1. 問題

地点Aから地点Bまで移動するときの最短経路を求めよ.

なお、川の幅を130(m),地点Aと川までの最短距離は40(m),地点Bと川までの距離を0(m),地点Aと地点Bの川に平行な距離は310(m)とする.また,川の流れには流れがあり,その速さは0.8(m/s)および4.0(m/s)とする.陸上の移動は徒歩とし、速度は1.11(m/s),船の速度は静水中で7(m/s)とする.このとき,川を渡る時間が最小となる時間を求め,また,そのときの川を渡り始める場所,点Pを求めよ.問いは以下の手順で答えよ.

問1 まず直感で解く.問2 エクセルで解く.問3 エクセル以外の好きな方法で解く.

1. 流れがない場合の解法と結果
   1. 直感で考えた場合

Aから岸まで垂直に直線で進んだ後,船で直線的にBまで行けばよいと考えたが,講義中に考えた.三木教授に話したところ,岸まで斜め横断した場合を排除した考えになっていると指摘され,考えを改めることになった.

* 1. エクセルで解いた場合

2.1での直感が誤っていたことを考慮した結果,Aから岸におろした垂線を仮にQとし,実際に渡り始める地点をPとして,PQの幅を逐次的に大きくした後,PからBまで直線で進むことにした.実際のexcelで試した結果の一部を図1に示す.

また,それぞれのセルで行われている演算式を表1にまとめる.PQの長さを増やし続ける試行を1600回行った.また最小値を示すセル番号は596であり,その時のPQの長さは5.90であった.

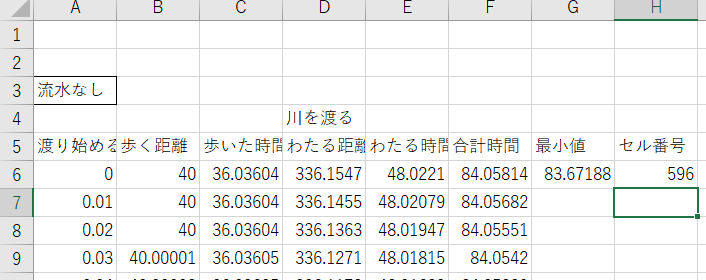


図1 エクセル実行画面(著者作成)

|  |  |  |
| --- | --- | --- |
| セル番号 | 名称 | 演算式 |
| A5 | PQ | =A6+0.01 (初期値A6=0としている) |
| B5 | 歩く距離 | =SQRT(40^2+A6^2) |
| C5 | 歩いた時間 | =B6/$I$2 |
| D5 | 渡る距離 | =SQRT((310-A6)^2+130^2) |
| E5 | 渡る時間 | =D6/$I$1 |
| F5 | 合計時間 | =C6+E6 |
| G5 | 最小値 | =MIN(F6:F1021) |
| H5 | 最初値のあるセル番号 | =MATCH(G6,F1:F1600,0) |

表1 セル番号,名称及び演算式(著者作成)

* 1. Pythonを使った場合

今回はJupyter Notebookを用いてPythonを使ったアルゴリズムを作った.まずJupyter Notebookはオープンソースウェブアプリケーションで,作ったプログラムを実行し,記録できるツールである.プログラムに関しては対話的に逐次実行できることも魅力である.またプログラム内容や実行結果はnotebookという形式で保存され,シェアすることも可能である[1].今回は,プログラムの主要箇所を図2に示す.次に何を実現したかを説明する.エクセルで解いたのと同様にPQの幅を変数startで表し,for文で1500回繰り返し0.01を加え続けている.最終的には

min=np.min(arr\_walk\_time+arr\_acrsriv\_time)で歩いた時間と川を渡った時間とを足したアレイの中から最小値を出力し,numpyのライブラリのnp.min()を用いて最小値の入っているインデックスを抜き出している.その結果を表2にまとめる.表2に示すようにインデックスの値がexcelの結果と一致しなくさらなる検討が必要そうだ.

|  |
| --- |
| start=0#渡り始めるところ  walk\_time\_data=[]#川を渡った時間のデータ  acrs\_riv\_time\_data=[]#歩いた時間のデータ  for i in range(1500):  start+=i\*0.01  dis\_walk=math.sqrt(start\*\*2+40\*\*2)  walk\_time=dis\_walk/v\_peo[0]#v\_peo[0]=1.11  walk\_time\_data.append(walk\_time)  dis\_acrs\_riv=math.sqrt((AB-start)\*\*2+width\_river\*\*2)  acrs\_time=dis\_acrs\_riv/v\_peo[1] #v\_peo[0]=7.0  acrs\_riv\_time\_data.append(acrs\_time)  arr\_walk\_time=np.array(walk\_time\_data)  arr\_acrsriv\_time=np.array(acrs\_riv\_time\_data)  min=np.min(arr\_walk\_time+arr\_acrsriv\_time)  index\_min=np.argmin(arr\_sum) |

