2020 年度 情報工学実験 II 授業資料

情報工学実験 II の 2020 年度の授業は遠隔授業で実施します。情報工学実験 II では次項の実施方法のうち A(1)A(2)と B を組合せて採用します。実験テーマ毎に実施の形態は異なり、また状況や進行に応じて適宜改善します。現時点での「離散・連続シミュレーション実験」についてはガイダンス資料中の説明を参照してください。
授業に必要な資料やテキストは郵送・ネット配信・電子メールへの添付送信で提供します。それぞれの手段で提供する資料には重複する部分、他の手段を補う部分があります。全体をよく参照し実験実施に役立ててください。
資料の入手々段に対応できない場合にも十分な受講機会を得られるよう支援します。授業実施にあたり情報の不足を感じる場合には遠慮なく申し出てください。
疑問の解消や質問のためにオンライン授業や電子メールによる問い合わせの受け付けを実施します。また、ネット配信により情報の訂正や更新を行なうことがありますので、授業に関する通知に注意を払ってください。
現時点で予定している「離散・連続シミュレーション実験」の進め方は次の通りです。

1. 授業資料を参照し自習で課題に取り組む。
2. 電子メールで質問事項をまとめ送信する。
3. オンライン授業での全体説明や追加の資料配信により問題を解消する。
4. 3 で解決しなかった問題についてオンライン授業中に TA に質問して問題を解決する。
5. 1 に戻って繰り返す。

質問の多くは受講者に共通するものなので、オンライン授業では全体への説明もあります。自分の質問以外の説明も得られるので活用してください。1〜5 を繰り返し、課題を完了してください。
課題の成果をもとにレポートを完成させ、提出してください。

愛媛大学における 2020 年度遠隔授業の実施方法

- 遠隔授業A(1)：動画などのネット配信による遠隔授業（同期型）
- 遠隔授業A(2)：動画などのネット配信による遠隔授業（非同期型）
- 遠隔授業B ：修学支援システム等のメールにより課題を与え、指導を行う遠隔授業

配付資料

事情により○印の資料のみを郵送します。○の無い資料はネット配信で入手できます。

配付資料について説明します。

本票以降の資料は以下の通りです。

1. ガイダンス資料（2×4 段組 4 頁）

   4 月 17 日のガイダンスで使用したスライドを一部修正したものです。授業全体の注意事項や遠隔授業に必要な準備について説明しています。とくに学外から教育用計算機システムを利用する方法について、よく確認をしてください。

② 離散シミュレーション実験テキスト（2 段組 6 頁）

③ 連続シミュレーション実験テキスト（2 段組 6 頁）

④ レポート提出の注意事項（2 段組 3 頁）

   実験課題のテキストとレポート作成の注意事項です。よく読んで取り組んでください。

5. 2019 年度授業スライド（2×4 段組 6 頁）

   昨年度の授業で使用したスライドです。通常の授業進行が分かりますので、自習での作業が中心となる遠隔授業の進捗目標を決めるための参考にしてください。また、5 日目のスライドにはレポートに記載すべきことについての説明があります。レポート作成に役立ててください。

6. 離散シミュレーション実験サンプルレポート（2 段組 6 頁）
7. 連続シミュレーション実験サンプルレポート（2 段組 7 頁）

   成績評価のために提出する 2 つのレポートのサンプルです。どのような構成が考えられるのが例として扱ってください。（このまま提出できるものではありません。）

8. エディタ Atom のインストールと設定（2×2 段組 2 頁）
9. コンパイラ gcc のインストールと設定（2×2 段組 6 頁）
10. Atom での C プログラミングと gcc でのコンパイル（2×2 段組 3 頁）

    メディアセンタの阿萬先生に用意していただいた Windows に学科計算機室に似たシンプルな C プログラミング環境を導入する方法を説明した資料です。

    Linux / MacOS の方はターミナルからコマンドを入力する導入方法が説明されますので、そちらを参照してください。

授業日程

授業日程です。6 回目からウェブアプリケーション実験が開始されます。日程が近くなったら、また説明をします。2 回目の授業資料郵送も利用するかもしれません。

第 0 回(04/17)：ガイダンス (スライド)
第 1 回(04/24)：離散シミュレーション実験（1）グラフの表現
第 2 回(05/01)：離散シミュレーション実験（2）最短経路探索
第 3 回(05/08)：連続シミュレーション実験（1）常微分方程式の解法
第 4 回(05/15)：連続シミュレーション実験（2）質点のシミュレーション
第 5 回(05/22)：レポート作成のためのまとめ

その他の資料・情報提供

離散・連続シミュレーション実験は次の URL で授業情報を提供します。

URL: http://comp.cs.ehime-u.ac.jp/~okano/jikken2/

まだ準備ができていませんが moodle からも情報を提供できる予定です。

情報工学実験 II 手順書

ソフトウェア—離散系のシミュレーション

version 1.0

愛媛大学工学部情報工学科

2020 年度

目次

1　ソフトウェア—離散系のシミュレーション

1.1　はじめに

計算機を用いて行なわれる様々な計算を、その対象となる集合の性質により「離散 」的な計算と「連続」的な計算とに分けて考えることがある。本節では 2 種類の計算のうち「離散」に属する計算におけるプログラミングの基礎を学ぶことを目的とする。また、そのための題材として、離散計算の重要な部分を成すグラフ理論に基づく計算機プログラムを取り上げる。具体的には、重み付き無向グラフを対象に最短路検索の問題に対する代表的な計算アルゴリズムである Dijkstra の方法を用いたプログラムを作成し、その評価をすることで離散的な計算における計算機プログラムの実際を体験・学習する。

本節の内容は先行する講義科目「情報数学 II」における学習成果を基礎に置いており、また履修にあたって学生がその内容を修得していることを前提とする。

1.2　実験の進行について

十分な学習成果を得るために、提示された実験課題は漏らすことなく解決する必要がある。

2 回の実験実施日への課題の配分は概ね自由で構わないが、1 回目には課題 3 までを必ず終了させること。

また、以下に出てくるサンプルプログラムは、授業情報のウェブサイトからダウンロードできるので、それを使うこと。

1.2.1　中間レポート

各実験実施日の終りには担当者による進捗状況の確認を受けなくてはならない。その日の実験を終える前に、取り組んだ課題に関する必要十分な報告書を提出すること。

報告書に併せて、必要に応じて口頭試問を行うので、報告書または試問の結果について、不十分であると指摘された場合は、時間内もしくは自習時間中に報告書の修正、内容に関する再学習を行うこと。全ての課題への取り組みが十分に成されているという担当者の確認を経ずに、その日の実験を終了することはできない。

進捗状況の確認は学習の円滑な進行のために行うものである。必要な作業・考察の全てを授業時間中に終えるためのアドバイスを得る機会であり、試験・審査の類ではない。単位の認定は実験の実施と提出課題の評価によってのみ行う。報告書に要した時間や修正指示の回数、口頭試問の良否は評価には影響しない。

