一、CDPR Matlab App 中的全局变量和类说明:

1.基本机械参数与不同模块的类

1.1 动锚点座类`CdprMovAnchor`

```
CdprMovAnchor MovAnchor
```

`MovAnchor.positon_origin_G`：3X8,global origin position,unit:m

`MovAnchor.positon_G`：3X8矩阵,全局坐标中的锚点座实时位置（绝对形式）,unit:m

`MovAnchor.positon_L`：1X8矩阵，锚点座局部一维坐标中的锚点座实时位置（增量形式）,unit:m

`MovAnchor.positon_LB`：8X2, local position boundary ,unit:m

`MovAnchor.dir_G`：3X8矩阵, 元素为0,1,-1，每列分别对应锚点座的局部坐标相对全局坐标系移动的正方向，unit:mm

```
%更新MovAnchor.positon_G的方式
MovAnchor.positon_G = MovAnchor.positon_G + MovAnchor.dir_G .*
MovAnchor.positon_L;
```

`MovAnchor.AnRotz_G`：1X8,锚点座坐标系相对全局坐标系对应的Z轴转角（自定义），unit：rad
`MovAnchor.PulRotz_L`：1X8,滑轮相对应锚点座坐标系的转角（逆时针为正），unit：rad
`MovAnchor.PulCenter`：3X8,滑轮圆心在全局坐标系的坐标，unit：m
`MovAnchor.caliPosition_L`：1X8，锚点校准位置处的局部坐标（为其全局坐标中可变的坐标分量）
`MovAnchor.endPosition_L`：1X8，锚点极限位置处的局部坐标（为其全局坐标中可变的坐标分量）
`MovAnchor.caliPosition_G`：3X8，锚点校准位置处的全局坐标
`MovAnchor.endPosition_G`：3X8，锚点极限位置处的全局坐标

1.2 末端动平台类`CdprPlatform`

```
CdprPlatform Platform
```

`Platform.anchorPosition_P`：3X8，平台锚点的局部坐标
`Platform.anchorPosition_G`：3X8，平台锚点的全局坐标

`Platform.anchorDis`：1X28，动平台上两两锚点间的距离，访问索引：
$index=(j-i)+\frac{[(8-1)+(8-i+1)]\times(i-1)}{2}\ (j>i,\ 且 i=1,\dots,7)$

`Platform.anchorUniVec_P`：3X28，动平台上两两锚点间的单位向量（P坐标系）$i->j$且$i<j$，访问索引：$index=(j-i)+\frac{[(8-1)+(8-i+1)]\times(i-1)}{2}\ (j>i,\ 且 i=1,\dots,7)$

`Platform.pose_G`：6X1, 实时广义坐标（中心点全局坐标+ZXY欧拉角）
`Platform.boundary`：3X2, 动平台全局坐标xyz分量的最大最小值（运动边界）

`Platform.poDistance`：设有n个障碍物，数组大小为1Xn，表示动平台与障碍物边界框的最短距离

```
%索引表达式，索引第i根绳与第j根绳间距
index = (j-i) + (7+(9-i))*(i-1)/2;
```

1.3 绳缆类`CdprCable`

```
CdprCable Cable
```

`Cable.E`：young's modulus, Pa
`Cable.A`：绳索横截面积，m^2
`Cable.K`：绳索的实时刚度（随长度变化）
`Cable.Length`：绳索的实时伸长长度
`Cable.UniVector`：绳索实时拉力方向单位向量（G系）

`Cable.UniVector_P`：绳索实时拉力方向单位向量（P系）

`Cable.TanLinVector`：从动平台锚点到滑轮切点的向量（G系，带长度与方向）

`Cable.TanLinVector_P`：从动平台锚点到滑轮切点的向量（P系，带长度与方向）

`Cable.voronoi`：2X8元胞数组，第i列表示第i个绳向量所在的voronoi region的类型和对应元素代号

`Cable.cpDistance`：1X8数组，以绳索向量与其voronoi region最近元素的夹角正弦值定义距离，为零则干涉，需要与安全距离比较，在点域时距离设为1（最安全）

`Cable.ccDistance`：1X28数组，记录绳两两之间的距离，访问索引：
$index = (j-i) + \frac{[(8-1)+(8-i+1)]\times(i-1)}{2}\ (j>i,\ 且 i=1,\dots,7)$

`Cable.coDistance`：设有n个障碍物，数组大小为nX8，每列为对应绳索与n个障碍物的距离。

`Cable.ccDis_min`：允许的最小绳索间距离

`Cable.ccDis_Intersect`：若当前步的绳索间距离小于`ccDis_Intersect`则下一步对两绳索进行交叉干涉检查

`Cable.cpDis_min`：允许的最小绳索与动平台间距离

`Cable. tensionDistrib`： %8X1，绳实时张力分布

1.4 障碍物边界框类`ObsBB`（AABB）

```
classdef ObsBB
    % obstacle boundind box 障碍物边界框类
    %   Detailed explanation goes here

    properties (Access = public)
        vertices;  %3X8, 存储顶点位置向量
        origin;  %3X1, 原点位置
        R0;  %包络球半径
        Rs;  %检测球半径
        ds;  %安全距离
        d_min; %最小允许距离
    end
```

1.5 其他机械参数

`R_encoderPully`：外置编码器连接滑轮的半径，m
`R_anchorPully`：锚点座出绳滑轮的半径，m
`S_leadScrew`：轨道丝杆的导程，m
`encoderAngle`：1X8 8个外置编码器的实时绝对转角，rad
`railMotorAngle`：1X8 8个轨道驱动电机编码器实时绝对转角，rad

`A_T`：6X8结构矩阵（全局坐标系） structur

2.用于轨迹规划的类与变量

2.1 自定义参数

`startPose`： 6X1 开始位姿
`endPose`： 6X1 结束位姿
`startTime`：开始时间
`endTime`：结束时间
`timeStep`：计算时间步长
`trajPlanMode`：线性的规划模式选择

2.2 保存末端平台轨迹规划信息的类`PlatformTraj`

`platformTraj.maxVelBound`：规定的不许超出的最大速度（自定义）
`platformTraj.maxAccelBound`：规定的不许超出的最大加速度（自定义）
`platformTraj.pos`：3XN，N个点的位置
`platformTraj.eular`：3XN，N个点的ZXY欧拉角
`platformTraj.vel`：3XN，N个点三个方向的速度
`platformTraj.accel`：3XN，N个点三个方向的加速度
`platformTraj.magVel`：1XN，N个点的速度大小（绝对值）
`platformTraj.magAccel`：1XN，N个点的加速度大小（绝对值）
`platformTraj.maxVel`：轨迹上的最大速度
`platformTraj.maxAccel`：轨迹上的最大加速度

