《机床数控技术》课程设计
说明书

班级：

学号：

姓名：BingoV

指导教师：

时间：2019 年 6 月

地点：

目录

一、课设任务

1. PL3——逐点比较法插补第二象限直线；
2. DC41——DDA 法插补第 1～2 象限逆圆弧。

二、课设要求

1. 具有数据输入界面，如输入直线插补的起点、终点，圆弧插补的起止点、圆心或半径、插补的步长等；
2. 具有插补过程的动态显示功能，如单步插补、连续插补等；
3. 插补的步长可调；
4. 直线的起点、圆弧的圆心在坐标系中的位置可变（即直线的起点、圆弧的圆心可不设定在坐标原点）；
5. 可以选择左移规格化和余数寄存器预置数的方法提高 DDA 法插补质量（两种功能都需要实现）；

要求使用 MATLAB 编程，并将程序输入数控插补硬件实验平台，实现仿真程序与实验平台插补运动的同步运行，通过电子绘图板绘制插补轨迹。

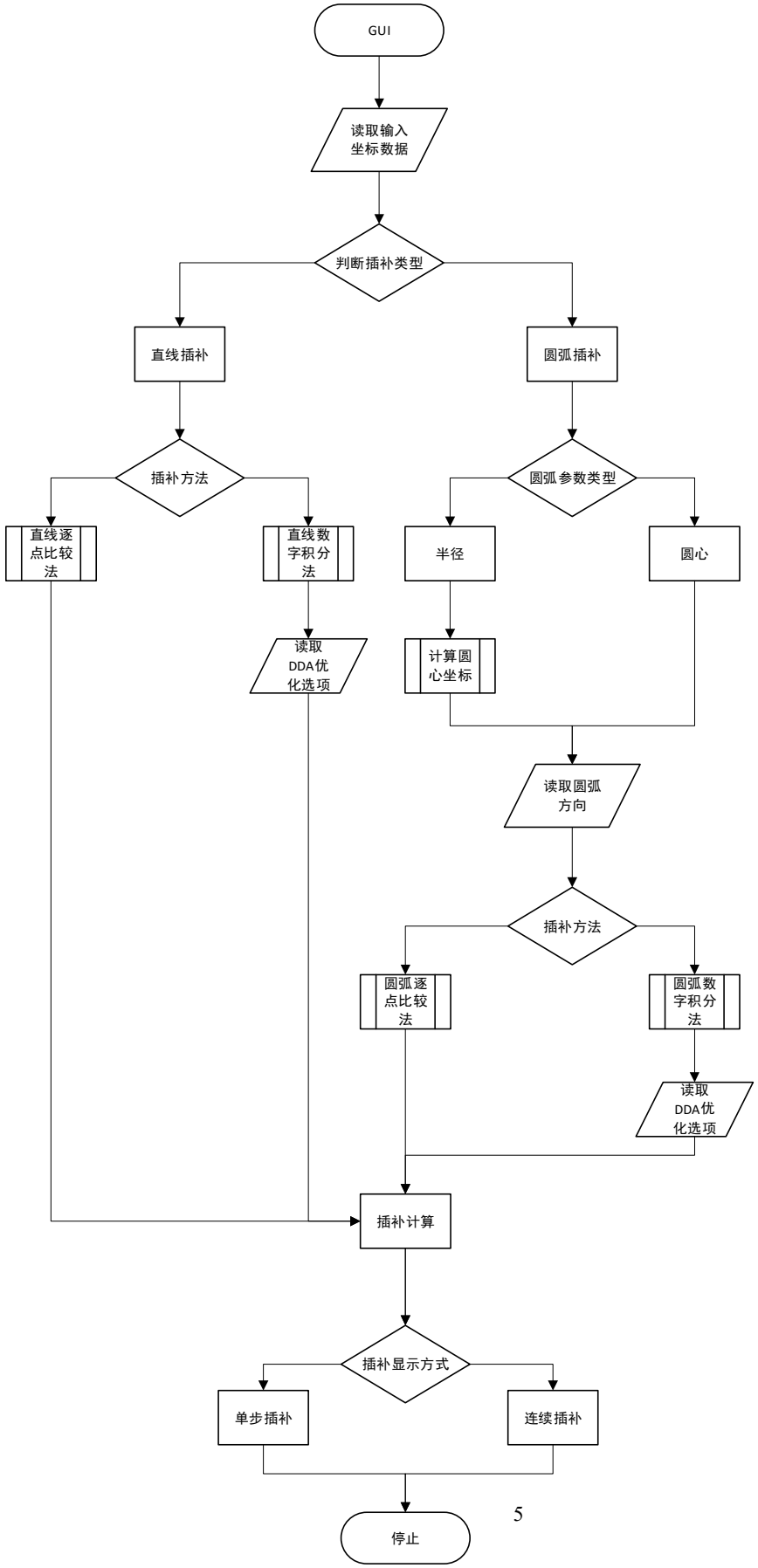
三、编程语言

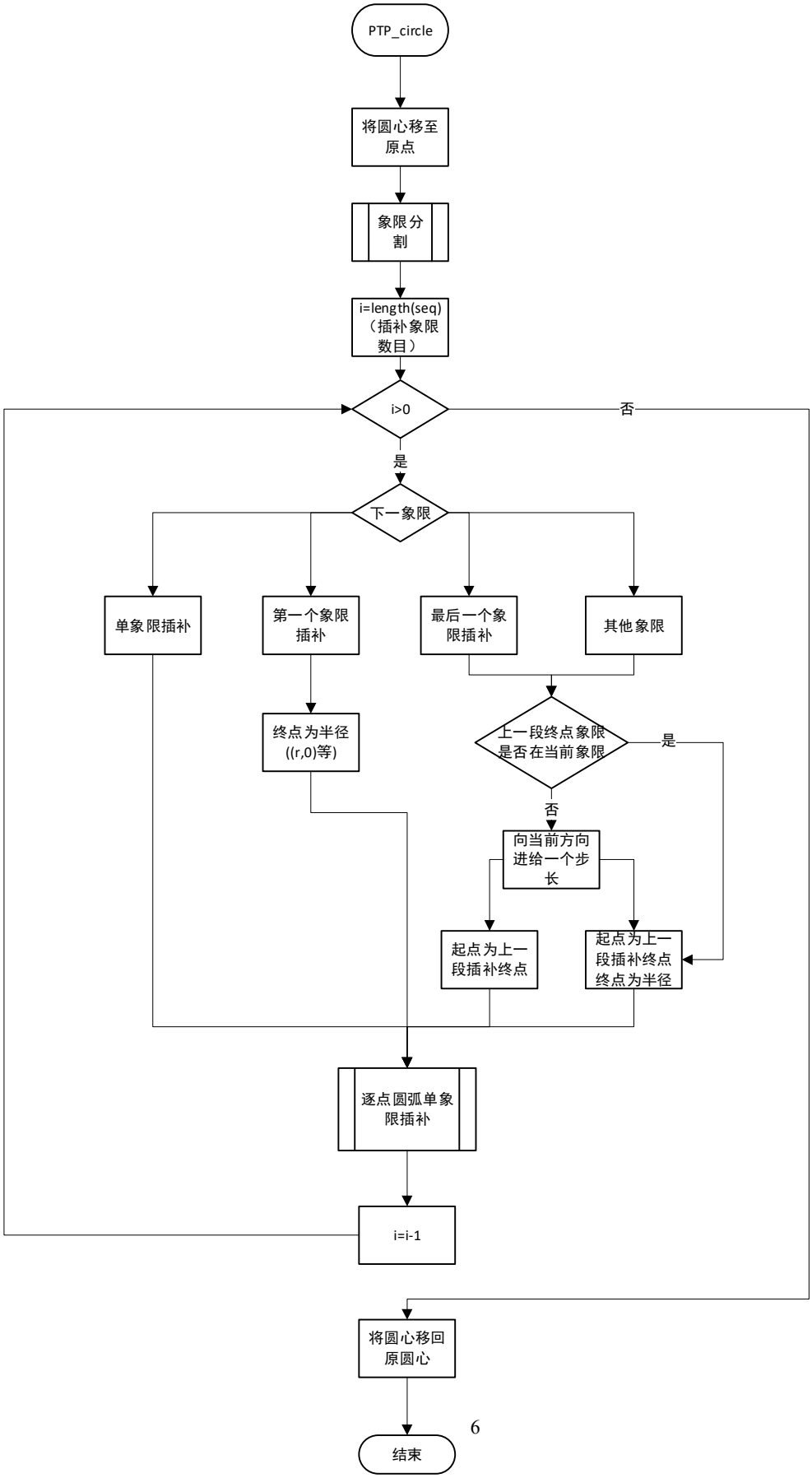
MATLAB

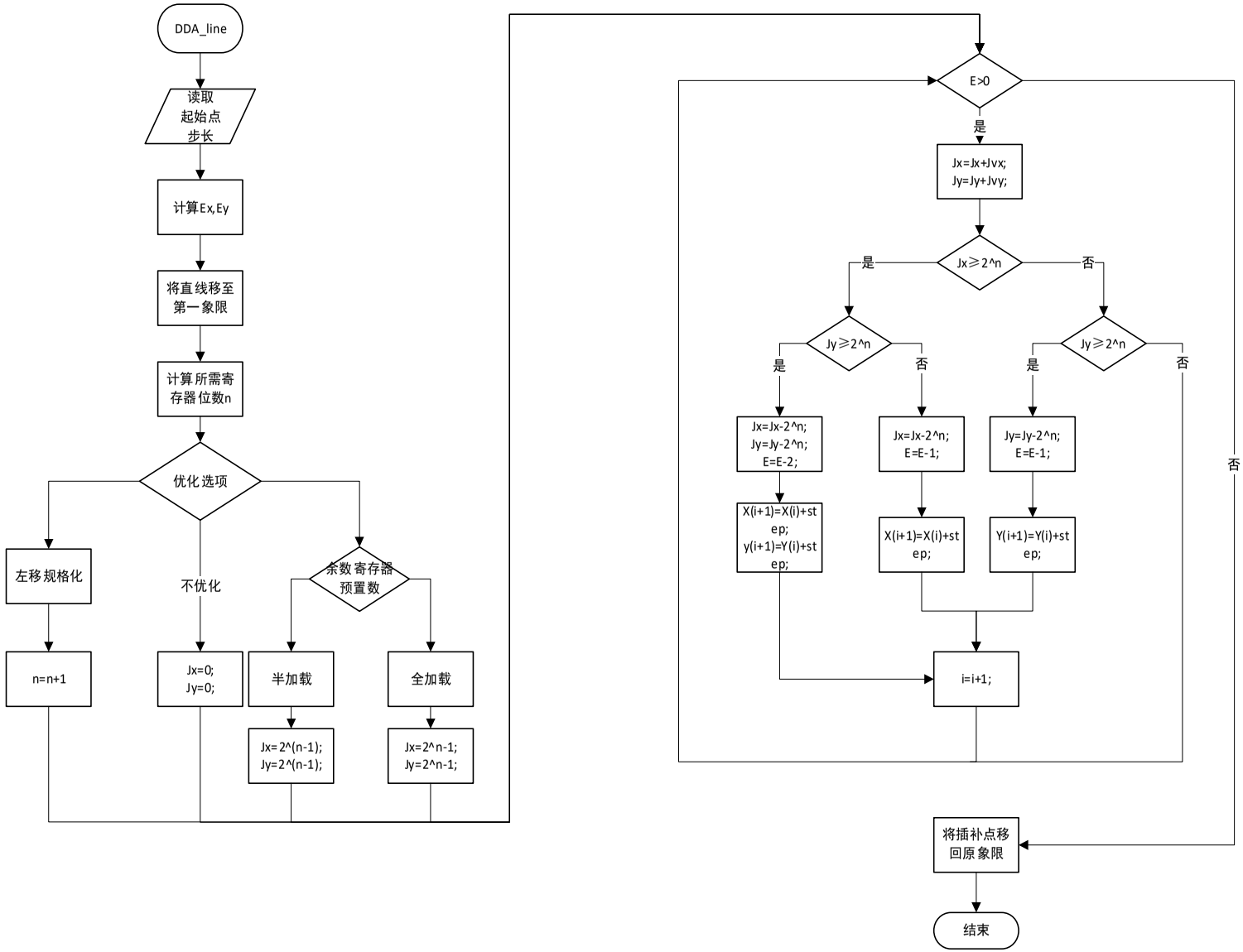
四、详细程序设计流程图

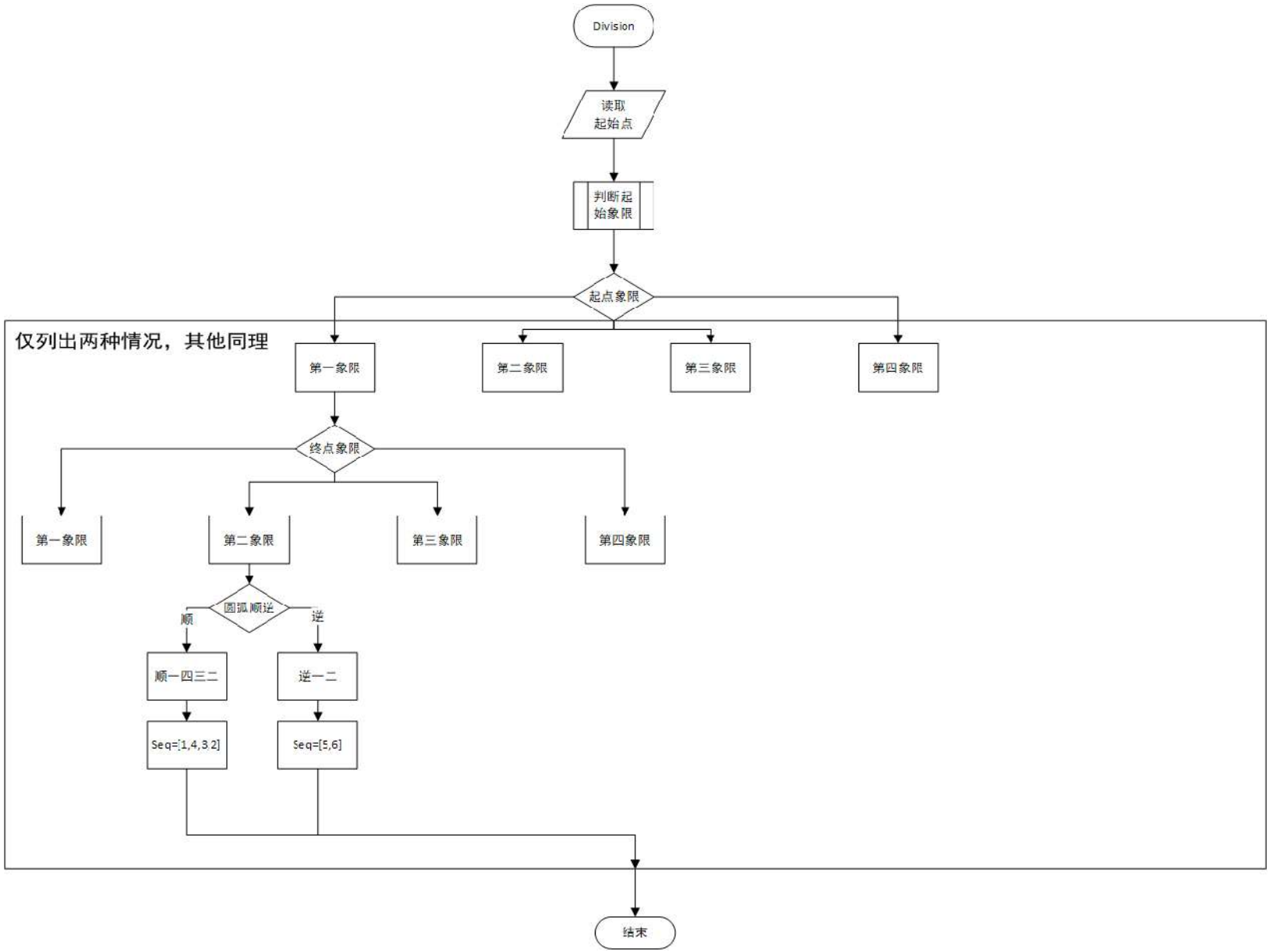
函数名	功能
GUI.m	是实现 GUI 的函数，提供 I/O，调用各个子函数完成插补运算以及显示。
