PhiPsi 子程序/函数功能和接口说明

1 以 PhiPsi 开头的子程序：输入信息读取及 PhiPsi 分支主程序

PhiPsi 程序中，与输入信息读取相关的子程序共有三个，分别是 *PhiPsi_Input_Filename*、*PhiPsi_Input_Keywords* 以及 *PhiPsi_Read_Input*。此外，根据分析类型的不同，PhiPsi 主程序调用不同的分支主程序，比如 *PhiPsi2D_Static*、*PhiPsi2D_I_Dynamic*、*PhiPsi2D_Static_HF*、*PhiPsi3D_Static* 等。这些以 PhiPsi 开头的子程序均无输入输出接口。表 1 汇总了这些子程序。

表 1 以 PhiPsi 开头的子程序说明（按名称排序）

子程序名	子程序功能
PhiPsi_Input_Filename	确定输入文件所在目录（*Work_Directory*）及输入文件的文件名（*Filename*），仅在以源码方式运行 PhiPsi（见前文 PhiPsi 运行方式 2 和运行方式 3）时有效
PhiPsi_Input_Keywords	该子程序中定义了程序的全部关键控制信息，仅在以源码方式（见前文 PhiPsi 运行方式 2 和运行方式 3）运行 PhiPsi 时有效
PhiPsi_Read_Input	从关键字文件读取 PhiPsi 控制信息，本子程序仅在以读取关键字文件的方式运行 PhiPsi（见前文 PhiPsi 运行方式 1 和运行方式 4）时有效
PhiPsi2D_I_Dynamic	PhiPsi 2D 隐式动态分析分支主程序
PhiPsi2D_Static	PhiPsi 2D 准静态分析分支主程序
PhiPsi2D_Static_HF	PhiPsi 2D 准静态水力压裂分析分支主程序
PhiPsi2D_Static_NonLinear	PhiPsi 2D 准静态非线性分析（塑性分析）分支主程序
PhiPsi3D_Static	PhiPsi 3D 准静态分析分支主程序

2 PhiPsi 主干子程序：确定增强节点、组集刚度矩阵、处理边界条件等

本小节的子程序为 PhiPsi 主干子程序，这些主干子程序共同构成 PhiPsi 分支主程序分析的基本框架，包括用于确定增强节点的 *Determine_Enriched_Nodes* 子程序、用于组集刚度矩阵的 *Assemble_Stiffness_Matrix_FEM* 子程序或 *Assemble_Stiffness_Matrix_XFEM* 子程序、用于处理边界条件的 *Boundary_Cond* 子程序等。表 2 汇总了这些子程序。需要说明的是，PhiPsi 程序中，变量 *isub*（或 *iter*）作为计数指示变量，在不同的子程序中有不同含义，有时表示载荷步号，有

时表示水力压裂分析的破裂步号，有时表示载荷子步号，有时表示迭代步号，PhiPsi 程序并未使用不同变量来区分。

表 2 以 PhiPsi 主干子程序用途及输入输出变量说明（按名称排序）

子程序名	子程序功能	子程序输入输出接口	
Assemble_Mass_Matrix_FEM	组集质量矩阵（用于 FEM 问题）	*Assemble_Mass_Matrix_FEM*(*isub*, *globalM*, *T_Freedom*)	
		isub	载荷子步号；整数；输入变量
		globalM(*T_Freedom*, *T_Freedom*)	质量矩阵；浮点数；输出变量
		T_Freedom	质量矩阵维数（自由度数目）；整数；输入变量
Assemble_Mass_Matrix_XFEM	组集质量矩阵（用于 XFEM 问题）	*Assemble_Mass_Matrix_XFEM*(*isub*, *c_globalM*, *c_Total_Freedom*)	
		isub	载荷子步号；整数；输入变量
		c_globalM(*c_Total_Freedom*,*c_Total_Freedom*)	质量矩阵；浮点数；输出变量
		c_Total_Freedom	质量矩阵维数（自由度数目）；整数；输入变量
Assemble_Stiffness_Matrix_FEM	组集刚度矩阵（用于 FEM 问题）	*Assemble_Stiffness_Matrix_FEM*(*isub*, *globalK*, *T_Freedom*, *Total_Num_G_P*)	
		isub	载荷子步号；整数；输入变量
		globalK(*T_Freedom*,*T_Freedom*)	刚度矩阵；浮点数；输出变量
		T_Freedom	刚度矩阵维数（自由度数目）；整数；输入变量
		Total_Num_G_P	总的高斯点数目；整数；输出变量
Assemble_Stiffness_Matrix_FEM_3D	组集刚度矩阵，并以全矩阵或稀疏矩阵的形式保存（用于 3D FEM 问题）	*Assemble_Stiffness_Matrix_FEM_3D*(*isub*, *globalK*, *globalK_COEF*, *globalK_JCOEF*, *globalK_Num_NZero*,*num_J_Col*)	
		isub	载荷子步号；整数；输入变量
		globalK(*Total_FD*,*Total_FD*)	刚度矩阵；浮点数；输出变量
		globalK_COEF(*Total_FD*,*num_J_Col*)	稀疏矩阵的元素值；浮点数；输出变量
		globalK_JCOEF(*Total_FD*,*num_J_Col*)	稀疏矩阵各行非零元素的列号；整数；输出变量
		globalK_Num_NZero(*Total_FD*)	稀疏矩阵各行非零元素的数目；整数；输出变量
		num_J_Col	稀疏矩阵存储器的最大列数；整数；输入变量
Assemble_Stiffness_Matrix_XFEM	组集刚度矩阵（用于	*Assemble_Stiffness_Matrix_XFEM*(*isub*, *c_globalK*, *c_Total_Freedom*, *Total_Num_G_P*)	

	XFEM 问题）	*isub*	载荷子步号；整数；输入变量
		c_globalK(*c_Total_Freedom*, *c_Total_Freedom*)	刚度矩阵；浮点数；输出变量
		c_Total_Freedom	自由度数目；整数；输入变量
		Total_Num_G_P	总的高斯点数目；整数；输出变量
Assemble_Stiffness_Matrix_XFEM_3D	组集刚度矩阵（用于 3D XFEM 问题）	*Assemble_Stiffness_Matrix_XFEM_3D*(*isub*, *c_globalK*, *c_Total_Freedom*, *Total_Num_G_P*)	
		isub	载荷子步号；整数；输入变量
		c_globalK(*c_Total_Freedom*, *c_Total_Freedom*)	刚度矩阵；浮点数；输出变量
		c_Total_Freedom	自由度数目；整数；输入变量
		Total_Num_G_P	总的高斯点数目；整数；输出变量
Boundary_Cond	处理位移边界条件	*Boundary_Cond*(*c_Total_FD*,*isub*, *freeDOF*,*c_num_FD*)	
		c_Total_FD	总的自由度数目；整数；输入变量
		isub	载荷子步号；整数；输入变量
		freeDOF(*c_Total_FD*)	活动自由度（无约束）向量；整数；输出变量
		c_num_FD	总的活动自由度（无约束）数目；整数；输出变量
Boundary_Cond_3D	处理位移边界条件（用于 3D 问题）	*Boundary_Cond_3D*(*c_Total_FD*, *isub*, *freeDOF*, *num_FreeD*)	
		c_Total_FD	总的自由度数目；整数；输入变量
		isub	载荷子步号；整数；输入变量
		freeDOF(*c_Total_FD*)	活动自由度（无约束）向量；整数；输出变量
		num_FreeD	总的活动自由度（无约束）数目；整数；输出变量
Boundary_Cond_HF	水力压裂分析流体压力边界条件处理（裂尖水压为零，故需要约束裂尖水压自由度）	*Boundary_Cond_HF*(*iter*, *freeDOF_HF*, *num_free_CalP*, *Local_freeDOF_HF*)	
		iter	载荷子步号；整数；输入变量
		freeDOF_HF(*num_Tol_CalP_Water*)	活动流体节点（流体压力非零）自由度向量（自由度编号在位移自由度基础上累加），其中 *num_Tol_CalP_Water* 为总的流体节点数目；整数；输出变量
		num_free_CalP	流体压力不为零的流体节点数目；整数；输出变量
		Local_freeDOF_HF(*num_Tol_CalP_Water*)	活动流体节点（流体压力非零）自由度向量（自由度编号不在位移自由度基础上累加），其中 *num_Tol_CalP_Water* 为总的流体节点数目；整数；输出变量
Check_Crack_Grows	根据最大周向拉应力准	*Check_Crack_Grows*(*ifra*, *iter*, *Yes_Grow*)	
		ifra	载荷步号；整数；输入变量

	则或加权平均最大主拉应力准则，检查裂缝是否发生了扩展和交汇（生成T型交叉），并确定扩展方向及新裂尖	*iter*	载荷子步号；整数；输入变量
		Yes_Grow	裂缝是否发生了扩展；逻辑型；输出变量
Clear_All_Results_Files	删除当前工作目录下的全部计算结果文件	注：（1）无输入输出变量；（2）本子程序用到了开源库 modfilesys（附录 J）	
Determine_Contact_State_by_Iteration	利用罚函数法迭代计算裂缝面接触状态	*Determine_Contact_State_by_Iteration*(*iter*, *ifra*, *Counter_Iter*, *Contact_DISP*, *c_Total_FD*, *usual_FD*, *enrich_FD*, *c_freeDOF*, *c_num_freeDOF*, *c_F*, *ori_globalK*, *CT_Jacobian*)	
		iter	载荷子步号；整数；输入变量
		ifra	载荷步号；整数；输入变量
		Counter_Iter	总的迭代数目计数器；整数；输入变量
		Contact_DISP(*c_Total_FD*)	位移向量；浮点数；输入输出变量
		c_Total_FD	总的自由度数目；整数；输入变量
		usual_FD	常规自由度数目；整数；输入变量
		enrich_FD	增强自由度数目；整数；输入变量
		c_freeDOF(*c_Total_FD*)	活动自由度向量；整数；输入变量
		c_num_freeDOF	活动自由度数目；整数；输入变量
		c_F(*c_Total_FD*)	外载荷向量；浮点数；输入变量
		ori_globalK(*c_Total_FD*,*c_Total_FD*)	原刚度矩阵；浮点数；输入变量
		CT_Jacobian(*c_Total_FD*,*c_Total_FD*)	接触迭代的雅可比矩阵；浮点数；输出变量
Determine_Enriched_Nodes	确定增强节点	*Determine_Enriched_Nodes*(*ifra*, *isub*)	
		ifra	载荷步号；整数；输入变量
		isub	载荷子步号；整数；输入变量
Determine_Enriched_Nodes_3D	确定增强节点（用于3D问题）	*Determine_Enriched_Nodes_3D*(*ifra*, *isub*)	
		ifra	载荷步号；整数；输入变量
		isub	载荷子步号；整数；输入变量
Determine_Enriched_Nodes_Cross_Check	确定增强节点过程中检查是否存在十字交叉型	*Determine_Enriched_Nodes_Cross_Check*(*ifra*, *isub*, *Yes_Cross_Ele)*	
		ifra	载荷步号；整数；输入变量
		isub	载荷子步号；整数；输入变量

	裂缝	*Yes_Cross_Ele*	是否存在十字交叉型裂缝；逻辑型；输出变量
Determine_Enriched_Nodes_Pre_Check	确定增强节点前的预检测	*Determine_Enriched_Nodes_Pre_Check*(*ifra*, *isub*)	
		ifra	载荷步号；整数；输入变量
		isub	载荷子步号；整数；输入变量
Get_Contact_State_Eles	获得单元接触状态	*Get_Contact_State_Eles*(*i_Contact*, *Yes_Contact_Con*)	
		i_Contact	接触迭代步号；整数；输入变量
		Yes_Contact_Con	接触迭代是否收敛；逻辑型；输出变量
Get_Contact_State_Eles_IN	获得单元接触状态	*Get_Contact_State_Eles_IN*(*i_Contact*, *c_Cracks_CalP_Aper*, *tem_Elem_Conta_Sta*, *Yes_Contact*, *Yes_Contact_Con*,*num_Contact_Ele*)	
		i_Contact	接触迭代步号；整数；输入变量
		c_Cracks_CalP_Aper (*Max_Num_Cr*,*Max_Num_Cr_CalP*)	各裂缝计算点（裂缝与单元网格的交点）处的开度；浮点数；输入变量
		tem_Elem_Conta_Sta(*Num_Elem*,*Max_Num_Cr*)	各单元相当于各裂缝的接触状态；整数；输入输出变量
		Yes_Contact	是否有裂缝发生接触；逻辑型；输出变量
		Yes_Contact_Con	接触迭代是否收敛；逻辑型；输出变量
		num_Contact_Ele	接触单元的数目；整数；输出变量
Get_Gauss_Disps	计算高斯点的位移，计算结果存储到全局变量 *DISP_x_Gauss* 和 *DISP_y_Gauss* 中	*Get_Gauss_Disps*(*isub*, *c_DISP*, *Total_Num_G_P*)	
		isub	载荷子步号；整数；输入变量
		c_DISP(*Total_FD*)	节点位移；浮点数；输入变量
		Total_Num_G_P	总的高斯点数目；整数；输入变量
Get_Gauss_Stress_FEM	计算高斯点的应力（用于 FEM 分析）	*Get_Gauss_Stress_FEM*(*isub*)	
		isub	载荷子步号；整数；输入变量
Get_Gauss_Stress_XFEM	计算高斯点的应力（用于 XFEM 分析）	*Get_Gauss_Stress_XFEM*(*isub*, *c_DISP*)	
		isub	载荷子步号；整数；输入变量
		c_DISP(*Total_FD*)	节点位移；浮点数；输入变量
Get_Material_Matrix	处理材料参数，形成应力应变关系矩阵等	无输入输出变量	
Get_Material_Matrix_3D	处理材料参数，形成应力应变关系矩阵等(用于 3D	无输入输出变量	

	问题）		
Get_Node_Stress_FEM	计算节点应力（用于 FEM 分析）	*Get_Node_Stress_FEM*(*isub*)	
		isub	载荷子步号；整数；输入变量
Get_Node_Stress_XFEM	计算节点应力（用于 XFEM 分析）	*Get_Node_Stress_XFEM*(*isub*, *c_DISP*)	
		isub	载荷子步号；整数；输入变量
		c_DISP(*Total_FD*)	节点位移；浮点数；输入变量
HF_Logic_Checking	水力压裂分析的逻辑检测及数据修正	无输入输出变量	
Initialize	初始化程序及相关变量（给参数赋初始值）	无输入输出变量	
Input_Check_Display	对输入信息进行检测，并在程序运行窗口中进行显示	无输入输出变量	
Location_Element_Stiff_Matrix	提取裂缝增强单元刚度矩阵在总刚度矩阵中的位置（自由度编号）	*Location_Element_Stiff_Matrix*(*i_E*, *i_C*, *POS_c_Crack*, *Location_ESM_C_Crack*, *num_Loc_ESM_C_Crack*, *Location_ESM_C_Cr_NoFEM*, *num_Loc_ESM_C_Cr_NoFEM)*	
		i_E	单元号；整数；输入变量
		i_C	裂缝号；整数；输入变量
		POS_c_Crack(*Num_Node*)	各节点相对于裂缝 *i_C* 的增强自由度编号，其中，*Num_Node* 为总的节点数目；整数；输入变量
		Location_ESM_C_Crack(*80*)	当前单元刚度矩阵在总刚中的位置（自由度编号），包含 FEM 自由度；整数；输出变量
		num_Loc_ESM_C_Crack(*60*)	当前单元刚度矩阵在总刚中的位置（自由度编号），不包含 FEM 自由度；整数；输入变量
		Location_ESM_C_Cr_NoFEM	当前单元刚度矩阵自由度数目，包含 FEM 自由度；整数；输出变量
		num_Loc_ESM_C_Cr_NoFEM	当前单元刚度矩阵自由度数目，不包含 FEM 自由度；整数；输出变量
Location_Element_Stiff_Matrix_3D	对于 3D 裂缝问题，提取裂缝增强单元刚度矩阵在总刚度矩阵中的位置（自	*Location_Element_Stiff_Matrix_3D(i_E*, *i_C*, *POS_c_Crack*, *Location_ESM_C_Crack*, *num_Loc_ESM_C_Crack*, *Location_ESM_C_Cr_NoFEM*, *num_Loc_ESM_C_Cr_NoFEM*)	
		i_E	单元号；整数；输入变量
		i_C	裂缝号；整数；输入变量
		POS_c_Crack(*Num*	各节点相对于裂缝 *i_C* 的增强自由度编

	由度编号）	*_Node*)	号；整数；输入变量
		Location_ESM_C_Crack(*100*)	当前单元刚度矩阵在总刚中的位置（自由度编号），包含 FEM 自由度；整数；输出变量
		num_Loc_ESM_C_Crack(*100*)	当前单元刚度矩阵在总刚中的位置（自由度编号），不包含 FEM 自由度；整数；输入变量
		Location_ESM_C_Cr_NoFEM	当前单元刚度矩阵自由度数目，包含 FEM 自由度；整数；输出变量
		num_Loc_ESM_C_Cr_NoFEM	当前单元刚度矩阵自由度数目，不包含 FEM 自由度；整数；输出变量
Location_Element_Stiff_Matrix_Hl	提取孔洞增强单元刚度矩阵在总刚度矩阵中的位置（自由度编号）	*Location_Element_Stiff_Matrix_Hl*(*i_E*, *i_H*, *POS_c_Hl*, *Location_ESM_C_Crack*, *num_Loc_ESM_C_Crack*, *Location_ESM_C_Cr_NoFEM*, *num_Loc_ESM_C_Cr_NoFEM*)	
		i_E	单元号；整数；输入变量
		i_H	孔洞号；整数；输入变量
		POS_c_Hl(*Num_Node*)	各节点相对于孔洞 *i_H* 的增强自由度编号；整数；输入变量
		Location_ESM_C_Crack(*80*)	当前单元刚度矩阵在总刚中的位置（自由度编号），包含 FEM 自由度；整数；输出变量
		num_Loc_ESM_C_Crack(*60*)	当前单元刚度矩阵在总刚中的位置（自由度编号），包含 FEM 自由度；整数；输出变量
		Location_ESM_C_Cr_NoFEM	当前单元刚度矩阵自由度数目，包含 FEM 自由度；整数；输出变量
		num_Loc_ESM_C_Cr_NoFEM	当前单元刚度矩阵自由度数目，不包含 FEM 自由度；整数；输出变量
Location_Element_Stiff_Matrix_Incl	提取夹杂增强单元刚度矩阵在总刚度矩阵中的位置（自由度编号）	*Location_Element_Stiff_Matrix_Incl*(*i_E*, *i_Incl*, *POS_c_Incl*, *Location_ESM_C_Crack*, *num_Loc_ESM_C_Crack*, *Location_ESM_C_Cr_NoFEM*, *num_Loc_ESM_C_Cr_NoFEM*)	
		i_E	单元号；整数；输入变量
		i_Incl	夹杂号；整数；输入变量
		POS_c_Incl(*Num_Node*)	节点相对于夹杂 *i_Incl* 的自由度编号；整数；输入变量
		Location_ESM_C_Crack(*80*)	当前单元刚度矩阵在总刚中的位置（自由度编号），包含 FEM 自由度；整数；输出变量
		num_Loc_ESM_C_Crack(*60*)	当前单元刚度矩阵在总刚中的位置（自由度编号），包含 FEM 自由度；整数；输出变量
		Location_ESM_C_Cr_NoFEM	当前单元刚度矩阵自由度数目，包含 FEM 自由度；整数；输出变量
		*num_Loc_ESM_C_*	当前单元刚度矩阵自由度数目，不包含

