航空宇宙情報システム学第三 レポート 2

工学部航空宇宙工学科三年 野本陽平 (03-180332)

2019 年 3 月 10 日

課題 1

問題

地上のシステム (車, 家電, プラント, 家など何でもいい) の中で使われている信頼性向上の工夫を一つ取り上げて, 以下の流れで議論しなさい.

1. どのような故障モードを想定しているか？
2. その故障モードで, どうやって安全性や動作の継続性を保証しているか？
3. 現在の方策は十分か.
4. さらに信頼度を上げるための改善点はないか. あなたのアイデアを書いてください.

回答

Solution. 春休みになった. 友人と遠くへ出かけたいがいつものようにお金がない. そういう学生の心強い味方が高速バスだ. だがバスは安い反面遅延しやすいという欠点がある. (個人的には, 都会ですらバスは電車よりかなり少ないと感じることが多いので, ちょっとした時間のズレすら大きな問題であると考えている.) ここではバスが早発したり遅延したりしないようにするためにどのような方策をしているのか, また将来的にどうすべきであるのか検討した. 国民生活センターによる回答によると, バスは早発が旅客自動車運送事業運輸規則違反になってしまうため遅めに運転しているのだという.[1] (普段は時間通りに来ないと早発と遅延どちらが原因で待たされているのか分からないが, 基本は遅延らしい.) そこでバスシステムの故障モードを遅延とする.

現在は大きく分けて 2 つの試みがとられている. 一つは, 乗客に予め運行情報を知らせようという, 利用者に依存する保証だ. GPS を元に現在のバスの場所を知らせてくれるアプリなどは既に存在している.[2] これにより家を出る前にバスがきちんと動いているか確認し, もし遅延していれば代替交通手段を利用するなどすることができる. 自治体によっては電話でしかるべきところに電話をすれば運行情報を教えてくれる. もう一つは, バッファーを組んで「終点に行くほど遅れが増えていく」という状態を防ぐことだ.[3] GPS によって収集されたデータをもとに早発しないように, だがなるべく遅延が発生しないようにダイヤが組まれている. 主に以上二つの方針により運行されている.

これらの方針は満足とはいえないと思う. アプリはバスのメインユーザーと思しき高齢者にはハードルが高く, アクセスが難しい. ダイヤ改正も運転手数や新しい交通手段などの変化がある度に検討するべきであるが, 乗り入れや諸々の兼ね合いの都合上そう頻繁に改正できるものではないらしい.

ではどうすればこの状況を打開できるのだろうか. 一つには機械学習を用いて時刻表を常に最適化するということが考えられるが, 「機械学習は技術的負債の高利貸しのクレジットカード」[4] という論文があるように, 使える予算的にもあまり良くない方策に思われる. そもそも他の車や天候に左右される待ち時間を操作するのにはかなり無理があり, それこそ (運行) 情報を広く知らせるかダイヤを統計的に最適化するくらいしか方法はない. そのため発想を逆転し, 客を待たせないというよりむしろ客が待っても不

平が出ないようにするのが目的とする, UX 的な考えを用いる. 空港などでは充電スペースやフリーワイファイにより, 退屈な時間を紛らわせる工夫がなされている. これをバス停に応用し, バス停でも充電やワイファイを利用できるようにするのはどうだろうか. 高齢者で言えば近くに自販機を設置するのも (昔ながらだが) 良いかもしれない. 勿論, バス料金を先払いしないと使えないようにするなどフリーライダーを防ぐ工夫は必要だが, このようなソフトなやり方でバス運行に対する不満を減らせるのではないかと思う. (信頼性を上げるという問題の趣旨からはずれている回答の可能性があります, その場合はすみません.) ■

課題 2

問題

GPS の資料の p.191 の例題 6.5 の解答の中の間違いを指摘しなさい. また, 以下のような GPS 衛星の配置になったときの, GDOP, PDOP, VDOP, HDOP, TDOP などをそれぞれ求めなさい.