PTP_line.m	实现逐点比较法直线插补的函数，输入值为起始点坐标以及插补步长，返回插补路径点矩阵
DDA_line.m	实现数字积分法直线插补的函数，输入值为起始点坐标、插补步长以及 DDA 法优化选项代码，返回插补路径点矩阵。
Division.m	实现圆弧象限分割的函数，输入起始点坐标及顺逆代号，返回分割后的象限序列代号。
CircleCenter.m	由半径求圆心坐标的函数，输入起始点左边及顺逆优劣弧代号，返回圆心坐标。
QuadJudge.m	象限判断函数，输入一点坐标，返回其所在象限值。

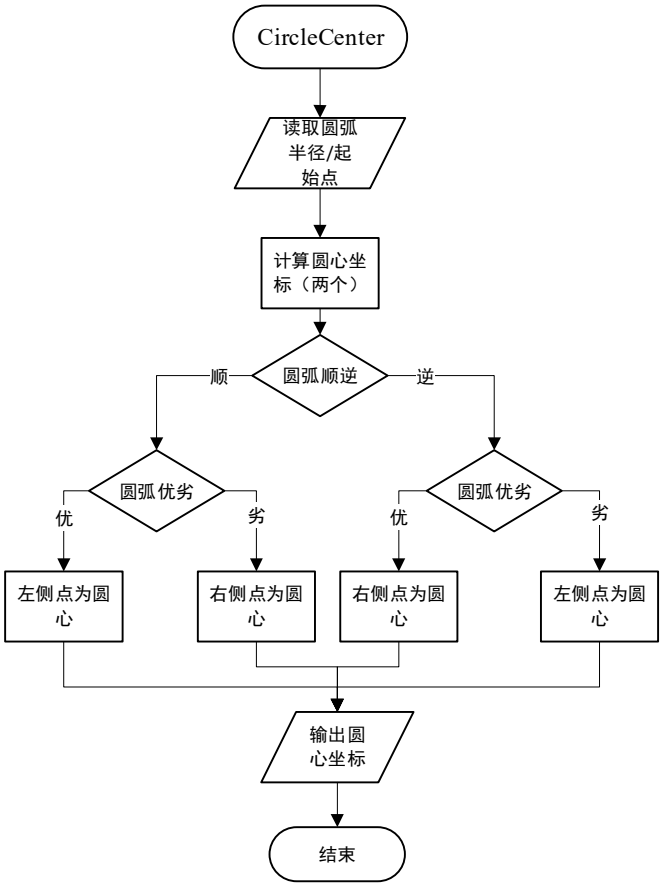
DDA_circle.m	实现数字积分法圆弧插补的函数，输入值为起始点及圆心坐标、顺逆代号、插补步长以及 DDA 法优化选项代号，返回值为圆弧插补点坐标。
DDA_quad.m	实现数字积分法单象限的插补，由 DDA_circle 调用，输入输入值为起始点及圆心坐标、顺逆代号、插补步长以及 DDA 法优化选项代号，返回值为单象限的圆弧插补点坐标。
PTP_circle.m	实现逐点比较法圆弧插补的函数，输入值为起始点及圆心坐标、顺逆代号以及插补步长，返回值为圆弧插补点坐标。
PTP_quad.m	实现数字积分法单象限的插补，由 PTP_circle 调用，输入输入值为起始点及圆心坐标、顺逆代号以及插补步长，返回值为单象限的圆弧插补点坐标。

