For office use only	Team Control Number	For office use only
T1 ________	**2017007**	F1 ________
T2 ________		F2 ________
T3 ________		F3 ________
T4 ________		F4 ________

2017

The International Mathematical Modeling Challenge (IM^{2}C) Summary Sheet

Nowadays, there are many international meetings that are being held worldwide. However, the time gap makes each participant to get tired easily. Since the meeting should be productive, the location of the meeting is very important. To find the most appropriate meeting location, we developed a mathematical model.

In our model, we included following concepts; Fatigue, Efficiency of work, Body Cycle, Sleeping and the Time Gap. First, we described the increasing rate of the fatigue with the efficiency and sleeping, where their coefficients are the function of the body cycle value. Then, we defined the body cycle as a periodic function of time. Thirdly, we made a function of efficiency by the fatigue. Using these three relationship, it was able to calculate the fatigue value at the time t.

Then, we calculated the time gap and time of flight depending on the locations. Assuming that everyone adapts to the new timeline in three days, we modified the body cycle function showing the change of the body cycle while adaptation. With the new body cycle function, it was able to calculate the fatigue during the meeting, which is held right away after the flight.

Using our model, we calculated the productivity of meeting. Then, developed two methods to find the meeting location with the maximum productivity. One method used divide-and-conquer tactics, by repeating 'zoom in' to the subpart of the map with the maximum productivity. Second method used greedy search, by calculating the productivity of the meeting at every airport. Either way, we got the most appropriate location for the given examples, and the answers from both ways were similar.

Our model describes the tiredness and the efficiency accurately, matching all the preceding model and researches about human tiredness. By using our model and algorithms, it finds the most appropriate meeting location quickly and precisely.

Table of Contents

I Introduction

Recently, there are many international meetings that are being held worldwide. Unfortunately, yet, it is seemed to be an anomaly to perfectly manage every need that engage in these conferences. In academic sense, there are so many variables to consider that it is almost impossible to determine where would be the best place to the conference, when would it be, and so on. To eliminate this anxiety, we should make an alternate solution or revamp the former one to choose the place to hold meeting that maximizes the efficiency of the meeting. The productivity of the meeting is decided by the fatigue of each participant. To maximize the productivity, it's important to control the fatigue.

So, we made a mathematical model that can estimate the fatigue of each participant. Fatigue depends on the work done and the rest taken. Using the relationship between the work done and the fatigue, relationship between fatigue and time is found.

When the participant travels to another place, the mismatch of his/her body cycle will disturb both the work and rest. By considering the time gap between home and the meeting place, it was possible to calculate the productivity through three days.

In total, our model calculates the participant's fatigue by time considering

1. **Efficiency of the work**
2. **Time of sleep**
3. **Body cycle**

And the participant's efficiency of the work is determined by

1. **The participant's fatigue**

By calculating the fatigue of each participant, it was possible to calculate the total time of productive meeting. Then, we developed the algorithm to search for the location with the maximum value of the total time of productive meeting. We applied our algorithm to the given examples, and was able to find the best place for the meeting.

II Assumptions and Justifications

Before developing the model, we made some assumptions to justify our model. Followings are the assumptions we made.

- **Assumption 1.** Every participant's schedule is identical. Sleeps at 0~8, and the meeting is done from 8~24. They act ideally.
  Justification Since each participants contributes equally, we set every participant identical except the home place.

- **Assumption 2.** It takes three days to perfectly adapt one's body cycle to the new place.
  Justification Average time that takes to get used to the new time zone is known to be 3 days.

- **Assumption 3.** The plane flies in the fastest direction, which is the great circle of the globe. Ignore the air streams.
  Justification Planes in real life actually fly near the great circle. Also, to get the average flight time, we ignored the air streams.

- **Assumption 4.** Every participants arrives at the same time, right before the meeting starts.
  Justification Since we can't let them to stay for a long time, we assumed that their staying time at the meeting area is identical.

- **Assumption 5.** In terms of each locations of the participants, we only consider the time difference as a variable.
  Justification Since the meeting is processed at indoors, other factors such as weather or climate does not affect a productivity of the meeting considerably.

Also, for the following research, we defined following variables.

- t: The time variable.
- $F(t)$: Tiredness/fatigue in scale of 0 to 1. Function of time.
- $E(t)$: Efficiency, the rate of production. The range is 0 to 1.

- $B(t)$: The function representing the body cycle. Function of time.
- P_1: The point on the globe that corresponds to the home place.
- P_2: The point on the globe that corresponds to the meeting place.
- d: The distance between P_1 and P_2.
- φ: The time gap between P_1 and P_2.
- L: The lowest value of efficiency that we consider 'productive'. If the efficiency value is higher than L, then he/she is productive.

III Flight Time Estimation

For a given two locations, each set to the departing and arriving places, we find the shortest time needed to move between those spots. And indeed, we would assume the shape of the earth as a sphere, and use a spherical coordinate for convenience.

Let the coordinates of two positions be $P_1(r_E,\ \theta_1,\ \phi_1)$, $P_2(r_E,\ \theta_2,\ \phi_2)$.

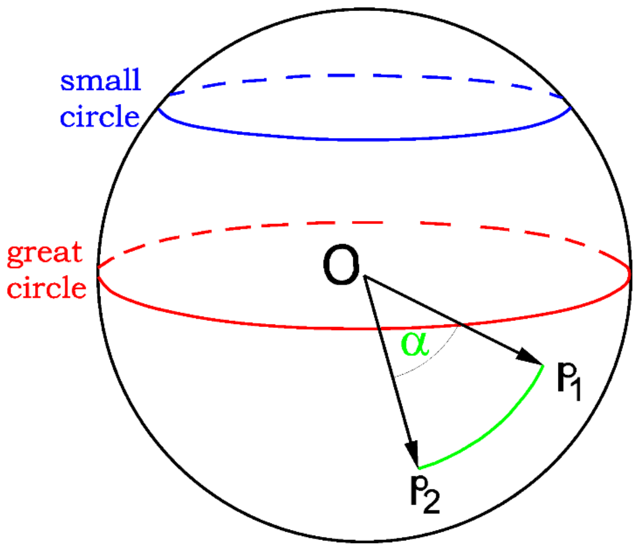

Fig. 1 Distance between two positions

For the plane α that passes through the origin O and two given points P_1, P_2, it is well known that the smallest distance between P_1, P_2 is the length of arc along the great circle, which is formed by the intersection of the given sphere and the plane α. By this fact, we simply get the length of arc by discovering the angle between two vectors $\overrightarrow{OP_1}$ and $\overrightarrow{OP_2}$. Then the shorter distance d along the great circle would be as following.

$$d = \frac{r_E + h_f}{u} \cos^{-1}(\cos\phi_1 \cos\phi_2 \cos(\theta_1 - \theta_2) + \sin\phi_1 \sin\phi_2) \qquad \mathbf{(3-1)}$$

where r_E is the earth's radius, u is the velocity of an airplane, and h_f is the height of an airplane. Average height of an airplane is $h_f = 36000ft$.

We don't need to consider the date line when it comes to the duration of flight. This is because that we just have to earn the time difference in a body cycle that repeats with 24 hours' period. Also absolute difference of local time results in longitude shift, and its relationship is given as follows.

$$\varphi = \frac{\phi_2 - \phi_1}{2\pi} \cdot 24hr \qquad \mathbf{(3-2)}$$

Let's go on with two simple examples. For the given two positions that is given by its spherical coordinates, calculate the time duration that is needed to be spent by the airplane. We supposed that plane's speed is 508.16mph.

Table 1 Comparison between calculated and actual flight duration

(latitude/longitude)	city 1	city2	Duration	Actual Duration
Case 1	Incheon (37.47/126.45)	Dallas (32.85/-96.85)	13.47hr	13.53hr
Case 2	Sofia (46.70/23.41)	Tokyo (35.67/139.75)	10.87hr	11.35hr

IV Relationship between Fatigue and Efficiency

We have to maximize the overall productivity of the meeting. Overall productivity is an abstract concept, so we decided to concern rate of production. Further explanation, we use term 'efficiency' as a rate of production. According to normal people's experience, fatigue is a major factor of rate of production. We consider fatigue as only factor of rate of production and contain other minor factors into fatigue. Now, rate of production is function that only depend on fatigue.

1. Fatigue

Fatigue is also an abstract concept. So we invest fatigue with validity by minor items. If we can represent fatigue as a function of measurable parameters, fatigue can be treated as an objective validity. We choose parameters which affect to fatigue as efficiency, sleep time, body cycle, and time difference. These are well known fatigue factors and less different types of people. More concretely, we give details about parameters.