		Cr_NoFEM	FEM 自由度；整数；输出变量
Natural_Fractures_Related_Preparation	天然裂缝相关准备	无输入输出变量	
Number_Enriched_Nodes	给增强节点编号，得到全局变量 *c_POS*、*c_POS_Hl*、*c_POS_Incl* 等	*Number_Enriched_Nodes*(*isub*)	
		isub	载荷子步号；整数；输入变量
Number_Enriched_Nodes_3D	给增强节点编号，用于 3D 问题，得到全局变量 *c_POS*	*Number_Enriched_Nodes_3D*(*isub*)	
		isub	载荷子步号；整数；输入变量
Read_Geo	读取模型几何信息，得到模型关键变量 *Coor* 、*Num_Node* 、*Num_Elem* 等	无输入输出变量	
Read_Geo_3D	读取模型几何信息（三维问题），得到模型关键变量 *Coor* 、*Num_Node* 、*Num_Elem* 等	无输入输出变量	
Stat_Crack_Connection	用于水力压裂分析统计各个裂缝之间的联通关系，以便确定压裂液的流动网络	*Stat_Crack_Connection*(*iter*)	
		iter	载荷子步号；整数；输入变量
Warning_Message	输出错误或警告信息	*Warning_Message*(*Mess_Type*, *Keywords*)	
		Mess_Type	
		Keywords	
Welcome	输出欢迎信息、软件版本号等	无输入输出变量	

3 以 *Cal* 开头的子程序

PhiPsi 源代码中 Cal 开头的子程序执行具体的计算任务，如计算裂缝增强单元的应变位移关系矩阵 **B** 的 *Cal_B_Matrix_Crack*、计算圆形夹杂增强单元 **B** 矩阵的 *Cal_B_Matrix_Circ_Incl*、计算接触问题法向和切向接触力的 *Cal_Contact_PN_and_PT*、计算裂缝开度的 *Cal_Crack_Aperture*、计算两条线段之间夹角的 *Cal_Angle_of_AB_and_BC* 以及计算点到线段符号距离的 *Cal_Signed_Distance* 等。表 3 列出了 PhiPsi 程序以 *Cal* 开头的主要子程序及其功能和输入输出接口说明。

表 3 PhiPsi 程序以 *Cal* 开头的工具子程序用途及输入输出变量说明（按名称排序）

子程序名	子程序功能	子程序输入输出接口	
Cal_and_Check_Junction_Point	对新生成的 Junction 点进行检查和修正，保证交叉点只需要一个 Junction 增强单元，若不满足，则进行修正	*Cal_and_Check_Junction_Point*(*ifra*, *iter*, *Point_B*, *Point_C*, *Point_D*, *Inter_x*, *Inter_y*)	
		ifra	载荷步号；整数；输入变量
		iter	载荷子步号；整数；输入变量
		Point_B(*2*)	B 点坐标；浮点数；输入变量
		Point_C(*2*)	C 点坐标；浮点数；输入变量
		Point_D(*2*)	D 点坐标；浮点数；输入变量
		Inter_x	Junction 交点 *x* 坐标；浮点数；输入输出变量
		Inter_y	Junction 交点 *y* 坐标；浮点数；输入输出变量
Cal_Angle_of_AB_and_BC	计算线段 AB 和 BC 的夹角	*Cal_Angle_of_AB_and_BC*(*Line_AB*, *Line_BC*, *angle_AB_BC*)	
		Line_AB(*2*,*2*)	线段 AB；浮点数；输入变量
		Line_BC(*2*,*2*)	线段 BC；浮点数；输入变量
		angle_AB_BC	线段 AB 和 BC 的夹角；浮点数；输出变量
Cal_Any_Point_Disp_KesiYita	计算任意点的位移	*Cal_Any_Point_Disp_KesiYita*(*elem*,*Kesi*,*Yita*,*i_G*,*c_DISP*,*P_Disp*)	
		elem	点所在单元号；整数；输入变量
		Kesi	点相对于所在单元的自然坐标 ζ；浮点数；输入变量
		Yita	点相对于所在单元的自然坐标 η；浮点数；输入变量
		i_G	高斯点号（注：该变量不起作用）；整数；输入变量
		c_DISP(*Total_FD*)	全部自由度的位移向量；浮点数；输入变量
		P_Disp(*2*)	计算得到的点的位移；浮点数；输出变量
Cal_Any_Point_D	计算模型中任意点	*Cal_Any_Point_Disp_KesiYita_3D*(*elem*, *Kesi*, *Yita*, *Zeta*, *i_G*,	

isp_KesiYita_3D	的位移，用于 3D 问题	*c_DISP*, *P_Disp*)	
		elem	点所在单元号；整数；输入变量
		Kesi	点相对于所在单元的自然坐标 ξ；浮点数；输入变量
		Yita	点相对于所在单元的自然坐标 η；浮点数；输入变量
		Zeta	点相对于所在单元的自然坐标 ζ；浮点数；输入变量
		i_G	高斯点号（注：该变量不起作用）；整数；输入变量
		c_DISP(*Total_FD*)	全部自由度的位移向量；浮点数；输入变量
		P_Disp(*2*)	计算得到的点的位移；浮点数；输出变量
Cal_Any_Point_Disp_KesiYita_NoFEM	计算模型中任意点的位移，与 *Cal_Any_Point_Disp_KesiYita* 的区别在于输入的位移向量仅仅包含增强自由度，不含常规自由度	*Cal_Any_Point_Disp_KesiYita_NoFEM*(*elem*, *Kesi*, *Yita*, *i_G*, *U_e*, *P_Disp*)	
		elem	点所在单元号；整数；输入变量
		Kesi	点相对于所在单元的自然坐标 ζ；浮点数；输入变量
		Yita	点相对于所在单元的自然坐标 η；浮点数；输入变量
		i_G	高斯点号（注：该变量不起作用）；整数；输入变量
		U_e(*Enrich_Freedom*)	增强位移自由度向量；浮点数；输入变量
		P_Disp(*2*)	计算得到的点的位移；浮点数；输出变量
Cal_Ap_and_Am	计算节点支撑域被裂缝片段剖分后，各子支撑域是否含有高斯积分点，相关理论见3.4.4 小节	*Cal_Ap_and_Am*(*c_Crack*, *c_Cr_Point*, *DOMAIN_Outline*, *m_DOMAIN_Outline*, *Domain_El*, *n_Domain_El*, *c_E*, *c_Node*, *Yes_Cutthrough*, *Yes_Ap_has_GP*, *Yes_Am_has_GP*)	
		c_Crack(*Max_Num_Cr_P*,*2*)	当前裂缝的裂缝点坐标；浮点数；输入变量
		c_Cr_Point	当前裂缝坐标点数目；整数；输入变量
		DOMAIN_Outline(*m_DOMAIN_Outline*,*2*)	当前节点的支撑域（由节点编号组成，这些节点相互连接组成支撑域），*DOMAIN_Outline*(*i*,*1*)和 *DOMAIN_Outline*(*i*,*2*)分别表示支撑域第 *i* 条边线的两个节点编号；整数；输入变量
		m_DOMAIN_Outline	支撑域的边线（每个边线由两个节点组成）数目；整数；输入变量
		Domain_El(*n_Domain_El*)	支撑域内的单元号列表；整数；输入变量
		n_Domain_El	支撑域内的单元数目（注：临近模型边界的节点支撑域内单元数少）；整数；输入变量
		c_E	当前单元号（注：该变量不起作用）；整数；

			输入变量
		c_Node	当前节点号（注：该变量不起作用）；整数；输入变量
		Yes_Cutthrough	支撑域是否被裂缝完全剖分；逻辑变量；输出变量
		Yes_Ap_has_GP	*Ap* 子支撑域含有高斯点；逻辑变量；输出变量
		Yes_Am_has_GP	*Am* 子支撑域含有高斯点；逻辑变量；输出变量
Cal_B_Matrix_Crack	计算裂缝问题的 **B** 矩阵（实际上是计算高斯点(ξ, η)对 **B** 矩阵的贡献），被 *Assemble_Stiffness_Matrix_XFEM* 及 *Get_Gauss_Stress_XFEM* 等子程序调用	*Cal_B_Matrix_Crack*(*kesi*, *yita*, *i_C*, *i_E*, *i_G*, *c_NN*, *c_X_NODES*, *c_Y_NODES*, *tem_B*, *num_tem_B*)	
		kesi	高斯点的自然坐标 ξ；浮点数；输入变量
		yita	高斯点的自然坐标 η；浮点数；输入变量
		i_C	裂缝号；整数；输入变量
		i_E	单元号；整数；输入变量
		i_G	Gauss 点号；整数；输入变量
		c_NN(*4*)	当前单元的 4 个节点号；整数；输入变量
		c_X_NODES(*4*)	当前单元 4 个节点的 *x* 坐标；浮点数；输入变量
		c_Y_NODES(*4*)	当前单元 4 个节点的 *y* 坐标；浮点数；输入变量
		tem_B(*3*,*80*)	**B** 矩阵（当前高斯点的贡献）；浮点数；输出变量
		num_tem_B	**B** 矩阵（当前高斯点的贡献）的实际列数，*tem_B*(*3*, *num_tem_B*)；整数；输出变量
Cal_B_Matrix_Circ_Incl	计算圆形夹杂问题的 **B** 矩阵（实际上是计算高斯点(ξ, η)对 **B** 矩阵的贡献），被 *Assemble_Stiffness_Matrix_XFEM* 及 *Get_Gauss_Stress_XFEM* 等子程序调用	*Cal_B_Matrix_Circ_Incl*(*kesi*, *yita*, *i_Incl*, *i_E*, *i_G*, *c_NN*, *c_X_NODES*, *c_Y_NODES*, *tem_B*, *num_tem_B*)	
		kesi	高斯点的自然坐标 ξ；浮点数；输入变量
		yita	高斯点的自然坐标 η；浮点数；输入变量
		i_Incl	夹杂号；整数；输入变量
		i_E	单元号；整数；输入变量
		i_G	Gauss 点号；整数；输入变量
		c_NN(*4*)	当前单元的 4 个节点号；整数；输入变量
		c_X_NODES(*4*)	当前单元 4 个节点的 *x* 坐标；浮点数；输入变量
		c_Y_NODES(*4*)	当前单元 4 个节点的 *y* 坐标；浮点数；输入变量
		tem_B(*3*,*80*)	**B** 矩阵（当前高斯点的贡献）；浮点数；输出变量
		num_tem_B	**B** 矩阵（当前高斯点的贡献）的实际列数，*tem_B*(*3*, *num_tem_B*)；整数；输出变量
Cal_B_Matrix_Crack_3D	计算 3D 裂缝问题的 **B** 矩阵（实际上	*Cal_B_Matrix_Crack_3D*(*kesi*, *yita*, *zeta*, *i_C*, *i_E*, *i_G*, *c_NN*, *c_X_NODES*, *c_Y_NODES*, *c_Z_NODES*, *tem_B*, *num_tem_B*)	

	是计算高斯点(ξ, η)对 **B** 矩阵的贡献），主要被子程序 *Assemble_Stiffness_Matrix_XFEM* 调用	*kesi*	高斯点的自然坐标 ξ；浮点数；输入变量
		yita	高斯点的自然坐标 η；浮点数；输入变量
		zeta	高斯点的自然坐标 ζ；浮点数；输入变量
		i_C	裂缝号；整数；输入变量
		i_E	单元号；整数；输入变量
		i_G	Gauss 点号；整数；输入变量
		c_NN(*8*)	当前单元的 8 个节点号；整数；输入变量
		c_X_NODES(*8*)	当前单元 8 个节点的 *x* 坐标；浮点数；输入变量
		c_Y_NODES(*8*)	当前单元 8 个节点的 *y* 坐标；浮点数；输入变量
		c_Z_NODES(*8*)	当前单元 8 个节点的 *z* 坐标；浮点数；输入变量
		tem_B(*6*,*100*)	**B** 矩阵（当前高斯点的贡献）；浮点数；输出变量
		num_tem_B	**B** 矩阵（当前高斯点的贡献）的实际列数，*tem_B*(*3*, *num_tem_B*)；整数；输出变量
Cal_B_Matrix_Hl	计算孔洞问题的 **B** 矩阵（实际上是计算高斯点(ξ, η)对 **B** 矩阵的贡献），被 *Assemble_Stiffness_Matrix_XFEM* 及 *Get_Gauss_Stress_XFEM* 等子程序调用	*Cal_B_Matrix_Hl(kesi*, *yita*, *i_H*, *i_E*, *i_G*, *c_NN*, *c_X_NODES*, *c_Y_NODES*, *tem_B*, *num_tem_B*)	
		kesi	高斯点的自然坐标 ξ；浮点数；输入变量
		yita	高斯点的自然坐标 η；浮点数；输入变量
		i_H	孔洞号；整数；输入变量
		i_E	单元号；整数；输入变量
		i_G	Gauss 点号；整数；输入变量
		c_NN(*4*)	当前单元的 4 个节点号；整数；输入变量
		c_X_NODES(*4*)	当前单元 4 个节点的 *x* 坐标；浮点数；输入变量
		c_Y_NODES(*4*)	当前单元 4 个节点的 *y* 坐标；浮点数；输入变量
		tem_B(*3*,*80*)	**B** 矩阵（当前高斯点的贡献）；浮点数；输出变量
		num_tem_B	**B** 矩阵（当前高斯点的贡献）的实际列数，*tem_B*(*3*, *num_tem_B*)；整数；输出变量
Cal_Contact_Contact_State_Gauss	计算接触 Newton-Raphson（牛顿-拉普森，可简写为 N-R）迭代各高斯点的接触状态	*Cal_Contact_Contact_State_Gauss*(*iter*, *ifra*, *Counter_Iter*, *i_NR_P*, *c_DISP*, *Yes_Contact*, *c_Elem_Conta_Sta*, *CT_State_Gauss*)	
		iter	迭代步号；整数；输入变量
		ifra	载荷子步号；整数；输入变量
		Counter_Iter	迭代计数器；整数；输入变量
		i_NR_P	Newton-Raphson 接触迭代次数；整数；输入变量
		c_DISP(*Total_FD*)	全部自由度的位移向量；浮点数；输入变量

		Yes_Contact	是否发生了接触；逻辑变量；输出变量
		c_Elem_Conta_Sta(*Num_Elem*, *num_Crack*)	各单元相对于各个裂缝的接触状态（=1 表示发生了接触）；整数；输出变量
		CT_State_Gauss(*num_Crack,Max_Num_Cr_CalP-1,2*)	每条裂缝各接触单元两个 Gauss 点的接触状态（= 0，分离；=1，粘结；=2，滑移）；整数；输出变量
Cal_Contact_Conve_Factor	检查接触 Newton-Raphson 迭代是否收敛并计算接触分析的收敛容差	*Cal_Contact_Conve_Factor*(*iter*, *ifra*, *Counter_Iter*, *i_NR_P*, *Conve_Tolerance*, *c_Total_Freedom*, *c_freeDOF*, *c_num_freeDOF*, *c_F*, *R_PSI*, *Last_R_PSI*, *delta_U*, *U*, *Contact_DISP_0*, *Yes_Conve*, *Conve_Factor*)	
		iter	迭代步号；整数；输入变量
		ifra	载荷子步号；整数；输入变量
		Counter_Iter	迭代计数器；整数；输入变量
		i_NR_P	Newton-Raphson 接触迭代次数；整数；输入变量
		Conve_Tolerance	接触收敛容差；浮点数；输入变量
		c_Total_Freedom	总的自由度数目；整数；输入变量
		c_freeDOF(*c_Total_Freedom*)	活动自由度向量；整数；输入变量
		c_num_freeDOF	活动自由度数目；整数；输入变量
		c_F(*c_Total_Freedom*)	外载荷向量；浮点数；输入变量
		R_PSI(*c_Total_Freedom*)	接触迭代的残差；浮点数；输入变量
		Last_R_PSI(*c_Total_Freedom*)	上一接触迭代步的残差；浮点数；输入变量
		delta_U(*c_Total_Freedom*)	位移增量向量；浮点数；输入变量
		U(*c_Total_Freedom*)	当前位移向量；浮点数；输入变量
		Contact_DISP_0(*c_Total_Freedom*)	接触迭代开始时的位移向量；浮点数；输入变量
		Yes_Conve	接触迭代是否已收敛；逻辑变量；输出变量
		Conve_Factor	接触迭代收敛容差；浮点数；输出变量
Cal_Contact_Jacobian	计算接触 Newton-Raphson 迭代的雅可比矩阵	*Cal_Contact_Jacobian*(*iter*, *ifra*, *Counter_Iter*, *i_NR_P*, *c_Total_Freedom*, *c_num_freeDOF*, *ori_globalK*, *freeDOF*, *Kn*, *Kn_Gauss*, *Kt_Gauss*, *CT_State_Gauss*, *CT_Jacobian*)	
		iter	迭代步号；整数；输入变量

		ifra	载荷子步号；整数；输入变量
		Counter_Iter	迭代计数器；整数；输入变量
		i_NR_P	Newton-Raphson 接触迭代次数；整数；输入变量
		c_Total_Freedom	总的自由度数目；整数；输入变量
		c_num_freeDOF	活动自由度数目；整数；输入变量
		ori_globalK(*c_Total_Freedom*, *c_Total_Freedom*)	第一个接触迭代步传入的刚度矩阵；浮点数；输入变量
		freeDOF(*c_Total_Freedom*)	活动自由度向量；整数；输入变量
		Kn	法向罚刚度；浮点数；输入变量
		Kn_Gauss(*num_Crack*,*Max_Num_Cr_CalP-1*,*2*)	接触单元 Gauss 点的法向罚刚度；浮点数；输入输出变量
		Kt_Gauss(*num_Crack*,*Max_Num_Cr_CalP-1*,*2*)	接触单元 Gauss 点的切向罚刚度；浮点数；输入输出变量
		CT_State_Gauss(*num_Crack,Max_Num_Cr_CalP-1*,*2*)	每条裂缝各接触单元两个 Gauss 点的接触状态（= 0，分离；=1，粘结；=2，滑移）；整数；输入变量
		CT_Jacobian(*c_Total_Freedom*, *c_Total_Freedom*)	接触迭代的雅可比矩阵；浮点数；输出变量
Cal_Contact_PN_and_PT	计算接触 Newton-Raphson 迭代分析的接触面法向和切向接触力	*Cal_Contact_PN_and_PT*(*iter*, *ifra*, *Counter_Iter*, *i_NR_P*, *c_Total_Freedom*, *c_num_freeDOF*, *Kn*, *Kn_Gauss*, *Kt_Gauss*, *fric_mu*, *c_DISP*, *delta_u_a*, *CT_State_Gauss*, *c_Elem_Conta_Sta*, *PC_Gauss_x*, *PC_Gauss_y*)	
		iter	迭代步号；整数；输入变量
		ifra	载荷子步号；整数；输入变量
		Counter_Iter	迭代计数器；整数；输入变量
		i_NR_P	Newton-Raphson 接触迭代次数；整数；输入变量
		c_Total_Freedom	总的自由度数目；整数；输入变量
		c_num_freeDOF	活动自由度数目；整数；输入变量

