2024 年 8 月 17 日

Supercon2024 本選問題: 森林火災の消火

1 目的

大規模な森林火災のニュースは世界的には決して珍しくない。森林火災の広がりを解析、予測することは防災の観点から重要である。一方で落雷などによって自然に起こる火災が、樹木の成長と拮抗しながら森林の形成にどのように関わっているかは興味深い自然科学の問題である。[*1] こうした森林火災、森林形成の研究において様々な数理モデルが用いられている。ここでは感染症の伝搬のモデルとして有名な SIR 模型を応用した数理モデルを考え、これを用いて森林火災の消火訓練のシミュレーションを行ってみよう。

2 森林火災モデル

森林に覆われたある島を考える。この島を、その周りを取り囲む海の一部を含めて図 1 のように格子状に区切る。セル (マスのこと) の座標を (x, y) のように表すと、島に含まれるセルでは必ず x と y がそれぞれ $1, 2, \ldots, L$ の整数値をとるものとする。

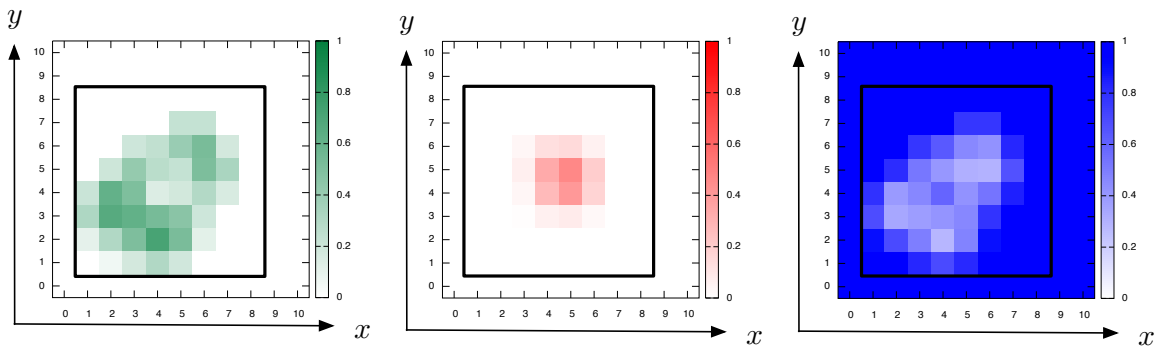
セルに $n = 0, 1, 2, \ldots$ のようにラベルをつけよう。各セル $n = 0, 1, 2, \ldots$ に時間 $t = 0, 1, 2, \ldots$ とともに変化する 3 つの変数 $T_n(t), F_n(t), E_n(t)$ を定義する。これらはそれぞれそのセルにおける燃えていない木の密度、燃えている木の密度、燃えてしまった木の密度を表している。(海の部分でもこれらの変数を仮想的に定義しておくと便利であるが、その説明は後回しにしよう。)

島の中のセルでは、これらの量が以下の式にしたがって時間変化しているとしよう。またまずは島の海岸線から離れた、島の内部を考える。

$$T_n(t+1) = T_n(t) - \phi(h_n(t))T_n(t) \tag{1}$$
$$F_n(t+1) = F_n(t) + \phi(h_n(t))T_n(t) - \gamma F_n(t) \tag{2}$$
$$E_n(t+1) = E_n(t) + \gamma F_n(t) \tag{3}$$

ここで時間は $t = 0, 1, 2,$ と離散的に刻まれている。時間の刻み幅は数時間程度に対応すると考えておこう。ここで $\phi(h_n(t))T_n(t)$ は燃えていない木に引火する過程を表してい

[*1] アメリカのイエローストーン国立公園では 1972 年に森林火災に関するそれまでの消火の方針を変更し、「あらゆる火災を見つけ次第消火する」というそれまでの方針を転換し、「小中規模のものはそのままにする」ことにした。その方が大規模な火災を防げるとの考えに基づく。



燃えていない木の密度分布　　燃えている木の密度分布　　燃えてしまった木の密度分布
(あるいは海)

