南北1次元エネルギーバランスモデルを用いた全球凍結条件の研究

岡山大学 理学部 地球科学科

05422510　鈴木 遼子

2013/02/14

要旨

南北 1 次元エネルギーバランスモデルを用いた数値計算プログラムを作成し, 太陽定数, 二酸化炭素の量, 熱拡散係数, 自転軸傾斜角の数値を変えて計算を行った. その結果に基づき, 全球凍結に陥る, 凍結状態から脱出する条件を含む気候の振舞いについて考察した.

本研究では時間変化しない年平均の太陽放射を与えて, その条件の下で平衡状態になったときの温度を計算した. 結果は, 太陽定数と二酸化炭素量は, どちらも値が大きいほど全球凍結には陥りにくくなった. 熱拡散係数や, 自転軸傾斜角を変えた場合は, それぞれ南北の温度差, 太陽放射による加熱量の差が大きいときほど全球凍結に陥りやすく, また脱しやすいという結果となった.

目 次

第1章　序論

1.1　研究背景

昨今の地質学的な研究を通して, 地球の表層環境が全球凍結を始めとした大きな気候変動を幾度と無く経験してきたらしいことが明らかになってきた.(田近, 2009 ; 多田, 2013 他)

その時々の惑星の気候は様々な要素が複雑に関係し合って決まっている. 地球に関して言えば, 大きなものでは太陽が段々明るくなっていることが知られており, また二酸化炭素を始めとする温室効果気体の量も地球史を通して変動していることがわかっている. 他にも地表面における陸と海の比率やその位置, 大気海洋大循環による南北の熱輸送の変化, 公転軌道の離心率など様々な要素が気候の決定に関わっている. また地球の場合, 月がその周囲を回っているため, 自転軸の傾きは大きな変化をしてこなかったとされているが, 火星のように軸の安定しない惑星も存在し, 軸の変動も惑星の気候の決定に大きな影響を与えていると考えられている.

1.2　研究目的

本研究の目的は, 南北 1 次元エネルギーバランスモデルを用いた数値計算プログラムを作成し, そのプログラムを用いて計算を行い, 全球凍結を含めた気候の振舞いについて考察することである.

第2章　南北1次元 エネルギーバランスモデル

2.1　モデルの概要

エネルギーバランスモデルとは実際の気候を単純化したモデルの 1 つである. 南北 1 次元エネルギーバランスモデルにおいては, 経度方向には一様であることを仮定し, 各緯度帯に出入りするエネルギーフラックスで各緯度帯の温度変化率が決まると考える. 式で書くと次のようになる.

$$C\frac{\partial T(x)}{\partial t} = SF_s(x)(1 - A(T(x))) - F_e(pCO_2, T(x)) + \frac{\partial}{\partial x}D(1 - x^2)\frac{\partial T(x)}{\partial x} \quad (2.1)$$

左辺は温度の時間変化率で C は熱容量である. 右辺第 1 項が恒星放射による加熱, 第 2 項が惑星放射による冷却, 第 3 項が南北の熱輸送をそれぞれ表している.$T(x)$ は経度方向に平均した地表面温度,x は緯度の正弦.S は太陽定数で F_s は緯度ごとの値を出すための係数.A はアルベド,F_e は惑星放射である

以下, それぞれの項について述べる.

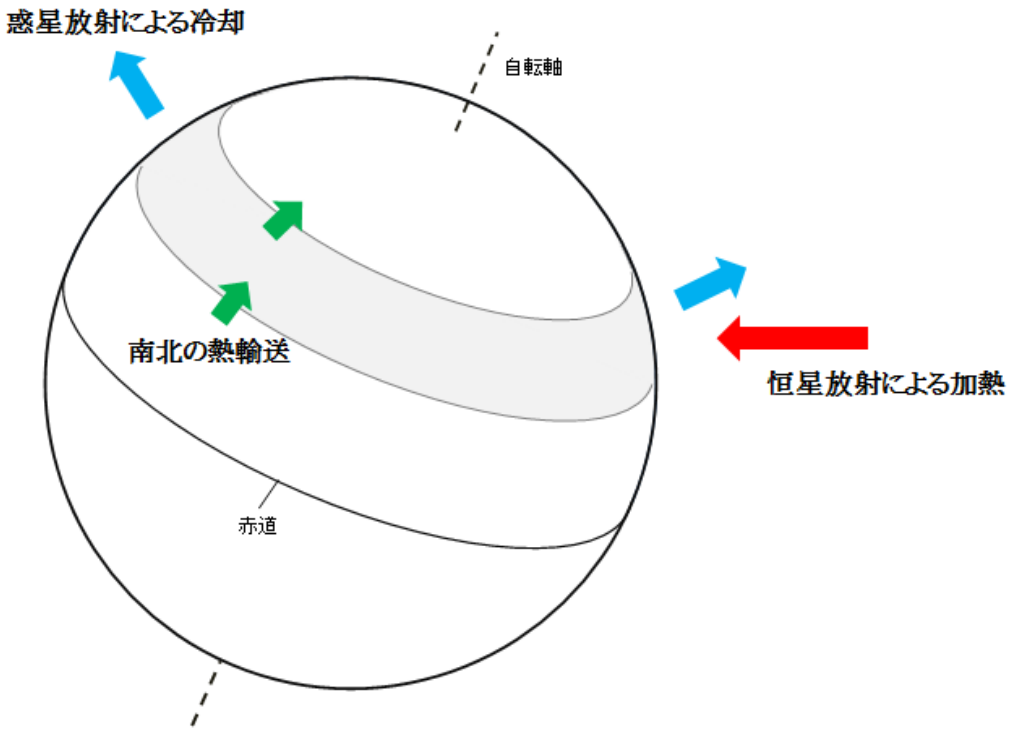

図 2.1: 南北 1 次元エネルギーバランスモデルの模式図.

2.1.1 太陽放射による加熱

惑星表層は恒星からの放射によって加熱される. 大気上端に入射する単位面積, 単位時間あたりの放射エネルギー (恒星放射フラックス) は, 緯度, 地方時によって異なり, また自転軸傾斜角や公転軌道の離心率などによって季節変化する. いくつかの自転軸傾斜角における年平均の太陽放射の緯度分布を図 2.2 に示す.

