数理解析研究所講究録
第1865巻 2013年 18-26

プログラミングを活用した課題解決型教育の実践例

東京都立産業技術高等専門学校・ものづくり工学科　齋藤　純一　(Jun-ichi Saito)
菊地　大輔　(Daisuke Kikuchi)
大田黒　紘之 (Hiroyuki Otaguro)
Department of Monodukuri Engineering,
Tokyo Metropolitan College of Industrial Technology

1　はじめに

本論文では，平成23年度に東京都立産業技術高等専門学校の2名の学生，菊地大輔と大田黒紘之に対して行った「課題解決型学習」の成果について報告する．ここでいう「課題解決型学習」とは，学生が自ら課題を設定し，仮説を立て試行錯誤を行い，解決方法を探りながら一定の結果を得る学習である．ただし，試行錯誤の部分でプログラミングを活用したことを強調しておく．

設定した課題は，2名でそれぞれ異なっている．1名は，単純な規則を設けた格子状の面に数字を並べ，それらの数が持つ関係を見つけ出すことを課題とした．数のもつ関係の詳細と成果については第2節で述べるが，ここで強調しておきたいことは，試行錯誤に自作のプログラムを用いているところである．大量の数字が並ぶ格子状の面をプログラムによって何通りも表現し，そこから特徴のある数列をいくつか発見したのち，理論的手法により数列どうしの関係を見抜くに至っている．つまり，具体例から一般化を試みるきっかけが，自らが作成したプログラムであったところに着目したい．

もう1名は，万有引力に係る式のみを用いて，惑星とその衛星および彗星の3体がそれぞれ影響を及ぼしつつ運動する様子（シミュレーション）を，プログラムで可能なかぎり正確に表現することを課題とした．こちらはプログラムそのものに試行錯誤を行っているところが特徴である．まず，プログラムで使用する方程式を選ぶところから始まり，数値計算の方法，誤差を少なくするための工夫，計算速度の向上，シミュレーションの表現方法など，さまざまなところで試行錯誤を行っている．詳細は第3節で述べるが，結果としてプログラムはまだ未完成ではあるが方向性をはっきりとさせており，将来的には学習教材として利用できるようプログラムを構築することを目標の1つとしたようである．

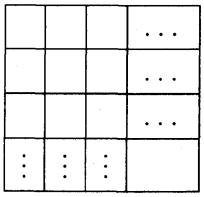
なお，第2節および第3節は，2名の学生がそれぞれ記している．課題の設定とその動機，問題解決に向けて自らがどう考え何を行ったか，が分かるように記した．今回の課題解決型学習で学生が得た成果も記してあることはもちろんであるが，学生自身でまとめた内容であることもふまえれば，この内容は課題解決型の教育実践の成果でもあると，私（齋藤）は考える．

2 実践例1 —格子面上に表現されるフィボナッチ数，カタラン数，累乗数—

2.1 動機と課題設定

当初，齋藤先生からもらった課題はカタラン数だった．コンビネーションの簡単な組み合わせで表現できるカタラン数であるが，パスカルの三角形のように数を斜めに並べて考えるとその本質が見えにくい気がした．何よりプログラムで扱いづらい．そこで縦横に並んだ格子面上でコンビネーションの規則を適用し，プログラムを組んで数値計算を行い，数を大量に並べ試行錯誤を行ったところ，フィボナッチ数，リュカ数，5の累乗が現れる一つの格子面を発見した．複数の数列が並ぶことに興味を感じ，その格子面上の数の関係を調べた．その成果を以下に示す．

右図のように長方形を縦，横に平行な線分で等間隔に分割したものを格子面と呼ぶことにし，分割してできた小さい長方形をマスと呼ぶことにする．縦の並びを列，横の並びを行と呼ぶことにする．