図 1　ある島における森林の状況の例: ある時刻 t における (左から) $T_n(t)$、$F_n(t)$,$E_n(t)$ の空間分布。島は一辺の長さを L とする正方形 (黒) の内側に必ず含まれ、島内のセルの座標 (x, y) は x と y がそれぞれ $1, 2, \ldots, L$ の整数値をとる。(この例では $L = 8$)。島以外の部分は海である。

る。その詳細は後述する。また γ を係数とする項は燃えている木の燃焼が終わってゆく過程を表している。(1)-(3) の和をとると、$T_n(t) + F_n(t) + E_n(t)$ はどの時間でも変化しないことがわかる。これは n 番目のセルにおける密度の合計で、以下では全てのセルでその値を 1 にする。

次に引火する過程を表す $\phi(h_n(t))T_n(t)$ の項を説明しよう。木の密度 $T_n(t)$ と、$\phi(h_n(t))$ の積になっていることに注目してほしい。引数になっている $h_n(t)$ の定義は後述するが、引火させようとする働きの強さ、のようなものである。関数 $\phi(h)$ は

$$\phi(h) = \begin{cases} 0 & h < h_c \\ \tanh(A(h - h_c)) & h > h_c \end{cases} \tag{4}$$

のように定義する (tanh は双曲線正接関数 (ハイパーボリックタンジェント) と呼ばれ、C++ 言語の数学関数では tanh() で求められる)。これは図 2 に示すように強い非線形性 (直線的でないこと) を持つ関数で、引数 h が閾値 h_c 以下では 0、h_c 以上で 0 でない値をとり、h が大きくなると 1 に収束する。h は引火させようとする働きの強さ、のようなものであるが、それが弱すぎると全く効果を及ぼさず、閾値 h_c を超えると急激に強く働くようになる、というモデルである。A が $\phi(h)$ の急峻さを決めている。

さて、$\phi(h_n(t))$ の引数になっている $h_n(t)$ の定義は

$$h_n(t) = \beta_1 F_n(t) + \beta_2 \Sigma_{m \in n \text{ の第 1 近接}} F_m(t) + \beta_3 \Sigma_{m \in n \text{ の第 2 近接}} F_m(t) \tag{5}$$

である。ここでまず β_1 を係数とする項は、同じセルの中において燃えている木から燃えていない木に引火する働きの強さを表している。また β_2 を係数とする項は、第 1 近接のセル（図 3 参照) にある燃えている木から引火する働きの強さを表している。記号 $\Sigma_{m\in n \text{ の第 1 近接}}$ は第一近接にあるセルに関する和を表す。同じように β_3 係数とする項は、第 2 近接のセルからの引火を表す。

さて海の部分のセルでは仮想的に $T_n(t) = F_n(t) = 0$,$E_n(t) = 1$ としておくと都合が良い。これらは時間変化しないが、少し考えてみるとわかるように、これらを定義しておくことにより、島の海岸線部分 (図 1 参照) での $T_n(t)$,$F_n(t)$,$E_n(t)$ の時間発展についても、上で説明した島の内部でのこれらの量の時間発展と全く同様に扱うことができる。

