SKSAT

vol.1

著 坂本 優太

表紙解説

T から始まる自動凍結で話題の例のアレ．最近，理系のプレゼンの中によく生息している．

5000 兆円あれば何ができるか．宇宙エレベーター作っても，恒星間ロケット作っても多分お釣りがくる．素晴らしい．

まずは弊学に理科実験棟が欲しいですねあとロケットエンジンの試験場とか．5000 兆円欲しい．

序文

さて，今年もやって参りました，小冊子のお時間です．

地学部の部誌，「ホルスト」における小冊子というものは，「なんか一人で何個か原稿書いちゃったし実質部誌！ w」みたいなテンションで生み出される，同じ人間が書いた原稿の集合体です．

ちなみにこの小冊子，「SKSAT」は，「タキスト」，「SKMT」，「SKMT-2」に続く，（たぶん）4 つ目の小冊子です．まあ，「SKMT-2」は去年僕が部誌の編集作業に追われながら書いたもので，クオリティも厚みも無かったのでノーカンかもしれませんが．

去年はちょっと，いやかなり，なんというかショボかったので，今年はずいぶん早くから原稿を書くことを考えていたのですが，おかしいですね，今この文章を書いているのは 9/7 のことです．しかも完成していない原稿がありますこれはアカン．

書くネタは考えていても直前になって書き始めたり，突然書きたいネタが増えたり，LaTeX の学習に時間がかかったりしたのがダメだった．

まあ，Word~~とかいうちょっとアレなソフトウェア~~で書くよりは圧倒的成長と圧倒的進捗をしたので良かったと思っています．

肝心の中身についてですが，本当は数値計算もっとやりたかったし、JAXA の新しいロケットエンジンのこととか書きたかったんですが，できませんでした．はい．もしかしたら未練から文化祭後に書いて来年の部誌に commit してしまうかもね．今年引退ですけど．

まあこんな感じのテンションで書いているので，温かい目で読んでやってください．

あと，この原稿の LaTeX ソースコードは GitHub に上げてある[1]ので，間違い等あればそちらに Issue なり PR なりしていただけるとありがたいです．

[1] https://github.com/sk2sat/holst-sksat

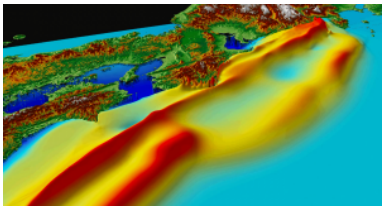
第1章　数値計算の理論

1.1　はじめに

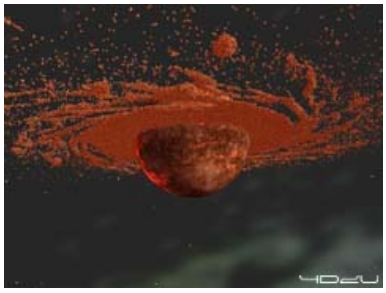
数値計算とは，代数的または解析的に解くことが難しい，もしくは計算量が膨大になってしまうような問題を，コンピューターを用いて計算する手法のこと．数値計算は本稿では、この数値計算という手法の基礎を解説する．

1.2　数値計算とシミュレーション

コンピューターシミュレーション，という言葉を聞いたことがあるだろうか．津波が都市のどこまで被害を及ぼすかや，天体の形成のシミュレーション，というのは聞いたことがあるかもしれない．



(a) 海洋研究開発機構の津波シミュレーション



(b) 国立天文台の月形成シミュレーション

図 1.1: コンピュータシミュレーションの例

このように，実際に実験を行うことが不可能であるような問題を研究するにあたって，コンピューターシミュレーションは現在では必要不可欠な技術となっている．

では，こうした物理現象をコンピューターシミュレーションを使って再現するには，どのようなことが必要なのかというと，具体的には 1.2 のような手順を踏む．

まずは，シミュレーションする物理現象をモデル化する．モデル化とは，現実の問題から必要な部分だけを取り出し，問題を単純化することだ．例えば，手に持っ

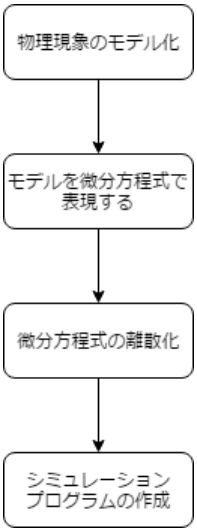


図 1.2: コンピューターシミュレーションの手順

たボールを静かに離して落下させたとしよう．このとき，実際の物理現象としては，ボールは手についた汗で滑ったり，空気抵抗によって減速されたり，回転していたりと，様々なことが起きており，これらは全てボールの運動に影響を及ぼす．しかし，これらによるボールの運動への影響はとても小さいため，「ボールの落下運動」を考えるにあたっては，無視しても問題はない．これも一種のモデル化である．さて，これで問題をモデル化することは出来たので，次にこのモデルを微分方程式で表現しよう．

1.3 微分方程式とは

そもそも，微分方程式とは何かというと，簡単に言えば微分形を含む方程式のことだ．例えば，

$$\frac{dx}{dt} = x \tag{1.1}$$

などがある．

1.3.1 微分の復習

ここで，微分とはなんだったかを軽く復習しよう．

平均変化率

まずは，直線の傾きのことを考えてみよう．ある直線を表す関数 $y = f(x)$ があるとして，この傾きを求めようと思ったら，どのようにすればいいだろうか．

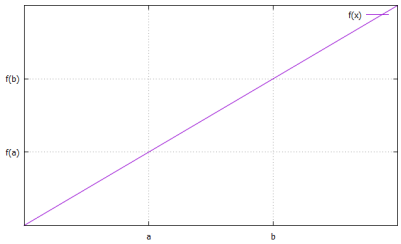


図 1.3: $f(x)$ の平均変化率

答えは簡単で，$f(x)$ の変化量を x の変化量で割ればいい．傾きを α とおくと，

$$\alpha = \frac{f(b) - f(a)}{b - a} \tag{1.2}$$

となる．このような傾き α のことを，$f(x)$ の x が a から b まで変化するときの平均変化率という．

平均変化率の極限

さて，直線の関数の平均変化率はすべての区間において同じなので，今度は曲線の関数の平均変化率について考えてみよう．曲線を表す関数 $f(x)$ があるとして，先程と同じように,x が a から b まで変化するときの平均変化率を考える．曲線の平均変化率を考えるときは，点 $(a, f(a))$ と点 $(b, f(b))$ を結ぶ直線 l を考え，この直線 l について，直線の平均変化率を求めればよい．そのため，平均変化率 α を求める式は

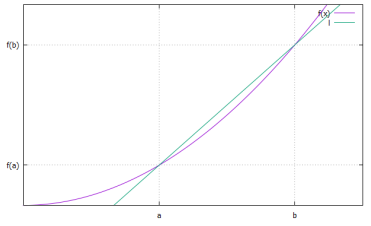


図 1.4: 曲線の平均変化率

1.2 と全く同じである．

これで曲線においても平均変化率を求めることが出来たわけだが，図 1.4 を見れば分かるように，あくまでこれは「平均の変化率」であり，$b-a$ の値が大きければ大きくなるほど，直線 l は曲線とはかけ離れていき，平均変化率は曲線の変化率とはなかなか言いにくくなってくる．