2.3 保存动锚点座轨迹规划信息的类`AnchorTraj`

```
classdef AnchorTraj
    % 保存动锚点单次轨迹规划信息的类
    %   Detailed explanation goes here

    properties (Access = public)
        optiFreq; %执行优化算法的频率
        maxAverVelBound; %单步最大允许频率
        maxStepBound; %单步最大允许长度
        maxStep; %由允许量综合计算得到的最大单步长度
        maxAverVel; %由允许量综合计算得到的最大单步平均速度
        pos; %1X8, 轨迹上各锚点位置
        averVel; %1X8,轨迹上各锚点平均速度
    end
```

2.4 保存绳索轨迹规划点上信息的类`CableTraj`

```
classdef CableTraj
    % 保存绳索轨迹规划点上信息的类
    %   Detailed explanation goes here

    properties (Access = public)
        cableLength; %8xn, 绳长
        cableTension; %8xn,绳拉力
    end
```

2.5 CDPR总体轨迹规划信息类`CdprTrajInfo`

`trajInfo.platformTraj`：末端平台轨迹规划信息对象

`trajInfo.anchorTraj`：动锚点轨迹规划信息对象

`trajInfo.cableTraj`：绳索在轨迹规划点上信息的对象

`trajInfo.time`：1XN，时间点序列
`trajInfo.Numer`：规划点的数量

`trajInfo.timeStep`：规划时间步

3.末端动平台凸包络拓扑信息

3.1 几何元素类

点

`Vertex`

```
classdef Vertex
    properties
        n_verticesId; %相邻点的index
        n_edgesId; %相邻边的index
        n_facesId; %相邻面的index
        localEdgeVector; %3X4,局部单元中边的单位向量
        coordinate; %坐标
     end
end
```

边

`Edge`

```
classdef Edge
    properties
        n_verticesId; %相邻点的index
        n_edgesId; %相邻边的index
        n_facesId; %相邻面的index
        v_facesId; %点连接面的index
     end
end
```

面

`Face`

```
classdef Face
    properties
        n_verticesId; %相邻点的index
        n_edgesId; %相邻边的index
        n_facesId; %相邻面的index
        v_edgesId; %点连接边的index
        v_facesId; %点连接面的index
        normal_vec; %平面外单位法向量
     end
end
```

3.2 局部单元的表示

信息储存在`vertex`结构体中邻边、邻点、邻面的排序中（三个数组一一对应并按绕该点逆时针排列），并将该点
到各个邻点的单位向量存在`vertex.localEdgeVector`中

3.3 Voronoi Region的表示

`Cable.voroRegion`：2X8元胞数组，第i列表示第i个绳向量所在的voronoi region的类型和对应元素代号
类型：`'v'`-点，`'f'`-面，`'e'`-边，`C`-干涉

```
Cable.voroRegion=2×8 cell array
    {'class of voronoi region'}    ...  {'class of voronoi region'}
    {'index of the corresponding element'}    ...   {'index of the corresponding
element'}
```

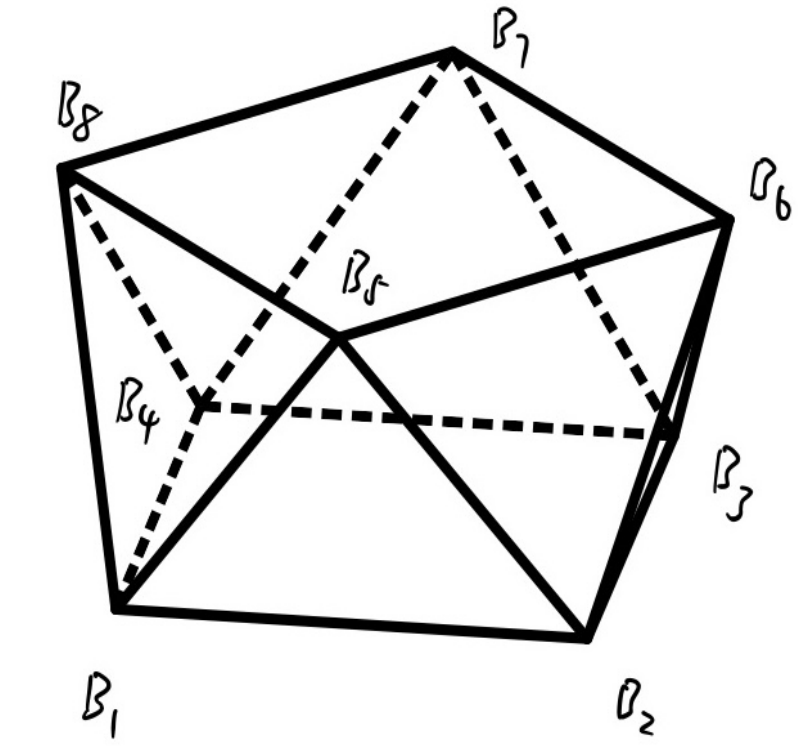
4.用于设置外力条件与可行力的类`WrenchSetting`

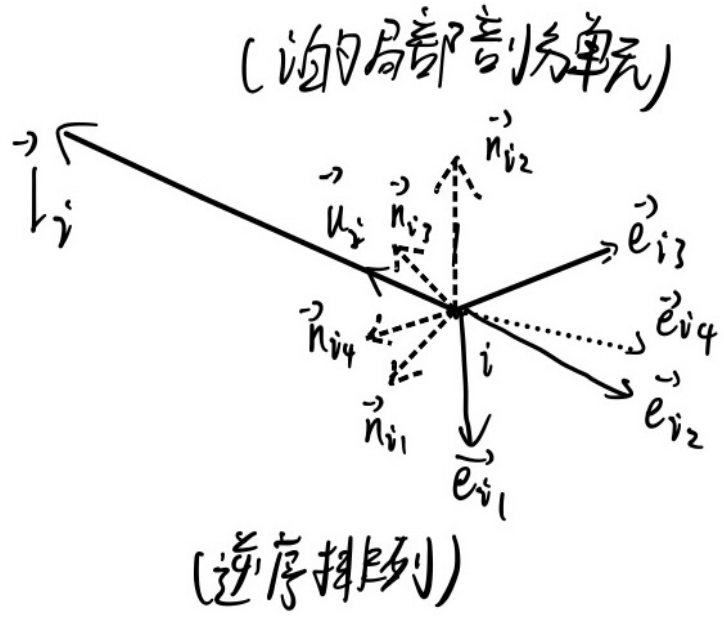
```
classdef WrenchSetting
    %设置外力条件与可行力的类
    properties (Access = public)
        wrench; %6X1，外力与外力矩
        feaspTensionRange; %8X2，张力上下限，(i,1)为第i根绳张力下限，(i,2)为第i根绳张力
上限
    end
```