		Kn	法向罚刚度；浮点数；输入变量
		Kn_Gauss(*num_Crack*,*Max_Num_Cr_CalP-1*,*2*)	接触单元 Gauss 点的法向罚刚度；浮点数；输入输出变量
		Kt_Gauss(*num_Crack*,*Max_Num_Cr_CalP-1*,*2*)	接触单元 Gauss 点的切向罚刚度；浮点数；输入输出变量
		fric_mu	裂缝面摩擦系数；浮点数；输入变量
		c_DISP(*c_Total_Freedom*)	总的位移向量；浮点数；输入变量
		delta_u_a(*c_Total_Freedom*)	位移增强向量；浮点数；输入变量
		CT_State_Gauss(*num_Crack,Max_Num_Cr_CalP-1*,*2*)	每条裂缝各接触单元两个 Gauss 点的接触状态（= 0，分离；=1，粘结；=2，滑移）；浮点数；输入输出变量
		c_Elem_Conta_Sta(*Num_Elem*, *num_Crack*)	各单元相对于各个裂缝的接触状态（=1 表示发生了接触）；整数；输入输出变量
		PC_Gauss_x(*num_Crack*,*Max_Num_Cr_CalP-1*,*2*)	转换到整体直角坐标系下之后的高斯积分点接触力之 *x* 方向分力；浮点数；输入输出变量
		PC_Gauss_y(*num_Crack*,*Max_Num_Cr_CalP-1*,*2*)	转换到整体直角坐标系下之后的高斯积分点接触力之 *y* 方向分力；浮点数；输入输出变量
Cal_Coor_by_KesiYita	根据点所在单元的自然坐标计算整体坐标	*Cal_Coor_by_KesiYita*(*kesi*, *yita*, *X_NODES*, *Y_NODES*, *Out_x*, *Out_y*)	
		kesi	点相对于所在单元的自然坐标 ξ；浮点数；输入变量
		yita	点相对于所在单元的自然坐标 η；浮点数；输入变量
		X_NODES(*4*)	点所在单元 4 个节点的 *x* 坐标；浮点数；输入变量
		Y_NODES(*4*)	点所在单元 4 个节点的 *y* 坐标；浮点数；输入变量
		Out_x	点在整体坐标系下的 *x* 坐标；浮点数；输出变量
		Out_y	点在整体坐标系下的 *y* 坐标；浮点数；输出变量
Cal_Coor_by_Ke	根据点所在单元的	*Cal_Coor_by_KesiYita_3D*(*kesi*, *yita*, *zeta*, *X_NODES*, *Y_NODES*,	

siYita_3D	自然坐标计算整体坐标，用于 3D 问题	*Z_NODES, Out_x*, *Out_y*, *Out_z*)	
		kesi	点相对于所在单元的自然坐标 ξ；浮点数；输入变量
		yita	点相对于所在单元的自然坐标 η；浮点数；输入变量
		zeta	点相对于所在单元的自然坐标 ζ；浮点数；输入变量
		X_NODES(*8*)	点所在单元 8 个节点的 *x* 坐标；浮点数；输入变量
		Y_NODES(*8*)	点所在单元 8 个节点的 *y* 坐标；浮点数；输入变量
		Z_NODES(*8*)	点所在单元 8 个节点的 *z* 坐标；浮点数；输入变量
		Out_x	点在整体坐标系下的 *x* 坐标；浮点数；输出变量
		Out_y	点在整体坐标系下的 *y* 坐标；浮点数；输出变量
		Out_z	点在整体坐标系下的 *z* 坐标；浮点数；输出变量
Cal_Crack_Aperture	计算裂缝开度（法向相对位移）	*Cal_Crack_Aperture*(*isub*, *c_DISP*)	
		isub	载荷子步号；整数；输入变量
		c_DISP(*Total_FD*)	位移自由度向量；浮点数；输入变量
Cal_Crack_Points_Info	获得裂缝计算点（裂缝与单元边的交点，对于水力压裂分析即为流体节点）的坐标、方向、所在裂缝片段号、所在单元号等信息	*Cal_Crack_Points_Info*(*isub*)	
		isub：计算子步号；整数；输入变量	
Cal_Crack_Tan_Relative_Disp	计算裂缝切向相对滑动位移	*Cal_Crack_Tan_Relative_Disp*(*isub*, *c_DISP*, *Cracks_CalP_Tan_Aper*)	
		isub	载荷子步号；整数；输入变量
		c_DISP(*Total_FD*)	位移自由度向量；浮点数；输入变量
		Cracks_CalP_Tan_Aper(*num_Crack*,*Max_Num_Cr_CalP*)	各条裂缝计算点的切向相对滑动位移；浮点数；输出变量
Cal_detJ	计算单元内某点的雅可比矩阵行列式	*Cal_detJ*(*kesi*, *yita*, *X_NODES*, *Y_NODES*, *detJ*)	
		kesi	点相对于所在单元的自然坐标 ξ；浮点数；输入变量
		yita	点相对于所在单元的自然坐标 η；浮点数；输入变量

		X_NODES(*4*)	点所在单元 4 个节点的 *x* 坐标；浮点数；输入变量
		Y_NODES(*4*)	点所在单元 4 个节点的 *y* 坐标；浮点数；输入变量
		detJ	雅可比矩阵行列式；浮点数；输出变量
Cal_detJ_3D	计算三维单元内某点的雅可比矩阵行列式	*Cal_detJ_3D*(*kesi*, *yita*, *zeta*, *X_NODES*, *Y_NODES*, *Z_NODES*, *detJ*)	
		kesi	点相对于所在单元的自然坐标 ξ；浮点数；输入变量
		yita	点相对于所在单元的自然坐标 η；浮点数；输入变量
		zeta	点相对于所在单元的自然坐标 ζ；浮点数；输入变量
		X_NODES(*8*)	点所在单元 8 个节点的 *x* 坐标；浮点数；输入变量
		Y_NODES(*8*)	点所在单元 8 个节点的 *y* 坐标；浮点数；输入变量
		Z_NODES(*8*)	点所在单元 8 个节点的 *z* 坐标；浮点数；输入变量
		detJ	雅可比矩阵行列式；浮点数；输出变量
Cal_Ele_B_N4	计算四节点四边形单元的应变-位移矩阵 **B**（仅 FEM）	*Cal_Ele_B_N4*(*i_E*, *X_NODES*, *Y_NODES*, *kesi*, *yita*, *ToTal_B*)	
		i_E	所在单元号；整数；输入变量
		X_NODES(*4*)	点所在单元 4 个节点的 *x* 坐标；浮点数；输入变量
		Y_NODES(*4*)	点所在单元 4 个节点的 *y* 坐标；浮点数；输入变量
		kesi	点相对于所在单元的自然坐标 ξ；浮点数；输入变量
		yita	点相对于所在单元的自然坐标 η；浮点数；输入变量
		ToTal_B(*3*,*8*)	**B** 矩阵；浮点数；输出变量
Cal_Ele_Mass_Matrix_N4	计算四节点四边形单元的质量矩阵（仅 FEM）	*Cal_Ele_Mass_Matrix_N4*(*X_NODES*, *Y_NODES*, *thick*, *c_density*, *kesi*, *yita*, *weight*, *localM*)	
		X_NODES(*4*)	点所在单元 4 个节点的 *x* 坐标；浮点数；输入变量
		Y_NODES(*4*)	点所在单元 4 个节点的 *y* 坐标；浮点数；输入变量
		thick	单元的厚度；浮点数；输入变量
		c_density	单元的材料密度；浮点数；输入变量
		kesi(*Num_Gauss_P_FEM*)	各个高斯积分点的自然坐标 ξ；浮点数；输入变量
		yita(*Num_Gauss_P_FEM*)	各个高斯积分点的自然坐标 η；浮点数；输入变量

		weight(*Num_Gauss_P_FEM*)	各个高斯积分点的积分权重；浮点数；输入变量
		localM(8,*8*)	单元质量矩阵；浮点数；输出变量
Cal_Ele_Num_by_Coors	根据点的坐标计算点所在单元的单元号	*Cal_Ele_Num_by_Coors*(*x*, *y, OUT_Elem*)	
		x	点的整体 *x* 坐标；浮点数；输入变量
		y	点的整体 *y* 坐标；浮点数；输入变量
		OUT_Elem	单元号；整数；输出变量
Cal_Ele_Num_by_Coors_3D	根据点的坐标计算点所在单元的单元号（用于 3D 问题）	*Cal_Ele_Num_by_Coors_3D*(*x*, *y*, *z*, *OUT_Elem*)	
		x	点的整体 *x* 坐标；浮点数；输入变量
		y	点的整体 *y* 坐标；浮点数；输入变量
		z	点的整体 *z* 坐标；浮点数；输入变量
		OUT_Elem	单元号；整数；输出变量
Cal_Ele_Num_by_Coors_YesInOn	根据点的坐标计算点所在单元的单元号，并返回点的相对位置（即点在单元边线上还是单元内）	*Cal_Ele_Num_by_Coors_YesInOn*(*x*, *y*, *OUT_Elem*, *Yes_In*, *Yes_On*)	
		x	点的整体 *x* 坐标；浮点数；输入变量
		y	点的整体 *y* 坐标；浮点数；输入变量
		OUT_Elem	单元号；整数；输出变量
		Yes_In	若点在单元内部，则为 True；逻辑变量；输出变量
		Yes_On	若点在单元边线上，则为 True；逻辑变量；输出变量
Cal_Ele_Stiffness_Matrix_3D_8nodes	计算三维八节点六面体单元的刚度矩阵（仅 FEM）	*Cal_Ele_Stiffness_Matrix_3D_8nodes*(*i_E*, *num_Gauss*, *X_NODES*,*Y_NODES*,*Z_NODES*, *c_D*, *kesi*, *yita*, *zeta*, *weight*, *localK*)	
		i_E	单元号；整数；输入变量
		num_Gauss	高斯积分点数目；整数；输入变量
		X_NODES(*8*)	点所在单元 8 个节点的 *x* 坐标；浮点数；输入变量
		Y_NODES(*8*)	点所在单元 8 个节点的 *y* 坐标；浮点数；输入变量
		Z_NODES(*8*)	点所在单元 8 个节点的 *z* 坐标；浮点数；输入变量
		c_D(*6*,*6*)	应力应变关系矩阵；浮点数；输入变量
		kesi(*num_Gauss*)	各个高斯积分点的自然坐标 ξ；浮点数；输入变量
		yita(*num_Gauss*)	各个高斯积分点的自然坐标 η；浮点数；输入变量
		zeta(*num_Gauss*)	各个高斯积分点的自然坐标 ζ；浮点数；输入变量
		weight(*num_Gauss*)	各个高斯积分点积分权重；浮点数；输入变量
		localK(24,*24*)	单元刚度矩阵；浮点数；输出变量
Cal_Ele_Stiffness	计算四节点四边形	*Cal_Ele_Stiffness_Matrix_N4(X_NODES*, *Y_NODES*, *thick*, *c_D*,	

_Matrix_N4	单元的刚度矩阵（仅 FEM）	*kesi*, *yita*, *weight*, *localK*)	
		X_NODES(*4*)	点所在单元 4 个节点的 *x* 坐标；浮点数；输入变量
		Y_NODES(*4*)	点所在单元 4 个节点的 *y* 坐标；浮点数；输入变量
		thick	单元的厚度；浮点数；输入变量
		c_D(*3*,*3*)	应力应变关系矩阵；浮点数；输入变量
		kesi(*Num_Gauss_P_FEM*)	各个高斯积分点的自然坐标 ξ；浮点数；输入变量
		yita(*Num_Gauss_P_FEM*)	各个高斯积分点的自然坐标 η；浮点数；输入变量
		weight(*Num_Gauss_P_FEM*)	各个高斯积分点积分权重；浮点数；输入变量
		localK(*8,8*)	单元刚度矩阵；浮点数；输出变量
Cal_Ele_Stress_and_B_N4	计算四节点四边形单元内某点（一般是高斯积分点）的应力、应力以及应变-位移矩阵 **B**	*Cal_Ele_Stress_and_B_N4*(*i_E*, *i_N*, *X_NODES*, *Y_NODES*, *c_D*, *kesi*, *yita*, *U*, *c_Strain*, *c_Stress*, *ToTal_B*, *detJ*)	
		i_E	单元号；整型；输入变量
		i_N	节点号（注：该变量不起作用）；整型；输入变量
		X_NODES(*4*)	单元 4 个节点的 *x* 坐标；浮点数；输入变量
		Y_NODES(*4*)	单元 4 个节点的 *y* 坐标；浮点数；输入变量
		c_D(*3*,*3*)	应力应变关系矩阵；浮点数；输入变量
		kesi	点相对于所在单元的自然坐标 ξ；浮点数；输入变量
		yita	点相对于所在单元的自然坐标 η；浮点数；输入变量
		U(*8*)	当前单元的节点位移向量；浮点数；输入变量
		c_Strain(*3*)	该点的应变；浮点数；输出变量
		c_Stress(*3*)	该点的应力；浮点数；输出变量
		ToTal_B(*3*,*8*)	应变-位移矩阵 **B**；浮点数；输出变量
		detJ	雅可比矩阵行列式；浮点数；输出变量
Cal_Ele_Stress_N4	计算四节点四边形单元内某点（一般是高斯积分点）的应力	*Cal_Ele_Stress_N4*(*i_E*, *i_N*, *X_NODES*, *Y_NODES*, *c_D*, *kesi*, *yita*, *U*, *c_Stress*)	
		i_E	单元号（注：该变量不起作用）；整数；输入变量
		i_N	节点号（注：该变量不起作用）；整数；输入变量
		X_NODES(*4*)	点所在单元 4 个节点的 *x* 坐标；浮点数；输入变量
		Y_NODES(*4*)	点所在单元 4 个节点的 *y* 坐标；浮点数；输入变量
		c_D(*3*,*3*)	应力应变关系矩阵；浮点数；输入变量

		kesi	点相对于所在单元的自然坐标 ξ；浮点数；输入变量
		yita	点相对于所在单元的自然坐标 η；浮点数；输入变量
		U(*8*)	当前单元的 4 个节点的位移向量；浮点数；输入变量
		c_Stress(*3*)	该点的应力；浮点数；输出变量
Cal_Equal_Division_Points	计算线段 AB 的等分点	*Cal_Equal_Division_Points*(*A*, *B*, *Num_Div*, *Yes_Include_Endpoint*, *Div_Points*, *Num_Div_Points*)	
		A(*2*)	A 点坐标；浮点数；输入变量
		B(*2*)	B 点坐标；浮点数；输入变量
		Num_Div	等分份数；整数；输入变量
		Yes_Include_Endpoint	输出结果是否包含线段端点；逻辑变量；输入变量
		Div_Points(*Max_Num_Seg_CalP*,*2*)	等分点的坐标；浮点数；输出变量
		Num_Div_Points	等分点个数；整数；输出变量
Cal_Equivalent_K	计算等效应力强度因子，见（3-36）式	*Cal_Equivalent_K*(*i_C*, *i_Tip*, *c_KI*, *c_KII*, *c_Kequ*)	
		i_C	裂缝号；整数；输入变量
		i_Tip	裂尖号；整数；输入变量
		c_KI	I 型应力强度因子；浮点数；输入变量
		c_KII	II 型应力强度因子；浮点数；输入变量
		c_Kequ	等效应力强度因子；浮点数；输出变量
Cal_F	计算裂尖增强函数的值：F_1~F_4	*Cal_F*(*r*, *theta*, *omega*, *c_mat_type*, *F*)	
		r	裂尖极坐标系坐标 *r*；浮点数；输入变量
		theta	裂尖极坐标系坐标 θ，单位为弧度；浮点数；输入变量
		omega	裂尖所在裂缝片段倾角，单位为弧度（注：该变量不起作用）；浮点数；输入变量
		c_mat_type	裂尖所在单元的材料类型；整数；输入变量
		F(*4*)	裂尖增强函数 F_1~F_4；浮点数；输出变量
Cal_F_dFdx_dFdy	计算裂尖增强函数 F_1~F_4 的偏导数	*Cal_F_dFdx_dFdy*(*r*, *theta*, *omega*, *c_mat_type*, *F*, *dFdx*, *dFdy*)	
		r	裂尖极坐标系坐标 *r*；浮点数；输入变量
		theta	裂尖极坐标系坐标 θ，单位为弧度；浮点数；输入变量
		omega	裂尖所在裂缝片段倾角，单位为弧度；浮点数；输入变量
		c_mat_type	裂尖所在单元的材料类型；整数；输入变量
		F(*4*)	裂尖增强函数 F_1~F_4；浮点数；输出变量
		dFdx(*4*)	裂尖增强函数 F_1~F_4 对 *x* 的偏导数；浮点数；输出变量

		dFdy(*4*)	裂尖增强函数 F_1~F_4 对 y 的偏导数；浮点数；输出变量
Cal_Gauss_Coors	计算全部高斯点的整体坐标	*Cal_Gauss_Coors*(*isub*, *Total_Num_G_P*, *c_G_CoorX*, *c_G_CoorY*)	
		isub	载荷子步号；整数；输入变量
		Total_Num_G_P	总的高斯点数目；整数；输入变量
		c_G_CoorX(*Total_Num_G_P*)	各高斯点的整体 x 坐标；浮点数；输出变量
		c_G_CoorY(*Total_Num_G_P*)	各高斯点的整体 y 坐标；浮点数；输出变量
Cal_Gauss_Points_3D_8nodes	计算高斯积分点的自然坐标及对应的权系数（用于 3D 单元）	*Cal_Gauss_Points_3D_8nodes*(*Num_GP*, *kesi*, *yita*, *zeta*, *weight*)	
		Num_GP	单元高斯点的数目；整数；输入变量
		kesi(*Num_GP*)	单元各高斯点的自然坐标 ξ；浮点数；输出变量
		yita(*Num_GP*)	单元各高斯点的自然坐标 η；浮点数；输出变量
		zeta(*Num_GP*)	单元各高斯点的自然坐标 ζ；浮点数；输出变量
		weight(*Num_GP*)	单元各高斯点的权重；浮点数；输出变量
Cal_Gauss_Points_QUAD	计算高斯积分点自然坐标及对应的权系数	*Cal_Gauss_Points_QUAD*(*Num_GP*, *kesi*, *yita*, *weight*)	
		Num_GP	单元高斯点的数目；整数；输入变量
		kesi(*Num_GP*)	单元各高斯点的自然坐标 ξ；浮点数；输出变量
		yita(*Num_GP*)	单元各高斯点的自然坐标 η；浮点数；输出变量
		weight(*Num_GP*)	单元各高斯点的权重；浮点数；输出变量
Cal_Gauss_Points_QUAD_for_SUBQUAD	计算子四边形剖分积分规则（3.7 节）下的高斯点自然坐标及权系数，每个子四边形有 4 个积分点	*Cal_Gauss_Points_QUAD_for_SUBQUAD*(*Num_QUAD*, *kesi*, *yita*, *weight*)	
		Num_QUAD	单元剖分的子四边形数目（默认为 16）；整数；输入变量
		kesi(*Num_QUAD*4*)	单元各高斯点的自然坐标 ξ；浮点数；输出变量
		yita(*Num_QUAD*4*)	单元各高斯点的自然坐标 η；浮点数；输出变量
		weight(*Num_QUAD*4*)	单元各高斯点的权重；浮点数；输出变量
Cal_HF_CheckConv	水力压裂分析 Newton-Raphson 迭代收敛性检测	*Cal_HF_CheckConv*(*iter*, *Last_Cr_CalP_Aper*, *Last_Cr_CalP_Pres*, *Yes_Conv*)	
		iter	迭代步号；整数；输入变量
		*Last_Cr_CalP_*	上一 Newton-Raphson 迭代步每条裂缝计算