First, efficiency. Fatigue is caused by mental or physical labor. Meeting is kind of both mental and physical labor. So, fatigue is affected by efficiency. How much we feel tired is proportional to amount of work, so we can empirically say that change of fatigue rate is proportional to efficiency. ($\frac{dF}{dt} \propto E$) However, the coefficient factor of efficiency is not constant. It depends on the type of work and the body condition. Since the type of work is fixed as a meeting, only body condition, or body cycle affects the coefficient factor. Body cycle is a function of t, so we defined the coefficient of efficiency as $f_1(t)$.

$$\frac{dF}{dt} = f_1(t)E \qquad \mathbf{(4-1)}$$

Sleep time is a second factor. Fatigue can be reduced by sleeping or resting. Since problem states "three intensive days", so we ignored daytime resting and only considered sleeping. Sleeping activity resolves the tiredness, so we set the sleeping time to affect fatigue on its increasing rate. To express the sleeping time, we set a function $R(t)$, which takes value of 1 while sleeping, and 0 while not.

$$R(t) = \begin{cases} 1, & 0 \le t \le 8 \\ 0, & 8 \le t \le 24 \end{cases} \qquad \mathbf{(4-2)}$$

Adding the sleeping factor to equation **(4-1)**, the following equation for fatigue is derived.

$$\boldsymbol{\frac{dF}{dt} = f_1(t)E - f_2(t)R} \qquad \mathbf{(4-3)}$$

Third, body cycle affects the fatigue. People has their own cycle. It depends on people's characteristic and environment. Since we assumed that every participant's characteristics are identical, so in our modeling, environment is the only factor of body cycle. Sleeping, day and night, and amount of insolation is major item of body cycle. Furthermore, sleeping, day and night is a phenomenon caused by an amount of insolation, so body cycle is function of amount of insolation. Amount of insolation changes by the altitude of sun. Altitude of the sun follows sinusoidal relationship by the time. To return the maximum value at noon and return the minimum at midnight, $B(t)$, the function of body cycle by the time t is defined as the following.

$$\boldsymbol{B(t) = \sin\left(\frac{2\pi}{24hr}(t-6)\right) \qquad (4-4)}$$

It is not directly affect to fatigue. But body cycle affects the overall body function. So, it determines fatigue causes' influence rate, represented as coefficient of fatigue causes'. As a result, It will affect as a coefficient of E and G, which are $f_1(t)$ and $f_2(t)$. When body cycle function's value is positive, it means people don't get tired too much. So coefficient of fatigue causes', $f_1(t)$ is small. Also, sleeping activity at noon doesn't resolve the fatigue effectively than at night. So positive body cycle value makes $f_2(t)$ small, too. Since it has to change body cycle's value drastically when the value is at average region and change body cycle's value fluently when value is extremal.

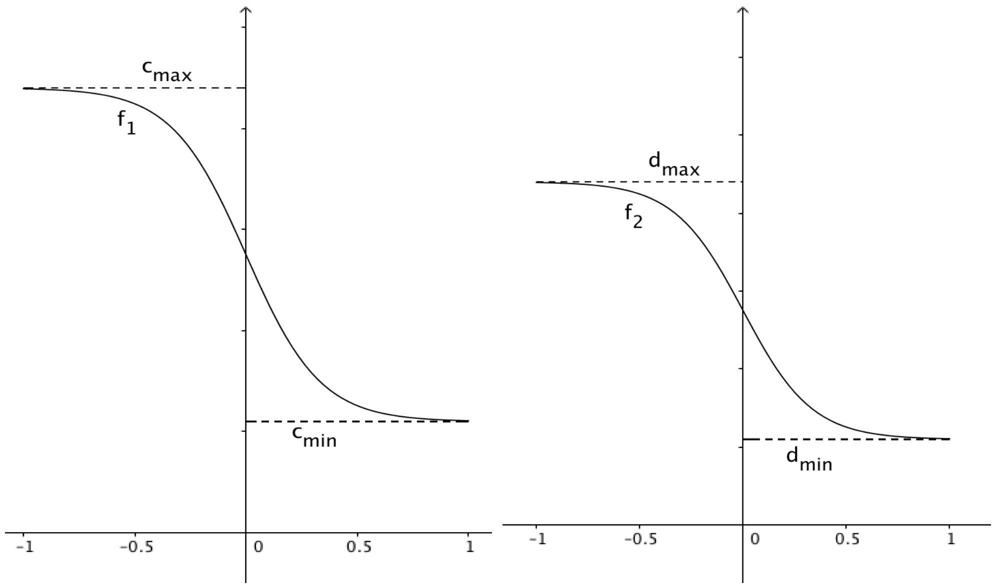

Fig. 2 Graph of $f_1(B)$ and $f_2(B)$

So we choose $\tanh x$ function to express this kind of movement of coefficient. When we set the equation to follow Fig.2's tendency, precise equation of $f_1(t)$ and $f_2(t)$ can be written as the following.

$$\boldsymbol{f_1 = \frac{c_{max} + c_{min}}{2} - \frac{c_{max} - c_{min}}{2}\tanh(3B(t))} \qquad \mathbf{(4-5)}$$
$$\boldsymbol{f_2 = \frac{d_{max} + d_{min}}{2} - \frac{d_{max} - d_{min}}{2}\tanh(3B(t))} \qquad \mathbf{(4-6)}$$

2. Efficiency

People can only work hard when they are lively. If he/she is tired enough, his/her productivity gets a rapid decrease. Considering this characteristic, we can set efficiency as a function of fatigue. People feel extreme tiredness when their fatigue exceed the critical point. However, except the interval near the critical point p_c, change of efficiency is subtle. Using $\tanh x$ function, we expressed the function of efficiency by the fatigue as the following.

$$\boldsymbol{E = h(F) = \frac{1}{2} - \frac{1}{2}\tanh(k(F - p_c))} \qquad \mathbf{(4-7)}$$

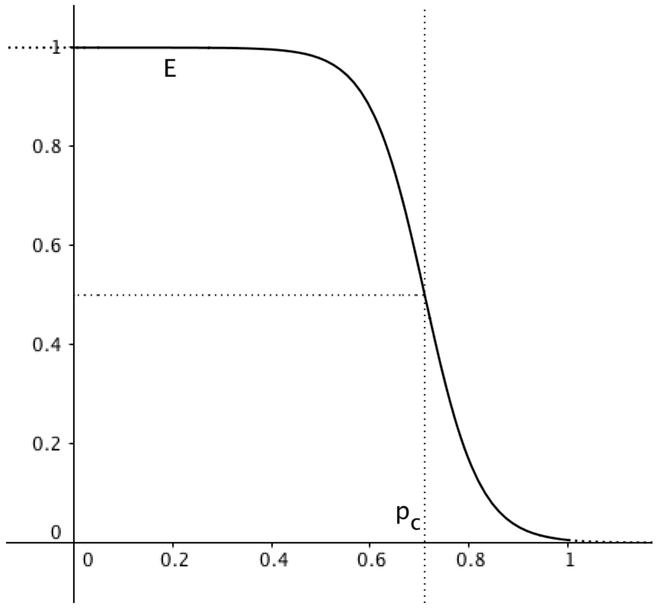

Fig. 3 Graph about efficiency and fatigue

We decided to consider a person as "efficient" if the one's efficiency is bigger than the average of maximum and minimum efficiency.

These are the major equations derived.

Table 2 Major differential equations about fatigue

Variable		Equations
Fatigue	F	$\frac{dF}{dt} = f_1(t)E - f_2(t)R \quad (4-3)$
	f_1	$f_1(t) = \frac{c_{max} + c_{min}}{2} - \frac{c_{max} - c_{min}}{2}\tanh(3B(t)) \quad (4-5)$
	f_2	$f_2(t) = \frac{d_{max} + d_{min}}{2} - \frac{d_{max} - d_{min}}{2}\tanh(3B(t)) \quad (4-6)$
Body Cycle	B	$B(t) = \sin\left(\frac{2\pi}{24hr}(t-6)\right) \quad (4-4)$
Efficiency	E	$E = h(F) = \frac{1}{2} - \frac{1}{2}\tanh(k(F-p_c)) \quad (4-7)$

3. Coefficient values

We developed the model about one's fatigue and efficiency. However, the coefficients are currently unknown. To estimate the fatigue value, we need to get $c_{max}, c_{min}, d_{max}, d_{min}, k, p_c$ values. Since it is hardly possible to compare with the real values, we decided to compare with the model developed in the preceding research.

In the preceding research "Sleep Homeostasis and Models of Sleep Regulation, Borb, A. A., & Achermann, P. (1999).", one's fatigue can be estimated by following two differential equations.

$$\frac{d}{dt}SWA(t) = rcSWA(t)\left(1 - \frac{SWA(t)}{S(t)}\right)\frac{F(t)}{F_U} - fc_R(SWA(t) - SWA_L)R(t) - fc_W(SWA(t) - SWA_L)W(t) \qquad (\mathbf{4-8})$$

$$\frac{d}{dt}F(t) = -gcSWA(t) + rs(F_U - F(t)) \qquad (\mathbf{4-9})$$

Which contains two dependent function $F(t)$ and $SWA(t)$. Using the assumption 1, the sleeping is processed during 0 a.m. to 8 a.m., and the rest of the time is assigned to activities.

The constants in the equation **(4-8, 9)** are the following values.

Table 3 Constants described in preceding research

fc	fc_R	fc_W	gc	rs	SWA_L	$SWA(t)$	F_U	$F(0)$
0.283	0.236	1	0.00835	0.0009167	0.0177	0.083	1	0.5563

By this assumption, $R(t)$ is empirically defined by 1 at the domain from 0 to 8 a.m., rest of the part would be zero. In directly opposite way, $W(t)$ would have the value given 0 at the domain from 0 to 8 a.m., rest of the part would be 1.

$$W(t) = \begin{cases} 0, & 0 \le t \le 8 \\ 1, & 8 \le t \le 24 \end{cases} \qquad \mathbf{(4-10)}$$

Since our model and the preceding research's model are both 1st-order nonlinear differential equation, by using the Runge-Kutta Method, it is possible to find the numerical solution for the given system of differential equations. For the solution of preceding research, one's fatigue during the 200days converges into specific periodic function that is constrained in particular interval. In the converging interval, the maximum fatigue value is $F_{max} = 0.753331$, the minimum fatigue value is $F_{min} = 0.598703$.