微分係数

そのため，今度は $b-a$ の値をどんどん小さくしてしまえばどうだろうか，というのが微分の第一歩である．$b-a$ の値をどんどん小さくするというのは，「b を限りなく a に近づける」ということに等しい．このように，ある値を別のある値に”限りなく近づける”ということを，「極限」といい，

$$b \to a \tag{1.3}$$

のように書く．さて，今考えたいのは $b \to a$ のときの平均変化率 α であった．極限を表す記号である lim を使って平均変化率を求める式 1.4 を書き直すと，1.4 のようになる．

$$\lim_{b \to a} \frac{f(b)-f(a)}{b-a} \tag{1.4}$$

しかし，このままでは少し分かりにくいので，$b-a=h$ とおいて書き直すと 1.5

$$\alpha = \lim_{h \to 0} \frac{f(a+h)-f(a)}{h} \tag{1.5}$$

のようになる．こうなると、α はもはや「平均の」変化率などではなく、「瞬間の」変化率と言えるだろう．このような、「瞬間の」変化率のことを数学的には「$x=a$ における微分係数」という．

$x=a$ における微分係数は $f'(a)$ と表す．

よって微分係数 $f'(a)$ の定義は 1.6 のようになる．

$$f'(a) = \lim_{h \to 0} \frac{f(a+h)-f(a)}{h} \tag{1.6}$$

導関数

前項で、$f(x)$ の $x = a$ における微分係数 $f'(a)$ を定義したわけだが，この a というのはある特定の x の値であった．これを特定の値 a ではなく，変数 x にあてはめると，$f'(x)$ という新たな関数が導かれる．この「導かれた関数」のことを「導関数」といい，ある関数 $f(x)$ からその導関数 $f'(x)$ を求めることを，「微分する」という．

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} \tag{1.7}$$

ちなみに，これは正確には関数 $f(x)$ を「x について」微分するという操作だが，何の変数で微分するかを明確に示したい場合には，導関数を 1.8 のように表すこともある．[1]

$$\frac{d}{dx} f(x) \tag{1.8}$$

1.4 微分方程式と物理現象

さて，ここまでで，数値計算したい物理現象のモデル化と，微分方程式のことが分かった訳だが，数値計算を行うには，モデルを微分方程式で書き表さなければならない．そこでここでは，物理現象と微分方程式の関係について考えてみよう．先程考えたモデルでは，落下するボールであった．このモデルをもっと簡潔に表すと，

- ボールが重力に引かれて落下している
- 空気抵抗は考えない

というものである．このボールは落下「運動」しているので，ニュートンの運動方程式

$$ma = F \tag{1.9}$$

を立てる．ボールに働く力は重力のみなので，鉛直上向き[2]を正にとると，ボールに働く力は，ボールの質量を m とすると，mg．よって，このボールの運動方程式は

$$ma = -mg \tag{1.10}$$

[1] 1.7 をラグランジュ記法，1.8 をライプニッツ記法という．
[2] 重力が働く方向と反対の方向．ようするに上向き．

となる．両辺を m で割ると，[3]

$$a = -g \tag{1.11}$$

よって，ボールの加速度はボールの質量によらず，重力加速度 g のにマイナスをつけた値であることが分かった．[4]

しかし，これではまだモデルを微分方程式として表せていない．右辺は定数[5]なので，左辺の加速度 a について考えてみよう．

加速度とはなんだったかというと，「速度の変化の割合」のことであった．つまり，

$$\text{平均加速度} = \text{速度の変化量} \quad / \quad \text{時間の変化量} \tag{1.12}$$

である．ここで，1.3.1 で出てきた，平均変化率のことを思い出してみてほしい．平均変化率とは，$y = f(x)$ としたとき，

$$\text{平均変化率} \quad = \quad y \text{ の変化量} \quad / \quad x \text{ の変化量} \tag{1.13}$$

であった．つまり，速度を時間 t に関する関数 $v(t)$ とすると，平均の加速度 $\overline{a}$ は $v(t)$ の平均変化率で表される．

$$\overline{a} = \frac{v(t + \Delta t) - v(t)}{\Delta t} \tag{1.14}$$

ここで，「平均」ではなく，「瞬間」の加速度を考えるために，$\Delta t \to 0$ とすると，

$$a = \lim_{\Delta t \to 0} \frac{v(t + \Delta t) - v(t)}{\Delta t} \tag{1.15}$$

となる．これは $v(t)$ の導関数の定義そのものである．よって，

$$a = \frac{dv}{dt} \tag{1.16}$$

となる．つまり，加速度 $a(t)$ は速度 $v(t)$ の導関数として表される．

そしてさらに，平均の速度の定義は

$$\text{平均速度} \quad = \quad \text{移動距離} \quad / \quad \text{時間の変化量} \tag{1.17}$$

であることから，同様に考えると

$$v = \frac{dx}{dt} \tag{1.18}$$

---

[3] $\because m \neq 0$

[4] これはつまり，空気抵抗を無視すれば，同じ高さから落とした物体は同時に着地する，ということである．

[5] 正確には，重力加速度の値は一定ではなく，落下して高度が下がれば下がるほどに大きくなっていく．しかし，ここでは重力加速度の値は一定 $(9.8m/s^2)$ としている．これもモデル化の一種である．

ということも分かる．

また，式 1.16,1.18 より，

$$a = \frac{d^2x}{dt^2} \tag{1.19}$$

でもある．

よって、式 1.19 を運動方程式 $ma = F$ に代入すると，

$$m\frac{d^2x}{dt^2} = F \tag{1.20}$$

となることから，運動方程式そのものが 2 階の微分方程式であることも分かる．

式 1.11 についても同様に考えると、

$$\frac{d^2x}{dt^2} \tag{1.21}$$

これで，物理現象を微分方程式で表すことが出来た．

1.5　オイラー法

物理現象を微分方程式で表せたところで，数値計算の話に戻ろう．

数値計算の手法には様々なものがあるが，その中でも，最も簡単な手法の一つであるオイラー法は，先ほど復習した微分の定義から簡単に導くことができる．導関数の定義を再掲すると，

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} \tag{1.22}$$

であった．ここで，$h \to 0$ としているが，コンピューターではこのような極限値は扱えない[6]ため，h を「限りなく 0 に近づける」のではなく，「そこそこ」0 に近い値を h とすることで，$f'(x)$ を近似することができる．これは，「平均変化率」と同じ考え方である．

$$f'(x) \approx \frac{f(x+h) - f(x)}{h} \tag{1.23}$$

式**??**より，$f(x)$ が既知であれば，h をある程度小さい値として同式に代入すると，$f'(x)$ に近い値，つまり近似値を得る事ができる．

では実際に，式**??**を使って導関数を近似してみよう．

[6]後述する

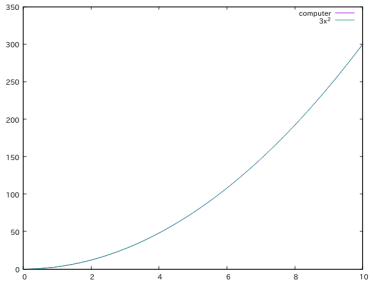


図 1.5: 数値微分の数値解と解析解

$f(x) = x^3$ とすると，導関数 $f'(x)$ の「正しい値」は $f'(x) = 3x^2$ である．このように，解析的に答えが分かっているとき，これを解析解という（これとは逆に，数値計算によって求めた解のことを数値解という）．では，$h = 0.01$ として計算し，数値解と解析解の値を比べてみよう．