二、绳索与动平台包络之间的碰撞检测

2.1 定义局部单元

如下图所示：





2.2 判定向量 $\vec{l}_i$ 所在的 $voronoi\ region$

2.2.1 在顶点域$i(vertex\ region)$的情况

$$\vec{u}_i \cdot \vec{e}_{ij} \leqslant 0\,(i=1,\ldots,8 j=1,\ldots,4)$$

2.2.2 在边域$\vec{e}_{ij}(edge\ region)$的情况

$$\begin{cases} \vec{e}_{ij} \cdot \vec{u}_i > 0 \\ \vec{u}_i \cdot (\vec{e}_{ij} \times \vec{n}_{ij}) = [\vec{e}_{ij}\ \vec{n}_{ij}\ \vec{u}_i] \geqslant 0 \\ \vec{u}_i \cdot (\vec{e}_{ij} \times \vec{n}_{ij-1}) = [\vec{e}_{ij}\ \vec{n}_{ij-1}\ \vec{u}_i] \leqslant 0 \end{cases}$$

2.2.3 在面域$\vec{n}_{ij}(face\ region)$的情况

$$\begin{cases} \vec{n}_{ij} \cdot \vec{u}_i > 0 \\ \vec{u}_i \cdot (\vec{e}_{ij} \times \vec{n}_{ij}) = [\vec{e}_{ij}\ \vec{n}_{ij}\ \vec{u}_i] < 0 \\ \vec{u}_i \cdot (\vec{e}_{ij+1} \times \vec{n}_{ij}) = [\vec{e}_{ij+1}\ \vec{n}_{ij}\ \vec{u}_i] > 0 \end{cases}$$

2.3 计算向量与其所在域元素的夹角（即距离）

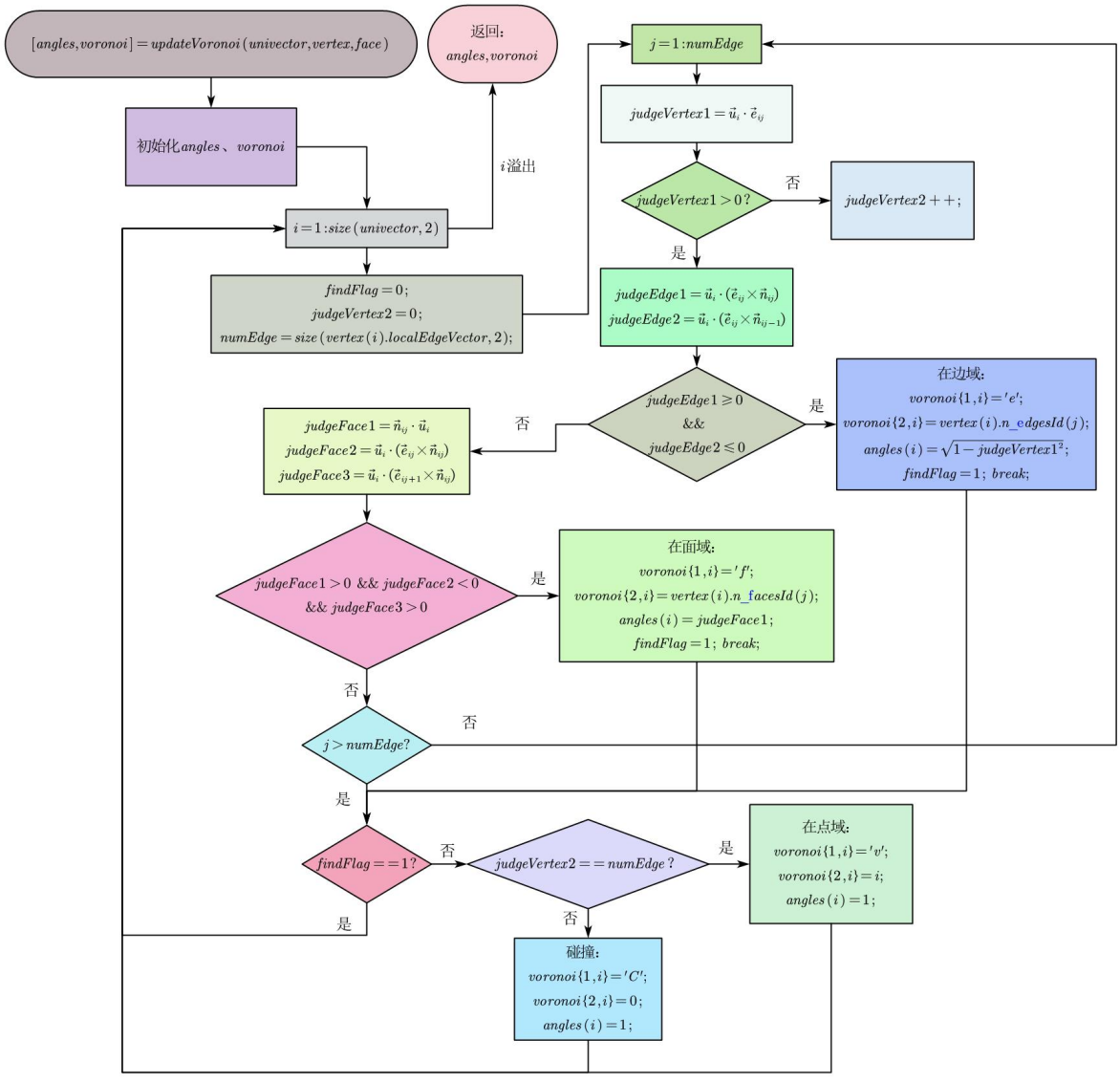
设定安全角度θ_s $(0 < \theta_s < 90^\circ)$

（1） 若在点域，设角度$\theta_s = 90^\circ$

（2） 若在边域，$\theta_i = \arccos\left(\vec{e}_{ij} \cdot^P \vec{u}_i\right)$

（3） 若在面域，$\theta_i = 90^\circ - \arccos\left(\vec{n}_{ij} \cdot^P \vec{u}_i\right)$

