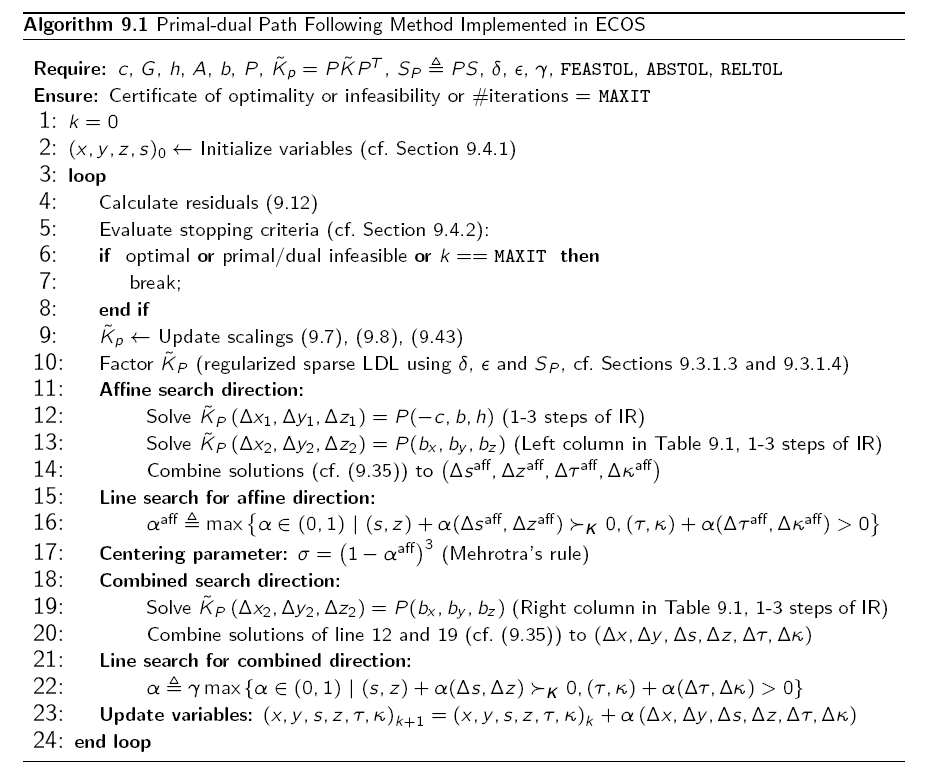
Ecos求解器

# 1 Ecos软件处理流程



# 2初始化参数



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 符号 | 含义 | 章节/页数 | 计算方式/来源 |
|  |  |  | C9.1/P164 | **Minimize**  （公式SOCP）  **Subjec to**  （公式SOCP）  （SOCP）(second-order cone programming) |
|  |  | 不等式约束矩阵 |
|  | () | 锥约束 |
|  |  | 等式约束矩阵 |
|  |  | 列向量 |
|  |  | 置换矩阵 |  | ECOS程序中AMD程序得到，相关公式如下：  （公式3.5a）  （公式3.5b）  （公式3.5c）  （公式3.5d） |
|  |  | 置换前kkt建模矩阵 | C9.3.1.3/P178 | （公式9.39）  其中/为n和p阶单位阵 |
|  |  | 置换后kkt建模矩阵 |  |  |
|  |  | -- |  | **（处理过程未使用，后续留意）** |
|  |  | 正则化系数 | C9.3.1.4/178 | 数据大概在10-7量级 |
|  |  | 动态正则化调节系数 | C9.3.1.4/179 | 数据大概在10-14量级 |
|  |  | 牛顿步长的调节系数 | C2.2.3/54 | 典型值为0.99 ，参见：Algorithm 2.5 |
|  | FEASTOL | ECOS\_OPTIMAL ECOS\_ABSTOL ECOS\_DINF退出条件 | C9.4.2/189 | 缺省参数1E-5 ，参见Table9.2 |
|  | ABSTOL | ECOS\_OPTIMAL退出条件 | 缺省参数1E-6，参见Table9.2 |
|  | RELTOL | ECOS\_OPTIMAL退出条件 | 缺省参数1E-6，参见Table9.2 |
|  | MAXIT | 最大迭代次数 | 缺省参数50，参见Table9.2 |

# 3代码处理流程

## 3.1 Line 1-2 初始化参数 iter\_k=0

****

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 符号 | 含义 | 章节/页数 | 计算方式/来源 |
|  | **（2次HW）** | **x原问题变量，**  **y等式约束拉格朗日乘子，**  **z不等式约束拉格朗日乘子**  **s松弛变量** | C9.4.1.1/187 | Step1初始求解（HW）：  (9.59）  **公式需要更新与实际代码中不符？对角线元素为**  **Step2 松弛因子：** (9.60）  **Step3 对偶变量****，****（HW）：** (9.62）  **Step4 z变量：** (9.63）  =0.9, =0.9， ，表示锥约束**代码中确定具体参数，**n=1时线性锥：，n=2时二阶锥,为集合下的一个参数 |

## 3.2 Line3 loop

****

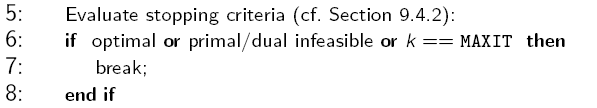
## 3.3 Line4~line23为loop内循环

## 3.4 Line 4 计算残差

****

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 符号 | 含义 | 章节/页数 | 来源 |
|  |  | **残差** | C9.2.3.1/170 | 原始等式：  (9.12）  **Step1：** 初值为1， |

## 3.5 Line5-8 判断循环退出条件

****

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 符号 | 含义 | 章节/页数 | 来源 |
|  |  | **退出条件** | 9.2.3.1/188 | Step1求解：  （9.64a）  （9.64a）  （9.64a）  （9.64b）  （xmind整理程序需要）  **Step2求解ECOS\_OPTIMAL退出条件：**  （9.65a）  （9.65b）  **Step3求解ECOS\_PINF退出条件：**  （9.65c）  **Step4求解ECOS\_NINF退出条件：**  （9.65d）  **Step5求解ECOS\_MAXIT退出条件：**  Iter\_k**=MAXIT** |

## 3.6 Line9 更新scaling

****

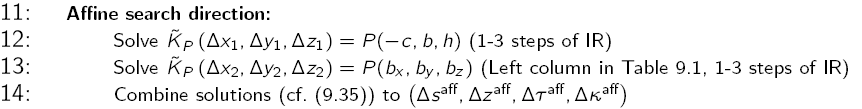
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 符号 | 含义 | 章节/页数 | 来源 |
|  |  | **NT-scaling**  **(变换矩阵)** | 9.2.2.1/168 | **Step1：**  NT-sacling for Positive Orthant  ： （9.7）  NT-sacling for Second-Order Cone ，  ：  ，，      （9.8）  最终的步进sacle矩阵和锥约束矩阵存在联系，参见（公式9.1） （取所有计算结果**1x1,2x2**）  W的存在是防止广义的A矩阵行列式为0 |
|  |  | **正则化参数** | C9.2.3.1/170 | **Step1：**  （9.39）  **Step2：** |

## 3.7 Line10：Factor 进行LDL文件前需要考虑的参数 （regularized sparse LDL using ,,）

****

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 符号 | 含义 | 章节/页数 | 来源 |
|  | 正则化参数  **两种方式** | **静态正则化** | 9.3.1.3/178 | （9.39）  正则化参数：太大将干扰线性系统，太小将使矩阵准正定困难。  合理数值可以让PKPT矩阵进行LDLT分解稳定，直到迭代趋于最优解，合理的  范围大约在10-6…10-8。代码中10-7（注论文中P189页考虑） |
|  | **动态正则化**  （**感觉是做锦上添花的效果，消除有限的精度误差，如何使用完全不理解**） | 9.3.1.4/179 | 参数与初始化一致  ，为符号向量与二阶锥相关（，，理解存在问题）  ，**针对稀疏K表达在现有的ECOS中实现如下**  with  **（暂不确定动态处理如何添加到代码中）** |
|  |  |  |  | （9.43）  （9.8）  （9.41b）  （9.41c）  （9.42b）  （9.42c）  （9.42d） |

## 3.8 Line11~14：Affine search direction（）含义不确定

****

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 符号 | 含义 | 章节/页数 | 来源 |
|  | **(HW)** | 第1次  求解结果 | C9.2.3.6/175 | 原始公式 （9.34a）  变换等式(HW)：  **原始公式如下：**  **公式变形备注**：AMD具体操作流程：，令，则  两边分别乘以，则，需要注意的是计算后获取的结果为，需要进行二次处理进行操作，得到最终的，论文中为（P60页中公式3.5a~d）算法要求的处理，程序中初步求解的结果，**通过refinement**完**成最终的计算处理**。 |
|  | **(HW)** | 第2次  求解结果 | C9.2.3.6/175 | 原始公式 （9.34b）  变换等式(HW)：    **Table9.1**   |  |  | | --- | --- | | **RHS** | **Affine** | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  |   其中:      （9.6） |
|  |  | **仿射搜索方向**  **融合第1/2次线性方程组求解结果计算** |  | （9.35a）  （9.35d）  （9.35e）  （9.35f） |