図 1.5 より，数値解と解析解はほぼ重なっており，ある程度の精度で数値解が解析解を近似できていることが分かる．このように，数値的に導関数の値を求める，つまり微分を行うことができたわけだが，式 1.23 を変形すると，1.25 のようになる．

$$f(x+h) - f(x) \approx hf'(x) \tag{1.24}$$

$$f(x+h) \approx f(x) + hf'(x) \tag{1.25}$$

式 1.25 より，$f(x)$ が未定義でも，どこかの $f(x)$ の値と導関数 $f'(x)$，h が分かっていれば，$f(x+h)$ の近似値を求めることができる，ということが分かる．

これにより何ができるかというと，$f'(x), h, f(a)$ が既知とすると，

$$f(a+h) \approx f(a) + hf'(a) \tag{1.26}$$

$$f(a+2h) \approx f(a+h) + hf'(a+h) \tag{1.27}$$

$$f(a+3h) \approx f(a+2h) + hf'(a+2h) \tag{1.28}$$

$$\ldots \tag{1.29}$$

というように，h ごとに「飛び飛び」に $f(x)$ の値を次々に近似していくことができる．このように，既知の $f'(x)$ を用いて $f(x)$ の値を求める（近似）ことを，「数値積分」という．

この場合は，$f(a)$ の値が最初に必要なので，この $f(a)$ のことを初期値といい，また，「飛び飛び」の度合いである h のことをステップ幅もしくは刻み幅という．

先程とおなじように，数値積分を実際にやってみよう．

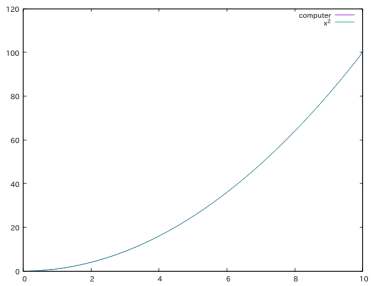


図 1.6: オイラー法による数値解と解析解

$f'(x) = 2x$ とすると，解析解は $f(x) = \int f'(x)dx = x^2 + C$（$C$ は積分定数）となる．ここで，初期値を $f(0) = 0$ とすると，$f(x)$ に代入して，

$$f(0) = 0 + C = 0 \tag{1.30}$$

$$\therefore C = 0 \tag{1.31}$$

より積分定数が定まり，$f(x) = x^2$.

$h = 0.01$ として，これらを数値積分し，数値解と解析解を重ねてみると，図 1.6 のようになる．

ここでも，数値解はある程度の精度で解析解を近似していることが分かる．

このようにして数値積分を行う手法のことを，数学者レオンハルト・オイラー[7]の名を冠して，オイラー法[8]という．

もちろん，数値計算法にはオイラー法以外にも様々な手法がある[9]が，このオイラー法はその中でもとても分かりやすいものである．

1.6 終わりに

本当はもう少し色々書きたかったが，時間が足りなくなってしまった．無念．

[7] 18 世紀の数学の中心となり，続く 19 世紀の厳密化・抽象化時代の礎を築いた．彼の名を冠した定理はたくさんある．

[8] 正確には，これは前進オイラー法というもので，これの他に陰解法である後進オイラー法や，修正オイラー法というものがある．

[9] ニュートン・ラフソン法やルンゲクッタ法などがある．

第2章　シミュレーション天文学

2.1　シミュレーション天文学とは

天文学と聞くと，望遠鏡で星を観測して云々，というイメージしか湧かないかもしれないが，最近の天文学というのは，実際に天体観測を行うもの[1]の他に，天体の運動や構造などの理論を考える理論天文学，そして，今回取り上げるような，天体のシミュレーションをするシミュレーション天文学など，様々な分野がある．

特に，シミュレーション天文学は，惑星の形成，銀河の衝突，ブラックホールの降着円盤，銀河の衝突，宇宙の大規模構造など，実際に観測することが難しい，もしくは観測できていても，それを確実に説明する理論がなかったり，計算量が膨大になったりするもののシミュレーションを行う．

国立天文台の 4D2U のウェブサイトなどを見ると，最新のシミュレーション天文学の成果を見ることができる．

2.2　N 体シミュレーション

さて，このようにさまざまな天文現象を扱うシミュレーション天文学だが，これらのシミュレーションは，実際にはどのように行っているのだろうか．ここでは，シミュレーション天文学の一分野である N 体シミュレーションについて考えてみよう．

N 体シミュレーションというのは，その名の通り，N 個[2]の物体の運動をシミュレーションするもので，多体シミュレーションとも呼ばれる．

天文学で N 体シミュレーションが用いられる分野としては，惑星形成理論などがある．

実は，現在の太陽系がどのように形成されたかということには謎が多く，様々な形成シナリオが提案されているが，いったいどれが正しいのか，ということは中々

[1] 実際に天体観測を行う天文学の中にも，目に見える可視光のほかに，赤外線を使って観測を行う赤外線天文学，X 線を使う X 線天文学，電波を使う電波天文学，最近では，ニュートリノを使ったニュートリノ天文学というものもある．

[2] N は任意の自然数

検証できるものではない.[3]

そのため，微惑星をコンピューター上で様々な条件で衝突，合体させ，現在の太陽系の形成の条件を探る，というのが惑星形成のシミュレーションであり，これはたくさんの微惑星の運動を計算しなければならないので，N 体シミュレーションの一種である.

2.3 2 体問題

さて，N 体シミュレーションでは，N 個の天体の運動を考えるわけだが，急にたくさんの天体の運動を考えるのは大変なので，まずは 2 つの天体の運動について考えてみよう．2 つの天体は，互いに万有引力で引っ張り合っている．万有引力の強さ

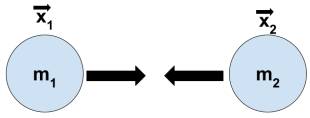


図 2.1: 2 つの天体が，互いに万有引力で引き合っている

は，2 つの天体の質量と距離の逆 2 乗に比例するので，2 つの天体の質量をそれぞれ m_1, m_2, 位置ベクトルを $\vec{x_1}, \vec{x_2}$ とおくと，それぞれの天体に働く万有引力を F_1, F_2 は

$$F_1 = -\frac{Gm_1m_2}{||\vec{x_1} - \vec{x_2}||^2} \frac{\vec{x_1} - \vec{x_2}}{||\vec{x_1} - \vec{x_2}||} \tag{2.1}$$

$$F_2 = -\frac{Gm_1m_2}{||\vec{x_2} - \vec{x_1}||^2} \frac{\vec{x_2} - \vec{x_1}}{||\vec{x_2} - \vec{x_1}||} \tag{2.2}$$

となる.

よって，天体同士の衝突を考えないとすると，天体にはたらく力は万有引力のみ

[3]シミュレーション以外での検証アプローチとして，太陽系形成初期の物質を調べる，というものがある．しかし，地球にある物質は熱変性や風化を受けているため，太陽系形成初期そのままの物質は全く残っていない．そのため，熱変性や風化を受けていない小惑星のサンプルを入手しよう，という試みが，あの「はやぶさ」である.