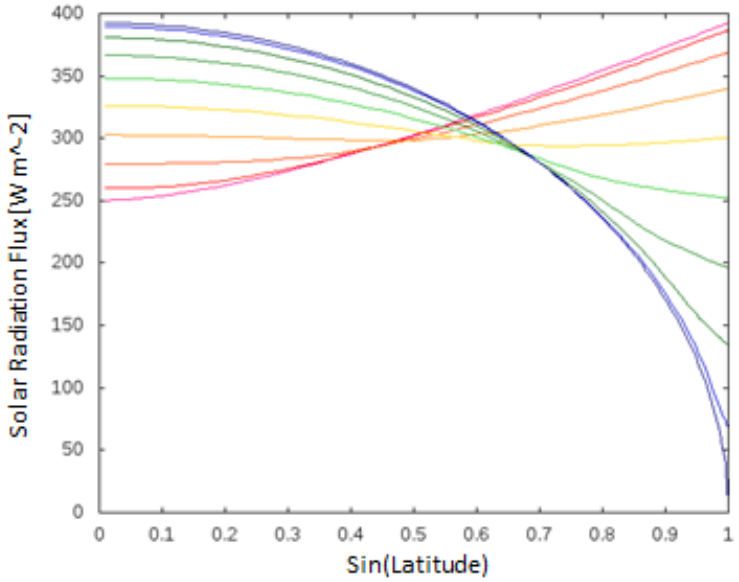

図 2.2: 放射フラックスの緯度分布. 横軸は緯度の正弦, 縦軸は年平均をとった太陽放射フラックス [Wm^{-2}]. 色の違いは自転軸傾斜角の違いに対応し,10 度刻みで青い線が傾斜角が 0 度, 赤い線が 90 度のときのフラックスである.

太陽放射量の計算

ある時刻 t に, 緯度 ϕ の地点における大気上端での単位面積当たりの太陽放射 I_t [W/m^2] は,

$$\begin{aligned} I_t &= S \times F_s \\ &= S \times \sin(s_h) \end{aligned} \tag{2.2}$$

$$\sin(s_h) = \sin\phi \sin\alpha + \cos\phi \cos\alpha \cos Z \tag{2.3}$$

$$\sin\alpha = \sin\delta \sin(L - L_0) \tag{2.4}$$

ここで,S は太陽定数,s_h は太陽高度角,α は赤緯,Z は時角,δ は自転軸傾斜角,L は黄径,L_0 は近日点経度である.

日の出または日没時の太陽高度角は 0 なので, このときの時角 Z_0 は式 (2.5) から求められる.

$$\cos Z_0 = -\frac{\sin\phi}{\cos\phi}\frac{\sin\alpha}{\cos\alpha} \tag{2.5}$$

ただし $\cos Z_0 <$ -1.0 となる場合は白夜であり $Z_0 = \pi$,1.0 $< \cos Z_0$ の場合は極夜で $Z_0 = 0$ とする.

よって日平均の太陽放射 I_{day} は

$$I_{day} = S \times \overline{\sin(s_h)} = S \times \frac{\int_{-Z_0}^{Z_0} (\sin\phi \sin\alpha + \cos\phi \cos\alpha \cos Z) dZ}{2\pi} \tag{2.6}$$

となる.

公転軌道を真円とする場合は惑星の黄径 L の時間変化率は一定なので, 積分して時間で割れば年平均の太陽放射が求まる.

2.1.2 惑星放射による冷却

大気を含むあらゆる物質は熱輻射を射出することによって冷却する. 大気上端から外向きに射出される単位面積, 単位時間あたりの放射エネルギー量を惑星放射フラックスと呼び, その大きさは大気の組成や総量, 地表面温度などに依存する.

本研究では惑星放射は地表面温度の関数として以下のような形で表すことができるものとした.

$$F_e = A + B \times T(x) \tag{2.7}$$

A,B の値は Caldeira and Kasting (1992) のものを用いて, 二酸化炭素分圧と温度に依存するようにした.

$$A = -326.4 + 9.161\varphi - 3.164\varphi^2 + 0.5468\varphi^3 \tag{2.8}$$

$$B = 1.953 - 0.04866\varphi + 0.01309\varphi^2 - 0.002577\varphi^3 \tag{2.9}$$

ここで,φ は基準である 300ppm で割った二酸化炭素分圧の自然対数. このパラメタライゼーションの有効範囲は $10^{-4}[\mathrm{bar}] < pCO_2 < 2[\mathrm{bar}]$, $194[\mathrm{K}] < T < 303[\mathrm{K}]$ である.

いくつかの二酸化炭素の量の場合の地表温度と地球放射フラックスの関係を図 2.3 に示す. 温度が高いほど放射フラックスは大きくなる. 二酸化炭素量が増えると, 温室効果が強くなり放射フラックスは減る.

2.1.3 南北熱輸送

南北の熱輸送は大陸の配置や分布, 大気海洋大循環などに左右されて変動する.

このモデルでは南北の熱輸送を拡散の形で表している. その拡散係数である D は, 以下のように大気圧に依存するものとした. ここで p_0 は基準の大気圧 (1atm) である. また, 境界条件として $\phi = 0, 90$ では熱輸送なしとした.

$$D = D_0 \times \frac{p_{total}}{p_0} \tag{2.10}$$

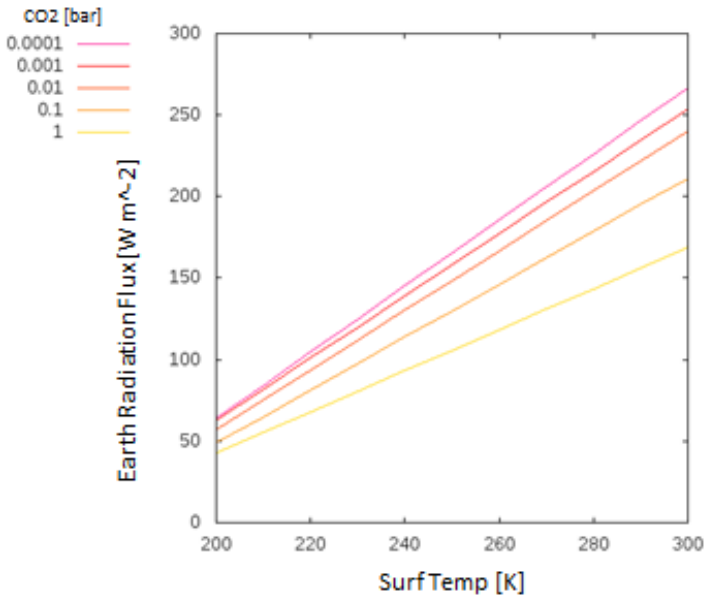

図 2.3: Caldeira and Kasting (1992) の近似式で計算された惑星放射フラックス. 横軸は地表面温度 [K], 縦軸は惑星放射フラックス [$\mathrm{Wm^{-2}}$]. 二酸化炭素の量が多いほど, その温度での放射フラックスは小さくなる.

2.1.4 アルベド

物体に放射エネルギーが入射するとき, エネルギーの一部は表面で反射される. 入射した放射量に対する反射した放射量の割合をアルベドと呼ぶ. アルベドは物質や表面の状態によって変わるが, 一般草木に覆われた土地よりも氷雪に覆われた土地の方がアルベドの値は大きくなる.