1.3　グラフと対応する数理モデル

「情報数学 II」の講義でも学んだ通り、グラフとは集合とその要素間の二項関係を併せたものであり、要素を頂点、要素間の関係を辺で置き変えた図形による表現である。

図 1 は 5 つの要素とその間の二項関係を 5 つの頂点 $\{1,2,3,4,5\}$ と 6 つの辺 $\{\{1,2\},\{1,5\},\{2,3\},\{2,5\},\{3,4\},\{3,5\}\}$ で表したものである。例えば、1:JR 2:堀端 3:市駅 4:温泉 5:本町 などと頂点に名前をつけてみれば、どこかの町の鉄道路線図のように見えてくる。

このようなグラフをデータとして C のプログラム中で扱うことを考える。

グラフの頂点には順番に番号を振ることにすれば、頂点に関してはその数だけを記憶しておけばよい。あとは、どのような辺が存在するか、すなわちどの要素間に関連があるかを記憶することが必要になる。

グラフの辺は頂点を端点に選んだものであり、頂点の数を n とすると、$n \times n$ 通りの辺が考えられる。これを 2 次元の配列を使って保持することを考える。

プログラムリスト 1(inputgraph.c) は標準入力から頂点の数と各辺の端点となる頂点の番号を読み込むサブルーチンと、そのサブルーチンを使って読み込んだ配列の内容を表示するプログラムの例である。

プログラムリスト **1**

無向グラフの入力プログラム例　　inputgraph.c

```
 1  #include <stdio.h>
 2  #include <stdlib.h>
 3  #include <limits.h>
 4
 5  #define NMAX 10                  /* 頂点の数の上限 */
 6  int graph[NMAX + 1][NMAX + 1];   /* 辺 (関連) の有無 */
 7  int n;                           /* 頂点の数 */
 8
 9  #define FALSE  0
10  #define TRUE   1
11
12  void inputgraph(void)            /* グラフデータ読込み */
13  {
14    int i, j;
15
16    puts("頂点の数を入力してください");
17    if (scanf("%d%*[^\n]", &n) != 1 || n > NMAX)
18      {
19        n = 0;  return;
20      }
21
22    for (i = 1; i <= n; i++)
23      for (j = 1; j <= n; j++)
24        graph[i][j] = FALSE;
25
26    puts("各辺の両端点を番号で入力してください");
27    while (scanf("%d%d%*[^\n]", &i, &j) == 2)
28      {
29        graph[i][j] = TRUE;
30        graph[j][i] = TRUE;
31      }
32  }
33
34  int main()
35  {
36    int i, j;
37
38    inputgraph();  /* 点の数 n, 辺の有無 graph[][] を入力 */
39
40    puts("入力データ graph(i,j)=F/T");
```

```
41    for (i = 1; i <= n; i++) {
42      printf("%3d",i);
43      for (j = 1; j <= i; j++) {
44        if (graph[i][j])
45  fputs("  T",stdout);
46        else
47  fputs("  F",stdout);
48      }
49      puts("");
50    }
51    fputs("   ",stdout);
52    for (i = 1; i <= n; i++) printf("%3d",i);
53    puts("");
54
55    return EXIT_SUCCESS;
56  }
```

inputgraph.c において、関数 inputgraph は標準入力から頂点数 n と各辺の端点番号を読み込む。グラフは配列 graph[][] に辺の有無を真 (1) 偽 (0) の整数値として格納される。すなわち、辺 $\{i, j\}$ が存在すれば graph[i][j] は真 (1) となり、存在しなければ偽 (0) となる。

また、inputgraph 関数の使用例として、main 関数では入力されたグラフの内容を一覧表示している。

課題 **1** プログラムリスト 1 または同等のプログラムを作成し、図 1 のグラフに対応するデータの入力と確認を行え。

課題 **2** プログラムリスト 1 では扱うグラフは無向グラフであると仮定され、要素 i と j を結ぶ辺の入力に対して、配列 graph の成分 graph[i][j] と graph[j][i] の両方を真にしている。

有向グラフを扱う方法を検討し、考案した方法に則して課題 1 で作成したプログラムを変更せよ。

プログラムリスト 1 ではグラフのデータを格納する配列はデータの入力に先立って定義されている。したがって、利用できるグラフの頂点数は配列の大きさに制限され、プログラムの実行時には最大頂点数は変更できない。

十分に大きな配列を定義しておくという解決策は不必要なメモリの占有を生じ、結果として、他のデータの取り扱いのみならず、同時に実行されている別のプログラムの処理にも悪影響を与えてしまうかもしれない。そこで、配列の大きさを必要に応じて動的に変更することを考える。C 言語では他の言語に比べ、配列はポインタという概念と強く結びついている。したがって、配列をうまく扱うためにはポインタの理解が必要となる。

全ての変数はメモリ上のどこかにその値を格納する。変数の値が格納されるメモリ上の位置を、ここでは便宜上、アドレスと呼ぶ。特定の変数のアドレスを取り出すためにはアドレス演算子 & が用いられる。すなわち、&a は変数 a のアドレスを返す。

一方、ポインタとはポインタ型の変数のことであり、ポインタ型変数は、その値としてアドレスを持つ変数である。

また、ポインタ型変数はどのような型の変数のためのアドレスを持つのか宣言時に指定する必要がある。このとき、宣言されたポインタ型変数を指定された変数型から派生したポインタ型変数と呼ぶ。例えば、型宣言中に、次のように宣言された変数 b は int 型変数から派生したポインタ型変数である。


```
int *b
```

このとき、プログラム中ではポインタ型変数 b の持つアドレスに格納された int 型の値を *b で参照できる。

すなわち、int 型変数 a と int 型から派生したポインタ型変数 b が

```
int a, *b
```

と宣言されたとき、b=&a として、b に a のアドレスを代入すれば、*b は &a の指し示すメモリに格納された int 型変数値を与えるので a==*b が成立する。b や a のアドレス &a に変化が無ければ、変数 *b や a によっても a==*b が成立する。注意すべきは、他の変数と同様、ポインタ型変数は宣言されただけでは、その値は不定ということである。

さらに、ポインタに格納されたアドレスの値を使って連続して確保された複数の領域を参照し値を格納することができる。例えば、int 型から派生したポインタ型変数 a

```
int *a
```

に対して十分に領域が確保され、そのアドレスが代入されているとき、a+1 は a の次の int 型の領域のアドレスを示す。つまり、

```
*a = 1; *(a+1) = 3; *(a+2) = 5;
```

などとして、それぞれ独立した変数として異なる値を格納し読み出すことができる。このとき、ポインタの指し示す領域に格納された値を参照する演算子*が加算演算子 + よりも優先順位の高い演算子であることに注意する必要がある。例えば上述の代入の後で、*a+*(a+1)、*(a+*a) で読み出される値はそれぞれ 4 と 3 である。

