数値シミュレーション実験

第3回レポート

提出日　2025年６月１８日

提出者　今村　優斗

学籍番号　2713240012-7

* 課題１

以下の漸化式で求められる変数*an* と*bn* を*a*0, *b*0 から*a*24, *b*24 まで求めて表示せよ．また，*an, bn*の*n* に対する変化をグラフにして示し，さらに，縦軸を*an*, 横軸を*bn* としたグラフを示せ．

解答：

図1にプログラム、図2に実行結果を示した。図1より、数列a,bを個別に計算するのではなく、同じfor文内で計算した。

#include <stdio.h>

#define N 25

int main(void){

    double a[N+1];

    double b[N+1];

    a[0]=1.0;

    b[0]=0.0;

    printf("%lf %lf\n",a[0],b[0]);

    for(int i=0;i<N;i++){

        a[i+1]=0.866\*a[i]-0.500\*b[i];

        b[i+1]=0.500\*a[i]+0.866\*b[i];

        printf("%lf %lf\n",a[i+1],b[i+1]);

    }

    return 0;

}

図1課題１のプログラム

$./k3-1

1.000000 0.000000

0.866000 0.500000

0.499956 0.866000

-0.000038 0.999934

-0.500000 0.865924

-0.865962 0.499890

-0.999868 -0.000076

-0.865848 -0.500000

-0.499824 -0.865924

0.000114 -0.999802

0.500000 -0.865771

0.865886 -0.499758

0.999736 0.000152

0.865695 0.500000

0.499692 0.865848

-0.000190 0.999670

-0.500000 0.865619

-0.865809 0.499626

-0.999604 -0.000229

-0.865543 -0.500000

-0.499560 -0.865771

0.000267 -0.999538

0.500000 -0.865467

0.865733 -0.499494

0.999472 0.000305

0.865390 0.500000

図２　課題１の実行結果

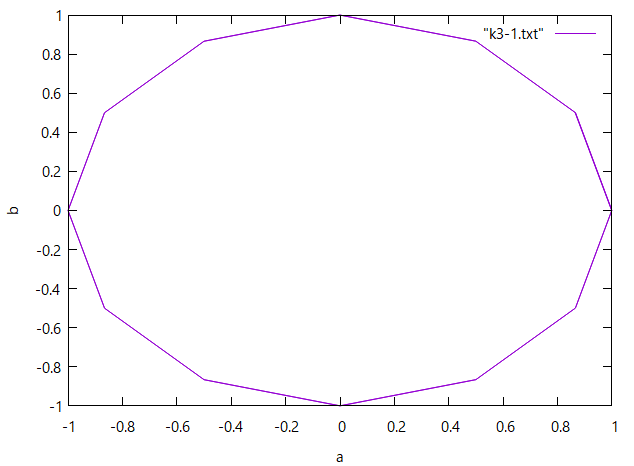


図３　課題1の結果のグラフ

* 課題２

反応式(1)～(3) で表現される反応で，GLU とG6P はどのような時間変化を示すか．オイラー法を用いて時間変化を計算するプログラムを作れ．また，結果をグラフに示せ．なお，GLU の初期値は，1.0 [mM]，GLU-HK およびG6P の初期値は0.0 [mM] とし，*CHKtotal* = 0*.*1 [mM], *k*1 = 1*.*0[/mM/min], *k*2 = 1*.*0 [/min], *k*3 = 0*.*1 [/min] とせよ．時間ステップ幅Δ*t* やシミュレーションする時間等は，現象に合わせて適切に選択せよ．

解答：

図4に課題2のプログラム、図5に結果のグラフを示した。図4より、それぞれの濃度を表す、配列を作成し、微分を行う関数fglu,fgluhk,fg6pを作成した。その後、オイラー法のプログラムで適宜関数を呼び出し計算した。シミュレーション幅と時間に関して、反応が落ち着くまでかなりの時間がかかったため、dtを600/36000分として600分間のシミュレーションを行った。図5より、紫色がglu濃度で、緑色がg6p濃度を示す。