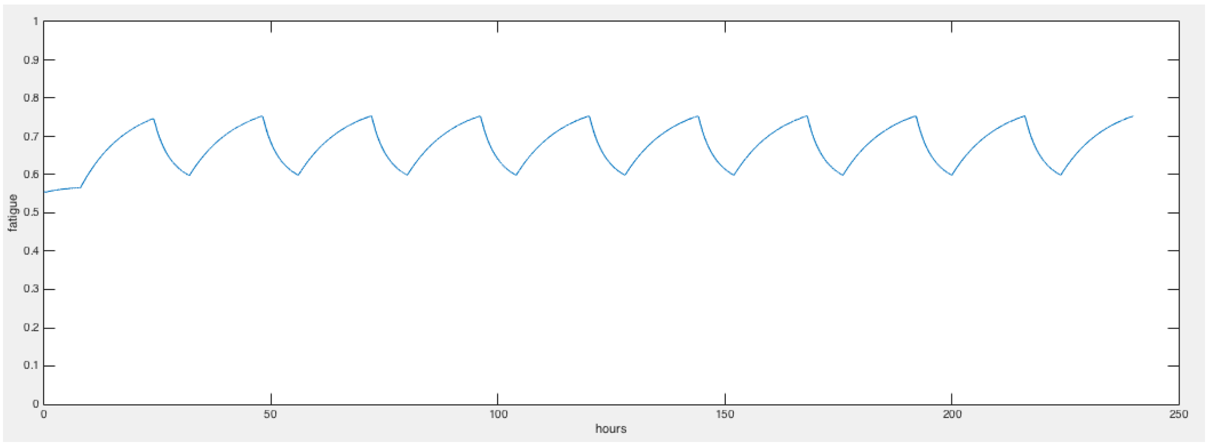

Fig. 4 The fatigue of first 240 hours calculated by preceding model

Using the converging preceding model, we sorted out the data for 3days and determined the most appropriate coefficients that would best match with the theoretical value, by varying 6 coefficients, c_{max}, c_{min}, d_{max}, d_{min}, k, p_c.

We chose the most appropriate coefficients by the following criterion.

1. For the tuples of specific coefficients (c_{max}, c_{min}, d_{max}, d_{min}, k, p_c), after 50 days passed, it converges to specific periodic function.
2. Since it converges to certain function after 50 days, we pick the data for 3 days after the 50 days passed.
3. From the data that converged to the theoretical value, and the data mentioned right above, we operate the data for the 3 days in 0.01-hour gap to earn 7200 deviations total.
4. Finding the most appropriate tuples of the coefficients is equivalent to the smallest RMS(root-mean-squared) average of the 7200 deviations above.

From the step given here, we earn the most appropriate tuples of the coefficients.

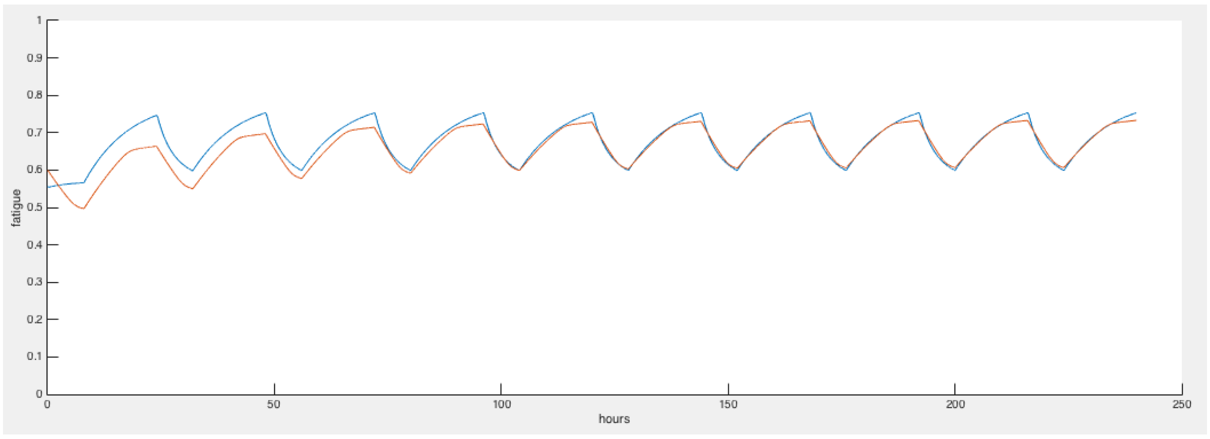

Fig. 5 Comparison between our model and preceding model on first 240 hours

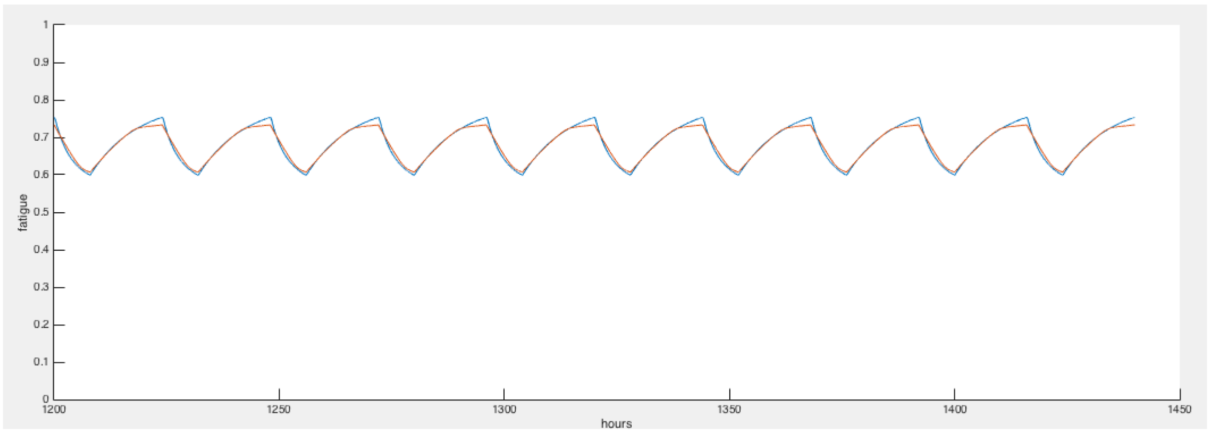

Fig. 6 Comparison between our model and preceding model when they converges

Table 4 Results of most appropriate coefficients

c_{max}	c_{min}	d_{max}	d_{min}	k	p_c
0.0003	0.00004	0.0003	0.00005	3.0	0.60

Using these coefficient values, we can rewrite the equations for our model.

Table 5 Major equations with real values

Variable		Equations
Fatigue	F	$\frac{dF}{dt} = f_1(t)E - f_2(t)R \quad (4-3)$
	f_1	$f_1(t) = 0.00017 - 0.00013\tanh(3B(t)) \quad (4-11)$
	f_2	$f_2(t) = 0.000175 - 0.000125\tanh(3B(t)) \quad (4-12)$
Body Cycle	B	$B(t) = \sin\left(\frac{2\pi}{24hr}(t-6)\right) \quad (4-4)$
Efficiency	E	$E = h(F) = \frac{1}{2} - \frac{1}{2}\tanh(3(F-0.6)) \quad (4-13)$

V Change of Body Cycle after the Flight

The most important characteristic of the meeting is that it is held internationally. This means that the fatigue of each participants differs depending on the time gap. However, longer each participant stays, their body cycle adapts to the local time of the meeting place. From the assumption 2, they only take 72 hours to adapt to a new cycle.