構成 1) 仰角 60 度に 180 度間隔で均等に 2 機, 仰角 30 度に 120 間隔で均等に 3 機 (仰角 60 度の衛星 1 機と同じ方位角に仰角 30 度の衛星の 1 機があるとする)

構成 2) 地平面ぎりぎり (仰角 0) に 120 度の等間隔で 3 機, 仰角 45 度に 1 機 (仰角 45 度の衛星の方位角は 0 から 120 度まで変化させ, グラフ化せよ.)

回答

誤りの指摘

Solution. まず誤りを指摘する. (6.170) 式の $\mathbf{A}$ について, 正しくは

$$\delta\mathbf{R} = \mathbf{A}\delta\mathbf{X} = \begin{bmatrix} \cos E & 0 & \sin E & 1 \\ -(1/2)\cos E & \sqrt{3/4}\cos E & \sin E & 1 \\ -(1/2)\cos E & -\sqrt{3/4}\cos E & \sin E & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \\ s \end{bmatrix}$$

である. また, 式 (6.173) から (6.175) の

$$\begin{bmatrix} \cos\left(\dfrac{E}{2}\right) - \sin\left(\dfrac{E}{2}\right) \end{bmatrix}$$

の係数は 4 でなく-4 である. ■

GDOP, PDOP, VDOP, HDOP, TDOP を求める

構成 1 について. 今利用者に対し衛星が 5 つ存在しているので, 行列 **A** は 5×4 行列になり, 以下のように書ける.

$$\begin{bmatrix} \sqrt{3}/2 & 0 & 1/2 & 1 \\ -\sqrt{3}/4 & 3/4 & 1/2 & 1 \\ -\sqrt{3}/4 & -3/4 & 1/2 & 1 \\ 1/2 & 0 & \sqrt{3}/2 & 1 \\ -1/2 & 0 & \sqrt{3}/2 & 1 \end{bmatrix}$$

これをもとに A^T と A の積の逆行列を求めそれぞれの定義に従って求めると以下の値を得る. 詳しい計算方法は配布資料に従っているほか, 末尾の Appendix にコードを載せている.

1　Answer for 構成 1

```
GDOP: 3.2439784662095983
PDOP: 2.779272959526454
TDOP: 1.6730326074756117
HDOP: 1.2264882813437332
VDOP: 2.4940097592594603
```

構成 2 について. 今利用者に対し衛星が 4 つ存在しているので, 行列 **A** は 4×4 行列になり, 以下のように書ける. ただし以下のものは 45 度の時のものである.

$$\begin{bmatrix} 1 & 0 & 0 & 1 \\ -1/2 & \sqrt{3}/2 & 0 & 1 \\ -1/2 & -\sqrt{3}/2 & 0 & 1 \\ 1/2 & 1/2 & 1/\sqrt{2} & 1 \end{bmatrix}$$

2　Answer for 構成 2 (45 度)

```
GDOP: 2.23606797749979
PDOP: 2.160246899469287
TDOP: 0.5773502691896257
HDOP: 1.1547005383792517
VDOP: 1.825741858350554
```

また, DOP の値は以下のように推移している. 全て方位角に依らない定数であることが伺える.