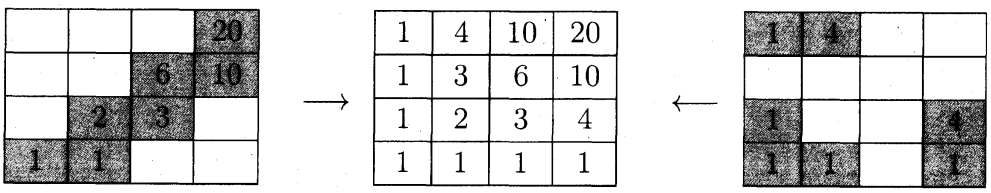
マスに以下の規則を満たすように数を記していく．

規則 そのマスに記す数は，左の数と下の数の和である．

例えば，4行4列の格子面の一番下の行と一番左の列の全てのマス に1を記したとき，格子面上には右図のように数が並ぶ．

1	4	10	20
1	3	6	10
1	2	3	4
1	1	1	1

値を決める最初の複数のマスの決め方は一通りではない．
例えば，真ん中の格子面をつくるには下図の左または右のようにどのマスをはじめに決定してもよい．



上のようにすべてのマスをただ一通りの数で埋めるため必要最小限の数を記した状態を「初期状態」と呼ぶことにする．

また，格子面は，行列と同様に A，B などと表すことにする．

2.2 格子面全体の集合とベクトル空間

2つの格子面 A と B を考える．次のことが成り立つ．証明は省略する．

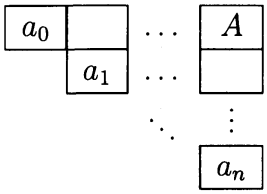
1. 格子面 A のマスの全ての値に定数 k をかけ合わせた格子面 kA は全てのマスで規則を満たす．

2. 格子面 A，B の対応するマス同士を足し合わせた格子面 $A+B$ は全てのマスで規則を満たす．

このことから，格子面全体の集合がベクトル空間をなすことが分かる．

右図にある A および $a_0, a_1, a_2, \cdots, a_n$ に対し

$$A = {}_n\mathrm{C}_0 \cdot a_0 + {}_n\mathrm{C}_1 \cdot a_1 + \cdots + {}_n\mathrm{C}_n \cdot a_n$$
$$= \sum_{k=0}^{n} {}_n\mathrm{C}_k \cdot a_k \quad \cdots \quad (1)$$



上記の等式が明らか成り立っている．

右のような格子面Pを考え，数列 a_n, b_n の関係を調べる．ただし $a_0 = b_0 = 1$ としておく．

$a_l = 0$（ただし l は負の整数）とすると式(1)より

$$b_n = \sum_{k=0}^{n} {}_{2n}C_{n-k} \cdot a_k$$

$P =$

0	1			b_3
0	1		b_2	
0	1	b_1		
0	b_0			
0	1	a_1		
0	1		a_2	
0	1			a_3

2.3 格子面 E_i による1次結合

1列目の全てのマスが0，2列目のマスの一つが1，そのマスから右上に斜めに並ぶ全てのマスが0の初期状態からできる格子面を E_0 とおく．

ここで，E_0 の初期状態の1が記されているマスを「マス (0,0)」と呼ぶことにし，このマスから下に α，右に β 移動したところのマスを「マス (α,β)」と呼ぶことにする．α は負でもよく，下向きを正としている．

次に，E_0 に記された数をすべて上と右に1マスずつ移し，空いた1列目のすべてのマスを0とした格子面を E_1 とおく．E_1 は規則を満たしている．以下同様に E_{n-1}(n は自然数) に記された数をすべて上と右に1マスずつ移し，空いた1列目のすべてのマスに0を記した格子面を E_n とおく．

$E_0 =$

0	1	2	2	**0**
0	1	1	**0**	-2
0	1	**0**	-1	-2
0	**1**	-1	-1	-1
0	1	-2	0	0
0	1	-3	2	0
0	1	-4	5	-2
0	1	-5	9	-7