このようなアドレスの加算を行なう表現には次のような [] を用いる方式もある。

```
a[0] = 1;  a[1] = 3;  a[2] = 5;
```

それぞれ、a に格納されたアドレスに続く「[] 内の値」番目の領域に格納された値を読み出すことができる。つまり、a[0] と*a、a[1] と*(a+1)、a[2] と*(a+2) が全て同等になる。

C の配列は、このようなポインタ型変数とそのアドレスに 対する演算によって実現されている。例えば、int 型配列が

```
int c[3]
```

等として宣言された場合、メモリ上に int 型の変数 3 つ分の領域が確保され、その位置がアドレスとして特定される。このとき、c[0]==*c, c[1]==*(c+1), c[2]==*(c+2), が成立する。つまり、*(c+j) ($j = 0, \ldots, 2$) により c[j] のアドレスに格納された値が参照できる。

このことは、配列の宣言が指定された個数の値をメモリ上に格納するためのアドレスの確保と、対応するポインタ型変数の宣言とアドレスの代入にほぼ等しいということを示している。実際、

```
int *d


d=malloc(3*sizeof(int))
```

として、int 型から派生したポインタ型変数 d を宣言し、その値として、malloc 関数で int 型変数 3 個分を確保したアドレスを代入すれば、配列と同様の表現 d[0], d[1], d[2] を使って int 型の値が参照できる。

つまり、ヒープ上に値を格納する領域を確保し、そのアドレスを返す関数、malloc を用いて、予め配列の大きさを定めてしまうことなく、動的に必要な大きさの配列を利用することができる。[*1]

課題 **3** ポインタを使い動的に配列を確保する方法を利用して、プログラム実行時に頂点数を柔軟に指定できるように課題 1 のプログラムを改良せよ。

また、改良されたプログラムにより、頂点数はどのくらいまで増やせるのか、理論的な限界を考察し、実際に確保できる限界を比較せよ。

さらに、データの格納方法をよく検討し、なるべく多くの頂点数を扱える方法を工夫せよ。

- ヒント

1. $n \times n$ 個の値を格納するためには、int graph[$n \times n$] として宣言された配列があれば良い。

   例えば、3 個の頂点 A, B, C からなるグラフの辺を考える。辺の両端点に、それぞれ 3 通りの頂点が選ばれる可能性があり、全体で $3 \times 3 = 9$ 通りの辺が存在し得る。そこで、この全ての頂点について、その存在ん真偽を変数に格納するならば、9 個の領域を確保すれば良いことになる。int v[3][3] 等として 2 次元配列を利用することも可能ではあるがより直接的には、次のようにポインタ型変数を 1 つ用いて必要な領域を確保することも考えられる。

   ```
   int *v, n;
   n = 3;
   v = malloc(n*n*sizeof(int));
   ```

   この方法であれば、 n の値をユーザが入力する等の仕組みを作って柔軟性を高めることも容易である。
2. 無向グラフであれば、$n \times n$ の配列の約半分は不要である。
3. ループした辺を認めないのであれば、対角成分は常に偽となる。

   前項の例では全ての組合せを認めているので、次の 9 通り全ての辺の存否について領域を確保することになってしまう。

   $$\{A,A\},\{A,B\},\{A,C\},\{B,A\},\{B,B\},\{B,C\},\{C,A\},\{C,B\},\{C,C\}.$$

   無向グラフを想定しているのであれば、例えば {A,B} と {B,A} の辺は一方だけが存在し、他方が存在しないということはあり得ず、両方の状態を別々に格納しておく必要は無い。また、ある点を出てまたその点に戻る自己ループにあたる辺を考える必要が無い場合も多い。上記の例であれば {A,A}、{B,B}、{C,C} がこれにあたる。これを併せて、自己ループの存在しない無向グラフのデータであれば、頂点数の 2 乗分ではなく、その半分以下の状態だけを格納しておけば全てのグラフの状態を表現できることが分かる。3 頂点の例で考えれば、次の 3 通りの辺の真偽だけを格納すれば良い。

   $$\{A,B\},\{A,C\},\{B,C\}.$$

[*1] 詳しくは malloc 関数のマニュアルページを参照すること。

1.4 最短経路検索と Dijkstra のアルゴリズム

前節までに取り扱ったグラフの各辺毎に固有の重みをかけたグラフを重みつきグラフと呼ぶ。図 1 のような路線図の各辺に、辺に対応する区間の運賃や所要時間、道のり等を重みとして付加すれば、さらに様々なモデルをグラフとして取り扱うことができるようになる。

重みつきグラフは、頂点の集合 V、辺の集合 E に加えて E の要素の関数 $w = w(e \mid e \in E)$ を併せたものである。例えば、図 2 は 9 つの頂点と 16 の辺からなる重みつき無向グラフである。

課題 **4** 課題 3 で作成したプログラムを改良し、重みつきグラフデータを入力するプログラムを作成せよ。

- ヒント

  配列 graph の値を辺の有無ではなく、辺の重みとすれば良い。辺の存在しない部分の値をどうするかを工夫せよ。

重みつきグラフ中の頂点を 2 つ選び、それぞれ s 点、t 点とする。s 点から出発して、次々と隣接する点へ各辺を辿って移動して t 点へ辿りつく経路を考えたとき、その方法は 1 通りとは限らない。経路のバリエーションがいく通りも考えられるとき、多くのバリエーションのうち、経路上の辺の重み全ての合計値が最も小さくなるようなものを最短経路と呼ぶ。

重みつきグラフと出発点 s 、終着点 t が与えられたとき、s から t への最短経路を求める問題を考える。重みつき無向グラフの最短経路を求める方法には Dijkstra のアルゴリズムと呼ばれるよく知られた方法が存在する。

以下では Dijkstra のアルゴリズムを図 2 で表されるグラフを用いて説明する。

Dijkstra のアルゴリズムの進行には、各点毎に付加される、s 点からその点までに辿る経路上の重み全ての和のうち最小のもの「最小距離」と、最小距離を実現する経路上の頂点リスト「最小経路」を用いる。各辺に付加される「重み」に対応して、このような各点に付加されるデータを「ラベル」と呼ぶ。このとき、以下のような手順で s 点から t 点への最短経路を求めることができる。

- Dijkstra のアルゴリズム

1. V を頂点、E を辺、w を辺の関数である重みとする。
2. V の部分集合 $V_1 = \{s\}$ とし、s に最小距離を示すラベル $l_s = 0$ を付加する。
3. V から V_1 を除いたものを V_2 とする。
4. E のうち、V_1 の要素だけを端点に持つものを E_1 とし、V_2 の要素だけを端点に持つものを E_2 とする。
5. E から E_1, E_2 を除いたものを B とする。
6. B の端点のうち、V_2 に属するものの順番に注目し v とする。
7. v の反対側の端点と、その最小距離ラベルを参照し、s から v までの最小距離 l_v と最小距離を与える経路上で v の直前の頂点 u を求める。
8. 最小の l_v を与える頂点を $v_{\min}$ とし、その直前の頂点を $u_{\min}$ とする。
9. $v_{\min}$ に最小距離を示すラベル $l_{v_{\min}}$ と直前の頂点 $u_{\min}$ を付加し、$v_{\min}$ を V_1 に加える。
10. t 点が V_1 に含まれていなければ、3 に戻る。
11. t 点に付加された最小距離のラベルが最小経路の距離となる。また、直前の頂点を示すラベルを t 点か