## 3.9 Line15~16：line search for affine direction:

****

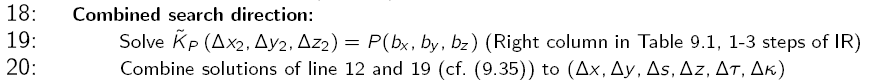
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 符号 | 含义 | 章节/页数 | 来源 |
|  |  | **仿射步长**  范围（0~1） |  | 搜索使更新后的满足：  锥约束1：  锥约束2：  **采用line search的方法**：  Step1 ：初始化；  Step2 ：判断是否满足2个锥约束要求,满足时进入step3不满足时，，重新进入step2；  Step3 ：退出搜索。 |

## 3.10 Line17：Centering Parameter 中心路径参数:

****

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 符号 | 含义 | 章节/页数 | 来源 |
|  |  | **中心参数**  范围（0~1） |  |  |

## 3.11 Line18~20：combined search direction

****

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 符号 | 含义 | 章节/页数 | 来源 |
|  | **(HW)** | 第3次  求解结果 | C9.2.3.6/175 | 原始等式：  Table9.1   |  |  | | --- | --- | | **RHS** | **Combined（centering & corrector）** | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  |   其中：  （9.6）  （9.2）  其中**为锥约束个数**，定义为，，线性/二阶锥计算结果均为1。  表征的集合 |
|  |  | 联合搜索方向  **融合第1/3次**  **求解结果** | C9.2.3.6/175 | （9.35a）  （9.35b）  （9.35c）  （9.35d）  （9.35e）  （9.35f） |

## 3.12 Line21：line search for combine direction:



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 符号 | 含义 | 章节/页数 | 来源 |
|  |  | 联合步长 |  | 搜索使更新后的满足：  锥约束1：  锥约束2：  **采用line search的方法**：  Step1 ：初始化；  Step2 ：判断是否满足2个锥约束要求,满足时进入step3不满足时，，重新进入step2；  Step3 ：退出搜索。 |

## 3.13 Line22：更新变量:



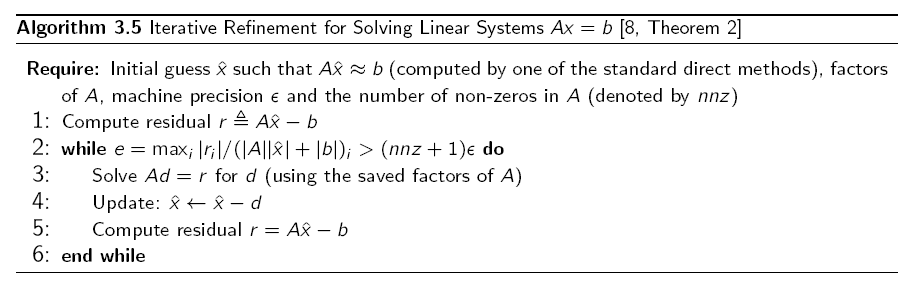
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 符号 | 含义 | 章节/页数 | 来源 |
|  |  | 原问题变量x，  等式约束拉格朗日乘子y，  松弛变量s，  不等式约束拉格朗日乘子z  松弛变量t  松弛变量k |  |  |

## 3.14 Line23：结束循环:

****

## 3.15 修正精度：重要

每次slove后迭代处理

****

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 符号 | 含义 | 章节/页数 | 来源 |
|  | **(HW)** | 修正后  问题变量 | C3.3/66 | Step1：计算残差  其中  Step2：循环判决  ，其中nnz为中非零元素个数，=10-14  线性方程组求解（HW）：  获取d  更新：  计算残差： |

# 4软件提升效率方式（王聪/施建峰）

## 4.1处理精度

1.1）Ecos算法内部数据运算精度double->**float**

## 4.2修正迭代退出条件

2.1）修正ecos中退出机制**< ecos.h>**

**#define MAXIT (100) –>(50) /\* maximum number of iterations \*/**

**#define FEASTOL (1E-7) –>(1E-3) /\* primal/dual infeasibility tolerance \*/**

**#define ABSTOL (1E-7) –>(1E-3) /\* absolute tolerance on duality gap \*/**

**#define RELTOL (1E-7) –>(1E-3) /\* relative tolerance on duality gap \*/**

**#define FTOL\_INACC (1E-4) –>(1E-2) /\* inaccurate solution feasibility tol. \*/**

**#define ATOL\_INACC (5E-5) –>(1E-2) /\* inaccurate solution absolute tol. \*/**

**#define RTOL\_INACC (5E-5) –>(1E-2) /\* inaccurate solution relative tol. \*/**

2.2）修正slove求解器中refinement退出机制

**#define NITREF (9) –>(1) /\* number of iterative refinement steps \*/**

## 4.3其他

3.1）修正Kp矩阵中w项参数**<cone.h>**

**#define** CONEMODE (0) **–>(1)** **/\* 0: expand to sparse cones (ECOS standard) \*/**

**/\* 1: dense cones (slow for big cones) \*/**

**/\* 2: dense of fixed size \*/**

# 4软件代码梳理

代码反推的公式如下







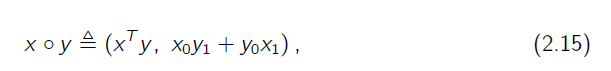
















Linesearch：





重复一下操作ncones次（二阶锥个数）

if( alpha > STEPMAX ) alpha = STEPMAX;

if( alpha < STEPMIN ) alpha = STEPMIN;

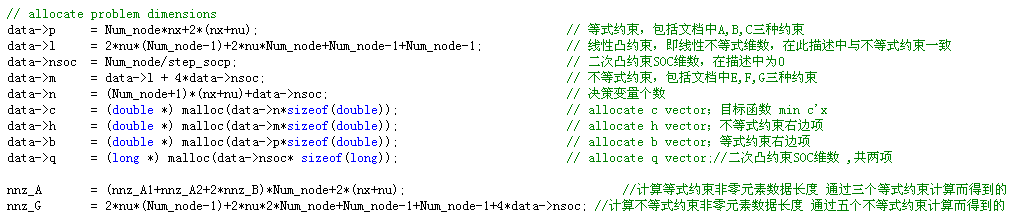


Refinement处理方式





## 4.1 ECOS\_setup



### 4.1.1输入要求

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **序号** | **目的** | **信号格式** | **信号名/初始值** | **数据**  **长度** | **说明** |
|  |  | idxint | n | 1 | number of primal variables x  决策变量个数 |
|  |  | idxint | m  /l+ q\* Ncones | 1 | number of conically constrained variables s  不等式约束，包括文档中E,F,G三种约束 |
|  |  | idxint | p | 1 | number of equality constraints  等式约束，包括文档中A,B,C三种约束 |
|  |  | idxint | l | 1 | number of lpcone 线性锥约束数量  线性凸约束，即线性不等式维数 |
|  |  | idxint | Ncones/16 | 1 | number of socone 二阶锥约束数量 |
|  |  | idxint\* | q/4 | ncones | 二次凸约束SOC维数 |
|  |  | pfloat\* | Gpr | nnz\_G | G矩阵csc压缩参数中data |
|  |  | idxint\* | Gjc | n+1 | G矩阵csc压缩参数中col\_num |
|  |  | idxint\* | Gir | m | G矩阵csc压缩参数中row\_idx |
|  |  | pfloat\* | Apr | nnz\_A | A矩阵csc压缩参数中data |
|  |  | idxint\* | Ajc | n+1 | A矩阵csc压缩参数中col\_num |
|  |  | idxint\* | Air | P | A矩阵csc压缩参数中row\_idx |
|  |  | pfloat\* | c | n | 目标函数 min c'x |
|  |  | pfloat\* | h | m | 不等式约束右边项 |
|  |  | pfloat\* | b | p | 等式约束右边项 |

A：计算等式约束非零元素数据长度 通过三个等式约束计算而得到的

G：计算不等式约束非零元素数据长度 通过五个不等式约束计算而得到的

### 4.1.2处理要求

#### 4.1.2.1初始化部分

1）完成mywork的数据结构分配 Memory allocated for WORK struct

mywork = (pwork \*)MALLOC(sizeof(pwork));

2）设置广义矩阵A维度信息：PRINTTEXT("Set dimensions\n");

mywork->n = n;

mywork->m = m;

mywork->p = p;

mywork->D = l + ncones;

**指针直接赋值处理--**Hung pointers for c, h and b into WORK struct

mywork->c = c;

mywork->h = h;

mywork->b = b;

3）完成mywork内部指针数据类型数据空间分配 Memory allocated for variables

mywork->x = (pfloat \*)MALLOC(n\*sizeof(pfloat));

mywork->y = (pfloat \*)MALLOC(p\*sizeof(pfloat));

mywork->z = (pfloat \*)MALLOC(m\*sizeof(pfloat));

mywork->s = (pfloat \*)MALLOC(m\*sizeof(pfloat));

mywork->lambda = (pfloat \*)MALLOC(m\*sizeof(pfloat));

mywork->dsaff\_by\_W = (pfloat \*)MALLOC(m\*sizeof(pfloat));

mywork->dsaff = (pfloat \*)MALLOC(m\*sizeof(pfloat));

mywork->dzaff = (pfloat \*)MALLOC(m\*sizeof(pfloat));

mywork->saff = (pfloat \*)MALLOC(m\*sizeof(pfloat));

mywork->zaff = (pfloat \*)MALLOC(m\*sizeof(pfloat));

mywork->W\_times\_dzaff = (pfloat \*)MALLOC(m\*sizeof(pfloat));

