearch procedure	-	-	-	Vector of parameters used only in the linesearch procedure. It is allocated only if linesearch=1 in *algorithmic_cv* block
double precision, pointer	lox(:,:)	Location (coordinates)	-	parameter_data	xi i=1,…,ndim	
double precision	phi_T	Objective function - Total	-	-	-	Total objective function d_PAR%phi_T = d_PAR%phi_R + d_PAR%phi_M
double precision	phi_M	Objective function - Misfit	-	-	-	Misfit objective function phi_M = 1/2* (y-h(s))t * R^-1 * (y-h(s))

double precision	phi_R	Objective function - Regularization	-	-	-	Regularization objective function phi_R = 1/2 ksit * HQHt * ksi + 1/2 ksit * (HX) * Qbb * (HXt) * ksi
integer, pointer	SenMethod(:)	Sensitivity calculation method	-	parameter_data	SenMethod	Not used right now. Just read from the parameter block.
integer, pointer	BetaAssoc(:)	Faces association	-	parameter_data	BetaAssoc	Used to associate each parameter to the corresponding beta.
integer, pointer	Group_type(:)	Vector of group type for each parameter	-	-	-	The Group_Type for each parameter is assigned based on the GroupName read in the *parameter_data* block compared with the groupname and grouptype indicated in the *parameter_groups* block
end type d_param						

type cv_observ	cv_OBS%	! CONTROL VARIABLES FOR OBSERVATIONS				
Variable type	*Variable name*	*Description*	*Defaults*	*Block in Input file*	*Name in Input file*	*Note*
integer	nobsgp	Number of observations groups	-	-	-	nobsgp is equal to the row number of the*observation_groups* block
integer	nobs	Number of observations	-	-	-	nobs is equal to the row number of the *observation_data* block
character (len=50), pointer	grp_name(:)	Name of the observations groups	-	observation_groups	groupname	Read but not used right now.
end type cv_observ						

type d_observ	d_OBS	! DATA FOR OBSERVATIONS				
Variable type	*Variable name*	*Description*	*Defaults*	*Block in Input file*	*Name in Input file*	*Note*
character (len=50), pointer	group(:)	Name of groups	-	observation_data	GroupName	Read but not used right now.
double precision, pointer	obs(:)	Vector of observations	-	observation_data	ObsValue	
double precision, pointer	h(:)	Current model output (calculated values in the observation points)	-	-	-	
character (len=50), pointer	obsnme(:)	names of observations	-	observation_data	ObsName	
double precision, pointer	weight(:)	Weight for R matrix	-	observation_data	Weight	Used in the R0 matrix as d_XQR%R0(i,i) = 1./(d_OBS%weight(i)**2)
end type d_observ						

type d_comlin	d_MOD	! DATA FOR COMMAND LINE ARGUMENTS				
Variable type	*Variable name*	*Description*	*Defaults*	*Block in Input file*	*Name in Input file*	*Note*
character (len=50), pointer	com	Command line	-	model_command_lines	Command	
character (len=50), pointer	dercom	derivative Command line	-	model_command_lines	DerivCommand	
end type d_comlin						

type cv_minout	cv_MIO%	! CONTROL VARIABLES FOR MODEL i/o				
Variable type	*Variable name*	*Description*	*Defaults*	*Block in Input file*	*Name in Input file*	*Note*
integer	ninsfle	Number of instruction files	-	-	-	ninsfle is equal to the row number of *model_input_files* block

integer	ntplfle	Number of template files	-	-	-	ntplfle is equal to the row number of *model_ioutput_files* block
end type cv_minout						

type d_minout	d_MIO%	! DATA FOR MODEL i/o				
Variable type	*Variable name*	*Description*	*Defaults*	*Block in Input file*	*Name in Input file*	*Note*
character(len=100),pointer	tpl(:)	Template file	-	model_input_files	TemplateFile	The dimension of the vector is based on ninsfle
character(len=100),pointer	infle(:)	Input file	-	model_input_files	ModInFile	The dimension of the vector is based on ninsfle
character(len=100),pointer	ins(:)	Instruction file	-	model_output_files	InstructionFile	The dimension of the vector is based on ntplfle
character(len=100),pointer	outfle(:)	Output file	-	model_output_files	ModOutFile	The dimension of the vector is based on ntplfle
end type d_minout						

type kernel_XQR	d_XQR%	! KERNELS OF X, Q AND R				
Variable type	*Variable name*	*Description*	*Defaults*	*Block in Input file*	*Name in Input file*	*Note*
double precision, pointer	X(:,:)	Deterministic base functions (Right now just 1 to associate each parameter to the corresponding beta)	-	-	-	
double precision, pointer	Q0(:,:)	Prior covariance before structural parameters (1 for nugget, distances for linear or exponential variogram)	-	-	-	
double precision, pointer	R0(:,:)	Covariance matrix of epistemic error before sig (R=sig*R0) R0=1/(d_Obs%weight)^2	-	-	-	
double precision	L	10 times maximum distance in Q0 matrix	-	-	-	
end type kernel_XQR						