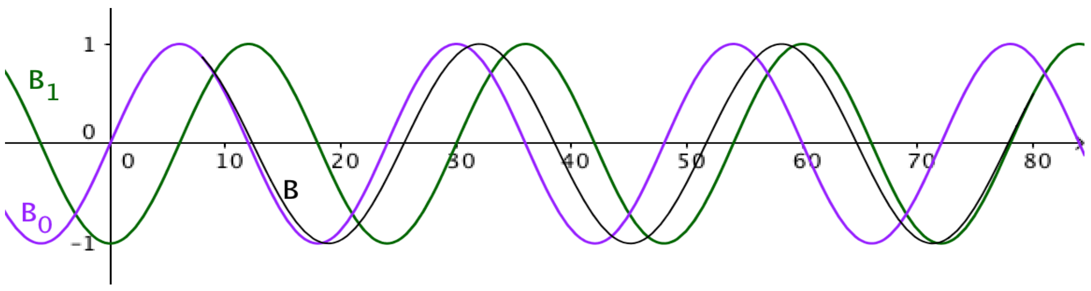

Fig. 7 Adaption of body cycle

Fig. 7 shows the adaption of the body cycle. The participant's body cycle was B_0, but when he arrives at the meeting place with the time gap of -6 hours, his body cycle starts to adapt to B_1. His body cycle while adapting is follows B. So, his body cycle is expressed as following.

$$Body\ cycle \begin{cases} B_0(t)\,,(t \le 8) \\ B(t)\,,(8 \le t \le 80) \\ B_1(t)\,,(80 \le t) \end{cases} \qquad \mathbf{(5-1)}$$

It is possible to get the general equation of B. Let's fix the time axis to the time at the meeting place. Then, $B_1(t) = \sin\left(\frac{2\pi}{24hr}(t-6)\right)$. If he came from the area with the time gap φ $(-12hr \le \varphi \le 12hr,\ \varphi \equiv \frac{24hr}{360°}(\phi_2 - \phi_1))$, then the previous body cycle $B_0(t) = \sin\left(\frac{2\pi}{24hr}(t-6-\varphi)\right)$. Using $B_0(t)$ and $B_1(t)$, $B(t)$ is the following.

$$B(t) = \sin\left(\frac{2\pi}{24hr}\frac{(72+\varphi)hr}{72hr}\left(t-6-\frac{74\varphi}{72+\varphi}\right)\right), (8 \le t \le 80) \qquad \mathbf{(5-2)}$$

By this body cycle equation, we reflect assumption 2, which we need three days to adapt to the time difference. Also, this equation made body cycle to adjust gradually. So, by substituting this equation to the previous differential equation **(4-5, 6, 11, 12)**, we can estimate the fatigue value during the meeting.

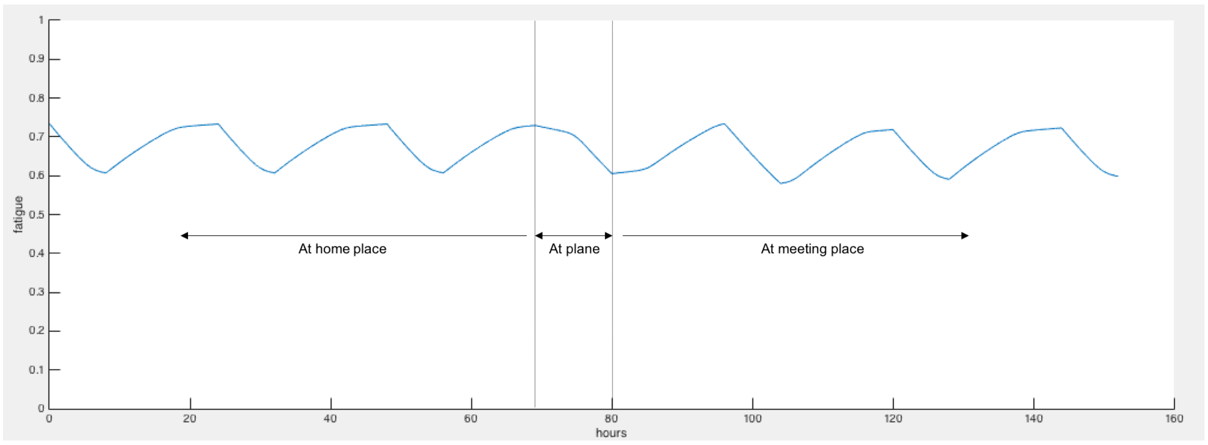

Fig. 8 Estimated fatigue value after the flight

The Fig. 8 shows the fatigue value of the participant who was from the home place with the 12 hours of time gap.

VI Maximizing the Productivity of the Meeting

1. Definition of productive meeting

In our problem, we have to choose place that maximize the overall productivity of the meeting. So, we defined productivity of the meeting's objective definition. We used the concept we defined in previous steps to define the productivity of the meeting. We found that in normal situation, the fatigue converges to a periodic function. We used the maximum and minimum values of that period function, which are $F_{max} = 0.753331$ and $F_{min} = 0.598703$.

We defined a constant L which is a boundary value to consider whether he/she is productive or non-productive. We chose the value of L as the following.

$$\exists L \; such \; that \; h(L) = \frac{h(F_{max}) + h(F_{min})}{2} = 0.672137$$

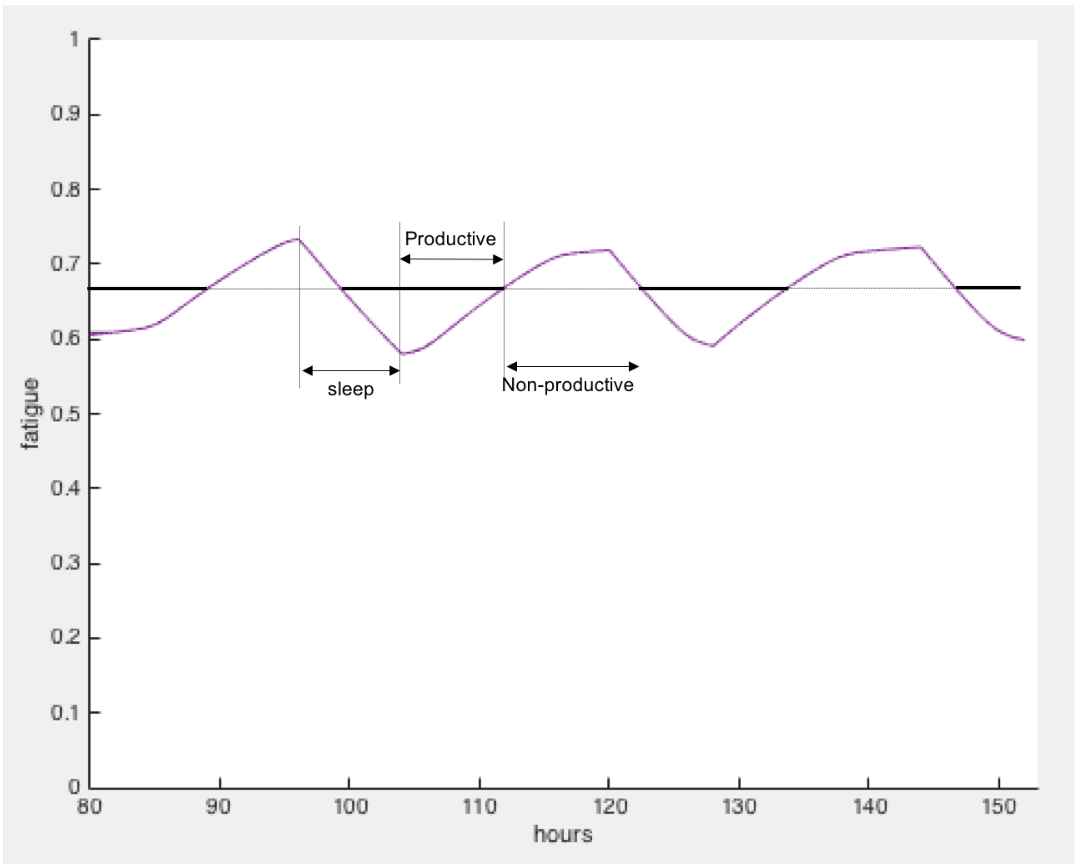

Fig. 9 Description of time of sleeping, productive and non-productive

If somebody's efficiency is higher than L, we considered "productive". For the meeting's productiveness, productive meeting need many people's productive participation (since a meeting is cooperative work). If $\frac{2}{3}$ of all participants' efficiency is higher than L, we can say that the meeting is a productive meeting.

The following figure shows how to calculate the productive meeting time at $n = 3$. Three color line shows the productive time of each participants. When there are two or more productive participants, we can say the meeting is productive.

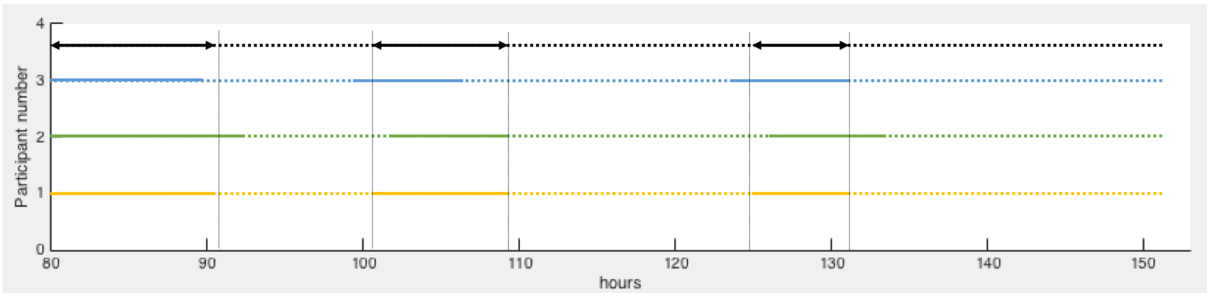

Fig. 10 Diagram showing the time of productive meeting when n=3

2. Algorithms to find appropriate location

What we found before was the fatigue graph for each human being, based on our model. Our eventual goal is to apply this graph into the condition of the problem.

The fatigue curve is maintained for general routine, and suddenly undertake some change when that person hop into the airplane. Let's say that the time for taking the airplane is T. And the airplane is planned to arrive at 8 a.m. with respect to the arrival local time. We may assume that the conference suddenly starts right after the arrival without delay.

