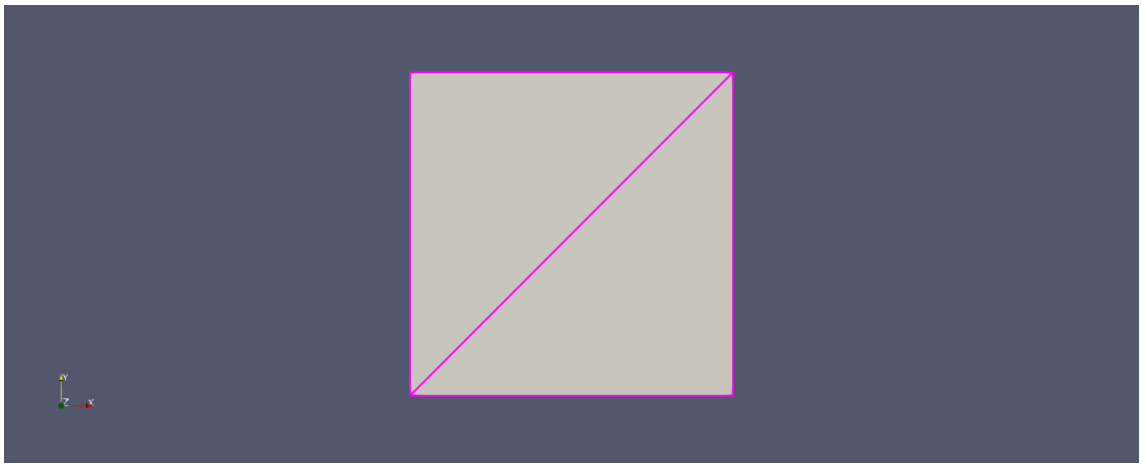
計算機工学最終レポート　7520035 ケニス銀河

(1) 異なる分割数で 2 種類の正方形のメッシュを作成し、計算せよ。 メッシュ図と温度コンター図を示せ。

正方形メッシュ 1



・プログラムの計算結果

t_array =

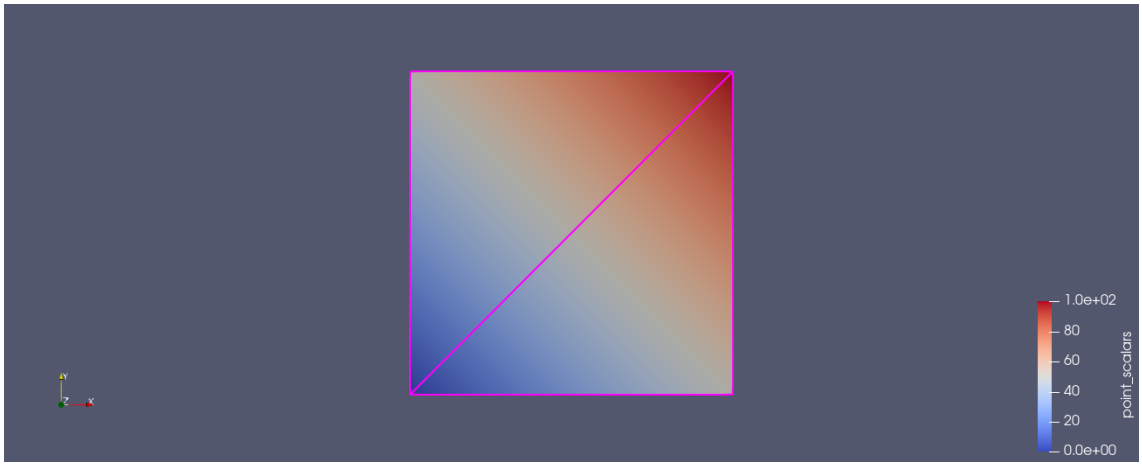