アルベドは散乱物質や雲の量にも依存するが, ここでは地表面の雪氷の有無による影響が大きいと考え, アルベドを温度の関数として以下のように与えた. アルベドの値とその温度依存性は, 真鍋 (1994) を参考に以下のようにとった. 氷の張り始める温度は 273K として,10K の幅を持って氷無しから全面氷のアルベドに変化するものとした.

$$A = \begin{cases} 0.3 & (T(x) \geqq 273K) \\ 0.62 - \frac{0.32}{10} \times (T(x) - 263) & (273K > T(x) \geqq 263K) \\ 0.62 & (263K > T(x)) \end{cases}$$

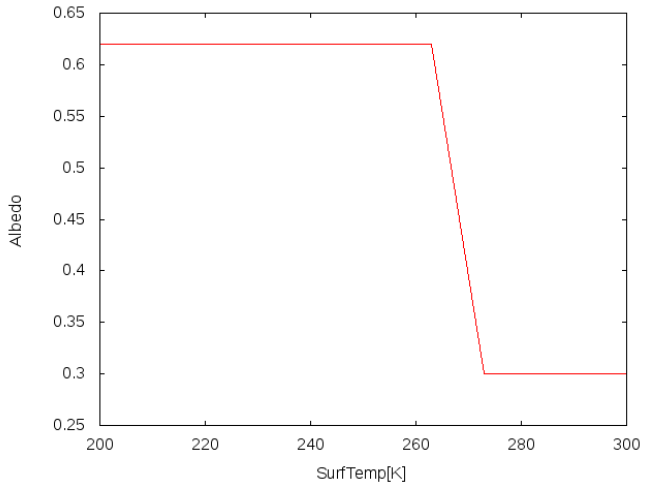

図 2.4: アルベドの温度依存性. 横軸は地表面の温度, 縦軸はアルベドの値. 地表面に氷が張るとアルベドの値は大きくなり, 氷が融けると小さくなる.

2.2 数値計算

式 (2.1) に初期条件を与え, 定常状態に達するまで時間積分することで表面温度の平衡解を求めた.

初期条件として, 氷無しの状態と全面氷の状態の 2 通りを与える. 本研究では惑星放射の季節変化を考えず, 年平均日平均の放射を与えた.

2.2.1 現在の地球

図 2.5 は表 2.1 のパラメタを与えて氷の無い状態から定常状態まで時間積分した結果である.

アルベドは式 (2.11) のように与え, 太陽定数と自転軸傾斜角を現在の地球と同じ値にし, 二酸化炭素の量と熱拡散係数の値は Ikeda and Tajika (1999) 等を参考した. 図 2.5 の青色の線はその計算結果, 赤色の十字は (Hartmann, 1999) である.

現在の地球を想定したこの計算は, おおよそ現在の地球を再現することができた.

表 2.1: 現在の地球設定のパラメタの値

S	:	太陽定数	1370	$\mathrm{Wm^{-2}}$
pCO_2	:	二酸化炭素分圧	3.3×10^{-4}	bar
D_0	:	熱拡散係数	0.6	$\mathrm{Wm^{-2}K^{-1}}$
δ	:	自転軸傾斜角	23.4	degree

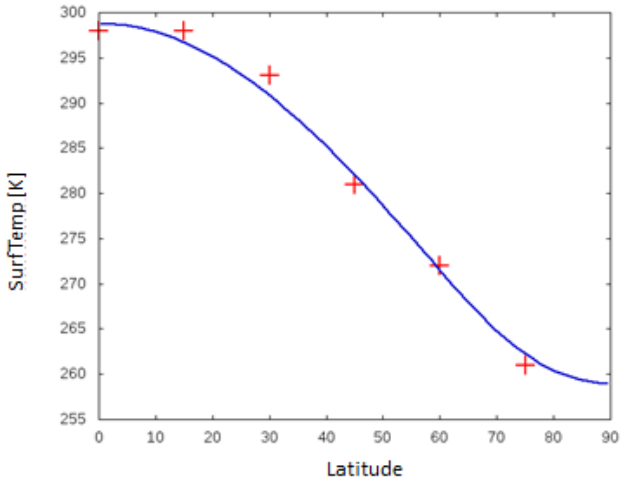

図 2.5: 地表面温度の緯度分布. 横軸は緯度 (北半球)[degrees], 縦軸は地表面温度 [K],

2.2.2 アイスアルベドフィードバック

二酸化炭素の量を変えたときの全球平均地表面温度の変化を. 図 2.6 に示す. 太陽定数は現在の値の 95 %である. 赤い点は氷の無い状態, 青い点は全面に氷が張っている状態を初期状態とした場合の結果である. アルベドに温度依存がある場合は多重平衡解が現れることが知られているが, この結果でも 2 本のブランチが現れた. 上のものは融けているもしくは一部だけが凍結した状態の解, 下は全球凍結した状態の解である.

二酸化炭素が増加すると温室効果が高まり平均温度は徐々に上がっていく. しかし, 二酸化炭素が 0.5[bar] ほどのところで全面に氷が張った解が消え, 氷が融けた解だけとなる. 全面に氷が張った状態で二酸化炭素が増えていった場合, 二酸化炭素が 0.5[bar] を超えたところで温度は一気にジャンプする. このような急激なジャンプは, 氷が融けてアルベドが小さくなることが原因で起こる. 一箇所でも氷が融けると, その部分で吸収される放射エネルギーが増えて加熱が強まり, さらに周囲を融かす, この過程が繰り返されることで周囲の温度は加速度的に変化する. この効果はアイスアルベドフィードバックと呼ばれ, 凍りつく際も同様に働く. 全球凍結していない状態で二酸化炭素を減らしていくと, 二酸化炭素が 0.001[bar] を下回ったところで凍結して温度が一気に下がる.

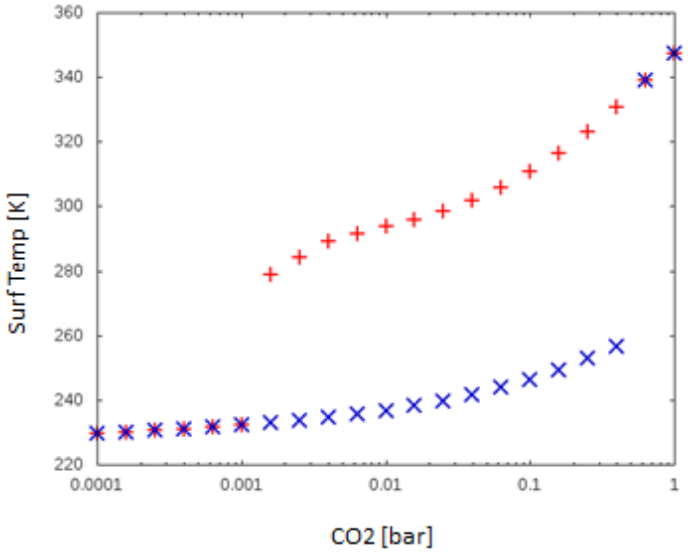

図 2.6: 二酸化炭素量が変化したときの全球平均温度の変化.
縦軸は全球平均地表面温度 [K], 横軸は二酸化炭素量 [bar].