#include <stdio.h>

#define N 36000

double glu[N+1];

double g6p[N+1];

double gluhk[N+1];

double hkfree[N+1];

double dglu[N+1];

double dgluhk[N+1];

double dg6p[N+1];

double dt=600.0/N;

double fglu(double k1,double glu,double hkfree,double gluhk,double k2){

    double q1=k1\*glu\*hkfree;

    double q2=k2\*gluhk;

    return (-q1+q2);

}

double fgluhk(double k1,double glu,double hkfree,double gluhk,double k2,double k3){

    double q1=k1\*glu\*hkfree;

    double q2=k2\*gluhk;

    double q3=k3\*gluhk;

    return (q1-q2-q3);

}

double fg6p(double k3,double gluhk){

    double q3=k3\*gluhk;

    return q3;

}

int main(void){

    glu[0]=1.0;

    gluhk[0]=0.0;

    g6p[0]=0.0;

    double k1=1.0;

    double k2=1.0;

    double k3=0.1;

    double hktotal=0.1;

    printf("%lf %lf %lf\n",dt\*0,glu[0],g6p[0]);

    for(int i=0;i<N;i++){

        hkfree[i]=hktotal-gluhk[i]; //hkfreeの濃度計算

        dglu[i]=fglu(k1,glu[i],hkfree[i],gluhk[i],k2);

        dgluhk[i]=fgluhk(k1,glu[i],hkfree[i],gluhk[i],k2,k3);

        dg6p[i]=fg6p(k3,gluhk[i]);

        glu[i+1]=glu[i]+dglu[i]\*dt;

        gluhk[i+1]=gluhk[i]+dgluhk[i]\*dt;

        g6p[i+1]=g6p[i]+dg6p[i]\*dt;

        printf("%lf %lf %lf\n",dt\*(i+1),glu[i+1],g6p[i+1]);

    }

    return 0;