0

50

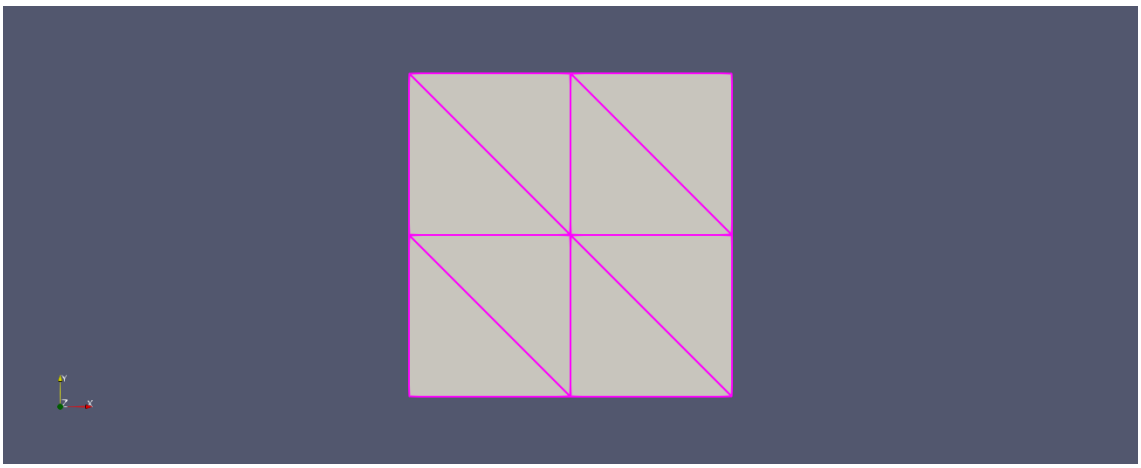
50

100

・可視化



正方形メッシュ 2



・プログラムの計算結果

t_array =

0

37.5000

50.0000

37.5000

50.0000

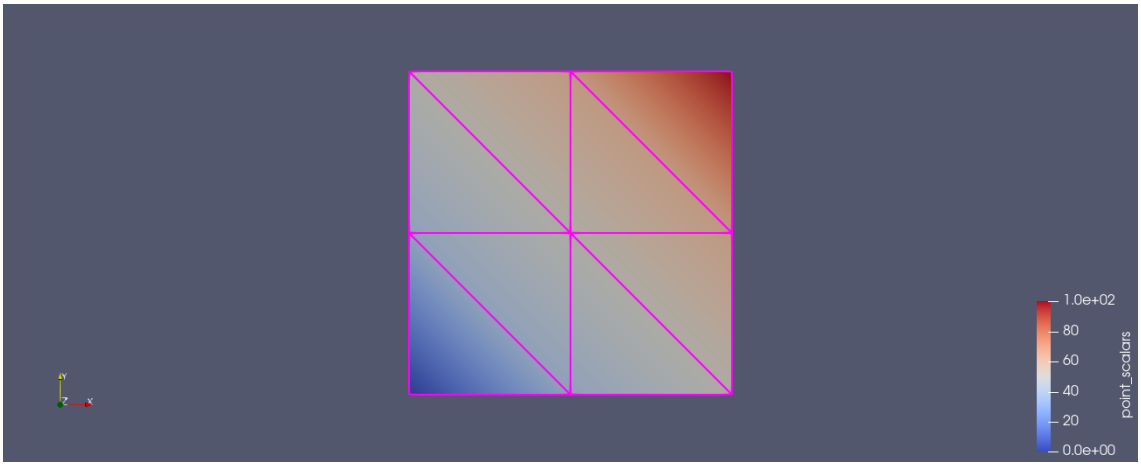
62.5000

50.0000

