*Aルート仕様書*

第２版

*２０２１年１月２８日*

株式会社スタットラボ

Aルート　仕様書（第２版）

2021.1.28

■関数：a\_route\_m

機能：

各時点の計測値から下限直線及び上限直線を求め、Aルート計測値から、実際の電力需要を推定する。

後述する以下の関数の機能を含む。

ul\_line

a\_route

使用法：

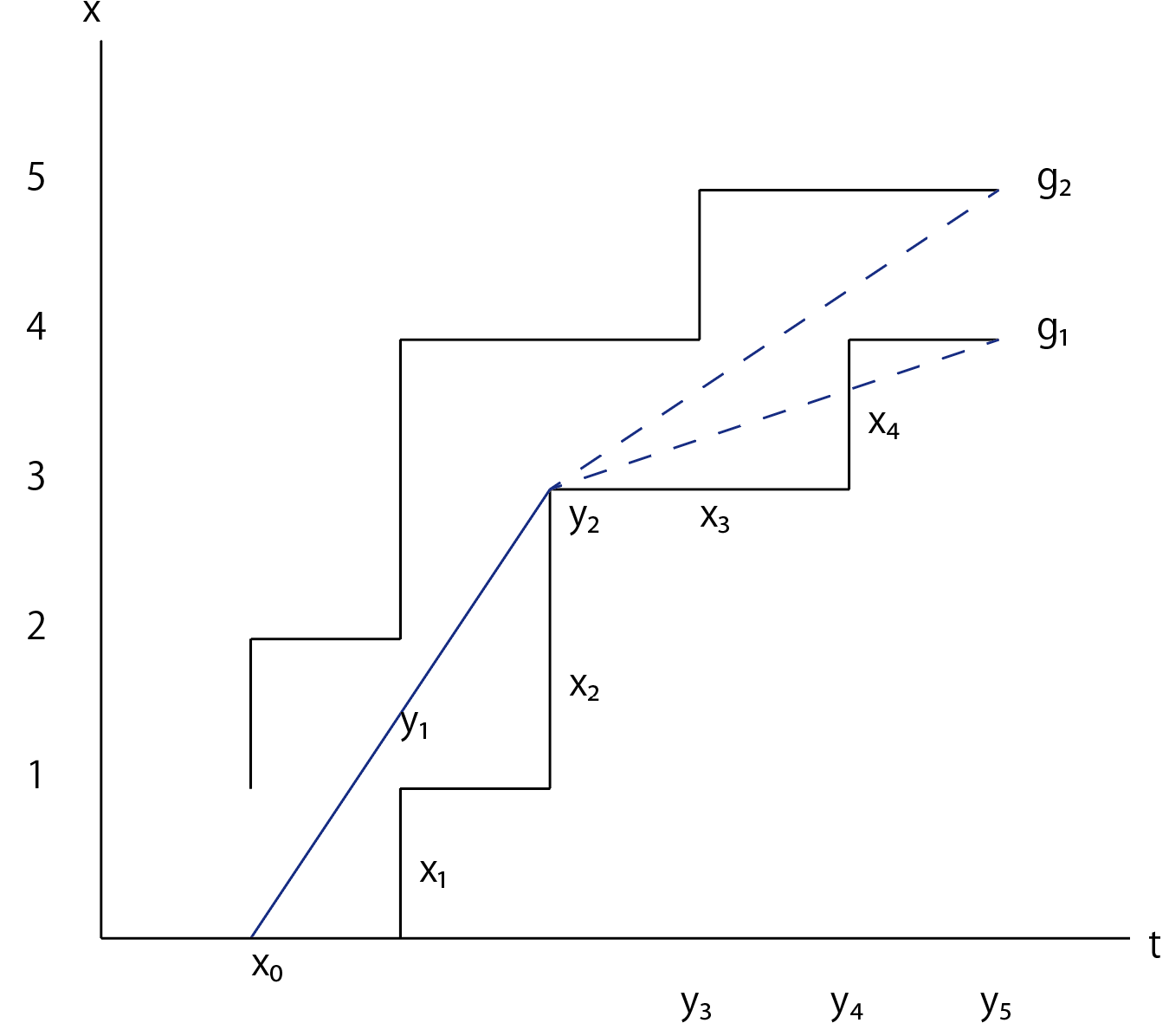
x,g1,g2,t,tpos,y,ycum = a\_route\_m(data1,p)

