技術紹介

ゼロからはじめるシミュレーション・プログラミング入門

国立遺伝学研究所　新分野創造センター　細胞建築研究室　　木村　暁

● はじめに

本稿はコンピュータ・シミュレーションに興味はあるものの、コンピュータ・プログラミングに対する知識・経験が全くない方のために書きました。実は５年前の私自身がそのような人でした。当時の自分を思い出し、知識ゼロの状態から手近な材料でプログラミングの「さわり」を体験できるように書いたつもりです。これを読んで「自分も挑戦してみたい」という人が現れれば幸いです。

● なぜ生物学者がシミュレーション？

「自分や他人がたてたモデルがどのような挙動を示すか、コンピュータを使って追いたい」というのは生命現象の理解を目指す生物学者にとって自然な発想ではないかと思います。問題は費用対効果の評価でしょう。「知識･時間・労力を要する」わりには「実際の研究に役立たないのではないか」というのが大方の意見だと思います。５年前に私は慶應義塾大学の大浪修一研究室に入り、ゼロから計算機を活用した生物研究を学びました。この経験から私は、

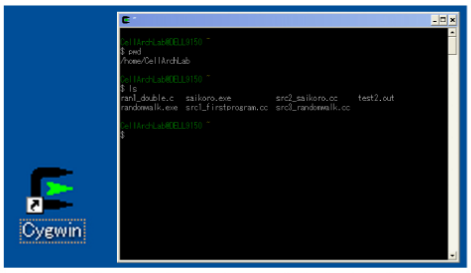

図１A. Windows にインストールされた Cygwin のアイコン（左）と、そのアイコンをクリックして Cygwin を起動したところ（右）。

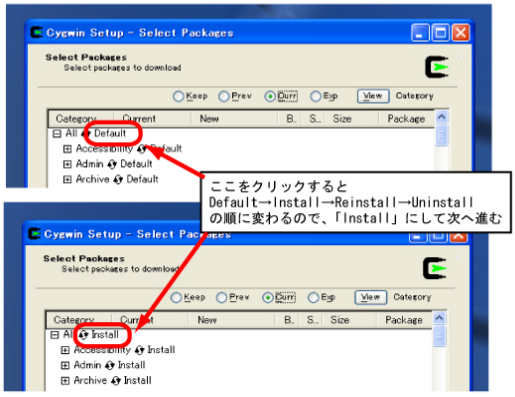

図１B. Cygwin インストール時の注意点。途中 [Select Packages] のところで、All Install を選んでください。パソコンの空き容量の少ない方は、その下の [Devel] のところだけ [Install] とするのでも結構です。

○　初歩的なプログラミング技術で単純なシミュレーションを行うだけでも多くの示唆が得られること
○　実際の研究の進展に役立つ貢献をシミュレーションから引き出すには、生物学者自らがシミュレーションの構築や実行の現場に携わることが効率的であること

の 2 点について確信するようになりました。複雑に多くの要素が絡み合う生命現象を詳細にシミュレーションすることは困難ですが、少なくとも我々が立てた仮説（多くの場合、単純）がどのような挙動を示すかについてはシミュレーション可能なはずです。みなさんも自分の仮説の挙動を自分自身でテストしてみませんか？

● ステップ1：道具を用意しよう

まずは道具(環境)を整えましょう。私は普通のパソコンに(Windows ではなく)Linux という UNIX 系 OS をインストールしたものを使っています。UNIX 系 OS には「C 言語」をはじめ様々なプログラム開発の道具が標準で含まれており、すぐにプログラムを作ることができます。Linux は無料で入手できますし、パソコンも安いもので十分です [1]。

今回は「さわりの部分の体験」ということで、Linux 用のパソコンを用意しない方法を使ってみます。Windows パソコンに「Cygwin」という無料ソフトをインストールすれば、UNIX 環境を体験することができます(**図1**)。

では、Windowsのパソコンに「Cygwin」をインストールしてみましょう。CygwinのWebサイト（http://cygwin.com/）の右上のボタン（Install Cygwin now）をクリックすると、Cygwinのインストールが開始されます。インストール作業は、直感的に進められると思います[2]。一点だけ、途中[Select Packages]のところで、[All Install]と設定してください(**図1B**)。インストールが完了すれば、「Cygwin」のアイコンがデスクトップにあらわれ、それをクリックすれば文字が入力できる黒い画面があらわれます（**図1A**）。

