**第一问主要源代码：**

**P1subProblem1.m**

% 第一问第一小问脚本

clear

readData

load brakingCurve.mat

As = 6;

At = 7;

targetTime = 110;

dwellTime = [];

deltaE = 0.03 \* 1000 \* 3600; % deltaE 为 0.1 千瓦时

[S,V,T,F,calS,calDist,Acce,interSta,totalT,totalE]=optimalStationAlgo( As,At,dwellTime,targetTime,speedLimit,gradient,...

curvature,brakingCurveS,brakingCurveV,curveTerminal,stationP,deltaE);

tempIndex = find(diff(T)==0);

if ~isempty(tempIndex)

S(tempIndex) = [];

V(tempIndex) = [];

T(tempIndex) = [];

F(tempIndex) = [];

calS(tempIndex) = [];

calDist(tempIndex) = [];

Acce(tempIndex) = [];

end

Time = 0:floor(T(end));

N = length(Time);

resultTable = zeros(N+1,9);

resultTable(1:N,1) = Time; % 时刻

secondV = interp1(T,V,Time);

resultTable(1:N,2) = secondV \* 100; % 实际速度(cm/s)

resultTable(1:N,3) = secondV \* 3.6; % 实际速度(km/h)

resultTable(1:N,4) = interp1(T,Acce,Time); % 计算加速度(m/s2)

resultTable(1:N,5) = interp1(T,calDist,Time); % 计算距离(m)

resultTable(1:N,6) = interp1(T,calS,Time); % 计算公里标(m)

secondS = interp1(T,S,Time); % 实际公里标(m)

for i = 1:N-1

[ ~,~,resultTable(i,7) ] = groundConditionFun( secondS(i),gradient,curvature );

end

secondF = interp1(T,F,Time); % 计算牵引力(N)

resultTable(1:N,8) = secondF;

resultTable(1:N,9) = secondF .\* secondV;

temp = (stationP(As) - stationP(At)) - resultTable(end-1,6);

resultTable(end,:) = [targetTime,0,0,0,resultTable(end-1,5)+temp,resultTable(end-1,6)+temp,resultTable(end,7),0,0];

Timestr = cell(N+1,1);

for i = 1:N+1

Timestr{i} = second2Time( i - 1 );

end

**primarySolutionFun.m**

function [ S,V,E,T,F,totalE,totalT,section ] =...

primarySolutionFun( s0,s1,speedLimit,gradient,...

curvature, brakingCurveS,brakingCurveV,curveTerminal )

%primarySolutionFun 生成两个站点之间的初始解

% 输入参数：

% s0 --- 初始公里标

% s1 --- 终点公里标

% speedLimit --- 速度限制矩阵

% gradient --- 轨道的坡度

% curvature --- 轨道的曲率

% brakingCurveS --- 区间制动曲线公里报向量

% brakingCurveV --- 区间制动曲线速度向量

% curveTerminal --- 区间制动曲线端点数据

%

% 输出参数：

% S --- 公里标向量

% V --- 对应的速度向量

% E --- 对应的消耗能量向量

% T --- 对应的时间向量

% F --- 对应的牵引力向量

% totalE --- 消耗的总能量

% totalT --- 消耗的总时间

% section.EndS --- 区间的初始和终止公里标

% section.EndV --- 区间的初始和终止速度

% section.SpeedLimit --- 区间的速度限制

% section.E --- 区间消耗的能量

% section.usedT --- 区间消耗的时间

% section.S --- 区间的公里标向量

% section.V --- 区间的速度向量

% section.T --- 区间的时间向量

% section.braking --- 区间的制动区间

index1 = find(speedLimit(:,3)>=s0,1);

index2 = find(speedLimit(:,3)>s1,1);

section.EndS = [];

section.SpeedLimit = [];

brakingTerminal = [];

if index1 == index2

section.EndS = [s0,s1];

section.SpeedLimit = speedLimit(index1,2);

else

for i = index1:-1:index2

if i == index1

section.EndS = [section.EndS;s0,speedLimit(i,1)];

section.SpeedLimit = [section.SpeedLimit;speedLimit(i,2)];

elseif i == index2

section.EndS = [section.EndS;speedLimit(i,3),s1];