2.4 程序流程图



三、绳索之间的碰撞检测

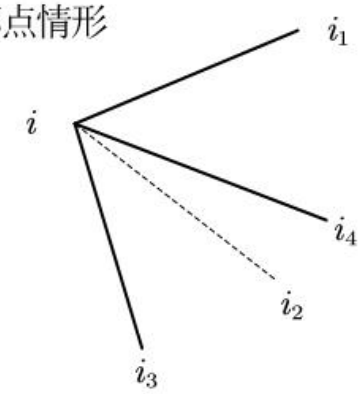
绳索之间的碰撞检测包含距离计算、短距离大步长内的干涉检查（CGAL库实现）。

3.1 距离计算原理说明

对于正常计算两绳索的距离需要计算$C_8^2 = 28$次，这里程序采用CGAL库的函数计算两线段间的最短距离。为了减少执行次数，程序利用前面计算的voronoi region的信息以及绳索向量两起点间距离不变的性质进行优化，在一些情况中简化计算，其原理如下图所示。

情形1： $ccDis = l_{ij}$

邻点情形



充要条件： $\vec{u}_i \cdot \vec{u}_{ij} \leqslant 0$ && $\vec{u}_j \cdot \vec{u}_{ji} \leqslant 0$

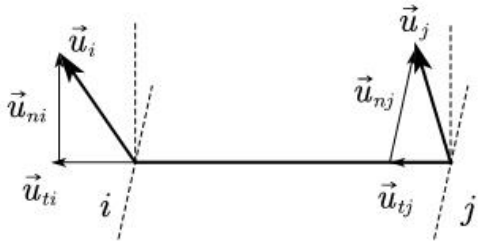
利用$voronoi\ region$：

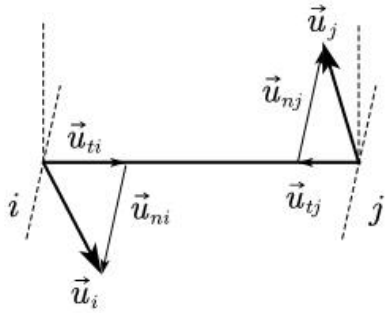
(1)当i、j均在点域时$ccDis = l_{ij}$

(2)当i、j其中一个在点域时(假设i在点域)，且另一个(j)在不与i相邻的边域或面域，则判断$a = \vec{u}_j \cdot \vec{u}_{ji}$，若$a > 0$则$ccDis = \sqrt{1-a^2}\, l_{ij}$

否则$ccDis = l_{ij}$

情形2： $ccDis = al_{ij}$





计算： $a_1 = \vec{u}_i \cdot \vec{u}_{ij};\ a_2 = \vec{u}_j \cdot \vec{u}_{ji};\ \vec{u}_{ti} = a_1 \vec{u}_{ij};\ \vec{u}_{tj} = a_2 \vec{u}_{ji};$

$\vec{u}_{ni} = \vec{u}_i - \vec{u}_{ti};\ \vec{u}_{nj} = \vec{u}_j - \vec{u}_{tj};$

充分条件：

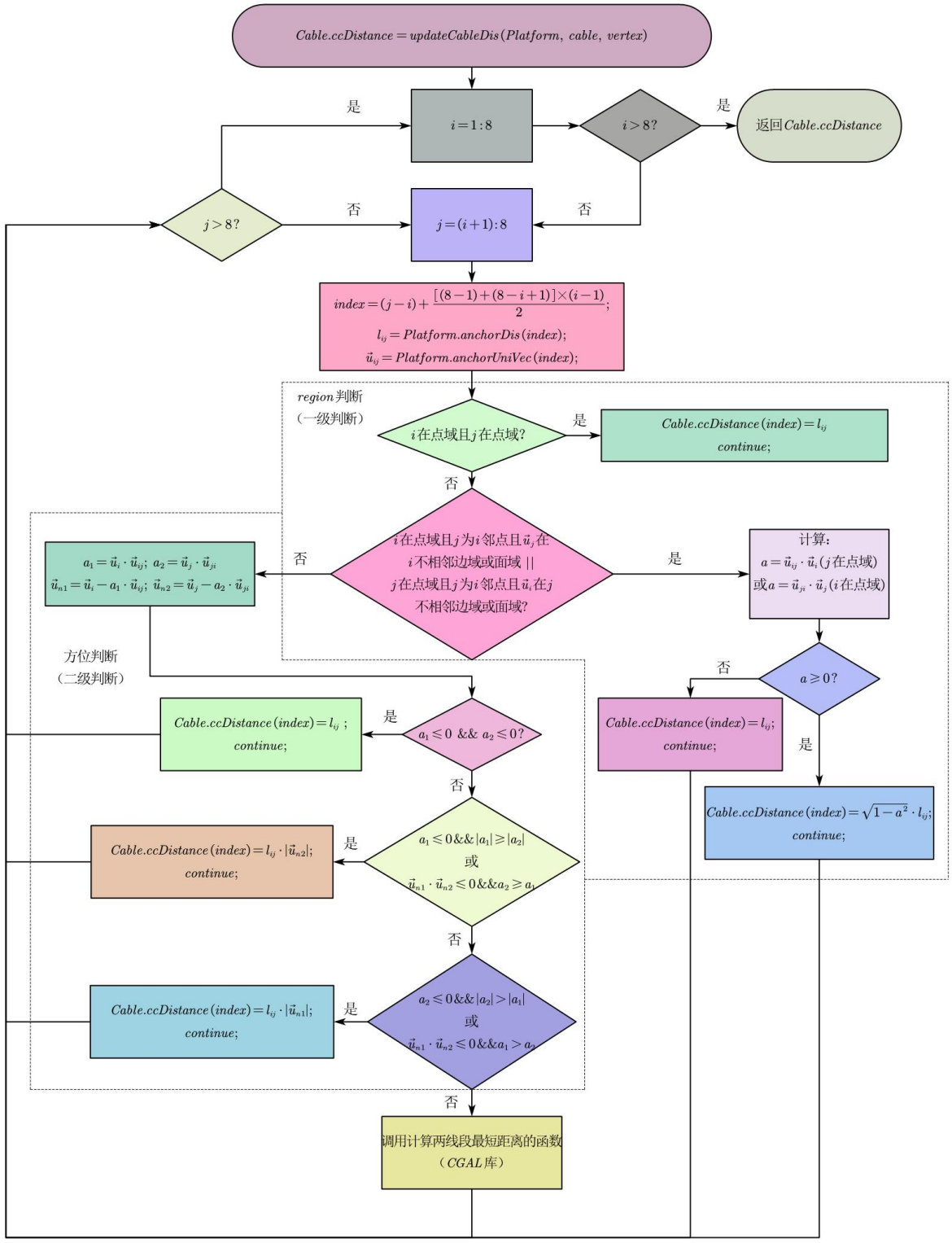
(1)$a_1 \leqslant 0$且$a_2 > 0$且$|a_1| \geqslant |a_2|$或$\vec{u}_{ni1} \cdot \vec{u}_{nj} \leqslant 0$且$a_2 \geqslant a_1$时： $ccDis = l_{ij} \cdot |\vec{u}_{nj}|$