, $E_1 =$

0	0	1	1	**0**
0	0	1	**0**	-1
0	0	**1**	-1	-1
0	**0**	1	-2	0
0	0	1	-3	2
0	0	1	-4	5
0	0	1	-5	9
0	0	1	-6	14

, $E_2 =$

0	0	0	1	**0**
0	0	0	**1**	-1
0	0	**0**	1	-2
0	**0**	0	1	-3
0	0	0	1	-4
0	0	0	1	-5
0	0	0	1	-6
0	0	0	1	-7

今までのことから，格子面 P は格子面 E_n の一次結合で表すことができる．

$$P = \sum_{k=0}^{\infty} b_k \cdot E_k$$

またマス $(\alpha,\ \beta)$ の値は　$(E_0$のマス (α,β) に記された数$) = (-1)^{\beta}({}_{\alpha}C_{\beta} + {}_{\alpha-1}C_{\beta-1})$ である.

$$\begin{aligned} a_n &= (P\text{のマス }(n,n)\text{に記された数}) \\ &= \sum_{k=0}^{n} b_k \cdot (E_0\text{のマス }(n+k,n-k)\text{に記された数}) \\ &= \sum_{k=0}^{n} (-1)^{n-k}({}_{n+k}C_{n-k} + {}_{n+k-1}C_{n-k-1}) \cdot b_k \\ &= \sum_{k=0}^{n} (-1)^{n-k} \frac{2n}{n+k} {}_{n+k}C_{n-k} \cdot b_k \end{aligned}$$

以上のことをまとめると

$$\begin{aligned} b_n &= \sum_{k=0}^{n} {}_{2n}C_{n-k} \cdot a_k \\ a_n &= \sum_{k=0}^{n} (-1)^{n-k} \frac{2n}{n+k} {}_{n+k}C_{n-k} \cdot b_k \end{aligned}$$

ここで $b_r = x^r$ とすれば

$$a_n = \sum_{k=0}^{n} (-1)^{n-k} \frac{2n}{n+k} {}_{n+k}C_{n-k} \cdot x^k$$

を得る. 実際に計算してみれば次の漸化式が成り立つことがわかる. ただし $a_0 = 2$ としている.

$$a_{n+2} + 2a_{n+1} + a_n = x\, a_{n+1} \quad \therefore \quad a_{n+2} - 2a_{n+1} + a_n = (x-4)\, a_{n+1}$$

最後の等式の左辺は階差の階差を表している.

$a_0 = 2,\ a_1 = x-2$ として漸化式から一般項を導く.

$$\alpha+\beta = x-2, \quad \alpha\beta = 1 \quad \text{より} \quad \alpha = \frac{x-2+\sqrt{x^2-4x}}{2}, \quad \beta = \frac{x-2-\sqrt{x^2-4x}}{2}$$

漸化式の解は α, β を使って次のように表される.

$$a_n = \alpha^n + \beta^n = \alpha^n + \alpha^{-n}$$

2.4　まとめ

2.1 節で述べた 5 の累乗とリュカ数の関係について考えてみる. 実際の格子面を下図に示す. リュカ数 $(2, 1, 3, 4, 7, 11, 18, 29, 47, \dots)$ の一般項は次の式で与えられる. ここで ϕ は黄金比である.

$$L_n = \phi^n + (-\phi)^{-n} \quad , \quad \phi = \frac{1+\sqrt{5}}{2}$$

これをみると a_n の一般項と形が似ている．ここで，$x=5$ とすると

$$\alpha = \phi^2$$

であることがわかる．したがって，$x=5$ のとき以下のようになる．

$$a_n = L_{2n}$$

これで最初の疑問が解けた．

0	1	8	40	165	
0	1	7	32		460
0	1	6		93	335
0	1		19	68	242
0		4	14	49	174
0	1		10	35	125
0	1	2		25	90
0	1	1	5		65
0	1	0	4	13	
0	1	-1	4	9	34
0	1	-2	5	5	25