ら逆順に s 点まで辿れば最小経路を構成する頂点のリストになる。

課題 **5** 図 2 の頂点 1 を s 点、頂点 9 を t 点とする。s 点から t 点への最短経路を Dijkstra のアルゴリズムを用いて求めよ。(プログラムは必要ない。)

プログラムリスト 2(dijkstra.c) は inputgraph 関数を用いて読み込んだグラフデータにおいて頂点 1 を出発点とした場合の各点への最短経路を Dijkstra のアルゴリズムを用いて求めるプログラムの例である。

ただし、グラフデータを表す int 型配列 graph[][] はその大きさが固定で、値が各辺の重みを表し、辺が存在しない場合には int 型の最大値 INT_MAX を値に持つことを前提としている。また、出発点は頂点 1 に固定で、先に示した Dijkstrat のアルゴリズムの説明と異なり、V_2 が空になるまで反復を繰り返し、全ての点への最小距離と最短経路がその点のラベルとして求まるように構成されている。

プログラムリスト **2**

Dijkstra のアルゴリズムプログラム例 dijkstra.c

```
 1  #include <stdio.h>
 2  #include <stdlib.h>
 3  #include <limits.h>
 4
 5  #define NMAX 10                  /* 点の数の上限 */
 6  int graph[NMAX + 1][NMAX + 1];   /* 辺 (関連) の有無 */
 7  int n;                           /* 点の数 */
 8
 9  #define START  1  /* 出発点 */
10  #define FALSE  0
11  #define TRUE   1
12
13  void inputgraph(void);            /* グラフデータ読込み */
14
15  int main()
16  {
17    int i, j, next, min;
18    static char visited[NMAX + 1];
19    static int leastdistance[NMAX + 1], previous[NMAX + 1];
20
21    inputgraph();  /* グラフデータ graph[][] の読込み */
22    for (i = 1; i <= n; i++) {
23      visited[i] = FALSE;  leastdistance[i] = INT_MAX;
24    }
25    leastdistance[START] = 0;  next = START;
26    do {
27      i = next;  visited[i] = TRUE;  min = INT_MAX;
28      for (j = 1; j <= n; j++) {
29        if (visited[j]) continue;
30        if (graph[i][j] < INT_MAX &&
31    leastdistance[i] + graph[i][j] < leastdistance[j]) {
32  leastdistance[j] = leastdistance[i] + graph[i][j];
```



```
33  previous[j] = i;
34        }
35        if (leastdistance[j] < min) {
36  min = leastdistance[j];  next = j;
37        }
38      }
39    } while (min < INT_MAX);
40    printf("点  直前の点  最短距離\n");
41    for (i = 1; i <= n; i++)
42      if (i != START && visited[i])
43        printf("%2d%10d%10d\n", i, previous[i], leastdistance[i]);
44    return EXIT_SUCCESS;
45  }
```

課題 **6** プログラムリスト 2 を変更して、次の仕様を満たすように改良せよ。

1. 課題 4 で作成したプログラムに対応して、実行時に頂点数を指定できる。
2. 出発点 s と終着点 t を指定して最短経路上の経由頂点リストを出力できる。

2 オンライン授業 1 回目まで進めるべき範囲について

本来、実験の第 1 日目は少なくとも課題 1、2 を完了し、課題 3 への取り組みを開始している必要があります。オンライン授業の 1 回目までにそれが実現するために必要なことを明確にし、疑問点等について質問できるようにしてください。

3 オンライン授業 2 回目以降までに進めるべき範囲について

本来、実験の第 2 日目では課題 1 から 6 までを一通り完了することが期待されます。少なくとも「離散シミュレーション実験」の課題のうちに手をつけていないもの、何をするべきか分かっていないものが無いようにしなければいけません。オンライン授業 2 回目までに、疑問点を解消するために必要な質問をまとめ、送信しておいてください。

4 備考

テキスト等に問題があればご指摘ください。問題点は適宜修正し、告知いたします。

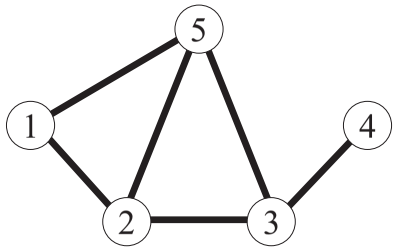


図 1　無向グラフ

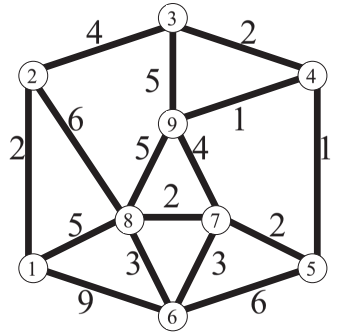


図 2　重みつき無向グラフ

情報工学実験 II 手順書

ソフトウェア課題
「連続系のシミュレーション (常微分方程式の数値計算)」

version 1.0

愛媛大学工学部情報工学科

2020 年度

第 1 章

はじめに

本実験では常微分方程式の数値計算を C 言語によるプログラムを用いて行う。古くより弾道計算、電気回路の過渡応答解析、近年では宇宙ロケットの軌道計算などなど、常微分方程式の初期値問題に頼られる実際問題は数知れない。そして、それらの問題のほとんどは、解析的に解くことが不可能あるいは困難であり、数値解法に訴えざるをえないものである。

1.1 準備

まず「予習課題」の節で示され課題を実施し、必要な知識を身に付けてから課題に取り組むこと。

1.2 実験

本来の本実験では、第 1 週目で第 2.2 節まで完了し、課題 1・2 の結果を得ることを想定している。同様に第 2 週目では、課題 3・4 の結果を得ることを想定している。予習課題と課題 1・2 を第 1 回目のオンライン授業で完了させる、あるいはその目処をつける必要がある。また、同様に課題 3・4 を第 2 回目のオンライン授業で完了させ、レポートの作成に進めるよう済ませなければならない。

第 2 章

常微分方程式の数値計算

常微分方程式の初期値問題の数値解法というのは、数値積分法の応用であり、まずは、数値積分法を十分理解する必要がある。

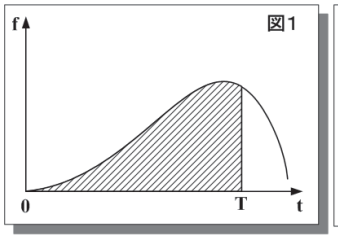
2.1 $dx/dt = ax$ の数値積分法 (一次精度計算)