なので，それぞれ運動方程式を立てると，

$$m_1 \frac{d^2\vec{x_1}}{dt^2} = -\frac{Gm_1m_2}{||\vec{x_1}-\vec{x_2}||^2}\frac{\vec{x_1}-\vec{x_2}}{||\vec{x_1}-\vec{x_2}||} \tag{2.3}$$

$$m_2 \frac{d^2\vec{x_2}}{dt^2} = -\frac{Gm_1m_2}{||\vec{x_2}-\vec{x_1}||^2}\frac{\vec{x_2}-\vec{x_1}}{||\vec{x_2}-\vec{x_1}||} \tag{2.4}$$

となる．両辺を m_1, m_2 でわって，

$$\frac{d^2\vec{x_1}}{dt^2} = -\frac{Gm_2}{||\vec{x_1}-\vec{x_2}||^2}\frac{\vec{x_1}-\vec{x_2}}{||\vec{x_1}-\vec{x_2}||} \tag{2.5}$$

$$\frac{d^2\vec{x_2}}{dt^2} = -\frac{Gm_1}{||\vec{x_2}-\vec{x_1}||^2}\frac{\vec{x_2}-\vec{x_1}}{||\vec{x_2}-\vec{x_1}||} \tag{2.6}$$

$\vec{r} = \vec{x_1} - \vec{x_2}$,$r = ||\vec{x_1} - \vec{x_2}||$ とおくと，この 2 式の差は

$$\frac{d^2}{dt^2}(\vec{x_1}-\vec{x_2}) = -G(m_1+M_2)\frac{\vec{r}}{r^3} \tag{2.7}$$

となる．これは 2 つの天体の相対運動を考えることに等しい．詳細は省略するが，この式を積分し，r についての微分方程式を解くと，これが楕円軌道を描くことが分かる．[4]

2.4 N 体問題の数値的解法

ということで，$N = 2$ のとき，つまり，「2 体問題」は解析的に解くことができるのだが，$N > 2$ のときはそうもいかなくなってくる．$N = 3$ のとき，つまり「3 体問題」のときについては，一部の条件を満たす有名な問題[5]の時に限っては解くことができるが，基本的には $N > 2$ のときは解析的に解くことは不可能である．

そこで，数値計算が用いられるのだ．多体系での重力で相互作用する粒子の運動方程式は，2 体問題の時と似ていて，

$$\frac{d^2\vec{x_i}}{dt^2} = -\sum_{j\neq i} Gm_j \frac{\vec{x_j}-\vec{x_i}}{||\vec{x_j}-\vec{x_i}||^3} \tag{2.8}$$

というようになる．ようするに，自分以外の他すべての天体との重力相互作用の和を考えるのだ．N 体シミュレーションでは，式 2.8 を直接数値積分することにより，多体系の時間発展を計算する．

[4] 僕はまだ面積速度一定の法則を導くところまでしか理解できていない... 今年中には理解したいところ．

[5] 1 つだけ質量がとても小さい，など

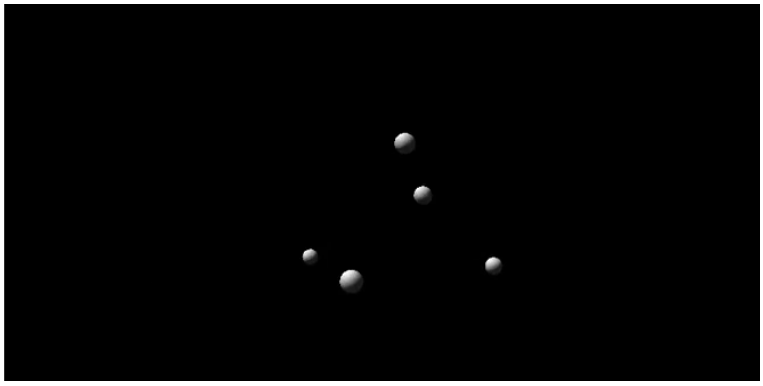


図 2.2: 実際に N 体シミュレーションをやってみた結果

数値計算は他の章を参考にしてもらうとして, これを実際にやり, POV-Ray でレンダリングしてみると, 図**??**のような結果が得られた. しかし, このシミュレーションでは数値積分にオイラー法を用いており, 精度が低く, 刻み幅制御もできないので, すぐに計算が破綻してしまう. そのため, 今後はそれらの点を改善していこうと思っている. このシミュレーションのソースコードは `https://github.com/asanotigaku/planet` に置いてあるので, よかったら見てみてほしい.

第3章　カッシーニ グランドフィナーレ

3.1　カッシーニとは

カッシーニとは，NASA [1]と ESA [2]が開発した土星探査機のこと．1997 年にタイタン IV ロケット[3]で打ち上げられた．名称は天文学者ジョバンニ・カッシーニに由来する．

3.2　カッシーニのミッション

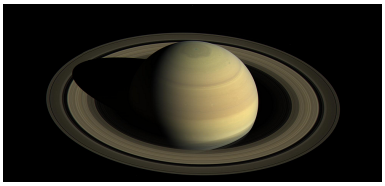
カッシーニの主なミッションは，土星の大気や、謎に包まれていた輪の構造、衛星の探査である．特に，衛星タイタンについては ESA が開発した小型探査機ホイヘンスを降下させ，その大気や地形の詳細な探査を行った．現在，最終ミッションである「グランドフィナーレ」を実行中だ．

3.3　土星観測の歴史

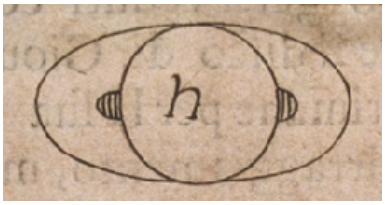
カッシーニの探査対象である土星の大きな特徴は，その大きな輪だ．小型の望遠鏡でも簡単に観測することが出来るので、各地の天文台の観望会などでも人気の天体となっている．

この特徴的な輪を初めて観測したのはガリレオ・ガリレイで，初めて望遠鏡を空に向けた翌年，1610 年のことだった．しかし，望遠鏡の精度が低かったため，輪とは判別できず，「耳のようなものがある」と表現した．また，1612 年には輪が地球の正面を向いて輪が見えなくなった[4]ため，ガリレオは困惑し，土星の英語名 saturn の

[1]アメリカ航空宇宙局
[2]欧州宇宙機関
[3]アメリカ空軍がスペースシャトル・チャレンジャー事故後に大型衛星打ち上げ用に開発したロケット．1989 年から 2005 年まで使用された．
[4]土星は回転軸を傾けたまま太陽の周りを公転しているため，地球から見たときに，環がはっきりと見える時期と環がほとんど見えなくなる時期がある．



(a) 大きな輪が特徴的な土星



(b) ガリレオによる土星のスケッチ

由来であるサートゥルヌスの神話のエピソード[5]になぞらえ，「土星は子供達を飲み込んだのか?」と言ったという．[6]

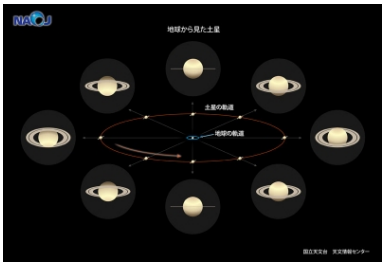


図 3.2: 土星の環の消失現象の仕組み

その後，1655 年にオランダの天文学者ホイヘンスがこの「耳のようなもの」がディスク状のものであると発表．さらにホイヘンスは土星本体と輪の間に隙間を発見した．[7]

さらにその後，1787 年にはピエール＝シモン・ラプラスが土星の環が小さな環の集まりであると提唱，1859 年にはジェームズ・クラーク・マクスウェル[8]が，土星の環が小天体の集まりで出来ていると提唱した．

その後，土星探査が大きく進展したのは，1979 年 9 月にパイオニア 11 号[9]が土星に最接近したときのことだった．パイオニア 10 号は土星に 21000km の距離まで接近し，E,F,G 環を発見したほか，衛星タイタンの気温が 250K 程度と測定．1980 年 11 月にはボイジャー 1 号が土星に到着，パイオニア 11 号では得られなかった高画質