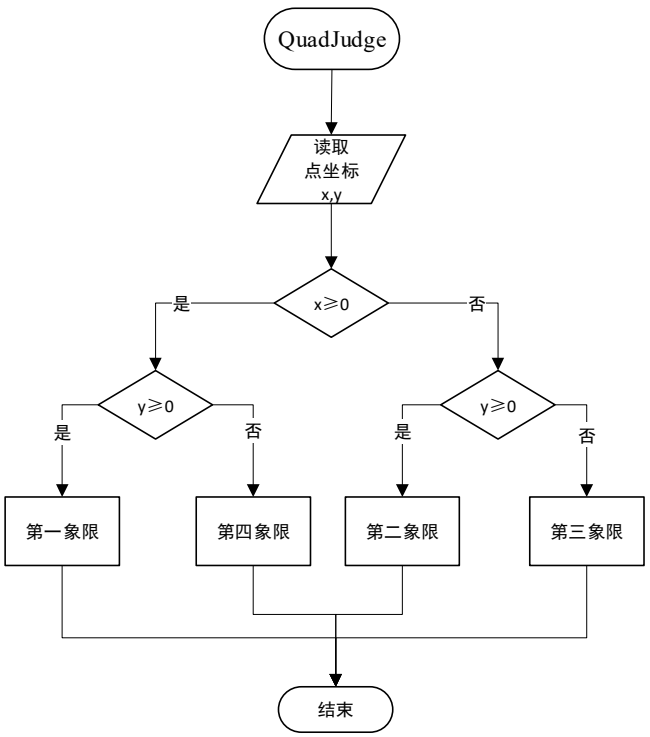

GUI
读取输入坐标数据
判断插补类型
直线插补
圆弧插补
插补方法
圆弧参数类型
直线逐点比较法
直线数字积分法
半径
圆心
读取DDA优化选项
计算圆心坐标
读取圆弧方向
插补方法
圆弧逐点比较法
圆弧数字积分法
读取DDA优化选项
插补计算
插补显示方式
单步插补
连续插补
停止

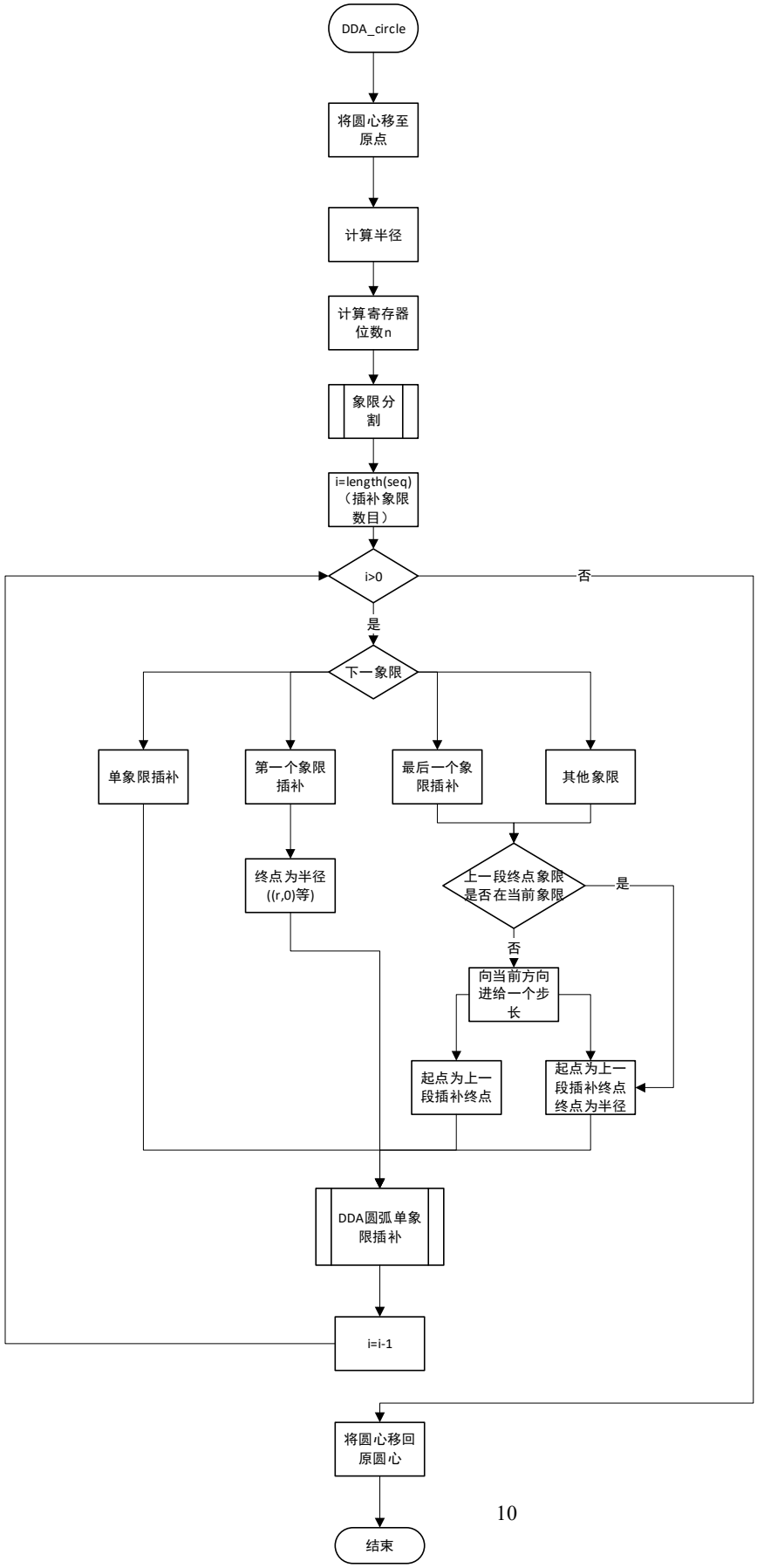

PTP_circle
将圆心移至原点
象限分割
i=length(seq)（插补象限数目）
i>0
否
是
下一象限
单象限插补
第一个象限插补
最后一个象限插补
其他象限
终点为半径((r,0)等)
上一段终点象限是否在当前象限
是
否
向当前方向进给一个步长
起点为上一段插补终点
起点为上一段插补终点终点为半径
逐点圆弧单象限插补
i=i-1
将圆心移回原圆心
结束

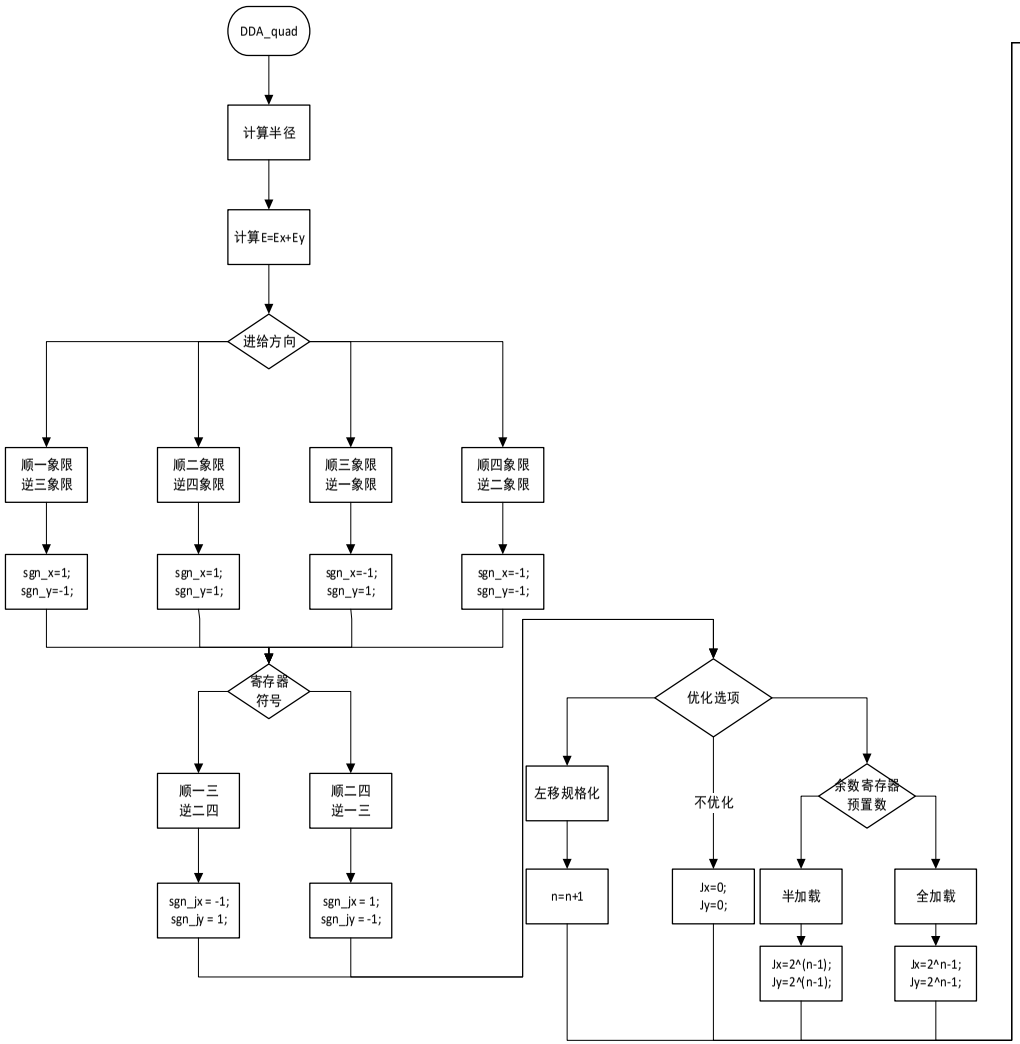

DDA_line
读取
起始点
步长
计算Ex,Ey
将直线移至
第一象限
计算所需寄
存器位数n
优化选项
左移规格化
不优化
余数寄存器
预置数
n=n+1
Jx=0;
Jy=0;
半加载
全加载
Jx=2^(n-1);
Jy=2^(n-1);
Jx=2^n-1;
Jy=2^n-1;
E>0
是
Jx=Jx+Jvx;
Jy=Jy+Jvy;
Jx≥2^n
是
否
Jy≥2^n
是
否
Jy≥2^n
是
否
Jx=Jx-2^n;
Jy=Jy-2^n;
E=E-2;
Jx=Jx-2^n;
E=E-1;
Jy=Jy-2^n;
E=E-1;
X(i+1)=X(i)+st
ep;
y(i+1)=Y(i)+st
ep;
X(i+1)=X(i)+st
ep;
Y(i+1)=Y(i)+st
ep;
i=i+1;
否
将插补点移
回原象限
结束