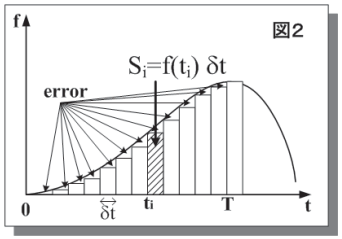
まず、微分方程式

$$\frac{dx}{dt} = f(x) \tag{2.1}$$

を考える。これを解いて x の時間変化 $x(t)$ を求めるには、$x = \int f(x)dt$ の積分を計算すればよい。

コンピューターで積分 (数値積分) するための基本原理は簡単である。
積分 $\int_0^T f(x)dt$ は t 対 $f(x)$ のグラフにおいて、図 1 斜線部の面積を求める事に他ならない。そこで図 2 の様に t 軸方向に分割して小さな四角形を作り、これらの面積の総和を求めればよい。この時、小さく分割するとより正確な面積が求まるが計算量は増え、逆に大きく分割すると粗い近似値しか得られない。





$x(T)$ を求める計算を式で表すと、

$$\begin{aligned} x(T) - x(0) &= \int_{t=0}^{T} f(t)dt \\ &\simeq f(0)\delta t + f(\delta t)\delta t + f(2\delta t)\delta t + \ldots\ldots + f((n-1)\delta t)\delta t \\ &= \sum_{i=0}^{n-1} f(t_i)\delta t \\ &= x(t_n) - x(t_0) \end{aligned} \tag{2.2}$$

となる。ただし、$T = n\delta t = t_n$ とした。きざみ δt は分割の幅を表している。任意時刻 T の $x(T)$ を知るには n-1 回の足し算が必要になる。

ここで $x(t_1), x(t_2), ..., x(t_n)$ を次々と効率良く求めるには、この式を少し変形した方がよい。具体的には式 (1.2) 中のすべての n を n-1 とした式と式 (1.2) を引き算して、

$$x(t_n) - x(t_{n-1}) = f(t_{n-1})\delta t \tag{2.3}$$

とする。すると $x(t_1)$ は $x(t_1) = x(t_0) + f(t_0)\delta t$ から求まり、$x(t_2)$ は $x(t_2) = x(t_1) + f(t_1)\delta t$ というように $x(t_i)$ は次々求まる。ちなみに式 (1.3) を δt で割ると、δt が十分小さい極限でそれは微分に関する『平均値の定理』を意味している。

以上の内容をプログラムで実現すると”dxdt.c”(4 ページ) のようになる。このプログラムでは、$f(x) = a * x$,$\delta t = dt$ になっている。初めに積分定数として $x(t_0)$ を与えなければならない事に注意せよ (このプログラムでは、適当 ($x0 = 1.0$) に与えてある)。
コンパイルコマンドおよびオプションは、プログラムの始めに書いてある通りである。

> コラム (コンパイルオプション)
> ここで触れているコンパイルオプション"-lm"は、"libm.a"ライブラリをコンパイル時にリンクすることを意味します。
> この"libm.a"というのは数学関数のためのライブラリです。つまり、プログラム中で数学関数を使う場合には、こういったライブラリをリンクしてください。

[予習課題 1]

$\frac{dx}{dt} = ax$ の一般解を求めよ。
また、$a = -1$ とし、初期条件として、$t = 0$ において $x = 1$ とした時の解を求めよ。

[課題 1]

サンプルプログラム dxdt.c をコンパイル・計算実行せよ。
$t = 1.0$ の時点での厳密解 x(1.0) との誤差を dt との関係で評価せよ。
つまり、dt の値を変更し、dt と誤差 ($t = 1$) の関係を調べよ。
(注意、$t = n * dt$ なので、$t = 1.0$ まで計算する時に dt を小さく (大きく) すれば n を大きく (小さく) しなければならない。サンプルプログラムでは、すでに dt から n を計算するようになっている。)
レポートでは、dt と誤差の関係をグラフで表すこと。


```
/*-------------------------------------------------------------*/
/*                    File Name: dxdt.c                        */
/*                compile: gcc dxdt.c -lm                      */
/*-------------------------------------------------------------*/

#include <stdio.h>
#include <stdlib.h>
#include <math.h>

/* 数値積分法 1 */

main()
{
   int i, n;
   float t, dt, x0, x1, a;

   /* 初期値設定 */
   t  = 0.0; /* 初期時間 */
   x0 = 1.0; /* 初期位置 */
   a  = -1.0; /* 定数 */

   /* 時間刻みの読み込み */
   printf("INPUT DATA dt ==> ");
   scanf("%f",&dt);

   /* 分割数 n の決定 */
   n=(int)(1.0/dt);

   printf("00\t,t=%f\t,x(t)=%f\n", t, x0);

   for(i=0;i<=n;i++)
   {
    x1 = x0 + a*x0*dt;
    t  = t + dt;

    printf("%02d\t,t=%f\t,x(t)=%f\n", i+1, t, x1);

    x0 = x1;
   }

   printf("THE END\n");
}
```

2.2　$dv/dt = ax$ の数値積分法 (一次精度計算)

次に、微分方程式 $d^2x/dt^2 = f(x)$ を考えよう。
これは力 f(x) のもとでの質点の運動方程式である。変形すると

$$\begin{aligned} dv/dt &= f(x) = ax \\ dx/dt &= v \end{aligned} \tag{2.4}$$

と書ける。2.1 と同様にして、これらそれぞれを同時に解けばよい。

[予習課題 2]

$\frac{d^2x}{dt^2} = ax$ の一般解を求めよ。
また、$a = -1$ とし、初期条件として、$t = 0$ において $x = 1, v = 0$ とした時の解を求めよ。

[課題 2]

未完成のプログラムを次ページに示す。$f(x) = a * x$ の場合についてサンプルプログラム (dvdt.c) をコピーして修正し (****の部分を直し)、完成させよ。また、完成していることを示す結果を提示し、説明しなさい。


```
/*-------------------------------------------------------------*/
/*                    File Name: dvdt.c                        */
/*                  compile: gcc dvdt.c -lm                    */
/*-------------------------------------------------------------*/

#include "stdio.h"
#include "stdlib.h"
#include "math.h"
/* 数値積分法２ */
main()
{
        int   i,n;
        float t,dt,x0,x1,v0,v1,a;

        t = 0.0;
        x0= 1.0;
        v0= 0.0;
        a= -1.0;

        printf("INPUT DATA dt ? \n");
        scanf("%f",&dt);

        n = (int)(1.0/dt);

        printf("00\tt= %f\tx(t)= %f\tv(t)= %f\n",t ,x0 ,v0);

        for(i =0 ; i<=n ; ++i)
        {
                  x1 = ****;
                  v1 = ****;
                  t = t + dt;
                  printf("%02d\tt= %f\tx(t)= %f\tv(t)= %f\n",i+1,t,x1,v1);
                  x0 = x1;
                  v0 = ****;
        }
        printf("THE  END  \n");
}
```