二酸化炭素の減少 (この場合では 0.001[bar] 以下) によって一旦全球凍結に陥ると, 二酸化炭素の温室効果だけでその状態から脱出するためには, 陥ったときよりもずっと多くの二酸化炭素 (この図では 100 倍以上の二酸化炭素) が必要になる.

第3章　実験

現在の地球を基準に, 太陽定数, 二酸化炭素の量, 熱拡散係数, 自転軸傾斜角の値を変えて計算しそれぞれの変化に対して氷の張り出す緯度がどのように変化するかを調べる.

3.1　実験設定

南北 1 次元エネルギーバランスモデルに, 全面氷の状態と氷のまったく無い状態の 2 通りの初期条件として与え, 定常状態になるまで時間積分した. 二酸化炭素量と太陽定数を変化させた場合 (**3.2.1**), 二酸化炭素量と自転軸傾斜角を変化させた場合 (**3.2.2**), 二酸化炭素量と熱拡散係数を変化させた場合 (**3.2.3**) の 3 つの場合について氷の張り出す緯度の変化を見た.

それぞれの変数のデフォルトの値は表 2.1, 変化させる範囲は表 3.1 とした.

表 3.1: 変数の範囲

S	:	0.6 〜 1.4	(相対比)
pCO_2	:	1.0×10^{-4} 〜 1.0	bar
D_0	:	0.01 〜 1.0	$\mathrm{Wm^{-2}K^{-1}}$
δ	:	0 〜 90	degree

3.2 結果

3.2.1 二酸化炭素と太陽定数を変えた場合

図 3.1 は初期状態が氷なしの場合, 図 3.2 は全面氷の場合の結果である. 横軸は二酸化炭素の量 [bar], 縦軸は太陽定数の相対比である. 線は 5 度刻みで, 青色が 0 度で赤色が 90 度. 赤い線より右側では氷がまったく無い状態, 青い線の左側では全球凍結状態が平衡解となる.

２つの図の形が異なるのは,**2.2.2** で述べたアルベドによる多重平衡解が現れているためである. どちらの図でも太陽定数が大きくなると, 恒星放射による加熱が強まるので, 氷の張る緯度は極側へ後退する. また二酸化炭素が増えた場合も, その温室効果によって冷却が減るため極側へ後退する.

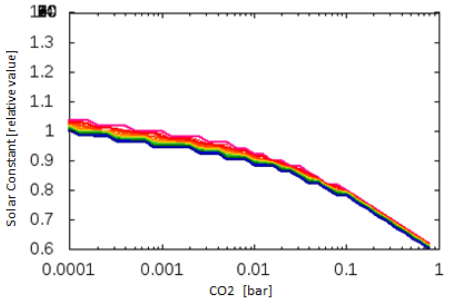

図 3.1: 初期状態:氷無し

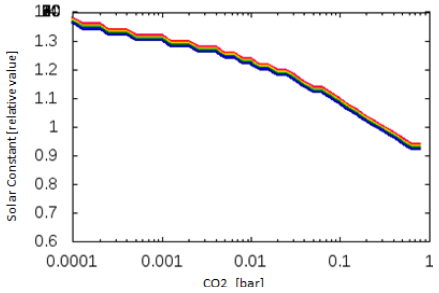

図 3.2: 初期状態:全面氷

太陽は徐々に明るくなっていることが知られ, 地球が誕生した 45 億年前は 7 割ほどの明るさだったといわれている. 計算に用いたプログラムはいくつもの要素を省略しているが, この結果だけを見るならば, 仮に二酸化炭素の量が現在と同程度 ($3.0 \sim 4.0 \times 10^{-4}$ [bar]) で変化しなかったとすれば, ずっと全球凍結の状態が続いていたということになる. また多重平衡解が存在するため, 温室効果気体の量が変動する場合でも, 一度全球凍結に陥ると太陽放射や二酸化炭素の量が大きく変化しなければその状態から脱出することはできないことがわかる.

3.2.2 二酸化炭素と自転軸傾斜角を変えた場合

次に二酸化炭素と自転軸傾斜角を変えた場合の氷の張る緯度の変化を見る. 図 3.3 と図 3.5 は氷の無い状態を, 図 3.4 と図 3.6 は全面氷の状態を初期状態とした結果である. 横軸は二酸化炭素, 縦軸は自転軸傾斜角. 先ほどと同様に線は 5 度刻みで, 青色が 0 度で赤色が 90 度. 赤い線より右側では氷がまったく無い状態, 青い線の左側では全球凍結状態が平衡解となる. また図 3.3 と図 3.4 は太陽定数が相対比で 1 の場合, 図 3.5 と図 3.6 は 0.9 の場合である.

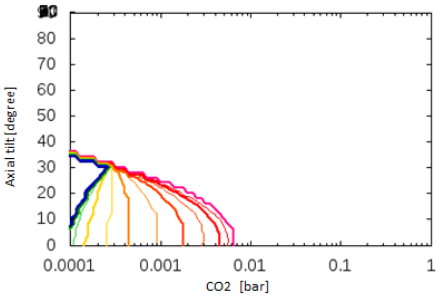

図 3.3: 初期状態:氷無し

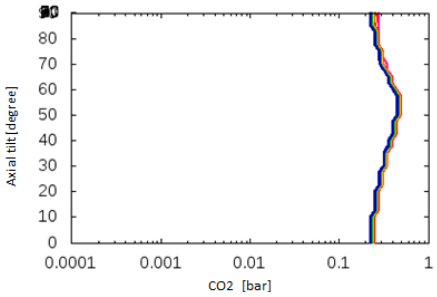

図 3.4: 初期状態:全面氷

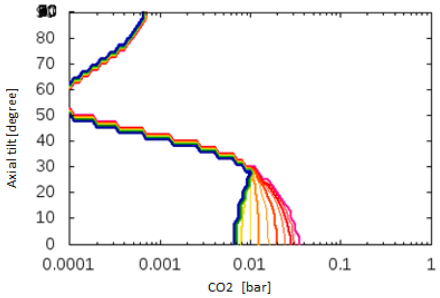

図 3.5: 初期状態:氷無し
太陽の明るさ:現在の 90 %

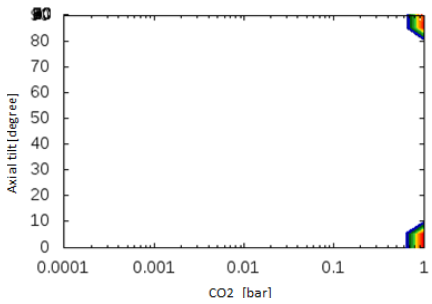

図 3.6: 初期状態:全面氷
太陽の明るさ:現在の 90 %

このような結果になった理由は, 太陽放射の分布による加熱の違いから説明できる. 自転軸傾斜角が変わると恒星放射フラックスは図 2.2 のように変化するが, その場合の地表面温度分布は図 3.8 や図 3.9 のようになる. 縦軸は地表面温度, 横軸は緯度の正弦である. 水色の破線は, それぞれ図 3.7 の黒い破線に対応した氷の張り始める温度（273[K]）である. 傾斜角ごとの色は図 2.2 と対応している. またこれらの図はアルベドを温度に関わらず 0.3 としている.