1. Calculating each participant's time gap

Their departing time would be $8 - \mathrm{T}$ with respect to the arrival local time. And we correspond the value $8 - \mathrm{T}$ with the remainder modular 24. Everyone repeats the pattern, and each one has the initial fatigue indicator that is specified by the departure local time $8 - \mathrm{T} - \varphi$. After that, since they are assumed to have a sleep during the whole flight time, fatigue monotonously decreases.

2. **Calculating each participant's fatigue**

 When they finish arriving to destination, fatigue gradually increases due to the intense meeting schedule. Fatigue is accumulated, forming a zig-zag increasing fatigue tendency by time.

 For a given criterion fatigue indicator f, while the fatigue is less than or equal to f, we may say that 'they work efficiently'. Then except for the sleeping time, we can consider the net efficient time during the 72 hours, respectively for each person. Step function that have a domain between 8h to 72h is made (0 for non-efficiency or sleep, 1 for hardworking). For this 'efficiency state function', we consider the total sum of the graphs of every individual participant.

3. **Calculating the length of a meeting's productive time**

 For total, say, n participants, we calculate the total sum graph of every n graphs. And our criterion is $\frac{2}{3}$ ratio of efficient people to the total number of people. For the time interval that the function is above that value (for example, at least 4 people is needed to make sure that the conference is being delivered well if there's 6 people in total.), the total amount of the time is our main interest. This is the function that puts out the total amount of 'efficient time' for the given initial condition as following; 'each participant's departing location', 'the location of meeting'. So, we have to find the very input 'the location of meeting' which maximizes the total amount of efficient time.

4. **Searching for the appropriate meeting location**

 Therefore, our goal is to discover the meeting location that maximizes the efficient time. We have two methods to search the most appropriate location of meeting.

 i) **Divide-and-Conquer method**

 We divided the globe as subparts that is sectioned into 9 pieces, each of the sizes and dimensions are equally distributed. In this manner, we find an area that maximizes the functional output, and we made it more precise by sectioning the area into 9 parts again, applying the same mechanism on them. By this idea, we ultimately make the range of the optimal conference place smaller and set more precise thesis about it.

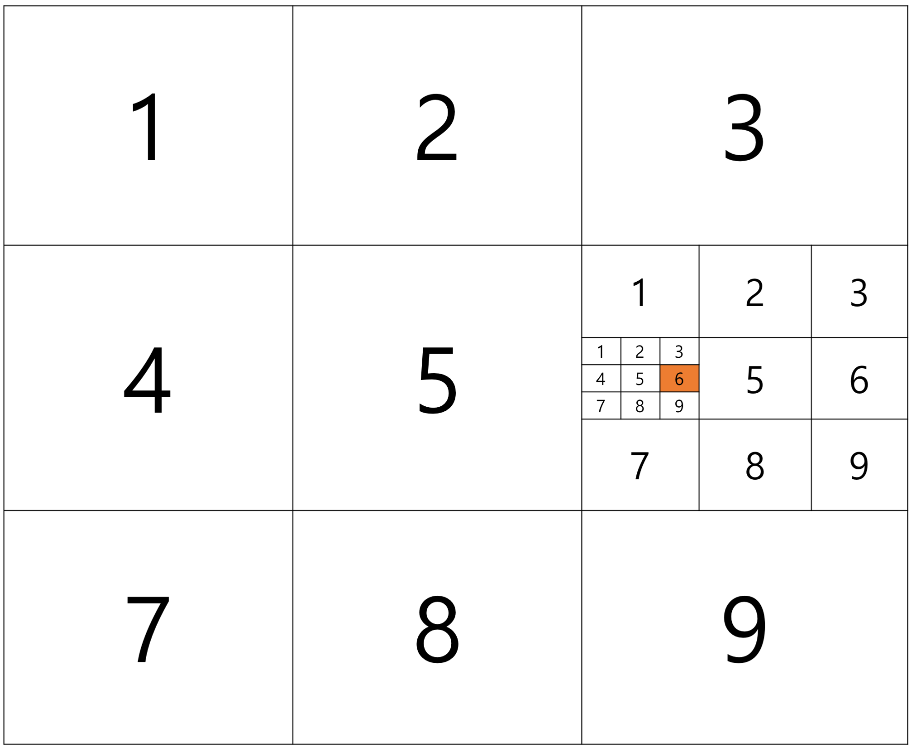

Fig. 11 Finding the appropriate location using Divide and Conquer method

ii) Greedy search method

In this method, we calculated the productive meeting time length for every airport in the world. Since every participants arrives at the meeting place by the plane, we assumed that the meeting place is placed near the airport. It is possible to check all the airports, so we used greedy method and chose the place with the maximum productive meeting time length.

VII Scenario 1) "Small meeting"

We have to find the place with the maximum "productive meeting" time. We used both Divide-and-Conquer method and greedy search method to find the most appropriate location. First, we find maximum latitude and longitude by Divide-and-Conquer method. Followings are the results of it.

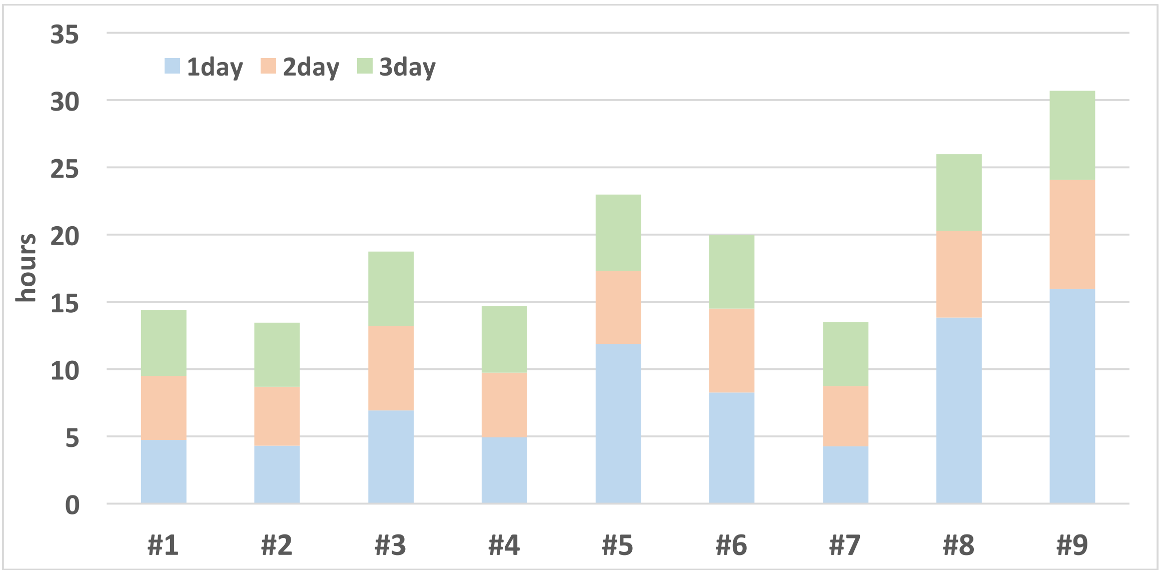

Fig. 12 Comparison of total productive time of each 9 sections at step 1

Table 6 The time length of productive meeting for each sections at each steps

Steps	#1	#2	#3	#4	#5	#6	#7	#8	#9
1st	14.40	13.48	18.74	14.71	22.99	20.00	13.51	25.98	30.69
2nd	27.43	25.28	27.67	27.07	30.69	29.06	29.59	31.49	31.35
3rd	31.58	30.29	29.26	31.81	31.35	29.4	32.45	31.90	29.52
4th	31.85	32.22	32.69	32.04	32.45	32.95	32.25	32.68	33.29

By running algorithm 4 times, we can get location as specific as we want. Its location is south altitude 22.365, and west longitude 98.176. But we got latitude and longitude, so we need one more step. We have to find the nearest airport. The nearest airport is Chile, Easter Island, Hanga Roa airport. By this method, we can get exact location of maximum "productive meeting" time, but we need one more steps to find airport. So we try other method to get appropriate airport directly.

Second, we compare every airport's "productive meeting" time by greedy search method. There are about 7000 airports. We get appropriate airports by comparing about 7000 airports with our algorithm. As a result, Chile, Easter island, Hanga Roa airport is the appropriate airport. Compare to the first method, we got the same consequence. So both methods are able to find the appropriate airport well.

Table 7 The most appropriate location found by each method

	Method 1	Method 2
Location	22°21′ 54"S 98° 10′ 33.6"W	Chile, Easter Island, Hanga Roa

VIII Scenario 2) "Big meeting"

We also have to find the place with the maximum "productive meeting" time. Unlike scenario 1, scenario 2 has 11 people and some of them are even at the same location. However, the method is identical since our algorithms calculates each individual. Like Scenario 1, first, we find maximum latitude and longitude by Divide-and-Conquer method. Followings are the results of it.

Table 8 The time length of productive meeting for each sections at each steps

Steps	#1	#2	#3	#4	#5	#6	#7	#8	#9
1st	15.69	13.56	2.03	16.79	13.96	26.01	28.65	31.99	31.59
2nd	25.37	25.87	29.98	31.3	31.99	32.45	32.58	32.93	34.47
3rd	33.18	33.75	34.84	33.65	34.47	35.98	33.46	32.61	31.69
4th	35.09	35.69	36.36	35.42	35.99	36.62	35.47	35.12	34.67
5th	36.32	36.53	36.75	36.39	36.62	36.82	36.38	36.20	36.32

By running algorithm 5 times, we can get location as specific as we want. Location of it is south altitude 80.748, and west longitude 56.311. But we got latitude and longitude, so we need one more step. We have to find the nearest airport. The nearest airport is Grytoiken, South Georgia islands' airport. By this method, we can get exact location of maximum "productive meeting" time, but we need one more steps to find airport. So we try other method to get appropriate airport directly.