2.3　二天体シミュレーション

二連星や地球と人工衛星の場合など、二天体の運動をシミュレーションしよう。二次元の場合、一般的な運動方程式は、

$$md^2x/dt^2 = F_x, \quad md^2y/dt^2 = F_y \tag{2.5}$$

である。(F_x, F_y) を天体に加わる万有引力とし、m_1, m_2 を天体の質量とする。天体 m_1 については、dvdt.c と同様に、これを

$$m_1 dV_x/dt = F_x, \quad m_1 dV_y/dt = F_y \tag{2.6a}$$

$$dx/dt = V_x, \quad dy/dt = V_y \tag{2.6b}$$

と書き直す。
天体 m_1, m_2 の位置をそれぞれ $(x_1, y_1, z_1), (x_2, y_2, z_2)$ とすると天体 m_1 に働く万有引力は、

$$(F_{x1}, F_{y1}) = (-Gm_1m_2r_x/|r|^3, -Gm_1m_2r_y/|r|^3) \tag{2.7}$$

ただし、$r_x = (x_1 - x_2)$, $r_y = (y_1 - y_2)$, $r = (r_x^2 + r_y^2)^{1/2}$ である。つまり r は 2 天体間の距離である。
一方、m_2 の方は（作用反作用の法則により）符号を逆にしただけの $(F_{x2}, F_{y2}) = (Gm_1m_2r_x/|r|^3,\ Gm_1m_2r_y/|r|^3)$ とすればよい。

[予習課題 3]

C 言語による、外部ファイルへの出力方法について調べよ。

[課題 3]

以下の未完成プログラムを完成させ、gnuplot で天体の軌道を作図せよ。また、このプログラムでは、標準出力へ結果を出力している。この結果を、外部ファイルへ出力するように変更しなさい。これまで同様、ファイルは、授業情報のウェブサイトからダウンロードしたものを利用せよ。レポートでは、初期条件等のシミュレーション条件を必ず記述すること。また、この tentai.c の条件では二天体は同一円軌道を描くはずである。同一円軌道にならない場合は、ならない理由を示せ。レポートでは、文章だけでなく、グラフを用いて示すこと。


```
/*-------------------------------------------------------------*/
/*             File Name: tentai.c                             */
/*             compile  : gcc tentai.c -lm                     */
/*-------------------------------------------------------------*/
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

/* 二天体シミュレーション */

/* 万有引力を計算 */
void force(x1, y1, x2, y2, mmg, fx, fy)
float x1,y1,x2,y2,mmg,*fx,*fy;
{
    float    r, xx, yy;

    xx = x1-x2;
    yy = y1-y2;

    r  = sqrt(xx*xx + yy*yy);

    *fx = -mmg*xx/(r*r*r);
    *fy = -mmg*yy/(r*r*r);
}

main()
{
    int    i, j, n;
    float    t, dt, g, fx1, fy1, m1, m2;
    float    x0[2], x1[2], vx0[2], vx1[2];
    float    y0[2], y1[2], vy0[2], vy1[2];
    /* [0]:天体1 , [1]:天体2 */

    n  = 200;    /* 時間ステップ数 */
    t  = 0.0;
    dt = 0.05;    /* 時間刻み */
    g  = 4.0;    /* 重力定数 */
    m1 = 1.0; m2 = 1.0;    /* 天体質量 */

    /* 初期位置の設定 */
    x0[0]  =  1.0;  y0[0] = 0.0;
    x0[1]  = -1.0;  y0[1] = 0.0;
    /* 初期速度の設定 */
    vx0[0] =  0.0; vy0[0] =  1.0;
    vx0[1] =  0.0; vy0[1] = -1.0;

    for(i=0;i<n;i++)
    {
        force(x0[0],y0[0],x0[1],y0[1],m1*m2*g,&fx1,&fy1);

        /* dt 後の各天体の位置と速度を求める */
        x1[0] =
        vx1[0]=
        y1[0] =
        vy1[0]=
        x1[1] =
        vx1[1]=
        y1[1] =
        vy1[1]=

        printf("%f\t%f\t%f\t%f\t%f\n", t, x1[0], y1[0], x1[1], y1[1]);

        for(j=0;j<2;j++)
        {
            x0[j]  =
            vx0[j] =
            y0[j]  =
            vy0[j] =
        }

        t  = t + dt;
    }
}
```

[課題 4]

標準入力等から初期条件 (位置、速度等) を変更できるようにプログラムを修正し、つまり、初期条件を変更するたびにコンパイルしなおす必要がないようにし、さまざまな初期条件により天体の軌道をシミュレーションせよ。
2 つ以上の条件でシミュレーションし、その結果をレポートにすること。また、レポートでは、それぞれの初期条件の設定根拠も同時に示すこと。

[課題 5：発展課題]

シミュレーション対象の天体の数を柔軟に増やせるように課題 4 で作成したプログラムを改良し、適切なパラメタのもとでシミュレーションを実行した結果を示せ。少なくとも、天体の数が 3、6、10 の場合を調べ、(1) シミュレーション結果がどの程度信頼できる（正しい or 誤差が少ない）と言えるのか、(2) シミュレーション結果からどのようなことが言えるのかを示せ。

[課題 6：発展課題]

これまで出てきた以外の微分方程式を数値的に解くプログラムを作成せよ。
落下運動や空気抵抗を考慮した落下運動。減衰振動など、どんな問題でも構わない。

1　離散・連続シミュレーション実験レポート提出の注意事項

2020 年 4 月 20 日

「離散シミュレーション実験」「連続シミュレーション実験」のレポートはそれぞれを 1 つにまとめて提出してください。提出期限は 2 つの実験の最終日から 1 週間後です。2020 年度（令和 2 年度）の場合は 5 月 29 日となります。

課題の進捗は自身の進め易いようにして構いませんが、オンライン授業では第 1-2 回目を離散シミュレーション実験に、第 3-4 回目を連続シミュレーション実験に割り当てて進行を予定しています。また、第 5 回目は全ての課題を一通り完了した状態で残された問題を解決する、あるいはレポート作成のために解決すべき問題を解決することを主な目標にして実施される予定です。オンライン授業をうまく利用できるように計画を立てて進めてください。

なお、あらかじめ強調しておきますが、合格点を得るための絶対条件は全ての課題を完了させること、そしてそのことを検証できるレポートを提出することです。課題が 1 つでも完了していない状態で合格になることはありません。また、課題の完了はレポートで明確に示されていなければなりません。何の説明もデータも示されていない課題があれば、その課題は未完了です。課題が完了しているかどうかを示すのはレポート提出者です。完了していることを示す方法はレポート中で説明し検証することです。以上のことを忘れずに課題に取り組んでください。