4）cone内部空间分配赋值操作 Memory allocated for cone struct

mywork->best\_x = (pfloat \*)MALLOC(n\*sizeof(pfloat));

mywork->best\_y = (pfloat \*)MALLOC(p\*sizeof(pfloat));

mywork->best\_z = (pfloat \*)MALLOC(m\*sizeof(pfloat));

mywork->best\_s = (pfloat \*)MALLOC(m\*sizeof(pfloat));

mywork->best\_info = (stats \*)MALLOC(sizeof(stats));

5）lpcone内部空间分配与初始化赋值操作 Memory allocated for LP cone

mywork->C->lpc = (lpcone \*)MALLOC(sizeof(lpcone));

mywork->C->lpc->p = l;

mywork->C->lpc->w = (pfloat \*)MALLOC(l\*sizeof(pfloat));

mywork->C->lpc->v = (pfloat \*)MALLOC(l\*sizeof(pfloat));

mywork->C->lpc->kkt\_idx = (idxint \*)MALLOC(l\*sizeof(idxint));

5）socone内部空间分配与初始化赋值操作 Memory allocated for second-order cones

**for( i=0; i<ncones; i++ )**{

conesize = (idxint)q[i];

mywork->C->soc[i].p = conesize;

mywork->C->soc[i].a = 0;

mywork->C->soc[i].eta = 0;

mywork->C->soc[i].q = (pfloat \*)MALLOC((conesize-1)\*sizeof(pfloat));

mywork->C->soc[i].skbar = (pfloat \*)MALLOC((conesize)\*sizeof(pfloat));

mywork->C->soc[i].zkbar = (pfloat \*)MALLOC((conesize)\*sizeof(pfloat));

#if CONEMODE == 0

mywork->C->soc[i].Didx = (idxint \*)MALLOC((conesize)\*sizeof(idxint));

#endif

#if CONEMODE > 0

mywork->C->soc[i].colstart = (idxint \*)MALLOC((conesize)\*sizeof(idxint));

#endif

cidx += conesize;

}

6）信息内部空间分配与初始化赋值操作 Memory allocated for info

mywork->info = (stats \*)MALLOC(sizeof(stats));

mywork->info->tfactor = 0;

mywork->info->tkktsolve = 0;

mywork->info->tfactor\_t1 = 0;

mywork->info->tfactor\_t2 = 0;

7）平衡矢量空间分配操作 Memory allocated for equilibration vectors

#if defined EQUILIBRATE && EQUILIBRATE > 0

mywork->xequil = (pfloat \*)MALLOC(n\*sizeof(pfloat));

mywork->Aequil = (pfloat \*)MALLOC(p\*sizeof(pfloat));

mywork->Gequil = (pfloat \*)MALLOC(m\*sizeof(pfloat));

#endif

8）配置参数空间分配与初始化操作 Written settings

mywork->stgs = (settings \*)MALLOC(sizeof(settings));

mywork->stgs->maxit = MAXIT;

mywork->stgs->gamma = GAMMA;

mywork->stgs->delta = DELTA;

mywork->stgs->eps = EPS;

mywork->stgs->nitref = NITREF;

mywork->stgs->abstol = ABSTOL;

mywork->stgs->feastol = FEASTOL;

mywork->stgs->reltol = RELTOL;

mywork->stgs->abstol\_inacc = ATOL\_INACC;

mywork->stgs->feastol\_inacc = FTOL\_INACC;

mywork->stgs->reltol\_inacc = RTOL\_INACC;

mywork->stgs->verbose = VERBOSE;

9）初始化AG矩阵，**同样为指针指向**

/\* Store problem data \*/

mywork->A = createSparseMatrix(p, n, Ajc[n], Ajc, Air, Apr);

mywork->G = createSparseMatrix(m, n, Gjc[n], Gjc, Gir, Gpr);

10）初始化ATGT矩阵，**同样为指针指向**

//转置处理//

At = transposeSparseMatrix(mywork->A);

Gt = transposeSparseMatrix(mywork->G);

10）初始化KU矩阵矩阵，/\* set up KKT system ？KU\*/ spmat\* KU

createKKT\_U(Gt, At, mywork->C, &Sign, &KU); （**施博士可以详细解释下**）

#if DEBUG > 0

dumpSparseMatrix(KU, "KU0.txt"); //完成KU矩阵下载

#endif

11）初始化KKT矩阵矩阵，Created memory for KKT-related data

注：（nK = KU->n为KU矩阵的列数？）;

mywork->KKT = (kkt \*)MALLOC(sizeof(kkt));

mywork->KKT->D = (pfloat \*)MALLOC(nK\*sizeof(pfloat));

mywork->KKT->Parent = (idxint \*)MALLOC(nK\*sizeof(idxint));

mywork->KKT->Pinv = (idxint \*)MALLOC(nK\*sizeof(idxint));

mywork->KKT->work1 = (pfloat \*)MALLOC(nK\*sizeof(pfloat));

mywork->KKT->work2 = (pfloat \*)MALLOC(nK\*sizeof(pfloat));

mywork->KKT->work3 = (pfloat \*)MALLOC(nK\*sizeof(pfloat));

mywork->KKT->work4 = (pfloat \*)MALLOC(nK\*sizeof(pfloat));

mywork->KKT->work5 = (pfloat \*)MALLOC(nK\*sizeof(pfloat));

mywork->KKT->work6 = (pfloat \*)MALLOC(nK\*sizeof(pfloat));

mywork->KKT->Flag = (idxint \*)MALLOC(nK\*sizeof(idxint));

mywork->KKT->Pattern = (idxint \*)MALLOC(nK\*sizeof(idxint));

mywork->KKT->Lnz = (idxint \*)MALLOC(nK\*sizeof(idxint));

mywork->KKT->RHS1 = (pfloat \*)MALLOC(nK\*sizeof(pfloat));

mywork->KKT->RHS2 = (pfloat \*)MALLOC(nK\*sizeof(pfloat));

mywork->KKT->dx1 = (pfloat \*)MALLOC(mywork->n\*sizeof(pfloat));

mywork->KKT->dx2 = (pfloat \*)MALLOC(mywork->n\*sizeof(pfloat));

mywork->KKT->dy1 = (pfloat \*)MALLOC(mywork->p\*sizeof(pfloat));

mywork->KKT->dy2 = (pfloat \*)MALLOC(mywork->p\*sizeof(pfloat));

mywork->KKT->dz1 = (pfloat \*)MALLOC(mywork->m\*sizeof(pfloat));

mywork->KKT->dz2 = (pfloat \*)MALLOC(mywork->m\*sizeof(pfloat));

mywork->KKT->Sign = (idxint \*)MALLOC(nK\*sizeof(idxint));

mywork->KKT->PKPt = newSparseMatrix(nK, nK, KU->nnz);

mywork->KKT->PK = (idxint \*)MALLOC(KU->nnz\*sizeof(idxint));

#### 4.1.2.2计算部分

/\* calculate ordering of KKT matrix using AMD \*/

P = (idxint \*)MALLOC(nK\*sizeof(idxint));

//程序小，无返回结果。无全局变量不理解

1. AMD\_defaults(Control);

1. amd\_result = AMD\_order(nK, KU->jc, KU->ir, P, Control, Info)

AMD\_debug\_init

AMD\_valid

AMD\_DEBUG1

amd\_malloc

amd\_free

AMD\_preprocess

AMD\_aat

AMD\_1

AMD\_2

clear\_flag

1. AMD\_info(Info);

/\* calculate inverse permutation and permutation mapping of KKT matrix \*/

1. pinv(nK, P, mywork->KKT->Pinv);
2. Pinv = mywork->KKT->Pinv;

#if DEBUG > 0

dumpDenseMatrix\_i(P, nK, 1, "P.txt");

dumpDenseMatrix\_i(mywork->KKT->Pinv, nK, 1, "PINV.txt");

#endif

1. permuteSparseSymmetricMatrix(KU, mywork->KKT->Pinv, mywork->KKT->PKPt , mywork->KKT->PK); 参考源码和论文中描述是否一致

**置换处理（完成元素的坐标调换处理）**

**PKPt\_t = P \* KU \* Pinv**

**PKPt = triu(PKPt\_t)+ triu(PKPt\_t’)-diag(diag(PKPt\_t))**

1. LDL\_symbolic2(

mywork->KKT->PKPt->n, /\* A and L are n-by-n, where n >= 0 \*/

mywork->KKT->PKPt->jc, /\* input of size n+1, not modified \*/

mywork->KKT->PKPt->ir, /\* input of size nz=Ap[n], not modified \*/

Ljc, /\* output of size n+1, not defined on input \*/

mywork->KKT->Parent, /\* output of size n, not defined on input \*/

mywork->KKT->Lnz, /\* output of size n, not defined on input \*/

mywork->KKT->Flag /\* workspace of size n, not defn. on input or output \*/

);

