多次元ニュートン法

野下浩司

2018-05-21

多次元ニュートン法を用いて，被食-捕食系の平衡点を見つけることを目指す．ここでは出来るだけ簡単な方法でプログラムを書こう．

1　解きたい問題

被食-捕食系

$$\begin{cases} \dfrac{dx}{dt} = ax - bxy & (1) \\ \dfrac{dy}{dt} = cxy - dy & (2) \end{cases}$$

の平衡点を数値的に求める．ただし，$x, y, a, b, c, d \geq 0$ とする．解析的には $\dfrac{dx}{dt} = 0, \dfrac{dy}{dt} = 0$ を解けば良いので，$(0,0), \left(\dfrac{d}{c}, \dfrac{a}{b}\right)$ の 2 つが平衡点として求まる．よって，これら 2 つの平衡点を数値的に求めることが今回解きたい問題となる．

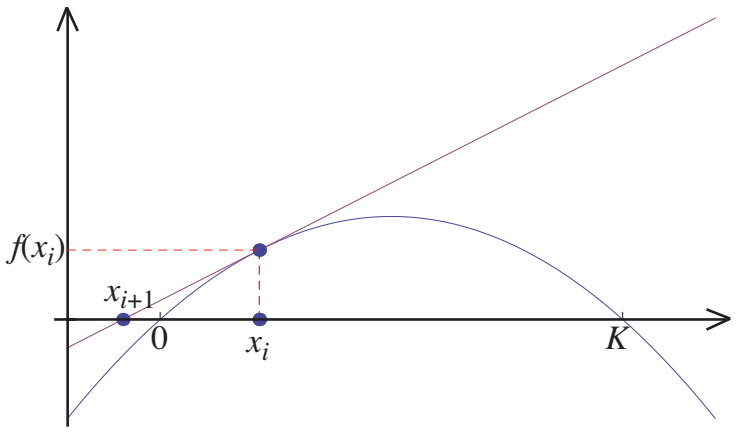
2　1 次元の場合の復習

ロジスティック成長モデルを例に，1 次元の場合を復習する．多次元の場合はその自然な拡張になる．

一般の 1 次元ニュートン法では $f(x) = 0$ の近似解を求めるために，

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} \tag{3}$$

を適当な初期値 x_0 から始めて，$x_1, x_2, \ldots, x_n$ と次々に求めればよかった．これは x_i での $f(x)$ の接線の x 切片をより良い近似値 x_{i+1} として採用してゆく，と解釈できる．



よって，初期値によって得られる近似解が異なる．

ロジスティック成長モデルを例にプログラムを組んでみよう．

$$\frac{dx}{dt} = r\left(1 - \frac{x}{K}\right)x \tag{4}$$

この場合，平衡点は $x_1^* = 0, x_2^* = K$ となる．これら 2 点を数値的に求める．通常，どの程度の精度で近似するかによって反復回数の制御を行うが，今回は単純化のため 1000 回の反復で計算を打ち切ることにする．接線の傾きは

$$f'(x) = r\left(1 - \frac{2}{K}x\right) \tag{5}$$

なので,

$$x_{i+1} = \frac{x_i^2}{2x_i - K}. \tag{6}$$

よってプログラムは

```
#include <stdio.h>

int main(void){
  int i;
  double x,xx,K;

  //パラメータ設定
  K=100.0;

  //初期値
```

```
    x=10.0;

    for(i=0;i<1000;i++){
      xx=x*x/(2*x-K);
      x=xx;
    }

    printf("%f\n",x);

    return 0;
  }
```

初期値に依存して，$x_1^* = 0$ 及び $x_2^* = K$ のどちらかが数値的に求められる．

3　n 次元のニュートン法

では，n 次元の場合に拡張しよう．

$$\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0} \tag{7}$$

という連立方程式を解くことを考える．ただし，

$$\boldsymbol{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}, \quad \boldsymbol{f}(\boldsymbol{x}) = \begin{pmatrix} f_1(x_1, x_2, \cdots, x_n) \\ f_2(x_1, x_2, \cdots, x_n) \\ \vdots \\ f_n(x_1, x_2, \cdots, x_n) \end{pmatrix} \tag{8}$$

$\boldsymbol{f}(\boldsymbol{x})$ の $\boldsymbol{x} = \boldsymbol{x}_i$ での一次近似（高次元での “接線”）を考えると，

$$\boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{f}(\boldsymbol{x}_i) + \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}=\boldsymbol{x}_i} (\boldsymbol{x} - \boldsymbol{x}_i) \tag{9}$$

となる．$\dfrac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}$ はヤコビ行列．$\boldsymbol{J}_{\boldsymbol{x}_i} = \left.\dfrac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}=\boldsymbol{x}_i}$ とし，$\boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{0}$ となる $\boldsymbol{x}$ をより良い近似解 $\boldsymbol{x}_{i+1}$ として採用すれば，

$$\boldsymbol{f}(\boldsymbol{x}_i) + \boldsymbol{J}_{\boldsymbol{x}_i}(\boldsymbol{x}_{i+1} - \boldsymbol{x}_i) = 0 \tag{10}$$

$$\boldsymbol{J}_{\boldsymbol{x}_i}(\boldsymbol{x}_{i+1} - \boldsymbol{x}_i) = -\boldsymbol{f}(\boldsymbol{x}_i) \tag{11}$$

$$\boldsymbol{x}_{i+1} - \boldsymbol{x}_i = -\boldsymbol{J}_{\boldsymbol{x}_i}^{-1} \boldsymbol{f}(\boldsymbol{x}_i) \tag{12}$$