1.1　提出するもの

現時点で利用可能であることが決まっている「離散シミュレーション実験」「連続シミュレーション実験」のレポート提出方法は電子メールへの添付ファイルによるものです。遠隔授業での実施に伴い、別の手段も提供すべく準備しています。準備が完了したら、その説明もします。また、レポート提出の際に困難があれば、その状況を具体的に説明して質問してください。

電子メールへの添付ファイルによりレポートを提出する場合、以下の規定を満たしてください。

- レポート提出先のメールアドレスは以下の通りです。
  `simulation@comp.cs.ehime-u.ac.jp`
- レポート提出メールの Subject: は以下のようにしてください。
  e18xxABCD 情報工学実験 2 レポート
  ※「e18xxABCD」の部分を情報工学科教育用計算機システムにおける、提出者のユーザ ID に置き換えてください。
- 送信者のメールアドレスを学科提供のものにしてください。
  送信を学内から行なえという意味でも、学内から行なえば良いという意味でもありません。学外からの送信の場合も送信者のアドレスを以下の形式のものに変えて送信してください。
  `e18xxABCD@cs.ehime-u.ac.jp`
  また、レポート提出者と送信者は一致させてください。代理提出は認められません。
  学外からの送信が必要でかつ送信元のアドレスを指示通りに変更することがどうしてもできない場合は、メディアセンターのアカウントを利用して送信してください。
- 2 つの実験のレポートをそれぞれ 1 つの PDF ファイルにまとめ、2 つの PDF ファイルを添付してください。ファイル名は以下のようにしてください。
  離散シミュレーション実験： e18xxABCD-risan.pdf
  連続シミュレーション実験： e18xxABCD-renzoku.pdf
  PDF ファイルには閲覧・印刷・改変等の制限は一切つけないでください。
  なお、LaTeX / Microsoft オフィスを使った場合は学科計算機室で PDF ファイルへの変換が可能です。情報工学科の学生は Microsoft オフィスおよび関連ソフトウェアを追加の利用料金なしでインストールする権利を有しています。LibreOffice 等の Openoffice.org とその互換ソフトは PDF ファイルへの出力機能を持っています。
- 2 つの実験で最終的に作成することのできたプログラムのソースファイルを添付してください。離散シミュレーション実験では課題 6 に用いたプログラム、連続シミュレーション実験では課題 4 および発展課題に用いたプログラムが対象です。ファイルの件数に関わらず、それぞれの実験のファイルを一つの圧縮書庫ファイルにまとめてください。ファイル名は以下のようにしてください。
  離散シミュレーション実験： e18xxABCD-risan.tgz
  連続シミュレーション実験： e18xxABCD-renzoku.tgz
  上記以外のファイル名を使わないようにしてください。したがって、圧縮書庫ファイルの形式は tar 形式を gz 形式で圧縮したものに限ります。文字コードの扱いに問題の多い zip ファイル等の形式は使わないでください。
- 電子メール本文に提出者と添付ファイルの情報を呈示してください。
  とくに事情の無い限り、以下の様式を修正し電子メール本文に含めてください。

```
「離散シミュレーション実験」「連続シミュレーション実験」課題提出
提出者氏名：○○ ××
学籍番号：  012345678X

添付ファイル一覧
レポート本文: e18xxABCD-risan.pdf, e18xxABCD-renzoku.pdf
作成したプログラムソースファイル
離散シミュレーション実験 (e18xxABCD-risan.tgz に格納)
e18xxABCD-risan.c, e18xxABCD-risan-inputgrah.c
連続シミュレーション実験 (e18xxABCD-renzoku.tgz に格納)
e18xxABCD-renzoku.c
～以下省略～
```

1.2　レポートの内容

授業時間中の実験だけでなく、レポートの作成も学習・教育の重要な機会です。十分に注力してください。

■離散シミュレーション実験について　レポートを書き始める前に、テキスト全体を読み直してください。それぞれの実験は課題を進めることにより、最後の課題で実施するプログラムの作成や、実行に必要な準備ができるようになっています。テキストに著された筋道をよく理解して、筋道に沿った説明をしてください。

離散シミュレーション実験では、課題 6 で作成する最短経路探索プログラムの作成に必要な要素を一つずつ完成するように課題が並べられています。以下に課題の意図について簡単な説明をしますので参考にしてくだ

さい。

課題 1・2 では、グラフデータをプログラム中で扱うためのデータ構造とそのためのプログラムコードの作成、動作確認が実施されます。レポートでは、ここで採用しているデータ構造とその入出力の仕組みを説明し、実際に説明した通りのプログラムが完成していることを具体的に示す必要があります。

課題 3 では、グラフデータの入力に際して「グラフの頂点数を自由に定められる」ことを最終的なプログラムの要件と捉えて、そのための方法について検討します。テキストにヒントの提示はありますが、具体的な実現方法は各自に任せられていますので、採用するアイディアや、それを実現するプログラムコードを作成します。方法と実装（プログラムコード）について検討し決定したのですから、その両方を評価して説明する必要があります。採用した方法において、どのようにデータを扱うことで、どのような仕様上の制限が生じているのか、実際の計算機資源の元での実装に適した仕様であるのか評価しなくてはなりません。作成したプログラムで確保できる、あるいは利用できるグラフの頂点数の限界がプログラムの評価において重要になることから、採用した方法に由来する制限が現実的に確保できる頂点数に比して十分に大きいのであれば問題は無いことになるのは理解できるでしょう。評価の視点は様々なものが考えられます。自分にとって重要だと思える視点から論理的な説明をしてください。

■2 つの独立したレポートを作成してください。　レポートは 2 つの実験それぞれについてまとめた 2 つの文書となります。適切に体裁を整え、それぞれが区別できるようにしてください。2 つの文書にそれぞれ表紙・表題をつける等が考えられます。

「離散シミュレーション実験」と「連続シミュレーション実験」を独立して評価できるように、それぞれで完結したレポートにしてください。一方のレポートを理解するためにもう一方のある部分に書かれている内容を参照する必要がある、あるいはその逆のような状況が無いようにしてください。

同様に, レポートの説明を理解するために実験テキストや他の資料を参照する必要が無いようにしてください。例えば、課題の内容について説明の無いまま、その結果や、回答のみを示さないでください。必要な記述は全てレポート中に含めてください。

■「図表」を文章で説明してください。　ここでは、レポート中に示される本文以外の文書や図、表等を全て含めて「図表」と呼びます。図表にはプログラムリストやプログラムの実行結果、テキスト等から引用された文章を含みます。