		Aper(*Max_Num_Cr*,*Max_Num_Cr_CalP*)	点开度；浮点数；输入变量
		Last_Cr_CalP_Pres(*Max_Num_Cr*,*Max_Num_Cr_CalP*)	上一 Newton-Raphson 迭代步每条裂缝计算点水压；浮点数；输入变量
		Yes_Conv	是否收敛；逻辑变量；输出变量
Cal_HF_Coor_by_Kesi	水力压裂分析根据水力裂缝高斯点的自然坐标 ξ 计算全局坐标	*Cal_HF_Coor_by_Kesi*(*kesi*, *x1*, *y1*, *x2*, *y2*, *Out_x*, *Out_y*)	
		kesi	高斯点的自然坐标 ξ；浮点数；输入变量
		x1	水力裂缝单元第 1 个节点（计算点）的 *x* 坐标坐标；浮点数；输入变量
		y1	水力裂缝单元第 1 个节点（计算点）的 *y* 坐标坐标；浮点数；输入变量
		x2	水力裂缝单元第 2 个节点（计算点）的 *x* 坐标坐标；浮点数；输入变量
		y2	水力裂缝单元第 2 个节点（计算点）的 *y* 坐标坐标；浮点数；输入变量
		Out_x	高斯点的全局 *x* 坐标；浮点数；输出变量
		Out_y	高斯点的全局 *y* 坐标；浮点数；输出变量
Cal_HF_Crack_Points_Info_Linear	获得裂缝计算点的类型、数目、坐标、方向、所在裂缝片段号、所在单元号等	*Cal_HF_Crack_Points_Info_Linear*(*isub*)	
		isub	载荷子步号（或迭代步号）；整数；输入变量
Cal_HF_delta_Time_Linear	计算水力压裂分析时间增量 Δt	*Cal_HF_delta_Time_Linear*(*Counter*, *Last_Cr_CalP_Aper*, *total_Time*, *delta_Time*)	
		Counter	总的迭代步数；整数；输入变量
		Last_Cr_CalP_Aper(*Max_Num_Cr*,*Max_Num_Cr_CalP*)	上一 Newton-Raphson 迭代步每条裂缝计算点开度；浮点数；输入变量
		total_Time	当前压裂时间；浮点数；输入变量
		delta_Time	时间增量 Δt；浮点数；输出变量
Cal_HF_Flow_Quan	计算流体节点的流速及流量	*Cal_HF_Flow_Quan*(*ifra*, *iter*, *Counter*, *total_time*)	
		ifra	破裂步号；整数；输入变量
		iter	迭代步号；整数；输入变量
		Counter	总的迭代步数；整数；输入变量
		total_time	当前压裂时间；浮点数；输入变量
Cal_HF_HFNode_Location	计算裂缝片段 AB 和单元边线的交点，作为流体单元的节点	*Cal_HF_HFNode_Location*(*A*, *B*, *Div_Points*, *Num_Div_Points*)	
		A(2)	裂缝片段 A 点坐标；浮点数；输入变量
		B(2)	裂缝片段 B 点坐标；浮点数；输入变量
		Div_Points(*Max*	流体单元节点坐标；浮点数；输出变量

		_Num_Seg_CalP,*2*)	
		Num_Div_Points	流体单元节点数目；整数；输出变量
Cal_HF_Initial_Aperture	水力压裂分析各个破裂步初始 N-R 迭代步的裂缝开度	*Cal_HF_Initial_Aperture*(*iter*, *ifra*, *Counter*, *Last_Cracks_CalP_Num_ifra*, *Last_Cr_CalP_Aper_ifra*, *Initial_Cr_CalP_Aper*)	
		iter	迭代步号；整数；输入变量
		ifra	破裂步号；整数；输入变量
		Counter	总的迭代步数；整数；输入变量
		Last_Cracks_CalP_Num_ifra(*Max_Num_Cr*, *Max_Num_Cr_CalP*)	上一步破裂步每条裂缝计算点水压；浮点数；输入变量
		Last_Cr_CalP_Aper_ifra(*Max_Num_Cr*)	上一破裂步每条裂缝计算点个数；整数；输入变量
		Initial_Cr_CalP_Aper(*Max_Num_Cr*,*Max_Num_Cr_CalP*)	初始 N-R 迭代步的裂缝开度；浮点数；输出变量
Cal_HF_Jacobian_NR	计算水力压裂 N-R 迭代的雅可比矩阵	*Cal_HF_Jacobian_NR*(*Counter*,*NR_Deri*,*Total_FD*,*num_FreeD*,*num_free_CalP*,*globalK*,*Coupled_Q*,*freeDOF*, *freeDOF_HF*, *Local_freeDOF_HF*,*delta_Time*, *H*)	
		Counter	总的迭代步数；整数；输入变量
		NR_Deri(*Total_FD+num_Tol_CalP_Water*,*Total_FD+num_Tol_CalP_Water*)	雅可比矩阵；浮点数；输出变量
		Total_FD	总的自由度数目；整数；输入变量
		num_FreeD	活动自由度数目；整数；输入变量
		num_free_CalP	总的流体自由度数目；整数；输入变量
		globalK(*Total_FD*,*Total_FD*)	刚度矩阵 **K**；浮点数；输入变量
		Coupled_Q(*num_Tol_CalP_Water*,*num_Tol_CalP_Water*)	流固耦合矩阵 **Q**；浮点数；输入变量
		freeDOF(*Total_FD*)	活动自由度向量；整数；输入变量
		freeDOF_HF(*num_Tol_CalP_*	活动流体节点（流体压力非零）自由度向量（自由度编号在位移自由度基础上累加）；

		Water)	整数；输入变量
		Local_freeDOF_HF(*num_Tol_CalP_Water*)	活动流体节点（流体压力非零）自由度向量（自由度编号不在位移自由度基础上累加）；整数；输入变量
		delta_Time	水力压裂分析时间增量 Δt；浮点数；输入变量
		H(*num_Tol_CalP_Water*,*num_Tol_CalP_Water*)	水力压裂分析的 **H** 矩阵；浮点数；输入变量
Cal_HF_Line_Searching	Newton-Raphson 迭代的线搜索，用于提升水力压裂分析 N-R 迭代的收敛性	*Cal_HF_Line_Searching*(*ifra*,*iter*,*Counter_Iter*,*num_ALlDOF*,*c_Total_FD*,*num_FreeD*,*num_free_CalP*,*Total_freeDOF*,*c_num_Tol_CalP_Water*,*num_Total_FD*,*freeDOF_HF*,*Local_freeDOF_HF*,*NR_Deri*,*delta_x*,*c_R*,*Initial_DISP*,*c_DISP*,*c_freeDOF*,*c_CalP_Pres*,*c_globalK*,*Coupled_Q*,*F_U*,*c_Temp_Cr_CalP_Aper*,*c_Temp_total_Time*,*Num_Line_Search*,*total_Time*,*x*)	
		ifra	破裂步号；整数；输入变量
		iter	迭代步号；整数；输入变量
		Counter_Iter	总的迭代步数；整数；输入变量
		num_ALlDOF	总的自由度数目（位移自由度+流体自由度）；整数；输入变量
		c_Total_FD	位移自由度数目；整数；输入变量
		num_FreeD	活动位移自由度数目；整数；输入变量
		num_free_CalP	活动流体自由度数目；整数；输入变量
		Total_freeDOF(*num_ALlDOF*)	总的活动自由度编号列表（位移自由度+流体自由度）向量；整数；输入变量
		c_num_Tol_CalP_Water	流体自由度数目；整数；输入变量
		num_Total_FD	总的活动自由度数目（位移自由度+流体自由度）；整数；输入变量
		freeDOF_HF(*c_num_Tol_CalP_Water*)	活动流体自由度编号列表（编号在位移自由度基础上累加）；整数；输入变量
		Local_freeDOF_HF(*c_num_Tol_CalP_Water*)	活动流体自由度编号列表（编号不在位移自由度基础上累加）；整数；输入变量
		NR_Deri(*c_Total_FD*+*c_num_Tol_CalP_Water*, *c_Total_FD*+*c_num_Tol_CalP_Water*)	Newton-Raphson 迭代的雅可比矩阵 **J**；浮点数；输入变量
		delta_x(*num_FreeD*+*num_free_CalP*)	Newton-Raphson 迭代增量向量；浮点数；输入变量

		c_R(*num_ALlDOF*)	Newton-Raphson 迭代残差 **R** 向量；浮点数；输入变量
		Initial_DISP(*c_Total_FD*)	初始迭代步的位移向量；浮点数；输入变量
		c_DISP(*c_Total_FD*)	位移向量；浮点数；输入变量
		c_freeDOF(*c_Total_FD*)	活动位移自由度编号列表；整数；输入变量
		c_CalP_Pres(*c_num_Tol_CalP_Water*)	流体压力向量；浮点数；输入变量
		c_globalK(*c_Total_FD*,*c_Total_FD*)	刚度矩阵 **K**；浮点数；输入变量
		Coupled_Q(*c_Total_FD*,*c_num_Tol_CalP_Water*)	流固耦合矩阵 **Q**；浮点数；输入变量
		F_U(*c_Total_FD*)	外载荷向量；浮点数；输入变量
		c_Temp_Cr_CalP_Aper(*Max_Num_Cr*, *Max_Num_Cr_CalP*)	各裂缝计算点的开度；浮点数；输入变量
		c_Temp_total_Time	输入的压裂时间；浮点数；输入变量
		Num_Line_Search	线性搜索次数计数；整数；输入输出变量
		total_Time	压裂时间；浮点数；输入输出变量
		x(*num_FreeD*+*num_free_CalP*)	线性搜索结束后的 Newton-Raphson 迭代结果向量；浮点数；输出变量
Cal_HF_Matrix_H_Linear	计算水力压裂分析的 **H** 矩阵	*Cal_HF_Matrix_H_Linear*(*ifra*, *Counter*, *Matrix_H*, *Last_Cr_CalP_Aper*)	
		ifra	载荷子步号；整数；输入变量
		Counter	总的迭代步数；整数；输入变量
		Matrix_H(*num_Tol_CalP_Water*,*num_Tol_CalP_Water*)	**H** 矩阵；浮点数；输出变量
		Last_Cr_CalP_Aper(*Max_Num_Cr*,*Max_Num_Cr_CalP*)	上一迭代步的裂缝开度；浮点数；输入变量

Cal_HF_Matrix_Q_Linear	计算水力压裂分析的流固耦合 **Q** 矩阵	*Cal_HF_Matrix_Q_Linear*(*Counter*, *Coupled_Q*, *c_Total_FD*)	
		Counter	总的迭代步数；整数；输入变量
		Coupled_Q(*c_Total_FD*,*num_Tol_CalP_Water*)	流固耦合矩阵 **Q**；浮点数；输出变量
		c_Total_FD	总的自由度数目；整数；输入变量
Cal_HF_Modify_Jacobian	水力压裂分析 N-R 迭代过程中，对雅可比矩阵进行修正	*Cal_HF_Modify_Jacobian*(*Counter*, *NR_Deri*, *Total_FD*, *num_free_CalP*, *freeDOF_HF*, *Local_freeDOF_HF*, *delta_Time*, *H*)	
		Counter	总的迭代步数；整数；输入变量
		NR_Deri(*Total_FD+num_Tol_CalP_Water*,*Total_FD+num_Tol_CalP_Water*)	雅可比矩阵；浮点数；输入输出变量
		Total_FD	总的自由度数目；整数；输入变量
		num_free_CalP	总的活动流体自由度数目；整数；输入变量
		freeDOF_HF(*num_Tol_CalP_Water*)	活动流体节点（流体压力非零）自由度向量（自由度编号在位移自由度基础上累加）；整数；输入变量
		Local_freeDOF_HF(*num_Tol_CalP_Water*)	活动流体节点（流体压力非零）自由度向量（自由度编号不在位移自由度基础上累加）；整数；输入变量
		delta_Time	水力压裂分析时间增量 Δt；浮点数；输入变量
		H(*num_Tol_CalP_Water*,*num_Tol_CalP_Water*)	**H** 矩阵；浮点数；输入变量
Cal_HF_Resid	水力压裂分析 N-R 迭代计算残差 **R**	*Cal_HF_Resid*(*ifra*, *iter*, *Counter_Iter*, *R*, *Total_FD*, *num_FreeD*, *num_free_CalP*, *globalK*, *Coupled_Q*, *F_U*, *Last_DISP*, *Last_Last_DISP*, *freeDOF*, *freeDOF_HF*, *Local_freeDOF_HF*, *Last_CalP_Pres*, *delta_Time*, *H*, *c_S*)	
		ifra	载荷子步号；整数；输入变量
		iter	Newton-Raphson 迭代步号；整数；输入变量
		Counter_Iter	总的迭代步数；整数；输入变量
		R(*Total_FD+num_Tol_CalP_Water*)	残差 **R**；浮点数；输出变量
		Total_FD	总的位移自由度数目；整数；输入变量
		num_FreeD	活动位移自由度数目；整数；输入变量
		num_free_CalP	总的活动流体自由度数目；整数；输入变量
		globalK(*Total_FD*,*Total_FD*)	刚度矩阵 **K**；浮点数；输入变量

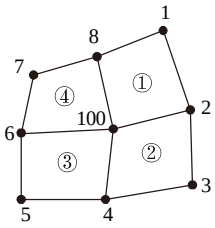
		Coupled_Q(*Total_FD*,*num_Tol_CalP_Water*)	流固耦合矩阵 **Q**；浮点数；输入变量
		F_U(*Total_FD*)	外载荷向量；浮点数；输入变量
		Last_DISP(*Total_FD*)	上一步的位移向量；浮点数；输入变量
		Last_Last_DISP(*Total_FD*)	上上步的位移向量；浮点数；输入变量
		freeDOF(*Total_FD*)	活动位移自由度向量；整数；输入变量
		freeDOF_HF(*num_Tol_CalP_Water*)	活动流体节点（流体压力非零）自由度向量（自由度编号在位移自由度基础上累加）；整数；输入变量
		Local_freeDOF_HF(*num_Tol_CalP_Water*)	活动流体节点（流体压力非零）自由度向量（自由度编号不在位移自由度基础上累加）；整数；输入变量
		Last_CalP_Pres(*num_Tol_CalP_Water*)	上一步的流体节点压力；浮点数；输入变量
		delta_Time	水力压裂分析时间增量 Δt；浮点数；输入变量
		H(*num_Tol_CalP_Water*,*num_Tol_CalP_Water*)	**H** 矩阵；浮点数；输入变量
		c_S(*num_Tol_CalP_Water*)	**S** 向量；浮点数；输入变量
Cal_HF_S	计算水力压裂分析的 **S** 向量	*Cal_HF_S*(*Counter*, *S*, *total_time*)	
		Counter	总的迭代步数；整数；输入变量
		S(*num_Tol_CalP_Water*)	**S** 向量；浮点数；输出变量
		total_time	压裂时间；浮点数；输入变量
Cal_JDomain_Element_of_Point	确定积分圆内的单元号，用于采用相互作用积分法计算应力强度因子	*Cal_JDomain_Element_of_Point*(*x*, *y*, *r_JDomain*, *J_Elem*, *num_J_Elem*, *Q_Elem_Nodes*)	
		x	裂尖 *x* 坐标；浮点数；输入变量
		y	裂尖 *y* 坐标；浮点数；输入变量
		r_JDomain	积分圆半径；浮点数；输入变量
		J_Elem(*300*)	积分圆内的单元号列表；整数；输出变量
		num_J_Elem	积分圆内的单元数目；整数；输出变量
		Q_Elem_Nodes(*num_Elem*,*4*)	积分圆内的各单元各节点的权函数 *q*（见 3.2.4 小节）的值；浮点数；输出变量
Cal_KesiYita_by_Coor	根据某点的整体坐标计算自然坐标	*Cal_KesiYita_by_Coor*(*Point*, *c_elem*, *Kesi*, *Yita*)	
		Point(*2*)	点的坐标；浮点数；输入变量
		c_elem	点所在单元号；整数；输入变量

		Kesi	点相对于所在单元的自然坐标 ξ；浮点数；输出变量
		Yita	点相对于所在单元的自然坐标 η；浮点数；输出变量
Cal_KesiYita_by_Coor_3D	根据某点的整体坐标计算自然坐标（用于 3D 问题）	*Cal_KesiYita_by_Coor_3D*(*Point*, *c_elem*, *Kesi*, *Yita*, *Zeta*)	
		Point(*3*)	点的坐标；浮点数；输入变量
		c_elem	点所在单元号；整数；输入变量
		Kesi	点相对于所在单元的自然坐标 ξ；浮点数；输出变量
		Yita	点相对于所在单元的自然坐标 η；浮点数；输出变量
		Zeta	点相对于所在单元的自然坐标 ζ；浮点数；输出变量
Cal_Length_of_ELe_by_Coors_Ave	计算某点所在单元的特征长度	*Cal_Length_of_ELe_by_Coors_Ave*(*x*, *y*, *Ave_Length*)	
		x	点的 *x* 坐标；浮点数；输入变量
		y	点的 *y* 坐标；浮点数；输入变量
		Ave_Length	点所在单元的特征长度（单元面积的平方根）；浮点数；输出变量
Cal_N	计算四节点四边形单元内某点的形函数矩阵	*Cal_N*(*kesi*, *yita*, *N*)	
		kesi	点相对于所在单元的自然坐标 ξ；浮点数；输入变量
		yita	点相对于所在单元的自然坐标 η；浮点数；输入变量
		N(*2*,*8*)	形函数矩阵；浮点数；输出变量
Cal_N_3D	计算八节点六面体单元内某点的形函数矩阵	*Cal_N_3D*(*kesi*, *yita*, *zeta*, *N*)	
		kesi	点相对于所在单元的自然坐标 ξ；浮点数；输入变量
		yita	点相对于所在单元的自然坐标 η；浮点数；输入变量
		zeta	点相对于所在单元的自然坐标 ζ；浮点数；输入变量
		N(*3*,*24*)	形函数矩阵；浮点数；输出变量
Cal_N_dNdkesi_J_detJ	计算四节点四边形单元内某点的形函数矩阵、形函数导数矩阵、雅可比矩阵（见 2.4 节）以及雅可比矩阵的行列式	*Cal_N_dNdkesi_J_detJ*(*kesi*, *yita*, *X_NODES*, *Y_NODES*, *detJ*, *J*, *N*, *dNdkesi*)	
		kesi	点相对于所在单元的自然坐标 ξ；浮点数；输入变量
		yita	点相对于所在单元的自然坐标 η；浮点数；输入变量
		X_NODES(*4*)	点所在单元节点的 *x* 坐标；浮点数；输入变量
		Y_NODES(*4*)	点所在单元节点的 *y* 坐标；浮点数；输入变量
		detJ	雅可比矩阵的行列式；浮点数；输出变量

		J(*2*,*2*)	雅可比矩阵；浮点数；输出变量
		N	形函数矩阵；浮点数；输出变量
		dNdkesi(*4*,*2*)	形函数导数矩阵；浮点数；输出变量
Cal_N_dNdkesi_J_detJ_3D	计算八节点六面体单元内某点（一般为高斯点）的形函数矩阵、形函数导数矩阵、雅可比矩阵以及雅可比矩阵的行列式	*Cal_N_dNdkesi_J_detJ_3D*(*i_G*, *kesi*, *yita*, *zeta*, *X_NODES*, *Y_NODES*, *Z_NODES*, *detJ*, *J*, *N*, *dNdkesi*)	
		i_G	单元高斯点号；整数；输入变量
		kesi	点相对于所在单元的自然坐标 ξ；浮点数；输入变量
		yita	点相对于所在单元的自然坐标 η；浮点数；输入变量
		zeta	点相对于所在单元的自然坐标 ζ；浮点数；输入变量
		X_NODES(*8*)	点所在单元节点的 *x* 坐标；浮点数；输入变量
		Y_NODES(*8*)	点所在单元节点的 *y* 坐标；浮点数；输入变量
		Z_NODES(*8*)	点所在单元节点的 *z* 坐标；浮点数；输入变量
		detJ	雅可比矩阵的行列式；浮点数；输出变量
		J(*3*,*3*)	雅可比矩阵；浮点数；输出变量
		N(*3*,*24*)	形函数矩阵；浮点数；输出变量
		dNdkesi(*8*,*3*)	形函数导数矩阵；浮点数；输出变量
Cal_NMass_Matrix_Circ_Incl	计算形函数矩阵（含增强函数），用于组集圆形夹杂的质量矩阵	*Cal_NMass_Matrix_Circ_Incl*(*kesi*, *yita*, *i_Incl*, *i_E*, *i_G*, *c_NN*, *c_X_NODES*, *c_Y_NODES*, *tem_N*, *num_tem_N*)	
		kesi	点相对于所在单元的自然坐标 ξ；浮点数；输入变量
		yita	点相对于所在单元的自然坐标 η；浮点数；输入变量
		i_Incl	夹杂号；整数；输入变量
		i_E	单元号；整数；输入变量
		i_G	高斯点号；整数；输入变量
		c_NN(4)	标准形函数（写成向量形式，即$[N_1, N_2, N_3, N_4]^T$）；浮点数；输入变量
		c_X_NODES(*4*)	点所在单元节点的 *x* 坐标；浮点数；输入变量
		c_Y_NODES(*4*)	点所在单元节点的 *y* 坐标；浮点数；输入变量
		tem_N(*2*,*80*)	形函数矩阵；浮点数；输出变量
		num_tem_N	形函数矩阵元素的实际列数，*tem_N*(*2*, *num_tem_N*)；整数；输出变量
Cal_NMass_Matrix_Crack	计算形函数矩阵（含增强函数），用于组集裂缝的质量	*Cal_NMass_Matrix_Crack*(*kesi*, *yita*, *i_C*, *i_E*, *i_G*, *c_NN*, *c_X_NODES*, *c_Y_NODES*, *tem_N*, *num_tem_N*)	
		kesi	点相对于所在单元的自然坐标 ξ；浮点数；