パラメータ：

data：n時点分の計測値（kw/h）

p : 最後の累積需要曲線が決まらない場合、yの値を決定するための設定

以下のよう時点が5までしか場合、時点t=6が決まっていないので、最後の累積需要曲線が決定できない。



y3〜y5の値が決まらないので、パラメータpの値によりｙの値は以下のように出力する。

p=0 : 欠測値、指定しない場合は0

p=1 : 下限値

P=2 : 上限値

P=3 : 下限値と上限値との中間値

出力：

　 x ：x軸の値（0,1,2,...,n）が入る。

g1：需要下限（１時点を追加し０時点として、計測値を１０倍する。）

g2：需要上限（１時点を追加し０時点として、計測値に０.１を加え１０倍する。）

t：累積需要曲線が需要上限または下限と接した時点。

tpos：累積需要曲線が需要上限と接した場合は２、下限と接した場合は１が入る。

y：推定需要。

ycum：yの合計値。

以下のように呼び出す。

|  |
| --- |
| data1 = [1.2,1.3,1.4,1.8,1.8,1.9,2.2,1.9,1.9,1.7,1.6]  x,g1,g2,t,tpos,y,ycum = a\_route\_m(data1,2)  print(t)  print(tpos)  print(g1)  print(g2)  print(y)  graphA(x,g1,g2,t,tpos) |

結果は以下のように、値は右から左に逆に得られる。時点 t=3 で累積需要曲線は下限と接し、時点t=11 で累積需要曲線は上限と接している。

[9, 7, 6, 3]

[1, 1, 1, 1]

[0.0, 12.0, 25.0, 39.0, 57.0, 75.0, 94.0, 116.0, 135.0, 154.0, 171.0, 187.0]

[1.0, 13.0, 26.0, 40.0, 58.0, 76.0, 95.0, 117.0, 136.0, 155.0, 172.0, 188.0]

[ 0. 13. 13. 13. 18.33333333 18.33333333　 18.33333333 22. 19. 19. 18. 16. ]

アルゴリズムについては、関数a\_routeを参照のこと。通常、Aルートの計算を行いグラフを表示するには、この関数a\_route\_mと関数graphAだけでよい。

■関数：graphA

機能：

Aルート計測値と推定された累積需要曲線のグラフを描く。

使用法：

graphA(x,g1,g2,t,tpos)