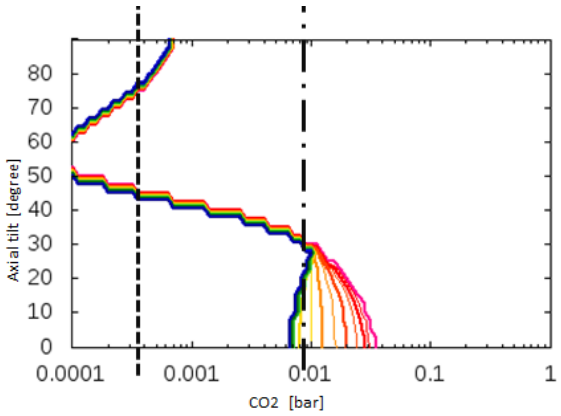

図 3.7: 図 3.5 と同じ.

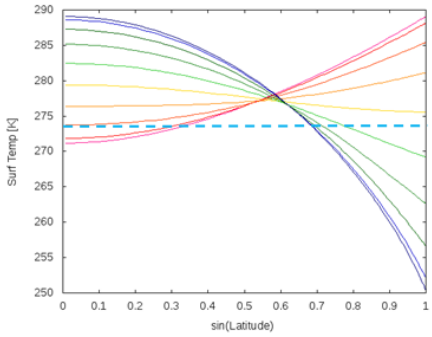

図 3.8: 二酸化炭素が 0.0003[bar].

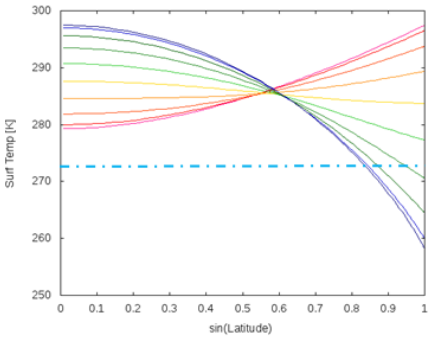

図 3.9: 二酸化炭素が 0.008[bar]

先ず図 3.5(3.7) について見る. 二酸化炭素量が 0.0003[bar] ほどの場合 (左の黒い破線),0 度から 40 度までと 75 度以上の傾斜では全球凍結状態、60 度前後のときは氷の無い状態が定常解となる. 図 3.8 を見ると, 傾斜が 80 度 (青) 以上, あるいは 40 度 (緑) 以下場合, 赤道, もしくは極側の温度が氷の張り始める温度である 273[K] を下回っている. よってそこから氷が張り始め, アイスアルベドフィードバック効果によって全体の温度も下がり全球凍結に陥ったと考えられる. 逆に 60 度 (黄) 前後の場合は温度の傾きがほぼ無く,273[K] を下回っていないためどこも凍りださず、平衡解も氷の無い状態となる. また, 仮にこの条件で自転軸傾斜各が 60 度だったものが,0 度側か 90 度側に傾くと, それだけで全球凍結になることもわかる.

もう少し二酸化炭素量が増えると凍りつく温度のラインが相対的に下へ下がる. 二酸化炭素が 0.003[bar] ほどの場合, 傾斜角が 50 度以上ならどこも 273[K] を下回らないため凍りださず, 氷の無い状態が平衡解となる. 更に二酸化炭素量が増え

0.03[bar] 程になると, 軸がほぼ直立している場合のみ, 極域だけが氷の張る温度を下回る. 二酸化炭素量が多く, 全体としては暖かいため氷はほとんど張らないが, 軸が直立の場合は極の加熱がとても小さいので, 極側の一部分だけは凍ってしまう.

最後に, 二酸化炭素量が 0.01[bar] ぐらいのときを見る. 図 3.7 を見ると 30 度くらいでは全球凍結に陥っていて, 直立している場合はほとんど凍るが, 赤道のみ融けていて全球凍結には陥っていない. これまで同様, 軸の傾きが小さい場合は極側に氷が張り, アイスアルベドフィードバックによって温度が下がってく. 傾斜が 30 度くらいの場合はその効果で全球凍結に陥る. しかし軸が直立している場合は,30 度のときと比べて赤道が強く加熱されている. そのため赤道だけは凍結を免れる.

全球凍結状態から抜ける場合を見る（図 3.4）. 先ほどまでとは逆で, 傾斜が 0 度や 90 度のときのように局所的に強く加熱される場合は, 強く加熱された場所から氷が融けだして全球凍結から脱出する.50 度付近は加熱が一様でアイスアルベドフィードバックが働かないため, もっとも融けにくくなる.

3.2.3 二酸化炭素と熱拡散係数を変えた場合

最後に二酸化炭素と熱拡散係数を変えた場合の結果を見る. 図 3.10 と図 3.11 はそれぞれ氷の無い状態, と全面氷の状態を初期状態とした場合の結果である. 横軸は二酸化炭素 [bar], 縦軸は熱拡散係数 [$\mathrm{Wm^{-2}K^{-1}}$]. 線の色は **3.2.1** と同様である.

熱拡散係数が大きくなるほど, 定常状態に達したときの南北の温度差は小さくなる. 自転軸傾斜角は現在の地球と同じなので恒星放射による加熱には偏りがあるが, 熱拡散係数が大きい場合周囲から熱が流れ込んでくるためなかなか凍結する温度まで下がらない. 同様の理由で全球凍結からも脱しにくい. 逆に熱拡散係数が小さい場合, 加熱の弱い極側は氷が張ってしまう (図 3.10). 全球凍結から抜ける場合 (図 3.11) でも, 加熱の集中する赤道では熱が運ばれていかないので, 融けやすい. しかし加熱の弱い極へもほとんど熱が運ばれないため, 極の氷はなかなかと融けきらない.