[5]サートゥルヌスはローマ神話の農耕神で，ギリシャ神話のクロノスと同一視される．シンボルは鎌．サートゥルヌスには，「自分の子供に殺される」という予言を恐れて，5 人の次々に飲み込んでいったという伝承がある．

[6]翌年にはまた土星の環が見えるようになったため，ガリレオはさらに困惑した．

[7]この功績から，土星の輪の隙間の一つは「ホイヘンスの空隙」(間隙，という場合もある)」という名前がつけられている．

[8]「マクスウェル方程式」の”あの”マクスウェルである．

[9]NASA の惑星探査計画，パイオニア計画の一環で，パイオニア 10 号の姉妹機．初の土星探査機である．

の画像を得られた．また，ジャー 1 号はタイタンに接近，タイタンの分厚い大気[10]を発見した．翌年 1981 年 8 月には，ボイジャー 2 号が土星に到着．新たな衛星や空隙を発見した．

パイオニアやボイジャーにより，それまでよく分かっていなかった土星の姿が少しづつ分かってきたが，これらの探査機はあくまで一時的に土星に「立ち寄った」だけで，継続的な土星の探査というのは行っていなかった．

人類が継続的に土星を探査する機会を得るには，今回取り上げるカッシーニが土星に到着する 2004 年まで待たねばならない．

3.4　探査機カッシーニによる土星探査

探査機カッシーニは，1997 年に打ち上げられた後，金星で 2 回，地球と木星で 1 回ずつスイングバイを繰り返して，2004 年に土星軌道に到達した．翌年 2005 年には小型探査機ホイヘンスを衛星タイタンに降下させて詳細な探査を行った他，土星の輪や大気の詳細な構造，衛星の探査を行った．もともと探査は 4 年間の予定であったが，探査機の状態が良く，また，タイタンを利用したクローズドスイングバイにより燃料が節約できていたため，探査計画は延長された．そして，土星軌道投入 10 周年の 2014 年に最終ミッション，「グランドフィナーレ」が発表された．

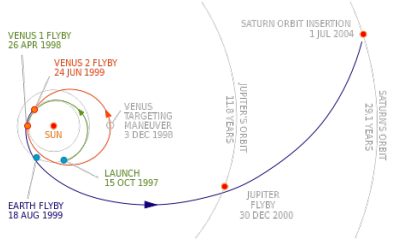


図 3.3: カッシーニの土星到達までの道のり

10 年以上にわたるカッシーニの探査は，土星やその輪，衛星に関する様々な成果を生んだ．

3.5　グランドフィナーレ

グランドフィナーレというのは，2014 年に発表された，カッシーニの最後のミッションである．具体的には，土星とその環の「すきま」を 22 回にわたり周回し，近

[10]分厚い大気の層をもつ衛星はタイタンのみである．

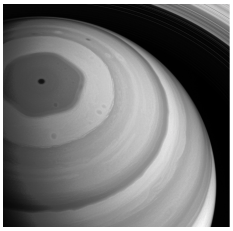


図 3.4: カッシーニにより初めて観測された，六角形の北極域の模様．

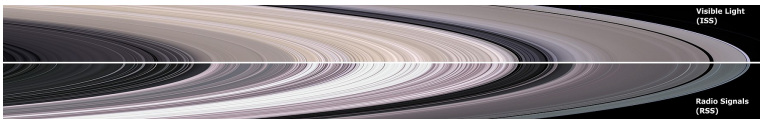


図 3.5: カッシーニによる 4 度斜めから見た C 環、B 環及び A 環の画像。上図は、2004 年 12 月 12 日に撮影された無着色の画像．下図は 2005 年 5 月 3 日に撮影され、粒子の径に応じて彩色した画像．

傍から土星の大気を調査，最後の周回を終えた後は，大気のサンプリングと地球へのデータ送信をしつつ，土星大気に突入し，「最期」を迎える，というものだ．22 回目の最後の周回は 9 月 9 日，文化祭 1 日目であり，「最期」は 15 日を予定している．そのため，文化祭期間中には，展示の中で最新のカッシーニの情報も伝えていこうと思う．

図 3.6: カッシーニはその役目を終えると，土星大気に突入して、燃え尽きる．

第4章　国産民間ロケットの挑戦

4.1　はじめに

MOMO，という名前のロケットがある．「ホリエモンロケット」というと，聞いたことがあるかもしれない．つい最近，今年の 7 月 30 日に初号機が打ち上げられたばかりのこのロケットは，北海道のベンチャー企業，インターステラ・テクノロジズが開発したロケットだ．残念ながら初号機は宇宙空間に到達することは叶わなかったが，それでも離床には成功し，データの採取もできた．そして，これは国内の民間企業では初の快挙である．ここでは，この MOMO というロケットを開発したインターステラ・テクノロジズの歴史について少し調べたので書いてみようと思う．

図 4.1: MOMO ロケットのイメージ図

4.2　はじまりは風呂場から

1997 年，H-Ⅱロケットを使用して，日本独自の有人宇宙船を打ち上げることはできないか，という検討会が，全国の宇宙好きの集まりで行われた．その後，H-Ⅱではなく，既成のエンジンを使用した新型ロケットを作り，それを使って同様のことをやる計画が立ち上がった．宇宙に強い関心を持つ堀江貴文氏に出資をもちかけ，ロシアのロケットエンジンの購入を試みるも，全くアドバンテージがない不利な交渉となり，あまりの高額さから購入を断念する．

2005 年，自分たちでロケットエンジンを作り，民間独自の低価格な小型ロケットを作ってしまおうという話になり，「なつのロケット団」[1]を結成．メンバーの住むアパートの風呂場で，最初期のロケットエンジンの試験を開始する．[2]

4.3　繰り返す燃焼試験

2006 年，堀江氏当時社長を務めていた SNS 株式会社の一事業としてロケットエンジンの開発に着手．2008 年には 30kgf 級の最初のロケットエンジンの燃焼試験に成功，翌 2009 年に開発拠点を北海道に移し，90kgf 級のエンジンの開発を開始．さらに 2010 年には 100kgf 級エンジンの開発に成功．2011 年には，最初のデモンストレーション打ち上げ機である「はるいちばん」他 3 機のロケットの打上試験に成功した．

2013 年には，北海道大樹町にインターステラテクノロジズ株式会社を設立し，現在の体制となった．インターステラテクノロジズ社は，大樹町にエンジン燃焼施設を整備し，本格的にエンジンの開発を進めた．そして，純民間商業ロケット「ポッキー」，姿勢制御飛行実験機「LEAP」の打ち上げに成功．さらに，高度 100km に到達する観測ロケット「MOMO」の開発を開始した．

図 4.2: インターステラテクノロジズ社が開発したロケット

4.4　インターステラのエンジン

インターステラのロケットエンジンは，エタノールと液体窒素を燃料とした液体燃料方式だ．これらの燃料は調達が容易であるため，頻繁な燃焼試験や，商用化した時の打ち上げコスト低下に大きく役立っている．

エンジンの構造は比較的単純で，燃料のタンクとは別に加圧用のヘリウムタンクを搭載して，ヘリウムガスの圧力で燃料を押し出し，インジェクターから混合・噴射，その後燃焼室で燃焼させて高圧の燃焼ガスを発生させ，グラファイト製のスロートで加速して噴き出す，というものだ．

[1]「なつのロケット」はメンバーの一人である「あさり　よしとお」氏の漫画のタイトル．
[2]これが，「始まりは風呂場から」と言われる所以である．

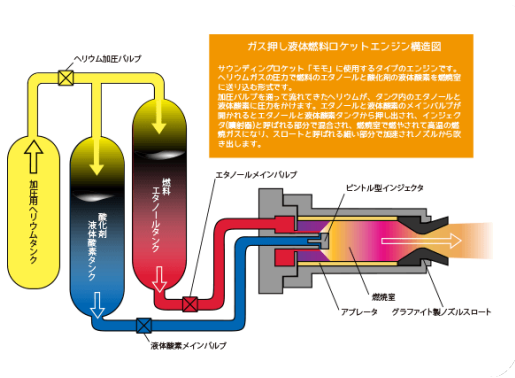


図 4.3: ロケットエンジンの仕組み

4.5 サウンディングロケット MOMO

このように，独自のロケットエンジンの開発を主軸として民間でロケットを作り，打ち上げてきたインターステラ社だが，同社がつい最近，7 月 30 日に打ち上げ実験を行ったのが，サウンディングロケット，「MOMO」だ.

