Fortran 演習 レポート課題

▷ 以下の課題から **3 題以上** を選択し，レポートにまとめて提出せよ．
▷ 提出期限は 2023 年 5 月 28 日 (日) の 23:59 JST(日本標準時) とする．
▷ 提出は `zip` もしくは `tar.gz` 形式のファイルを ITC-LMS にアップロードすること．
▷ 質問・相談は随時受け付ける．

レポート作成についての注意点（従わない場合は減点の可能性あり！）

▷ ChatGPT などのツールの使用は許可する．ただし，どのように使用したか明記すること．特に有用と思われる場合には ChatGPT とのやり取りを https://sharegpt.com/ などを使って分かるように提出することを推奨する．
▷ レポートには図表を用いるなど，分かり易くする工夫を十分にすること．必要と思われる場合には用いたアルゴリズムや，ソースコードの簡単な説明，実装上の工夫などについても述べよ．
▷ レポートは単一の PDF ファイルおよび各課題に用いたソースコードを `zip` または `tar.gz` 形式で圧縮して提出すること．
▷ 圧縮ファイルを解凍したときに `s2326xx_fortran` （`2326xx` は学生証番号）というディレクトリができるようにすること．これには `s2326xx_fortran` というディレクトリを作成して，そこに全てのファイルを配置してから圧縮すればよい．
▷ ソースコードは適切な字下やコメントの挿入などによって，他人にとって読み易くなっているものが好ましい．
▷ 特に必要がない限りはソースコードはレポート本文に貼り付けなくてよい．
▷ いくつかの課題ではレポート課題に取り組むにあたってこちらが用意したファイルが必要になるので，まずは必要なファイルを手元にコピーしておくこと．例えば自分のホームディレクトリ直下の `report` というディレクトリにコピーするには

```
1: $ cd ~/report
2: $ wget https://amanotk.github.io/fortran-resume-public/report/files.tar.gz
3: $ tar -zxvf files.tar.gz
```

とすれば`~/report/files` 以下に必要なファイルが展開される．

課題　感想

Fortran 演習およびレポート課題など全般に対する感想を述べよ．(分かったこと，分からなかったこと，資料についてのコメントなどなど．内容については問わないので何でも正直に．)

課題　多倍長演算

通常の浮動小数点数 (実数) の精度には限りがあり，任意精度で近似値を求めたい場合には用いることが出来ず，多倍長演算 (または任意精度演算) と呼ばれる手法を用いることになる．ここでは多倍長演算が用いられる代表例である円周率 π の計算を考えよう．

以下では最も簡単な実装を考える．まず任意の数値 x を 10 進数で

$$x = (-1)^s \sum_{j=1}^{N} \tilde{x}_j \times 10^{-j+e}$$

の形で表そう．ここで $\tilde{x}_j \in \{0,1,2,3,4,5,6,7,8,9\}$ は各桁の数値を，$s \in \{0,1\}$ は符号を，$e \in \mathcal{Z}$ は値の絶対値を保持するための整数である．従って項数 N を十分大きく取り，$\tilde{x}_j$ を配列として保持すれば，メモリの許す限り任意の精度 (桁数) で数値を表すことが出来る．例えば $x = 1.41$ を表すには $N \geq 3$ とし，$\tilde{x}_1 = 1, \tilde{x}_2 = 4, \tilde{x}_3 = 1$ ($\tilde{x}_4$ 以降は全て 0)，$s = 0, e = 1$ とすれば良い．なお以下の円周率の計算では $s = 0, e = 1$ の場合が扱えれば十分であるので常にこれを仮定しても良い．すなわち，小数点以下 N 桁までの数値は長さ N の整数配列で表すことが出来る．(整数部まで含めれば長さは $N + 1$ である．)