Macintosh のパソコンの場合は、Mac OS X 自身が UNIX 系の OS なので、そのまま UNIX を使用することができます。[Applications] の中の [Utilities] フォルダにある [Terminal]を選択すると文字を打ち込む画面が出てくるはずです。これから紹介する C 言語でのプログラミングには Xcode(OS X 10.2 以前の場合は Developer Tools)なるツールをインストール(ダウンロード無料)する必要があります。

● ステップ2：C言語のプログラムを作ろう

では早速、プログラムを作ってみましょう。ここでは C 言語を使います。C 言語は初心者に理解しやすく、応用も効くので基本的なプログラミング言語です[3]。

最初の例は次の５行からなるプログラムです。このプログラムをパソコン付属のテキストエディタ等(Windows なら

「メモ帳（Notepad）」ソフト）を使って打ち込み、「program1.c」と名前をつけて保存してください(**図2**)。

```
<program1.c>-----------------------------------
#include <stdio.h>
int main()
{
  printf ("nuclear dynamics is important!¥n");
}
-------------------------------------------------
```

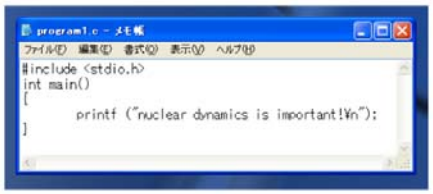

図２．メモ帳でのプログラムの打ち込みの様子。

保存場所は［ローカルディスク(C:)］→[cygwin]→[home]→[ユーザー名]というフォルダにします(**図3**)。Windows vista の場合は[ローカルディスク(C:)]→[ユーザー(Users)]→[ユーザー名]、Macintosh の場合は[Macintosh HD]→[Users]→[ユーザー名] となるようです。

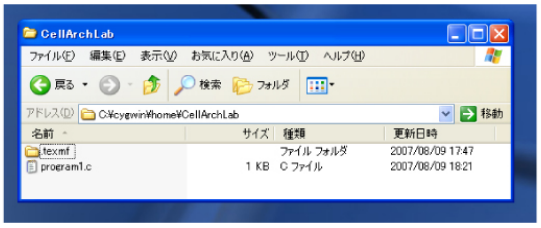

図３．パソコン上にできた cygwin 用のフォルダ

● ステップ3：プログラムを動かしてみよう

このプログラムを動かしましょう。Cygwin を起動してあらわれる黒い画面、もしくは Mac の Terminal に戻ります。ここで、まず「ls･」（･は改行）と入力してみてください。さきほど作ったprogram1.cというファイル名が表示されるはずです（**図４①**）。これが確認できたら「gcc program1.c･」と打ち込んでください(**図4②**)。この操作は作ったプログラムを機械の言葉に翻訳するコンパイルと呼ばれる作業になります。これにより同じフォルダに「a.exe」（または「a.out」）というファイルができます（「ls･」と入力して確認してみましょう）（**図4③**)。いよいよ実行です。「./a.exe･」（または「./a.out･」(以下同じ)）と入力してください。画面には「nuclear dynamics is important!」と表示されるはずです(**図4④**)。これで最初のプログラムは成功です。

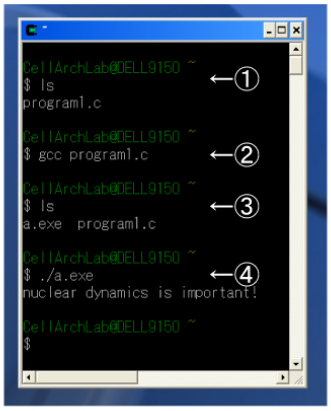

図４．プログラムの存在を確認し（①）、プログラムをコンパイルし（②）、実行ファイルの存在を確認し（③）、実行した（④）様子。

● ステップ4：ランダムさがシミュレーションの醍醐味

私が考えるシミュレーションの大きな魅力は確率的に起こる事象の挙動を解析できることです。生体内では、例えば分子の移動も熱ゆらぎによる影響を大きく受けるなど、決定論ではなく確率論的な事象がほとんどです。個々の事象は確率的に起こるのに、それが積み重なった全体としては合目的的で再現性のある挙動を示すのが生命現象の不思議な点であり、興味のあるところです。こういった現象について人間の頭の中だけで理解を試みるのは困難です。コンピュータを使えば、確率的に起こるさまざまな場合についての挙動を検討できます。