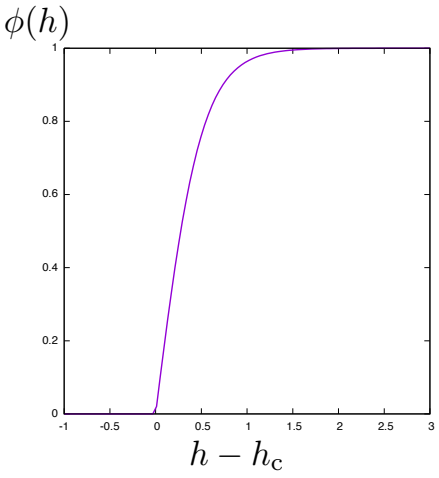


図 2　関数 $\phi(h)$ の関数形。 ここでは $A = 2$ としている。

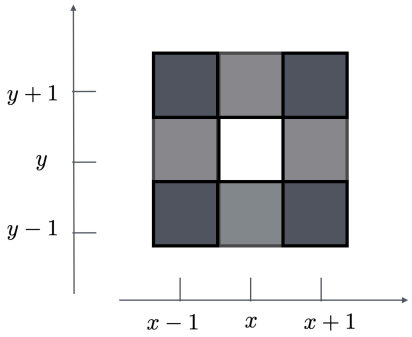


図 3　座標 (x, y) にあるセル（白）の第 1 近接は薄いグレーの 4 つのセルである。また第 2 近接は濃いグレーの 4 つのセルである。

3　問題

この島の森林火災のための消火活動のシミュレーションをする。島の森林の状況、すなわち $T_n(t)$,$F_n(t)$,$E_n(t)$ $(n = 0, 1, 2, \ldots, N_c - 1)$ は人工衛星を通じてモニターされている。ここで N_c は島に属するセルの総数である。ある時刻 (これを時刻 $t = 0$ としよう) に複数の箇所で火災が同時に起こったことが観測され、消火隊に出動が要請される。

消火隊は普段島外にいて、出動要請を受けてから島に向かう。島に到着して作業できるのは時刻 t_w においてだけである、としよう。t_w は定数で指定される。消火隊が到着するまでの間、すなわち時刻 $t = 0$ から t_w までにおいて森林の状態は (1)-(3) に従って変化す

る。消火隊が目指していることは、作業後十分時間が経過したある時刻 $t_{\rm c}$ における燃えていない木の密度の総和

$$S = \sum_{n=0}^{N_c-1} T_n(t_{\rm c}) \tag{6}$$

をなるべく大きくするように消火作業をデザイン (計画) することである。$t_{\rm c}$ は定数で指定される。[*2]

消火隊ができる作業は燃えていない木を切り倒すことのみとしよう。n 番目のセルでこの作業をすると

$$T_n(t_{\rm w}) \to T_n(t_{\rm w}) - \Delta T_n \qquad E_n(t_{\rm w}) \to E_n(t_{\rm w}) + \Delta T_n \tag{7}$$

となる。$\Delta T_n(\geq 0)$ は切り倒された木の密度である。もちろんその許される最大値は作業前の $T_n(t_{\rm w})$ である。すなわち $\Delta T_n(n = 0, 1, 2, \ldots, N_c - 1)$ を決めることが作業のデザインである。予算等の制約のため作業の量は次のように制限されている。切り倒す木の密度の総和は不等式

$$\sum_{n=0}^{N_c-1} \Delta T(n) < R \sum_{n=0}^{N_c-1} F_n(t_{\rm w}) \tag{8}$$

を満たさなければならない。$\sum_{n=0}^{N_c-1} F_n(t_{\rm w})$ は燃えている木の密度の総和である。R は定数で指定される。

簡単のため、作業の時間は無視できてその間における森林の状態変化はないとしよう。作業後の $T_n(t_{\rm w})$,$F_n(t_{\rm w})$,$E_n(t_{\rm w})$ $(n = 0, 1, 2, \ldots, N_c - 1)$ を初期状態として、その後の $t_{\rm w}+1, t_{\rm w}+2, \ldots$ の森林の状態は (1)-(3) に従って変化を続けるとする。

問題:S をなるべく大きくするように $\Delta T_n(n = 0, 1, 2, \ldots, N_c - 1)$ を決めて欲しい。
時刻 $t_{\rm w}$ と $t_{\rm c}$、時刻 $t = 0$ での森林の状態が問題で与えられる。

4 詳細

- 入力として与えられるのは、10 個の問題に関するデータである。島は一辺の長さ $L = 1000$ の正方形領域 (図 1 参照) に入っている。それぞれの問題について

[*2] 実際には十分時間が経過するとはほぼ確実に雨が降って火が消えるであろうと考え、その推測のもとで上の $t_{\rm c}$ が設定されている、というシナリオである。

– 島に含まれるセルの総数 $N_{\rm c}$：その数は問題によるが 10^5 程度である。
– その一つ一つ $n = 0, 1, 2, \ldots N_{\rm c} - 1$ の座標 $(x_{\rm island}[n], y_{\rm island}[n])$ および時刻 $t = 0$ における $T_n(0)$ および $F_n(0)$。($E_n(0) = 1 - T_n(0) - F_n(0)$ に注意。)