となり，

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i - \boldsymbol{J}_{\boldsymbol{x}_i}^{-1} \boldsymbol{f}(\boldsymbol{x}_i) \tag{13}$$

を得る．1次元でのニュートン法と見比べてみると，高次元への自然な拡張となっていることが分かる．しかし通常は，これを直接扱うのは（逆行列の演算が入っているため）厄介なので，

$$\Delta\boldsymbol{x}_i = \boldsymbol{J}_{\boldsymbol{x}_i}^{-1} \boldsymbol{f}(\boldsymbol{x}_i) \tag{14}$$

として，これを満たす $\Delta\boldsymbol{x}_i$ を

$$\boldsymbol{J}_{\boldsymbol{x}_i} \Delta\boldsymbol{x}_i = \boldsymbol{f}(\boldsymbol{x}_i) \tag{15}$$

に対しガウス-ジョルダンの掃出法や LU 分解などを用いて解くことで求める．

今回は，行列の演算は本題では無いので，逆行列を求め直接計算することにする．

4　被食-捕食系　２次元の場合の例として

被食-捕食系の微分方程式系を確認し，ニュートン法に必要なヤコビ行列とその逆行列を求めよう．

$$\begin{cases} \dfrac{dx}{dt} = ax - bxy \\ \dfrac{dy}{dt} = cxy - dy \end{cases}$$

の右辺をそれぞれ $f_1(x,y), f_2(x,y)$ とすれば，

$$f_1(x,y) = ax - bxy = 0 \tag{16}$$
$$f_2(x,y) = cxy - dy = 0 \tag{17}$$

平衡点を求めるためには，この連立方程式を解けば良い．ニュートン法により解を求めるため，まず，$\boldsymbol{f}(x,y)$ のヤコビ行列を求める．求めるヤコビ行列を $\boldsymbol{J}$ とすれば

$$\boldsymbol{J}(x,y) = \begin{pmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} \end{pmatrix} = \begin{pmatrix} a - by & -bx \\ cy & cx - d \end{pmatrix}. \tag{18}$$

よって，ヤコビアンは

$$|\boldsymbol{J}(x,y)| = acx + bdy - ad \tag{19}$$

となり，$\boldsymbol{J}(x,y) \neq 0$ の場合のみを考えれば，逆行列はクラメールの公式を用い

$$\boldsymbol{J}^{-1}(x,y) = \frac{1}{acx + bdy - ad} \begin{pmatrix} cx - d & bx \\ -cy & a - by \end{pmatrix} \tag{20}$$

と求まる．最終的に，

$$\boldsymbol{x_{i+1}} = \boldsymbol{x_i} - \boldsymbol{J}_{x_i}^{-1} \boldsymbol{f}(\boldsymbol{x_i}) \tag{21}$$

$$\begin{pmatrix} x_{i+1} \\ y_{i+1} \end{pmatrix} = \begin{pmatrix} x_i \\ y_i \end{pmatrix} - \frac{1}{acx_i + bdy_i - ad} \begin{pmatrix} cx_i - d & bx_i \\ -cy_i & a - by_i \end{pmatrix} \begin{pmatrix} ax_i - bx_iy_i \\ cx_iy_i - dy_i \end{pmatrix} \tag{22}$$

$$= \begin{pmatrix} x_i \\ y_i \end{pmatrix} - \frac{1}{acx_i + bdy_i - ad} \begin{pmatrix} acx_i^2 - adx_i \\ bdy_i^2 - ady_i \end{pmatrix} \tag{23}$$

$$= \frac{x_iy_i}{acx_i + bdy_i - ad} \begin{pmatrix} bd \\ ac \end{pmatrix} \tag{24}$$

よって，プログラムは以下のようになる．

```
#include <stdio.h>

int main(void){
  int i;
  double x,xx,y,yy;
  double a,b,c,d;
  double x0,y0;

  //パラメータ設定
  a=2.0;
  b=3.0;
  c=1.0;
  d=2.0;

  //初期値
  x=1.5;
  y=0.5;

  for(i=0;i<1000;i++){
    xx=b*d*x*y/(a*c*x+b*d*y-a*d);
    yy=a*c*x*y/(a*c*x+b*d*y-a*d);
    x=xx;
    y=yy;
```

```
    }
    printf("%f, %f\n",x,y);

    return 0;
}
```

初期値に依存して，平衡点 $(x_1^*, y_1^*) = (0,0), (x_2^*, y_2^*) = \left(\frac{d}{c}, \frac{a}{b}\right)$ のいずれかが数値的に求まる．

基本的に，より高次元の場合にも同様に適用できる．