}

図4　課題2のプログラム

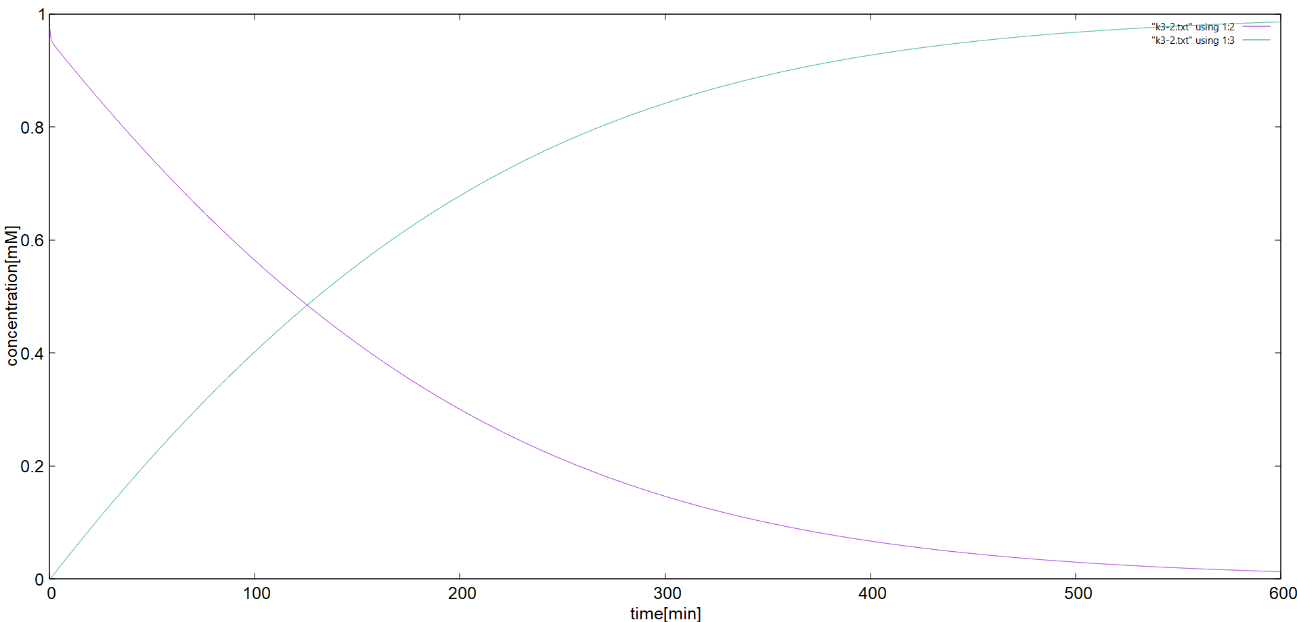


図5　課題2の結果のグラフ

* 課題３

課題2で，GLU の初期値を，1.0 [mM] から1000.0 [mM] 程度の範囲で変更して実験した場合，G6P 産生の初速度はGLU の初期濃度に対してどのように変化するか，課題2 のプログラムのGLUの初期値を変更して調べよ．さらに，その結果を，横軸をGLU の初期濃度，縦軸をG6P の産生速度としたグラフで示せ．グラフの軸に単位を記述すること．また，どのようにして初速度を決定したかも説明せよ．

解答：

図6にプログラム、図7に結果のグラフ、図8に課題2のgluhkを含めたグラフを示した。図6より、課題2のプログラムが、時間によるgluとg6pの濃度変化を表しているため利用した。具体的には、時間の微小区間dtを0.001として100回実行し、得られたg6pの濃度変化を得る。この操作をglu濃度が1.0〜1000.0まで繰り返しグラフを作成する。そのために、二重for文内のint kのfor文にてglu濃度を決定し、その値でオイラー法によりg6p濃度を計算した。図7より、得られた数値をグラフにして表した。また、図8より、水色の線がgluhk濃度を示しており、0.5min付近まで急激に上昇していることがわかる。よって初速度を0~0.5minまでの時間とした。

#include <stdio.h>

#define N 1000

double glu[N+1];

double cglu[1000];

double vg6p[1000];

double g6p[N+1];

double gluhk[N+1];

double hkfree[N+1];

double dglu[N+1];

double dgluhk[N+1];

double dg6p[N+1];

double dt=0.5/1000;

double fglu(double k1,double glu,double hkfree,double gluhk,double k2){

    double q1=k1\*glu\*hkfree;

    double q2=k2\*gluhk;

    return (-q1+q2);

}

double fgluhk(double k1,double glu,double hkfree,double gluhk,double k2,double k3){

    double q1=k1\*glu\*hkfree;

    double q2=k2\*gluhk;

    double q3=k3\*gluhk;

    return (q1-q2-q3);

}

double fg6p(double k3,double gluhk){

    double q3=k3\*gluhk;

    return q3;

}

int main(void){

    for(int k=1;k<=1000;k++){

        glu[0]=k;

        gluhk[0]=0.0;

        g6p[0]=0.0;

        double k1=1.0;

        double k2=1.0;

        double k3=0.1;

        double hktotal=0.1;

        for(int i=0;i<N;i++){

            hkfree[i]=hktotal-gluhk[i]; //hkfreeの濃度計算

            dglu[i]=fglu(k1,glu[i],hkfree[i],gluhk[i],k2);

            dgluhk[i]=fgluhk(k1,glu[i],hkfree[i],gluhk[i],k2,k3);

            dg6p[i]=fg6p(k3,gluhk[i]);

            glu[i+1]=glu[i]+dglu[i]\*dt;

            gluhk[i+1]=gluhk[i]+dgluhk[i]\*dt;

            g6p[i+1]=g6p[i]+dg6p[i]\*dt;

        }

    vg6p[k]=g6p[100]-g6p[0];

    cglu[k]=k;

    printf("%lf %lf\n",cglu[k],vg6p[k]);

    }

    return 0;

