

# Évolution d'une population [si04] - Exercice

Karine Zampieri, Stéphane Rivière

Unisciel

sciel

algotprog

UNIVERSITÉ  
HAUTE-ALSACE

Version 22 mai 2018

## Table des matières

<b>1</b>	<b>Évolution d'une population / pgepidemie</b>	<b>2</b>
1.1	Facteur de contagion . . . . .	2
1.2	Calcul et affichage de l'évolution . . . . .	4
1.3	Statistiques sur la population malade . . . . .	6
<b>2</b>	<b>Références générales</b>	<b>9</b>



## C++ - Évolution d'une population (Solution)



**Mots-Clés** Simulation ■

**Requis** Axiomatique impérative (sauf Fichiers) ■

**Fichiers** rsepidemie3.txt ■

**Difficulté** ●○○ (1 h) ■



### Objectif

Cet exercice étudie l'évolution d'une population lors d'une épidémie.

# 1 Évolution d'une population / pgepidemie

## 1.1 Facteur de contagion

L'épidémie est caractérisée par un facteur de contagion  $k$ . La population malade (c.-à-d. le nombre de personnes malades) est donnée en millions (nombre réel).

Soit  $p_{j-1}$  la population malade à l'instant  $j - 1$ . La population malade à l'instant  $j$  est donnée par :

$$p_j = p_{j-1} + k p_{j-1}(1 - p_{j-1})$$

avec  $p_0$  la population initiale.



Écrivez une fonction `contagion(k,p)` qui calcule et renvoie la population malade à l'instant suivant. Le paramètre  $k$  (réel) est le facteur de contagion d'une épidémie et  $p$  (réel) est la population malade à l'instant courant.



Écrivez une fonction `populationMalade(k,p0,n)` qui calcule et renvoie la population malade  $p_n$  calculée à un instant  $n$  (entier) à partir d'une population initiale  $p_0$  (réel), l'épidémie ayant un facteur de contagion  $k$  (réel).



Écrivez une procédure `demanderPopulation(p)` qui saisit une population dans  $p$  (réel) comprise entre 0 (inclus) et 1 (inclus) (à vérifier).  
Affichez l'invite :

Population dans [0,1]?



Écrivez une procédure `saisirDonnees(k,p0,n)` qui saisit :

- Un facteur de contagion dans  $k$  (réel).
- Une population initiale dans  $p_0$  (réel) entre 0 et 1.
- Un instant dans  $n$  (entier).

Affichez les invites supplémentaires :

Facteur de contagion?

Instant maximal?



Validez vos fonctions et procédures avec la solution.

**Solution C++** @[pgepidemie.cpp]

```
/**
 * Contagion suivante
 * @param[in] k - facteur de contagion
 * @param[in] p - population courante
 * @return la contagion suivante issue de (k,p)
 */
double contagion(double k, double p)
{
    return p+k*p*(1.0-p);
}

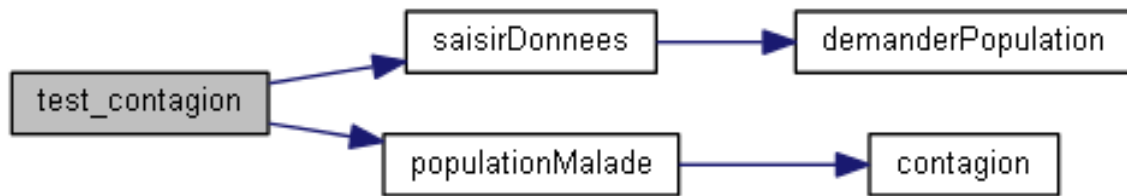
/**
 * Population malade
 * @param[in] k - facteur de contagion
 * @param[in] p0 - population initiale
 * @param[in] n - instant
 * @return la population issue de (k,p0) a l'instant n
 */
double populationMalade(double k, double p0, int n)
{
    double pn=p0; // population courante
    for (int j=1 ; j<=n ; ++j)
    {
        pn = contagion(k,pn);
    }
    return pn;
}

/**
 * Saisit de la population
 * @return une population dans [0,1]
 */
double saisiePopulation()
{
    double p;
    do {
        cout<<"Population dans [0,1]? ";
        cin>>p;
    } while (not((0.0<=p) and (p<=1.0)));
    return p;
}

/**
 * Saisit les donnees d'une epidemie
 * @param[out] k -
 * @param[out] p0 -
 * @param[out] n -
 */
void saisirDonnees(double& k, double& p0, int& n)
{
    cout<<"Facteur de contagion? ";
    cin>>k;
    p0 = saisiePopulation();
    cout<<"Instant? ";
    cin>>n;
}
```