section.SpeedLimit = [section.SpeedLimit;speedLimit(i,2)];

else

section.EndS = [section.EndS;speedLimit(i,[3 1])];

section.SpeedLimit = [section.SpeedLimit;speedLimit(i,2)];

end

end

end

if (section.EndS(1,1) - section.EndS(1,2)) <= 3

section.EndS(2,1) = section.EndS(1,1);

section.EndS(1,:) = [];

section.SpeedLimit(1) = [];

end

if (section.EndS(end,1) - section.EndS(end,2)) <= 3

section.EndS(end-1,2) = section.EndS(end,2);

section.EndS(end,:) = [];

section.SpeedLimit(end) = [];

end

sectionNum = size(section.EndS,1);

section.EndV = zeros(sectionNum,2);

section.E = zeros(sectionNum,1);

section.usedT = zeros(sectionNum,1);

section.S = cell(sectionNum,1);

section.V = cell(sectionNum,1);

section.T = cell(sectionNum,1);

section.F = cell(sectionNum,1);

M = 194295; % 列车质量 kg

highSpeed = 50/3.6; % 最初加速到的最高速度

L = 10000;

S = linspace(s0,s1,L);

Ssize = size(S);

V = zeros(Ssize);

E = zeros(Ssize);

F = zeros(Ssize);

% 终点制动曲线

[ endBrakingCurveV,endBrakingCurveS ] = brakingCurveFun( s0, s1, 0, gradient, curvature );

E(1) = 0; % 消耗能量初始化

V(1) = 0; % 初始速度为0

% 牵引阶段

i = 2;

while (i<length(V) && V(i-1) < highSpeed)

[ Fmax ] = maxTractionFun( V(i - 1) );

[W] = totalResistanceFun(V(i - 1), S(i-1), gradient, curvature);

capacityMaxA = (Fmax - W) / M; % 能够达到的最大加速度

if capacityMaxA > 1

a = 1; % 实际加速度, 因为题目限制最大加速度不能超过1

totalF = M \* a; % 合力

F(i - 1) = totalF + W; % 实际牵引力

else

a = capacityMaxA;

F(i - 1) = Fmax;

end

V(i) = sqrt((V(i-1))^2 + 2 \* a \* (S(i-1) - S(i)));

if checkMeetEndBrakingCurve(S(i),V(i),endBrakingCurveS,endBrakingCurveV)

V(i-1:end) = interp1(endBrakingCurveS,endBrakingCurveV,S(i-1:end),'pchip');

brakingTerminal = [brakingTerminal;S(i-1),S(end)]; % 记录制动区间

E(i:end) = E(i-1);

i = length(V);

elseif checkMeetBrakingCurve(S(i),V(i),brakingCurveS,brakingCurveV,curveTerminal)

index = find(curveTerminal(:,1)>S(i),1);

tempEnd = curveTerminal(index,2);

index2 = find(S<tempEnd,1);

V(i-1:index2) = interp1(brakingCurveS,brakingCurveV,S(i-1:index2),'pchip');

brakingTerminal = [brakingTerminal;S(i-1),S(index2)]; % 记录制动区间

E(i:index2) = E(i-1);

i = index2 + 1;

break

else

E(i) = E(i-1) + F(i - 1) \* (S(i-1) - S(i));

i = i + 1;

end

end

% 惰行与制动阶段

while (i<=length(V))

[W] = totalResistanceFun(V(i - 1), S(i-1), gradient, curvature);

a = -W / M; % 惰行加速度

if a < -0.04

a = - 0.04;

realF = M \* a;

F(i - 1) = realF + W;

end

V(i) = sqrt((V(i-1))^2 + 2 \* a \* (S(i-1) - S(i)));

E(i) = E(i-1) + F(i - 1) \* (S(i-1) - S(i));

if checkMeetEndBrakingCurve(S(i),V(i),endBrakingCurveS,endBrakingCurveV)