	矩阵		输入变量
		yita	点相对于所在单元的自然坐标 η；浮点数；输入变量
		i_C	裂缝号；整数；输入变量
		i_E	单元号；整数；输入变量
		i_G	高斯点号；整数；输入变量
		c_NN(4)	标准形函数（写成向量形式，即$[N_1, N_2, N_3, N_4]^T$）；浮点数；输入变量
		c_X_NODES(*4*)	点所在单元节点的 *x* 坐标；浮点数；输入变量
		c_Y_NODES(*4*)	点所在单元节点的 *y* 坐标；浮点数；输入变量
		tem_N(*2*,*80*)	形函数矩阵；浮点数；输出变量
		num_tem_N	形函数矩阵元素的实际列数，*tem_N*(*2*, *num_tem_N*)；整数；输出变量
Cal_NMass_Matrix_Hl	计算形函数矩阵（含增强函数），用于组集孔洞的质量矩阵	*Cal_NMass_Matrix_Hl*(*kesi*, *yita*, *i_H*, *i_E*, *i_G*, *c_NN*, *c_X_NODES*, *c_Y_NODES*, *tem_N*, *num_tem_N*)	
		kesi	点相对于所在单元的自然坐标 ξ；浮点数；输入变量
		yita	点相对于所在单元的自然坐标 η；浮点数；输入变量
		i_H	孔洞号；整数；输入变量
		i_E	单元号；整数；输入变量
		i_G	高斯点号；整数；输入变量
		c_NN(4)	标准形函数（写成向量形式，即$[N_1, N_2, N_3, N_4]^T$）；浮点数；输入变量
		c_X_NODES(*4*)	点所在单元节点的 *x* 坐标；浮点数；输入变量
		c_Y_NODES(*4*)	点所在单元节点的 *y* 坐标；浮点数；输入变量
		tem_N(*2*,*80*)	形函数矩阵；浮点数；输出变量
		num_tem_N	形函数矩阵元素的实际列数，*tem_N*(*2*, *num_tem_N*)；整数；输出变量
Cal_Point_Aperture	计算裂缝上某点的开度	*Cal_Point_Aperture*(*c_Crack*, *Point*, *c_DISP*, *Cr_Omega*, *Aperture*)	
		c_Crack	裂缝号；整数；输入变量
		Point(2)	点的坐标；浮点数；输入变量
		c_DISP(*Total_FD*)	位移自由度向量；浮点数；输入变量
		Cr_Omega	点所在裂缝片段的倾角（弧度）；浮点数；输入变量
		Aperture	裂缝开度；浮点数；输出变量
Cal_Point_Apertu	计算裂缝上某点的	*Cal_Point_Aperture_and_Tangent_disp*(*c_Crack*, *Point*, *c_DISP*,	

re_and_Tangent_disp	开度以及切向相对滑动位移	*Cr_Omega*, *Aperture*, *Tangent_disp*)	
		c_Crack	裂缝号；整数；输入变量
		Point(2)	点的坐标；浮点数；输入变量
		c_DISP(*Total_FD*)	位移自由度向量；浮点数；输入变量
		Cr_Omega	点所在裂缝片段的倾角（弧度）；浮点数；输入变量
		Aperture	裂缝开度；浮点数；输出变量
		Tangent_disp	切向相对滑动位移；浮点数；输出变量
Cal_Point_Aperture_and_Tangent_disp_NoFEM	计算裂缝上某点的开度以及切向相对滑动位移（与 *Cal_Point_Aperture_and_Tangent_disp* 相比，输入变量位移向量仅包含增强自由度）	*Cal_Point_Aperture_and_Tangent_disp_NoFEM*(*c_Crack*, *Point*, *U_e*, *Cr_Omega*, *Aperture*, *Tangent_disp*)	
		c_Crack	裂缝号；整数；输入变量
		Point(2)	点的坐标；浮点数；输入变量
		U_e(*Enrich_Freedom*)	增强位移自由度向量；浮点数；输入变量
		Cr_Omega	点所在裂缝片段的倾角（弧度）；浮点数；输入变量
		Aperture	裂缝开度；浮点数；输出变量
		Tangent_disp	切向相对滑动位移；浮点数；输出变量
Cal_Point_Aperture_NoFEM	计算裂缝上某点的开度（与 *Cal_Point_Aperture* 相比，输入变量位移向量仅包含增强自由度）	*Cal_Point_Aperture_NoFEM*(*c_Crack*, *Point*, *U_e*, *Cr_Omega*, *Aperture*)	
		c_Crack	裂缝号；整数；输入变量
		Point(2)	点的坐标；浮点数；输入变量
		U_e(*Enrich_Freedom*)	增强位移自由度向量；浮点数；输入变量
		Cr_Omega	点所在裂缝片段的倾角（弧度）；浮点数；输入变量
		Aperture	裂缝开度；浮点数；输出变量
Cal_SIFs_DIM	位移插值法法计算应力强度因子	*Cal_SIFs_DIM*(*iter*, *c_DISP*)	
		iter	载荷子步号；整数；输入变量
		c_DISP(*Total_FD*)	位移自由度向量；浮点数；输入变量
Cal_SIFs_DIM_3D	位移插值法法计算应力强度因子（用于 3D 问题）	*Cal_SIFs_DIM_3D*(*iter*, *c_DISP*)	
		iter	载荷子步号；整数；输入变量
		c_DISP(*Total_FD*)	位移自由度向量；浮点数；输入变量
Cal_SIFs_IIM	相互作用积分法计算应力强度因子	*Cal_SIFs_IIM*(*iter*, *Display_info*, *c_DISP*)	
		iter	载荷子步号；整数；输入变量
		Display_info	是否输出反馈信息；逻辑变量；输入变量
		c_DISP(*Total_FD*)	位移自由度向量；浮点数；输入变量
Cal_Sign	符号函数：若>0.0，则返回 1.0；若=0.0，	*Cal_Sign*(*Variable*, *Sign_O*)	
		Variable	输入变量；浮点数；输入变量

	则返回 0.0；若<0.0，则返回-1.0	*Sign_O*	输出变量；浮点数；输入变量
Cal_Sign_1_and_0	符号函数：若>0.0，则返回 1.0；若=0.0，则返回 0.0；若<0.0，则返回 0.0	*Cal_Sign_1_and_0*(*Variable*, *Sign_O*)	
		Variable	输入变量；浮点数；输入变量
		Sign_O	输出变量；浮点数；输入变量
Cal_Signed_Distance	计算点 C 到线段 AB 的符号距离（见 3.4.5 小节）	*Cal_Signed_Distance*(*Line_AB*, *Point_C*, *S_Distance*)	
		Line_AB(*2*,*2*)	线段 AB 坐标；浮点数；输入变量
		Point_C(*2*)	点 C 坐标；浮点数；输入变量
		S_Distance	符号距离；浮点数；输出变量
Cal_Support_Domain_of_Node	获得节点的支撑域对应的单元号以及支撑域单元边线节点号，以下图为例说明： 	*Cal_Support_Domain_of_Node*(*iNode*,*DOMAIN_Outline*, *m_DOMAIN_Outline*, *Domain_El*, *n_Domain_El*)	
		iNode	节点号（左图，该值为 100）；整数；输入变量
		DOMAIN_Outline(*20*,*2*)	支撑域的各段单元边线的两个节点编号（左图，*DOMAIN_Outline*(*1:8*,*1*)=[1, 2, 3, 4, 5, 6, 7, 8, 1]^T，*DOMAIN_Outline*(*1:8*,*2*)=[2, 3, 4, 5, 6, 7, 8, 1, 2]^T）；整数；输出变量
		m_DOMAIN_Outline	支撑域的单元边线段数（左图，该值为 8，即支撑域由 8 段单元边线组成：1-2；2-3；3-4；4-5；5-6；6-7；7-8；8-1）；整数；输出变量
		Domain_El(*10*)	支撑域内的单元号列表；整数；输出变量
		n_Domain_El	支撑域内的单元数目；整数；输出变量
Cal_Vol_8nodes	计算八节点六面体单元的体积	*Cal_Vol_8nodes*(*i_Elem*,*x*,*y*,*z*,*vol*)	
		i_Elem	单元号；整数；输入变量
		x(*8*)	8 个节点的 *x* 坐标；浮点数；输入变量
		y(*8*)	8 个节点的 *y* 坐标；浮点数；输入变量
		z(*8*)	8 个节点的 *z* 坐标；浮点数；输入变量
		vol	单元体积；浮点数；输出变量
Cal_Weighted_Ave_Stress_of_Point	计算某点的加权平均应力	*Cal_Weighted_Ave_Stress_of_Point*(*c_Key_Ave_Stress*, *Point*, *Search_R*,*a_Weight*,*l_Ordina*,*WA_S_xx*,*WA_S_yy*,*WA_S_xy*)	
		c_Key_Ave_Stress	加权平均算法；整数；输入变量
		Point(*2*)	点的坐标；浮点数；输入变量
		Search_R	搜索半径，即加权计算以该值为半径的原内的高斯点的应力；浮点数；输入变量
		a_Weight	参数 *a*（仅在 *c_Key_Ave_Stress*=1 时需要）；浮点数；输入变量
		l_Ordina	参数 *l*（仅在 *c_Key_Ave_Stress*=2 时需要）；浮点数；输入变量
		WA_S_xx	加权平均应力的 *x* 方向分量；浮点数；输出变量

		WA_S_yy	加权平均应力的 *y* 方向分量；浮点数；输出变量
		WA_S_xy	加权平均应力的 *xy* 剪应力分量；浮点数；输出变量

4 以 *Tool* 开头的工具子程序

PhiPsi 源程序中有大量以 *Tool* 开头的子程序，顾名思义，这些子程序以工具程序的形式存在，表 2 和表 3 列出的子程序会大量调用这些工具子程序。表 4 列出了 PhiPsi 源程序以 *Tool* 开头的子程序。

表 4 PhiPsi 程序以 *Tool* 开头的工具子程序用途及输入输出变量说明（按名称排序）

子程序名	子程序功能	子程序输入输出接口	
Tool_Area_Polygon	给定多边形顶点坐标，计算多边形的面积	*Tool_Area_Polygon*(*x*, *y*, *nb*, *area*)	
		x(*nb*)	组成多边形的顶点 *x* 坐标数组；浮点数；输入变量
		y(*nb*)	组成多边形的顶点 *y* 坐标数组；浮点数；输入变量
		nb	组成多边形的顶点数目；整数；输入变量
		area	多边形面积；浮点数；输出变量
Tool_Area_Tri_2D	给定顶点坐标，计算二维空间中三角形面积	*Tool_Area_Tri_2D*(*ABC*, *area*)	
		ABC(*3*,*2*)	三角形顶点坐标，*ABC*(*i*,*1*)为第 *i* 个顶点的 *x* 坐标，*ABC*(*i*,*2*)为第 *i* 个顶点的 *y* 坐标）；浮点数；输入变量
		area	三角形面积；浮点数；输出变量
Tool_Area_Tri_3D	给定顶点坐标，计算三维空间中三角形面积	*Tool_Area_Tri_3D*(*ABC*, *area*)	
		ABC(*3*,*3*)	空间三角形顶点坐标，*ABC*(*i*,*1*)为第 *i* 个顶点的 *x* 坐标，*ABC*(*i*,*2*)为第 *i* 个顶点的 *y* 坐标，*ABC*(*i*,*3*)为第 *i* 个顶点的 *z* 坐标）；浮点数；输入变量
		area	三角形面积；浮点数；输出变量
Tool_abc_of_Parabola	给定 3 个点，计算通过这 3 个点的抛物线方程的系数 *a*、*b*、*c*	*Tool_abc_of_Parabola*(*Point_1*, *Point_2*, *Point_3*, *a*, *b*, *c*)	
		Point_1(*2*)	点 1 坐标；浮点数；输入变量
		Point_2(*2*)	点 2 坐标；浮点数；输入变量
		Point_2(*2*)	点 3 坐标；浮点数；输入变量
		a	抛物线方程的系数 *a*；浮点数；输出变量
		b	抛物线方程的系数 *b*；浮点数；输出变量
		c	抛物线方程的系数 *c*；浮点数；输出变量
Tool_Dis_Point_to_3D_Quad	计算空间中一点到空间四边形(P_1, P_2, P_3, P_4)的距离及垂足，思路是拆分成两个三角	*Tool_Dis_Point_to_3D_Quad*(*Point*,*Quad_P1*,*Quad_P2*, *Quad_P3*, *Quad_P4*, *Distance*, *PER*, *Yes_PER_in*, *Yes_PER_on*)	
		Point(*3*)	空间点坐标；浮点数；输入变量

	形(P_1, P_2, P_3)和(P_1, P_3, P_4)，然后分别调用 *Tool_Cal_Dis_Point_to_3D_Tri*	*Quad_P1*(*3*)	空间四边形点 1 坐标；浮点数；输入变量
		Quad_P2(*3*)	空间四边形点 2 坐标；浮点数；输入变量
		Quad_P3(*3*)	空间四边形点 3 坐标；浮点数；输入变量
		Quad_P4(*3*)	空间四边形点 4 坐标；浮点数；输入变量
		Distance	点到空间四边形的距离；浮点数；输出变量
		PER(*3*)	垂足坐标；浮点数；输出变量
		Yes_PER_in	垂足是否在四边形内；逻辑变量；输出变量
		Yes_PER_on	垂足是否在四边形边界上；逻辑变量；输出变量
Tool_Dis_Point_to_3D_Tri	计算空间中一点到空间三角形(P_1, P_2, P_3)的距离及垂足	*Tool_Dis_Point_to_3D_Tri*(*Point*,*Tri_P1*,*Tri_P2*,*Tri_P3*, *Distance*, *PER*, *Yes_PER_in*, *Yes_PER_on*)	
		Point(3)	空间点坐标；浮点数；输入变量
		Tri_P1(3)	空间三角形点 1 坐标；浮点数；输入变量
		Tri_P2(3)	空间三角形点 2 坐标；浮点数；输入变量
		Tri_P3(3)	空间三角形点 3 坐标；浮点数；输入变量
		Distance	点到空间三角形的距离；浮点数；输出变量
		PER(3)	垂足坐标；浮点数；输出变量
		Yes_PER_in	垂足是否在三角形内；逻辑变量；输出变量
		Yes_PER_on	垂足是否在三角形边界上；逻辑变量；输出变量
Tool_Dis_Point_to_Poly	计算点 P 到多边形的距离，若 P 在多边形外则为正，在多边形内则为负（注意：这里的多边形是闭合形式的，即 *npol*= 边数+1）	*Tool_Dis_Point_to_Poly*(*x*, *y*, *xpol*, *ypol*, *npol*, *Dis*)	
		x	P 点的 *x* 坐标；浮点数；输入变量
		y	P 点的 *y* 坐标；浮点数；输入变量
		xpol(*npol*)	多边形的顶点 *x* 坐标；浮点数；输入变量
		xpol(*npol*)	多边形的顶点 *y* 坐标；浮点数；输入变量
		npol	*npol*= 顶点数目+1；整数；输入变量
		Dis	点 *P* 到多边形的距离；浮点数；输出变量
Tool_Dis_Point_to_Seg	计算点 C 到线段 AB 的距离（法线距离）	*Tool_Dis_Point_to_Seg*(*A*, *B*, *C*, *Dis*)	
		A(*2*)	AB 端点 A 坐标；浮点数；输入变量
		B(*2*)	AB 端点 B 坐标；浮点数；输入变量
		C(*2*)	点 C 坐标；浮点数；输入变量
		Dis	点 C 到 AB 的距离；浮点数；输出变量
Tool_Error_Between_3_Variables	计算 3 个变量 *A*、*B* 和 *C* 之间的最大误差（Error = abs(x-y) / max(x,y)）	*Tool_Error_Between_3_Variables*(*A*, *B*, *C*, *Error*)	
		A	变量 *A*；浮点数；输入变量
		B	变量 *B*；浮点数；输入变量
		C	变量 *C*；浮点数；输入变量
		Error	最大误差；浮点数；输出变量
Tool_Foot_Point_to_Seg	计算点 C 到线段 AB 的垂足	*Tool_Foot_Point_to_Seg*(*A*, *B*, *C*, *Foot*)	
		A(*2*)	AB 端点 A 坐标；浮点数；输入变量
		B(*2*)	AB 端点 B 坐标；浮点数；输入变量
		C(*2*)	点 C 坐标；浮点数；输入变量
		Foot(*2*)	垂足坐标；浮点数；输出变量
Tool_Intersectio	计算线段 AB 和 CD	*Tool_Intersection*(*A*, *B*, *C*, *D*, *X*, *Y*, *Yes_Cross*)	

n	的交叉点	*A*(*2*)	AB 端点 A 坐标；浮点数；输入变量
		B(*2*)	AB 端点 B 坐标；浮点数；输入变量
		C(2)	CD 端点 C 坐标；浮点数；输入变量
		D(2)	CD 端点 D 坐标；浮点数；输入变量
		X	交叉点 *x* 坐标；浮点数；输出变量
		Y	交叉点 *y* 坐标；浮点数；输出变量
		Yes_Cross	线段 AB 和 CD 是否有交叉点；逻辑变量；输出变量
Tool_Length_of_Crack	计算指定裂缝的长度，即各裂缝片段长度的和	*Tool_Length_of_Crack*(*i_Crack*, *Length_Crack*)	
		i_Crack	裂缝号；整数；输入变量
		Length_Crack	裂缝长度；浮点数；输出变量
Tool_Normal_vector_of_3D_Tri	计算空间三角形的法向量	*Tool_Normal_vector_of_3D_Tri*(*Tri_P1*,*Tri_P2*,*Tri_P3*,*Np*)	
		Tri_P1(*3*)	空间三角形 P_1 点坐标；浮点数；输入变量
		Tri_P2(*3*)	空间三角形 P_2 点坐标；浮点数；输入变量
		Tri_P3(*3*)	空间三角形 P_3 点坐标；浮点数；输入变量
		Np(*3*)	法向量；浮点数；输出变量
Tool_Point_Giv_2P_and_L	给定线段 AB，C 点为 AB 方向上的点，已知 AC 长度为 *L*，计算 C 点坐标	*Tool_Point_Giv_2P_and_L*(*A*, *B*, *L*, *Out_C*, *Statu*, *L_BC*)	
		A(*2*)	AB 端点 A 的坐标；浮点数；输入变量
		B(*2*)	AB 端点 A 的坐标；浮点数；输入变量
		L	给定的 AC 距离；浮点数；输入变量
		Out_C(*2*)	C 点坐标；浮点数；输出变量
		Statu	返回 C 点相对于 AB 的位置；整数；输出变量
			Statu = -1： *----------*----------* A C B
			Statu = 1 *----------*----------* A B C
			Statu = 0，B 点和 C 点重合 *----------* A B(C)
		L_BC	线段 BC 的长度；浮点数；输出变量
Tool_Point_Given_Point_Normal_and_Distance_3D	给定原空间点坐标、法线方向，以及两点之间的距离，计算新的空间点坐标	*Tool_Point_Given_Point_Normal_and_Distance_3D*(*Point*, *Normal_Vector*, *Distance*, *New_Point*)	
		Point(*3*)	原空间点坐标；浮点数；输入变量
		Normal_Vector(*3*)	从原点到新点的法向量；浮点数；输入变量
		Distance	两点之间的距离；浮点数；输入变量
		New_Point(*3*)	新空间点坐标；浮点数；输入变量
Tool_Principal_Stresses_2D	计算主应力	*Tool_Principal_Stresses_2D*(*S_xx*, *S_yy*, *S_xy*, *S_1*, *S_3*, *theta_stress*)	
		S_xx	*x* 方向应力；浮点数；输入变量
		S_yy	*y* 方向应力；浮点数；输入变量