Écrivez une procédure `test_contagion` qui saisit les données d'une épidémie  $(k, p_0, n)$  puis calcule et affiche la population malade  $p_n$ .



Testez. Exemple d'exécution :

```

Facteur de contagion? 2.4
Population dans [0,1]? 0.3
Instant maximal? 18
==> La population malade à l'instant 18 est 1.1920279464
  
```



Validez votre procédure avec la solution.

**Solution C++** @[pgepidemie.cpp]

```

void test_contagion()
{
    double k;
    double p0;
    int n;
    saisirDonnees(k,p0,n);
    cout<<"==> La population malade a l'instant "<<n<<" est
         "<<populationMalade(k,p0,n)<<endl;
}
  
```

## 1.2 Calcul et affichage de l'évolution



Écrivez une procédure `afficherPopulationMalade(k, p0, n)` qui calcule et affiche les populations malades  $p_1, \dots, p_n$  calculées à partir d'une population initiale  $p_0$  (réel), l'épidémie ayant un facteur de contagion  $k$  (réel).

Affichez :

```

==> La population malade à l'instant ... est ...
  
```



Validez votre procédure avec la solution.

**Solution C++** @[pgepidemie.cpp]

```

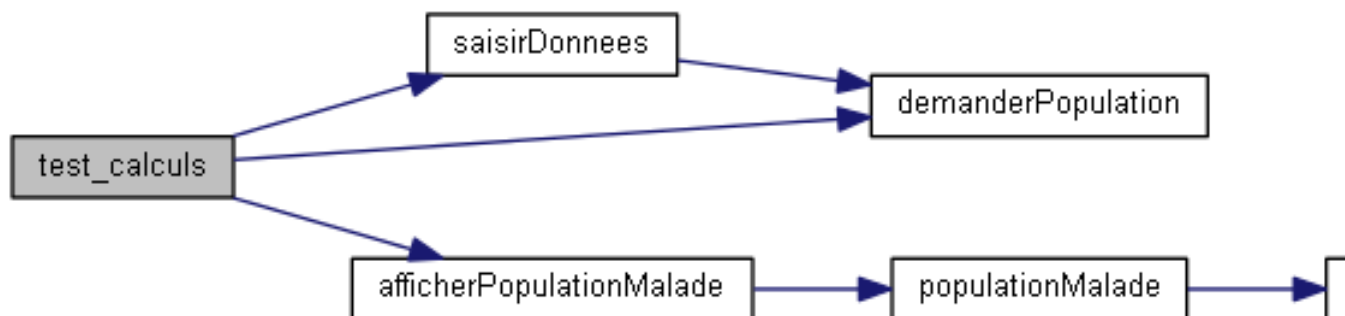
/**
 Affichage de l'évolution
 @param[in] k - facteur de contagion
 @param[in] p0 - population initiale
 @param[in] n - instant
 */
void afficherPopulationMalade(double k, double p0, int n)
{
    for (int j=1 ; j<=n ; ++j)
    {
        cout<<"==> La population malade a l'instant "<<j<<" est
        "<<populationMalade(k,p0,j)<<endl;
    }
}

```



Écrivez une procédure `test_calculs` qui :

- Demande et saisit les données d'une épidémie ( $k, p_0, n$ ).
- Calcule et affiche les populations malades de l'instant 1 à  $n$ .
- Redemande une population initiale, affiche les populations malades correspondantes... la boucle s'arrêtant quand l'utilisateur tape une population initiale  $p_0$  valant 0.