62.5000

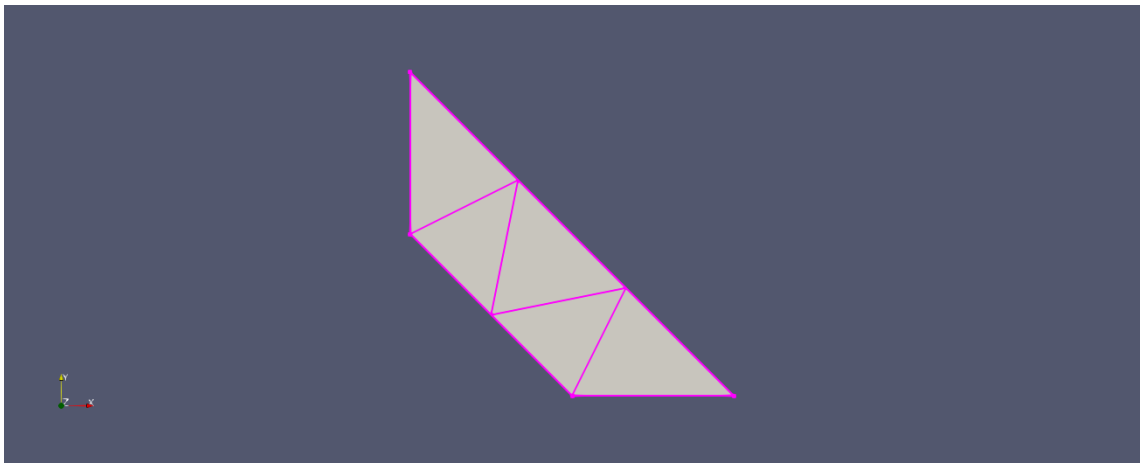
100.0000

・可視化



(2) 複雑形状に対して解像度の異なるメッシュを複数作成し、計算 せよ。計算結果を用いて計算時間と計算精度について考察せよ。

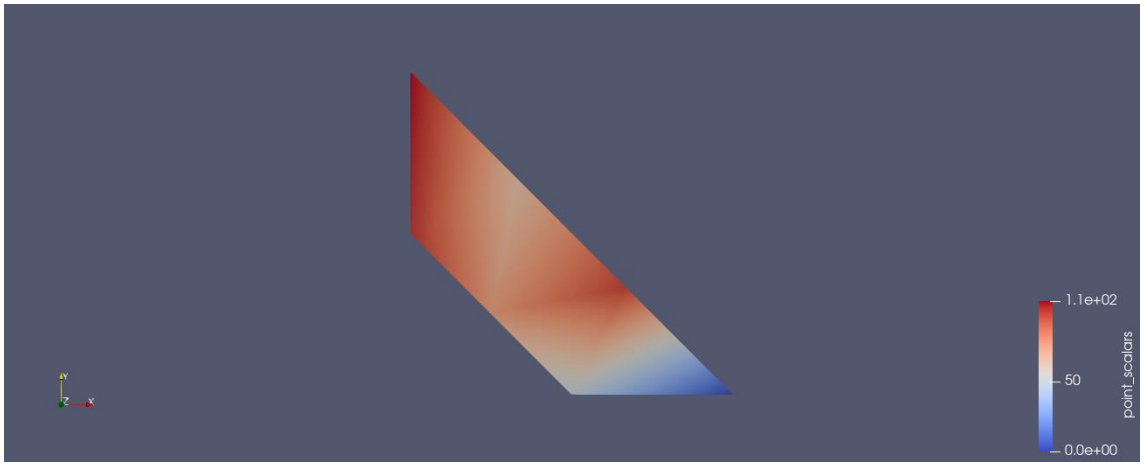
・メッシュ 1(gmesh を用いて作成)