}

図6　課題３のプログラム

グラフ

AI 生成コンテンツは誤りを含む可能性があります。

図7　課題3の結果のグラフ

アプリケーション

AI 生成コンテンツは誤りを含む可能性があります。

図8　課題2のgluhkを含めた図

* 課題４

上記のGLU とHK の反応は，第２回の課題で扱ったミカエリス・メンテン式で近似できる反応

と同じ反応形式である．したがって，式(18) で計算される反応速度は，酵素総量の濃度を[*E*]*tot*，

基質の濃度を[*S*]，反応生成物の濃度を[*P*] とする下記のミカエリス・メンテン式で近似できるはずである．下記の式を使って，課題2 と同じ実験条件で反応生成物の時間変化を計算するプログラムを作れ．また，結果をグラフに示せ．さらに，課題2 で計算した反応生成物の時間変化と，ミカエリス・メンテン式で計算される反応生成物の時間変化にはどのような差があったか説明し，なぜその差が生じたかを式を参照しながら説明せよ．

解答：

図9にプログラム、図10に結果のグラフ、図11に比較のグラフを示した。図9のプログラムより、前回課題を参考に基質を配列s、生成物を配列pとして作成し、オイラー法によって求めた。図11より、紫色の線が課題4、緑色の線が課題2を示す。両者を比較してみるとほとんど差が生じなかったが、特に反応初期から中盤はミカエリス・メンテン式のほうが早い。これは、課題2では、G6Pの生成に酵素基質複合体であるGLU-HKが必要となるが、初期濃度では0.0であったため初めからG6Pを生成することができない。これに対し、ミカエリス・メンテン式では、GLU-HKが必要ないため初めからG6Pを生成することが出来る。この差によって少しのズレが生じた。

#include <stdio.h>

#define N 36000

double s[N+1];

double p[N+1];

double dpdt[N+1];

double dt=600.0/N;

double vmax=0.1\*0.1;

double km=(1.0+0.1)/1.0;

double f(double s){

    return ((vmax\*s)/(km+s));

}

int main(void){

    s[0]=1.0;

    p[0]=0.0;

    printf("%lf %lf %lf\n",dt\*0,s[0],p[0]);

    for(int i=0;i<N;i++){

        dpdt[i]=f(s[i]);

        p[i+1]=p[i]+dpdt[i]\*dt;

        s[i+1]=s[i]-dpdt[i]\*dt;

        printf("%lf %lf %lf\n",dt\*(i+1),s[i],p[i]);

    }

    return 0;

}

図9　課題４のプログラム

グラフ

AI 生成コンテンツは誤りを含む可能性があります。

図10　課題4の結果のグラフ

グラフ

AI 生成コンテンツは誤りを含む可能性があります。

図11　課題4と課題２の反応物を比較したグラフ

* 課題５

ヘキソキナーゼの総濃度を[*HK*]*tot*，遊離しているヘキソキナーゼの濃度を[*HK*]*free*，GLU,

G6P, ATP, ADP, HK-GLU, HK-ATP, HK-GLU-ATP の濃度を，それぞれ[*GLU*], [*G*6*P*], [*ATP*],

[*ADP*], [*HKGLU*], [*HKATP*], [*HKGLUATP*] として，上記の反応式で成り立つ微分方程式と

酵素量の保存式を示せ．

解答：

以下に、それぞれの微分方程式と酵素量の保存式を示す。

* 微分方程式
* 酵素量の保存式
* 課題６

GLU とATP が関与するヘキソキナーゼの反応について，GLU とG6P はどのような時間変

化を示すか，オイラー法を用いて時間変化を計算するプログラムを作れ．また，結果をグラフに示

せ．GLU の初期値は1.0 [mM]，ATP の初期値は0.5 [mM]，GLU-HK, HK-ATP, HK-GLU-ATP

およびG6P,ADP の初期値は0.0 [mM] とし，[*HK*]*tot* = 0*.*1 [mM], *k*1 = 1*.*0 [/mM/min], *k*2 = 0*.*5

[/min], *k*3 = 0*.*5 [/mM/min], *k*4 = 1*.*0 [/min], *k*5 = 0*.*1 [/min] とせよ．時間ステップ幅Δ*t* やシミュレーションする時間等は，現象に合わせて適切に選択せよ．