V(i-1:end) = interp1(endBrakingCurveS,endBrakingCurveV,S(i-1:end),'pchip');

brakingTerminal = [brakingTerminal;S(i-1),S(end)]; % 记录制动区间

E(i:end) = E(i-1);

i = length(V)+1;

elseif checkMeetBrakingCurve(S(i),V(i),brakingCurveS,brakingCurveV,curveTerminal)

index = find(curveTerminal(:,1)>S(i),1);

tempEnd = curveTerminal(index,2);

index2 = find(S<tempEnd,1);

V(i-1:index2) = interp1(brakingCurveS,brakingCurveV,S(i-1:index2),'pchip');

brakingTerminal = [brakingTerminal;S(i-1),S(index2)]; % 记录制动区间

E(i:index2) = E(i-1);

i = index2 + 1;

else

i = i + 1;

end

end

diffS = abs(diff(S));

meanV = mean([V(1:end-1);V(2:end)]);

T = cumsum([0,diffS ./ meanV]);

totalT = T(end);

totalE = E(end);

% 求解区间消耗能量与时间

for i = 1:sectionNum

tempE = interp1(S,E,section.EndS(i,:),'pchip');

%tempT = interp1(S,T,section.EndS(i,:),'pchip');

tempV = interp1(S,V,section.EndS(i,:),'pchip');

section.E(i) = diff(tempE);

%section.usedT(i) = diff(tempT);

section.EndV(i,:) = tempV;

end

for i = 1:sectionNum

if i == sectionNum

tempIndex = S<=section.EndS(i,1)&S>=section.EndS(i,2);

else

tempIndex = S<=section.EndS(i,1)&S>section.EndS(i,2);

end

section.S{i} = S(tempIndex);

tempT = T(tempIndex);

section.T{i} = tempT - tempT(1);

section.usedT(i) = section.T{i}(end);

section.V{i} = V(tempIndex);

section.F{i} = F(tempIndex);

end

% 求解区间制动区间

section.braking = cell(sectionNum,1);

for i = 1:sectionNum

section.braking{i} = sectionBrakingFun(brakingTerminal,section.EndS(i,:));

end

end

function sectionBraking=sectionBrakingFun(brakingTerminal,sectionEndS)

index1 = find(brakingTerminal(:,2)<sectionEndS(1),1);

index2 = find(brakingTerminal(:,2)<sectionEndS(2),1);

if isempty(index1)

if isempyt(index2)

sectionBraking = brakingTerminal;

else

sectionBraking = brakingTerminal(1:index2,:);

sectionBraking(end,2) = sectionEndS(2);

end

else

if isempty(index2)

sectionBraking = brakingTerminal(index1:end,:);

sectionBraking(1,1) = sectionEndS(1);

elseif index1 == index2 && sectionEndS(2)>brakingTerminal(index1,1)

sectionBraking = [];

else

sectionBraking = brakingTerminal(index1:index2,:);

sectionBraking(1,1) = sectionEndS(1);

sectionBraking(end,2) = sectionEndS(2);

end

end

end

**optimalStationAlgo.m**

function [S,V,T,F,calS,calDist,Acce,interSta,totalT,totalE,brakingTerminal,...

successFlag]=optimalStationAlgo( As,At,dwellTime,targetT,speedLimit,gradient,...

curvature,brakingCurveS,brakingCurveV,curveTerminal,stationP,deltaE)