サウンディングロケットというのは, 弾道飛行を行う観測ロケットのことで, MOMO では 20kg までの実験・観測機器をペイロードとして搭載し，高度 100km を目指す. ペイロードはパラシュートで回収出来るため，気球などでは到達が難しい高高度の観測や，微小重力状態での実験を行うことができる.[3]

[3]MOMO では 260 秒間の微小重力状態を作り出すことが可能.

7 月 30 日に打ち上げられたのは，この MOMO の初号機である．

初号機の打ち上げはもともと，29 日に行う予定であったが，濃霧のため「視程 600m 以上であること」という打ち上げ条件を満たせず，30 日に延期した．しかし，30 日の早朝にバルブのトラブルのため打ち上げ 2 分前に一時中止，次のウィンドウである 12:20 の打ち上げを目指した．その後も打ち上げの延期があったものの，どうにか最終ウィンドウとなる 16:32 に打ち上げを行った．

何度も待った上での打ち上げに，濃霧でロケットが全く見えなかったものの，見学者たちはロケットエンジンの音が遠ざかっていくのを聞き，喜んだ．

だが，その後テレメトリ消失のため，離床 80 秒後に地上からの指令で制御落下を行ったとの情報が入った．

30 日夜に行われた会見で，打ち上げの詳細な状況が発表された．会見の内容によると，実際には緊急停止を行ったのは離床 66 秒後であり，その原因はテレメトリが突然消失したからだった．テレメトリが消失したのは高度約 10km のことで，これは最大動圧点，「max Q」に相当する．max Q はロケット関係者の間で「機体を壊してしまうことがある大気圏内の大きなハードル」として知られており，max Q を乗り越えられるかどうか，というのは宇宙ロケットにおいて非常に重要なポイントである．会見では，空力加熱を発生させる機体の出っ張りなどに課題があるかもしれない，と機体破損の原因について言及した一方，年内には後継機を打ち上げたい，という強気な姿勢も見られた．

この「失敗」は一部のメディアで批判的に報道された．確かに，「宇宙空間への到達」という最終目標は達成できなかったが，それでも，何回にも及ぶ打ち上げ延期の対応や，テレメトリが取得できたところまでのデータにより得られた知見は IST にとってかけがえのないものになったのだと思う．宇宙開発においては，サクセスクライテリアのうちフルサクセスが失敗しても，そこであきらめるのではなく，成功した部分から得られた知見を次に生かし，失敗した部分の原因究明と対策が何よりも重要だということは，宇宙開発の歴史が証明している．インターステラの皆さんには，ぜひとも今回の「失敗」を生かし，次につなげて欲しいと思った夏休みだった．

第5章　BoCCHAN-1 OBC を触ってみた話

ここでは，BoCCHAN-1 という OBC [1]を使うにあたって色々やったので，それについて書こうと思います．

5.1　BoCCHAN-1 is 何

そもそも BoCCHAN-1 とは何かというと，開発元の木村研究室のサイト[2] を見てみると，

> 木村研究室が開発した技術を使った小型高性能な衛星用計算機ボードで，6cm 四方の大きさでありながら従来の超小型衛星の OBC とは一線を画す性能を発揮します．

とあります．

大学発の小型人工衛星，「ほどよし」の主計算機として使われていたりする，けっこうすごいやつです．

え？なんでそんな特殊なコンピュータ使ってんのかって？実は今年，東京理科大学がやっている宇宙教育プログラム[3] というものに参加していまして，そこでやる CanSat 実験用に使っています（贅沢すぎでは）．

5.2　機能とか

なんとこいつ Linux が動きます．第 1 回 CHD [4] のときこれを聞いて、静かに興奮しておりました．アーキテクチャは SH4 [5] ．組み込みってかんじですね．

[1] On Board Computer の略．
[2] `http://www.kimura-lab.net`
[3] `https://www.tus.ac.jp/uc`
[4] CanSat Hands-on Discussion
[5] https://ja.wikipedia.org/wiki/SuperH SH4 ってなんか聞いたことがあるようなって思ってたけど、OSECPU の hh4 と名前が似てる．関係ないけど.)

インターフェースについては，I2C とか UART とか GPIO とかイケるみたいで中々良い．

Xbee で通信もできるし．

5.3 チュートリアルやってみた

宇宙教育プログラム LETUS [6]にアップロードされたクイックスタートガイドやユーザーズマニュアルとかを見ながらチュートリアルとかをやってみました．

そこに書いてあった開発手順を簡単に紹介すると，

- 開発環境一式が入っている仮想マシンイメージをダウンロード
- 仮想マシン内の”ほどよし SDK”が導入された Eclipse を起動して，”Executable with HODOYOSHI SDK”プロジェクトを選択して作成
- C/C++でコーディングしてビルド
- Eclipse の中にいる”BoCCHAN-1 ttyConsole”から BoCCHAN-1 にシリアル接続
- バイナリを”BoCCHAN-1 ttyConsole”に D&D して転送
- ttyConsole から実行してみる

みたいなかんじでした．

5.4 開発環境の自作

というわけで，チュートリアルは出来たのですが，ここでいくつか問題が発生しました．問題とは何かというと，

- 僕のマシンが非力すぎて長時間仮想マシン (しかも Ubuntu) で作業とかちょっとツライさん
- Eclipse というか IDE ﾁｮｯﾄ..vim [7]でやりたい

[6]理科大の教育用のなにかの宇宙教育 PG 用鯖．掲示板とかファイルアップロード出来るのはいいなと思うけど通知とか遅らせられないんですかね．頻繁に見に行くというのは地味にめんどい．調べてみたら moodle っていう OSS を使ってるっぽくて，Ruby で書かれた API ラッパーがあったから試してみようとしたのだけれど，管理者権限が無いといけないみたいで諦めた．仕方ないから BeautifulSoup からログインするスクリプト書いたけど，めんどくさくなってそれで終わってる．新規投稿が来たら Slack に流すみたいなのが出来れば便利なんだけど．

[7]高機能なテキストエディタ．えまっくす...?なんですかそれは（戦争）

はい．開発環境自作のお時間です．

まずはEclipseどうなってんのかなと思ったので，適当な”Executable with HODOYOSHI SDK”プロジェクトを作って，Makefile を見てみました．IDE が生成した Makefile とかクソ長いんじゃないの・・・とか思ったものの、かなり短かったのでやってることはすぐわかりました (Release だけだったし)．主要な部分を抜き出すと、リスト 5.1,5.2 のようになっていました．