		S_xy	*xy* 剪应力；浮点数；输入变量
		S_1	最大主应力；浮点数；输出变量
		S_3	最小主应力；浮点数；输出变量
		theta_stress	最大主应力的方向；浮点数；输出变量
Tool_Shorten_or_Extend_Line	对线段 AB 的端点进行调整（偏置），得到新的端点：根据已有线段 AB 和端点（‘A’或‘B’），计算偏置距离△*L* 后的新的端点 A 或 B	*Tool_Shorten_or_Extend_Line*(*Line_AB*, *delta_L*, *Point_String*, *new_Line_AB*, *new_Point*)	
		Line_AB(*2*,*2*)	线段 AB 坐标（*x_A* = *Line_AB*(*1*,*1*)；*y_A* = *Line_AB*(*1*,*2*)；*x_B* = *Line_AB*(*2*,*1*)；*y_B* = *Line_AB*(*2*,*2*)）；浮点数；输入变量
		delta_L	偏置距离，若 *delta_L* > 0 则增加 AB 长度，若 *delta_L* < 0 则减小 AB 长度；浮点数；输入变量
			例如： A B ●----------● (1) *Point_String* ='A'，*delta_L* < 0： |<---*delta_L*--->| ●----------● A_new B (2) *Point_String* ='B'，*delta_L* > 0： |<---*delta_L*--->| ●----------● A B_new
		Point_String	端点标记，‘A’或‘B’；字符串（长度为 1 字符）；输入变量
		new_Line_AB(*2*,*2*)	新的线段 AB 坐标；浮点数；输出变量
		new_Point(*2*)	新的端点坐标；浮点数；输出变量
Tool_Shorten_or_Extend_Line_3D	对三维空间中的线段 AB 的端点进行调整（偏置），得到新的端点：根据已有线段 AB 和端点（‘A’或‘B’），计算偏置距离△*L* 后的新的端点 A 或 B	*Tool_Shorten_or_Extend_Line_3D*(*Line_AB*, *delta_L*, *Point_String*, *new_Line_AB*, *new_Point*)	
		Line_AB(*2*,*3*)	线段 AB 坐标（*x_A* = *Line_AB*(*1*,*1*)；*y_A* = *Line_AB*(*1*,*2*)；*z_A* = *Line_AB*(*1*,*3*)；*x_B* = *Line_AB*(*2*,*1*)；*y_B* = *Line_AB*(*2*,*2*)；*z_B* = *Line_AB*(*2*,*3*)）；浮点数；输入变量
		delta_L	偏置距离，若 *delta_L* > 0 则增加 AB 长度，若 *delta_L* < 0 则减小 AB 长度；浮点数；输入变量
			例如： A B ●----------● (1) *Point_String* ='A'，*delta_L* < 0： |<---*delta_L*--->| ●----------● A_new B

			(2) *Point_String* ='B'，*delta_L* > 0： |<---*delta_L*--->| ●----------● A B_new
		Point_String	端点标记，‘A’或‘B’；字符串（长度为 1 字符）；输入变量
		new_Line_AB(2,2)	新的线段 AB 坐标；浮点数；输出变量
		new_Point(3)	新的线段 AB 坐标；浮点数；输出变量
Tool_ThetaX_ThetaY_ThetaZ_3D_rotation	计算总体坐标系到裂缝面坐标系的转换矩阵（9.5.2 小节）	*Tool_ThetaX_ThetaY_ThetaZ_3D_rotation*(*c_line*, *Vector_Origi_1*, *Vector_Origi_2*, *Vector_Origi_3*, *Vector_Crack_1*, *Vector_Crack_2*, *Vector_Crack_3*, *ThetaX*, *ThetaY*, *ThetaZ*, *T_Matrix*)	
		c_line	当前线号；整数；输入变量
		Vector_Origi_1(*3*)	总体坐标系下裂缝面坐标系的 *x* 坐标轴向量；浮点数；输入变量
		Vector_Origi_2(*3*)	总体坐标系下裂缝面坐标系的 *y* 坐标轴向量；浮点数；输入变量
		Vector_Origi_3(*3*)	总体坐标系下裂缝面坐标系的 *z* 坐标轴向量；浮点数；输入变量
		Vector_Crack_1(*3*)	裂缝面坐标系下 *x* 坐标轴向量（$[1,0,0]^T$）；浮点数；输入变量
		Vector_Crack_2(*3*)	裂缝面坐标系下 *y* 坐标轴向量（$[0,1,0]^T$）；浮点数；输入变量
		Vector_Crack_3(*3*)	裂缝面坐标系下 *z* 坐标轴向量（$[0,0,1]^T$）；浮点数；输入变量
		ThetaX	旋转角 θ_x；浮点数；输出变量
		ThetaY	旋转角 θ_y；浮点数；输出变量
		ThetaZ	旋转角 θ_z；浮点数；输出变量
		T_Matrix(*3*,*3*)	转换菊展；浮点数；输出变量
Tool_Two_Points_Inters_x_Axis	计算 AB 两点构成的直线与 *x* 轴的交点坐标	*Tool_Two_Points_Inters_x_Axis*(*A*, *B*, *x_0*)	
		A(*2*)	AB 端点 A 的坐标；浮点数；输入变量
		B(*2*)	AB 端点 B 的坐标；浮点数；输入变量
		x_0	AB 两点构成的直线与 *x* 轴的交点的 *x* 坐标；浮点数；输出变量
Tool_Check_Oscillation_by_6_Variables	检查 6 个数据是否以重复的形式出现，如 A、B、A、B、A、B 或 A、B、C、A、B、C，该子程序用于检测收敛过程是否发生了震荡	*Tool_Check_Oscillation_by_6_Variables*(*Input_Var*, *Yes_Oscill*)	
		Input_Var(*6*)	6 个输入数据；浮点数；输入变量
		Yes_Oscill	是否发生了震荡；逻辑变量；输出变量
Tool_Count_Lin	本函数统计文件中的	*Tool_Count_Lines*(*Full_Name*)	

es	数据总行数	*Full_Name*	文件名（含完整路径）；字符串（长度为 200 字符）；输入变量
		Tool_Count_Lines	数据总行数；整数；函数返回值
Tool_Function_2Point_Dis	本函数计算 A、B 两点之间的距离	*Tool_Function_2Point_Dis*(*A*,*B*)	
		A(2)	点 A 坐标；浮点数；输入变量
		B(2)	点 B 坐标；浮点数；输入变量
		Tool_Function_2Point_Dis	两点之间的距离；浮点数；函数返回值
Tool_Function_BETA	β 函数 $B(p, q)$	*Tool_Function_BETA*(*P*, *Q*, *BT*)	
		P	β 函数输入变量；浮点数；输入变量
		Q	β 函数输入变量；浮点数；输入变量
		BT	β 函数值；浮点数；输出变量
Tool_Function_GAMMA	Gamma 函数	*Tool_Function_GAMMA*(*X*,*GA*)	
		X	Gamma 函数输入变量；浮点数；输入变量
		GA	Gamma 函数值；浮点数；输出变量
Tool_Function_hyp2f1	高斯超几何函数	*Tool_Function_hyp2f1*(*a*, *b*, *c*, *z, out_sum*)	
		a	超几何函数输入变量；浮点数；输入变量
		b	超几何函数输入变量；浮点数；输入变量
		c	超几何函数输入变量；浮点数；输入变量
		z	超几何函数输入变量；浮点数；输入变量
		out_sum	超几何函数输出变量；浮点数；输出变量
Tool_Generate_Circles	随机生成若干个彼此不相交的圆	*Tool_Generate_Circles*(*M_x_min*, *M_x_max*, *M_y_min*, *M_y_max*, *Ave_R*, *Delta_R*, *num_Circle*, *check_R*, *Max_Dis_to_Edge*, *c_Center_Cir*, *c_R_Cir*)	
		M_x_min	模型坐标范围，*x* 方向最小坐标值；浮点数；输入变量
		M_x_max	模型坐标范围，*x* 方向最大坐标值；浮点数；输入变量
		M_y_min	模型坐标范围，*y* 方向最小坐标值；浮点数；输入变量
		M_y_max	模型坐标范围，*y* 方向最大坐标值；浮点数；输入变量
		Ave_R	圆的平均半径；浮点数；输入变量
		Delta_R	圆的半径的变化范围（即随机生成的可能的半径为：[*Ave_R* - *Delta_R*, *Ave_R* + *Delta_R*]）；浮点数；输入变量
		num_Circle	要生成的圆的数目；整数；输入变量
		check_R	检测半径，确保以 *check_R* 为半径的圆内无其他圆心；浮点数；输入变量
		Max_Dis_to_Edge	允许的离模型边界最近距离，即防止生成的圆离模型边界太近；浮点数；输入变量
		c_Center_Cir(	生成的各圆的圆心坐标；浮点数；输出变量

		num_Circle,*2*)	
		c_R_Cir(*num_Circle*)	生成的各个圆的半径；浮点数；输出变量
Tool_Generate_Inscribed_Poly	随机生成多边形：先随机生成若干个彼此不相交的圆，再生成圆的内接多边形	*Tool_Generate_Inscribed_Poly*(*M_x_min*, *M_x_max*, *M_y_min*, *M_y_max*, *Ave_R*, *Delta_R*, *num_Rand_Poly*, *num_Vert_Poly*, *check_R*, *Max_Dis_to_Edge*, *Poly_Coor_x*, *Poly_Coor_y)*	
		M_x_min	模型坐标范围，*x* 方向最小坐标值；浮点数；输入变量
		M_x_max	模型坐标范围，*x* 方向最大坐标值；浮点数；输入变量
		M_y_min	模型坐标范围，*y* 方向最小坐标值；浮点数；输入变量
		M_y_max,	模型坐标范围，*y* 方向最大坐标值；浮点数；输入变量
		Ave_R	圆的平均半径；浮点数；输入变量
		Delta_R	圆的半径的变化范围（即随机生成的可能的半径为：[*Ave_R* - *Delta_R*, *Ave_R* + *Delta_R*]）；浮点数；输入变量
		num_Rand_Poly	要生成的多边形的数目；整数；输入变量
		num_Vert_Poly	多边形的边数；整数；输入变量
		check_R	检测半径，确保以 *check_R* 为半径的圆内无其他圆心；浮点数；输入变量
		Max_Dis_to_Edge	允许的离模型边界的最近距离，即防止生成的多边形离模型边界太近；浮点数；输入变量
		Poly_Coor_x(*num_Rand_Poly*,*num_Vert_Poly*)	生成的各多边形顶点的 *x* 坐标；浮点数；输出变量
		Poly_Coor_y(*num_Rand_Poly*,*num_Vert_Poly*)	生成的各多边形顶点的 *y* 坐标；浮点数；输出变量
Tool_Generate_Natural_Fractures	随机生成天然裂缝，算法是：逐个生成，后面生成的裂缝中点不应落在其他裂缝中点所在半径为 *R* 的圆内	*Tool_Generate_Natural_Fractures*(*M_x_min*, *M_x_max*, *M_y_min*, *M_y_max*, *Length_Cr*, *Orien_Cr*, *Length_Cr_Delta*, *Orien_Cr_Delta*, *num_Na_Cr*, *R*, *Na_Cr*)	
		M_x_min	模型坐标范围，*x* 方向最小坐标值；浮点数；输入变量
		M_x_max	模型坐标范围，*x* 方向最大坐标值；浮点数；输入变量
		M_y_min	模型坐标范围，*y* 方向最小坐标值；浮点数；

			输入变量
		M_y_max	模型坐标范围，*y* 方向最大坐标值；浮点数；输入变量
		Length_Cr	天然裂缝平均长度；浮点数；输入变量
		Orien_Cr	天然裂缝平均倾角（单位是度）；浮点数；输入变量
		Length_Cr_Delta	裂缝长度变化范围（即随机生成的可能的长度为：[*Length_Cr* - *Length_Cr_Delta*, *Length_Cr* + *Length_Cr_Delta*]）；浮点数；输入变量
		Orien_Cr_Delta	裂缝倾角变化范围（即随机生成的可能的长度为：[*Length_Cr* - *Length_Cr_Delta*, *Length_Cr* + *Length_Cr_Delta*]，单位是度）；浮点数；输入变量
		num_Na_Cr	要生成的天然裂缝的数目；整数；输入变量
		R	检测半径，即裂缝中点以 *R* 为半径的圆内无其他裂缝中点，该参数可控制所生成天然裂缝的稀疏程度；浮点数；输入变量
		Na_Cr(*num_Na_Cr*,2,2)	生成的各个天然裂缝的坐标；浮点数；输出变量
Tool_Get_Current_Time	获得当前时间	*Tool_Get_Current_Time*(*c_data*, *date_time*, *millisecond*)	
		c_data	当前日期；字符串（长度为 10 字符）；输出变量
		date_time(*8*)	当前时间对应的整数数组（例如，date_time(*5*)对应当前小时数、date_time(*6*)对应当前分钟数、date_time(*7*)对应当前秒数等）；整数；输出变量
		millisecond	当前时间转换成对应的毫秒数；整数；输出变量
Tool_Get_num_Data_from_a_String	统计由逗号隔开的字符串中数据的个数	*Tool_Get_num_Data_from_a_String*(*String*, *num_Char*, *num_Data*, *Yes_Even*)	
		String(*num_Char*)	输入的字符串；字符串（长度为 *num_Char* 字符）；输入变量
		num_Char	字符串长度；整数；输入变量
		num_Data	数据个数；整数；输出变量
		Yes_Even	数据个数是否为偶数；逻辑变量；输出变量
Tool_Get_Value_from_x_y_Curve	给定 *x* 值，从 *y*-*x* 曲线（由一系列数据点组成）上按照线性插值得到对应的 *y* 值	*Tool_Get_Value_from_x_y_Curve*(*Curve_x*, *Curve_y*, *num_Point*, *x*, *y*)	
		Curve_x(*num_Point*)	数据点 *x* 值；浮点数；输入变量
		Curve_y(*num_Point*)	数据点 *y* 值；浮点数；输入变量
		num_Point	数据点个数；整数；输入变量
		x	指定的 *x* 值；浮点数；输入变量

		y	待确定的 *y* 值；浮点数；输出变量
Tool_HF_Pres_Inv_Transform	将水力压裂流固耦合计算得到的各裂缝各自的流体节点水压，转换成全部流体节点水压	*Tool_HF_Pres_Inv_Transform*(*iter*, *temp_Pres*, *CalP_Pres*)	
		iter	迭代步号；整数；输入变量
		temp_Pres(*Max_Num_Cr*, *Max_Num_Cr_CalP*)	每条裂缝各自的流体节点水压（*Max_Num_Cr* 为最大裂缝数目，全局变量；*Max_Num_Cr_CalP* 为每条裂缝允许的最大流体节点数目，全局变量）；浮点数；输入变量
		CalP_Pres(*num_Tol_CalP_Water*)	整体流体节点压力数组（*num_Tol_CalP_Water* 为全部流体节点数目，全局变量）；浮点数；输出变量
Tool_HF_Pres_Transform	将水力压裂流固耦合计算得到的全部流体节点的水压转换成各条裂缝各自的流体节点水压，以便后处理过程中使用	*Tool_HF_Pres_Transform*(*iter*, *CalP_Pres*, *temp_Pres*)	
		iter	迭代步号；整数；输入变量
		CalP_Pres(*num_Tol_CalP_Water*)	整体流体节点压力数组（*num_Tol_CalP_Water* 为全部流体节点数目，全局变量）；浮点数；输入变量
		temp_Pres(*Max_Num_Cr*, *Max_Num_Cr_CalP*)	每条裂缝各自的流体节点水压（*Max_Num_Cr* 为最大裂缝数目，全局变量；*Max_Num_Cr_CalP* 为每条裂缝允许的最大流体节点数目，全局变量）；浮点数；输出变量
Tool_Interpolation_Linear_find_x	已知 A、B 两点，给定通过 AB 两点的直线方程的 *y* 值，计算对应的 *x* 值	*Tool_Interpolation_Linear_find_x*(*Point_A*, *Point_B*, *x*, *y*)	
		Point_A(*2*)	A 点的坐标；浮点数；输入变量
		Point_B(*2*)	B 点的坐标；浮点数；输入变量
		x	待确定的 *x* 值；浮点数；输出变量
		y	给定的 *y* 值；浮点数；输入变量
Tool_Interpolation_Linear_find_y	已知 A、B 两点，给定通过 AB 两点的直线方程的 *x* 值，计算对应的 *y* 值	*Tool_Interpolation_Linear_find_y*(*Point_A*, *Point_B*, *x*, *y*)	
		Point_A(*2*)	A 点的坐标；浮点数；输入变量
		Point_B(*2*)	B 点的坐标；浮点数；输入变量
		x	给定的 *x* 值；浮点数；输入变量
		y	待确定的 *y* 值；浮点数；输出变量
Tool_Intersection_Line_and_Circle	计算线段 AB 和圆的交点坐标	*Tool_Intersection_Line_and_Circle*(*Circle_x0*, *Circle_y0*, *R*, *Seg_A*, *Seg_B*, *num_Inter*, *State*, *Inter*)	
		Circle_x0	圆心 *x* 坐标；浮点数；输入变量
		Circle_y0	圆心 *y* 坐标；浮点数；输入变量
		R	圆的半径；浮点数；输入变量
		Seg_A(*2*)	线段 AB 的 A 点坐标；浮点数；输入变量
		Seg_B(*2*)	线段 AB 的 B 点坐标；浮点数；输入变量
		num_Inter	交点数目；整数；输出变量
		State	线段 AB 相对于圆的位置；整数；输出变量
			State = -1：不相交（AB 两点均在在圆内） *State* = 0：不相交（AB 两点均在在圆外） *State* = 1：相切 *State* = 2：相交
		Inter(*2*,*2*)	交叉点坐标（最多 2 个）；浮点数；输出变量