ここでは円周率の計算に，Machin の公式および $\tan^{-1}(x)$ の Taylor 展開を用いることによって得られる以下の級数展開を用いよう．

$$\pi = \sum_{k=0}^{\infty} \frac{(-1)^k}{2k+1} (a_k - b_k) = (a_0 - b_0) - \frac{a_1 - b_1}{3} + \frac{a_2 - b_2}{5} - \frac{a_3 - b_3}{7} \cdots$$

ただし

$$a_k \equiv 16 \left(\frac{1}{5}\right)^{2k+1} \quad b_k \equiv 4 \left(\frac{1}{239}\right)^{2k+1}$$

と置いた．簡単な評価から，小数点以下 l 桁目まで (ただし $l \gg 1$) を正しく求めるには $k = [l/2\log_{10}(5)]$ 程度までで上記の級数を打ち切って良いことが分かる．

ここで，$a_{k+1} = a_k/5^2$，$b_{k+1} = b_k/239^2$ で与えられることに注意すれば，

- ▷ 多倍長数同士の加算 (足し算)
- ▷ 多倍長数同士の減算 (引き算)
- ▷ 多倍長数の整数 (通常の `integer`) での除算 (割り算)

が実装出来れば円周率の計算が出来ることが分かる．筆算の要領でこれらの演算を実装し，実際に円周率の多倍長計算を行うプログラムを作成せよ．以下は小数点以下 100 桁までを求めた結果である．

```
1: $ ./a.out 100
2: PI = 3.1415926535 8979323846 2643383279 5028841971 6939937510
3:        5820974944 5923078164 0628620899 8628034825 3421170679
```

この例ではコマンドラインで求める桁数 (この場合は 100) を与えている．このようにコマンドライン引数を利用するには，サブルーチン `get_command_argument` (Fortran 2003 以降では標準) を利用すると良い．ただし，必ずしもこの例のように実装する必要は無く，標準入力から項数を読み込むようにしても良い．

なお結果の確認用プログラムとして `kadai2.py` を用意してある．例えば小数点以下 10000 桁までを求めるには以下のように実行すれば良い．(`mpmath` モジュールがない人は適宜インストールしよう．)

```
1: $ python ./kadai2.py 10000
2: PI = 3.1415926535 8979323846 2643383279 5028841971 6939937510
3:        5820974944 5923078164 0628620899 8628034825 3421170679
4: # ... 中略 ...
5:        2645600162 3742880210 9276457931 0657922955 2498872758
6:        4610126483 6999892256 9596881592 0560010165 5256375678
```

課題　太陽風 (求根法)

太陽風の基本的性質は定常で 1 次元球対称の圧縮性流体方程式で記述される (Parker の太陽風理論)．このモデルは一般の恒星風や，惑星大気の流体力学的散逸を理解するにあたって基礎となっている．ここで，ポリトロピックな状態方程式 $p \propto \rho^\gamma$ を採用すると，太陽風の速度 u の距離 x に対する依存性は以下の方程式の解として与えられる．

$$u^{\gamma+1} - \left(\frac{4}{x} + \epsilon\right) u^{\gamma-1} + \frac{2}{\gamma-1} x^{2-2\gamma} = 0.$$

ただし，$x \equiv r/r_c$ は臨界点の距離 r_c で規格化した太陽中心からの距離，$u = v/a_c$ は臨界点での音速 a_c で規格化した速度，$\epsilon = (5-3\gamma)/(\gamma-1)$ は規格化されたエネルギーに対応する保存量である．なお，γ は比熱比で，太陽風に対応する物理的な解を持つためには $1 < \gamma < 5/3$ の範囲になければならない．この時，$x \to \infty$ の極限を取ると，無限遠での速度が $u_\infty = \sqrt{\epsilon}$ となる．(物理的内容は理解していなくても出来るようになっているので細かいことは考えなくてよい．)