の組みがデータとなっている。これらは提供されるヘッダーに含まれる `SC_input()` 関数を呼ぶことで読み込まれる。この関数によって 10 問分のデータ全てが一度に読み込まれる。

- 消火活動する時刻 $t_{\rm w}$ は 10 に固定される。また、消火活動の成果を判定する時刻 $t_{\rm c}$ は 500 に固定される。
- 作成するプログラムは各セル $n = 0, 1, 2, \ldots, N_{\rm c} - 1$ において切り倒す木の量 ΔT_n である。これは提供されるヘッダーに含まれる `SC_output (int i_prob)` 関数を呼ぶことで書き出される。これは問題 (`i_prob=0,1,2,..9`) ごとに行う。
- プログラムはタイムアウトで強制終了させるので終了処理は書かなくて良い。それまでに結果を、問題ごとに何度出力しても構わないが、制限時間内に出力された最後の結果が審査に使われる。

5 勝利条件

- 提出されたプログラムで問題を 10 問解く。実行時間の上限は全体で 10 分とする。実行時間内に出力されたそれぞれの問題の最後の結果を審査に採用する。問題には問題番号 $0, 1, 2, \ldots, 9$ が割り振られているが、出力の順は問題番号の順にしたがう必要はない。また、10 問全ての解答が提出されていなくても、解答が提出された問題について審査する。
- 各問について、S が大きいものから順に順位をつけ、1 位 5 点、2 位 4 点、3 位 3 点、4 位 2 点、5 位 1 点、以下 0 点のように点数化する。
- 10 問の点数を合計して総合順位をつける。点数が等しい時には、総計算時間 (下記の `SC_output` で出力される「計算開始からの経過時間」で、制限時間内に出力された最後のもの) が短いチームを上位とする。

6 ヘッダーファイル

配布するコンテスト専用ヘッダーファイル `sc24.h` に以下のように必要な定数・変数・配列・関数が定義されている。

```
#include <time.h>
# define N_prob_SC 10: 問題の数 N_prob
# define L_max_SC 1000 : 島を囲む正方形領域の一辺の長さ L
# define Nc_max_SC L_max_SC*L_max_SC : その領域に含まれるセルの数
# define BILLION  1000000000.0 : 時間計測用

const double beta1_SC=0.8; : パラメータ  β1 ((3) 参照)
const double beta2_SC=0.4; : パラメータ β2 ((3) 参照)
const double beta3_SC=0.2; : パラメータ β3 ((3) 参照)
const double gamma_SC=0.1; : パラメータ γ ((3) 参照)
const double hc_SC=0.1; : パラメータ hc ((4) 参照)

const double A_SC=2.0; : パラメータ A ((4) 参照)

const int tw_SC=10; : 消火活動する時刻 tw
const int tc_SC=500; : 消火活動の結果を評価をする時刻 tc
const double R_SC=0.5; : 切る木の数を制限するパラメータ R ((8) 参照)

int Nc_SC[N_prob_SC];: 島内のセルの総数 Nc
int x_island_SC[N_prob_SC][Nc_max_SC]; : 島内の各セルの x 座標
int y_island_SC[N_prob_SC][Nc_max_SC];: y 座標
double T_SC[N_prob_SC][Nc_max_SC]; : t = 0 で燃えていない木の密度
                                     Tn(0)
double F_SC[N_prob_SC][Nc_max_SC]; : t = 0 で燃えている木の密度
                                     Fn(0)

double Delta_T_SC[N_prob_SC][Nc_max_SC]; : t = tc で切る木の密度
                                           ΔTn

struct timespec ts0_SC; : 時間計測関係
```

```
void SC_input(){   : 入力関数
  int rank = 0;
#ifdef MPI_VERSION
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
#endif
  if(0 == rank){
    for (int i_prob=0; i_prob < N_prob; i_prob++){
      scanf("%d\n",&Nc_SC[i_prob]);
      for (int n=0; n< Nc_SC[i_prob]; n++){
        scanf("%d %d %lf %lf\n",\
        &x_island_SC[i_prob][n],&y_island_SC[i_prob][n],\
        &T_SC[i_prob][n]),&F_SC[i_prob][n]);
      };
    };
  };
#ifdef MPI_VERSION
 for (int i_prob=0; i_prob < N_prob_SC; i_prob++){
    MPI_Bcast(&Nc_SC[i_prob], 1,\
            MPI_INT, 0, MPI_COMM_WORLD);
    MPI_Bcast(&x_island_SC[i_prob][0], Nc_SC[i_prob],\
            MPI_INT, 0, MPI_COMM_WORLD);
    MPI_Bcast(&y_island_SC[i_prob][0], Nc_SC[i_prob], \
            MPI_INT, 0, MPI_COMM_WORLD);
    MPI_Bcast(&T_SC[i_prob][0], Nc_SC[i_prob], \
            MPI_DOUBLE, 0, MPI_COMM_WORLD);
    MPI_Bcast(&F_SC[i_prob][0], Nc_SC[i_prob], \
            MPI_DOUBLE, 0, MPI_COMM_WORLD);
  };
#endif
  if (0 == rank){
    clock_gettime(CLOCK_REALTIME, &ts0_SC); : 時間計測開始
```

```
  };
};

double SC_time_spent(){
        struct timespec ts;
    clock_gettime(CLOCK_REALTIME, &ts);
    double time_spent = (double)(ts.tv_sec  - ts0_SC.tv_sec )
             + (double)(ts.tv_nsec - ts0_SC.tv_nsec) / BILLION;
        return time_spent;
}

void SC_output(int i_prob){
      : 出力関数 (i_prob は出力する問題の番号 ∈ {0,1,2,..,N_prob − 1} )
  int rank = 0;
#ifdef MPI_VERSION
  MPI_Comm_rank(MPI_COMM_WORLD, &rank);
#endif
  if(0 == rank){
    clock_gettime(CLOCK_REALTIME, &ts2);
    double time_spent = (ts2.tv_sec - ts1.tv_sec) + \
                        (ts2.tv_nsec -ts1.tv_nsec) / BILLION;
    printf("# elapsed time %lf sec\n",SC_time_spent());
      : 計算開始からの経過時間を出力
    printf("#prob,Nc_SC[i_prob]= %d %d \n",i_prob,Nc_SC[i_prob]);
    for (int n=0; n < Nc_SC[i_prob]; n++){
      printf("%lf\n",Delta_T_SC[i_prob][n]);
      fflush(NULL);
    };
  };
```