注意深くみれば，リュカ数の左にはフィボナッチ数が，更にその左にはフィボナッチ数の平方数が並んでいることが分かる．さらには，リュカ数の右にはフィボナッチ数の5倍の数が並んでいることも分かる．

今後はこれらの数列も含め，いろいろなマスに散らばる数の関係性についても解明していきたい．

3 実践例2 —小惑星接近時における3体問題のシミュレーションプログラム—

3.1 動機と課題設定

当初，興味本位で万有引力のシミュレーションプログラムを作成していた．プログラムを改良していく上で，齋藤先生にシミュレーションではどのような計算方法をすれば良いのか相談をした際に，実際にあった出来事「2011年11月8日に，地球からわずか32.5万kmの位置を小惑星YU-55（直径400m程度・アポロ群の小惑星）が通過したこと」を例に，円運動をしている物体の近くを大質量の物体が通過したらどうなるのかといった，より現実的なテーマのシミュレーションの提案を頂いた．現実にはYU-55の通過において地球への衝突等の直接的な影響は無かったが，通過の仕方・状況が変われば影響が出る可能性がある．この可能性の有無をシミュレーションし，表現することを課題とした．

3.2 シミュレーションプログラムの開発

シミュレーションプログラムは当初，高校物理で学ぶ v=at , x=vt の式を用いた．このtに小さな値を代入することによって計算を行なっていた．具体的には，単純に万有引力による加速度にtをかけて速度を求め，さらに(1/2)tをかけることによって位置を計算していた．しかし，この方法では計算による誤差が非常に大きく，手計算による解析解とシミュレーションによる数値解にかなりの差が見られた．また，タイムステップであるtの値を非常に小さくしすぎたため，計算精度やシミュレーション時間がかかることが問題となった．解決方法として，4次のルンゲクッタ法（RK4）を利用するよう

にプログラムを修正した．RK4法は，4次のテイラー展開を用いているので一回あたりの演算で発生する誤差はタイムステップの4乗オーダーとなり (例として Δt=0.1 なら 10^{-4} オーダであるので5桁目から誤差が発生する)，解析解とシミュレーションによる数値解はほぼ一致するようになった．以下にルンゲクッタ法の計算例を示した．

$$y_{i+1} = y_i + \frac{1}{6}\Big(k_1 + 2k_2 + 2k_3 + k_4\Big)$$

$$k_1 = hf(x_i, y_i)$$

$$k_2 = hf\Big(x_i + \frac{1}{2}h, y_i + \frac{k_1}{2}\Big)$$

$$k_3 = hf\Big(x_i + \frac{1}{2}h, y_i + \frac{k_2}{2}\Big)$$

$$k_4 = hf(x_i + h, y_i)$$

以上をふまえ，地球・月・小惑星の3体が影響を及ぼしながら運動する様子を表現したシミュレーションプログラムを作成した．プログラムはJavaベースのProcessingで開発を行った．今回のプログラムにおいて想定しているシミュレーションのモデルは以下の通りである（図1）．地球に比べ他の2体の星の質量は小さいことから，地球は質点として固定し，そのまわりを月が周回するモデルとした．月は円運動を保つ最小の速度ベクトルが初期値として与えられており，それによって円運動を保っている．