この方程式を数値的に解くプログラムを作成し，速度 u および，局所的な音速 $a = x^{1-\gamma} u^{(1-\gamma)/2}$ で定義されるマッハ数 $M = u/a$ を距離の関数として図示せよ．無限遠での速度が $\sqrt{\epsilon}$ に漸近することを確かめること．ただし，太陽風に対応する物理的な解 ($x < 1$ で亜音速 $M < 1$， 臨界点 $(x, u) = (1, 1)$ を通り，$x > 1$ で超音速 $M > 1$ となる解) を求めること (それ以外にも解が存在する)． 図示する際には横軸 x はログスケールとし，太陽近傍 $x \ll 1$ から十分遠方 $x \gg 1$ までの依存性が分かるようにすること．また，いくつかの異なる比熱比 (例えば $\gamma = 1.3, 1.4, 1.5, 1.6$ など) についての解を比較せよ．(図 1 を参照せよ.)

なお，ある x に対して物理的な解を見つけてしまえば，それを初期値として用いることで，その近傍の x に対する物理的な解がすぐに見つかる．例えば臨界点 $(x, u) = (1, 1)$ が解になっていることは直ちに確かめられるので，ここから少しずつ x を変化させて数値解を求めていけば良い．小さな δ (例えば 0.05) に対して， $x_{new} = (1 \pm \delta) x_{old}$ のように x を変化させながら解を求めていくと $x \ll 1$ から $x \gg 1$ までの広い範囲の解を求める際にも効率が良い．

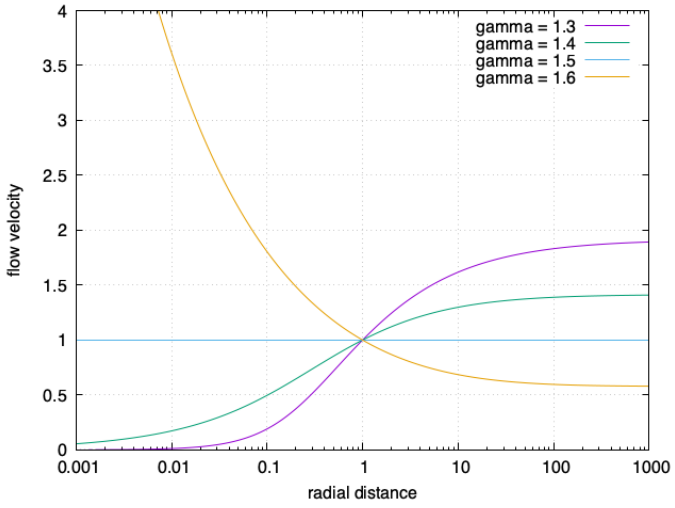

図 1: 太陽風速度の距離依存性

課題　セルオートマトン

セルオートマトン (Cellular Automaton) と呼ばれるモデルを実装しよう．1 次元的に並んだ N 個のセルが与えられ，各セルが 0 もしくは 1 という状態を持つとき，これを $x_i (i = 1, \ldots, N)$ と表す．セルオートマトンでは k 世代目の $x_{i-1}^k, x_i^k, x_{i+1}^k$ の状態からある規則に基づいて $k+1$ 世代目の x_i^{k+1} を決定する．x_i^k は 0 か 1 の 2 通りなので，$x_{i-1}^k, x_i^k, x_{i+1}^k$ の組 111, 110, 101, 100, 011, 010, 001, 000 の 8 パターンに対する x_i^{k+1} を指定すれば良い．例えば 111 → 0, 110 → 0, 101 → 0, 100 → 1, 011 → 1, 010 → 1, 001 → 1, 000 → 0 という規則は x_i^{k+1} を並べたものを 2 進数表記の整数と見ると $00011110_2 = 30_{10}$ (2 進数表記の 00011110 は 10 進数では 30) となるためルール 30 と呼ばれている．このようなセルオートマトンに基づくセルの状態遷移を計算するプログラムを作成せよ．ただしプログラムは以下の仕様を満たすものとする．