开辟存储空间

Lir = (idxint \*)MALLOC(lnz\*sizeof(idxint));

Lpr = (pfloat \*)MALLOC(lnz\*sizeof(pfloat));

1. mywork->KKT->L = ecoscreateSparseMatrix(nK, nK, lnz, Ljc, Lir, Lpr);
2. permuteSparseSymmetricMatrix(KU, mywork->KKT->Pinv, mywork->KKT->PKPt, NULL); ？ 为什么和前面第6行代码基本一致

### 4.1.3输出要求pwork

**L：子结构体内部初始化**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **序号** | **目的** | **信号格式** | **信号名** | **数据**  **长度** | **说明** |
|  | dimensions | idxint | n | 1 | number of primal variables x |
|  | idxint | m | 1 | number of conically constrained variables s |
|  | idxint | p | 1 | number of equality constraints |
|  | idxint | D | 1 | degree of the cone （ETE**为锥约束个数**）  例如线性锥202个二次锥15个，总和为217个 |
|  | **variables** | pfloat\* | **x** | n | Primal variables |
|  | pfloat\* | **y** | P | multipliers for equality constaints ?等式约束拉格朗日乘子 |
|  | pfloat\* | **z** | m | multipliers for conic inequalities ? 不等式约束拉格朗日乘子 |
|  | pfloat\* | **s** | m | slacks for conic inequalities松弛因子 |
|  | pfloat\* | lambda | m | scaled variable(计算公式见（9.6）) |
|  | pfloat | **kap** | 1 | kappa (homogeneous embedding) |
|  | pfloat | **tau** | 1 | tau (homogeneous embedding) |
|  | **best iterate seen so far**  **variables** | pfloat\* | best\_x | n | primal variables |
|  | pfloat\* | best\_y | p | multipliers for equality constaints |
|  | pfloat\* | best\_z | m | multipliers for conic inequalities |
|  | pfloat\* | best\_s | m | slacks for conic inequalities |
|  | pfloat | best\_kap | 1 | kappa (homogeneous embedding) |
|  | pfloat | best\_tau | 1 | tau (homogeneous embedding) |
|  | pfloat | best\_cx | 1 |  |
|  | pfloat | best\_by | 1 |  |
|  | pfloat | best\_hz | 1 |  |
|  | **stats\*** | **best\_info** | **1** | info of best iterate |
|  | **temporary stuff holding search direction etc.** | pfloat\* | dsaff | m |  |
|  | pfloat\* | dzaff | m |  |
|  | pfloat\* | **W\_times\**  **\_dzaff** | m |  |
|  | pfloat\* | **dsaff\_by\**  **\_W** | m |  |
|  | pfloat\* | saff | m |  |
|  | pfloat\* | zaff | m |  |
|  | **cone** | **cone\*** | **C** | **1/L** |  |
|  | **problem data** | **spmat\*** | **A** |  |  |
|  | **spmat\*** | **G** |  |  |
|  | pfloat\* | c | c | 空间大小和输入空间大小一致 |
|  | pfloat\* | b | b |  |
|  | pfloat\* | h | h |  |
|  | **equilibration vector**  **if defined EQUILIBRATE && EQUILIBRATE**  **> 0** | pfloat | \*xequil | n |  |
|  | pfloat | \*Aequil | p |  |
|  | pfloat | \*Gequil | n |  |
|  | **scalings of problem data** | pfloat | resx0 |  |  |
|  | pfloat | resy0 |  |  |
|  | pfloat | resz0 |  |  |
|  | **residuals** | pfloat | \*rx |  |  |
|  | pfloat | \*ry |  |  |
|  | pfloat | \*rz |  |  |
|  | pfloat | rt |  |  |
|  | pfloat | hresx |  |  |
|  | pfloat | hresy |  |  |
|  | pfloat | hresz |  |  |
|  | **temporary storage** | pfloat | cx |  |  |
|  | pfloat | by |  |  |
|  | pfloat | hz |  |  |
|  | pfloat | sz |  |  |
|  | **KKT System** | **Kkt\*** | **KKT** |  |  |
|  | **info struct** | **stats\*** | **info** |  |  |
|  | **settings struct** | **settings\*** | **stgs** | **1** |  |

## 4.8 软件代码相关信息

### 结构体stats

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **序号** | **目的** | **信号格式** | **信号名** | **说明** |
|  |  | pfloat | pcost |  |
|  |  | pfloat | dcost |  |
|  |  | pfloat | pres |  |
|  |  | pfloat | dres |  |
|  |  | pfloat | pinf |  |
|  |  | pfloat | dinf |  |
|  |  | pfloat | pinfres |  |
|  |  | pfloat | dinfres |  |
|  |  | pfloat | gap |  |
|  |  | pfloat | relgap |  |
|  |  | pfloat | sigma |  |
|  |  | pfloat | mu |  |
|  |  | pfloat | step |  |
|  |  | pfloat | step\_aff |  |
|  |  | pfloat | kapovert |  |
|  |  | idxint | iter |  |
|  |  | idxint | nitref1 |  |
|  |  | idxint | nitref2 |  |
|  |  | idxint | nitref3 |  |
|  | PROFILING  >0 | pfloat | tsetup | Ecos\_setup程序运行时间 |
|  | pfloat | tsolve |  |
|  | PROFILING  > 1 | pfloat | tfactor / 0 |  |
|  | pfloat | tkktsolve / 0 |  |
|  | pfloat | torder | 计算排序时间AMD\_defaults，AMD\_order排序时间 |
|  | pfloat | tkktcreate | 计算kkt矩阵生成时间 |
|  | pfloat | ttranspose | 计算转置总时间信息 |
|  | pfloat | tperm |  |
|  | pfloat | tfactor\_t1 / 0 |  |
|  | pfloat | tfactor\_t2 / 0 |  |

### 结构体cone

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **序号** | **条件** | **信号格式** | **信号名** | **数据**  **长度** | **说明** |
|  | l > 0 | **lpcone\*** | **lpc** | 1 | LP cone |
|  | **socone\*** | **soc** | ncones | Second-Order cone |
|  | idxint | nsoc | 1 | Number of second-order cones |

### 结构体lpcone

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **序号** | **条件** | **信号格式** | **信号名**  **/初始值** | **数据**  **长度** | **说明** |
|  | l > 0 | idxint | P / l | 1 | dimension of cone |
|  | pfloat\* | w | l | scalings |
|  | pfloat\* | v | l | = w^2 - saves p multiplications |
|  | **idxint\*** | **kkt\_idx** | **l** | **Indices indices of KKT matrix to which scalings w^2 map** |

### 结构体socone

备注：

1）conesize = (idxint)q[i];

2）二阶锥初始化循环内部：cidx += conesize

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **序号** | **条件** | **信号格式** | **信号名**  **/初始值** | **数据**  **长度** | **说明** |
|  | l > 0 | idxint | p | 1 | dimension of cone |
|  | pfloat\* | skbar | conesize  （二阶锥维数） | temporary variables to work with （参见公式9.8） |
|  | pfloat\* | zkbar | conesize | temporary variables to work with  （参见公式9.8） |
|  | pfloat | a / 0 | 1 | **= wbar(1)**  **的第一个数据** |
|  |  | pfloat | d1 | 1 | first element of D |
|  |  | pfloat | W | 1 | = q'\*q |
|  |  | pfloat | eta / 0 | 1 | eta = (sres / zres)^(1/4) |
|  |  | pfloat | eta\_square | 1 | eta^2 = (sres / zres)^(1/2) |
|  |  | pfloat\* | q | (conesize-1) | **= wbar(2:end)**  **的第2个以后的 数据** |
|  | CONEMODE  ==  0 | idxint\* | Didx | conesize | indices for D |
|  | pfloat | u0 | 1 | eta |
|  | pfloat | u1 | 1 | u = [u0; u1\*q] |
|  | pfloat | v1 | 1 | v = [0; v1\*q] |
|  | CONEMODE  >  0 | idxint\* | colstart | conesize | colstart[n] gives index in KKT matrix where  the nth column of this scaling matrix in (3,3)  block starts |
|  | pfloat | c | 1 | = 1 + a + w/(1+a) |
|  | pfloat | d | 1 | = 1 + 2/(1+a) + w/(1+a)^2 |

### 结构体spmat

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **序号** | **目的** | **信号格式** | **信号名** | **说明** |
|  |  | idxint\* | jc |  |
|  |  | idxint\* | ir |  |
|  |  | pfloat\* | pr |  |
|  |  | idxint | n |  |
|  |  | idxint | m |  |
|  |  | idxint | nnz |  |