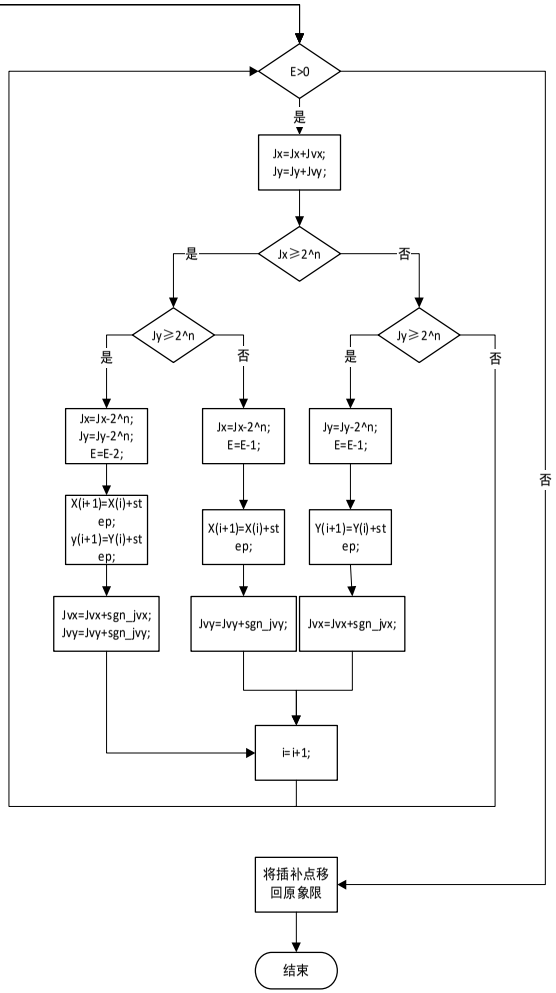

Division
读取
起始点
判断起
始象限
起点象限
仅列出两种情况，其他同理
第一象限
第二象限
第三象限
第四象限
终点象限
第一象限
第二象限
第三象限
第四象限
圆弧顺逆
顺
逆
顺一四三二
逆一二
Seq=[1,4,3,2]
Seq=[5,6]
结束

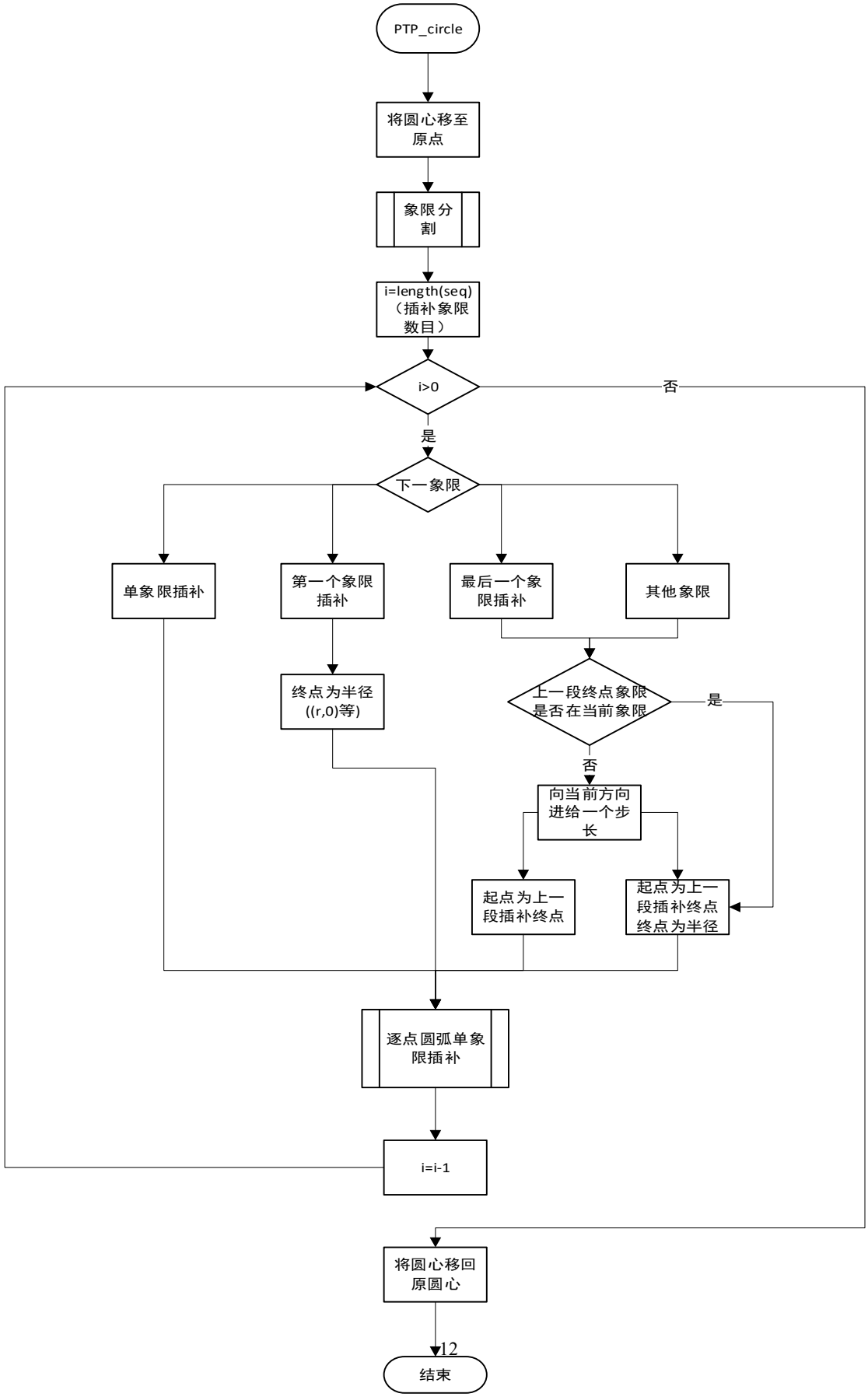

CircleCenter
读取圆弧
半径/起
始点
计算圆心坐
标（两个）
圆弧顺逆
顺
逆
圆弧优劣
圆弧优劣
优
劣
优
劣
左侧点为圆
心
右侧点为圆
心
右侧点为圆
心
左侧点为圆
心
输出圆
心坐标
结束