- ▷ 標準入力の 1 行目からルールを表す 10 進の整数，2 行目からセル数 N，3 行目から状態遷移を計算するステップ数 M を読み込み，それ以降の行から N 個分の初期条件を読み込むものとする．`kadai4_1.txt`, `kadai4_2.txt`, `kadai4_3.txt` はこの書式で記述されたファイルになっているので，これらをリダイレクトで読み込ませることが出来れば良い．
- ▷ 計算には境界条件 (端の値 x_1 および x_N) が必要になるが，これは初期値のままで固定とする．(すなわち，x_2 から x_{N-1} までを更新すれば良い．)

このプログラムに `kadai4_1.txt`, `kadai4_2.txt`, `kadai4_3.txt` を入力として与えた場合の状態遷移の様子を図示せよ．例えば `kadai4_1.txt` を入力として，結果を `kadai4_1.dat` に書き込む場合には

```
1: $ ./a.out < kadai4_1.txt > kadai4_1.dat
```

のように実行出来れば良い．

ヒント

- ▷ ルールは長さ 8 の整数配列を用いて記述することが出来る．$x_{i-1}^k, x_i^k, x_{i+1}^k$ の組をこの整数配列のインデックスに (例えば $j = 2^0 \times x_{i-1}^k + 2^1 \times x_i^k + 2^2 \times x_{i+1}^k$ のように) マッピングしてやると良い．
- ▷ 出力したファイル (以下では `data.dat`) を gnuplot で図示する際には

```
1: > set palette gray negative
2: > plot 'data.dat' matrix with image
```

などとすると良いかもしれない．(ただし出力するデータファイルの形式に依存する．)

課題　ヒルベルト曲線

2 次元の単位正方形を $2^{n+1} \times 2^{n+1}$ の均等なセルに分割しよう．これらの全てのセルの中心を一度ずつ通る連続した交わることのない曲線を空間充填曲線 (space-filling curve) と呼ぶ．ヒルベルト曲線はこの空間充填曲線の一つの例である．与えられた任意の非負整数 $n \geq 0$ についてヒルベルト曲線を生成するプログラムを作成し，例えば $n = 0, 1, 2, 3, 4, 5$ のヒルベルト曲線を図示せよ．(点列をファイルに出力し，gnuplot 等で図示すれば良い.)

なお，2 次元のヒルベルト曲線は図 2 のようにカタカナの「コ」の字およびそれを回転させた 4 つの図 (LDR, URD, RUL, DLU) を基準とし，これらの曲線を再帰的に接続していったものになっている．m 次のヒルベルト曲線を下付き添字 $_m$ で表すと，これは $m-1$ 次のヒルベルト曲線とそれらをつなぐ直線によって以下のように再帰的に定義することが出来る．

$$
\begin{aligned}
\mathrm{LDR}_m &: \mathrm{DLU}_{m-1} \to \mathrm{L} \to \mathrm{LDR}_{m-1} \to \mathrm{D} \to \mathrm{LDR}_{m-1} \to \mathrm{R} \to \mathrm{URD}_{m-1} \\
\mathrm{URD}_m &: \mathrm{RUL}_{m-1} \to \mathrm{U} \to \mathrm{URD}_{m-1} \to \mathrm{R} \to \mathrm{URD}_{m-1} \to \mathrm{D} \to \mathrm{LDR}_{m-1} \\
\mathrm{RUL}_m &: \mathrm{URD}_{m-1} \to \mathrm{R} \to \mathrm{RUL}_{m-1} \to \mathrm{U} \to \mathrm{RUL}_{m-1} \to \mathrm{L} \to \mathrm{DLU}_{m-1} \\
\mathrm{DLU}_m &: \mathrm{LDR}_{m-1} \to \mathrm{D} \to \mathrm{DLU}_{m-1} \to \mathrm{L} \to \mathrm{DLU}_{m-1} \to \mathrm{U} \to \mathrm{RUL}_{m-1}
\end{aligned}
$$