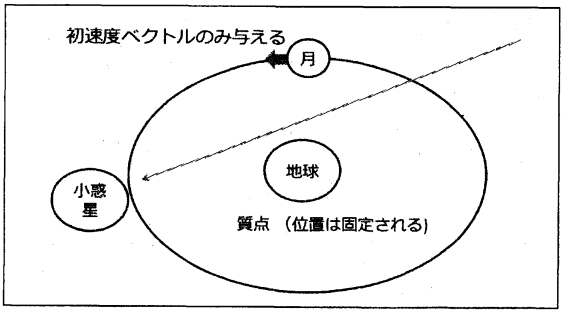


図 1: シミュレーションのモデル

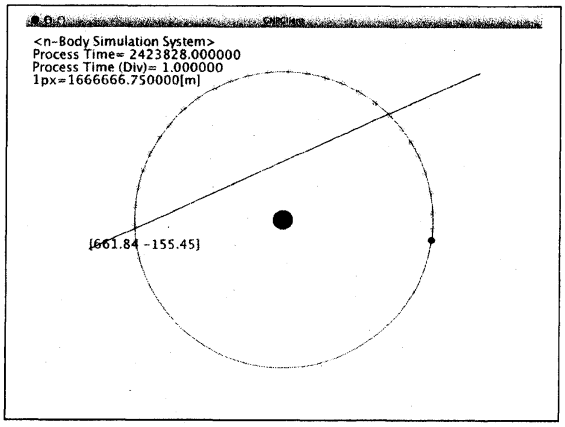


図 2: 実際のプログラムの画面

3.2.1 プログラム内部で使用している計算式

ある物体Aの位置を $(x_a,\ y_a)$ で表し，質量を m とする．Aとは別の物体Bの位置を $(x_b,\ y_b)$ で表し，質量を M とする．物体Aと物体Bは互いに万有引力によって力が生じているとする．その時，物体Aの生じる加速度の各成分は以下のようになる．加速度は時間に関する二階常微分方程式になる事が知られている．RK4法を2回適用する事によって加速度から位置を算出する事が可能であるが，各成分 $(x,\ y)$ の加速度の式は各

成分 $(x,\ y)$ の位置に関する連立方程式となっている．この式にRK4法を適用するには，RK4法を多変数に拡張する必要性がある．

$$m\frac{d^2x_a}{dt^2} = \frac{GMm}{r^2}\frac{x_a}{r} = \frac{GMmx_a}{r^3} \quad \text{つまり} \quad \frac{d^2x_a}{dt^2} = \frac{GMx_a}{r^3}$$

$$m\frac{d^2y_a}{dt^2} = \frac{GMm}{r^2}\frac{y_a}{r} = \frac{GMmy_a}{r^3} \quad \text{つまり} \quad \frac{d^2y_a}{dt^2} = \frac{GMy_a}{r^3}$$

$$r = \sqrt{(x_a - x_b)^2 + (y_a - y_b)^2}$$

3.2.2 シミュレーションの手順

(1) 各物体のパラメータ決定
これは物体の質量・初期速度等を決める過程であり，プログラム初期化と同時に行われる．
(2) 計算対象の物体を決定
(3) 計算対象の物体に生じる万有引力を計算
(4) 計算対象の物体にRK4法を適用
(5) 計算対象の物体を変更

下記に2変数によるRK4法の計算手順を示した．関数 f は x 成分の加速度の関数（2変数関数），関数 g は y 成分の加速度の関数（同様に2変数関数）である．変数 $x,\ y$ は物体の座標を表し，その一回微分である x', y' は物体の速度 $(x,\ y$ 成分 $)$ を表す．また，二階微分の形である x'', y'' は物体の加速度を表す．dt はタイムステップである．x', y' に物体の初速度を代入し，$x,\ y$ に物体の位置座標を代入し，以下のRK4による計算式を計算する事によって新しい位置を計算する事ができる．なお，以下の計算式はプログラミングを意識して表記している．