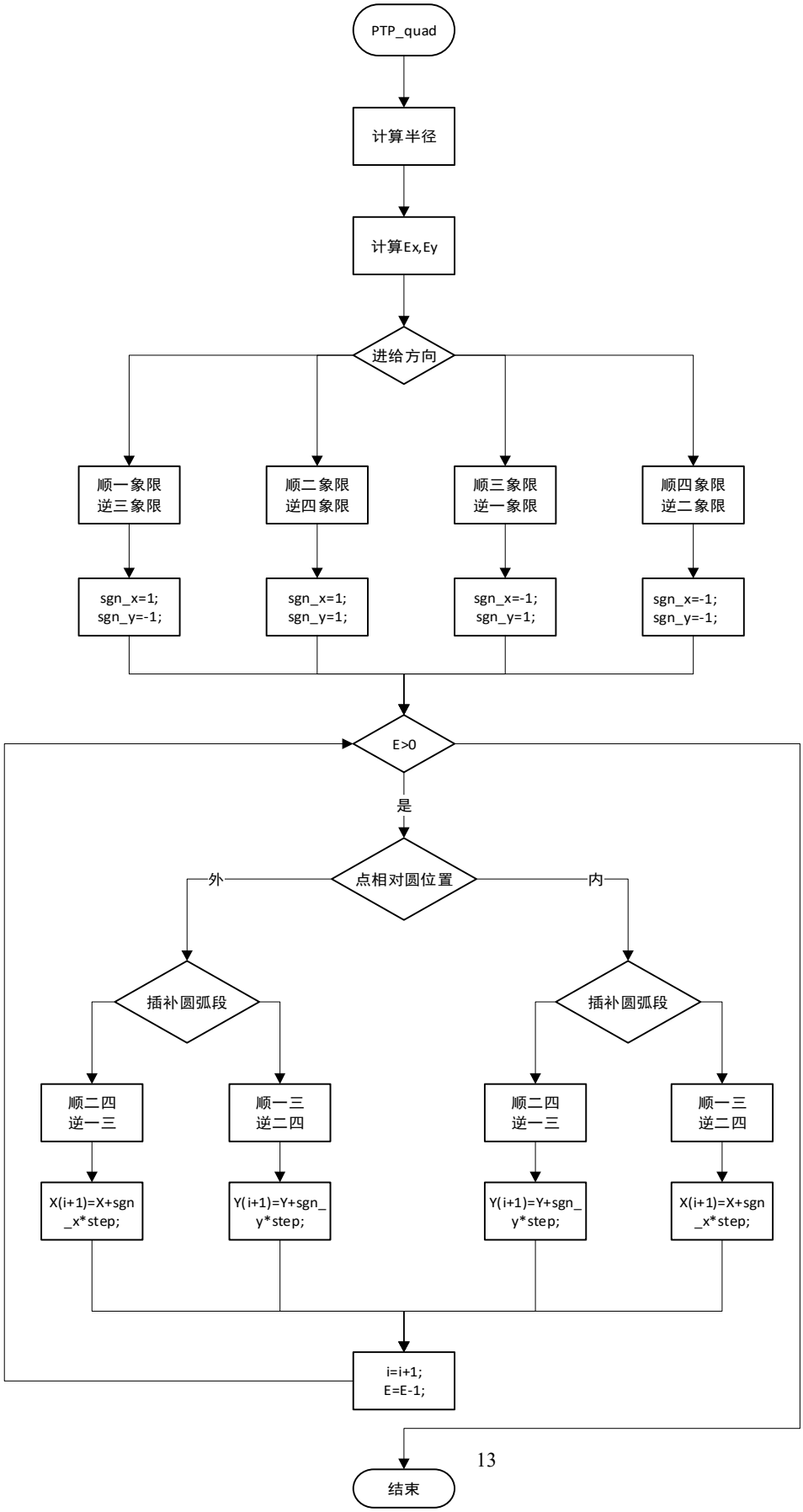

QuadJudge
读取
点坐标
x,y
x≥0
是
否
y≥0
y≥0
是
否
是
否
第一象限
第四象限
第二象限
第三象限
结束


DDA_circle
将圆心移至原点
计算半径
计算寄存器位数n
象限分割
i=length(seq)（插补象限数目）
i>0
否
是
下一象限
单象限插补
第一个象限插补
最后一个象限插补
其他象限
终点为半径((r,0)等)
上一段终点象限是否在当前象限
是
否
向当前方向进给一个步长
起点为上一段插补终点
起点为上一段插补终点终点为半径
DDA圆弧单象限插补
i=i-1
将圆心移回原圆心
结束


DDA_quad
计算半径
计算E=Ex+Ey
进给方向
顺一象限
逆三象限
顺二象限
逆四象限
顺三象限
逆一象限
顺四象限
逆二象限
sgn_x=1;
sgn_y=-1;
sgn_x=1;
sgn_y=1;
sgn_x=-1;
sgn_y=1;
sgn_x=-1;
sgn_y=-1;
寄存器
符号
顺一三
逆二四
顺二四
逆一三
sgn_jx = -1;
sgn_jy = 1;
sgn_jx = 1;
sgn_jy = -1;
优化选项
左移规格化
不优化
余数寄存器
预置数
n=n+1
Jx=0;
Jy=0;
半加载
全加载
Jx=2^(n-1);
Jy=2^(n-1);
Jx=2^n-1;
Jy=2^n-1;


E>0
是
Jx=Jx+Jvx;
Jy=Jy+Jvy;
Jx≥2^n
是
否
Jy≥2^n
Jy≥2^n
是
否
是
否
Jx=Jx-2^n;
Jy=Jy-2^n;
E=E-2;
Jx=Jx-2^n;
E=E-1;
Jy=Jy-2^n;
E=E-1;
X(i+1)=X(i)+step;
Y(i+1)=Y(i)+step;
X(i+1)=X(i)+step;
Y(i+1)=Y(i)+step;
Jvx=Jvx+sgn_jvx;
Jvy=Jvy+sgn_jvy;
Jvy=Jvy+sgn_jvy;
Jvx=Jvx+sgn_jvx;
i=i+1;
否
将插补点移回原象限
结束


PTP_circle
将圆心移至原点
象限分割
i=length(seq)（插补象限数目）
i>0
否
是
下一象限
单象限插补
第一个象限插补
最后一个象限插补
其他象限
终点为半径((r,0)等)
上一段终点象限是否在当前象限
是
否
向当前方向进给一个步长
起点为上一段插补终点
起点为上一段插补终点终点为半径
逐点圆弧单象限插补
i=i-1
将圆心移回原圆心
结束


PTP_quad
计算半径
计算Ex,Ey
进给方向
顺一象限
逆三象限
顺二象限
逆四象限
顺三象限
逆一象限
顺四象限
逆二象限
sgn_x=1;
sgn_y=-1;
sgn_x=1;
sgn_y=1;
sgn_x=-1;
sgn_y=1;
sgn_x=-1;
sgn_y=-1;
E>0
是
点相对圆位置
外
内
插补圆弧段
插补圆弧段
顺二四
逆一三
顺一三
逆二四
顺二四
逆一三
顺一三
逆二四
X(i+1)=X+sgn_x*step;
Y(i+1)=Y+sgn_y*step;
Y(i+1)=Y+sgn_y*step;
X(i+1)=X+sgn_x*step;
i=i+1;
E=E-1;
结束