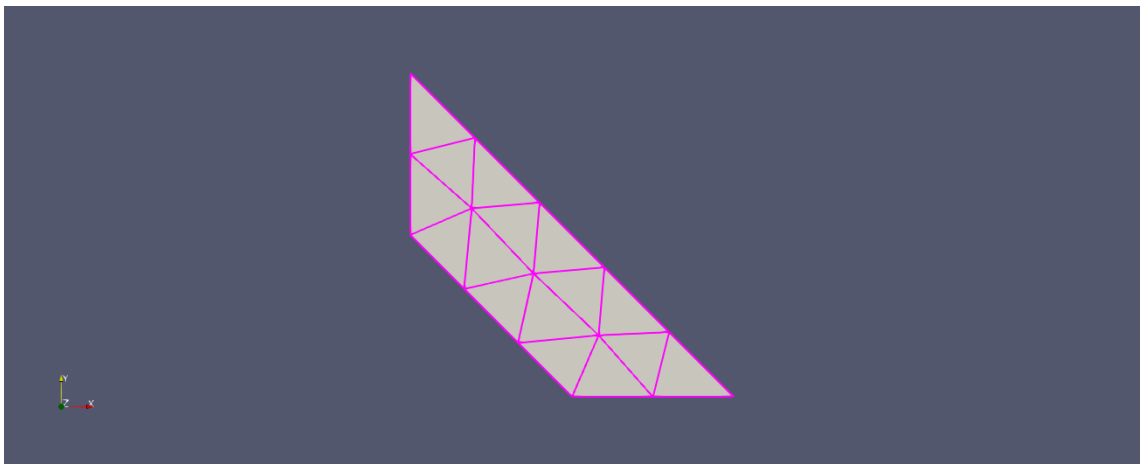
・プログラムの計算結果

t_array = 計算時間＝0.0034(大体毎回このくらいのオーダー)