### 结构体KKT

nK为KU矩阵中列数，也就是Ax=b中A的列数，b的行数

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **序号** | **条件** | **信号格式** | **信号名**  **/初始值** | **数据**  **长度** | **说明** |
|  |  | spmat | PKPt | newSparseMatrix  (nK,  nK,  KU->nnz) | Permuted KKT matrix, upper part only |
|  |  | spmat\* | L |  | LDL factor L |
|  |  | pfloat\* | D | nK | diagonal matrix D |
|  |  | idxint\* | work1 | nK | workspace needed for factorization  **kkt\_solve函数中Px变量：**  **表征LDLx=b 的3次临时求解结果** |
|  |  | pfloat\* | work2 | nK | workspace needed for factorization  **kkt\_solve函数中dPx变量：**  **表征LDLx=b中的refinement中3次临时求解结果** |
|  |  | pfloat\* | work3 | nK | workspace needed for factorization  kkt\_solve函数中e变量：  表征 |
|  |  | pfloat\* | work4 | nK | **表征LDLx=b中的refinement中第一次次b参数** |
|  |  | pfloat\* | work5 | nK | truez |
|  |  | pfloat\* | work6 | nK | **Gdx这个没有想清楚，问什么需要单独存储** |
|  |  | pfloat\* | RHS1 | nK | Right hand side 1 |
|  |  | pfloat\* | RHS2 | nK | Right hand side 2 |
|  |  | pfloat\* | dx1 | mywork->n | search direction of size n |
|  |  | pfloat\* | dx2 | mywork->n | search direction of size n |
|  |  | pfloat\* | dy1 | mywork->p | search direction of size p |
|  |  | pfloat\* | dy2 | mywork->p | search direction of size p |
|  |  | pfloat\* | dz1 | mywork->m | search direction of size m |
|  |  | pfloat\* | dz2 | mywork->m | search direction of size m |
|  |  | idxint\* | P | nK | permutation |
|  |  | idxint\* | Pinv | nK | reverse permutation |
|  |  | idxint\* | PK | KU->nnz | permutation of row indices of KKT matrix |
|  |  | idxint\* | Parent | nK | Elimination tree of factorization()，**LDL\_symbolic2() LDL前空间分配** |
|  |  | idxint\* | Sign | nK | Permuted sign vector for regularization  **LDL\_numeric2函数中使用** |
|  |  | idxint\* | Pattern | nK | idxint workspace needed for factorization |
|  |  | idxint\* | Flag | nK | idxint workspace needed for factorization  **LDL\_symbolic2() LDL前空间分配** |
|  |  | idxint\* | Lnz | nK | idxint workspace needed for factorization  **LDL\_symbolic2() LDL前空间分配，每列数据个数，剔除对角线元素** |
|  |  | pfloat | delta | 1 | size of regularization |

### ­结构体settings

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **序号** | **目的** | **信号格式** | **信号名**  **/初始值** | **说明** |
|  |  | pfloat | gamma/GAMMA | scaling the final step length |
|  |  | pfloat | delta/DELTA | regularization parameter |
|  |  | pfloat | eps/EPS | regularization threshold |
|  |  | pfloat | feastol/FEASTOL | primal/dual infeasibility tolerance |
|  |  | pfloat | abstol/ABSTOL | absolute tolerance on duality gap |
|  |  | pfloat | reltol/RELTOL | relative tolerance on duality gap |
|  |  | pfloat | feastol\_inacc/FTOL\_INACC | primal/dual infeasibility relaxed tolerance |
|  |  | pfloat | abstol\_inacc/ATOL\_INACC | absolute relaxed tolerance on duality gap |
|  |  | pfloat | reltol\_inacc/RTOL\_INACC | relative relaxed tolerance on duality gap |
|  |  | idxint | nitref/NITREF | number of iterative refinement steps |
|  |  | idxint | maxit / MAXIT | maximum number of iterations |
|  |  | idxint | verbose/VERBOSE | verbosity bool for PRINTLEVEL < 3 |

数据初始化参见ecos.h

### 结构体spmat

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **序号** | **目的** | **信号格式** | **信号名**  **/初始值** | **说明** |
|  |  | idxint\* | jc | 矩阵csc压缩参数中col\_num |
|  |  | idxint\* | ir | 矩阵csc压缩参数中row\_idx |
|  |  | pfloat | pr | 矩阵csc压缩参数中data |
|  |  | idxint | n | 矩阵中列数 |
|  |  | idxint | m | 矩阵中行数 |
|  |  | idxint | Nnz | 非零元素个数 |

**Font\_style：结构体数据类型**

**Pinv函数 P(i)=j 转置后P(j)=I，将j=P(i)代入得到**

**P(P(i))=i**

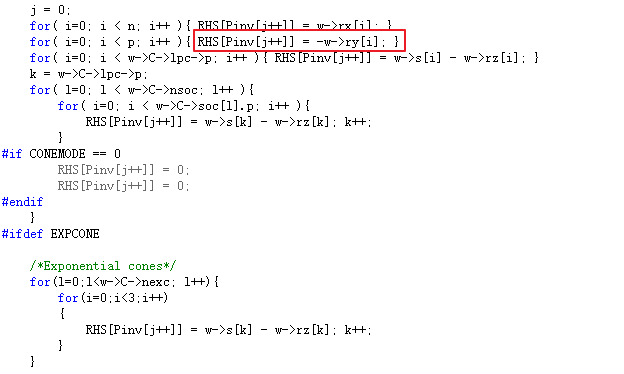
## 4.8 软件问题

1）指数锥的问题需进行二次确认

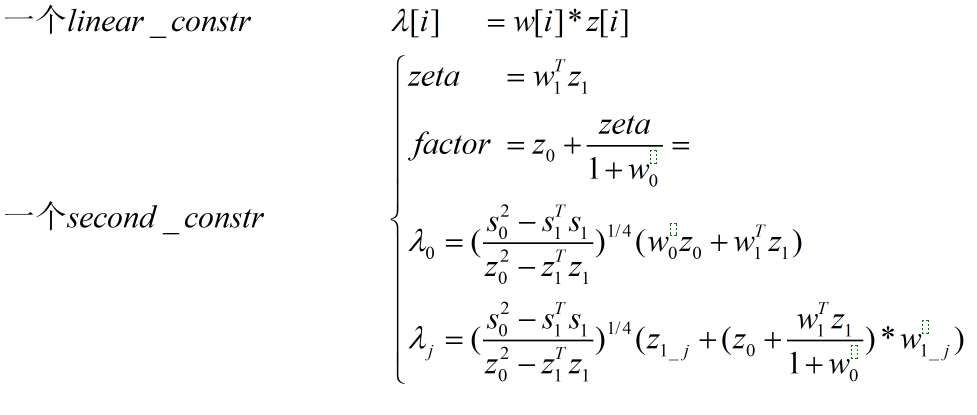
Glblopts.h中存在定义，但ecos\_setup中对指数锥的数量赋值为0，建议算法人员对其进行详细说明

#define EXPCONE /\*When defined the exponential cone solver code is enabled\*/

2）迭代过程中第一次求解RHS1-c.b,h不发生变化，AFFINE SEARCH DIRECTION



3）算法人员确认scale公式的正确性 如果w改变lambda同时改变？



4）迭代过程中第1/2次线性方程组求解combine过程中公式不符？

|  |  |  |
| --- | --- | --- |
| **序号** | **代码反推** | **理论公式** |
| **1** |  |  |
| **2** |  |  |
| **3** |  |  |

其中：

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

# 5 ARM移植工作

## 硬件加速前提

硬件加速的前提需完成算法流程梳理，对算法执行过程中时间瓶颈进行确认，先前的时间统计信息核心处理单元为TI公司的C6678-DSP芯片，虽然具有一定的参考性，但由于课题算法实现平台为ZCU102开发版，内部核心处理单元为xilinx公司的ZU9EG，需要重新完成ecos代码的ARM处理器移植操作。统计ECOS算法内部核心运算单元（LDL分解，前代、回代）的运行次数与处理时间，确定处理百分比。

## ARM软件移植信息

1）下载github官网-- <https://github.com/embotech/ecos> 最新版本程序（2.0.7）。

2）建立VS软件工程，采用src/runecos.c作为整个工程的主函数。

3）主要修改以下部分完成信息统计与功能验证：

3.1）添加矩阵LDL分解的次数统计kkt\_factor\_cnt，添加矩阵前代/D/回代三次求解的时间信息统计分别为一下参数：ldl\_lsolve2\_cnt，ldl\_lsolve2\_time，ldl\_dsolve\_cnt，ldl\_dsolve\_time，ldl\_ltsolve\_cnt，ldl\_ltsolve\_time；