(2)$a_2 \leqslant 0$且$a_1 > 0$且$|a_2| > |a_1|$或$\vec{u}_{ni} \cdot \vec{u}_{nj} \leqslant 0$且$a_1 > a_2$时： $ccDis = l_{ij} \cdot |\vec{u}_{ni}|$

3.2 程序流程图

3.2.1 距离计算程序流程图

距离计算程序流程图如下所示：



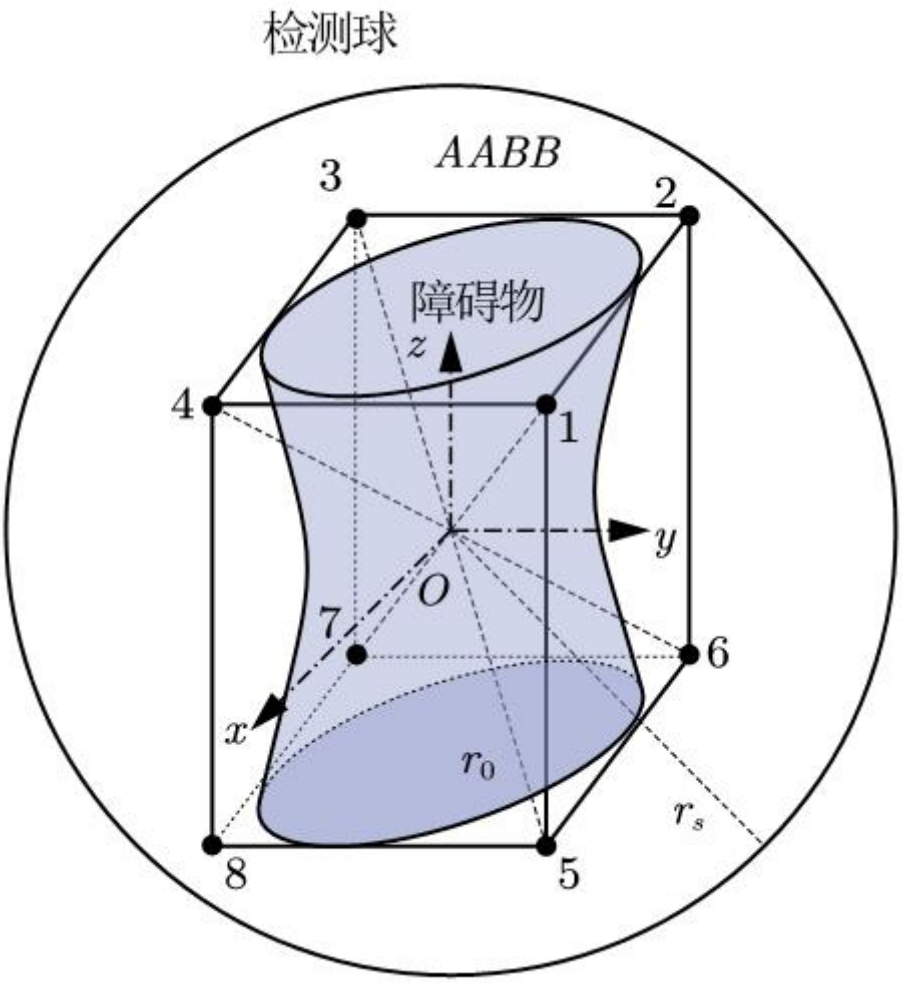
3.2.2 干涉检测程序

使用CGAL库，与MATLAB对接的MEX程序：`doIntersect_CGAL.mexw64`

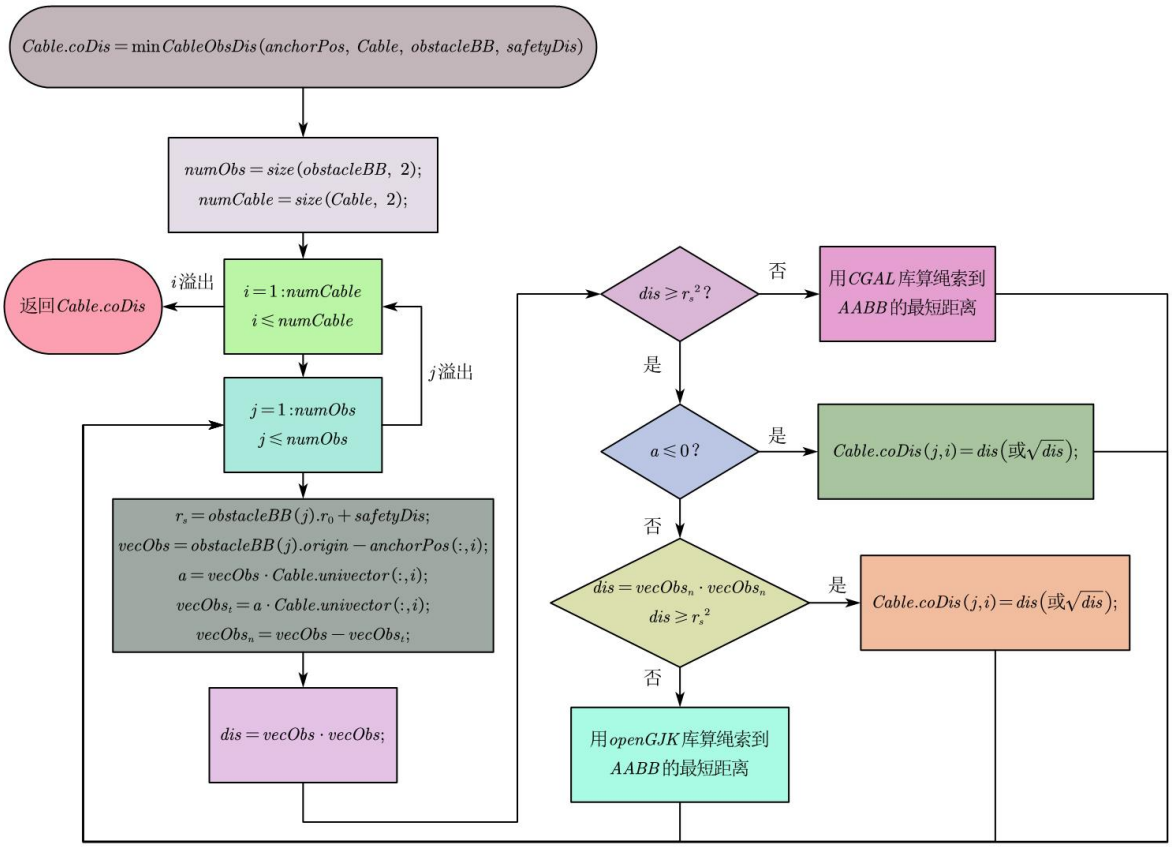
四、绳索与障碍物之间的碰撞检测

4.1 原理说明

障碍物由AABB(aligned axis bounding box)包络，设定一个检测球区域以简化计算，设AABB的外包络球半径为r_0，设定安全距离d_s，则检测球半径$r_s = r_0 + d_s$，当绳索在该检测球以外时则计算绳索与球心的距离（将该距离作为绳索到障碍物的距离），若该距离小于检测球半径（绳索在球内），则用CGAL库计算绳索到AABB的最短距离，设定一个允许的最小距离$d_{min}(d_s > d_{min})$，可设$d_s = (1.5 \sim 2)d_{min}$。原理图如下：