以上のようなヘッダーファイル sc24.h を

```
# include <stdio.h>
# include <math.h>
# include <omp.h>
# include <mpi.h>
# include "sc24.h"
```

のようにイングルードして使う。sc24.h は最後にインクルードする。また MPI を使わない時 mpi.h をインクルードしない。

1. 関数 SC_input() はプログラム中で最初の実行文とすること (MPI を使用するときには、最初の実行文を MPI_init にする必要があり、SC_input() はその直後の実行文とする)。これ以前に実行文を書いて はいけない。SC_input を実行することにより必要なデータが標準入出力から読み込まれる。サンプルデータの場所については 14.1 節を参照。
2. 解答の ΔT_n を配列 Delta_Tree_SC[n] に格納し、関数 SC_output(int i_prob) を呼ぶことにより i_prob 番目 ($\in \{0, 1, 2, .., N_{\rm prob} - 1\}$) の問題に対する解答が出力される。SC_output() 関数以外の方法で出力してはならない。SC_output() 関数は何度呼んでも良い。
3. SC_timespent() 関数は時間計測用に使って良い。
4. ヘッダーファイルは書き換えてはならない (採点時には採点用のヘッダーと置き換える)。

7 サンプルプログラム

サンプルプログラムと関連するスクリプトを
/system/lecture/SuperCon2024/Sample_Program 以下におく。

- simple.c : C 言語で書かれているが mpnc++ でコンパイルできる。チューニングされたものではないが、配列の作り方の例、mpi 並列化の例、時間発展部分の書き方の例 (ベクトル化を効かせるべきところ)、ライブラリを利用した乱数発生器 (10 節) の使い方の例、などの参考にしてほしい。
- compile.sh: コンパイル用のスクリプト例。コマンドラインから sh compile.sh によってコンパイルできる。(9 節)
- simple.sh : qsub するジョブスクリプトの例 (11 節)