3.2）添加不同问题描述下的ecos求解器输入参数（github缺省参数，月球着陆问题参数-**巩庆海/施健峰帮忙**，火箭回收问题参数-**骆无意帮忙**）

3.3）添加输入数据/过程数据/输出数据导出程序完成数据的导入/导出，完成ecos程序理解-**施健峰/王聪辅助**。

## ARM处理结果(debug)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **时间信息统计对比** | | | | |
| **参数类型** |  | **1次ECOS**  **处理操作** | **ARM时间(ms)** | **CPU时间(ms)** |
| **github 缺省参数问题描述** | Ecos\_setup |  | 4.24 | 0.708 |
| Ecos\_solve | KKT\_factor\_num/13 | 15.39 | 1.227 |
| ldl\_ltsolve\_num/92 | 6.41668 | 0.31778329 |
| ldl\_dsolve\_num/92 | 2.61379 | 0.11802588 |
| ldl\_lsolve2\_num/92 | 8.40117 | 0.40477888 |
| others | 38.16836 | 1.93241195 |
| Solve\_time | 70.99 | 4 |
| TOTAL |  | **75.23** | **4.709** |
| **待加速**  **部分占比** |  | **43.63%** | **43.91%** |
| **Lunar-Landing 问题描述** | Setup |  | 17.77 | 2.59 |
| Solve | KKT\_factor\_num/19 | 152.3 | 10.156 |
| ldl\_ltsolve\_num/115 | 75.40503 | 3.54160745 |
| ldl\_dsolve\_num/115 | 12.09147 | 0.53610345 |
| ldl\_lsolve2\_num/115 | 91.94037 | 3.9037197 |
| others | 194.11313 | 10.2515694 |
| Solve\_time | 525.85 | 28.389 |
| TOTAL |  | **543.62** | **30.978** |
| **待加速**  **部分占比** |  | **61.02%** | **58.55%** |
| **Earth-Landing 问题描述** | Setup |  | 6.25 | 0.997 |
| Solve | KKT\_factor\_num/12 | 29.15 | 2.244 |
| ldl\_ltsolve\_num/67 | 12.7974 | 0.68294316 |
| ldl\_dsolve\_num/67 | 2.47924 | 0.11802588 |
| ldl\_lsolve2\_num/67 | 15.75538 | 0.74472665 |
| others | 39.72798 | 2.07830431 |
| Solve\_time | 99.91 | 5.868 |
| TOTAL |  | **106.16** | **6.865** |
| **待加速**  **部分占比** |  | **56.69%** | **55.20%** |

## ARM处理结果(release版本)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 时间信息统计对比 | | | | |
| **参数类型** |  | **1次ECOS处理操作** | **ARM时间(ms)** | **CPU时间(ms)** |
| **github 缺省参数问题描述** | Ecos\_setup |  | **1.09** | **0.708** |
| Ecos\_solve | KKT\_factor\_num/13 | 11.545 | **1.227** |
| ldl\_ltsolve\_num/92 | 1.70649 | **0.31778329** |
| ldl\_dsolve\_num/92 | 0.73077 | **0.11802588** |
| ldl\_lsolve2\_num/92 | 1.71091 | **0.40477888** |
| others | **7.98983** | **1.93241195** |
| Solve\_time | **23.683** | **4** |
| TOTAL |  | **24.77** | **4.709** |
| **待加速**  **部分占比** |  | **63.36%** | **43.91%** |
| **Lunar-Landing 问题描述** | Setup |  | **4.94** | **2.59** |
| Solve | KKT\_factor\_num/19 | 81.52 | **10.156** |
| ldl\_ltsolve\_num/115 | 20.76322 | **3.54160745** |
| ldl\_dsolve\_num/115 | 3.23782 | **0.53610345** |
| ldl\_lsolve2\_num/115 | 19.27559 | **3.9037197** |
| others | **43.15337** | **10.2515694** |
| Solve\_time | **167.95** | **28.389** |
| TOTAL |  | **172.9** | **30.978** |
| **待加速**  **部分占比** |  | **72.18%** | **58.55%** |
| **Earth-Landing 问题描述** | Setup |  | **1.68** | **0.997** |
| Solve | KKT\_factor\_num/12 | 16.85 | **2.244** |
| ldl\_ltsolve\_num/67 | 3.57546 | **0.68294316** |
| ldl\_dsolve\_num/67 | 0.68889 | **0.11802588** |
| ldl\_lsolve2\_num/67 | 3.36523 | **0.74472665** |
| others | **8.54042** | **2.07830431** |
| Solve\_time | **33.02** | **5.868** |
| TOTAL |  | **34.71** | **6.865** |
| **待加速**  **部分占比** |  | **70.53%** | **55.20%** |

## ARM/CPU/DSP处理器信息

|  |  |  |  |
| --- | --- | --- | --- |
| 处理器信息统计对比 | | | |
| **指标** | **Cortex™-A53** | **i7-8700k** | **DSP6678** |
| **主频** | 1.5GHz  **（实际1.099G）** | 3.7GHz | 1.25G  **（实际1.0G）** |
| **缓存L1** | 32KB I / D | 三级缓存12MB | 32KB I / D |
| **缓存L2** | 1MB, | 512KB LL2 |
| **片上存储资源** | 256KB | 4MB(核间共享) |
| **内存带宽DDR4** | 64bit\*2133M | 41.6GB/s | 64bit\*1333M |

感觉ARM相对与DSP处理时间的差异，可能主要在以下几方面

1）编译链接平台：ARM为GCC，DSP为CCS；

2）处理器平台：ARM-A53为ARM-V8的处理器架构，DSP为TI基于KeyStone的多核固定浮点数字信号处理器；

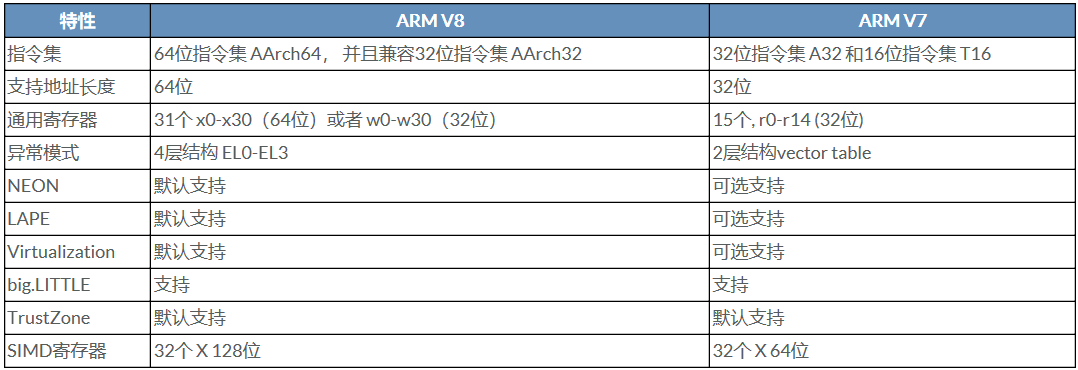
3）内部L2缓存空间，ARM为DSP的2倍大小，资料中表示：针对与大规模的数据计算，相对大的L2缓存空间具有较大优势。

## 后期工作

1）完成模块级数据传输与硬件化算法设计，通过实验进行硬件化加速效果的评估。由于对ZYNQ系列的FPGA不熟悉，**导致前期工作存在失误**，但各个模块单元的耗时比例固定，早期硬件化基本方案思路不变。

2）梳理与细化算法处理流程。确定嵌入式处理软件优化空间，目前完成代码与论文梳理过程主要集中ecos\_slove和ecos\_setup部分，有些地方还没有来的及细化，应该有些地方还有错误，后续需细化修改。

3）软硬件协同的系统优化问题；算法的处理并行与硬件资源复用；为满足**满足硬件的数据输入/处理要求**更改软件端设计。



# 6 PL端目前问题

## 6.1 HP通讯接口

PL与PS端双向HP总线通讯，指标：数据交互位宽：128bit，总线频率100Mhz，理论带宽为12.8Gb/s

缺点：PS与PL之间通讯需要将PS的数据刷新到DDR芯片中，再从DDR导入到PL模块内部，里面降低系统处理的实时性

## ACP总线接口（大家看看有熟悉的么）

直接完成PS内部cache与PL之间的通讯，大大提高实时性，

# 7 PS端硬件加速接口代码

## 7.1 LDL分解

代码位置为LDL.c中 LDL\_numeric2

原始LDL分解LDL\_numeric2()函数调用关系

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **序号** | **函数端口** | **方向** | **调用模块** | **备注说明** |
|  | LDL\_int n, | Input | KKT->PKPt->n, | /\* A and L are n-by-n, where n >= 0 \*/ |
|  | LDL\_int Ap [ ], | Input | KKT->PKPt->jc, | /\* input of size n+1, not modified \*/ |
|  | LDL\_int Ai [ ], | Input | KKT->PKPt->ir, | /\* input of size nz=Ap[n], not modified \*/ |
|  | double Ax [ ], | Input | KKT->PKPt->pr, | /\* input of size nz=Ap[n], not modified \*/ |
|  | LDL\_int Lp [ ], | Input | KKT->L->jc, | /\* input of size n+1, not modified \*/  是否不需要传递 |
|  | LDL\_int Parent [ ], | Input | KKT->Parent, | /\* input of size n, not modified \*/  是否不需要传递 |
|  | LDL\_int Sign[ ], | Input | KKT->Sign, | /\* input of size n, sign vector of diagonal entries for regularization \*/  **感觉需要传递处理，涉及动态正则化部分，数值是否会发生变化，初始化一次即可？**  D[k] = Sign[k]\*D[k] <= eps ? Sign[k]\*delta : D[k]; |
|  | double eps, | Input | eps, | /\* input, regularization threshold \*/ |
|  | double delta, | Output | delta, | /\* input, regularization parameter \*/ |
|  | LDL\_int Lnz [ ], | Output | KKT->Lnz, | /\* output of size n, not defn. on input \*/ |
|  | LDL\_int Li [ ], | Output | KKT->L->ir, | /\* output of size lnz=Lp[n], not defined on input \*/ |
|  | double Lx [ ], | Output | KKT->L->pr, | /\* output of size lnz=Lp[n], not defined on input \*/ |
|  | double D [ ], | workspace | KKT->D, | /\* output of size n, not defined on input \*/ |
|  | double Y [ ], | workspace | KKT->work1, | /\* workspace of size n, not defn. on input or output \*/ |
|  | LDL\_int Pattern [ ], | workspace | KKT->Pattern, | /\* workspace of size n, not defn. on input or output \*/ |
|  | LDL\_int Flag [ ] | workspace | KKT->Flag | /\* workspace of size n, not defn. on input or output \*/ |