Testez. Extrait d'exécution, l'intégral étant fourni en téléchargement :

@[rsepidemie2.txt]

```

Facteur de contagion? 2.4
Population dans [0,1]? 0.3
Instant maximal? 18
==> La population malade à l'instant 1 est 0.804
==> La population malade à l'instant 2 est 1.1822016
...
==> La population malade à l'instant 18 est 1.1920279464
Population dans [0,1]? 0

```



Validez votre procédure avec la solution.

**Solution C++** @[pgepidemie.cpp]

```

void test_calculs()
{
    double k;
    double p0;
    int n;
    saisirDonnees(k,p0,n);
    while (p0!=0.0)
    {
        afficherPopulationMalade(k,p0,n);
        p0 = saisiePopulation();
    }
}

```

### 1.3 Statistiques sur la population malade

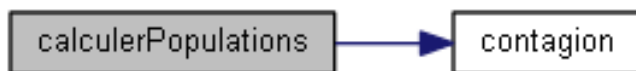
Ce problème calcule quelques indicateurs sur l'évolution de la maladie. Afin de réaliser les analyses, on stocke les populations malades  $p_j$  dans un tableau.



Définissez la constante `TMAX=100` (nombre maximal de populations malades) et le type `Populations` comme étant un tableau de `TMAX` réels où seront stockées les populations malades.



Écrivez une procédure `calculerPopulations(p,k,p0,n)` qui calcule les  $n$  (entier) populations malades  $p_j$  dans une `Populations p` à partir d'une population initiale  $p_0$  (réel), l'épidémie ayant un facteur de contagion  $k$  (réel).



On souhaite savoir si la maladie a tendance à être répandue (les  $p_j$  sont plutôt supérieurs à  $p_0$ ) ou restreinte (les  $p_j$  sont plutôt inférieurs à  $p_0$ ).

Écrivez une fonction `populationBasse(p,n)` qui calcule et renvoie le nombre de fois où une population  $p_j$  est strictement inférieure à la population initiale  $p_0$ , les  $n$  (entier) populations malades étant stockées dans une `Populations p`.



On veut savoir si la population risque d'atteindre un seuil critique. Pour cela il faut connaître la population minimum.

Écrivez une fonction `populationMinimum(p,n)` qui calcule et renvoie la population minimale des  $n$  (entier) populations  $p_j$  stockées dans une `Populations p` sans tenir compte de la population initiale  $p_0$ .



Validez vos fonctions et procédure avec la solution.

**Solution C++**    @[pgepidemie.cpp]

```
/**
 * Calcul des populations
 * @param[out] p - les Populations calculees
 * @param[in] k - facteur de contagion
 * @param[in] p0 - population initiale
 * @param[in] n - taille de p
 */
void calculerPopulations(Population& p, double k, double p0, int n)
{
    p[0] = p0;
    for (int j=1 ; j<n ; ++j)
    {
        p[j] = contagion(k, p[j-1]);
    }
}
```

```
/**
 * Population basse
 * @param[in] p - les Populations
 * @param[in] n - taille de p
 * @return la population basse des n populations p
 */
int populationBasse(const Population& p, int n)
{
    int rs=0;
    for (int j=1 ; j<n ; ++j)
    {
        if (p[j]<p[0])
        {
            ++rs;
        }
    }
    return rs;
}
```

```
/**
 * Population minimale
 * @param[in] p - les Populations
 * @param[in] n - taille de p
 * @return la population minimale ses n populations p, p0 exclus
 */
double populationMinimum(const Population& p, int n)
{
    double vmin=p[1];
    for (int j=2 ; j<n ; ++j)
    {
        if (p[j]<vmin)
        {
            vmin = p[j];
        }
    }
    return vmin;
}
```



Écrivez une procédure `afficherPopulationMalade2(p,n)` qui affiche les `n` (entier) populations malades stockées dans une `Populations p`. Affichez :

==> La population malade à l'instant ... est ...



Validez votre procédure avec la solution.