リスト 5.1: subdir.mk(コンパイル部分)

```
1 %o:../%.cpp
2     @echo 'Building file: $<'
3     @echo 'Invoking: SH4-Linux G++ Compiler'
4     sh4-linux-g++ -I/home/shdevelop/hodosdk/include/ -O0 -Wall -c -
          fmessage-length=0 -MMD -MP -MF"$(@:%.o=%.d)" -MT"$(@:%.o=%.d)"
           -o "$@" "$<"
5     @echo 'Finished Building: $<'
```

リスト 5.2: makefile(リンク部分)

```
1 hoge: $(OBJS) $(USER_OBJS)
2     @echo 'Building target: $@'
3     @echo 'Invoking: SH4-Linux G++ Linker'
4     sh4-linux-g++ -L/home/shdevelop/hodosdk/lib -o "hoge" $(OBJS) $(
          USER_OBJS) $(LIBS)
5     @echo 'Finished building target: $@'
```

ちなみに$(LIBS) は、objects.mk にて、リスト 5.3 のように定義されていました．

リスト 5.3: $(LIBS) の定義

```
1 LIBS := -lpthread -lm -lhodoyoshi -lrt
```

ようするに，SuperH 向けの g++でコンパイルして，ほどよし SDK 内のライブラリをリンクしてるだけです．なんだこれならホスト環境でも全然出来そうじゃん．

なんか pacman [8]でそれっぽいのが見つからなかったのと，ほどよし SDK とかが結構古いバージョンのもので構成されてたので，Ubuntu 環境に移動して SH4 向けの gcc,g++を探してみると，Debian wiki に書いてありました [9]．apt-get でリスト 5.4 のようにしてインストールします．

```
1 # apt-get install gcc-sh4-linux-gnu g++-sh4-linux-gnu
```

[8]Arch Linux のパッケージマネージャ．Arch はいいぞ．

[9]https://wiki.debian.org/SH4

よーし，これでいけるでしょう，ということで，

リスト 5.4: sh4-linux-gnu でコンパイル，qemu でテスト実行

```
1 $ vim hello.c
2 $ sh4-linux-gnu-gcc hello.c -o hello
3 $ file hello
4 hello: ELF 32-bit LSB executable, Renesas SH, version 1 (SYSV),
      dynamically linked, interpreter /lib/ld-linux.so.2, BuildID[sha1
      ]=..., for GNU/Linux 3.2.0, not stripped
5 $ qemu-sh4 -L /usr/sh4-linux-gnu/ ./hello
6 Hello, World!
```

うん，いいかんじ．じゃあこれを BoCCHAN-1 に転送して・・・あれ？転送ってどうやってやってるんだ？シリアル接続してコンソールに D&D ってあれなにやってんの？

クイックスタートガイドだけではよく分からなかったので，ユーザーズマニュアルを見てみると，Eclipse の ttyConsole の代わりに Tera Term [10] を使ったファイルのアップロード・ダウンロード方法が書いてありました．読んでみると，Tera Term というより，それに付属している ZMODEM とかいうものでバイナリをやり取りしているみたいです．

ZMODEM ってなんぞ？ということで，調べてみます．`https://ja.wikipedia.org/wiki/ZMODEM` によると，YMODEM というバイナリ転送プロトコルを改善したものらしい．1986 年... 生まれてねえ...

じゃあ YMODEM ってなんなの？これも Wiki を見てみます．`https://ja.wikipedia.org/wiki/YMODEM`

ZMODEM よりは説明が詳しい。これは 1985 年． 1 年で名前変えたのか...

そして，これはこれで XMODEM とかいうのを発展させたものらしいので，それも見てみる．`https://ja.wikipedia.org/wiki/XMODEM`

パソコン通信で広く使われてたそうな（こいつは何年なんだ...）．

さて、wiki だとあんまり使い方とかが分からなかったのでまた色々と調べてみると，シリアル接続や telnet 接続，SSH などでリモートにログインしている時に，受信側で rz、送信側で sz を実行すると (速度は遅いが) バイナリが双方向に転送できるらしい．なるほど．

でも Linux で Tera Term とかは使えないし，出来れば端末内で済ませたいなあと思ったら，screen コマンドなるものを見つけたのでこれを使ってみました．BoCCHAN-1 を PC につなぐと/dev/ttyUSB0 が生えるので，screen コマンドを使って，ボードレート 115200 で接続します．

[10] Windows 用のターミナルエミュレータ (そういえば作者の方が FF 内... ひょええ)

リスト 5.5: screen コマンドで/dev/ttyUSB0 にボードレート 115200 で接続

```
$ screen /dev/ttyUSB0 115200
```

BoCCHAN-1 は電源を入れると自動的に Linux が起動して，ログインまで済ましてくれているので，あとは ZMODEM を使ってバイナリを転送すれば良い．

具体的には，BoCCHAN-1 側で rz を実行したら Ctrl+A+":exec sz hello" とかやるとファイルが転送できる．

ようやくバイナリが転送出来たので，張り切って BoCCHAN-1 上で実行してみる．

```
$ ./hello
sh: ./hello: not found
```

... は？

```
$ ls hello
hello
```

は？？？

なんでえええええええええ（泣）

これ, 1 週間ぐらい理由が分かりませんでした．

最初は転送がうまく行かなかったのかとか色々考えて, 配布された仮想環境でビルドしてできたバイナリを同じ方法で転送してみたりしたのですが, これはうまくいくんですよねえ...

原因は, うまく行く方とうまく行かない方のバイナリを file コマンドにかけてみたら分かりました．CTF [11]かよ．

file コマンドの実行結果をうまく行く方とうまく行かない方で比べてみたのがリスト 5.6,5.7.

リスト 5.6: うまく行く方

```
$ file hello
hello: ELF 32-bit LSB executable, Renesas SH, version 1 (SYSV),
    dynamically linked, interpreter /lib/ld-uClibc.so.0, not stripped
```

リスト 5.7: うまく行かない方

```
$ file hello
hello: ELF 32-bit LSB executable, Renesas SH, version 1 (SYSV),
    dynamically linked, interpreter /lib/ld-linux.so.2, BuildID[sha1
    ]=..., for GNU/Linux 3.2.0, not stripped
```

[11]Capture The Flag の略語．もともと互いに相手陣地の旗を奪い合う騎馬戦や棒倒しに似た野外ゲームのことで，そこから派生して，ファーストパーソン・シューティングゲームなどの e スポーツや，コンピュータセキュリティなどの分野でも用いられる．ここではコンピュータセキュリティの分野の CTF のこと．筆者は一応 Harekaze というチームに所属しているが，中々貢献できていないので頑張りたい．

どうやら、インタプリタとやらが違うらしい．このインタプリタってなんぞ？調べてみたら，ELF [12]の共用ライブラリを mmap に配置するあいつでした．確かにそれ違ったら動かないよね．じゃあうまく行く方の ELF インタプリタ，/lib/ld-uClibc.so.0 ってなんなんだ... なんか libc [13]っぽい名前だけど...

これも調べてみたら,uClibc という組み込み向けの libc 用のインタプリタ, みたいなかんじのものでした．「libc って glibc でしょ？」みたいな先入観が抜け切れていませんでした．反省．

ということで，どうやら uClibc を使わなきゃいけないみたいですが，使用する libc を変更するとかやろうとすると，gcc の設定ファイル弄ったり色々しないといけないらしく，どうも面倒です．

でも，そういえば仮想環境の方では普通に sh4 用の g++を使っているだけでした．つまり何らかの方法があるはずです．

ここで、ユーザーズマニュアルを眺めていたら，buildroot という文字列を見つけました．BoCCHAN-1 で動いている Linux はこいつを使ってビルドしたらしいですね．ということで buildroot について調べてみると，公式サイト[14] があったので見てみると，

> Buildroot is a simple, efficient and easy-to-use tool to generate embedded Linux systems through cross-compilation.