46.6563
0
100.0000
106.6875
75.4359
97.6668
62.1043



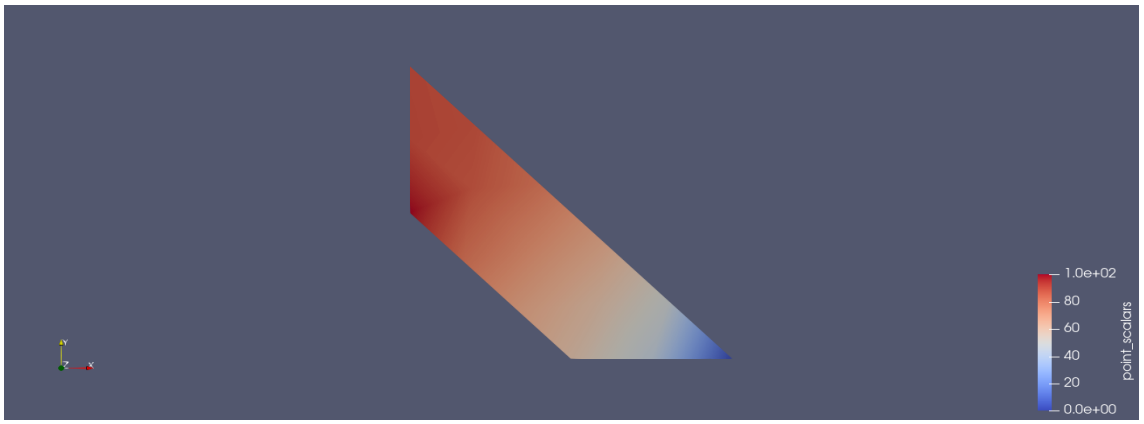
・メッシュ 2（解像度がメッシュ 1 より高い）



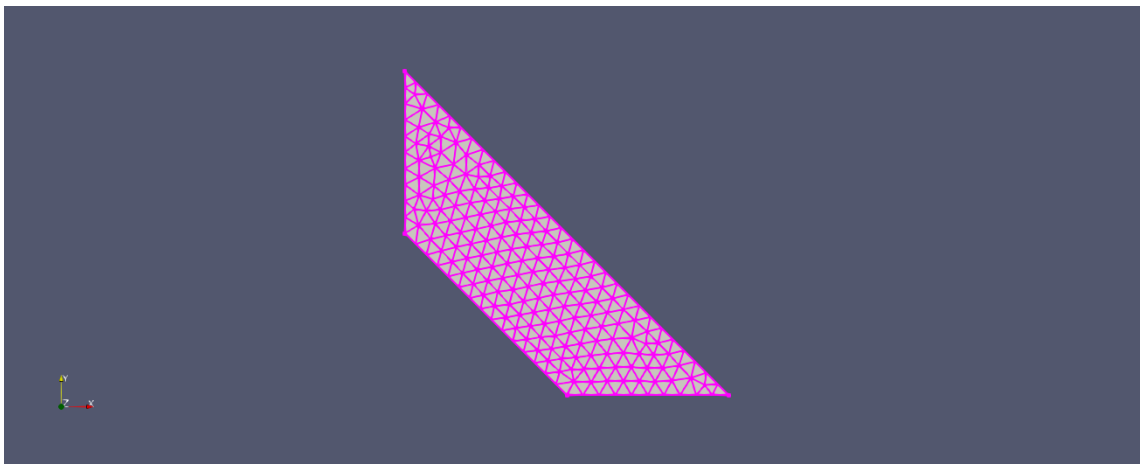
・プログラムの計算結果

t_array =（一部抜粋！！）　　計算時間= 0.0088（大体このオーダー）

57.5346
0
100.0000
89.3084
80.6177
67.5928
42.9417
47.1840
65.5218
80.0980
88.2776
89.7379
73.3307
87.0796…



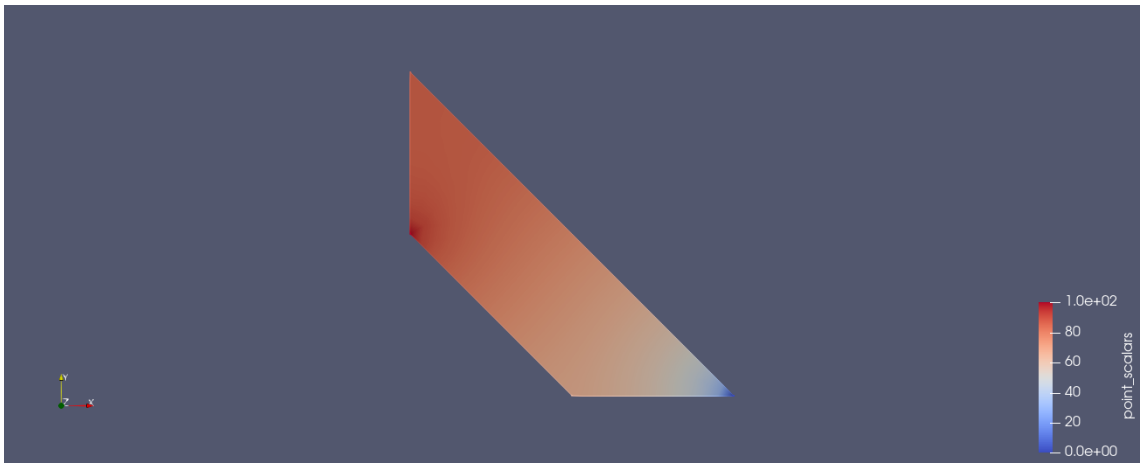
・メッシュ 3（メッシュ 1,2 よりも解像度が高い）



・プログラムの計算結果（入らないので一部抜粋）

t_array =　　計算時間＝0.0157（大体このオーダー）

64.1669
0
100.0000
84.8181
90.7485
86.3846
83.5463
81.2698
79.2597
77.3985
75.6298



（２）考察

・計算時間について

各メッシュの計算時間の部分を見ればわかるが、やはりメッシュが複雑になるほど計算時間が増えている。しかし計算時間はその時のパソコンの状態にも依存するので、一概に自分が計測した時間が正しいとは言えない。各メッシュで1000回くらいプログラムを動かして、平均時間を出せばそれなりに正確な考察ができると思う。

・計算精度について

メッシュが複雑になればなるほど、計算精度が高くなっていることがわかる。メッシュ 1 やメッシュ 2 を見ると、メッシュの線に沿って温度が同じになっているように見えるが、メッシュ３は解像度が高いので、ごく自然に温度の分布が計算されていることがわかる。