五、源程序及变量说明

```
function varargout = GUI(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name', mfilename, ...
                   'gui_Singleton', gui_Singleton, ...
                   'gui_OpeningFcn', @GUI_OpeningFcn, ...
                   'gui_OutputFcn', @GUI_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end

function GUI_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;
global one_step
one_step = 1;
global gridd;
gridd = 0;
set(handles.line,'Value',1)
set(handles.center,'Enable','off');
set(handles.Input_center_x,'Enable','off');
set(handles.Input_center_y,'Enable','off');
set(handles.radius,'Enable','off');
set(handles.radius_r,'Enable','off');
set(handles.shun_arc,'Enable','off');
set(handles.ni_arc,'Enable','off');
set(handles.minor_arc,'Enable','off');
set(handles.high_arc,'Enable','off');
set(handles.ptp,'Value',1);
set(handles.nothing,'Value',1);
set(handles.nothing,'Enable','off');
set(handles.left_shift,'Enable','off');
set(handles.preset,'Enable','off');
set(handles.something,'Enable','off');
guidata(hObject, handles);

function varargout = GUI_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;

function pushbutton1_Callback(hObject, eventdata, handles)

function Input_x1_Callback(hObject, eventdata, handles)

function Input_x1_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function Input_y1_Callback(hObject, eventdata, handles)

function Input_y1_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
```

```
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function Input_x2_Callback(hObject, eventdata, handles)

function Input_x2_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function Input_y2_Callback(hObject, eventdata, handles)


function Input_y2_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function Input_step_Callback(hObject, eventdata, handles)

function Input_step_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

%直线插补选项的回调函数
function line_Callback(hObject, eventdata, handles)
set(handles.center,'Enable','off');
set(handles.Input_center_x,'Enable','off');
set(handles.Input_center_y,'Enable','off');
set(handles.radius,'Enable','off');
set(handles.radius_r,'Enable','off');
set(handles.shun_arc,'Enable','off');
set(handles.ni_arc,'Enable','off');
set(handles.minor_arc,'Enable','off');
set(handles.high_arc,'Enable','off');

%圆弧插补选项的回调函数
function circle_Callback(hObject, eventdata, handles)
set(handles.center,'Enable','on');
set(handles.radius,'Enable','on');
set(handles.shun_arc,'Enable','on');
set(handles.ni_arc,'Enable','on');
if get(handles.center,'Value')
    set(handles.Input_center_x,'Enable','on');
    set(handles.Input_center_y,'Enable','on');
else
    set(handles.radius_r,'Enable','on');
    set(handles.minor_arc,'Enable','on');
    set(handles.high_arc,'Enable','on');
end


function left_shift_Callback(hObject, eventdata, handles)
set(handles.something,'Enable','off');


function popupmenu1_Callback(hObject, eventdata, handles)


function popupmenu1_CreateFcn(hObject, eventdata, handles)
if ispc &&
```

```
isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function center_Callback(hObject, eventdata,
handles)
set(handles.radius_r,'Enable','off');
set(handles.minor_arc,'Enable','off');
set(handles.high_arc,'Enable','off');
set(handles.Input_center_x,'Enable','on');
set(handles.Input_center_y,'Enable','on');


function Input_center_x_Callback(hObject,
eventdata, handles)


function Input_center_x_CreateFcn(hObject,
eventdata, handles)
if ispc &&
isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function radius_Callback(hObject, eventdata,
handles)
set(handles.radius_r,'Enable','on');
set(handles.minor_arc,'Enable','on');
set(handles.high_arc,'Enable','on');
set(handles.Input_center_x,'Enable','off');
set(handles.Input_center_y,'Enable','off');


function radius_r_Callback(hObject, eventdata,
handles)


function radius_r_CreateFcn(hObject, eventdata,
handles)
if ispc &&
isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function shun_arc_Callback(hObject, eventdata,
handles)


function ni_arc_Callback(hObject, eventdata,
handles)

function minor_arc_Callback(hObject, eventdata,
handles)


function high_arc_Callback(hObject, eventdata,
handles)


function ptp_Callback(hObject, eventdata,
handles)
set(handles.nothing,'Enable','off');
set(handles.left_shift,'Enable','off');
set(handles.preset,'Enable','off');
set(handles.something,'Enable','off');


function dda_Callback(hObject, eventdata,
handles)
set(handles.nothing,'Enable','on');
set(handles.left_shift,'Enable','on');
set(handles.preset,'Enable','on');
if get(handles.preset,'Value')
    set(handles.something,'Enable','on');
end
```

```
function single_Callback(hObject, eventdata, handles)
sts = handles.sts;
global one_step
one_step = one_step+1;
if one_step > length(sts)
    warndlg('已完成插补！','警告','modal');
    return
end
sing = sts(1:one_step,:);
hold on
plot (sing(:,1),sing(:,2),'b');
```

```
function conti_Callback(~, eventdata, handles)
sts = handles.sts;
hold on
plot (sts(:,1),sts(:,2),'b');
```

```
function reset_Callback(hObject, eventdata, handles)
cla
global one_step
one_step = 1;
set(handles.Input_x1,'String','');
set(handles.Input_y1,'String','');
set(handles.Input_x2,'String','');
set(handles.Input_y2,'String','');
set(handles.Input_step,'String','');
set(handles.Input_center_x,'String','');
set(handles.Input_center_y,'String','');
set(handles.radius_r,'String','');
set(handles.line,'Value',1);
set(handles.center,'Value',1);
set(handles.shun_arc,'Value',1);
set(handles.minor_arc,'Value',1);
set(handles.center,'Enable','off');
set(handles.radius,'Enable','off');
set(handles.shun_arc,'Enable','off');
set(handles.ni_arc,'Enable','off');
set(handles.radius_r,'Enable','off');
set(handles.minor_arc,'Enable','off');
set(handles.high_arc,'Enable','off');
set(handles.Input_center_x,'Enable','off');
set(handles.Input_center_y,'Enable','off');
set(handles.nothing,'Value',1);
set(handles.something,'Value',1);
set(handles.ptp,'Value',1);
set(handles.left_shift,'Enable','off');
set(handles.preset,'Enable','off');
set(handles.nothing,'Enable','off');
set(handles.something,'Enable','off');
```

```
function preset_Callback(hObject, eventdata, handles)
set(handles.something,'Enable','on');
```

```
function something_Callback(hObject, eventdata, handles)
```

```
function something_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function Input_center_y_Callback(hObject, eventdata, handles)
```

```
function Input_center_y_CreateFcn(hObject, eventdata, handles)
```

```
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function pushbutton6_Callback(hObject, eventdata, handles)


function calculate_Callback(hObject, eventdata, handles)
start_x = str2num(get(handles.Input_x1,'String'));
start_y = str2num(get(handles.Input_y1,'String'));
end_x = str2num(get(handles.Input_x2,'String'));
end_y = str2num(get(handles.Input_y2,'String'));
step = str2num(get(handles.Input_step,'String'));
if ((start_x-end_x)^2+(start_y-end_y)^2)<step^2
    warndlg('步长过长！','警告','modal');
    return
end

if get(handles.line,'Value')
    plot ([start_x,end_x],[start_y,end_y],'r');
    axis equal
    if get(handles.dda,'Value')
        if get(handles.nothing,'Value')
            left_s = get(handles.left_shift,'Value');
            preset = 1;
        elseif get(handles.left_shift,'Value')
            left_s = get(handles.left_shift,'Value');
            preset = 1;
        elseif get(handles.preset,'Value')
            left_s = 0;
            preset = get(handles.something,'Value');
        end
        sts = DDA_line( start_x,start_y,end_x,end_y,step,preset,left_s );
    else
        sts = PTP_line(start_x,start_y,end_x,end_y,step);
    end


elseif get(handles.circle,'Value')    %圆弧插补

    if get(handles.shun_arc,'Value') %读取顺逆
        sn = 1;
    elseif get(handles.ni_arc,'Value')
        sn = 0;
    end

    if get(handles.radius,'Value') %半径条件
        if get(handles.high_arc,'Value')
            procon = 1;
        elseif get(handles.minor_arc,'Value')
            procon = 0;
        end
        radius = str2num(get(handles.radius_r,'String'));
        fake_r = sqrt((start_x-end_x)^2+(start_y-end_y)^2);
        if (fake_r>2*radius)
            warndlg('半径参数错误！','警告','modal');
            return;
        end
        cen = CircleCenter( start_x,start_y,end_x,end_y,radius,sn,procon );
        center_x = cen(1);
        center_y = cen(2);
    elseif get(handles.center,'Value')
        center_x = str2num(get(handles.Input_center_x,'String'));
        center_y = str2num(get(handles.Input_center_y,'String'));
```

```
    end
    r1 = sqrt( (start_x-center_x)^2+(start_y-center_y)^2 );
    r2 = sqrt( (end_x-center_x)^2+(end_y-center_y)^2 );
    if (r1/r2>1.05||r1/r2<0.95)
        warndlg('圆心参数错误！','警告','modal');
        return;
    end
    if get(handles.dda,'Value') %数字积分法
        if get(handles.nothing,'Value')
            left_s = get(handles.left_shift,'Value');
            preset = 1;
        elseif get(handles.left_shift,'Value')
            left_s = get(handles.left_shift,'Value');
            preset = 1;
        elseif get(handles.preset,'Value')
            left_s = get(handles.left_shift,'Value');
            preset = get(handles.something,'Value');
        end
        sts = DDA_circle( start_x,start_y,end_x,end_y,center_x,center_y,step,sn,preset,left_s);
    elseif get(handles.ptp,'Value') %逐点比较法
        sts = PTP_circle( start_x,start_y,end_x,end_y,center_x,center_y,step,sn );
    end
    ntn = PTP_circle( start_x,start_y,end_x,end_y,center_x,center_y,0.005,sn );
    plot (ntn(:,1),ntn(:,2),'r')
    hold on
    plot(center_x,center_y,'r*')
    axis equal
end
    handles.sts = sts;
    guidata(hObject, handles);


function circle_all_ButtonDownFcn(hObject, eventdata, handles)


function nothing_Callback(hObject, eventdata, handles)
set(handles.something,'Enable','off');

% --------------------------------------------------------------------
function AxesMenu_Callback(hObject, eventdata, handles)

% --------------------------------------------------------------------
function grid_Callback(hObject, eventdata, handles)
global gridd;
gridd = ~gridd;
if gridd
    set(handles.grid,'Checked','On');
    grid on
else
    set(handles.grid,'Checked','Off');
    grid off
end

% --------------------------------------------------------------------
function Clear_Callback(hObject, eventdata, handles)
global one_step
one_step = 1;
cla;

% --------------------------------------------------------------------