4.2 程序流程图



五、静态避障（末端轨迹已知）

5.1 程序说明

5.1.1 参数与变量

$\boldsymbol{P}_{traj}$：轨迹数据　　$timeSque$：轨迹时间序列　　k_{op}：优化频率，k_{op}个时间步长执行一次优化算法

$maxStep$：锚点轨迹$\boldsymbol{A}_{traj}$上相邻两个点分量的最大间隔

$anchorTrajId$：锚点轨迹点对应时间序列$timeSque$的索引信息

$\boldsymbol{A}_{traj}$：锚点轨迹点信息

5.2.2 优化目标与优化方案

目标1：力分布均匀，见附录1

方案1：绳索构型与分布力同时优化

（1）优化函数：$f_1 = \min \frac{\sum_{i=1}^{m} (f_i - \bar{f})^2}{m}$

（2）优化变量：8个动锚点步长+8个绳索力

方案2：先优化绳索构型，在力分布算法中再分配力

（1）优化函数：$f_1 = \min \left\| \sum_{i=1}^{8} \vec{u}_i \right\|_2^2$

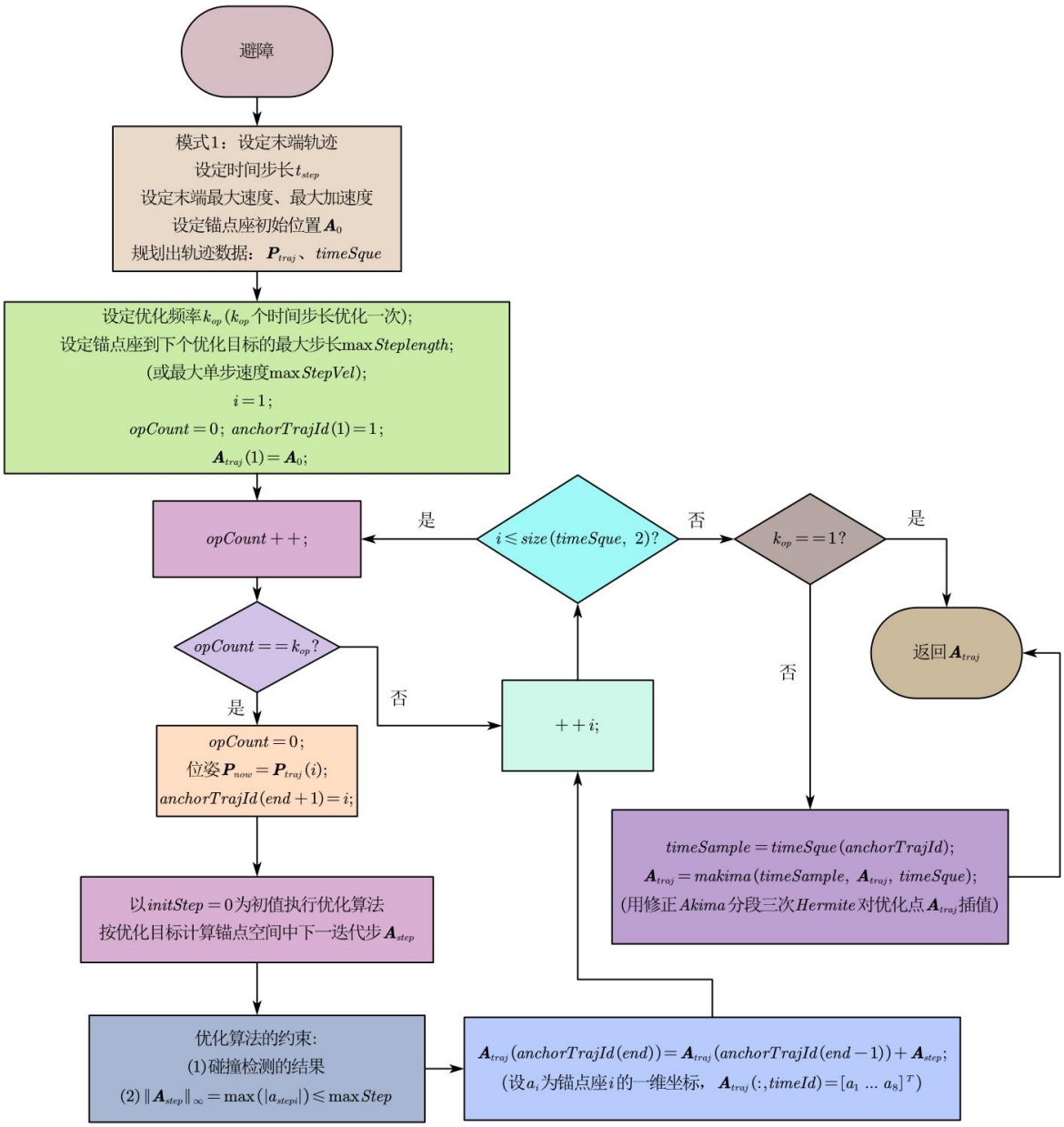
（2）优化变量：8个动锚点步长+8个绳索力

目标2：离障碍物最远

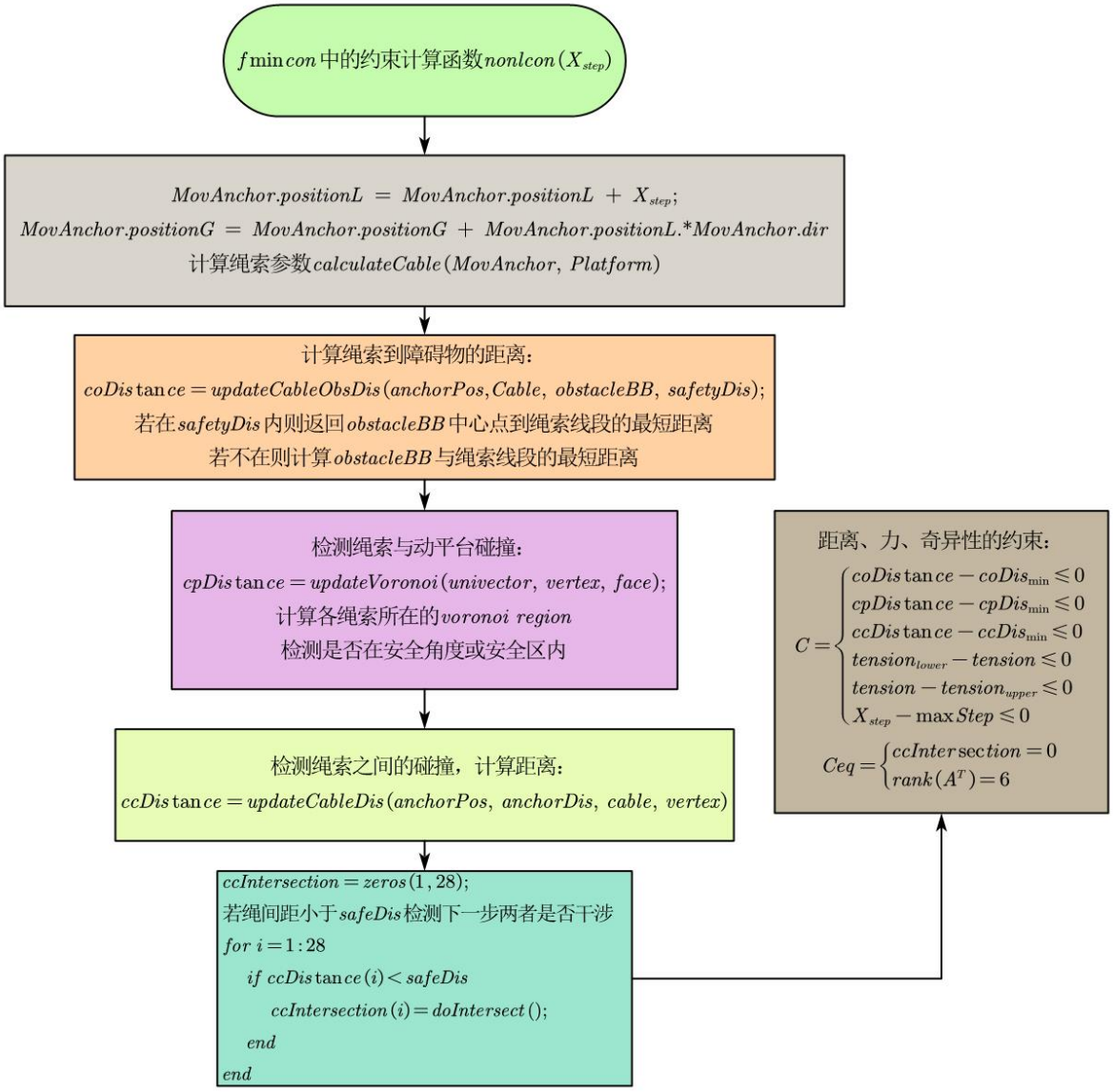
优化函数：$f_2 = \min(-\|Cable.coDistance\|_2)$