8　スーパーコンピュータ

本選では阪大 CMC のスパコン squid を用いる。 ベクトルノード (ベクトルエンジン NEC SX-Aurora TSUBASA Type20A(10 コア) 8 基 主記憶容量：48GB) を用いる。

このコンテストのために `CONTEST` というジョブクラスが定義されており、これを用いる。このジョブクラスではベクトルノード 1 ノードを使い、15 分の計算ができる。ただし後述のように制限時間は 10 分とする。

- ベクトルノードなのでプログラムの**ベクトル化**ができると計算が高速化する。時間発展 (1)-(3) はベクトル化しやすいことに注目して欲しい。
- ベクトルエンジン (ve) 1 つにつき、10 コアが付随している。これだけならば **OpenMP による並列化**ができる。この並列化をコンパイラにまかせる**自動並列化**という機能もある。
- 1 ノードにある 8 個のベクトルエンジン （$8 \times 10 = 80$ コア）を使い切るためには **MPI による並列化**が必要である。MPI だけで, あるいは OpenMP と組み合わて。

9　コンパイルの方法

ベクトルノードを用いるのでコンパイルにおいて当然ながらベクトル化は行いたい。一方、並列化は MPI（分散メモリー型並列化)、OpenMP（自動並列化を含む）(共有メモリー型並列化)、これらを組みあわせる、という場合が考えられる。そこで以下のようにコンパイルする。

- **基本形** (`OpenMP`(自動ベクトル化を含む）を用いず、並列化は `MPI` のみによる場合）: プログラム名を `prog.cpp` とすると `module load BaseVEC`
  `mpinc++  -O4 -report-all  prog.cpp`

  注：MPI を使わない場合でもコンパイラは `mpinc++`、上のオプションで良い。ベクトル化は可能な限りなされる。(mpi なしのときに使う `nc++` と結果的に同じ働きになる。)
- OpenMP を用いる場合 (MPI を使わない場合を含む)
  コンパイルオプションに `-fopenmp` を加える。
- 自動並列化を用いる場合 (MPI を使わない場合を含む)
  コンパイルオプションに `-mparallel` を加える。

上で用いられているコンパイルオプションの意味は

- **O4**：最適化オプション - 最大限のベクトル化、最適化を行うコンパイルをする。
- **report-all** :: コード生成リスト、診断メッセージリスト、編集リスト、インラインリスト、オプションリスト、ベクトルリストを出力する。 上の場合、`prog.L` という名前のファイルができる。 **これはベクトル化、OpenMP（自動並列化）による並列化がどの程度なされたかを知る有用な情報になる。**

なお、乱数発生のために下記の ASL を用いるためには追加でコンパイルオプションが必要。 下記資料を参考に。

- SQUID の利用方法 (この中の「ベクトルノードの利用方法」)
- NEC SDK C/C++Compiler ユーザーズガイド

10 乱数

プログラムで乱数が必要な場合があるだろう。乱数の生成は、NEC Numeric Library Collection (NEC NLC) に入っている ASL 統合インタフェースにある乱数生成を用いるのが高速で良い。C のライブラリとなっているが C++ で使える.

- `#include <asl.h>`をプログラムの上に入れる。
- コンパイルとリンクを行う前に、コンパイル環境設定スクリプトを実行する必要あり。
  - sh 系のシェルの場合:
    ```
    source /opt/nec/ve/nlc/3.1.0/bin/nlcvars.sh mpi i64
    ```
  - csh 系のシェルの場合:
    ```
    source /opt/nec/ve/nlc/3.1.0/bin/nlcvars.csh mpi i64
    ```

  ここで `mpi`,`64` はそれぞれ mpi(分散メモリ) 対応させる場合、64 ビット対応させるためのオプションで不要なら外す。
- コンパイルオプションに付け加えるものはこちらを参照。なお、mpi を使わない場合でも `mpinc++` で OK。