```

```
function figure1_CreateFcn(hObject, eventdata, handles)
ha=axes('units','normalized','pos',[0 0 1 1]);
uistack(ha,'down');
ii=imread('shuimo.jpeg');
image(ii);
colormap gray
set(ha,'handlevisibility','off','visible','on');

% --------------------------------------------------------------------
function lllllll_Callback(hObject, eventdata, handles)
```

```
function [ sts ] = PTP_line( start_x,start_y,end_x,end_y,step )
%start_x,start_y,end_x,end_y,step 分别为起始点坐标
%逐点比较法—直线插补
Ex = round( abs( start_x-end_x )/step );
Ey = round( abs( start_y-end_y )/step );
E = Ex+Ey;
%将直线起点移至坐标原点
trans_x =abs( end_x-start_x );
trans_y =abs( end_y-start_y );
%判断直线的走向
if end_x>=start_x
    if end_y>=start_y
        quad = 1;
    else
        quad = 4;
    end
else
    if end_y>=start_y
        quad = 2;
    else
        quad = 3;
    end
end
%初始化累加函数
F = zeros(E+1,1);
%初始化路径存储函数
sts = zeros(E+1,2);
%累加循环
for i = 1:E
    if F(i)>=0
        F(i+1) = F(i)-trans_y;
        sts(i+1,2) = sts(i,2);
        sts(i+1,1) = sts(i,1)+step;
    else
        F(i+1) = F(i)+trans_x;
        sts(i+1,1) = sts(i,1);
        sts(i+1,2) = sts(i,2)+step;
    end
end
```

```
switch quad
    case 2
        sts(:,1) = -sts(:,1);
    case 3
        sts = -sts;
    case 4
        sts(:,2) = -sts(:,2);
end
sts(:,1) = sts(:,1)+start_x;
sts(:,2) = sts(:,2)+start_y;
plot
(sts(:,1),sts(:,2),[start_x,end_x],[start_y,end_y]);
end
```

```
function [ sts ] = DDA_line( start_x,start_y,end_x,end_y,step,preset,left_shift )
%DDA_line DDA-直线插补
%x1,y1 为起点坐标
%x2,y2 为终点坐标
% step 为步长
%将直线起点移至坐标原点
trans_x =abs( end_x-start_x );
trans_y =abs( end_y-start_y );
%判断直线移至原点后的象限
if end_x>=start_x
    if end_y>=start_y
        quad = 1;
    else
        quad = 4;
    end
else
    if end_y>=start_y
        quad = 2;
    else
        quad = 3;
    end
end
m = max(trans_x,trans_y);
%确定寄存器位数
m = log2(m/step);
if rem(m,1)
    n = ceil(m);
else
    n = m+1;
end
if left_shift
else
    n = n+1;
end

%终点判别寄存器
Ex = round( abs( start_x-end_x )/step );
Ey = round( abs( start_y-end_y )/step );
E = Ex+Ey;
```

```
%初始化寄存器
Jvx = round(trans_x/step);
Jvy = round(trans_y/step);
switch preset
    case 1
        Jx = 0;Jy = 0;
    case 2
        Jx = 2^(n-1);Jy = 2^(n-1);
    case 3
        Jx = 2^n-1;Jy = 2^n-1;
end
sts = [0,0];
%累加循环
i = 1;
while(E)
    Jx = Jx+Jvx;
    Jy = Jy+Jvy;
    if Jx>=2^n&&Jy>=2^n
        Jx = Jx-2^n;
        Jy = Jy-2^n;
        sts(i+1,:) = sts(i,:)+step;
        E = E-2;
    else
        if Jx>=2^n
            Jx = Jx-2^n;
            sts(i+1,1) = sts(i,1)+step;
            sts(i+1,2) = sts(i,2);
            E = E-1;
        else
            if Jy>=2^n
                Jy = Jy-2^n;
                sts(i+1,1) = sts(i,1);
                sts(i+1,2) = sts(i,2)+step;
                E = E-1;
            else
                continue
            end
        end
    end
    i = i+1;
end
```

```
%将直线打回原象限
switch quad
    case 2
        sts(:,1) = -sts(:,1);
    case 3
        sts = -sts;
    case 4
        sts(:,2) = -sts(:,2);
end
%将起点移至原起点
sts(:,1) = sts(:,1)+start_x;
sts(:,2) = sts(:,2)+start_y;
end
```

```
function sts = DDA_circle( start_x,start_y,end_x,end_y,center_x,center_y,step,sn,preset,left_shift)
%DDA_circle DDA_quad 法-圆弧插补
% 顺一二三四为 1234，逆一二三四为 5678
r1 = sqrt( (start_x-center_x)^2+(start_y-center_y)^2 );
r2 = sqrt( (end_x-center_x)^2+(end_y-center_y)^2 );
if r1/r2<1.05&&r1/r2>0.95
    r = (r1+r2)/2;
else
    disp('Error');
    return;
end
%计算所需寄存器位数
n = ceil(log2(r/step));
%将圆心移至原点
x1 = start_x-center_x;
y1 = start_y-center_y;
x2 = end_x-center_x;
y2 = end_y-center_y;
% Ex1 = round( abs( start_x-r )/step);
% Ex2 = round ( abs( ( round( abs( start_x-r )/step )*step-( r-x1 )+ ( r-x2 ) )/step ) );
% Ex = Ex1+Ex2;
% Ey = round( abs( start_y-end_y )/step );
%对象限进行分割
seq = Division(x1,y1,x2,y2,sn);
%初始化路径存储矩阵
sts = [x1,y1];
%按分割后的象限依次进行圆弧插补
for i =1:length(seq)
    switch i
        case length(seq) == 1 %单象限的插补
            sts = DDA_quad( x1,y1,x2,y2,n,step,seq(i),preset,left_shift);
        case 1  %起点所在象限的插补
            switch seq(i)
                case 1
                    sx_2=r;sy_2=0;
                case 2
                    sx_2=0;sy_2=r;
                case 3
                    sx_2=-r;sy_2=0;
                case 4
                    sx_2=0;sy_2=-r;
                case 5
                    sx_2=0;sy_2=r;
                case 6
                    sx_2=-r;sy_2=0;
                case 7
                    sx_2=0;sy_2=-r;
                case 8
                    sx_2=r;sy_2=0;
end
```

```
                sts                                 =
[sts;DDA_quad(  x1,y1,sx_2,sy_2,n,step,seq(i),pr
eset,left_shift)];
            case length(seq) %终点所在象限的插
补
                sx_1 = sts(size(sts,1),1);
                sy_1 = sts(size(sts,1),2);
                sts                                 =
[sts;DDA_quad(  sx_1,sy_1,x2,y2,n,step,seq(i),pr
eset,left_shift)];
            otherwise %其他象限的插补
                %判断变象限是否需要补步
                if
(QuadJudge(sts(size(sts,1),1),sts(size(sts,1),2))~=
seq(i)||QuadJudge(sts(size(sts,1),1),sts(size(sts,1),
2))~=
seq(i)+4)&&(sts(size(sts,1),1))&&(sts(size(sts,1),
2))
                    %当前插补点所在象限
                    if
seq(i)==5||seq(i)==6||seq(i)==7||seq(i)==8
                        Quad1 = seq(i)-4;
                    else
                        Quad1 = seq(i);
                    end
                    %上一点所在象限
                    Quad2                           =
QuadJudge(sts(size(sts,1),1),sts(size(sts,1),2));
                    switch Quad1
                        case 1
                            switch Quad2
                                case 2
                                    sts             =
[sts;[sts(size(sts,1),1)+step,sts(size(sts,1),2)]];
                                case 4
                                    sts             =
[sts;[sts(size(sts,1),1),sts(size(sts,1),2)+step]];
                            end
                        case 2
                            switch Quad2
                                case 1
                                    sts             =
[sts;[sts(size(sts,1),1)-step,sts(size(sts,1),2)]];
                                case 3
                                    sts             =
[sts;[sts(size(sts,1),1),sts(size(sts,1),2)+step]];
                            end
                        case 3
                            switch Quad2
                                case 2
                                    sts             =
[sts;[sts(size(sts,1),1),sts(size(sts,1),2)-step]];
                                case 4
                                    sts             =
[sts;[sts(size(sts,1),1)-step,sts(size(sts,1),2)]];
                            end
                        case 4
                            switch Quad2
                                case 1
                                    sts             =
[sts;[sts(size(sts,1),1),sts(size(sts,1),2)-step]];
                                case 3
                                    sts             =
[sts;[sts(size(sts,1),1)+step,sts(size(sts,1),2)]];
                            end
                    end
                end
                switch seq(i)
                    case 1
%                       sx_1=0;sy_1=r;
                        sx_2=r;sy_2=0;
                    case 2
%                       sx_1=-r;sy_1=0;
                        sx_2=0;sy_2=r;
                    case 3
%                       sx_1=0;sy_1=-r;
                        sx_2=-r;sy_2=0;
                    case 4
%                       sx_1=r;sy_1=0;
                        sx_2=0;sy_2=-r;
                    case 5
                        sx_2=0;sy_2=r;
```

```
%                       sx_1=r;sy_1=0;
                case 6
                    sx_2=-r;sy_2=0;
%                       sx_1=0;sy_1=r;
                case 7
                    sx_2=0;sy_2=-r;
%                       sx_1=-r;sy_1=0;
                case 8
                    sx_2=r;sy_2=0;
%                       sx_1=0;sy_1=-r;
            end
            sts =
[sts;DDA_quad( sts(size(sts,1),1),sts(size(sts,1),2
),sx_2,sy_2,n,step,seq(i),preset,left_shift)];
    end
end
%将插补后的轨迹圆心由原点移至原圆心
sts = sts+[center_x,center_y];
end
```

```
function [ sts ] =
DDA_quad( start_x,start_y,end_x,end_y,n,step,se
q,preset,left_shift)
%  对不同象限顺逆圆弧进行插补
%x1,y1 为起点坐标
%x2,y2 为终点坐标
% n 为寄存器位数
% step 为步长
% seq 为象限及顺逆代号
%终点判别寄存器初始化
Ex = round( abs( end_x-start_x )/step );
Ey = round( abs( end_y-start_y )/step );
switch seq %寄存器修正符号判断
    case {1,3,6,8}
        sgn_jx = -1;sgn_jy = 1;
    case {2,4,5,7}
        sgn_jx = 1;sgn_jy = -1;
end
if left_shift
    n = n+1;
    sgn_jx = 2*sgn_jx;
    sgn_jy = 2*sgn_jy;
end
switch seq %进给方向符号判断
    case {1,7}
        sgn_x=1;sgn_y=-1;
    case {2,8}
        sgn_x=1;sgn_y=1;
    case {3,5}
        sgn_x=-1;sgn_y=1;
    case {4,6}
        sgn_x=-1;sgn_y=-1;
end
%赋寄存器初值
Jvx = round(abs(start_y/step));
Jvy = round(abs(start_x/step));
bs = 2^n;m=1;
%起点赋值，累加寄存器初始化
sts(1,:) = [start_x,start_y];
switch preset
    case 1
```

```
        Jx = 0;Jy = 0;
    case 2
        Jx = 2^(n-1);Jy = 2^(n-1);
    case 3
        Jx = 2^n-1;Jy = 2^n-1;
end
%累加循环
while (Ex>0||Ey>0)
    if Ex>0
        Jx = Jx+Jvx;
    end
    if Ey>0
        Jy = Jy+Jvy;
    end
    if Jx>bs %Jx 溢出
        if Jy>bs %Jy 溢出
            Jx = Jx-bs;
            Jy = Jy-bs;
            Jvx = Jvx+sgn_jx;
            Jvy = Jvy+sgn_jy;
            sts(m+1,:)                    =
sts(m,:)+[sgn_x*step,sgn_y*step];
            Ex = Ex-1;
            Ey = Ey-1;
        else %Jy 未溢出
            Jx = Jx-bs;
            Jvy = Jvy+sgn_jy;
            sts(m+1,:)                    =
sts(m,:)+[sgn_x*step,0];
            Ex = Ex-1;
        end
    else %Jx 未溢出
        if Jy>bs %Jy 溢出
            Jy = Jy-bs;
            Jvx = Jvx+sgn_jx;
            sts(m+1,:)                    =
sts(m,:)+[0,sgn_y*step];
            Ey = Ey-1;
        else %Jy 未溢出
            continue
        end
    end
    m = m+1;
end
end
```

```
function seq = Division( transtart_x,transtart_y,transend_x,transend_y,sn )
%Division 对输入的数据进行象限分割
%transtart_x,transtart_y,transend_x,transend_y 分别为转换后的起始点坐标
%sn=1 为顺时针，sn=0 为逆时针
%返回值 seq 为分割后的象限插补序列
switch QuadJudge(transtart_x,transtart_y)
    case 1  %起点第一象限
        switch QuadJudge(transend_x,transend_y)
            case 1 %终点为第一象限
                if sn == 1 && transtart_x>transend_x
                    seq = [1,4,3,2,1];