%optimalStationAlgo 车站之间控制优化算法

% 输入参数：

% As --- 起始车站号

% At --- 终点车站号

% dwellTime --- 站点的停留时间

% targetT --- 全程的目标时间

% speedLimit --- 速度限制数据

% gradient --- 坡度数据

% curvature --- 曲率数据

% brakingCurveS --- 区间制动曲线的公里标向量

% brakingCurveV --- 区间制动曲线的速度向量

% curveTerminal --- 区间制动曲线的端点数据

% stationP --- 各站点的公里标数据

% deltaE --- 每次分配的能量

% 输出参数：

% S --- 公里标向量

% V --- 对应的速度向量

% T --- 对应的时间向量

% F --- 对应的牵引力向量

% calS --- 对应的计算公里标向量

% calDist --- 对应的计算距离向量

% Acce --- 对应的加速度向量

% interSta --- 站间数据

% totalT --- 全程消耗的总时间

% totalE --- 全程消耗的总能量

% brakingTerminal --- 全程的制动区间端点数据

A = As:At; % 起始车站到终点车站的向量

N = length(A) - 1; % 站间区间的个数

% 区间数据初始化

section.S = cell(0); % 区间的公里标向量

section.V = cell(0); % 区间的速度向量

section.T = cell(0); % 区间的时间向量

section.F = cell(0); % 区间的牵引力向量

section.braking = cell(0); % 区间制动区间

section.EndS = []; % 区间的端点公里标

section.EndV = []; % 区间的端点速度

section.SpeedLimit = []; % 区间的速度限制

section.E = []; % 区间的能量消耗

section.UsedT = []; % 区间的花费时间

interStaSectionNum = zeros(N,1); % 站间区间的子区间数目

% 计算两个站间的初始解，将所有站间初始解的区间信息保存到section中

for i = 1:N

[ ~,~,~,~,...

~,~,~,tempSection ] = ...

primarySolutionFun( stationP(A(i)),stationP(A(i+1)),speedLimit,gradient,...

curvature, brakingCurveS,brakingCurveV,curveTerminal );

interStaSectionNum(i) = length(tempSection.S); % 保存站间的子区间数目

section.S = [section.S;tempSection.S];

section.V = [section.V;tempSection.V];

section.T = [section.T;tempSection.T];

section.F = [section.F;tempSection.F];

section.braking = [section.braking;tempSection.braking];

section.EndS = [section.EndS;tempSection.EndS];

section.EndV = [section.EndV;tempSection.EndV];

section.UsedT = [section.UsedT;tempSection.usedT];

section.SpeedLimit = [section.SpeedLimit;tempSection.SpeedLimit/3.6]; % 将km/h转换为m/s

section.E = [section.E;tempSection.E];

end

% 对所有的区间进行优化

sectionNum = length(section.S); % 总的区间数目

totalT = sum(section.UsedT); % 路程消耗的总时间

firstFlag = 1; % 判断是否是第一次循环

saveTimePerE = zeros(sectionNum,1); % 区间的单位能量节省时间向量

previousT = 0;

successFlag = 1;

while totalT>targetT

% 如果是第一次循环，对所有的区间计算单位能量节省时间

if firstFlag == 1

for i = 1:sectionNum

[ ~,~,~,~,tempSectionTotalT,tempSectionLeftE ] = optimalSectionAlgo( section.EndS(i,1),...

section.EndS(i,2),section.EndV(i,1),section.EndV(i,2),section.E(i)+deltaE,section.SpeedLimit(i),gradient,curvature );

saveTimePerE(i) = (section.UsedT(i) - tempSectionTotalT) / (deltaE - tempSectionLeftE);

end

else % 如果不是第一次循环，仅计算更新过的区间的单位能量节省时间

[ ~,~,~,~,tempSectionTotalT,tempSectionLeftE ] = optimalSectionAlgo( section.EndS(i,1),...

section.EndS(i,2),section.EndV(i,1),section.EndV(i,2),section.E(i)+deltaE,section.SpeedLimit(i),gradient,curvature );

saveTimePerE(i) = (section.UsedT(i) - tempSectionTotalT) / (deltaE - tempSectionLeftE);

end

firstFlag = 0; % 第一次循环后，firstFlag为0

[~,i]=max(saveTimePerE); % 选取单位能量节省时间最多的区间

% 对选中的区间进行优化

[ section.S{i},section.V{i},section.T{i},section.F{i},section.UsedT(i),tempSectionLeftE,section.braking{i} ] = optimalSectionAlgo( section.EndS(i,1),...

section.EndS(i,2),section.EndV(i,1),section.EndV(i,2),section.E(i)+deltaE,section.SpeedLimit(i),gradient,curvature );

section.E(i) = section.E(i) + deltaE - tempSectionLeftE;

% 重新计算路程总时间

totalT = sum(section.UsedT);

if abs(totalT-previousT)<0.001

successFlag = 0;

break

end

previousT = totalT;

disp(totalT)

end

interSta =section2interSta(section,interStaSectionNum);

[S,V,T,F,calS,calDist,Acce,totalT,totalE,brakingTerminal] = interSta2all(interSta,dwellTime,A,stationP);

end

**optimalSectionAlgo.m**

function [ S,V,T,F,totalT,E,brakingTerminal ] = optimalSectionAlgo( s0,s1,v0,vt,E,speedLimit,gradient,curvature )