5.2 程序流程图

5.2.1 基于优化算法的主程序部分



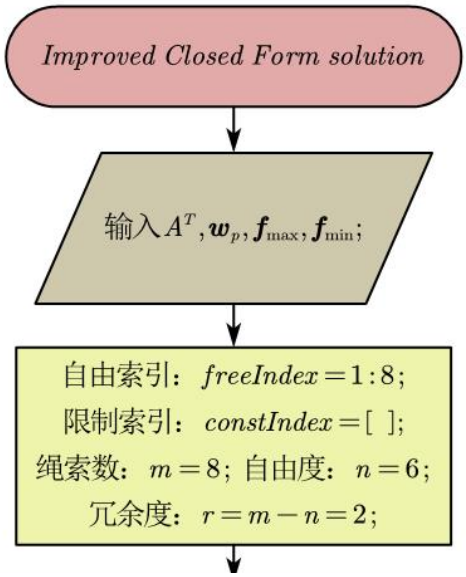
5.2.2 计算约束的部分

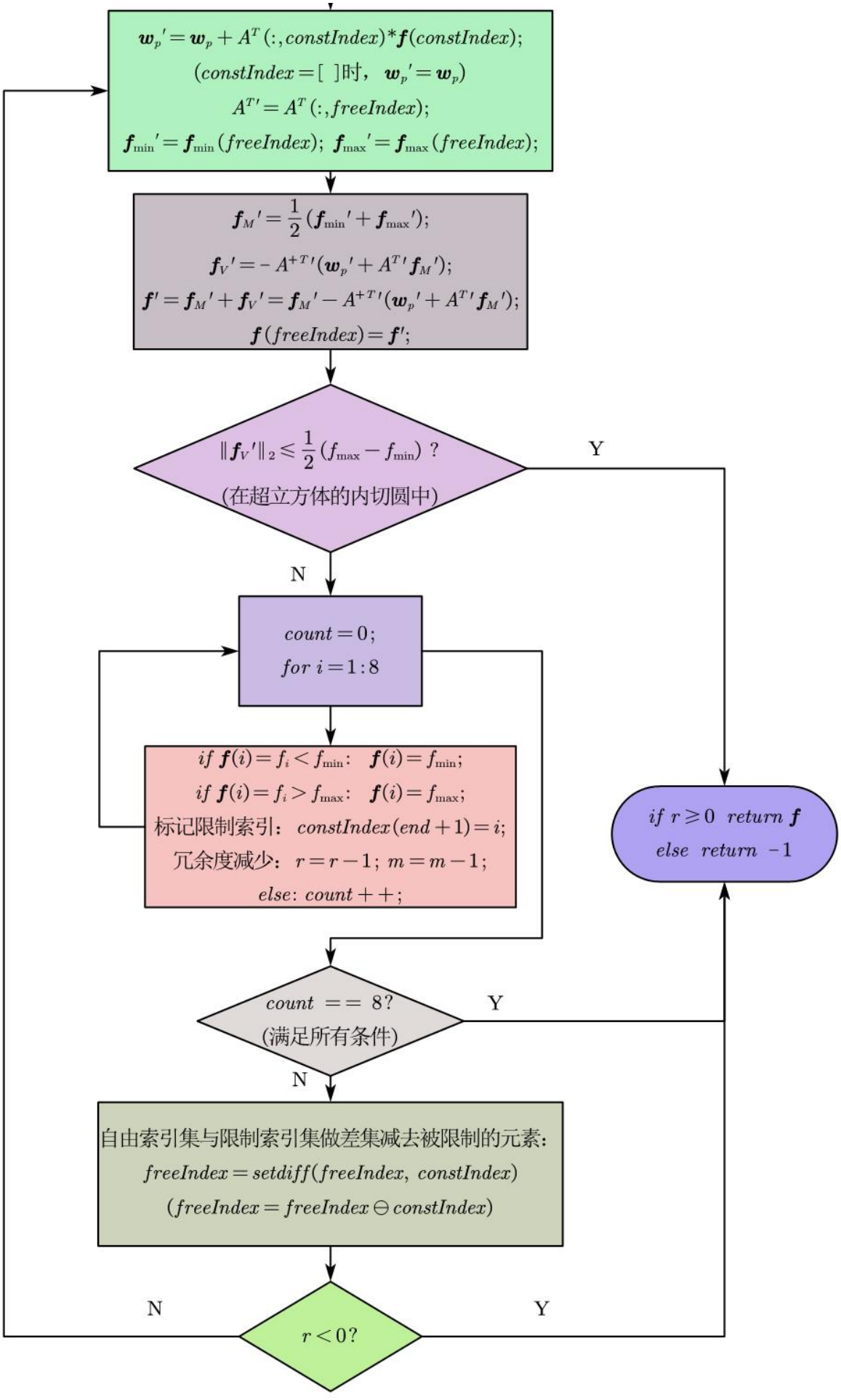


六、力分配算法

6.1 Improved Closed-Form solution

6.1.1 算法流程




$\boldsymbol{w}_p' = \boldsymbol{w}_p + A^T(:, constIndex) * \boldsymbol{f}(constIndex);$
$(constIndex = [\ ]$时，$\boldsymbol{w}_p' = \boldsymbol{w}_p)$
$A^{T\prime} = A^T(:, freeIndex);$
$\boldsymbol{f}_{\min}' = \boldsymbol{f}_{\min}(freeIndex);\ \boldsymbol{f}_{\max}' = \boldsymbol{f}_{\max}(freeIndex);$
$\boldsymbol{f}_M' = \frac{1}{2}(\boldsymbol{f}_{\min}' + \boldsymbol{f}_{\max}');$
$\boldsymbol{f}_V' = -A^{+T\prime}(\boldsymbol{w}_p' + A^{T\prime}\boldsymbol{f}_M');$
$\boldsymbol{f}' = \boldsymbol{f}_M' + \boldsymbol{f}_V' = \boldsymbol{f}_M' - A^{+T\prime}(\boldsymbol{w}_p' + A^{T\prime}\boldsymbol{f}_M');$
$\boldsymbol{f}(freeIndex) = \boldsymbol{f}';$
$\|\boldsymbol{f}_V'\|_2 \le \frac{1}{2}(f_{\max} - f_{\min})$?
(在超立方体的内切圆中)
Y
N
$count = 0;$
$for\ i = 1:8$
$if\ \boldsymbol{f}(i) = f_i < f_{\min}$: $\boldsymbol{f}(i) = f_{\min};$
$if\ \boldsymbol{f}(i) = f_i > f_{\max}$: $\boldsymbol{f}(i) = f_{\max};$
标记限制索引：$constIndex(end+1) = i;$
冗余度减少：$r = r - 1;\ m = m - 1;$
$else$: $count++;$
$if\ r \ge 0\ \ return\ \boldsymbol{f}$
$else\ \ return\ -1$
$count\ ==\ 8$?
(满足所有条件)
Y
N
自由索引集与限制索引集做差集减去被限制的元素：
$freeIndex = setdiff(freeIndex,\ constIndex)$
$(freeIndex = freeIndex \ominus constIndex)$
N
Y
$r < 0$?

6.1.2 引用