                else
                    if sn == 1 && transtart_x<transend_x
                    seq = 1;
                    else
                        if sn == 0 && transtart_x>transend_x
                        seq = 5;
                        else
                            seq = [5,6,7,8,5];
                        end
                    end
                end
            case 2 %终点为第二象限
                if sn == 1
                    seq = [1,4,3,2];
                else
                    seq = [5,6];
                end
            case 3 %终点为第三象限
                if sn == 1
                    seq = [1,4,3];
                else
                    seq = [5,6,7];
                end
            case 4 %终点为第四象限
                if sn == 1
                    seq = [1,4];
                else
                    seq = [5,6,7,8];
                end
        end
    case 2 %起点第二象限
        switch QuadJudge(transend_x,transend_y)
            case 1
                if sn == 1
                    seq = [2,1];
                else
                    seq = [6,7,8,5];
                end
            case 2
                if sn == 1 && transtart_x>transend_x
                    seq = [2,1,4,3,2];
                else
                    if sn == 1 && transtart_x<transend_x
                    seq = 2;
                    else
                        if sn == 0 && transtart_x>transend_x
                        seq = 6;
                        else
                            seq = [6,7,8,5,6];
                        end
                    end
                end
            case 3
                if sn == 1
                    seq = [2,1,4,3];
                else
                    seq = [6,7];
                end
```

```
            case 4
                if sn == 1
                    seq = [2,1,4];
                else
                    seq = [6,7,8];
                end
        end
    case 3 %起点第三象限
        switch
QuadJudge(transend_x,transend_y)
            case 1
                if sn == 1
                    seq = [3,2,1];
                else
                    seq = [7,8,5];
                end
            case 2
                if sn == 1
                    seq = [3,2];
                else
                    seq = [7,8,5,6];
                end
            case 3
                if sn == 1 &&
transtart_x>transend_x
                    seq = 3;
                else
                    if sn == 1 &&
transtart_x<transend_x
                    seq = [3,2,1,4,3];
                    else
                        if sn == 0 &&
transtart_x>transend_x
                        seq = [7,8,5,6,7];
                        else
                            seq = 7;
                        end
                    end
                end
            case 4
                if sn == 1
                    seq = [3,2,1,4];
                else
                    seq = [7,8];
                end
        end
    case 4 %起点第四象限
        switch
QuadJudge(transend_x,transend_y)
            case 1
                if sn == 1
                    seq = [4,3,2,1];
                else
                    seq = [8,5];
                end
            case 2
                if sn == 1
                    seq = [4,3,2];
                else
                    seq = [8,5,6];
                end
            case 3
                if sn == 1
                    seq = [4,1,2,3];
                else
                    seq = [8,7];
                end
            case 4
                if sn == 1 &&
transtart_x>transend_x
                    seq = 4;
                else
                    if sn == 1 &&
transtart_x<transend_x
                    seq = [4,3,2,1,4];
                    else
                        if sn == 0 &&
transtart_x>transend_x
                        seq = [8,5,6,7,8];
                        else
                            seq = 8;
                        end
```

```
            end
        end
    end
end
end
```

```
function [ cen ] = CircleCenter( x1,y1,x2,y2,r,sn,procon )
%CircleCenter 根据圆弧起点终点以及顺逆优劣计算圆心坐标
%    此处显示详细说明
%顺圆弧 sn=1，逆圆弧 sn=0；优弧 procon=1，劣弧 procon=0；
syms a b;
%接圆心方程
[a,b] = solve([(x1-a)^2+(y1-b)^2==r^2,(x2-a)^2+(y2-b)^2==r^2],[a,b]);
if length(a)==1
    cen = [(x2-x1)/2,(y2-y1)/2];
    return;
end
if ((y1-y2)*double(a(1))+(x2-x1)*double(b(1))+x1*y2-x2*y1)>0
    cen1 = double([a(1),b(1)]);
    cen2 = double([a(2),b(2)]);
else
    cen1 = double([a(2),b(2)]);
    cen2 = double([a(1),b(1)]);
end

if sn
    if procon
        cen = cen1;
    else
        cen = cen2;
    end
else
    if procon
        cen = cen2;
    else
        cen = cen1;
    end
end
end
```

```
function quad = QuadJudge( x,y )
% QuadJudge    判断点的象限
if x>=0
    if y>=0
        quad=1;
    else
        quad=4;
    end
else if y>=0
        quad=2;
    else
        quad=3;
    end
end
end
```

```
function [ sts] = PTP_circle( start_x,start_y,end_x,end_y,center_x,center_y,step,sn )
%PTP_line 逐点比较法-圆弧插补
% 顺一二三四为 1234，逆一二三四为 5678
%start_x,start_y,end_x,end_y,center_x,center_y 分别为起点、终点、圆心的坐标
%step 为步长，sn 为顺逆标识符
r1 = sqrt( (start_x-center_x)^2+(start_y-center_y)^2 );
r2 = sqrt( (end_x-center_x)^2+(end_y-center_y)^2 );
if r1/r2<1.05&&r1/r2>0.95
    r = (r1+r2)/2;
else
    disp('Error');
    return;
end
%将圆心移至原点
x1 = start_x-center_x;
y1 = start_y-center_y;
x2 = end_x-center_x;
y2 = end_y-center_y;
% Ex1 = round( abs( start_x-r )/step);
% Ex2 = round ( abs( ( round( abs( start_x-r )/step )*step-( r-x1 )+ ( r-x2 ) )/step ) );
% Ex = Ex1+Ex2;
% Ey = round( abs( start_y-end_y )/step );
%对象限进行分割
seq = Division(x1,y1,x2,y2,sn);
%初始化路径存储矩阵
sts = [x1,y1];
%按分割后的象限依次进行圆弧插补
for i =1:length(seq)
    switch i
        case length(seq) == 1 %单象限的插补
            sts = PTP_quad( x1,y1,x2,y2,step,seq(i));
            case 1    %起点所在象限的插补
            switch seq(i)
                case 1
```

```
                sx_2=r;sy_2=0;
            case 2
                sx_2=0;sy_2=r;
            case 3
                sx_2=-r;sy_2=0;
            case 4
                sx_2=0;sy_2=-r;
            case 5
                sx_2=0;sy_2=r;
            case 6
                sx_2=-r;sy_2=0;
            case 7
                sx_2=0;sy_2=-r;
            case 8
                sx_2=r;sy_2=0;
        end
        sts                                =
[sts;PTP_quad( x1,y1,sx_2,sy_2,step,seq(i))];
    case length(seq) %终点所在象限的插
补
        sx_1 = sts(size(sts,1),1);
        sy_1 = sts(size(sts,1),2);
        sts                                =
[sts;PTP_quad( sx_1,sy_1,x2,y2,step,seq(i))];
    otherwise %其他象限的插补
        %判断变象限是否需要补步
        if
(QuadJudge(sts(size(sts,1),1),sts(size(sts,1),2))~=
seq(i)||QuadJudge(sts(size(sts,1),1),sts(size(sts,1),
2))~=
seq(i)+4)&&(sts(size(sts,1),1))&&(sts(size(sts,1),
2))
            %当前插补点所在象限
            if
seq(i)==5||seq(i)==6||seq(i)==7||seq(i)==8
                Quad1 = seq(i)-4;
            else
                Quad1 = seq(i);
            end
            %上一点所在象限
            Quad2                          =
QuadJudge(sts(size(sts,1),1),sts(size(sts,1),2));
                switch Quad1
                    case 1
                        switch Quad2
                            case 2
                                sts        =
[sts;[sts(size(sts,1),1)+step,sts(size(sts,1),2)]];
                            case 4
                                sts        =
[sts;[sts(size(sts,1),1),sts(size(sts,1),2)+step]];
                        end
                    case 2
                        switch Quad2
                            case 1
                                sts        =
[sts;[sts(size(sts,1),1)-step,sts(size(sts,1),2)]];
                            case 3
                                sts        =
[sts;[sts(size(sts,1),1),sts(size(sts,1),2)+step]];
                        end
                    case 3
                        switch Quad2
                            case 2
                                sts        =
[sts;[sts(size(sts,1),1),sts(size(sts,1),2)-step]];
                            case 4
                                sts        =
[sts;[sts(size(sts,1),1)-step,sts(size(sts,1),2)]];
                        end
                    case 4
                        switch Quad2
                            case 1
                                sts        =
[sts;[sts(size(sts,1),1),sts(size(sts,1),2)-step]];
                            case 3
                                sts        =
[sts;[sts(size(sts,1),1)+step,sts(size(sts,1),2)]];
                        end
                end
            end
            switch seq(i)
```

```
                case 1
                    sx_2=r;sy_2=0;
                case 2
                    sx_2=0;sy_2=r;
                case 3
                    sx_2=-r;sy_2=0;
                case 4
                    sx_2=0;sy_2=-r;
                case 5
                    sx_2=0;sy_2=r;
                case 6
                    sx_2=-r;sy_2=0;
                case 7
                    sx_2=0;sy_2=-r;
                case 8
                    sx_2=r;sy_2=0;
            end
            sts =
[sts;PTP_quad( sts(size(sts,1),1),sts(size(sts,1),2),
sx_2,sy_2,step,seq(i))];
        end
end
%将插补后的轨迹圆心由原点移至原圆心
sts = sts+[center_x,center_y];
end
```

```
function [ sts ] =
PTP_quad( start_x,start_y,end_x,end_y,step,seq )
%PTP_quad 使用逐点比较法对各个象限的圆
弧进行插补
%x1,y1 为起点坐标
%x2,y2 为终点坐标
% step 为步长
% seq 为象限及顺逆代号
r = sqrt((end_x)^2+(end_y^2));
%终点判别寄存器初始化
Ex = round( abs( end_x-start_x )/step );
Ey = round( abs( end_y-start_y )/step );
E = Ex+Ey;
%进给方向符号判断
switch seq
    case {1,7}
        sgn_x=1;sgn_y=-1;
    case {2,8}
        sgn_x=1;sgn_y=1;
    case {3,5}
        sgn_x=-1;sgn_y=1;
    case {4,6}
        sgn_x=-1;sgn_y=-1;
end
% F = zeros(E+1,1);
sts = [start_x,start_y];
%插补判断循环
for i = 1:E
    %判断当前点相对圆的位置
    if (sqrt( (sts(i,1))^2+(sts(i,2)^2)))>=r
        switch seq
            case {2,4,5,7}
                sts(i+1,2) = sts(i,2);
                sts(i+1,1) =
sts(i,1)+sgn_x*step;
            case {1,3,6,8}
                sts(i+1,1) = sts(i,1);
                sts(i+1,2) =
sts(i,2)+sgn_y*step;
        end
    else
```

```
            switch seq
                case {2,4,5,7}
                    sts(i+1,1) = sts(i,1);
                    sts(i+1,2)                  =
sts(i,2)+sgn_y*step;
                case {1,3,6,8}
                    sts(i+1,2) = sts(i,2);
                    sts(i+1,1)                  =
sts(i,1)+sgn_x*step;
            end
        end
end

end
```