$$\begin{aligned}
x' &= vx & vx' &= f(x,\ y)\\
y' &= vy & vy' &= g(x,\ y)\\
kx_1 &= vx * dt & kv_1 &= f(x,\ y) * dt\\
ky_1 &= vy * dt & kw_1 &= g(x,\ y) * dt\\
kx_2 &= (vx + kv_1/2) * dt & kv_2 &= f(x + kx_1/2,\ y + ky_1/2) * dt\\
ky_2 &= (vy + kw_1/2) * dt & kw_2 &= g(x + kx_1/2,\ y + ky_1/2) * dt\\
kx_3 &= (vx + kv_2/2) * dt & kv_3 &= f(x + kx_2/2,\ y + ky_2/2) * dt\\
ky_3 &= (vy + kw_2/2) * dt & kw_3 &= g(x + kx_2/2,\ y + ky_2/2) * dt\\
kx_4 &= (vx + kv_3) * dt & kv_4 &= f(x + kx_3,\ y + ky_3) * dt\\
ky_4 &= (vy + kw_3) * dt & kw_4 &= g(x + kx_3,\ y + ky_3) * dt\\
x &= x + \frac{(kx_1 + 2 * kx_2 + 2 * kx_3 + kx_4)}{6} & vx &= vx + \frac{(kv_1 + 2 * kv_2 + 2 * kv_3 + kv_4)}{6}\\
y &= y + \frac{(ky_1 + 2 * ky_2 + 2 * ky_3 + ky_4)}{6} & vy &= vy + \frac{(kw_1 + 2 * kw_2 + 2 * kw_3 + kw_4)}{6}
\end{aligned}$$

また，実際のプログラムにおいて上記計算式の関数 f, g は以下の関数で実装されている．下記の関数は，引数 h によって計算する成分を変更する事ができる．

ソースコード 1: プログラムの例

```
//変数myselfで指定されたIDの物体に受ける加速度を算出する関数
double f(ArrayList PlanetList,int myself , double h, double x, double y , double z){
  //定数を含む変数等の初期化
  double res=0.0;
  double temp,G=6.673 * Math.pow(10,-11.0);
  //計算ループ(ArrayList内の物体の数だけループする)
  for(int i=0; i< PlanetList.size(); i++){
        if(i==myself) continue; //自分自身から受ける力は計算しない
        PVectorD vec = new PVectorD( ((PlanetClass)PlanetList.get(i)).Position );
        vec.sub(x,y,z); //引数で与えられた座標で減算
        if(h==0) temp=vec.x; else if(h==1) temp=vec.y; else if(h==2) temp=vec.z;
        else temp=0; //引数hはどの方向の演算を行うかを表している
        //今回のループで計算した力による加速度を加算する
        res+= G * ((PlanetClass)PlanetList.get(i)).m * temp * Math.pow(vec.mag(),-3.0);
  }
  return res;
}
```

3.3 開発したプログラムの特徴

プログラムの計算速度を向上するため，プログラムを2つに分離した．プログラムはTCP/IP のソケット通信によってクライアント・サーバー側にわけられており，ネットワーク経由で演算結果を共有させる事が可能である．クライアント側にはサーバーへのコマンド送信・表示スケール等の調整用のインターフェースが存在する．また，ウィンドウの任意のポイントをドラッグする事によって，そのポイントよりドラッグした方向へと小惑星を通過させる事ができる．シミュレートする物体は，クラスによって定義されArrayList によって管理されている．その為，シミュレートする物体は動的に変更する事が可能であり，今回のような3体問題だけではなく重力多体問題 (物体数が百個以上) 等もシミュレーション出来るようにプログラムをした．

3.4 まとめ

今回の研究によって，プログラムで数値計算としての積分を行う方法を学習する事が出来た．また，計算精度についても考えるきっかけとなった．今回の研究を通して学んだ事をより深め，今後はシミュレーションの精度向上，速度向上等を行いたいと考えている．また，本校航空宇宙工学コースの教員より逃亡星（大質量の星によって他の星が加速される現象）のパターンもシミュレーションできるようにしてはどうかとアドバイスを頂いた．その為，今回開発したプログラムは地球・月・小惑星のパターンだけではなく，様々なシミュレーションも出来るように改良した．

今後はこのソフトウェアを，万有引力を用いた様々なシミュレーションを手軽に見積もる事のできるプログラムへと改良し，教育を目的とした学習教材になれば良いと考えている．

参考文献

[1] J.H. コンウェイ，R.K. ガイ,「数の本」(根上生也 訳)，シュプリンガー・ジャパン．

[2] 数学セミナー Vol.48, No.7, 574,「天体力学と数学」，日本評論社．