確率的に起こる事象をコンピュータで再現するためには、コンピュータに「サイコロ」を振らせてランダムな数を作り出させる必要があります。ここでは簡便のために C 言語の rand 関数を用いてランダムに 0.0-1.0 の数を生成します。この数に６を掛けて切り上げれば、１－６の整数をランダムに生成できることになります（**図5**）。ここで、rand 関数を用いた方法では完全にランダムな乱数が生成できないことが指摘されており、より高い質の乱数を発生させる方法が開発されています[4]。研究に使うプログラムにはそのような高性能の乱数発生方法を使ってください。

```
<saikoro.c>-----------------------------------
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
int main()
{
        int saikoro;
        float rnd;
        srand(time(NULL));
        rnd = (float)rand()/RAND_MAX;
        saikoro = (int)(rnd*6.0)+1;
        printf ("saikoro: %d¥n",saikoro);
}
```

図５．サイコロプログラムの実行。１－６の数字が出るが、実行するたびに異なる数字がでる（①②）。

● ステップ5：ランダムウォークをシミュレーション！

コンピュータの長所は、人間が嫌になるような繰り返し作業を、黙々と、正確に、素早くこなしてくれることです。この長所とランダムさを組み合わせることによって、確率的に起こる事象の組み合わせで何が起こるかを検討することが可能です。

例として、ランダムウォークに挑戦してみましょう。1/4の確率で、東西南北のいずれかの方向に１歩すすむという作業を 1000 回繰り返すプログラムを作ってみます。原点(0,0)から出発します。先ほどと同様に１－４の数字がランダムに発生させ、この数が１なら東(x 軸正方向)に、２なら西(x 軸負方向)に、３なら北(y 軸正方向)に、それ以外（４）なら南(y 軸負方向)に１ずつすすむという作業を 1000 回繰り返すようにプログラムを作りました。分岐(場合分け)や繰り返し処理はプログラムの基本中の基本で、C 言語ではそれぞれ「if」や「for」を使って命令します。各ステップでの現在地の座標を出力するようにしています。

```
<randomwalk.c>----------------------------------
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
int main()
{
        int STEP=1000;
        int x, y;
        int dice;
        float rnd;
        int i;
        srand(time(NULL));
        x=0; y=0;
        printf ("%d %d¥n",x,y);
        for (i=0; i<STEP; i++) {
                rnd = (float)rand()/RAND_MAX;
                dice = (int)(rnd*4.0)+1;
                if (dice==1) {
                        x=x+1;
                } else if (dice==2) {
                        x=x-1;
                } else if (dice==3) {
                        y=y+1;
                } else {
                        y=y-1;
                }
                printf ("%d %d¥n",x,y);
        }
}
-----------------------------------------------
```

このプログラムをこれまで同様実行してみましょう。２つの数字（xy 座標）が繰り返し出力されるはずです。これでは何のことかわからないので、結果を図で表したいということになります。実行結果を図で表す簡便な方法として、表計算ソフト「Microsoft Excel」を使ってランダムウォークの様子を見てみましょう。これまでプログラムの実行時には「./a.exe」と入力していましたが、少し付け足して「./a.exe >> output.txt」と入力します（**図6①**）。そうすると、output.txt というテキストファイルが作られ、そこに結果である数字の羅列が出力されます。

図６．ランダムウォークプログラムの実行。実行時に「./a.exe >> output.txt」とする（①）と実行結果が画面に表示されるのではなく、output.txt というファイルに格納される（②）。