とのこと．

組み込み Linux 向けのクロスコンパイル用のツールを簡単に作る・・・？どういうことだ・・・？

調べてみると，Buildroot を使って別のアーキテクチャ用のクロスコンパイル環境を作るという記事[15] があったので見てみました．この記事ではターゲットアーキテクチャは MIPS でしたが，この Buildroot というやつはマルチアーキテクチャの組み込み Linux 向けのクロスコンパイラやリンカ，さらに Linux イメージまで作ってくれるもの，ということが分かりました．

```
1 $ make menuconfig
```

で TUI の分かりやすい画面で細かいところまで設定出来るのが良いですね．ちょっと crosstool-ng っぽいような気もします．

TOOLCHAIN の”C library”を見てみると，デフォルトで uClibc が選択されていました．これはいけそうだ。

[12]Executable and Linkable Format の略．Linux における exe ファイルみたいなものと思ってくれて構わない（かなり違うけど）．ようするにバイナリファイル．

[13]標準 C ライブラリ．C 言語でプログラミングをするときによく使う関数を集めたもの．

[14]`https://buildroot.org`

[15]`http://www.ne.jp/asahi/it/life/it/embedded/buildroot/buildroot_mips.html`

本当は仮想環境の buildroot と gcc を同じバージョンにしたり，色々と設定を合わせたりした方がいいとは思うのですが，面倒だったのでとりあえず TARGET Architecture を SuperH，Target Architecture Variant を sh4(SH4 Little Endian) に設定しました．

ついでに，C++を使いたいので”GCC Options”の”Enable C++ support”にもチェックを入れておきます[16]．

これで，”.config”というファイルに設定が保存されるので，make して数時間ほど待つと，buildroot/output/host/usr/bin 下にクロスコンパイラやリンカが大量に生えます．

```
$ ls output/host/usr/bin
...
sh4-buildroot-linux-uclibc-ar@
sh4-buildroot-linux-uclibc-as@
sh4-buildroot-linux-uclibc-c++@
sh4-buildroot-linux-uclibc-cc@
sh4-buildroot-linux-uclibc-g++@
sh4-buildroot-linux-uclibc-gcc@
sh4-buildroot-linux-uclibc-gcc-5.4.0@
...
toolchain-wrapper*
```

このコンパイラでプログラムをビルドして，ZMODEM で BoCCHAN-1 に転送すると...

```
$ ./hello
Hello, BoCCHAN-1!
```

ｷﾀｱｱｱｱｱ!!!

いやー，時間かかりました．宇宙教育プログラム受講生でここまでしたの僕ぐらいでしょ（謎のイキリ）（無駄な努力）（素直に配布された環境使え）

今回作った開発環境もどきは GitHub の以下のリポジトリに置いときました。

`https://github.com/sk2sat/BoCCHAN-1`

なんか間違ってるとことかあったら Pull Request なり Issue なり頂けると嬉しいです（BoCCHAN-1 ユーザーそんなにいないだろ

5.5 余談

ほどよし SDK のサイト[17] 落ちてる（数年前は割と活動してたっぽい）し，色々と BoCCHAN-1 とかほどよし SDK の情報少なすぎでは...

[16] 最初気づかなくてビルドし直した

[17] `http://www.hodoyoshi.org`

もっと情報がオープンになったらいいなーと思うので，CanSat 合宿の時にでも木村教授に話してみたいです．

あと，この原稿は僕のブログ記事 (http://sksat.hatenablog.com/entry/2017/08/07/203615) を元にしました．無断転載とかではないのであしからず．また，今回部誌に載せるにあたって Markdown から LaTeX に移行したり，加筆修正を行ったりしました．

参考文献

[1] 越塚誠一, 柴田和也, 室谷浩平 粒子法入門 流体シミュレーションの基礎から並列計算と可視化まで　丸善出版 (2014)

[2] 富阪幸治, 花輪知幸, 牧野淳一郎 編 シリーズ現代の天文学 第 14 巻 シミュレーション天文学 日本評論社 (2007)

[3] 飽本一裕 今日から使える 微分方程式 講談社サイエンティフィック (2010)

[4] https://www.esrij.com/industries/case-studies/55425/

[5] http://4d2u.nao.ac.jp

[6] http://www.ele.kanagawa-it.ac.jp/~takeo/sk/NumericalComputation.pdf

[7] https://hackmd.io/EwRgpmCsDsoLQA4CcA2AZnALBBiELDmgQCMBDABkzQoQrMiA

[8] https://ja.wikipedia.org/wiki/%E3%82%AB%E3%83%83%E3%82%B7%E3%83%BC%E3%83%8B_(%E6%8E%A2%E6%9F%BB%E6%A9%9F)

[9] https://www.nasa.gov/jpl/cassini/nasa-and-esa-celebrate-10-years-since-titan-

[10] https://www.nasa.gov/feature/jpl/new-movie-shows-cassinis-first-dive-over-satu

[11] http://www.astronomy2009.jp/ja/webproject/life-g/08.html

[12] http://www.geocities.jp/planetnekonta2/hanasi/saturn/saturn.html

[13] https://www.astroarts.co.jp/special/2017saturn/index-j.shtml

[14] http://www.bbc.com/japanese/40920970

[15] http://solarviews.com/eng/saturnbg.htm

[16] http://planetary.jp/topics/JPL/2017-6920-jpl.html

[17] http://www.istellartech.com

[18] http://fanfun.jaxa.jp/feature/detail/1104.html

[19] http://www.itmedia.co.jp/news/articles/1003/10/news071.html

[20] http://b-gata-diary.com/archives/2163

あとがき

僕がてふてふ[18]してるところを顧問がみて，「全部式番号ついててウザくね」，みたいなことを言われました．確かにそうですね，はい．てふてふ初心者[19]だからゆるちて...

この小冊子は GitHub リポジトリ[20] に置いてあるので，もしかしたら加筆修正とかするかもしれません．スターとかつけてくれると点に登って昇天します．間違いとかあったら PR とかいただけると嬉しいです．

[18] LaTeX.
[19] 夏休みぐらいから始めました．
[20] `https://github.com/sk2sat/holst-sksat`

追記　速報

9/6 の太陽フレアについて

日本時間 2017 年 9 月 6 日（水），太陽中央部分の黒点で、2 回の太陽フレアが観測されました．このうち，日本時間 20 時 53 分に発生した現象の最大 X 線強度は通常の 1000 倍以上，X9.3 に及ぶ大型のもので，これに伴い、高温のコロナガスが地球方向に噴出したこと及び高エネルギーのプロトン粒子の増加が確認されました．この現象による，GPS 衛星を含む人工衛星の障害や、電離層の乱れによる短波通信障害，送電線への影響が予想されます．地球方向に噴出されたコロナ質量放出は 8 日 15 時から翌 9 日 0 時にかけて地球に到達する見通しです．

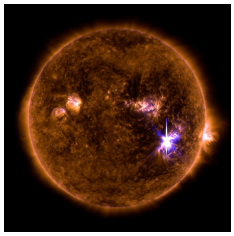


図 5.1: 6 日に発生した太陽フレア

速報:9/7 の太陽フレアについて

9/7 14:41(UTC) に，また X クラスの太陽フレアが発生しました．