`Pott A. Cable-driven Parallel Robots: Theory and Application[M]. Switzerland: Springer International Publishing AG, 2018: 94-95`

附录

1.受力与结构矩阵分析

1.1 绳索拉力平衡方程

$$\begin{bmatrix} \overrightarrow{u_1} & \overrightarrow{u_2} & \cdots & \overrightarrow{u_8} \\ \overrightarrow{b_1}\times\overrightarrow{u_1} & \overrightarrow{b_2}\times\overrightarrow{u_2} & \cdots & \overrightarrow{b_8}\times\overrightarrow{u_8} \end{bmatrix} \begin{bmatrix} \overrightarrow{f_1} \\ \overrightarrow{f_2} \\ \vdots \\ \overrightarrow{f_8} \end{bmatrix} = -\vec{w}$$

其中定义结构矩阵$A^T = \begin{bmatrix} \overrightarrow{u_1} & \overrightarrow{u_2} & \cdots & \overrightarrow{u_8} \\ \overrightarrow{b_1}\times\overrightarrow{u_1} & \overrightarrow{b_2}\times\overrightarrow{u_2} & \cdots & \overrightarrow{b_8}\times\overrightarrow{u_8} \end{bmatrix}$，并令$\vec{f} = \begin{bmatrix} \overrightarrow{f_1} & \overrightarrow{f_2} & \cdots & \overrightarrow{f_8} \end{bmatrix}^T$

求通解：$\exists\lambda_H, \vec{f} = -A^{+T}\vec{w} + H\lambda_H$

1.2 MATLAB计算绳拉力

```
%% 力分布情况计算
%构造结构矩阵（structure matrix）
A = zeros(6,8);
for i=1:8
    bi = Platform_Next.anchorPosition_G(:, i) - Platform_Next.pose_G(1:3);
    A(:, i) = [Cable.UniVector(:, i); cross(bi,Cable.UniVector(:, i))];
end
%A_psedoInv = psedoInverse_svd(A); %计算A的伪逆（svd法，稳定性更好，较慢）
A_psedoInv = A'*inv(A'*A); %计算A的伪逆（计算逆时需要保证A'A非奇异）
H = null(A); %计算A的零空间（核空间）中的一个基
W =wrenchSetting.wrench; %外力与力矩
tension = - A_psedoInv * W + H*lamda; %计算张力分布向量
```