パラメータについて：

x：x軸の値。0、1、2、. . . .、n

g1：需要下限（単位は100w/h、n’=n+1時点分の値）

g2：需要上限（単位は100w/h、n+1時点分の値）

t：関数a\_routeの出力。

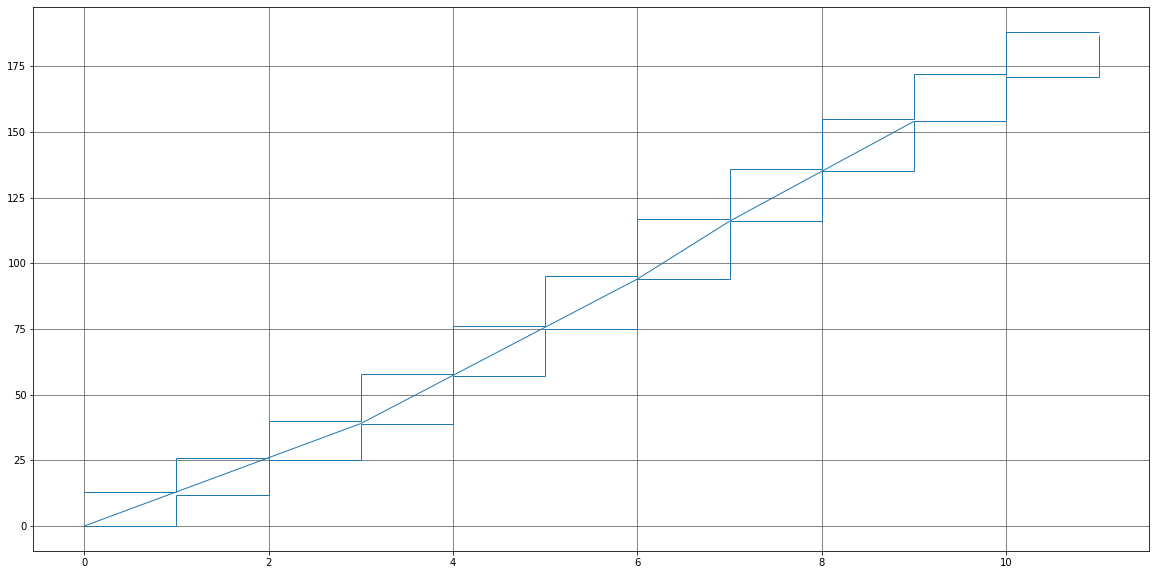
tpos：関数a\_routeの出力。

各パラメータは関数a\_route\_mの結果を編集せずにそのまま使うことが必要である。

例：

|  |
| --- |
| data1 = [1.2,1.3,1.4,1.8,1.8,1.9,2.2,1.9,1.9,1.7,1.6]  x,g1,g2,t,tpos,y,ycum = a\_route\_m(data1,2)  print(y)  graphA(x,g1,g2,t,tpos) |

以下のようなグラフが描画できる。



データとの対応を見てみる。

print(y)

[ 0. 13. 13. 13. 18.33333333 18.33333333

18.33333333 22. 19. 19. 18. 16. ]

■関数：ul\_line

機能：

各時点の計測値から下限直線及び上限直線を求める。

使用法：

x,g1,g2 = ul\_line(data)

パラメータ：

data：n時点分の計測値（kw/h）

出力：

　 x ：x軸の値（0,1,2,...,n）が入る。

g1：需要下限（１時点を追加し０時点として、計測値を１０倍する。）

g2：需要上限（１時点を追加し０時点として、計測値に０.１を加え１０倍する。）

以下の場合はエラーとする：

データの件数が３未満：

エラーを表示してExceptionで停止する。

データの値が全て0または負（0.1よりも小さい値も0とする）：

エラーを表示してExceptionで停止する。

データに欠即値が含まれる：

エラーを表示してExceptionで停止する。

例：

print(data)

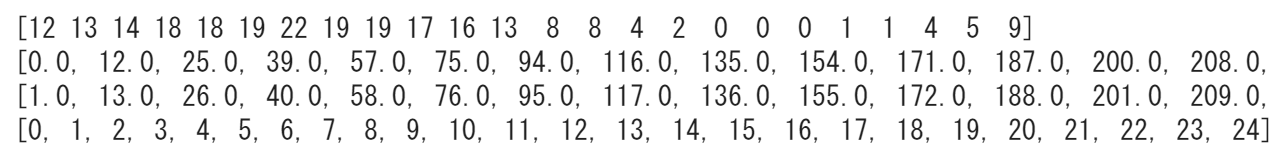
x,g1,g2 = ul\_line(data)

print(g1)

print(g2)

print(x)

結果は以下のようになる。ただし、g1とg2の表示は途中で切れている。



■関数：a\_route(pos,ts,te,g1,g2,p)

機能：

Aルート計測値から、実際の電力需要を推定する。

使用法：

a\_route(pos,ts,te,g1,g2,p)