Tool_Rank_LineSeg_Points	将无序的组成多段线的点从头至尾重新排序，使其首尾相接（排序算法：根据各点至端点 1 的距离进行排序）	*Tool_Rank_LineSeg_Points*(*Line_Segs*, *num_Points*)	
		Line_Segs(*num_Points*,*2*)	多段线坐标（端点和连接点）；浮点数；输入输出变量
		num_Points	多段线坐标点（端点和连接点）数目；整数；输出变量
Tool_Read_File	从特定格式的文件中读取数据	*Tool_Read_File*(*Full_Name*, *Case_Type*, *m*, *n*, *Temp_DATA*, *Flag_Blank*)	
		Full_Name	文件名（包含完整路径）；字符串（长度为 200 字符）；输入变量
		Case_Type	文件类型标识（文件名后缀，包括'*node*'、'*elem*'、'*boux*'、'*bouy*'、'*focx*'以及'*focy*'等）；字符串（长度为 4 字符）；输入变量
		m	待读取的数据第 1 维的长度；整数；输入变量
		n	待读取的数据第 2 维的长度；整数；输入变量
		Temp_DATA(*m*, *n*)	读取到的数据；浮点数；输出变量
		Flag_Blank	文件是否为空；逻辑变量；输出变量
Tool_Save_Files_Dou	保存数据（浮点数）到文件	*Tool_Save_Files_Dou*(*isub*, *file_type*, *DATA*, *m*, *n*)	
		isub	计算步数；整数；输入变量
		file_type	文件类型（文件名后缀）；字符串（长度为 4 字符）；输入变量
		DATA(*m*,*n*)	待保存的数据；浮点数；输入变量
		m	待保存数据第 1 维的长度；整数；输入变量
		n	待保存数据第 2 维的长度；整数；输入变量
Tool_Save_Files_Int	保存数据（整数）到文件	*Tool_Save_Files_Int*(*isub*, *file_type*, *DATA*, *m*, *n*)	
		isub	计算步数；整数；输入变量
		file_type	文件类型（文件名后缀）；字符串（长度为 4 字符）；输入变量
		DATA(*m*,*n*)	待保存的数据；整数；输入变量
		m	待保存数据第 1 维的长度；整数；输入变量
		n	待保存数据第 2 维的长度；整数；输入变量
Tool_Signed_Distance_Point_to_Circle	计算点到圆的符号距离：圆内为负，圆外为正，圆上为 0	*Tool_Signed_Distance_Point_to_Circle*(*x0*, *y0*, *r*, *Point_C*, *S_Distance*)	
		x0	圆心 *x* 坐标；浮点数；输入变量
		y0	圆心 *y* 坐标；浮点数；输入变量
		r	圆的半径；浮点数；输入变量
		Point_C(*2*)	点的坐标；浮点数；输入变量
		S_Distance	符号距离；浮点数；输出变量
Tool_Sort_by_End_to_End	对多段线坐标点进行排序同时删除重复的坐标点，使其首尾相接；与子程序	*Tool_Sort_by_End_to_End*(*m*, *m_OP*, *Input_Outline*, *Output_Outline*, *cou*)	
		m	多段线坐标点数目；整数；输入变量
		m_OP	执行排序操作的多段线坐标点数目，即对前

	Tool_Rank_LineSeg_Points 有相似之处，但本子程序功能更强		*m_OP* 个坐标点进行排序操作；整数；输入变量
		Input_Outline(*m*,*2*)	输入的多段线坐标点；浮点数；输入变量
		Output_Outline(*m*,*2*)	排序后的多段线坐标点；浮点数；输出变量
		cou	删除重复点后的坐标点数；整数；输出变量
Tool_Volume_Hexahedron	计算六面体（ABCD-EFGH）的体积	*Tool_Volume_Hexahedron*(*A*, *B*, *C*, *D*, *E*, *F*, *G*, *H*, *Volume*)	
		A(*3*)	六面体坐标点 A；浮点数；输入变量
		B(*3*)	六面体坐标点 B；浮点数；输入变量
		C(*3*)	六面体坐标点 C；浮点数；输入变量
		D(*3*)	六面体坐标点 D；浮点数；输入变量
		E(*3*)	六面体坐标点 E；浮点数；输入变量
		F(*3*)	六面体坐标点 F；浮点数；输入变量
		G(*3*)	六面体坐标点 G；浮点数；输入变量
		H(*3*)	六面体坐标点 H；浮点数；输入变量
		Volume	六面体体积；浮点数；输出变量
Tool_Volume_Pyramid	计算四棱锥的体积	*Tool_Volume_Pyramid*(*A*, *B*, *C*, *D*, *E*, *Volume*)	
		A(*3*)	四棱锥坐标点 A；浮点数；输入变量
		B(*3*)	四棱锥坐标点 B；浮点数；输入变量
		C(*3*)	四棱锥坐标点 C；浮点数；输入变量
		D(*3*)	四棱锥坐标点 D；浮点数；输入变量
		E(*3*)	四棱锥坐标点 E；浮点数；输入变量
		Volume	四棱锥的体积；浮点数；输出变量
Tool_Yes_Crack_Polygon_Intersects	判断指定裂缝是否跟多边形相交	*Tool_Yes_Crack_Polygon_Intersects*(*c_Crack*, *Clo_Supp_Domain*, *num_Domain_P*, *Yes_Intersect*)	
		c_Crack	裂缝号；整数；输入变量
		Polygon(*num_Point*,*2*)	多边形坐标点；浮点数；输入变量
		num_Point	多边形坐标点的数目；整数；输入变量
		Yes_Intersect	裂缝是否跟多边形相交；逻辑变量；输出变量
Tool_Yes_Cricle_Polygon_Intersects	判断圆是否跟多边形相交（假如有一条边的两个端点中有一个在圆内，一个在圆外，则判断为圆与多边形相交）	*Tool_Yes_Cricle_Polygon_Intersects*(*x0*, *y0*, *r*, *Polygon*, *num_Point*, *Yes_Intersect*)	
		x0	圆心 *x* 坐标；浮点数；输入变量
		y0	圆心 *y* 坐标；浮点数；输入变量
		r	圆的半径；浮点数；输入变量
		Polygon(*num_Point*,*2*)	多边形坐标点；浮点数；输入变量
		num_Point	多边形坐标点的数目；整数；输入变量
		Yes_Intersect	是否跟多边形相交；逻辑变量；输出变量
Tool_Yes_In_Poly	判断一点是否在多边形内（闭合多边形，即 *npol* = 边数 + 1）	*Tool_Yes_In_Poly*(*x*, *y*, *xpol*, *ypol*, *npol*, *Yes_INOUT*)	
		x	点的 *x* 坐标；浮点数；输入变量
		y	点的 *y* 坐标；浮点数；输入变量

		xpol(*npol*)	多边形的 *x* 坐标；浮点数；输入变量
		ypol(*npol*)	多边形的 *y* 坐标；浮点数；输入变量
		npol	*npol* = 边数 + 1；整数；输入变量
		Yes_INOUT	是否在多边形内；逻辑变量；输出变量
Tool_Yes_On_Line	判断一点是否在线段 AB 上	*Tool_Yes_On_Line*(*x, y, A, B, Yes_ON*)	
		x	点的 *x* 坐标；浮点数；输入变量
		y	点的 *y* 坐标；浮点数；输入变量
		A	线段的 A 点坐标；浮点数；输入变量
		B	线段的 B 点坐标；浮点数；输入变量
		Yes_ON	是否在线段 AB 上；逻辑变量；输出变量
Tool_Yes_On_Line_Endpoint	判断一点是否在线段 AB 上的端点上	*Tool_Yes_On_Line_Endpoint*(*x*, *y*, *A*, *B*, *Yes_ON*)	
		x	点的 *x* 坐标；浮点数；输入变量
		y	点的 *y* 坐标；浮点数；输入变量
		A	线段的 A 点坐标；浮点数；输入变量
		B	线段的 B 点坐标；浮点数；输入变量
		Yes_ON	是否在线段 AB 上的端点上；逻辑变量；输出变量
Tool_Yes_On_Poly	判断某点是否在多边形（闭合多边形，即 *npol* = 边数 + 1）边界上	*Tool_Yes_On_Poly*(*x*, *y*, *xpol*, *ypol*, *npol*, *Yes_ONOUT*)	
		x	点的 *x* 坐标；浮点数；输入变量
		y	点的 *y* 坐标；浮点数；输入变量
		xpol(*npol*)	多边形的 *x* 坐标；浮点数；输入变量
		ypol(*npol*)	多边形的 *y* 坐标；浮点数；输入变量
		npol	*npol* = 边数 + 1；整数；输入变量
		Yes_ONOUT	是否在多边形边界上；逻辑变量；输出变量
Tool_Yes_PER_in_3D_Tri	判断空间一点到空间三角形平面的垂足是否在该空间三角形内部	*Tool_Yes_PER_in_3D_Tri*(*PER*, *Tri_A*, *Tri_B*, *Tri_*C, *Yes_PER_in*)	
		PER(*3*)	垂足点的坐标；浮点数；输入变量
		Tri_A(*3*)	空间三角形 A 点坐标；浮点数；输入变量
		Tri_B(*3*)	空间三角形 B 点坐标；浮点数；输入变量
		Tri_C(*3*)	空间三角形 C 点坐标；浮点数；输入变量
		Yes_PER_in	垂足是否在该空间三角形内部；逻辑变量；输出变量
Tool_Yes_Point_in_3D_Hexahedron	判断空间一点是否在空间六面体（ABCD-EFGH）内，拆分成 3 个空间四面体来判断，即 H-BCGF、H-EABF、H-ABCD	*Tool_Yes_Point_in_3D_Hexahedron*(*Point*, *A*, *B*, *C*, *D*, *E*, *F*, *G*, *H*, *Yes_in*, *Yes_on*)	
		Point(*3*)	空间点的坐标；浮点数；输入变量
		A(*3*)	六面体坐标点；浮点数；输入变量
		B(*3*)	六面体坐标点；浮点数；输入变量
		C(*3*)	六面体坐标点；浮点数；输入变量
		D(*3*)	六面体坐标点；浮点数；输入变量
		E(*3*)	六面体坐标点；浮点数；输入变量
		F(*3*)	六面体坐标点；浮点数；输入变量
		G(*3*)	六面体坐标点；浮点数；输入变量
		H(*3*)	六面体坐标点；浮点数；输入变量
		Yes_in	是否在空间六面体内；逻辑变量；输出变量

		Yes_on	是否在空间六面体边界上；逻辑变量；输出变量
Tool_Yes_Point_in_3D_Pyramid	判断空间一点是否在空间四棱锥（A-BCDE）里面，拆分成两个空间四面体来判断，即 A-BCE、A-CDE	*Tool_Yes_Point_in_3D_Pyramid*(*Point*, *A*, *B*, *C*, *D*, *E*, *Yes_in*, *Yes_on*)	
		Point(3)	空间点的坐标；浮点数；输入变量
		A(3)	四棱锥坐标点；浮点数；输入变量
		B(3)	四棱锥坐标点；浮点数；输入变量
		C(3)	四棱锥坐标点；浮点数；输入变量
		D(3)	四棱锥坐标点；浮点数；输入变量
		E(3)	四棱锥坐标点；浮点数；输入变量
		Yes_in	是否在空间四棱锥内；逻辑变量；输出变量
		Yes_on	是否在空间四棱锥边界上；逻辑变量；输出变量
Tool_Yes_Point_in_3D_Tetrahedron	判断空间一点是否在空间四面体（A-BCD）内	*Tool_Yes_Point_in_3D_Tetrahedron*(*Point*, *A*, *B*, *C*, *D*, *Yes_in*, *Yes_on*)	
		Point(3)	空间点的坐标；浮点数；输入变量
		A(3)	四面体坐标点；浮点数；输入变量
		B(3)	四面体坐标点；浮点数；输入变量
		C(3)	四面体坐标点；浮点数；输入变量
		D(3)	四面体坐标点；浮点数；输入变量
		Yes_in	是否在空间四面体内；逻辑变量；输出变量
		Yes_on	是否在空间四面体边界上；逻辑变量；输出变量
Tool_Yes_Point_in_Inclusions	判断某点是否在某一夹杂内部	*Tool_Yes_Point_in_Inclusions*(*x*, *y*, *Yes_in_Incl*, *c_Incl_Num*)	
		x	点的 *x* 坐标；浮点数；输入变量
		y	点的 *y* 坐标；浮点数；输入变量
		Yes_in_Incl	是否在夹杂内部；逻辑变量；输出变量
		c_Incl_Num	对应的夹杂号；整数；输出变量
Tool_Yes_Two_Circles_Intersects	判断两个圆是否有交集	*Tool_Yes_Two_Circles_Intersects*(*x0*, *y0*, *R0*, *x1*, *y1*, *R1*, *Yes_Intersect*)	
		x0	第 1 个圆的圆心 *x* 坐标；浮点数；输入变量
		y0	第 1 个圆的圆心 *y* 坐标；浮点数；输入变量
		R0	第 1 个圆的半径；浮点数；输入变量
		x1	第 2 个圆的圆心 *x* 坐标；浮点数；输入变量
		y1	第 2 个圆的圆心 *y* 坐标；浮点数；输入变量
		R1	第 2 个圆的半径；浮点数；输入变量
		Yes_Intersect	是否相交；逻辑变量；输出变量
Tool_Generate_Random_Number	生成随机数（0-1 之间的数），见 3.8.3 小节	*Tool_Generate_Random_Number*(*out_number*)	
		out_number	生成随机数；浮点数；输出变量

5 矩阵和向量操作相关的子程序

科学计算编程中涉及的数据大多数是矩阵和向量，作者在编写 PhiPsi 程序时，编写了若干矩阵和向量操作相关的子程序，以方便其他子程序的调用。此外，还编写了子程序 *Matrix_Solve_LSOE*，专门用于线性方程组的求解。表 5 汇总了这些子程序。

表 5 PhiPsi 程序矩阵和向量操作相关子程序用途及输入输出变量说明（按名称排序）

子程序名	子程序功能	子程序输入输出接口	
Init_Random_Seed	根据系统时间生成随机数种子	*Init_Random_Seed*()	
		无输入输出变量	
LSOE_GAUSS	高斯消去法求解线性方程组 **KU**＝**F**，其中 **K** 是 *n*×*n* 的矩阵	*LSOE_GAUSS*(*Lin*, *Limrow*, *Limcol*, *N*, *U*, *Singul*)	
		Lin(*Limrow*, *Limcol*)	线性方程组系数矩阵，对于 **KU**＝**F** 这一问题，*Lin*(1:*n*,1:*n*) = **K**，*LIN*(1:*n*,n+1) = **F**；浮点数；输入变量
		Limrow	等于 *n*；整数；输入变量
		Limcol	等于 *n*+1；整数；输入变量
		n	**K** 的阶数；整数；输入变量
		U(*n*)	待求的 **U** 向量；浮点数；输出变量
		Singul	是否奇异；逻辑变量；输出变量
Matrix_Count_Row_Int	统计矩阵各行出现的次数（比如，假如第 1 行和第 3 行以及第 9 行相同，则第 1 行出现 3 次），并存入 *Count_Row*(*m*)	*Matrix_Count_Row_Int*(*m*, *n*, *Matrix*, *Count_Row*, *Num_Once*)	
		m	矩阵行数；整数；输入变量
		n	矩阵列数；整数；输入变量
		Matrix(*m*,*n*)	待统计的矩阵；整数；输入变量
		Count_Row(*m*)	各行出现的次数；整数；输出变量
		Num_Once	仅出现一次的矩阵行的行数；整数；输出变量
Matrix_Det	计算方阵的行列式	*Matrix_Det*(*n*, *Matrix*, *det*)	
		n	方阵的阶数；整数；输入变量
		Matrix(*n*,*n*)	待计算的方阵；浮点数；输入变量
		det	方阵的行列式；浮点数；输出变量
Matrix_Diagonalization	方阵对角化：各行元素求和放在对角位置	*Matrix_Diagonalization*(*n*, *Matrix*)	
		n	方阵的阶数；整数；输入变量
		Matrix(*n*,*n*)	方阵；浮点数；输入输出变量
Matrix_Inverse	求方阵 **A** 的逆矩阵，输出到 **C**	*Matrix_Inverse*(*A*, *C*, *n*)	
		A(*n*, *n*)	待计算的方阵；浮点数；输入变量
		C(*n*, *n*)	方阵的逆阵；浮点数；输出变量
		n	方阵的阶数；整数；输入变量
Matrix_Inverse_2x2	求 2×2 方阵 **A** 的逆矩阵	*Matrix_Inverse_2x2*(*Matrix_A*, *Matrix_invA*)	
		Matrix_A(*2*,*2*)	待计算的方阵 **A**；浮点数；输入变量
		Matrix_invA(	方阵 **A** 的逆阵；浮点数；输出变量

		2,*2*)	
Matrix_Inverse_3x3	求 3×3 方阵 **A** 的逆矩阵	*Matrix_Inverse_3x3*(*Matrix_A*, *Matrix_invA*)	
		Matrix_A(*3*,*3*)	待计算的方阵 **A**；浮点数；输入变量
		Matrix_invA(*3*,*3*)	方阵 **A** 的逆阵；浮点数；输出变量
Matrix_Inverse_Lapack	利用 Lapack 库求方阵 **A** 的逆矩阵	*Matrix_Inverse_Lapack*(*a*, *n*)	
		A(*n*, *n*)	待计算的方阵；浮点数；输入输出变量
		n	方阵的阶数；整数；输入变量
Matrix_Norm2	求方阵的二范数（平方和开根号）	*Matrix_Norm2*(*n*, *Matrix*, *Norm_2*)	
		n	方阵的阶数；整数；输入变量
		Matrix(*n*,*n*)	待计算的方阵；浮点数；输入变量
		Norm_2	方阵的二范数；浮点数；输出变量
Matrix_Num_Nonzero_Dou	统计矩阵非零元素个数	*Matrix_Num_Nonzero_Dou*(*m*, *n*, *Matrix*, *Num_Nonzero*)	
		m	矩阵的行数；整数；输入变量
		n	矩阵的列数；整数；输入变量
		Matrix(*m*,*n*)	输入矩阵；浮点数；输入变量
		Num_Nonzero	非零元素的个数；整数；输出变量
Matrix_Solve_LSOE	求解线性方程组（支持多种求解器，见附录 F）**KU** = **F**	*Matrix_Solve_LSOE*(*Key_Indent*, *Key_LSOE_Sys*, *Key_SLOE*, *K*, *F*, *U*, *n*)	
		Key_Indent	屏幕交互输出的空格缩进数目；整数；输入变量
		Key_LSOE_Sys	**K** 是否对称（=1 则 **K** 对称）；整数；输入变量
		Key_SLOE	求解器号（附录 F 及附录 N）；整数；输入变量
		K(*n*,*n*)	线性方程组系数矩阵 **K**；浮点数；输入变量
		F(*n*)	**F** 向量；浮点数；输入变量
		U(*n*)	待求的 **U** 向量；浮点数；输出变量
		n	**K** 的阶数；整数；输入变量
Matrix_Solve_LSOE_Sparse	线性方程组 **KU** = **F**（其中 **K** 以 *CSR* 稀疏矩阵格式存储，详见附录 P）的求解（UMFPACK 求解器）	*Matrix_Solve_LSOE_Sparse*(*Key_Indent*, *Key_LSOE_Sys*, *Key_SLOE*, *K_CSR_NNZ*, *K_CSR_aa*, *K_CSR_ja*, *K_CSR_ia*, *F*, *U*, *n*)	
		Key_Indent	屏幕交互输出的空格缩进数目；整数；输入变量
		Key_LSOE_Sys	**K** 是否对称（=1 则 **K** 对称）；整数；输入变量
		Key_SLOE	求解器号；整数；输入变量
		K_CSR_NNZ	以 *CSR* 稀疏矩阵格式存储的 **K**—非零元素个数；整数；输入变量
		K_CSR_aa(*K_CSR_NNZ*)	以 *CSR* 稀疏矩阵格式存储的 **K**—非零元素；浮点数；输入变量
		K_CSR_ja(*K_CSR_NNZ*)	以 *CSR* 稀疏矩阵格式存储的 **K**—各非零元素对应的列号；整数；输入变量