Second, we compare every airport's "productive meeting" time by greedy search method. As a result, Grytoiken, South Georgia island's airport is the appropriate airport. Compare to the first method, we got the same consequence. So both methods are able to find the appropriate airport well.

Table 9 The most appropriate location found by each method

	Method 1	Method 2
Location	$80° 44' 52.8"S\ 56° 18' 39.6"W$	Grytoiken, South Georgia islands' airport

IX Conclusion

The given problem was to find the most appropriate meeting location to maximize the productivity of meeting. We focused on the changing abstract concept to objective concept. By reasonable relation between abstract concept and objective concept, we can express abstract concept by equation of computable items. We express productivity by efficiency, fatigue, body cycle, sleeping time, and time difference.

Since each participant are from different time zones, jet lag of each participants affects the productivity. In our model, we defined a body cycle that depends on the time zone. It was possible to derive the relationship between the fatigue and efficiency of each one. We solved the equations numerically, and was able to calculate the fatigue of each participant.

With the fatigue values of each participant by the time, it was possible to estimate the total time length of the meeting when it was productive. To find the best location, we repeated these steps of calculation for several locations. We developed two algorithms to find the best place. The one is using divide-and-conquer method, by repeating the steps of dividing the map into subpart and finding the best subpart. Other method is a greedy method, which checks every airport in the world and finding the most appropriate location. Both methods give us the same results, so we can say that both our algorithms to find the optimal meeting location were valid.

.

X Appendix

1. Codes

A. Code for calculating the fatigue using our model

```
#include<stdio.h>
#include<math.h>
#define pi 3.141592653
#define T (24.0*60)
#define dt 1
#define c3 0.00004
#define c4 0.0003
#define c5 0.0003
#define c6 0.00005
#define k 3.0
#define pc 0.6
#define phi 0.0
FILE *fp1;

double Piro[999999];

double fp(double p)
{
        return 0.5-0.5*(tanh(k*(p-pc)));
}
double time(double t)
{
        while(t>24)
        {
                t-=24;
        }
        return t;
}
double rhythm(int t)
{
        double kk = 72/(72-phi);
        return sin(2*pi/(2400.0*kk)*(double)(t-600)+(2*pi/24)*phi);
}

double c1(int t)
{
        return (c3+c4)/2.0+(c4-c3)/2*tanh(3*rhythm(t));
}

double c2(int t)
{
        return (c5+c6)/2.0+(c6-c5)/2*tanh(3*rhythm(t));
}

double piroprime1(double p,int t)
{
        return c1(t)*fp(p);
}

double piroprime2(double p,int t)
{
        return -c2(t)*p;
}

void findpiro(int t)
{
        double k1,k2,k3,k4;
        double ti = (double)t/100.0;
        ti = time(ti);
        if(ti>8)
        {
                k1 = piroprime1(Piro[t],t);
                k2 = piroprime1(Piro[t]+0.5*dt*k1,t+0.5*dt);
                k3 = piroprime1(Piro[t]+0.5*dt*k2,t+0.5*dt);
                k4 = piroprime1(Piro[t]+dt*k3,t+dt);
        }
        else
        {
                k1 = piroprime2(Piro[t],t);
                k2 = piroprime2(Piro[t]+0.5*dt*k1,t+0.5*dt);
                k3 = piroprime2(Piro[t]+0.5*dt*k2,t+0.5*dt);
                k4 = piroprime2(Piro[t]+dt*k3,t+dt);
        }

        Piro[t+1] = Piro[t] + (1.0/6.0)*dt*(k1+2*k2+2*k3+k4);
        if(Piro[t+1]<=0)
        {
               Piro[t+1]=0;
        }
}

int main(void)
{
        fp1 = fopen("output.csv","w");
        Piro[0] = 0.6;
        int t;

        for(t=0;t<=T*100;t++)
        {
                findpiro(t);
        }

        for(t=(T-240)*100;t<=T*100;t++)
        {
                fprintf(fp1,"%d,%lf₩n",t,Piro[t]);
        }
        fclose(fp1);
}
```

B. Code for calculating the fatigue using the preceding research

```
#include<stdio.h>
#include<math.h>
#define T (24.0*60.0)
```

```
#define h 0.01
#define rc 0.283
#define su 1
#define fcr 0.236
#define fcw 1
#define swal 0.0177
#define gc 0.00835
#define rs 0.0009167
FILE *fp1;
double S[999999];
double SWA[999999];

double time(double t)
{
        while(t>24)
        {
                t-=24;
        }
        return t;
}

int remt(int t)
{
        double ti = (double)t/100.0;
        ti = time(ti);
        if(ti<8) return 1;
        else return 0;
}
int w(int t)
{
        double ti = (double)t/100.0;
        ti = time(ti);
        if(ti<8) return 0;
        else return 1;
}
```

```
double f(double s,double swa,int t)
{
        return (-1)*gc*swa + rs*(su-s);
}

double g(double s,double swa,int t)
{
        return rc*swa*(1-(swa/s))*(s/su)-fcr*(swa-swal)*remt(t)-fcw*(swa-swal)*w(t);
}

void findsswa(int t)
{
        double k11,k12,k21,k22,k31,k32,k41,k42;
        k11 = f(S[t],SWA[t],t);
        k12 = g(S[t],SWA[t],t);
        k21 = f(S[t]+(k11*h*0.5),SWA[t]+(k12*h*0.5),t+(h*0.5));
        k22 = g(S[t]+(k11*h*0.5),SWA[t]+(k12*h*0.5),t+(h*0.5));
        k31 = f(S[t]+(k21*h*0.5),SWA[t]+(k22*h*0.5),t+(h*0.5));
        k32 = g(S[t]+(k21*h*0.5),SWA[t]+(k22*h*0.5),t+(h*0.5));
        k41 = f(S[t]+(k31*h),SWA[t]+(k32*h),t+(h));
        k42 = g(S[t]+(k31*h),SWA[t]+(k32*h),t+(h));

        S[t+1] = S[t] + (1.0/6.0)*(k11+2*k21+2*k31+k41);
        SWA[t+1] = SWA[t] + (1.0/6.0)*(k12+2*k22+2*k32+k42);
}

int main(void)
{
        fp1 = fopen("output.csv","w");
        int t;
        S[0] = 0.5563;
        SWA[0] = 0.083;
        for(t=0;t<=T*100;t++)
        {
                findsswa(t);
        }
        for(t=(T-240)*100;t<=T*100;t++)
        {
                fprintf(fp1,"%d,%lf₩n",t,S[t]);
        }
        fclose(fp1);
}
```

C. Code for finding the appropriate coefficients of our model

```
#include<stdio.h>
#include<math.h>
#define pi 3.141592653
#define T (24.0*50)
#define dt 1
#define phi 0.0
FILE *fp1,*fp2;
double c3,c4,c5,c6,k,pc;
double sum;

double Piro[999999];
double Pref[10000];

void START(void)
{
        int i;
        for(i=0;i<=999990;i++)
        {
                Piro[i] = 0;
        }
}
double fp(double p)
{
        return 0.5-0.5*(tanh(k*(p-pc)));
}
double time(double t)
{
	while(t>24)
	{
		t-=24;
	}
	return t;
}
double rhythm(int t)
{
	double kk = 72/(72-phi);
	return sin(2*pi/(2400.0*kk)*(double)(t-600)+(2*pi/24)*phi);
}

double c1(int t)
{
	return (c3+c4)/2.0+(c4-c3)/2*tanh(3*rhythm(t));
}

double c2(int t)
{
	return (c5+c6)/2.0+(c6-c5)/2*tanh(3*rhythm(t));
}

double piroprime1(double p,int t)
{
	return c1(t)*fp(p);
}
double piroprime2(double p,int t)
{
	return -c2(t)*p;
}
void findpiro(int t)
{
	double k1,k2,k3,k4;
	double ti = (double)t/100.0;
	ti = time(ti);
	if(ti>8)
```

```
        {
                k1 = piroprime1(Piro[t],t);
                k2 = piroprime1(Piro[t]+0.5*dt*k1,t+0.5*dt);
                k3 = piroprime1(Piro[t]+0.5*dt*k2,t+0.5*dt);
                k4 = piroprime1(Piro[t]+dt*k3,t+dt);
        }
        else
        {
                k1 = piroprime2(Piro[t],t);
                k2 = piroprime2(Piro[t]+0.5*dt*k1,t+0.5*dt);
                k3 = piroprime2(Piro[t]+0.5*dt*k2,t+0.5*dt);
                k4 = piroprime2(Piro[t]+dt*k3,t+dt);
        }

        Piro[t+1] = Piro[t] + (1.0/6.0)*dt*(k1+2*k2+2*k3+k4);
        if(Piro[t+1]<=0)
        {
               Piro[t+1]=0;
        }
}

int main(void)
{
        fp1 = fopen("output.csv","w");
        fp2 = fopen("input.txt","r");
        int t;
        int i;
        for(i=0;i<=7200;i++)
        {
                fscanf(fp2,"%lf₩n",&Pref[i]);
        }
        for(c3=0.00003;c3<=0.00005;c3+=0.00001)
                for(c4=0.0003;c4<=0.0005;c4+=0.0001)
                        for(c5=0.0003;c5<=0.0005;c5+=0.0001)
                        for(c6=0;c6<=0.000075;c6+=0.000025)
                        for(k=3;k<=3.5;k+=0.5)
```

```
			for(pc=0.55;pc<=0.66;pc+=0.05)
			{
				sum = 0;
				START();
				printf("%lf %lf %lf %lf %lf %lf₩n",c3,c4,c5,c6,k,pc);
				for(t=0;t<=T*100;t++)
				{
					findpiro(t);
				}
				for(i=0;i<=7200;i++)

				{
					sum+=pow(Pref[i]-Piro[(int)(T*100)-7200+i],2);
				}
				sum/=7200.0;
				fprintf(fp1,"%lf,%lf,%lf,%lf,%lf,%lf,,%lf₩n",c3,c4,c5,c6,k,pc,sqrt(sum));
			}
	fclose(fp1);
}
```