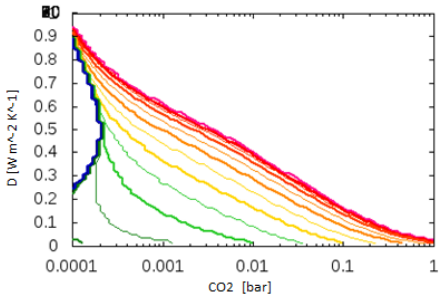

図 3.10: 初期状態:氷なし

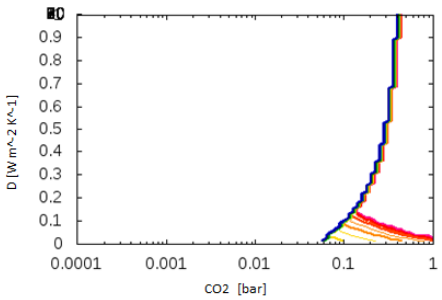

図 3.11: 初期状態:全面氷

3.3 考察

非常に限定された条件と仮定のもとでの結果ではあるが, 太陽の明るさや二酸化炭量を始めとする温室効果気体, 南北での熱輸送などは地球史を通して変化してきたと考えられており, このようなシミュレーションの結果も, 地質学的研究とあわせることで過去の地球環境を推測するための指標となりうると考えられる.

自転軸の傾斜については, 地球の場合は月の影響で変化は微々たるものであったとされているが, 計算結果を見ると傾斜が変動するだけで全球凍結に陥ったり, 逆に凍結状態から抜け出すこともあるうることがわかった. 一旦全球凍結状態に陥ると環境が大きく変動しない限りそこから抜けることは難しいため, ある意味で安定であると言える. しかし長期的に見れば大きな変動を引き起こす要素であることには変わりないので, 自転軸が安定であったことは地球環境にとっては幸運であったと言えるかもしれない.

とはいえ, 本研究はあくまで年平均の恒星放射を与えた場合の結果であり, 本来のように季節変化を考えるのであれば話はまた違ってくる. 特に自転軸の傾斜が大きくなるほど, 季節ごとの放射の差は大きくなる. また, 季節変化を考えるならば熱容量も重要な要素となってくるため地表の陸海比や海氷の有無なども問題となり, 一層複雑になる.

季節ごとの差が大きいと, 例えば夏には加熱が極端に大きく, 冬にはほとんど無いといった場合も考えられる. もし冬の間に凍ってしまうと, アルベドが大きくなり, 夏になって放射が増えても吸収するエネルギーが少なくなるため氷が融けず, 最終的に全球凍結状態に陥る可能性もある. 逆に夏に極端に強く加熱される場合, 冬場の放射量にさほど関わらず, 夏には必ず氷が融けるという可能性もある. その惑星が全球凍結に陥るかどうかを決めるのが冬の放射量の少なさなのか, それとも夏の強い加熱なのかは, はっきりとわかっていない.

第4章　まとめ

南北 1 次元エネルギーバランスモデルを用いたプログラムを作成し, 年平均日射を与えて太陽定数, 二酸化炭素量, 熱拡散係数, 自転軸傾斜角の値を変えた計算を行い, 全球凍結に陥る, 凍結状態から脱出する条件を含めた気候の振舞いについて考察を行った.

古気候や他の惑星の気候の振舞いについて, さらに踏み込んだ研究をするためには, 今回扱わなかった恒星放射の季節変化や熱容量に関わる地表面の陸海比など取り込んだモデルが必要になる. それらは今後の課題である.

謝辞

本研究をおこなうにあたり, ご指導いただきました指導教員である はしもとじょーじ 准教授に心より感謝致します. また, 日常の議論を通じて多くの知識や示唆を頂いきました同研究室の先輩方や同期の皆様にも感謝いたします

参考文献

Caldeira, K. and J. F. Kasting, 1992 : Susceptibility of the early Earth to irreversible glaciation caused by carbon dioxide clouds, Nature,359, 226-228.

Ikeda, T. and E. Tajika, 1999 : A study of the energy balance climate model with CO2-dependent outgoing radiation: implication for the glaciation during the Cenozoic, Geophysical Research Letters, Vol.26, NO.3, 349-352

Hartmann, D. L. 1994 : Global Physical Climatology, ACADEMIC PRESS, pp.411.

高山歌織, 2002 : 火星大気- 極冠システムの一次元エネルギーバランスモデルについて, 北海道大学修士論文.