Excel を起動し、［ファイル(F)］→［開く(O)］で output.txtを探します。この時、ファイルの種類を「すべてのファイル(*.*)」としておくことがポイントです。output.txtを開こうとすると、「テキストファイルウィザード」というものがでてきます。「テキストファイルウィザード 1/3」では[次へ]を選択し、「テキストファイルウィザード 2/3」で[区切り文字]のところで「スペース(S)」にチェックを入れて[完了(F)]とします。すると、2 列からなる表があらわれます。この状態からグラフ（データポイントを折れ線でつないだ散布図）を作成すれば、原点を出発点としたランダムウォークの軌跡が図示されます(**図7**)。

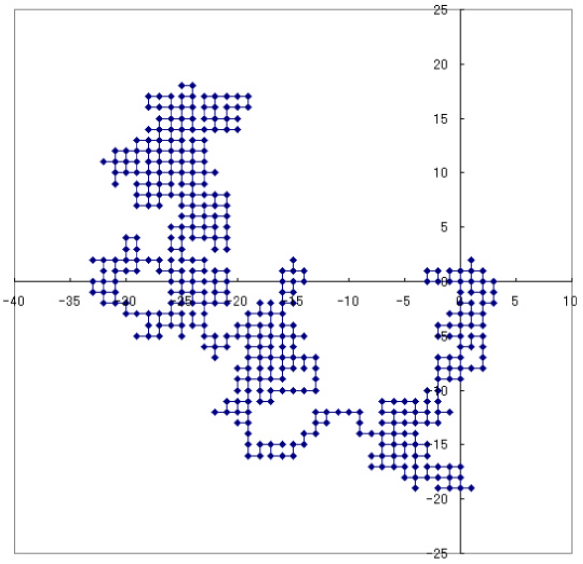

図７．表計算ソフト Excel で表示したランダムウォークプログラムの実行結果。もちろん実行毎にパターンは変わる

● おわりに

私自身は今でもプログラミング初心者です。これを機会に一人でも多くの人にシミュレーションに興味を持っていただきたいことから、恥を忍んで(?)この原稿を書きました。シミュレーションで生命現象を忠実・精密に再現することはまだまだ困難ですが、構築したモデルの十分性・不十分性を議論するにはかなり有効なツールと考えます。モデルを提案する際に、そのシミュレーションを示すことが当たり前になる日もそう遠くないかもしれません[5]。

本稿で記載した内容は、筆者が研究員としてお世話になった慶應義塾大学（文科省振興調整費「システム生物学者育成プログラム」）・大浪修一研究室（現・理研 GSC）で学んだことです。大浪修一先生および濱橋秀互博士、京田耕司博士をはじめとする研究室員の方々に感謝します。新しいことにチャレンジする際には、最初の敷居を越えるところが最大の難関です。私は大浪研究室の方々から強力な支援をいただきましたが、私自身も微力ながら誰かの挑戦の役に立ちたいと考えています。不明な点やご指摘、ご相談があれば遠慮なくご連絡ください。

● 参考文献など

Linux、Cygwin、C 言語の情報はインターネット上に多数掲載されており、検索エンジン(google など)を利用して、ほとんどの疑問は解決できます。OS, ソフトウェアも無料でダウンロード可能なので、費用をかける必要はありません。以下は、本から学びたい方のための参考文献です。

[1] 大津真他著「Fedora 7 ビギナーズバイブル（インストール DVD 付）（毎日コミュニケーションズ）」など
[2] 小川淳一著「Cygwin 徹底入門（ソーテック社）」など
[3] 倉薫著「C 言語①②（翔泳社）」など
[4] C 言語で数値計算をする際に参照する「Numerical Recipes in C （和訳：C 言語による数値計算のレシピ、技術評論社)」という有名な本があります。この中にも乱数生成のプログラムが掲載されています。但し、この乱数生成法にも欠点が指摘されています。高精度な乱数生成アルゴリズムとして評価が高いのは、メルセンヌ・ツイスタ法と呼ばれる方法です。
[5] シミュレーションを使った研究の長所・欠点については次の総説も参考にしてください：木村暁、大浪修一「コンピュータシミュレーションを利用した核の配置のダイナミクス解析」*蛋白質核酸酵素* **51**, 2172-2179 (2006)