D. Code for finding the appropriate meeting place – method 1

```
#include<stdio.h>
#include<math.h>
#define pi 3.141592653
#define T (24.0*51)
#define c3 0.00004
#define c4 0.0003
#define c5 0.0003
#define c6 0.00005
#define k 3.0
#define pc 0.6
#define dt 1
#define u 817.804247
#define Re 6371
#define Pf 0.672137
```

```
FILE *fp1, *fp2, *fp3, *fp4;

double Piro[999999];
double Pi[3000];
double thetai[20],thetaf[7000],phii[20],phif[7000];
double phi,FT;
int check[7000][15000];
int realcheck[7000][15000];
int cccheck[7000];

double rad(double x)
{
	return pi/180.0*x;
}
void IV(void)
{
	int i,a,b;
	for(i=0;i<=900000;i++)
	{
		Piro[i] = 0;
	}
}
void place(void)
{
	fp3 = fopen("place.txt","r");
	fp4 = fopen("airport.txt","r");
	int i;
	for(i=1;i<=11;i++)
	{
		fscanf(fp3,"%lf %lf",&thetai[i],&phii[i]);
		thetai[i] = rad(thetai[i]);
		phii[i] = rad(phii[i]);
	}
	/*

	for(i=1;i<=6977;i++)
```

```
{
        fscanf(fp4,"%lf %lf",&thetaf[i],&phif[i]);
        thetaf[i] = rad(thetaf[i]);
        phif[i] = rad(phif[i]);
}
*/
double t1 = -78.89;
double t2 = -81.11;
double p1 = -55.56;
double p2 = -60;
t1 = rad(t1);
t2 = rad(t2);
p1 = rad(p1);
p2 = rad(p2);

thetaf[1] = -77.8740;
phif[1] = -34.62616;

thetaf[2] = -79.75;
phif[2] = -83.30;

thetaf[1] = (5*t1+1*t2)/6.0;
phif[1] = (5*p1+1*p2)/6.0;

thetaf[2] = (5*t1+1*t2)/6.0;
phif[2] = (3*p1+3*p2)/6.0;

thetaf[3] = (5*t1+1*t2)/6.0;
phif[3] = (1*p1+5*p2)/6.0;

thetaf[4] = (3*t1+3*t2)/6.0;
phif[4] = (5*p1+1*p2)/6.0;

thetaf[5] = (3*t1+3*t2)/6.0;
phif[5] = (3*p1+3*p2)/6.0;
```

```
	thetaf[6] = (3*t1+3*t2)/6.0;
	phif[6] = (1*p1+5*p2)/6.0;

	thetaf[7] = (1*t1+5*t2)/6.0;
	phif[7] = (5*p1+1*p2)/6.0;

	thetaf[8] = (1*t1+5*t2)/6.0;
	phif[8] = (3*p1+3*p2)/6.0;

	thetaf[9] = (1*t1+5*t2)/6.0;
	phif[9] = (1*p1+5*p2)/6.0;

}
double fp(double p)
{
	return 0.5-0.5*(tanh(k*(p-pc)));
}
double time(double t)
{
	while(t>24)
	{
		t-=24;
	}
	return t;
}
void flighttime(int i,int j)
{
	double the = acos(cos(thetai[i])*cos(thetaf[j])*cos(phii[i]-phif[j]) + sin(thetai[i])
*sin(thetaf[j]));
	FT = Re*the/u;
}
double rhythm(int t)
{
	double ti = double(t)/100.0;
	if(ti<8.0)
	{
```

```
                return sin(2*pi/24.0*(ti-6.0-phi));
        }
        else if(ti<80.0)
        {
                return sin((2*pi/24.0)*((72.0+phi)/72.0)*(ti-6.0-(74.0*phi/(72.0+phi))));
        }
        else
        {
                return sin((2*pi/24.0)*(ti-6.0));
        }
}
double dphi(int a,int b)
{
        double delphi = phif[b]-phii[a]+pi;
        if(delphi<0) delphi+=(2*pi);
        if(delphi>2*pi) delphi-=(2*pi);
        return delphi-pi;
}

double c1(int t)
{
        return (c3+c4)/2.0+(c4-c3)/2*tanh(3*rhythm(t));
}
double c2(int t)
{
        return (c5+c6)/2.0+(c6-c5)/2*tanh(3*rhythm(t));
}

double piroprime1(double p,int t)
{
        return c1(t)*fp(p);
}
double piroprime2(double p,int t)
{
        return -c2(t)*p;
}
```

```
void findpiro(int t)
{
        t+=2400;
        double k1,k2,k3,k4;
        double ti = (double)t/100.0;
        ti = time(ti);

        if(ti>8)
        {
                k1 = piroprime1(Piro[t],t);
                k2 = piroprime1(Piro[t]+0.5*dt*k1,t+0.5*dt);
                k3 = piroprime1(Piro[t]+0.5*dt*k2,t+0.5*dt);
                k4 = piroprime1(Piro[t]+dt*k3,t+dt);
        }
        else
        {
                k1 = piroprime2(Piro[t],t);
                k2 = piroprime2(Piro[t]+0.5*dt*k1,t+0.5*dt);
                k3 = piroprime2(Piro[t]+0.5*dt*k2,t+0.5*dt);
                k4 = piroprime2(Piro[t]+dt*k3,t+dt);
        }
        Piro[t+1] = Piro[t] + (1.0/6.0)*dt*(k1+2*k2+2*k3+k4);
        if(Piro[t+1]<=0)
        {
               Piro[t+1]=0;
        }
}
void findpirofly(int t)
{
        t+=2400;
        double k1,k2,k3,k4;
        double ti = (double)t/100.0;
        ti = time(ti);

        k1 = piroprime2(Piro[t],t);
        k2 = piroprime2(Piro[t]+0.5*dt*k1,t+0.5*dt);
        k3 = piroprime2(Piro[t]+0.5*dt*k2,t+0.5*dt);
        k4 = piroprime2(Piro[t]+dt*k3,t+dt);
        Piro[t+1] = Piro[t] + (1.0/6.0)*dt*(k1+2*k2+2*k3+k4);
        if(Piro[t+1]<=0)
        {
                Piro[t+1]=0;
        }
}

void ccheck(int b,int t)
{
        t+=2400;
        double ti = (double)t/100.0;
        ti = time(ti);
        if(ti>8)
        {
                if(Piro[t]<0.672137)
                {
                        //printf("a");
                        check[b][t]++;
                }
        }
}

int main(void)
{
        fp1 = fopen("output.csv","w");
        fp2 = fopen("input.txt","r");
        int t;
        int i;
        int x;
        place();
        for(i=0;i<=2400;i++)
        {
                fscanf(fp2,"%lf₩n",&Pi[i]);
        }
```

```
int a=1,b=10;

//for(b=1;b<=1;b++)
{
	printf("b : %d₩n",b);
	//for(a=5;a<=5;a++)
	{
		IV();
		//printf("a : %d    b : %d₩n",a,b);
		phi = (24.0/2/pi)*dphi(a,b);
		flighttime(a,b);
		//printf("phi : %lf   T : %lf₩n",phi,FT);
		if(FT<8)
		{
			x = int((8.0-FT)*100.0);
			//printf("%lf₩n₩n",Pi[x]);
			for(i=0;i<=4800;i++)
				Piro[i] = Pi[x];
		}
		else
		{
			x = int((32.0-FT)*100);
			//printf("%lf₩n₩n",Pi[x]);
			for(i=0;i<=4800;i++)
				Piro[i] = Pi[x];
		}
		for(t=0;t<=int(FT*100);t++)
		{
			findpirofly(int((8.0-FT)*100)+t);
		}
		for(t=(int((8.0-FT)*100)+int(FT*100));t<=8000;t++)
		{
			findpiro(t);
		}
		for(t=(int((8.0-FT)*100)+int(FT*100));t<=8000;t++)
		{
```

```
                        ccheck(b,t);
                    }
                }

                for(t=(int((8.0-FT)*100)+int(FT*100));t<=8000;t++)
                {
                    int s=0;
                    if(check[b][t+2400]>=8)
                    {
                        realcheck[b][t] = b;
                        cccheck[b]++;
                    }

                }

        }
        for(t=0;t<=10400;t++)
        {
                {
                     fprintf(fp1,"%d,",cccheck[t]);
                }
                fprintf(fp1,"₩n");
        }

        fclose(fp1);
}
```