パラメータ（入力）：

関数a\_routeは再帰呼び出しにより、自分自身を繰り返し呼び出すため、初期値を与える。

pos：下限と接する場合は１、上限と接する場合は２で、最初は１を与える。

ts：累積需要曲線の始点として最初に０を与える。

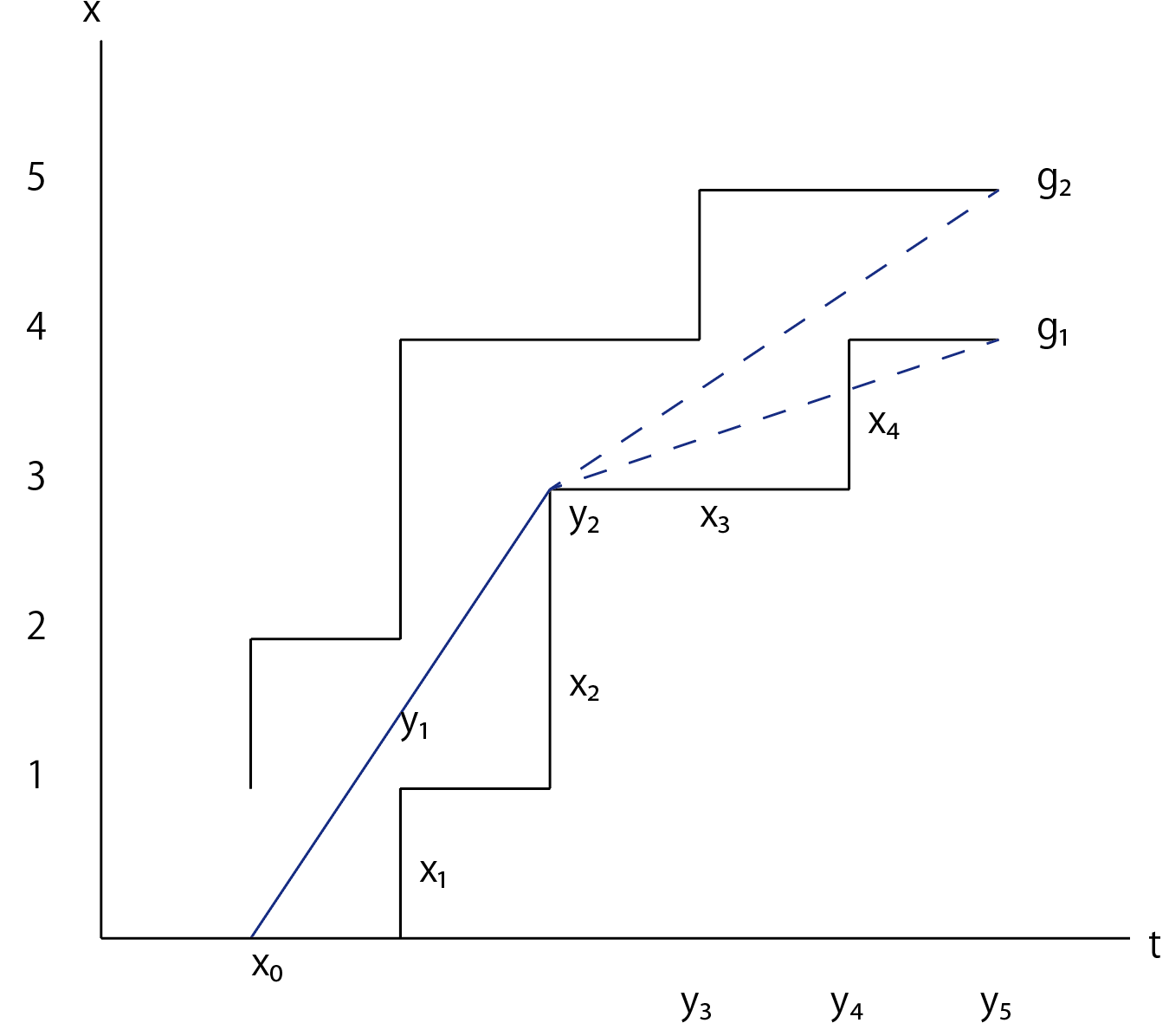
te：累積需要曲線の始点の２つ先の点として最初に２を与える。

g1：需要下限（単位は100w/h、n’=n+1時点分の値）

g2：需要上限（単位は100w/h、n+1時点分の値）

p : 最後の累積需要曲線が決まらない場合、yの値を決定するための設定

以下のよう時点が5までしか場合、時点t=6が決まっていないので、最後の累積需要曲線が決定できない。



y3〜y5の値が決まらないので、パラメータpの値によりｙの値は以下のように出力する。

p=0 : 欠測値

p=1 : 下限値

P=2 : 上限値

P=3 : 下限値と上限値との中間値

global変数として以下の変数を定義しておく必要がある。結果はこの変数に格納される。

t：累積需要曲線が需要上限または下限と接した時点。

tpos：累積需要曲線が需要上限と接した場合は２、下限と接した場合は１が入る。

y：推定需要

ycum：yを計算するときのワークエリア

　　　結果出力：

以下のglobal変数に結果が入る。

t：累積需要曲線が需要上限または下限と接した時点。

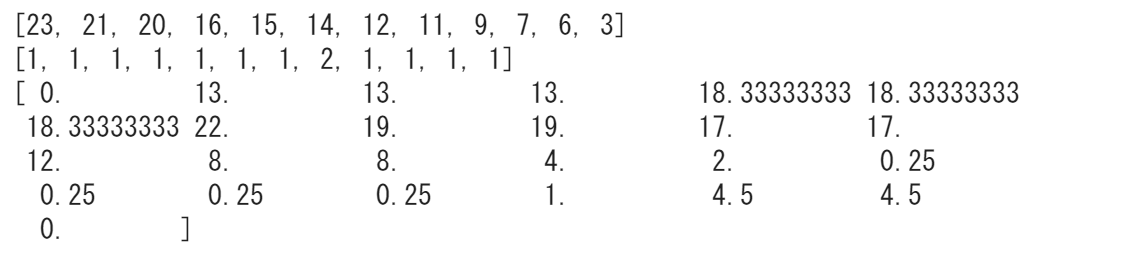
tpos：累積需要曲線が需要上限と接した場合は２、下限と接した場合は１が入る。

y：推定需要

以下のように呼び出す。

|  |
| --- |
| t = []  tpos = []  ycum = 0.0  y = np.zeros(len(g1))  a\_route(1,0,2,g1,g2)  print(t)  print(tpos)  print(y) |

結果は以下のように、値は右から左に逆に得られる。時点 t=3 で累積需要曲線は下限と接し、時点t=11 で累積需要曲線は上限と接している。



関数が呼び出された最初の時点では以下のようになっている。最初の累積需要曲線は始点がts=0、終点がte=2になっている。累積需要曲線の候補f1とf2はそれぞれ需要下限と需要上限に時点teで接している。f1またはf2の一方または両方が前の時点で上限及び下限からはみ出していなければteの時点を１増やして、f1またはf2がはみ出さないか調べる。

グラフ

自動的に生成された説明

アルゴリズム：