%optimalSectionAlgo 区间最优控制算法

% 输入参数：

% s0 --- 初始公里标

% s1 --- 终止公里标

% v0 --- 初始速度 单位：m/s

% vt --- 终止速度 单位：m/s

% E --- 消耗能量

% speedLimit --- 速度限制矩阵

% gradient --- 轨道的坡度

% curvature --- 轨道的曲率

% 输出参数：

% S --- 公里标向量

% V --- 对应的速度向量

% T --- 对应的时间向量

% F --- 对应的牵引力向量

% totalT --- 消耗的总时间

% E --- 剩余的能量

% brakingTerminal --- 制动曲线端点数据

M = 194295; % 列车质量 kg

L = 10000;

S = linspace(s0,s1,L);

V = zeros(size(S));

F = zeros(size(S));

brakingTerminal = [];

% 终点制动曲线

[ endBrakingCurveV,endBrakingCurveS ] = brakingCurveFun( s0, s1, vt\*3.6, gradient, curvature );

% 设置初始速度

V(1) = v0;

% 牵引阶段

i = 2;

while (i < length(V) && V(i - 1) < speedLimit && E > 0)

[ Fmax ] = maxTractionFun( V(i - 1) );

[W] = totalResistanceFun(V(i - 1), S(i-1), gradient, curvature);

capacityMaxA = (Fmax - W) / M; % 能够达到的最大加速度

if capacityMaxA > 1

a = 1; % 实际加速度, 因为题目限制最大加速度不能超过1

totalF = M \* a; % 合力

F(i - 1) = totalF + W; % 实际牵引力

else

a = capacityMaxA;

F(i - 1) = Fmax;

end

V(i) = sqrt((V(i-1))^2 + 2 \* a \* (S(i-1) - S(i)));

if checkMeetEndBrakingCurve(S(i),V(i),endBrakingCurveS,endBrakingCurveV)

V(i-1:end) = interp1(endBrakingCurveS,endBrakingCurveV,S(i-1:end),'pchip');

brakingTerminal = [brakingTerminal;S(i-1),S(end)];

i = length(V);

else

E = E - F(i - 1) \* (S(i-1) - S(i));

i = i + 1;

end

end

% 巡航阶段

while (i < length(V) && E > 0)

V(i) = V(i - 1);

if checkMeetEndBrakingCurve(S(i),V(i),endBrakingCurveS,endBrakingCurveV)

V(i-1:end) = interp1(endBrakingCurveS,endBrakingCurveV,S(i-1:end),'pchip');

brakingTerminal = [brakingTerminal;S(i-1),S(end)];

i = length(V);

else

[W] = totalResistanceFun(V(i - 1), S(i-1), gradient, curvature);

E = E - W \* (S(i-1) - S(i));

i = i + 1;

end

end

% 惰行阶段

while (i <= length(V))

[W] = totalResistanceFun(V(i - 1), S(i-1), gradient, curvature);

a = -W / M; % 惰行加速度

if a < -0.04

a = -0.04;

realF = M \* a;

F(i - 1) = realF + W;

end

V(i) = sqrt((V(i-1))^2 + 2 \* a \* (S(i-1) - S(i)));

E = E - F(i - 1) \* (S(i-1) - S(i));

if checkMeetEndBrakingCurve(S(i),V(i),endBrakingCurveS,endBrakingCurveV)

V(i-1:end) = interp1(endBrakingCurveS,endBrakingCurveV,S(i-1:end),'pchip');

brakingTerminal = [brakingTerminal;S(i-1),S(end)];

i = length(V)+1;

else

i = i + 1;

end

end

diffS = abs(diff(S));

meanV = mean([V(1:end-1);V(2:end)]);

T = cumsum([0,diffS ./ meanV]);

totalT = T(end);

end

**第二问主要源代码：**

**problem2.m**

% 第二问脚本

clear

addpath ../problem1

readData

load brakingCurve.mat

As = 1;