E. Code for finding the appropriate meeting place – method 2

```
#include<stdio.h>
#include<math.h>
#define pi 3.141592653
#define T (24.0*51)
#define c3 0.00004
#define c4 0.0003
#define c5 0.0003
#define c6 0.00005
```

```
#define k 3.0
#define pc 0.6
#define dt 1
#define u 817.804247
#define Re 6371
#define Pf 0.672137

FILE *fp1, *fp2, *fp3, *fp4;

double Piro[999999];
double Pi[3000];
double thetai[20],thetaf[7000],phii[20],phif[7000];
double phi,FT;
int check[7000][15000];
int realcheck[7000][15000];
int cccheck[7000];

double rad(double x)
{
        return pi/180.0*x;
}
void IV(void)
{
        int i,a,b;
        for(i=0;i<=900000;i++)
        {
                Piro[i] = 0;
        }
}
void place(void)
{
        fp3 = fopen("place.txt","r");
        fp4 = fopen("airport.txt","r");
        int i;
        for(i=1;i<=11;i++)
        {
```

```
        fscanf(fp3,"%lf %lf",&thetai[i],&phii[i]);
        thetai[i] = rad(thetai[i]);
        phii[i] = rad(phii[i]);
    }



    for(i=1;i<=6977;i++)
    {
        fscanf(fp4,"%lf %lf",&thetaf[i],&phif[i]);
        thetaf[i] = rad(thetaf[i]);
        phif[i] = rad(phif[i]);
    }
    /*
    double t1 = -78.89;
    double t2 = -81.11;
    double p1 = -55.56;
    double p2 = -60;
    t1 = rad(t1);
    t2 = rad(t2);
    p1 = rad(p1);
    p2 = rad(p2);

    thetaf[1] = -77.8740;
    phif[1] = -34.62616;

    thetaf[2] = -79.75;
    phif[2] = -83.30;
    /*
    thetaf[1] = (5*t1+1*t2)/6.0;
    phif[1] = (5*p1+1*p2)/6.0;

    thetaf[2] = (5*t1+1*t2)/6.0;
    phif[2] = (3*p1+3*p2)/6.0;

    thetaf[3] = (5*t1+1*t2)/6.0;
    phif[3] = (1*p1+5*p2)/6.0;
```

```
	thetaf[4] = (3*t1+3*t2)/6.0;
	phif[4] = (5*p1+1*p2)/6.0;

	thetaf[5] = (3*t1+3*t2)/6.0;
	phif[5] = (3*p1+3*p2)/6.0;

	thetaf[6] = (3*t1+3*t2)/6.0;
	phif[6] = (1*p1+5*p2)/6.0;

	thetaf[7] = (1*t1+5*t2)/6.0;
	phif[7] = (5*p1+1*p2)/6.0;

	thetaf[8] = (1*t1+5*t2)/6.0;
	phif[8] = (3*p1+3*p2)/6.0;

	thetaf[9] = (1*t1+5*t2)/6.0;
	phif[9] = (1*p1+5*p2)/6.0;
	*/
}
double fp(double p)
{
	return 0.5-0.5*(tanh(k*(p-pc)));
}
double time(double t)
{
	while(t>24)
	{
		t-=24;
	}
	return t;
}
void flighttime(int i,int j)
{
	double the = acos(cos(thetai[i])*cos(thetaf[j])*cos(phii[i]-phif[j])+sin(thetai[i]) * sin(thetaf[j]));
```

```
        FT = Re*the/u;
}
double rhythm(int t)
{
        double ti = double(t)/100.0;
        if(ti<8.0)
        {
                return sin(2*pi/24.0*(ti-6.0-phi));
        }
        else if(ti<80.0)
        {
                return sin((2*pi/24.0)*((72.0+phi)/72.0)*(ti-6.0-(74.0*phi/(72.0+phi))));
        }
        else
        {
                return sin((2*pi/24.0)*(ti-6.0));
        }
}
double dphi(int a,int b)
{
        double delphi = phif[b]-phii[a]+pi;
        if(delphi<0) delphi+=(2*pi);
        if(delphi>2*pi) delphi-=(2*pi);
        return delphi-pi;
}

double c1(int t)
{
        return (c3+c4)/2.0+(c4-c3)/2*tanh(3*rhythm(t));
}
double c2(int t)
{
        return (c5+c6)/2.0+(c6-c5)/2*tanh(3*rhythm(t));
}

double piroprime1(double p,int t)
```

```
{
        return c1(t)*fp(p);
}
double piroprime2(double p,int t)
{
        return -c2(t)*p;
}
void findpiro(int t)
{
        t+=2400;
        double k1,k2,k3,k4;
        double ti = (double)t/100.0;
        ti = time(ti);

        if(ti>8)
        {
                k1 = piroprime1(Piro[t],t);
                k2 = piroprime1(Piro[t]+0.5*dt*k1,t+0.5*dt);
                k3 = piroprime1(Piro[t]+0.5*dt*k2,t+0.5*dt);
                k4 = piroprime1(Piro[t]+dt*k3,t+dt);
        }
        else
        {
                k1 = piroprime2(Piro[t],t);
                k2 = piroprime2(Piro[t]+0.5*dt*k1,t+0.5*dt);
                k3 = piroprime2(Piro[t]+0.5*dt*k2,t+0.5*dt);
                k4 = piroprime2(Piro[t]+dt*k3,t+dt);
        }
        Piro[t+1] = Piro[t] + (1.0/6.0)*dt*(k1+2*k2+2*k3+k4);
        if(Piro[t+1]<=0)
        {
               Piro[t+1]=0;
        }
}
void findpirofly(int t)
{
	t+=2400;
	double k1,k2,k3,k4;
	double ti = (double)t/100.0;
	ti = time(ti);

	k1 = piroprime2(Piro[t],t);
	k2 = piroprime2(Piro[t]+0.5*dt*k1,t+0.5*dt);
	k3 = piroprime2(Piro[t]+0.5*dt*k2,t+0.5*dt);
	k4 = piroprime2(Piro[t]+dt*k3,t+dt);
	Piro[t+1] = Piro[t] + (1.0/6.0)*dt*(k1+2*k2+2*k3+k4);
	if(Piro[t+1]<=0)
	{
		Piro[t+1]=0;
	}
}

void ccheck(int b,int t)
{
	t+=2400;
	double ti = (double)t/100.0;
	ti = time(ti);
	if(ti>8)
	{
		if(Piro[t]<0.672137)
		{
			//printf("a");
			check[b][t]++;
		}
	}
}

int main(void)
{
	fp1 = fopen("output.csv","w");
	fp2 = fopen("input.txt","r");
	int t;
int i;
int x;
place();
for(i=0;i<=2400;i++)
{
        fscanf(fp2,"%lf₩n",&Pi[i]);
}
int a=1,b=10;

//for(b=1;b<=1;b++)
{
        printf("b : %d₩n",b);
        //for(a=5;a<=5;a++)
        {
                IV();
                //printf("a : %d    b : %d₩n",a,b);
                phi = (24.0/2/pi)*dphi(a,b);
                flighttime(a,b);
                //printf("phi : %lf   T : %lf₩n",phi,FT);
                if(FT<8)
                {
                        x = int((8.0-FT)*100.0);
                        //printf("%lf₩n₩n",Pi[x]);
                        for(i=0;i<=4800;i++)
                                Piro[i] = Pi[x];
                }
                else
                {
                        x = int((32.0-FT)*100);
                        //printf("%lf₩n₩n",Pi[x]);
                        for(i=0;i<=4800;i++)
                                Piro[i] = Pi[x];
                }
                for(t=0;t<=int(FT*100);t++)
                {
                        findpirofly(int((8.0-FT)*100)+t);
                }
                for(t=(int((8.0-FT)*100)+int(FT*100));t<=8000;t++)
                {
                        findpiro(t);
                }
                for(t=(int((8.0-FT)*100)+int(FT*100));t<=8000;t++)
                {
                        ccheck(b,t);
                }
                /*
                for(t=(int((8.0-FT)*100)+int(FT*100));t<=8000;t++)
                {
                        fprintf(fp1,"%d₩n",check[b][t+2400]);
                }
                */

        }

        for(t=(int((8.0-FT)*100)+int(FT*100));t<=8000;t++)
        {
                int s=0;
                if(check[b][t+2400]>=8)
                {
                        realcheck[b][t] = b;
                        cccheck[b]++;
                }

        }

    }
/*
    for(b=1;b<=6977;b++)
    {
        fprintf(fp1,"%d₩n",cccheck[b]);
    }
*/
```

```
	for(t=0;t<=10400;t++)
	{
		//for(b=1;b<=10;b++)
		{
			fprintf(fp1,"%lf,",Piro[t]);
		}
		fprintf(fp1,"₩n");
	}

	fclose(fp1);
}
```

XI References

[1] Borb, A. A., & Achermann, P. (1999). Sleep homeostasis and models of sleep re gulation. *Journal of biological rhythms*, *14*(6), 559-570.