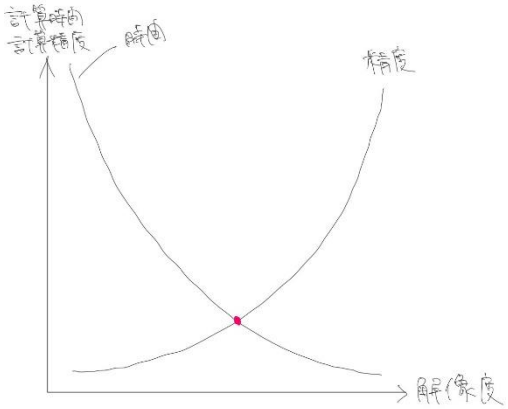
(3) 計算精度をよくするにはどうすればよいかを考えて意見を述べよ。

(2)からもわかるようにメッシュを細かくして、解像度を高めることにより、計算精度は上がり、ごく自然な分布が作れると思う。

(4) 計算時間を短くするためにはどうすればよいか考えて意見を述べよ。

(2)からわかるように単純に計算時間を短くしたいのなら、メッシュのサイズを大きくし、解像度を低くすればよいと思う。しかし、現実にはそうはいかないので、解析のアルゴリズムを余分のない、よりスマートな物にするしかないと思う。中でも連立方程式の求解に多くの時間が使われるので、より良いアルゴリズムを使うとよいだろう。

(3)(4)の考察



上の図からわかるように計算精度と計算時間の両方を高めることは難しい。よって、どっちに比重を置くかによって妥協する点を見つけるしかないと思う。

・プログラムの紹介

プログラムの説明は、コードの中にあります。

```
%時間計測スタート
start_time = tic;
%vtkファイルを少し変えて読み込みやすいテキストファイルにし、読み込み
M = importdata('C:\Users\81801\Desktop\計算機3\Programing final\readgnesh3.txt');
num_nodes= M(1,1)              %節点数
num_elements = M(1,2)          %要素数
nodes = zeros(num_nodes,3);    %節点と要素行列の初期化
elements = zeros(num_elements,3);
for i = 2:num_nodes+1          %節点行列作成
    nodes(i-1,1) = M(i,1);
    nodes(i-1,2) = M(i,2);
    nodes(i-1,3) = M(i,3);
end
for i = 1:num_elements         %要素行列作成
    elements(i,1)= M(num_nodes+1+i,1);
    elements(i,2)= M(num_nodes+1+i,2);
    elements(i,3)= M(num_nodes+1+i,3);
end
bc_nodes = [1,2,3,4];          %境界条件の設定
bc_values = [0,100,0,100];
nodes
elements

%Ke,K,T,右辺ベクトルの初期化
ke = zeros(3,3);
k = zeros(num_nodes,num_nodes);
t_array = zeros(num_nodes,1);
output_array = zeros(num_nodes,1);


%各三角形の係数行列の計算
```

```
 for p = 1:num_elements
        v1 = elements(p,1)+1;
        v2 = elements(p,2)+1;
        v3 = elements(p,3)+1;

        x1 =nodes(v1,1);
        x2 =nodes(v2,1);
        x3 =nodes(v3,1);
        y1 =nodes(v1,2);
        y2 =nodes(v2,2);
        y3 =nodes(v3,2);

        a1=y2-y3;
        a2=y3-y1;
        a3=y1-y2;
        b1=x3-x2;
        b2=x1-x3;
        b3=x2-x1;
        s=(x1*y2+x2*y3+x3*y1-x1*y3-x2*y1-x3*y2)/2;
        %Keベクトルの作成
        ke = [a1*a1 + b1*b1 a1*a2+b1*b2 a1*a3+b1*b3;a2*a1+b2*b1 a2*a2+b2*b2 a2*a3+b2*b3;a3*a1+b3*b1 a3*a2+b3*b2 a3*a3+b3*b3]/(4*s);
        %Keから各値をとりKに代入
        for i = 1:3;
            for j = 1:3;
                k(elements(p,i)+1,elements(p,j)+1) = k(elements(p,i)+1,elements(p,j)+1) + ke(i,j);
            end
        end
 end
 %境界条件に合わせるためにKベクトルに0,1を追加
 for i= 1:num_bcs;
     output_array(bc_nodes(i)+1) = bc_values(i);


    for p = 1:num_nodes;
        if p==bc_nodes(i)+1
            k(p,p) = 1;
        else
            k(bc_nodes(i)+1,p) = 0;
        end
    end
end
%T（温度）の計算
t_array = inv(k)*output_array
%時間計測終了
end_time = toc(start_time)
disp(end_time)
```