関数a\_routeは初期値からスタートして、teが時点数n’に至るまで再帰的に呼び出される。

例えば、終点teの時点を進めていき、最初に以下のようにf1とf2の両方が上限及または下限からはみ出したとする。

グラフ

自動的に生成された説明

１時点前では少なくとも片方がはみ出していなかったので、１時点前の状態を見てみる。

f2がはみ出していないので時点３では、上限の方に累積需要曲線が接することになる。ここで、tには３、tposには２が加えられる。

グラフ, 図形, 四角形

自動的に生成された説明

次にtsとteを入れ替えて、a\_route関数を呼び出す。

ts = te

te = ts + 2

四角形 が含まれている画像

自動的に生成された説明

関数の中の処理は以下のようになる。

|  |
| --- |
| teがデータの長さ以上の場合は終了。  return  te>ts+1かどうかの確認し、  te=<ts+1であれば、teを増やす。  下限直線f1のf1[te-1]、上限直線f2のf2[te-1]を計算する。  ２点を通る直線の式より計算できる。  起点が下の場合：  f1x = (g1[te]-g1[ts])\*(te-i)/(te-ts) + (ts\*g1[te]-te\*g1[ts])/(ts-te)  f2x = (g2[te]-g1[ts])\*(te-i)/(te-ts) + (ts\*g2[te]-te\*g1[ts])/(ts-te)    起点が上の場合：  f1x = (g1[te]-g2[ts])\*(te-i)/(te-ts) + (ts\*g1[te]-te\*g2[ts])/(ts-te)  f2x = (g2[te]-g2[ts])\*(te-i)/(te-ts) + (ts\*g2[te]-te\*g2[ts])/(ts-te)  以下の４つの場合に分けて再帰呼び出しをする。実際ははじめの３つの場合は同じなので一つにまとめている。  前の時点で下限直線も上限直線も接触しない。  　　teを1つ進める。  a\_route(pos,ts,te+1,g1,g2,t1,t2)  下限直線が下に接触。  　teを1つ進める。  a\_route(pos,ts,te+1,g1,g2,t1,t2)  上限直線が上に接触。  　teを1つ進める。  a\_route(pos,ts,te+1,g1,g2,t1,t2)  上も下も接触する。  1つ前の時点で、上か下のどちらが接触していないか確認し、起点を設定する。  　　a\_route(pos,ts,te,g1,g2,t1,t2) |