At = 14;

targetTime = 2086 - 75/2\*12; % 1636

dwellTime = ones(1,12)\*75/2;

tic

deltaE = 0.03 \* 1000 \* 3600; % deltaE 为 0.1 千瓦时

[S,V,T,F,calS,calDist,Acce,interSta,totalT,totalE,brakingTerminal]=...

optimalStationAlgo( As,At,dwellTime,targetTime,speedLimit,gradient,...

curvature,brakingCurveS,brakingCurveV,curveTerminal,stationP,deltaE);

toc

tempIndex = find(diff(T)==0);

if ~isempty(tempIndex)

S(tempIndex) = [];

V(tempIndex) = [];

T(tempIndex) = [];

F(tempIndex) = [];

calS(tempIndex) = [];

calDist(tempIndex) = [];

Acce(tempIndex) = [];

end

Time = 0:floor(T(end));

N = length(Time);

resultTable = zeros(N+1,9);

resultTable(1:N,1) = Time; % 时刻

secondV = interp1(T,V,Time);

resultTable(1:N,2) = secondV \* 100; % 实际速度(cm/s)

resultTable(1:N,3) = secondV \* 3.6; % 实际速度(km/h)

resultTable(1:N,4) = interp1(T,Acce,Time); % 计算加速度(m/s2)

resultTable(1:N,5) = interp1(T,calDist,Time); % 计算距离(m)

resultTable(1:N,6) = interp1(T,calS,Time); % 计算公里标(m)

secondS = interp1(T,S,Time); % 实际公里标(m)

for i = 1:N-1

[ ~,~,resultTable(i,7) ] = groundConditionFun( secondS(i),gradient,curvature );

end

secondF = interp1(T,F,Time); % 计算牵引力(N)

resultTable(1:N,8) = secondF;

resultTable(1:N,9) = secondF .\* secondV;

temp = (stationP(As) - stationP(At)) - resultTable(end-1,6);

resultTable(end,:) = [targetTime,0,0,0,resultTable(end-1,5)+temp,resultTable(end-1,6)+temp,resultTable(end,7),0,0];

Timestr = cell(N+1,1);

for i = 1:N+1

Timestr{i} = second2Time( i - 1 );

end

**P2subProblem1.m**

% 第二问第二小问脚本

% 计算制动与牵引数据

[ brakingTimeSection,brakingSectionEnergy,tractionSection ] = brakingTractionFun( brakingTerminal,S,T,F,brakingCurveS,brakingCurveEreg );

H = 2\*60 + (11-2)\*60\*rand(1,99);

tic

[ objVelue ] = objFun( H,brakingTimeSection,brakingSectionEnergy,tractionSection );

toc

gaObjFun = @(x)objFun( x,brakingTimeSection,brakingSectionEnergy,tractionSection );

Aeq = ones(1,99);

beq = 63900;

LB = 2\*60\*ones(1,99);

UB = 11\*60\*ones(1,99);

PopInitRange = [2\*60;11\*60];

PopulationSize = 20;

nvars = 99;

[x,fval,exitflag,output,population,score] = subProblem1GA(gaObjFun,nvars,Aeq,beq,LB,UB,PopInitRange,PopulationSize);

**P2subProblem2.m**

% 第二问第二小问脚本

% 计算制动与牵引数据

[ brakingTimeSection,brakingSectionEnergy,tractionSection ] = brakingTractionFun( brakingTerminal,S,T,F,brakingCurveS,brakingCurveEreg );

H = 2\*60 + (11-2)\*60\*rand(1,240-1);

tic

[ objValue ] = objFun2( H,brakingTimeSection,brakingSectionEnergy,tractionSection );

toc

bestObjValue = 0;

for j = 1:1000

H = zeros(1,240-1);

s = 1.5;

H(1) = 5\*60 + rand(1) \* (s\*60);

cumH = H(1);

for i = 2:(240-1-10)

if (cumH>7200 && cumH<12600) || (cumH>43200 && cumH<50400)

H(i) = 2\*60 + rand(1) \* (0.5\*60);

cumH = cumH + H(i);

else

H(i) = 5\*60 + rand(1) \* (s\*60);

cumH = cumH + H(i);

end

end

targetT = 63900;

temp = (targetT - sum(H))/10;

H(240-10:end) = temp;