1）至少需要传递eps、delta、sign[L]3类数据长度

## 7.2 LDLTx=b求解流程

代码位置为kkt.c中下面三个函数

LDL\_lsolve2(nK, Pb, KKT->L->jc, KKT->L->ir, KKT->L->pr, Px);

LDL\_dsolve(nK, Px, KKT->D,ldl\_dsolve\_cnt);

LDL\_ltsolve(nK, Px, KKT->L->jc, KKT->L->ir,KKT->L->pr);

原始LDL分解LDL\_lsolve2()函数调用关系

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **序号** | **函数端口** | **方向** | **调用模块** | **备注说明** |
|  | int n, | Input | nK, | nk维LDL线性方程组求解 |
|  | double B [], | Input | Pb | nK维Pb向量 |
|  | int Lp [ ], | Input | KKT->L->jc | L矩阵CSC压缩格式col列积分信息 |
|  | int Li [ ], | Input | KKT->L->ir | L矩阵CSC压缩格式row列序号信息 |
|  | double Lx [ ], | Input | KKT->L->pr | L矩阵CSC压缩格式val矩阵数值信息 |
|  | double X [ ] | Output | Px | 前代处理后的临时求解结果 |

原始LDL分解LDL\_dsolve()函数调用关系

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **序号** | **函数端口** | **方向** | **调用模块** | **备注说明** |
|  | int n, | Input | nK | nk维LDL线性方程组求解 |
|  | double X [ ], | Input/Ouput | Px | 前代处理后的临时求解结果/D矩阵处理后临时结果 |
|  | double D [ ] | Input | KKT->D, | D对角矩阵 |

原始LDL分解LDL\_lsolve()函数调用关系；

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **序号** | **函数端口** | **方向** | **调用模块** | **备注说明** |
|  | LDL\_int n | Input | nK, | nk维LDL线性方程组求解 |
|  | double X [ ] | Input/Ouput | Px | D矩阵处理后临时结果/**最终处理结果** |
|  | LDL\_int Lp [ ] | Input | KKT->L->jc | L矩阵CSC压缩格式col列积分信息 |
|  | LDL\_int Li [ ] | Input | KKT->L->ir | L矩阵CSC压缩格式row列序号信息 |
|  | double Lx [ ] | Input | KKT->L->pr | L矩阵CSC压缩格式val矩阵数值信息 |

**FPGA完成前向后向迭代处理后，PFPGA加速后接口调用关系：fpga\_solve(nk,Pb,Px)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **序号** | **函数端口** | **方向** | **调用** | **备注说明** |
|  | LDL\_int nk, | Input | KKT->PKPt->n, | nk维LDL线性方程组求解 |
|  | LDL\_double Pb | Input | KKT->PKPt->jc, | nK维Pb向量 |
|  | LDL\_double Px | Input | KKT->PKPt->ir, | nK维求解结果 |

# 8 PS-PL间stream数据流通讯协议

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | pl\_clk | I | 1 | PS与PL之间双向传输时钟信息 |

## 8.1 端口信息

1）PS接收PL运算**高速**输出端口描述

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **序号** | **名称** | **方向** | **位宽** | **含义** |
| 1 | s\_axis\_tvalid | I | 1 | PL->FIFO数据传输帧数据有效信号 |
| 2 | s\_axis\_tready | O | 1 | FIFO接收准备好信号 |
| 3 | s\_axis\_tdata | I | 128 | PL->FIFO数据传输帧数据信号 |
| 4 | s\_axis\_tkeep | I | 16(128/8) | PL->FIFO数据传输帧数据字节有效信号 |
| 5 | s\_axis\_tlast | I | 1 | PL->FIFO数据传输帧结束标志信号 |

2）PS发送原始数据**高速**输出端口描述

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **序号** | **名称** | **方向** | **位宽** | **含义** |
| 1 | m\_axis\_mm2s\_tvalid | O | 1 | PS->PL数据传输帧数据有效信号 |
| 2 | m\_axis\_mm2s\_tready | I | 1 | PL接收准备好信号 |
| 3 | m\_axis\_mm2s\_tdata | O | 128 | PS->PL数据传输帧数据信号 |
| 4 | m\_axis\_mm2s\_tkeep | O | 16(128/8) | PS->PL数据传输帧数据字节有效信号 |
| 5 | m\_axis\_mm2s\_tlast | O | 1 | PS->PL数据传输帧结束标志信号 |

3）PS与PL运算**低速**双向端口描述（地址空间0x0000~0x03ff）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **序号** | **名称** | **方向** | **位宽** | **含义** |
| 1 | bram\_addr\_a, | O | 12 | 地址数据 |
| 2 | bram\_clk\_a, | O | 1 | 时钟 |
| 3 | bram\_en\_a | O | 1 | 使能信号 |
| 4 | bram\_rddata\_a | O | 32 | 读数据信号 |
| 5 | bram\_rst\_a | O | 1 | 复位信号（操作RAM空间，没有想清楚） |
| 6 | bram\_we\_a | O | 4 | 写数据字节有效信号 |
| 7 | bram\_wrdata\_a | O | 32 | 写数据信号 |

## 8.2 PS->PL数据传输协议

**PS->PL传输帧协议**

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **名称** | **位宽** | **含义** |
| 1  命令帧格式 | FRAME\_ID | 32 | 0XA0—初始化A矩阵  0XA1—初始化AT矩阵  0XA2—迭代计算A矩阵  0XA3—迭代计算AT矩阵  0X1A0—初始化A矩阵，请求LD回传数据  0X1A1—初始化AT矩阵，请求LD回传数据  0X1A2—迭代计算A矩阵，请求LD回传数据  0X1A3—迭代计算AT矩阵，请求LD回传数据  0XA8—传递K参数信息  0XB0 —初始处理求解b1向量  0X1B0—初始处理求解b1向量量，请求FX/BX数据  0XB1—初始处理求解b2向量  0X1B1—初始处理求解b2向量，请求FX/BX数据  0XB2—初始处理**并行**求解b1和b2向量  0XB8—迭代处理求解b1向量  0X1B8—迭代处理求解b1向量，请求FX/BX数据  0XB9—迭代处理求解b2向量  0X1B9—迭代处理求解b2向量，请求FX/BX数据  0XBA—迭代处理**并行**求解b1和b2向量  0XBB—迭代处理求解b3向量  0X1BB—迭代处理求解b3向量，请求FX/BX数据  0XC0—初始化过程中L的列积分信息  0XC1—初始化过程中LT的列积分信息  0XC2—初始化过程中L矩阵  0XC3—初始化过程中LT矩阵  0XC4—迭代计算中L矩阵  0XC5—迭代计算中LT矩阵  **m\_axis\_mm2s\_tdata[31:0]** |
| LENGTH | 32 | 命令长度 8bit字节为基本计数单位，m\_axis\_mm2s\_tdata[63:32] |
| CNT | 32 | ECOS求解器调用计数  m\_axis\_mm2s\_tdata[95:64] |
| ITER\_NUM | 32 | 一次求解内部迭代次数  m\_axis\_mm2s\_tdata[127:96] |
| **2**  参数帧格式 | Data | 64 | 双精度Eps数据，m\_axis\_mm2s\_tdata[63:32] |
| Data | 64 | 双精度Delta数据，m\_axis\_mm2s\_tdata[127:64] |
| 3  数据帧格式0x01 | Row | 32 | 行地址m\_axis\_mm2s\_tdata[31:0] |
| Col | 32 | 列地址m\_axis\_mm2s\_tdata[63:32] |
| Data | 64 | 双精度数据m\_axis\_mm2s\_tdata[127:64] |
| 4  数据帧格式0x02 | Col | 32bit | 整型数据，列积分/Sign信息，第0，4，8，12，…4N-4列数据信息，m\_axis\_mm2s\_tdata[31:0] |
| 32bit | 整型数据，列积分/Sign信息，第1，5，9，13，…4N-3列数据信息，m\_axis\_mm2s\_tdata[63:32] |
| 32bit | 整型数据，列积分/Sign信息，第2，6，10，14，…4N-2列数据信息， m\_axis\_mm2s\_tdata[95:64] |
| 32bit | 整型数据，列积分/Sign信息，第3，7，11，15，…4N-1列数据信息，m\_axis\_mm2s\_tdata[127:96] |
| 5  数据帧格式  0x03 | Data | 64bit | 双精度数据，当前b向量奇数数据信息，第0，2，4，6，…2N-2数据信息， m\_axis\_mm2s\_tdata[63:0] |
| 64bit | 双精度数据，当前b向量偶数数据信息，第1，3，5，7，…2N-1数据信息，m\_axis\_mm2s\_tdata[127:64] |
| 5  数据帧格式  0x04 | Data | 64bit | 双精度数据，当前b0向量数据信息，第0，1，2，3，4，5，6，…N-1数据信息， m\_axis\_mm2s\_tdata[63:0] |
| 64bit | 双精度数据，当前b1向量数据信息，第0，1，2，3，4，5，6，…N-1数据信息，m\_axis\_mm2s\_tdata[127:64] |