こうした図表について、必ず本文中で十分な説明をしてください。本文で説明をしない図表は削除してください。図表の説明では、その出自と意図を明らかにしてください。

書籍や資料から引用した図表には、引用元を特定できる分かり易い情報を付記してください。書籍であれば、著者・書名・出版社・引用箇所の情報を文中で述べるか、傍注・脚注に注記する、文末にまとめて提示します。提示方法は各自で最も適切なものを選んでください。ただし、1 つのレポート中で情報提示の仕方がバラバラになることは避けてください。実験テキストからの引用の場合も同様に引用元としてテキストを挙げ情報を示してください。引用でない図表は全て、何らかの方法で自ら作成したものということになります。その出自について責任を持って説明できるようにしてください。

引用の場合も、そうでない場合も、図表は本文の説明を補強するものです。文章の代りに図表を用いるのではなく、図表を見ることで文章が明確になり、分かり易くなるようにしてください。また、図表から読み取ることのできる情報は様々です。同じ図表を示したからといって誰もが同じ部分に注目し、同じことを連想するわけではありません。図表から何を読み取って、どのような判断をするべきなのか、文章で明確に示してください。例えば「結果は図×の通り」「プログラムリスト○を作成した」という文は、その図表が何を意図したものか、呈示によってどのようなことを主張しているのか理解することはできません。「図 x の項目 y と z を

比較すれば、計算時間が…」というように具体的に、曖昧な部分の無いように説明してください。

以下の 3 点の欠けた図表は示す意味がありません。減点対象であり、レポートは受理されません。

1. 引用資料が何であるか
2. 引用資料のどの部分からどのような情報が読み取れるのか
3. 引用資料から読み取れる情報からどのような結論が得られるのか

■添付ファイルと本文中の引用を区別してください。　課題で、プログラム作成が要求されている場合は完成したプログラムのソースファイルを添付して提出してください。これはコンパイル等の確認をするためのものであり、レポート本文中の引用とは別のものです。

レポート本文中の説明でプログラムリストを参照する必要がある場合は、添付ファイルを参照するのではなく、レポート中に必要箇所を引用してください。また、引用の際は、引用資料として十分な説明を付与してください。逆にレポート本文中にプログラムリストの一部または全部を引用した場合でも、これを添付ファイルの代わりとすることはできません。ソースファイルが必要な場合はレポート本文とは別のファイルとして提出してください。

1.3　レポート評価基準

コミュニケーション能力の養成が情報工学実験 2 の学習・教育目標の一つです。離散・連続シミュレーション実験では実験レポートの作成によりこの目標を達成します。授業時間中に十分な実習を済ませていて、質問等で疑問を解決していれば、十分な準備ができています。実験の内容を正しく説明できれば全員がレポートを受理されるはずです。それでも、再提出が必要になるのは、以下のような問題があった場合です。

- レポートの説明に欠落や間違いがある。
  結論や結論に至る説明に間違いがある場合、それを修正する必要があります。また、比較的多いのが、何故その結論に至るのか説明が無いこと、根拠となる事実や情報が示されていないことです。
  実験時間中に TA の方々や教員の説明することは、作業を進め、自ら学習するためのヒントです。与えられたヒントやキーワードをもとに、調査・学習をした成果をレポートで表現してください。
- 必須課題が完了していない
  必須課題は全て完了していなくてはなりません。課題の要求の一部でも満たさない場合は、レポートは受理されません。未完了の必須課題がある場合は、自習・補習により完成させてください。
- 提出物が揃わない
  提出物が全て揃うまではレポートは受理されません。離散シミュレーション実験、連続シミュレーション実験のレポートを両方完成させて提出してください。片方ずつバラバラに提出することはしないでください。
- 文章表現に問題がある
  内容を理解するのが困難な文章は、譬えその意図が正しくてもレポートを受理できません。
  次のような問題が多数見受けられますので、参考にしてください。
  - 説明の文章が不十分もしくは存在しない。
  - 文法的間違いや、論理的矛盾により文意が確定しない。
  - 図表の提示のみで文章による説明が無い。
    C の関数等のプログラムを構成する要素だけでは「文章」になりません。箇条書のリストも「文

章」ではありません。プログラム、外の文書からの引用は図表として提示し、その内容を別途文章で説明してください。

- 誤字・脱字、言葉や記号の用法に間違いがある
  誤字脱字、引用符が対応していない、形式段落の字下げが正しくない、句読点の使い方がおかしい、C の関数名や変数名の大文字・小文字の間違いなどが多いようです。
- 文体の不統一、語法の揺れ・乱れがある
  「です・ます調」「だ・である調」の混在、箇条書等以外の場所での体現止めの乱用を避け、仮名漢字使い・送り仮名のレポート中の用法は統一してください。また、同じ事物を別の表現で表したり、別の事物を同じ表現で表したりすることの無いようにしてください。

- 必要な情報が示されていない
  レポート作成者の知識ではなく、レポート読者の知識に沿って説明してください。書く人だけが知っていることを前提として説明してはいけません。具体的には、一般常識と、実際にレポート中に記述のあることだけで、全てを理解できるように説明してください。
  例えば、作成したプログラムのテスト環境にどのようなサーバがあるか等、実験室の環境についての知識、サンプルプログラムの内容等、テキストに書いてあること、実験時間中に教員・TA の説明したこと、は全て知られていないものとしてレポートを書いてください。
- 章・節・項目の題名と、その内容が一致しない。
  「方法」には方法、「原理」には原理を示してください。「結果と考察」とするなら、結果とそれに対応する考察を適切な方法で著してください。とくに誤りが多いのが「方法」と称した部分の記述内容です。
  離散・連続シミュレーション実験のレポートにおいて「方法」という表題に対応するのは『課題の要求を実現するための』方法です。『課題の要求を実現するための』方法と関係の無い「方法」を表題を立てて説明する必要はありません。
  頻出する誤りを以下に例示します。自分のレポートに同じような部分が無いことを確認してください。
  - 「目的」にレポートや実験ではなく授業の目的が書かれている。
  - 「方法」に実験時間中の作業内容が書かれている。
    システムコール・ライブラリコール関数の仕様が書かれているが、それをどのように利用して課題の要求を実現したのかは書かれていない。
  - 「原理」に C プログラムのコンパイル方法・リンク方法・実行方法が書かれている。
  - 「結果」にプログラムリスト等の図表しかない。
  - 「考察」の主張を直前までに示した情報や事実だけからは導くことができない。感想が書いてある。
  - 「参考文献」に引用元でない文献が掲載されている。(詳細は「不必要な情報が示されている」)
- プログラムリストの引用とファイル提出を混同している。
  電子メールに添付するソースファイルは動作確認等を行うためのものです。レポート中で、説明のために引用されたプログラムリストでは動作確認等を行うことができません。逆に、添付されたソースファイルには、その内容に関する適切な情報が提示されていません。したがって、説明のための引用に添付ファイルを利用することもできません。
- 不必要な情報が示されている
  サンプルプログラムの入手方法、コンパイル方法の説明は不要です。参考文献にはレポートを読む際に参照すべき資料をその理由が分かるように示してください。レポートを書く際に提出者が参考にした資料は不要です。テキストを書き写す必要はありません。