[ objValue ] = objFun2( H,brakingTimeSection,brakingSectionEnergy,tractionSection );

if temp > 300 && objValue < bestObjValue

bestH = H;

bestObjValue = objValue;

end

end

**objFun.m**

function [ objVelue ] = objFun( H,brakingTimeSection,brakingSectionEnergy,tractionSection )

%objFun 第二问第一小问目标函数

% 此处显示详细说明

cumH = cumsum(H);

% 合并牵引区间

for i = 1:length(H)

tractionSection = mergeTractionSection(tractionSection,tractionSection+cumH(i));

end

objVelue = saveEnergyFun(brakingTimeSection,brakingSectionEnergy,tractionSection);

% 计算节约能量

for i = 1:length(H)

objVelue = objVelue + saveEnergyFun(brakingTimeSection+cumH(i),brakingSectionEnergy,tractionSection);

end

objVelue = -objVelue;

disp(objVelue)

end

**第三问主要源代码：**

**simulation.m**

function [ result ] = simulation( n,interSta,speedLimit,gradient,curvature,brakingCurveS,brakingCurveV,curveTerminal,stationP )

%simulation 仿真函数

% 此处显示详细说明

delayTime = zeros(n,1);

lateTime = zeros(n,1);

delaytotalE = zeros(n,1);

delayType = randsrc(n,1,[0,1,2;0.7,0.2,0.1]);

fileID = fopen('simResult.txt','a');

for i = 1:length(delayType)

if delayType(i)==0

delayTime(i) = 0;

elseif delayType(i)==1

delayTime(i) = rand(1)\*10;

else

delayTime(i) = 10 + rand(1)\*(60-10);

end

end

delaySta = randi([2,12],n,1);

for i = 1:n

if delayTime(i) == 0

lateTime(i) = 0;

delaytotalE(i) = 0;

else

[ ~,~,~,~,~,~,~,~,delaytotalE(i),~,lateTime(i) ] =...

delayFun( delaySta(i),delayTime(i),interSta,speedLimit,gradient,curvature,...

brakingCurveS,brakingCurveV,curveTerminal,stationP );

end

fprintf(fileID,'%f\t%d\t%f\t%f\t%f\n',delaySta(i),delayType(i),delayTime(i),lateTime(i),delaytotalE(i))

end

result = [delaySta,delayType,delayTime,lateTime,delaytotalE];

end

**delayFun.m**

function [ S,V,T,F,calS,calDist,Acce,totalT,totalE,brakingTerminal,lateTime ] =...

delayFun( delaySta,delayTime,interSta,speedLimit,gradient,curvature,...

brakingCurveS,brakingCurveV,curveTerminal,stationP )

%delayFun 延迟调整函数

A = 1:14;

dwellTime = ones(1,12)\*75/2;

deltaE = 0.001 \* 1000 \* 3600; % deltaE 为 0.1 千瓦时

lateTime = 0;

mayDelay = delaySta:(14-1);

delayTimeVec = zeros(size(mayDelay));

for i = 1:length(mayDelay)

aveDelay = delayTime / i;

delayTimeVec(1:i) = aveDelay;

successFlag = ones(1,length(delayTimeVec));

tempInterSta = cell(1,length(delayTimeVec));

for k = 1:length(delayTimeVec)

if delayTimeVec(k)~=0

targetT = interSta{mayDelay(k)}.UsedT - delayTimeVec(k);

[~,~,~,~,~,~,~,tempInterSta{k},~,~,~,successFlag(k)]=optimalStationAlgo( mayDelay(k),mayDelay(k)+1,dwellTime,targetT,speedLimit,gradient,...

curvature,brakingCurveS,brakingCurveV,curveTerminal,stationP,deltaE);

end

end

if all(successFlag)

for j = 1:i

interSta{mayDelay(j)} = tempInterSta{j}{1};

dwellTime(mayDelay(j)-1) = dwellTime(mayDelay(j)-1) + delayTimeVec(j);

lateTime = delayTime - delayTime/i;

end

break

end

end

[S,V,T,F,calS,calDist,Acce,totalT,totalE,brakingTerminal] = interSta2all(interSta,dwellTime,A,stationP);

end