ここで U, D, L, R はそれぞれ上下左右に長さ $1/2^{n+1}$ の直線を引くことを意味する．

ヒント

▷ 最も簡単な実装は再帰的サブルーチンもしくは関数 (`recursive`) を用いて上記の関係を表現する方法であろう．

▷ 曲線を表す点列の座標値を保持し，この値を順次更新しながら出力していけばよい．例えば U なら $y \to y + 1/2^{n+1}$，L ならば $x \to x - 1/2^{n+1}$ など．

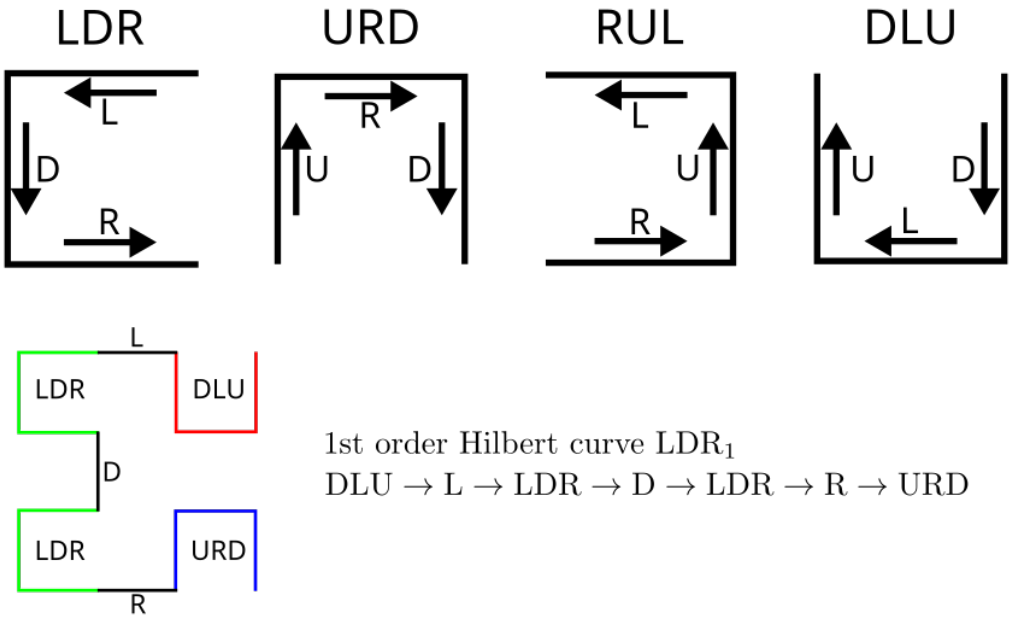

図 2: ヒルベルト曲線

課題　ソート

演習ではバブルソート (bubble sort) を扱ったが，これに加えてヒープソート (heap sort)，マージソート (merge sort)，クイックソート (quick sort) などのアルゴリズムを用いた場合の計算速度を測定・比較しよう．

`kadai6.f90` に定義されているメインプログラムでは，まず正確にソートされているかのテストを行い，様々なデータサイズに対して実際に計算時間を測定する．このメインプログラムでは各アルゴリズムを用いた整数配列に対するソートのサブルーチンがモジュール `mod_sort` の内部手続きとして `bsort` (バブルソート)，`hsort` (ヒープソート)，`msort`(マージソート)，`qsort`(クイックソート) として定義されているものと仮定する．これらのサブルーチンは `kadai6_sort.f90` のモジュール `mod_sort` の内部手続きとして予め定義されているが，`hsort`，`msort`，`qsort` の中身は全てバブルソート (`bsort` を呼出すだけ) になっている．これら 3 つのサブルーチンを全て正しく実装し，計算時間の測定結果を図示せよ．ただし新しく実装したサブルーチンの計算速度の計測を行うには，`kadai6_sort.f90` のフラグ `test_?sort` (`?`は `q`，`m`，`h`，`b` のいずれか) の値も変更する必要がある．例えば `qsort` を正しく実装し，計算時間の測定を行うには `test_qsort` の値を`.true.` に変更すること．