11　ジョブスクリプト

ジョブスクリプトの例を下に示す。**基本形** OpenMP(自動並列化を含む）を用いない場合。(mpi も使わない場合を含む。)

```
#!/bin/bash
#PBS -q CONTEST : コンテスト用のキューである。
#PBS --group= sc24groupXX  : 【グループ名】
#PBS -l elapstim_req=0:15:00 : 制限時間は 15 分
#PBS --venode=8 : ベクトルエンジンを 8 つを使う場合（最大 8）
#PBS -T necmpi : mpi を使う場合必要
cd $PBS_O_WORKDIR
module load BaseVEC : ベクトルノードでの環境変数の設定
source /opt/nec/ve/nlc/3.1.0/bin/nlcvars.sh mpi i64: NLC ライブラリ
を使用する場合必要（上記の ASL が入っている）不要なら外す。
mpirun -venode -np 80 ./a.out < inputfile > outputfile
: 総並列数を-np の後に指定。 総並列数 = 利用するベクトルエンジンの数
(#PBS --venode) × 10(ベクトルエンジンの搭載コア数)
```

OpenMP(自動並列化を含む）を用いる場合 (mpi を使わない場合を含む。)

```
#!/bin/bash
#PBS -q CONTEST : コンテスト用のキューである。
#PBS --group= sc24groupXX  : 【グループ名】
#PBS -l elapstim_req=0:15:00 : 制限時間は 15 分
#PBS --venode=8 : ベクトルエンジンを 8 つを使う場合（最大 8）
#PBS -T necmpi : mpi を使わなければ外す。
#PBS -v OMP_NUM_THREADS=10 :  各ベクトルエンジン内の OpenMP(自動並列化を含む）による並列数
cd $PBS_O_WORKDIR
module load BaseVEC : ベクトルノードでの環境変数の設定
source /opt/nec/ve/nlc/3.1.0/bin/nlcvars.sh mpi i64: NLC ライブラリを使用する場合必要（上記 ASL が入っている。）不要なら外す。
mpirun -venode -np 8 ./a.out < inputfile > outputfile : mpi による総並列数（OMP_NUM_THREDS のファクターは入らない）を-np の後に指定。
```

12 提出物

以下 XX はチームの番号

- プログラムのソースコード sc24groupXX.cpp
- コンパイルのスクリプト compile-sc24groupXX.sh
- バッチジョブのためのジョブスクリプト sc24groupXX.sh

を提出する。なお、ソースコードは必ずひとつのファイルにまとめてあること。次節「サンプルプログラム」を参考に。分割したファイルや makefile の使用は認めない。提出方法の詳細は期間中に説明する。

13　サンプルプログラム

サンプルプログラムと関連するスクリプトを
`/system/lecture/SuperCon2024/Sample_Program`　以下におく。

- `sc24.h` このコンテストのヘッダーファイル
- `simple-mpi.c` 並列化に関しては mpi のみのサンプルプログラム
- `compile-simple-mpi.c` これをコンパイルするためのシェルスクリプト
- `simple-mpi.sh` 実行ファイルを CONTEST キューで qsub するジョブスクリプト

14　付録：問題サンプル

14.1　問題サンプルの場所

問題サンプルは`/system/lecture/SuperCon2024/Prob_Samples` 以下にある。`sample01, sample02,..` などでそれぞれに１０問分のデータが入っている。標準入力から`./a.out < sample01` などとすれば読み込むことができる。

14.2　問題サンプルの生成方法の概要

- **島の形状の生成**: 便宜的に次のように森林火災を利用して人工的に島の形状を生成している。$L \times L$ の正方形領域を考え (図 1 参照)、その全てのセルで $T_n = 1$ $(F_n = E_n = 0)$ としておく。この正方形領域の中心付近 ($L/2 \times L/2$ の大きさの領域) でランダムに 10 個のセルを選び、そこで $F_n = 1 (T_n = E_n = 0)$ とする。そこから (1)-(3) にしたがって時間発展させ、正方形領域の端に火が到達 $(F_n > 0.0)$ したところで時間発展を止める。この時点で $T_n < 0.1$ となっているセルの集合を「島」と定義する。
- **問題として与える森林の初期状態の作り方**: 上の方法で生成した島の内部のセル $n = 0, 1, 2, \ldots, N_c - 1$ に対して次のような森林形成のシミュレーションを行う。
  - 初期 $(t = 0)$ の密度分布: $T_n(0)$ は $0.6 < T < 1.0$ の範囲に一様分布した乱数として定める。島内に火災はなく $F_n(0) = 0$、$E_n(0) = 1 - T_n(0)$ とする。ここから時間発展させる。
  - 時間発展 (1)-(3) に加えて下記の 3 つのプロセスが同時に進行する。