多田隆治, 2013 : 気候変動を理学する, みすず書房, pp.287

田近英一, 2009 : 凍った地球 - スノーボールアースと生命進化の物語, 新潮社, pp.195

真鍋淑郎, 1984 : 気候変動論, 増田耕一編

付録

ソースコード

以下に今回作成したプログラムのソースコードを示す. このプログラムは初期条件 (太陽定数, 二酸化炭素の量, 熱拡散係数, 自転軸傾斜角, 初期地表面温度) を与えて定常状態まで時間積分し, 表面温度の平衡解を求めるものである.

```
1  module constant
2   implicit none
3  !! constant !!
4   real*8,parameter :: pi  = 3.14159265358979323846264338327950288d0
5   real*8,parameter :: Cp  = 1006d0    !specific heat         [W s / kg / K]
6   real*8,parameter :: S   = 1370d0    !solar constant        [W / m^2]
7   real*8,parameter :: RE  = 6400d3    !Earth radius          [m]
8   real*8,parameter :: e   = 0.016d0   !revolution eccentricity
9   real*8,parameter :: AM  = 10.0d4    !normal Air mass(?)    [kg / m^2]
10  real*8,parameter :: aE  = 1.0d0     !normal Air            [bar]
11  real*8,parameter :: CO20= 300.0d-6  !normal CO2            [bar]
12 !! variation default !!
13  real*8,parameter :: Ar  = 0.3d0     !regolis Albedo
14  real*8,parameter :: Ai  = 0.62d0    !Ice Albedo
15  real*8,parameter :: Tg  = 263.0d0   !freezing Temp         [K]
```

```
16    real*8,parameter :: Tr  = 273.0d0   !regolis  Temp          [K]
17    real*8,parameter :: h   = 3660.0d0  !DELTA-time             [second]
18    real*8,parameter :: CC  = 0.01d0    !convergence condition [W / m^2]
19   end module constant
20   program EBM
21    use constant
22    implicit none
23     real*8,dimension(90) :: Ts         !surfTemp              [K]
24     real*8,dimension(90) :: Sl         !YearAverageCoef
25     real*8 dphi                        !DELTA-Lat             [radian]
26     integer :: n = 90                  !Area number(Area-i)
27   ! default value !
28     real*8 :: Ts0   = 300.0d0          !surfaceTemperture     [K]
29     real*8 :: Sce   = 1.0d0            !SunCoefficient
30     real*8 :: del   = 23.4d0           !AxialTilt             [degree]
31     real*8 :: D     = 0.6d0            !ThermalDiffusionCoef  [W / m^2 / K]
32     real*8 :: CO2   = 330.0d-6         !CO2                   [bar]
33   ! loop variable !
34     integer i                          !Lat
35     integer :: k1 = 1                  !Sce
36     integer :: k3 = 1                  !D
37     integer :: k2 = 1                  !delta(AxialTilt)
38     integer :: k4 = 1                  !CO2
39   !! DELTA-Lat !!
40   dphi = pi / (2.0d0 * n)
```

```
41  !! file open !!
42  open(10, file ='***.dat',status='replace')
43  open(20, file ='***Iceline.dat',status='replace')
44   write(20,*) "# CO2","# D","AxisTilt(delta)","SunCoef(Sce)",&
45               "# IceLat(i)","# IceLat(sin(i))","SurfTemp(i)"
46   write(10,*) "# lat(i)","# Lat(sin(i))","# SurfTemp(i)",&
47               "# AxisTilt(delta)","# SunCoef(Sce)","# D","# CO2"
48  !!!! VARIATION LOOP !!!!
49  !! Loop del !!
50  do k2 = 1,46
51    del = 0.0d0
52    del = - 2.0d0 + 2.0d0 * k2   !
53   !!! Caliculation Sl !!!
54   call AxisTilt(Sl,del,dphi)
55  !! Loop Sun coefficient !!
56    do k1 = 1,41
57      Sce = 0.0d0
58      Sce = 0.58d0 + 0.02d0 * k1
59  !! Loop D !!
60      do k3 = 1,40
61         D = 0.0d0
62         D = 0.025d0 * k3
63  !! Loop CO2 !!
64        do k4 = 1,41
65          CO2 = 0.0d0
66          CO2 = 10.0d0 ** (-4.1d0 + 0.1d0 * k4)
```

```
67  !!!! MAIN CALICULATION !!!!!!
68            call fluxCal(Ts,Ts0,Sl,Sce,D,CO2,dphi,n)
69  !! loop check !!
70              write(*,*) "delta  Sce  D  CO2 :" , k2, k1, k3, k4
71  !!! WRITING !!!
72   !! Temperture gradient !!
73          do i = 1,n
74           write(10,*) (i-0.5),sin((i-0.5)*dphi),Ts(i),del,Sce,D,CO2
75          end do
76           write(10,*) "  "
77           write(10,*) "  "
78   !! Ice line Lat !!
79           if(    Ts( 1) < Tg)then
80                  write(20,*)  CO2, D, del, Sce, 0,  0.0d0, Ts(1)
81           elseif(Ts(90) > Tg)then
82                  write(20,*)  CO2, D, del, Sce, 90, 1.0d0, Ts(90)
83           else
84              do i = 2,n
85               if(Ts(i) < Tg  .and. (Ts(i-1) > Tg)) then
86                write(20,*) CO2,D,del,Sce,i-1,sin((i-1)*dphi),Ts(i-1),Ts(i)
87                exit
88               end if
89              end do
90           end if
91          end do !k4
92        write(20,*) "  "
```

```
93        end do !k1
94      end do !k3
95    !    write(20,*) "  "
96    end do !k2
97    !! close file !!
98    close(20)
99    close(10)
100   end program EBM
101   subroutine fluxCal(Ts,Ts0,Sl,Sce,D,CO2,dphi,n)
102    use constant
103    implicit none
104   real*8,dimension(90) :: Ts           !surfTemp              [K]
105   real*8,dimension(90) :: Sl           !YearAverageCoef
106   real*8 Ts0                           !initial surfTemp      [K]
107   real*8 Sce                           !SunCoefficient
108   real*8 D                             !ThermalDiffusionCoef  [W/m^2/K]
109   real*8 CO2                           !CO2                   [bar]
110   real*8 dphi                          !DELTA-Lat             [radian]
111   integer n
112   real*8,dimension(90) :: FS  = 0.0d0  !Sun Flux at Area-i    [W / m^2]
113   real*8,dimension(90) :: sE  = 0.0d0  !surface area of Area-i [m^2]
114   real*8,dimension(90) :: FE  = 0.0d0  !EarthFlux at Area-i   [W / m^2]
115   real*8,dimension(90) :: FD  = 0.0d0  !netDiffusionFlux      [W / m^2]
116   real*8,dimension(90) :: dF  = 0.0d0  !net Flux at Area-i    [W / m^2]
117   real*8,dimension(90) :: dTdt = 0.0d0 !DELTA-Temp/DELTA-time [K/s]
118   real*8 ppm                           !particalPressure of CO2 [bar]
```

```
119  real*8 lpg                              !ln of partical pressure
120  real*8 Tca,Tcb                          !CO2-Temp const(Caldeira&Kasting)
121  real*8 Tca2,Tcb2,ppm2,lpg2,FE2
122  real*8 A                                !Albedo
123  ! loop variable !
124  integer i                               !Latitude loop
125  integer day                             !day loop
126  integer sec                             !second loop
127  integer smax
128  smax = nint(86400.0d0 / h)
129  !!! initialize !!!
130  ppm  = 0.0d0
131  ppm2 = 0.0d0
132  do i = 1,n
133    Ts(i) = Ts0
134  end do
135  !!! CALICULATION !!!
136  !! surface area at i !!
137  open(91,file="sE90.dat", status="old")
138   read(91,*) sE
139  close(91)
140  !! Diffinision Temputure constant (Caldeira and Kasting) !!
141       ppm  = CO2 / (aE - CO20 + CO2)
142       lpg  = log(ppm / CO20)
143       Tca  = -326.4d0 + (9.161d0  * lpg)          &
```

```
144                       - (3.164d0  * lpg**2.0d0)    &
145                       + (0.5468d0 * lpg**3.0d0)
146       Tcb  = 1.953d0  - (0.04866d0  * lpg)         &
147                       + (0.01309d0  * lpg**2.0d0)  &
148                       - (0.002577d0 * lpg**3.0d0)
149       ppm2 = 2.0d0 /  (aE - CO20 + 2.0d0)
150       lpg2 = log(ppm2 / CO20)
151       Tca2 = -326.4d0 + (9.161d0  * lpg2)          &
152                       - (3.164d0  * lpg2**2.0d0)   &
153                       + (0.5468d0 * lpg2**3.0d0)
154       Tcb2 =  1.953d0 - (0.04866d0  * lpg2)        &