注

1. AXI-Stream总线当有效数据长度不满足128bit的整数倍时，**根据tlast和tkeep信号，完成最后有效数据寄存，具体参考AXI\_stream总线手册；**
2. 参数帧格式0x01：计算LDL处理时才存在，FRAME\_ID=0XA0~0XA3，命令帧后加入eps和delta数据用于LDL计算处理**；**
3. 数据帧格式0x01：矩阵数据传输。矩阵有效数据的传输暂定按照**row/col/val**的顺序进行，如果需要进一步提升带宽，这部分通讯协议再更新处理，（这部分应该在HP总线修改为ACP后进行），矩阵分别为
   1. FRAME\_ID=0XA0~0XA3，PS发送矩阵A至PL使其进行LDL分解；
   2. FRAME\_ID=0X1A0~0X1A3，PS发送矩阵A至PL使其进行LDL分解，请求PL返回对应的LD分解矩阵；
   3. FRAME\_ID=0XC2~0XC5，PS发送PS的LD分解结果至PL；
4. 数据帧格式0x2：向量数据传输。PS发送进行LDL分解时需要参数sign值至PL和PS发送矩阵A和AT的列积分信息至PL。
   1. FRAME\_ID=0XA8，PS发送进行LDL分解时需要参数sign值至PL；
   2. FRAME\_ID=0XC0~0XC1，PL和PS发送矩阵A和AT的列积分信息至PL；
   3. 注：列积分信息说明：L矩阵列积分数据总量为N+1列，第一列数据始终为0，L矩阵将对角线元素设置为0，L矩阵实际解码过程中需差分后加1处理（现有处理方式为在proproc.c代码调用一次LDL\_numeric2后使用分解后的L和转置后的Lt矩阵中的jc进行传输，后续可以看下LDL.c代码中关于LDL\_Symbolic函数中的处理，该函数运行过程中可以得到L的列积分信息，争取可以简单修改下提前获取L的行积分信息）。
5. 数据帧格式0x03：向量数据传输。PS发送进行AX=b求解时需要参数b值至PL
   1. FRAME\_ID=0XB0/0XB1，PS发送**初始求解**计算b值（b1或b2）至PL;
   2. FRAME\_ID=0X1B0/0X1B1，PS发送**初始求解**计算b值（b1或b2）至PL，并请求PL返回中间过程数据；
   3. FRAME\_ID=0XB2，PS发送**初始求解并行**计算2次b值（b1和b2）至PL;
   4. FRAME\_ID=0XB8/0xB9/0XBB，PS发送**迭代**求解计算3次b（b1或b2或b3）值至PL;
   5. FRAME\_ID=0X1B8/0x1B9/0X1BB，PS发送**迭代**求解计算3次b（b1或b2或b3）值至PL并请求PL返回中间过程数据；
   6. FRAME\_ID=0XBA，PS发送**迭代求解并行**计算2次b值（b1和b2）至PL;
   7. 注：发送1个b数据时，PS将数据b[2n]和b[2n+1]完成拼接发送至PL，2个b值发送按照并行方式进行，b0和b1拼接方式进行，（b0[0],b1[0]）…..（b0[N-1], b1[N-1]）,拼接为128位传输。

## 8.3 PL->PS数据传输协议

**PL->PS传输帧协议**

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **名称** | **位宽** | **含义** |
| 1  命令帧格式 | FRAME\_ID | 32 | **（调试信息）**  0XD0A0—初始处理A矩阵分解后L+D矩阵信息；  0XD0A1—初始处理AT矩阵分解后L+D矩阵信息；  0XD0A2—迭代处理A矩阵分解后L+D矩阵信息；  0XD0A3—迭代处理AT矩阵分解后L+D矩阵信息；  0XD1B0—初始处理求解b1过程中FX/BX数据；  0XD1B1—初始处理求解b2过程中FX/BX数据；  0XD1B8—迭代处理求解b1过程中FX/BX数据；  0XD1B9—迭代处理求解b2过程中FX/BX数据；  0XD1BB—迭代处理求解b3过程中FX/BX数据；  0XF0B0 —初始处理求解b1向量；  0XF0B1—初始处理求解b2向量；  0XF0B2—初始处理**并行**求解b1和b2向量；  0XF0B8—迭代处理求解b1向量；  0XF0B9—迭代处理求解b2向量；  0XF0BA—迭代处理**并行**求解b1和b2向量；  0XF0BB—迭代处理求解b3向量；  s\_axis\_tdata[31:0] |
| LENGTH | 32 | 命令长度 8bit字节为基本计数单位；  s\_axis\_tdata[63:32] |
| CNT | 32 | ECOS求解器调用计数  s\_axis\_tdata[95:64] |
| ITER\_NUM | 32 | 一次求解内部迭代次数  s\_axis\_tdata[127:96] |
| 2  数据帧格式0x11 | Row | 32 | 行地址s\_axis\_tdata[31:0] |
| Col | 32 | 列地址s\_axis\_tdata[63:32] |
| Data | 64 | 双精度数据s\_axis\_tdata[127:64] |
| 3  数据帧格式  0x12 | Data | 64bit | 双精度数据，求解前代计算结果Fx/Bx  向量数据信息s\_axis\_tdata[63:0] |
| Data | 64bit | 双精度数据，求解回代计算结果Fx/Bx向量数据信息s\_axis\_tdata[63:0] |
| 4  数据帧格式  0x13 | Data | 64bit | 双精度数据，求解结果 x向量数据信息，第0，2，4，6，…2N-2个数据信息，s\_axis\_tdata[63:0] |
| Data | 64bit | 双精度数据，求解结果x向量数据信息，第1，3，5，7，…2N-1个数据信息，s\_axis\_tdata[127:64] |
| 5  数据帧格式  0x14 | Data | 64bit | 双精度数据，求解结果x1向量数据信息s\_axis\_tdata[63:0] |
| Data | 64bit | 双精度数据，求解结果x2向量数据信息s\_axis\_tdata[63:0] |

注：

1. 数据帧格式0x11：调试帧PL发送矩阵A的LDL分解结果；
   1. FRAME\_ID=0xD0A0~0xD0A3，矩阵有效数据的传输暂定按照**row/col/val**的顺序进行，时有效。
2. 数据帧格式0x12：调试帧PL发送前代fx和回代bx处理结果；
   1. FRAME\_ID=0XD1B0/0XD1B1/0XD1B8/0XD1B9/0XD1BB，fx和bx的数据发送按照串行方式进行，先发送fx，再发送bx，fx[0], fx [1]….. fx [N-1], bx [0], bx [1]….. bx [N-1],拼接为128位传输。当N为奇数时，最后一个fx[N-1]和bx[0]拼接一个128bit进行传输。后续数据为bx[2n+1]和bx[2n]完成拼接发送。
3. 数据帧格式0x13：PL发送线性方程组x求解结果；
   1. RAME\_ID=0XF1B0/0XF1B1/0XF1B8/0XF1B9/0XF1BB，x数据发送按照串行方式进行，x[2n]和x[2n+1]拼接为128位传输。
4. 数据帧格式0x14：PL发送线性方程组两次并行计算的x结果，
5. RAME\_ID=0XF0B2/0XF0BA，2个x值发送按照并行方式进行，x0和x1拼接方式进行，（x0[0],x1[0]）…..（x0[N-1], x1[N-1]）,拼接为128位传输。

注：当有效数据长度不满足128bit的整数倍时，**根据tlast和tkeep信号，完成最后有效数据寄存，**比如格式数据帧格式2：当N为奇数时，最后一个x[2n+1]在低位字节，并将进行传输。后续数据为bx[2n+1]和bx[2n]完成拼接发送。信号m\_axis\_mm2s\_tkeep置为0x00FF

# 9 PS-PL间寄存器通讯协议

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **地址序号** | **名称** | **方向** | **位宽** | **含义** |
| 0 | Soft\_rst | 输入 | 32 | 55 – 复位处理 |
| 1 | Mode | 输入 | 32 | 55 – 测试模式  AA – 正常工作模式 |
| 2 | Reg\_nk | 输入 | 32 | 矩阵维度 |
| 4 | Reg\_LD\_nz | 输出 | 32 | LD矩阵非零值 |
| 5 | PL\_Date\_1 | 输出 | 32 | 年XXXX |
| 6 | PL\_Date\_2 | 输出 | 32 | 月日XX XX |
| 7 | PL\_Version | 输出 | 32 | 年XXXX 0100开始 |