1. 島内の各セルにおいて単位時間あたり確率 $p_{\mathrm{lightening}}$ で「落雷」があり、そのとき、そのセルで $F_n(t) \to F_n(t) + T_n(t)$, $T_n(t) \to 0$ となる。
2. 島内の全てのセル $\forall n$ で木が成長（成長率 δ) し、$T_n(t+1) = T_n(t) \ldots +$ $\delta E_n(t)$ となる。 このとき $E_n(t+1) = E_n(t) \ldots - \delta E_n(t)$ となる。
3. 単位時間ある確率 p_{rain} で全島に降雨があり、全てのセル $\forall n$ で火災が消える $E_n(t) \to E_n(t) + F[n]$,$F_n(t) \to 0$ $(\forall n)$

以上のような時間発展モデルを用いて、森林形成のシミュレーションを行う。パラメータとして $\beta_1 = 0.4$, $\beta_2 = 0.2$,$\beta_3 = 0.1$, $\gamma = 0.1$, $h_{\mathrm{c}} = 0.1$ また、$p_{\mathrm{lightening}} = 10^{-5}$, $\delta = 10^{-3}$, $p_{\mathrm{rain}} = 0.03$ を用いる。[*3]これで 10^4 ステップほど時間発展させたのちの状況の例が図 4 である。

– 上のシミュレーションで 10^4 ステップほど時間発展させた状態を取り出し、全てのセル $(\forall n)$ での火災を消す $E_n(t) \to E_n(t) + F[n]$,$F_n(t) \to 0$ $(\forall n)$。
– 島内のセルからランダムに 10 個のセルを選び、そこに同時に落雷が起こったとし、そこで $F_n(t) \to F_n(t) + T_n(t)$, $T_n(t) \to 0$ とする。この状態において消火隊に出動が要請されたとする。この時刻をあらためて $t = 0$ とする。

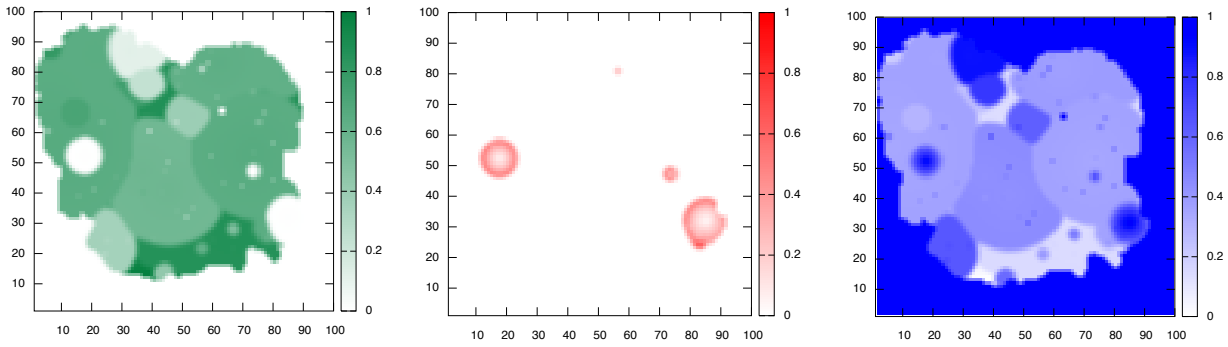


図 4　森林の状況の例: この例では $L = 100$ である。パラメータとして $\beta_1 = 0.4$, $\beta_2 = 0.2$,$\beta_3 = 0.1$,$\gamma = 0.1$,$h_{\mathrm{c}} = 0.1$ また、$p_{\mathrm{lightening}} = 10^{-5}$, $\delta = 10^{-3}$,$p_{\mathrm{rain}} = 0.03$ を用いた.

[*3] 問題の中ではこれら 1-3 は起きないとしている。