155                       + (0.01309d0  * lpg2**2.0d0) &
156                       - (0.002577d0 * lpg2**3.0d0)
157   !!! Caliculation SunFlux !!!
158             call SunFlux(FS,Sl,Sce,n)
159   !! Loop time !!
160             do day = 1,100000000
161              do sec = 1,smax
162   !!! Caliculation DiffusionFlux !!!
163               call fluxD(Ts,FD,D,CO2,dphi,n)
164   !! Loop Latitude !!
165               do i = 1,n
166                 FE(i) = Tca + Tcb * Ts(i)
167                   if(Ts(i) < 200d0)then
168                     FE2 = 0.0d0
```

```
169                      FE2 = Tca2 + Tcb2 * Ts(i)
170                     if((FE(i) < 0.0d0).and.(FE2 < 0.0d0))then
171                        FE(i) = 0.0d0
172                     else if(FE(i) < FE2)then
173                        FE(i) = FE2
174                     end if
175                   end if
176   !! Albedo !!
177                    if     (Ts(i) < Tg)then
178                       A = Ai
179                    else if(Ts(i) > Tr)then
180                       A = Ar
181                    else
182                       A = Ai - ((Ai-Ar)/(Tr-Tg)) * (Ts(i)-Tg)
183                    end if
184   !! net flux !!
185                 dF(i) =   FS(i) * (1.0d0 - A)                    &
186                         - FE(i)                                  &
187                         + FD(i) * ((2.0d0 * pi * RE**2.0d0)/ sE(i))
188   !! DELTA-Temp / DELTA-time !!
189                 dTdt(i) = dF(i) / (Cp * AM) !!!!
190                 Ts(i)   = Ts(i) + dTdt(i) * h
191               end do! i
192               end do! sec
193   !! Check convergence condition !!
194      do i = 1,n
```

```
195         if(abs(dF(i)) > CC)then
196            go to 10
197         end if
198       end do
199   exit
200   10 continue
201              end do !day
202   !! check dF !!
203              do i = 1, n
204                write(*,*) (i-0.5), Ts(i), dF(i)
205              end do
206               write(*,*) "  "
207   !! check time !!
208      write(*,*) "[day second] : ", day, sec
209   end subroutine fluxCal
210   subroutine SunFlux(FS,Sl,Sce,n)
211    use constant
212    implicit none
213   real*8,dimension(90) :: FS           !Sun Flux at Area-i    [W / m^2]
214   real*8,dimension(90) :: Sl           !YearAverageCoef
215   real*8 Sce                           !Sce
216   integer n                            !Area number
217   integer i                            !Latitude loop
218   !! initializing !!
219   do i= 1,90
220     FS(i) = 0.0d0
```

```
221   end do
222   !! Loop Lat !!
223     do i = 1,90
224      FS(i) = Sce * S * Sl(i)
225     end do
226   end subroutine SunFlux
227   subroutine fluxD(Ts,FD,D,CO2,dphi,n)
228    use constant
229    implicit none
230   real*8,dimension(90) :: Ts           !surfTemp              [K]
231   real*8,dimension(90) :: FD           !netDiffusionFlux      [W / m^2]
232   real*8 D                             !ThermalDiffusionCoef  [W / m^2 / K]
233   real*8 CO2                           !CO2                   [bar]
234   real*8 dphi                          !DELTA-Lat             [radian]
235   integer n
236   real*8,dimension(90) :: uFD          !DiffusionFLux         [W / m^2]
237   real*8 phi                           !latitude
238   real*8 Dp                            !D depending on atm    [W / m^2 / K]
239   real*8 airtotal                      !total Air Pressure    [bar]
240   integer i
241   !! initializing !!
242   airtotal = 0.0d0
243   Dp       = 0.0d0
244    do i = 1,90
245      FD(i) = 0.0d0
246    end do
```

```
247  !! diffinision Dp !!
248  airtotal = (aE - CO20) + CO2
249  Dp = D * (airtotal / aE)
250  !! Loop !!
251    ! uFD caliculation  !
252    do i= 1,n-1
253      phi = dphi * i
254      uFD(i) = Dp * cos(phi) * (Ts(i) -Ts(i+1)) / dphi
255    end do
256    ! FD caliculation !
257    do i = 1,n
258      if (i == 1) then
259       FD(i) = (         - uFD(i))
260      else if (i == n) then
261       FD(i) = (uFD(i-1)         )
262      else
263       FD(i) = (uFD(i-1) - uFD(i))
264      end if
265    end do
266  return
267  end subroutine fluxD
268  subroutine AxisTilt(Sl,del,dphi)
269  use constant
270  implicit none
271  real*8,dimension(90)  :: Sl          !YearAverageCoef
272  real*8 del                           !AxialTilt              [degree]
```

```
273  real*8 dphi                                !DELTA-Lat             [radian]
274  real*8 L0                                  !Long of perihelion    [radian]
275  real*8 dL                                  !DELTA-L               [radian]
276  real*8 phi                                 !Latitude
277  real*8 phi0
278  real*8 Z                                   !hour angle            [radian]
279  real*8 dZ                                  !DELTA-hour angle      [radian]
280  real*8 delrad                              !AxialTilt             [radian]
281  real*8 rad                                 !degree >> radian
282  real*8,dimension(90) :: cosZ = 0.0d0 !cos(Z)
283  real*8,dimension(365) :: L = 0.0d0   !Celestial Long        [radian]
284  real*8,dimension(90,365) :: sh=0.0d0 !sin(SunHeight)
285  integer i, j, kd
286  rad  = pi / 180.0d0
287  phi0 = -0.5d0 * dphi
288  dL   = 2.0d0*pi / 365.0d0
289  L0   = pi/2.0d0
290  dZ   = pi / 180.0d0 / 4.0d0
291  !! initialize !!
292  delrad = 0.0d0
293  do j=1,365
294   do i = 1,90
295    Sl(i) = 0.0d0
296    sh(i,j) = 0.0d0
297    cosZ(i) = 0.0d0
```

```
298     end do
299   end do
300   !! degree >> radian !!
301   delrad = del * rad
302   !!! Loop revolution !!!
303   do j = 1,365
304     L(j) = dL * j
305   !!!! Loop rotation !!!!
306     do i = 1,90
307       phi =  - phi0 + dphi * i
308       cosZ(i) = - sin(phi) *  sin(delrad)*sin(L(j)-L0)
309                   / cos(phi) / (sqrt(1-(sin(delrad)*sin(L(j)-L0))**2.0d0))
310         if(cosZ(i) < -1.0d0) then
311           do kd = 1,1440
312             Z = dZ * kd
313             sh(i,j) = sh(i,j)
314                + ((((1.0d0 + e*cos(L(j))) / (1.0d0 - e**2.0d0)   ) **2.0d0) &
315                * (sin(phi)*sin(delrad)*sin(L(j)-L0)                          &
316                + cos(Z)*cos(phi)*sqrt(1-(sin(delrad)*sin(L(j)-L0))**2.0d0)))
317           end do
318         else if(cosZ(i) > 1.0d0)then
319             sh(i,j) = sh(i,j)
320         else
321           do kd = 1,1440
322              Z  = -acos(cosZ(i)) + dZ * kd
```

```
323             if(Z > acos(cosZ(i)))exit
324             sh(i,j) = sh(i,j)
325                + ((((1.0d0 + e*cos(L(j))) / (1.0d0 - e**2.0d0)   ) **2.0d0) &
326                + (sin(phi)*sin(delrad)*sin(L(j)-L0)                          &
327                + cos(Z)*cos(phi)*sqrt(1-(sin(delrad)*sin(L(j)-L0))**2.0d0)))
328           end do
329         end if
330     end do
331   end do
332   do i = 1,90
333     do j = 1,365
334       Sl(i) = Sl(i) + sh(i,j)
335     end do
336       Sl(i) = Sl(i)  / 1440.0d0  / 365.0d0
337       write(*,*) Sl(i)
338   end do
339   end subroutine AxisTilt
```