図2Pythonでの主な計算部分(著者作成)

|  |  |
| --- | --- |
| 変数名 | 結果 |
| min | 83.67191380997302 |
| index\_min | 34 |

表2結果(著者作成)

3 流れがある場合の解放と結果

3.1 直感で考えた場合

陸を進むには流れがない場合と同様に考えたのち川を進む場合は流速分,船の進む速さの縦成分から引けばよいと考えた.

3.2 エクセルで解いた場合

先に船の進む角度(θ)を変数として川に垂直な速度成分を求め,川を横切る時間(仮にtとする)を求めた.その後,時間t内にどれだけ川に対して平行に進んだかを計測し,その差分とAから川までの距離とで進むべき距離出し,計測する方法をとった.以下にexcelで行った各セルの演算式を表3に示す.また結果は流速0.8の時,89.27191(s),セル番号1217となりPは1.211(m)地点,また,流速4.0の時は140.786(s)かかり1.388にPがあることが分かった.

|  |  |  |
| --- | --- | --- |
| セル番号 | 名称 | 演算式 |
| P5 | θ | =P6+0.001 (初期値P6=0としている) |
| Q5 | X成分 | =$I$1\*COS(P6) (但し, I1=7.0) |
| R5 | Y成分 | =$K$1\*SIN(P6) |
| S5 | Y成分-流速 | =R6-$N$1 (但し,N1は川の流れ) |
| T5 | 横切る時間 | =$R$1/Q6 (R1=130) |
| U5 | 進んだ縦の距離 | =S6\*T6 |
| V5 | 徒歩で縦に進む距離 | =$R$2-U6 |
| W5 | 徒歩で進むべき距離 | =SQRT(40^2+V6^2) |
| X5 | 徒歩時間 | =W6/$K$2 |
| Y5 | 合計時間 | =X6+T6 |
| Z5 | 最小値 | =MIN(Y6:Y1576) |
| AA5 | 最小値のあるセル | =MATCH(Z6,Y1:Y1576,0) |

表3 excel実行画面のセルと使った演算式の対応表(著者作成)

3.3 Pythonで行った場合

Pythonで行った場合の主要なアルゴリズムを考え方はexcelと同様で,川を渡るときの傾きthetaを0.001ずつ増やしていく作業をfor文を用いて1576回繰り返した.1576回行ったのは.とするのに都合が良かったからである.また配列v\_riv=[0.8,4.0]とし手入力で流速を変えられる.流速0.8のときは89.27191043954147で最小時間を取り,その値をとるインデックスは1211となりPQ間が1.211(m)を示している.流速4.0の時は140.78595271022306で最小値,その時のインデックスは1388でPQ間が1.388(m)を示している.excelの時とほぼ一致したといえる.

|  |
| --- |
| theta=0#川を渡るときの角度(θ)  walk\_time\_data=[]#歩いた時間のデータ  acrs\_riv\_time\_data=[]#川を渡った時間のデータ  for i in range(1576):  #先に船で渡る時間とその時進んだ時間を計算する  theta=i\*0.001  v\_x=v\_peo[1]\*math.cos(theta)#速度をx成分に分解したとき  v\_y=v\_peo[1]\*math.sin(theta)#速度をy成分に分解したとき  v\_y=v\_y-v\_riv[0]  acrs\_time=width\_river/v\_x#横切る時間  acrs\_riv\_time\_data.append(acrs\_time)  dis\_y=v\_y\*acrs\_time#船で縦に進んだ距離  dis\_y=AB-dis\_y#残りの距離  #歩いて進めべき時間を計算する  dis\_walk=math.sqrt(dis\_y\*\*2+dis\_A\_riv\*\*2)#歩く距離  walk\_time=dis\_walk/v\_peo[0]  walk\_time\_data.append(walk\_time)  arr\_walk\_time=np.array(walk\_time\_data)  arr\_acrs\_riv\_time=np.array(acrs\_riv\_time\_data)  arr\_sum=arr\_walk\_time+arr\_acrs\_riv\_time  min=np.min(arr\_sum)  index\_min=np.argmin(arr\_sum) |

図3 Pythonでのアルゴリズム一部(著者作成)

4. 参考文献または参考URL

[1] Jupyter Notebook, <https://jupyter.org/>