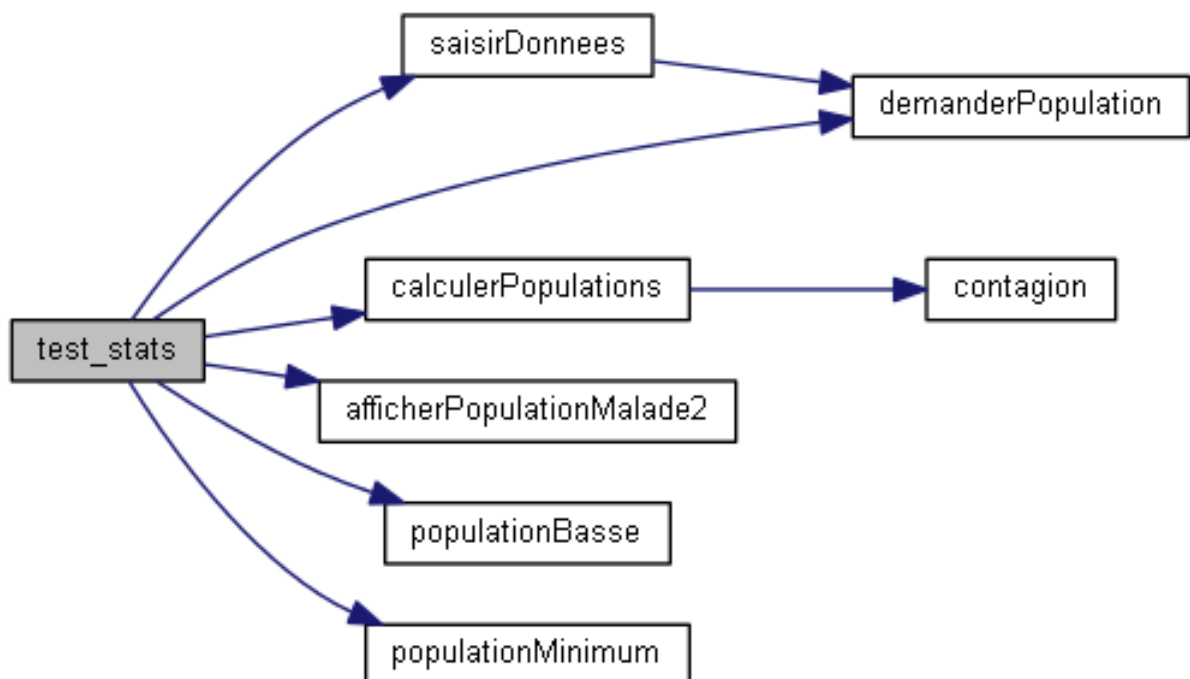
### Solution C++ @[pgepidemie.cpp]

```
/**
 * Affiche les n populations du tableau p
 * @param[in] p - les Populations
 * @param[in] n - taille de p
 */
void afficherPopulationMalade2(const Population& p,int n)
{
    for (int j=0 ; j<n ; ++j)
    {
        cout<<"==> La population malade a l'instant "<<j<<" est "<<p[j]<<endl;
    }
}
```



Écrivez une procédure `test_stats` qui :

- Demande et saisit les données d'une épidémie dans `(k,p0,n)`.
- Calcule et affiche les populations malades de l'instant 1 à `n` ainsi que la population basse et la population minimum.
- Redemande une population initiale, affiche les populations malades correspondantes... la boucle s'arrêtant quand l'utilisateur tape une population initiale `p0` valant 0.







Testez. Extrait d'exécution, l'intégral étant fourni en téléchargement ici.  
@[rs-epidemie3.txt]

```
Facteur de contagion? 2.4
Population dans [0,1]? 0.3
Instant maximal? 18
==> La population malade à l'instant 1 est 0.3
==> La population malade à l'instant 2 est 0.804
...
==> La population malade à l'instant 18 est 0.6372151211
Population basse 0
Population minimum 0.6246827884
Population dans [0,1]? 0
```



Validez votre procédure avec la solution.

**Solution C++** @[pgepidemie.cpp]

```
void test_stats()
{
    double k;
    double p0;
    int n;
    saisirDonnees(k,p0,n);
    Population p;
    while (p0!=0.0)
    {
        calculerPopulations(p,k,p0,n);
        afficherPopulationMalade2(p,n);
        cout<<"Population basse "<<populationBasse(p,n)<<endl;
        cout<<"Population minimum "<<populationMinimum(p,n)<<endl;
        p0 = saisiePopulation();
    }
}
```

## 2 Références générales

Comprend [] ■