具体的には，サブルーチンを実装しフラグの値を変更した後に，以下のようにコンパイル・実行し，出力結果 (以下の例では `kadai6.dat`) を gnuplot 等を用いて図示すれば良い．ただし縦軸・横軸ともに対数スケールとすること．また，計算時間のデータサイズ N に対する依存性が理論的な関係性を満たすことも確認せよ．(図 3 参照．バブルソートでは $O(N^2)$，それ以外のアルゴリズムは全て $O(N \log N)$ の依存性を持つ．)

```
1:  $ gfortran kadai6_sort.f90 kadai6.f90
2:  $ ./a.out > kadai6.dat
```

例えば `kadai6.dat` の中身は以下の様になるだろう．

```
1:        16        0.163E-05    0.298E-05    0.206E-05    0.198E-05
2:        64        0.755E-05    0.132E-04    0.950E-05    0.231E-04
3:       256        0.395E-04    0.548E-04    0.553E-04    0.251E-03
4:      1024        0.166E-03    0.216E-03    0.236E-03    0.361E-02
5:      4096        0.763E-03    0.938E-03    0.110E-02    0.626E-01
6:     16384        0.311E-02    0.422E-02    0.512E-02    0.120E+01
7:     65536        0.136E-01    0.185E-01    0.232E-01    0.206E+02
```

左からソートする配列の長さ，`qsort`，`msort`，`hsort`，`bsort` のそれぞれについての計算時間である．ただし実際の値はこれとは異なるだろう．なお，`test_?sort` が `.false.` の場合は計算時間は (測定されずに)0 となる．

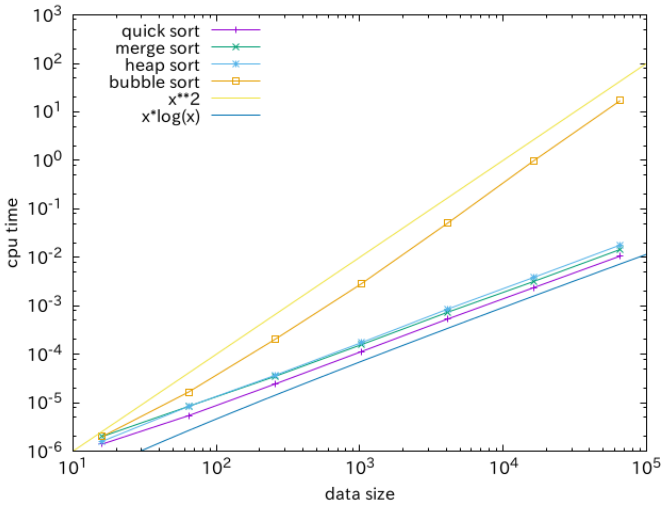

図 3: 各種アルゴリズムによるソート時間計測の例

課題　自由課題

▷ 例 1: 来年度以降の 3 年生へのレポート課題および解答例を考える．難しすぎず，簡単すぎず，出来た時に学生が「面白い」と思えるような教育的なもの，さらには課題として well-defined であるものを高く評価する．(要するに課題にそのまま採用出来るようなレベルのものが望ましい．)

▷ 例 2: Fortran に限らずプログラミングや計算科学に関連することで，自分で興味を持ったことを調べてまとめる．例えば，並列計算 (OpenMP が最も簡単である)，GPU プログラミング，C と Fortran の相互利用 (`iso_c_binding` モジュールを使ったプログラム作成)，Python と Fortran の相互利用，Fortran 2008 の派生型 (オブジェクト志向プログラミング)，他のプログラミング言語と Fortran の比較，など．