		K_CSR_ia(*n*+1)	以 *CSR* 稀疏矩阵格式存储的 **K**—**K** 中第 *i* 行的首个非零元素在 *K_CSR_aa* 中的位置，且 *K_CSR_ia*(*n*+1)= *K_CSR_ia*(1) + *K_CSR_NNZ*；整数；输入变量
		F(*n*)	**F** 向量；浮点数；输入变量
		U(*n*)	待求的 **U** 向量；浮点数；输出变量
		n	**K** 的阶数；整数；输入变量
Matrix_Sort_Dou	对矩阵元素（浮点数）进行排序	*Matrix_Sort_Dou*(*m*, *n* ,*Matrix*)	
		m	矩阵行数；整数；输入变量
		n	矩阵列数；整数；输入变量
		Matrix(*m*,*n*)	矩阵；浮点数；输入输出变量
Matrix_Sort_Int	对矩阵元素（整数）进行排序	*Matrix_Sort_Int*(*m*, *n*, *Matrix*)	
		m	矩阵行数；整数；输入变量
		n	矩阵列数；整数；输入变量
		Matrix(*m*,*n*)	矩阵；整数；输入输出变量
Matrix_Sort_Row_Dou	将矩阵的第 *Start_m* 行到*Finish_m*的每行元素（浮点数）按照升序排序	*Matrix_Sort_Row_Dou*(*m*, *n*, *Start_m*, *Finish_m*, *Matrix*)	
		m	矩阵行数；整数；输入变量
		n	矩阵列数；整数；输入变量
		Start_m	开始处理的行数；整数；输入变量
		Finish_m	结束处理的行数；整数；输入变量
		Matrix(*m*,*n*)	矩阵；浮点数；输入输出变量
Matrix_Sort_Row_Int	将矩阵的第 *Start_m* 行到*Finish_m*的每行元素（整数）按照升序排序	*Matrix_Sort_Row_Int(m*, *n*, *Start_m*, *Finish_m*, *Matrix)*	
		m	矩阵行数；整数；输入变量
		n	矩阵列数；整数；输入变量
		Start_m	开始处理的行数；整数；输入变量
		Finish_m	结束处理的行数；整数；输入变量
		Matrix(*m*,*n*)	输入矩阵；整数；输入输出变量
Matrix_Symm_Is_Dou	检查矩阵是否是对称矩阵，矩阵元素为浮点数	*Matrix_Symm_Is_Dou*(*n*, *Matrix*, *Yes_Symm*)	
		n	方阵的阶数；整数；输入变量
		Matrix(*n*,*n*)	待检查的矩阵；浮点数；输入变量
		Yes_Symm	矩阵是否是对称矩阵；逻辑变量；输出变量
Matrix_Unique_Row_Dou	删除矩阵（矩阵元素为浮点数）重复的行，操作范围：第 1 行到第 *m_Finish* 行	*Matrix_Unique_Row_Dou*(*m*, *n*, *m_Finish*, *Matrix*, *Uniqued_Matrix*, *Uniqued_m*, *Uni_Mat_Count*)	
		m	矩阵行数；整数；输入变量
		n	矩阵列数；整数；输入变量
		m_Finish	结束处理的行数；整数；输入变量
		Matrix(*m*,*n*)	输入矩阵；浮点数；输入变量
		Uniqued_Matrix(*m*,*n*)	删除重复行后的输出矩阵；浮点数；输出变量
		Uniqued_m	新矩阵的行数；整数；输入变量
		Uni_Mat_Count(*m*)	新矩阵每行重复的次数；整数；输出变量
Matrix_Unique	删除 *m* 行 2 列的矩阵	*Matrix_Unique_Row_Dou_m_x_2_Tol*(*m*, *Matrix*, *Tol*,	

_Row_Dou_m_x_2_Tol	重复的行，操作范围：第 1 行到最后 1 行	*Uniqued_Matrix*, *Uniqued_m*, *Uni_Mat_Count*)	
		m	矩阵的行数；整数；输入变量
		Matrix(*m*,*2*)	矩阵；浮点数；输入变量
		Tol	容差；浮点数；输入变量
		Uniqued_Matrix(*m*,*2*)	删除重复行后的输出矩阵；浮点数；输出变量
		Uniqued_m	新矩阵的行数；整数；输入变量
		Uni_Mat_Count	新矩阵每行重复的次数；整数；输出变量
Matrix_Unique_Row_Int	删除矩阵（矩阵元素为整数）重复的行，操作范围：第 1 行到第 *m_Finish* 行	*Matrix_Unique_Row_Int*(*m*, *n*, *m_Finish*, *Matrix*, *Uniqued_Matrix*, *Uniqued_m*, *Uni_Mat_Count*)	
		m	矩阵行数；整数；输入变量
		n	矩阵列数；整数；输入变量
		m_Finish	结束处理的行数；整数；输入变量
		Matrix(*m*,*n*)	输入矩阵；整数；输入变量
		Uniqued_Matrix(*m*,*n*)	删除重复行后的输出矩阵；整数；输出变量
		Uniqued_m	新矩阵的行数；整数；输入变量
		Uni_Mat_Count	新矩阵每行重复的次数；整数；输出变量
Matrixes_Equal_Is_Dou	对比两个矩阵（矩阵元素为浮点数）是否相同	*Matrixes_Equal_Is_Dou*(*Matrix_A*, *Matrix_B*, *m*, *n*, *Yes*)	
		Matrix_A(*m*,*n*)	输入矩阵 **A**；浮点数；输入变量
		Matrix_B(*m*,*n*)	输入矩阵 **B**；浮点数；输入变量
		m	矩阵行数；整数；输入变量
		n	矩阵列数；整数；输入变量
		Yes	矩阵是否相同；逻辑变量；输出变量
Matrixes_Equal_Is_Int	对比两个矩阵（矩阵元素为整数）是否相同	*Matrixes_Equal_Is_Int*(*Matrix_A*, *Matrix_B*, *m*, *n*, *Yes*)	
		Matrix_A	输入矩阵 **A**；整数；输入变量
		Matrix_B	输入矩阵 **B**；整数；输入变量
		m	矩阵行数；整数；输入变量
		n	矩阵列数；整数；输入变量
		Yes	矩阵是否相同；逻辑变量；输出变量
Vector_belongs_Matrix_Is_Dou	检查向量 *Vector* 是否属于矩阵（矩阵元素为浮点数）	*Vector_belongs_Matrix_Is_Dou*(*m*, *n*, *Matrix*, *Vector*, *Location*, *Yes*)	
		m	矩阵行数；整数；输入变量
		n	矩阵列数；整数；输入变量
		Matrix(*m*,*n*)	输入矩阵；浮点数；输入变量
		Vector	输入向量；浮点数；输入变量
		Location	所在行数；整数；输出变量
		Yes	是否属于矩阵；逻辑变量；输出变量
Vector_belongs	检查向量 *Vector* 是否	*Vector_belongs_Matrix_Is_Int*(*m*, *n*, *Matrix*, *Vector*, *Location*, *Yes*)	

_Matrix_Is_Int	属于矩阵（矩阵元素为整数）	*m*	矩阵行数；整数；输入变量
		n	矩阵列数；整数；输入变量
		Matrix(*m*,*n*)	输入矩阵；整数；输入变量
		Vector(*n*)	输入向量；整数；输入变量
		Location	所在行数；整数；输出变量
		Yes	是否属于矩阵；逻辑变量；输出变量
Vector_Cross_Product3	两个向量的叉乘积，向量长度为 3	*Vector_Cross_Product_3*(*a*, *b*, *Out_Vec*)	
		a(*3*)	输入向量；浮点数；输入变量
		b(*3*)	输入向量；浮点数；输入变量
		Out_Vec(*3*)	输出向量；浮点数；输出变量
Vector_Location_Int	寻找整数变量 *Variable* 是否在向量 *Vector* 中并返回所在位置	*Vector_Location_Int*(*n*, *Vector*, *Variable*, *location*, *Yes_In*)	
		n	向量的维度；整数；输入变量
		Vector(*n*)	向量；整数；输入变量
		Variable	变量；整数；输入变量
		location	所在位置；整数；输出变量
		Yes_In	是否在 *Vector* 中；逻辑变量；输出变量
Vector_Norm2	向量的二范数（平方和开根号）	*Vector_Norm2*(*n*, *Vector*, *Norm_2*)	
		n	向量的维度；整数；输入变量
		Vector(*n*)	向量；浮点数；输入变量
		Norm_2	向量的二范数；浮点数；输出变量
Vector_Normalize	向量归一化（正则化）	*Vector_Normalize*(*n*, *Vector*)	
		n	向量的维度；整数；输入变量
		Vector(*n*)	向量；浮点数；输入输出变量
Vector_Sort_Dou	向量（元素为浮点数）元素排序	*Vector_Sort_Dou*(*n*, *Vector*)	
		n	向量的维度；整数；输入变量
		Vector(*n*)	向量；浮点数；输入输出变量
Vector_Sort_Int	向量（元素为整数）元素排序	*Vector_Sort_Int*(*n*, *Vector*)	
		n	向量的维度；整数；输入变量
		Vector	向量；整数；输入输出变量
Vector_Unique_Dou	删除向量（元素为浮点数）重复元素	*Vector_Unique_Dou*(*n*, *n_Op_F*, *Vector*, *Uniqued_Vec*, *Uniqued_n*)	
		n	向量的维度；整数；输入变量
		n_Op_F	操作对象为第 1 到第 *n_Op_F* 个元素；整数；输入变量
		Vector(*n*)	输入向量；浮点数；输入变量
		Uniqued_Vec(*n*)	删除重复元素之后的新向量；浮点数；输入变量
		Uniqued_n	新向量的有效数据的个数；整数；输出变量
Vector_Unique_Int	删除向量（元素为整数）重复元素	*Vector_Unique_Int*(*n*, *n_Op_F*, *Vector*, *Uniqued_Vec*, *Uniqued_n*)	
		n	向量的维度；整数；输入变量
		n_Op_F	操作对象为第 1 到第 *n_Op_F* 个元素；整数；输入变量
		Vector(*n*)	输入向量；整数；输入变量
		Uniqued_Vec(	删除重复元素之后的新向量；整数；输入变量

		n)	
		Uniqued_n	新向量的有效数据的个数；整数；输出变量
Vector_ZeroOut_Neg_value	将向量中的负值元素归零	*Vector_ZeroOut_Neg_value*(*Vector*, *n*)	
		Vector(*n*)	输入向量；浮点数；输入输出变量
		n	向量的维度；整数；输入变量
Vectors_Equal_Is_Dou	对比两个向量 **a** 和 **b**（元素为浮点数）是否相同	*Vectors_Equal_Is_Dou*(*Vector_a*, *Vector_b*, *n*, *Yes*)	
		Vector_a(*n*)	向量 **a**；浮点数；输入变量
		Vector_b(*n*)	向量 **b**；浮点数；输入变量
		n	向量的维度；整数；输入变量
		Yes	两个向量是否相同；逻辑变量；输出变量
Vectors_Equal_Is_Int	对比两个向量 **a** 和 **b**（元素为整数）是否相同	*Vectors_Equal_Is_Int*(*Vector_a*, *Vector_b*, *n*, *Yes*)	
		Vector_a(*n*)	向量 **a**；整数；输入变量
		Vector_b(*n*)	向量 **b**；整数；输入变量
		n	向量的维度；整数；输入变量
		Yes	两个向量是否相同；逻辑变量；输出变量
Vectors_Multi	两个向量相乘得到矩阵，比如（3*1） *（1*4），得到 3*4 的矩阵	*Vectors_Multi*(*Vector1*, *n1*, *Vector2*, *n2*, *Matrix_OUT*)	
		Vector1(*n1*)	向量 1；浮点数；输入变量
		n1	向量 1 的维度；整数；输入变量
		Vector2(*n2*)	向量 2；浮点数；输入变量
		n2	向量 2 的维度；整数；输入变量
		Matrix_OUT(*n1*,*n2*)	计算得到的矩阵；浮点数；输出变量

6 载荷相关子程序

表 6 汇总了 PhiPsi 程序中组集载荷向量以及计算载荷因子的子程序。

表 6 PhiPsi 程序载荷相关子程序用途及输入输出变量说明（按名称排序）

子程序名	子程序功能	子程序输入输出接口	
Force_Factor	计算载荷因子	*Force_Factor*(*Lambda*,*isub*,*Yes_Last_Growth*)	
		Lambda	载荷因子；浮点数；输出变量
		isub	载荷子步号；整数；输入变量
		Yes_Last_Growth	上一步是否发生了裂缝扩展；逻辑型；输入变量
Force_Vector	组集外载荷向量	*Force_Vector*(*c_Total_FD*,*isub*,*Yes_Biot*,*Lambda*,*globalF*)	
		c_Total_FD	自由度数目；整数；输入变量
		isub	载荷子步号；整数；输入变量
		Yes_Biot	是否考虑比奥固结；逻辑型；输入变量
		Lambda	载荷因子；浮点数；输入变量
		globalF(*c_Total_FD*)	外载荷向量；浮点数；输出变量
Force_Vector_3D	组集外载荷向量	*Force_Vector_3D*(*c_Total_FD*,*isub*,*Lambda*,*globalF*)	

	（3D 问题）	*c_Total_FD*	自由度数目；整数；输入变量
		isub	载荷子步号；整数；输入变量
		Lambda	载荷因子；浮点数；输入变量
		globalF(*c_Total_FD*)	外载荷向量；浮点数；输出变量

7 以 *Save* 开头的文件保存相关子程序

表 7 汇总了 PhiPsi 程序中与数据保存相关的子程序。

表 7 PhiPsi 程序以 *Save* 开头的子程序用途及输入输出变量说明（按名称排序）

子程序名	子程序功能	子程序输入输出接口	
Save_Coors_of_Fracture_Zone	保存破裂区坐标文件，以便后处理绘图	无输入输出变量	
Save_Crack_Tan_Relative_Disp	保存 *isub* 计算步裂缝面切向开度；结果文件类型*.*ctap_i*（注：其中的 *i* 为载荷子步号，下同）	*Save_Crack_Tan_Relative_Disp*(*isub*, *Cracks_CalP_Tan_Aper*)	
		isub	载荷子步号；整数；输入变量
		Cracks_CalP_Tan_Aper(*num_Crack*, *Max_Num_Cr_CalP*)	裂缝面切向位移；浮点数；输入变量
Save_Disp	保存 *isub* 计算步节点位移，包括常规自由度和增强自由度；结果文件类型*.*disp_i* 和*.*disn_i*（注：二者数据略有不同，详见程序代码）	*Save_Disp*(*isub*)	
		isub	载荷子步号；整数；输入变量
Save_Dof_Force	保存 *isub* 计算步位移自由度上的载荷，包括作用在增强节点上的水压载荷（用于水力压裂分析）；结果文件类型为*.*fxdf_i* 和*.*fydf_i*	*Save_Dof_Force*(*isub*, *F*, *all_Solid_Freedom*)	
		isub	载荷子步号；整数；输入变量
		F(*all_Solid_Freedom*)	全部位移自由度上的外载荷向量；浮点数；输入变量
		all_Solid_Freedom	全部位移自由度的数目；整数；输入变量
Save_Ele_Contac_State	保存 *isub* 计算步各单元的接触状态；结果文件类型为*.*elcs_i*	*Save_Ele_Contac_State*(*isub*)	
		isub	载荷子步号；整数；输入变量
Save_Ele_State_S	保存 *isub* 计算步单	*Save_Ele_State_Stress_1_3*(*isub*)	

tress_1_3	元应力状态，即是否满足 σ_1-σ_3 > *Tol*；结果文件类型为 **.elss_i*	*isub*	载荷子步号；整数；输入变量
Save_File_F_D_Curve	保存 *isub* 计算步特定节点的载荷-位移曲线；结果文件类型为**.fdcu_i*	*Save_File_F_D_Curve*(*isub*, *c_Lambda*, *c_node_num*)	
		isub	载荷子步号；整数；输入变量
		c_Lambda	载荷因子；浮点数；输入变量
		c_node_num	节点号；整数；输入变量
Save_Files_Cr_CalP	保存 *isub* 步裂缝计算点相关信息，包括计算点坐标（**.apex_i*、**.apey_i*）、计算点所在裂缝片段的倾角（**.cori_i*）、计算点裂缝开度（**.cape_i*）、计算点压强（**.cpre_i*）、流速（**.cvel_i*）、流量（**.cqua_i*）等	*Save_Files_Cr_CalP*(*isub*)	
		isub	载荷子步号；整数；输入变量
Save_Files_Crack	保存 *isub* 计算步裂缝相关文件，具体包括 **.crax_i*、**.cray_i*、**.ennd_i*、**.elty_i*、**.celc_i*、**.crab_i*、**.celt_i*、**.celv _i*、**.celj_i*、**.ctty_i*、**.posi_i*、**.njel_i*、**.njhl_i* 等	*Save_Files_Crack*(*isub*)	
		isub	载荷子步号；整数；输入变量
Save_Files_Crack_3D	保存 *isub* 计算步裂缝相关文件（3D 问题），**.crax_n*、**.cray_n*、**.craz_n*、**.ennd_n*、**.elty_n*、**.posi_n* 等文件	*Save_Files_Crack_3D*(*isub*)	
		isub	载荷子步号；整数；输入变量
Save_Files_Holes	保存 *isub* 计算步孔洞相关文件：**.hlcr*、**.ennh_i*、**.elth_i*、**.posh_i* 等文件	*Save_Files_Holes*(*isub*)	
		isub	载荷子步号；整数；输入变量
Save_Files_Inclusions	保存 *isub* 计算步夹杂相关文件：**.jzcr*、**.ennj_i*、**.eltj_i*、**.posj_i* 等文件	*Save_Files_Inclusions*(*isub*)	
		isub	载荷子步号；整数；输入变量

Save_Gauss_Coors	保存 *isub* 计算步的高斯积分点坐标：结果文件类型为 **.gcor_i*	*Save_Gauss_Coors*(*isub*, *Total_Num_G_P*, *c_G_CoorX*, *c_G_CoorY*)	
		isub	载荷子步号；整数；输入变量
		Total_Num_G_P	总的高斯积分点数目；整数；输入变量
		c_G_CoorX(*Total_Num_G_P*)	各高斯积分点的 *x* 坐标；浮点数；输入变量
		c_G_CoorY(*Total_Num_G_P*)	各高斯积分点的 *y* 坐标；浮点数；输入变量
Save_Gauss_Disps	保存 *isub* 计算步的高斯积分点上的位移；结果文件类型为**.disg_i*	*Save_Gauss_Disps*(*isub*,*Total_Num_G_P*)	
		isub	载荷子步号；整数；输入变量
		Total_Num_G_P	总的高斯积分点数目；整数；输入变量
Save_Gauss_Stress	保存 *isub* 计算步的高斯积分点上的应力；结果文件类型为**.strg_i*	*Save_Gauss_Stress*(*isub*,*Total_Num_G_P*)	
		isub	载荷子步号；整数；输入变量
		Total_Num_G_P	总的高斯积分点数目；整数；输入变量
Save_HF_Injection_Pressure	保存各破裂步注水点的水压和对应的时间；结果文件类型为**.injp_i*	*Save_HF_Injection_Pressure*(*ifra*, *c_Time*)	
		ifra	破裂步号（载荷步号）；整数；输入变量
		c_Time	压裂时间；浮点数；输入变量
Save_HF_time	保存各计算步的时间；结果文件类型为**.hftm*	*Save_HF_time*(*imf*, *ifra*, *Counter_Iter*, *c_Time*)	
		imf	分段压裂的压裂段号（对于非分段水力压裂分析，该参数无意义）；整数；输入变量
		ifra	载荷步号；整数；输入变量
		Counter_Iter	总的迭代步号；整数；输入变量
		c_Time	时间；浮点数；输入变量
Save_Info_for_Matlab_Post	保存关键信息，以便 Matlab 后处理；结果文件类型为 **.post*	无输入输出变量	
Save_SIFs_KI_and_KII	保存 *isub* 计算步各裂缝各裂尖的应力强度因子；结果文件类型为**.sifs_i*	*Save_SIFs_KI_and_KII*(*isub*)	
		isub	载荷子步号；整数；输入变量
Save_Stress_Node	保存 *isub* 计算步的节点应力；结果文件类型为**.strn_i*	*Save_Stress_Node*(*isub*)	