解答：

図12に課題6のプログラム、課題13に結果のグラフを示した。図12より、それぞれの濃度を配列にして、課題5の微分方程式を行う関数を作成した。それらを組み合わせオイラー法をプログラムし、計算した。反応がかなり長い時間かかったので、dtを1500/90000として1500分間シミュレーションした。図12より、紫色の線がGLU濃度、緑色の線がG6P濃度を示す。

#include <stdio.h>

#define N 90000 //[s]

double glu[N+1];

double g6p[N+1];

double gluhk[N+1];

double hkfree[N+1];

double atp[N+1];

double hkatp[N+1];

double hkgluatp[N+1];

double adp[N+1];

double dglu[N+1];

double dgluhk[N+1];

double dhkfree[N+1];

double datp[N+1];

double dhkatp[N+1];

double dhkgluatp[N+1];

double dadp[N+1];

double dg6p[N+1];

double dt=1500.0/N;   //[min]

double k1=1.0;

double k2=0.5;

double k3=0.5;

double k4=1.0;

double k5=0.1;

double hktot=0.1;

double fglu(double glu,double hkfree,double hkatp,double gluhk,double hkgluatp){

    double q1=k1\*glu\*hkfree;

    double q11=k1\*glu\*hkatp;

    double q2=k2\*gluhk;

    double q21=k2\*hkgluatp;

    return (-q1-q11+q2+q21);

}

double fgluhk(double glu,double hkfree,double gluhk,double atp,double hkgluatp){

    double q1=k1\*glu\*hkfree;

    double q2=k2\*gluhk;

    double q31=k3\*atp\*gluhk;

    double q41=k4\*hkgluatp;

    return (q1-q2-q31+q41);

}

double fhkfree(double glu,double hkfree,double gluhk,double atp,double hkatp,double hkgluatp){

    double q1=k1\*glu\*hkfree;

    double q2=k2\*gluhk;

    double q3=k3\*atp\*hkfree;

    double q4=k4\*hkatp;

    double q5=k5\*hkgluatp;

    return (-q1+q2-q3+q4+q5);

}

double fatp(double atp,double hkfree,double gluhk,double hkatp,double hkgluatp){

    double q3=k3\*atp\*hkfree;

    double q31=k3\*atp\*gluhk;

    double q4=k4\*hkatp;

    double q41=k4\*hkgluatp;

    return (-q3-q31+q4+q41);

}

double fhkatp(double atp,double hkfree,double hkatp,double glu,double hkgluatp){

    double q3=k3\*atp\*hkfree;

    double q4=k4\*hkatp;

    double q11=k1\*glu\*hkatp;

    double q21=k2\*hkgluatp;

    return (q3-q4-q11+q21);

}

double fhkgluatp(double atp,double hkfree,double hkatp,double glu,double hkgluatp,double gluhk){

    double q31=k3\*atp\*gluhk;

    double q41=k4\*hkgluatp;

    double q11=k1\*glu\*hkatp;

    double q21=k2\*hkgluatp;

    double q5=k5\*hkgluatp;

    return (q31-q41+q11-q21-q5);

}

double fg6p(double hkgluatp){

    double q5=k5\*hkgluatp;

    return q5;

}

double fadp(double hkgluatp){

    double q5=k5\*hkgluatp;

    return q5;

}

int main(void){

    glu[0]=1.0;

    atp[0]=0.5;

    gluhk[0]=0.0;

    hkatp[0]=0.0;

    hkgluatp[0]=0.0;

    g6p[0]=0.0;

    adp[0]=0.0;

    printf("%lf %lf %lf\n",dt\*0,glu[0],g6p[0]);

    for(int i=0;i<N;i++){

        hkfree[i]=hktot-gluhk[i]-hkatp[i]-hkgluatp[i]; //hkfreeの濃度計算

        dglu[i]=fglu(glu[i],hkfree[i],hkatp[i],gluhk[i],hkgluatp[i]);

        dgluhk[i]=fgluhk(glu[i],hkfree[i],gluhk[i],atp[i],hkgluatp[i]);

        dhkfree[i]=fhkfree(glu[i],hkfree[i],gluhk[i],atp[i],hkatp[i],hkgluatp[i]);