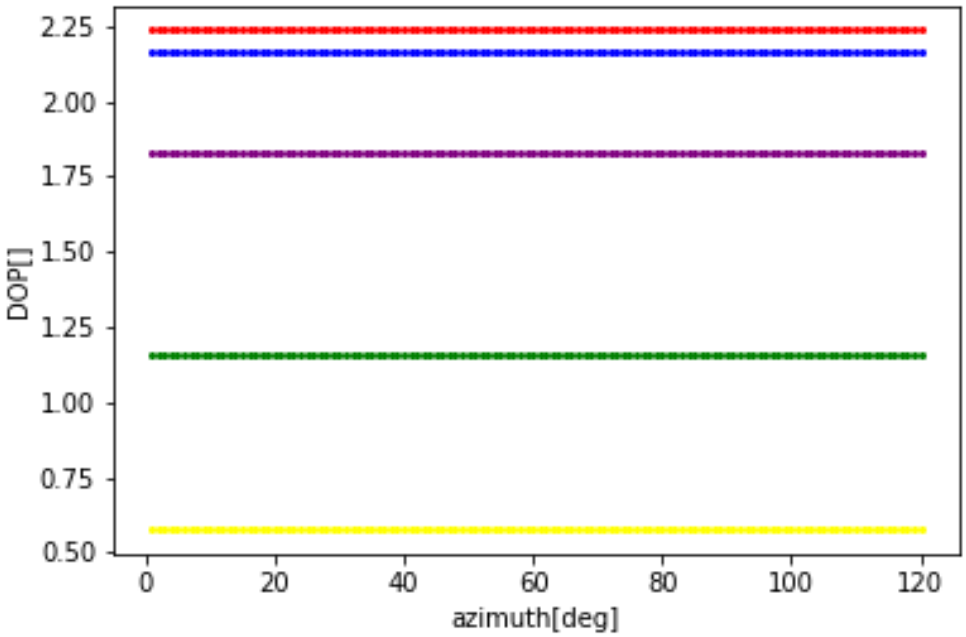

図 1　方位角毎の各 DOP の値の推移

Reference

[1] http://www.kokusen.go.jp/t box/data/t box-faq qa2017 40.html 国民生活センター

[2] http://www.fujitsu.com/jp/about/resources/case-studies/westbg/2016/2016-04.html 富士通

[3] https://driverk.com/archives/18211

[4] 「仕事ではじめる機械学習」, オライリー・ジャパン

Appendix

3　問題 2 に使用した Python コード

```
import numpy as np
import matplotlib.pyplot as plt
A = np.array([[np.sqrt(3)/2, 0, 1/2, 1],
              [-np.sqrt(3)/4, 3/4, 1/2, 1],
              [-np.sqrt(3)/4, -3/4, 1/2, 1],
              [1/2, 0, np.sqrt(3)/2, 1],
              [-1/2, 0, np.sqrt(3)/2, 1]])
M = np.linalg.inv(np.dot(A.T, A))
GDOP = np.sqrt(M[0][0]+M[1][1]+M[2][2]+M[3][3])
PDOP = np.sqrt(M[0][0]+M[1][1]+M[2][2])
TDOP = np.sqrt(M[3][3])
HDOP = np.sqrt(M[0][0]+M[1][1])
```

```
VDOP = np.sqrt(M[2][2])
print("GDOP:",GDOP)
print("PDOP:",PDOP)
print("TDOP:",TDOP)
print("HDOP:",HDOP)
print("VDOP:",VDOP)

def azimuth_change(azimuth):
    cos, sin = np.cos(azimuth*np.pi/180), np.sin(azimuth*np.pi/180)
    A = np.array([[1, 0, 0, 1],
                  [-1/2, np.sqrt(3)/2, 0, 1],
                  [-1/2, -np.sqrt(3)/2, 0, 1],
                  [cos/np.sqrt(2), sin/np.sqrt(2), 1/np.sqrt(2), 1]])
    M = np.linalg.inv(np.dot(A.T, A))
    GDOP = np.sqrt(M[0][0]+M[1][1]+M[2][2]+M[3][3])
    PDOP = np.sqrt(M[0][0]+M[1][1]+M[2][2])
    TDOP = np.sqrt(M[3][3])
    HDOP = np.sqrt(M[0][0]+M[1][1])
    VDOP = np.sqrt(M[2][2])
    return GDOP, PDOP, TDOP, HDOP, VDOP
x = np.linspace(0, 2*np.pi/3, 121)
for i in range(1,121):
    plt.plot(i, azimuth_change(i)[0], marker='o', color='red', ms=2, label='GDOP
        ')
    plt.plot(i, azimuth_change(i)[1], marker='o', color='blue', ms=2, label='PDOP
        ')
    plt.plot(i, azimuth_change(i)[2], marker='o', color='yellow', ms=2, label='
        TDOP')
    plt.plot(i, azimuth_change(i)[3], marker='o', color='green', ms=2, label='
        HDOP')
    plt.plot(i, azimuth_change(i)[4], marker='o', color='purple', ms=2, label='
        VDOP')
plt.xlabel('azimuth[deg]')
plt.ylabel('DOP[]')
plt.tight_layout
plt.savefig(fname='ref.png')
```