        datp[i]=fatp(atp[i],hkfree[i],gluhk[i],hkatp[i],hkgluatp[i]);

        dhkatp[i]=fhkatp(atp[i],hkfree[i],hkatp[i],glu[i],hkgluatp[i]);

        dhkgluatp[i]=fhkgluatp(atp[i],hkfree[i],hkatp[i],glu[i],hkgluatp[i],gluhk[i]);

        dg6p[i]=fg6p(hkgluatp[i]);

        dadp[i]=fadp(hkgluatp[i]);

        glu[i+1]=glu[i]+dglu[i]\*dt;

        gluhk[i+1]=gluhk[i]+dgluhk[i]\*dt;

        hkfree[i+1]=hkfree[i]+dhkfree[i]\*dt;

        atp[i+1]=atp[i]+datp[i]\*dt;

        hkatp[i+1]=hkatp[i]+dhkatp[i]\*dt;

        hkgluatp[i+1]=hkgluatp[i]+dhkgluatp[i]\*dt;

        g6p[i+1]=g6p[i]+dg6p[i]\*dt;

        adp[i+1]=adp[i]+dadp[i]\*dt;

        printf("%lf %lf %lf\n",dt\*(i+1),glu[i+1],g6p[i+1]);

    }

    return 0;

}

図12　課題6のプログラム

グラフ

AI 生成コンテンツは誤りを含む可能性があります。

図13　課題6の結果のグラフ

* 課題７

GLU, ATP とHK の反応は，第2 回の課題11 で扱った二つの基質がある場合のミカエリス・

メンテン式で近似できるはずである．第2 回の課題11 で，*KmS*1 = 0*.*5, *KmS*2 = 2*.*0, *Vmax* = 0*.*01

として，課題6 の初期濃度を使ってG6P の時間変化を計算するプログラムを作れ．また，結果をグラフとして示し，課題6 の結果と比較せよ．

解答：

図14にプログラム、図15に結果のグラフ、図16に課題6と7を比較するグラフを示した。図14より、前回の課題11を参考にプログラムした。配列s1、s2、pを作成し、ミカエリス・メンテン式を関数fpにプログラムしオイラー法によって計算した。図16より、課題6を紫色の線で、課題7を緑色の線で示した。両者を比較すると、ミカエリス・メンテンで近似したグラフのほうが早いことが読み取れる。これは、課題4と同様に、課題6はGLUとATPだけで生成されているのではなくほかの複数の反応が組み合わさっているものである。これに対し、ミカエリス・メンテン式はGLUとATPの二つによって生成されている。この違いが両者の濃度に差が出る原因となっている。

#include <stdio.h>

#define N 90000

double s1[N+1]; //glu

double s2[N+1]; //atp

double p[N+1];  //g6p

double dpdt[N+1];

double dt=1500.0/N;

double kms1=0.5;

double kms2=2.0;

double vmax=0.01;

double fp(double s1,double s2){

    double a1=vmax\*s1\*s2;

    double a2=kms1\*kms2;

    double a3=s1/kms1;

    double a4=s2/kms2;

    double a5=(s1\*s2)/a2;

    return ((a1/a2)/(1+a3+a4+a5));

}

int main(void){

    s1[0]=1.0;

    s2[0]=0.5;

    p[0]=0.0;

    printf("%lf %lf %lf\n",dt\*0,s1[0],p[0]);

    for(int i=0;i<N;i++){

        dpdt[i]=fp(s1[i],s2[i]);

        p[i+1]=p[i]+dpdt[i]\*dt;

        s1[i+1]=s1[i]-dpdt[i]\*dt;

        s2[i+1]=s2[i]-dpdt[i]\*dt;

        printf("%lf %lf %lf\n",dt\*(i+1),s1[i+1],p[i+1]);

    }

    return 0;

}

図13　課題7のプログラム

グラフ

AI 生成コンテンツは誤りを含む可能性があります。

図14　課題７の結果のグラフ

グラフ

AI 生成コンテンツは誤りを含む可能性があります。

図15　課題6と7の生成物を比較したグラフ