ISSN 2616-5775 Vol. 4, Issue 1: 35-43, DOI: 10.25236/AJCIS.2021.040106

Numerical simulation of biological network

Alexander Liang

The Lawrenceville School, Lawrenceville, NJ 08648, USA
Email: 1052231061@qq.com

Abstract: *Nowadays, more and more unmanned aerial vehicles are used in the logistics industry. Yet these delivery drones are heavy, needing to be charged very often and can't transport many packages. We hope to solve these issues by reducing the weight of wings and slightly increasing the size of the drone by using elastic transportation network based on energy optimization. This would help extend the work time of these drones and allowing for maximum transportation amount. We first figured out the total elastic potential energy of the wings and the gravity potential energy. Then we minimized the total energy to find the best geometric value (radius of each pipelines within the wings). Next simulated on MATLAB, wings were built according to the pipeline wing radius and the total energy result was updated continuously to constantly minimize the energy with a more ideal pipeline radiuses. We expect a final pipeline elastic structure similar to that of a dragonfly's wing with minimum total energy that will significantly help reduce the weight of large drone wings. This project will help design significantly lighter wings of drones based on elastic transportation network mainly through MATLAB simulations. Drones can then be expanded in size to transport more products and fly for longer periods.*

Keywords: *network, simulation, transportation*

1. Introduction

The wings of a large unmanned drone are sturdy and can withstand a decent amount of impact, but they cannot allow for long flight time. To try and solve this problem, scientists usually turn to look for new adequate materials as the solution for a lighter yet sturdy drone wing.

But this method is tedious and unsuccessful; for it would perhaps take even decades to find the ideal material, too long in the world of developing technology. However, the wings of dragonflies are light, allowing for long periods of flight. But the dragonfly wing can't withstand substantial impact/damage like a large drone's wings can. Airplane wings and dragonfly wings have features that can complement and make up for each other's shortcomings. Thus, we think by using existing materials to make a network-like unmanned aircraft wing similar to that of a dragonfly's wings, we can create wings that find a better blend between sturdiness and flight time. The idea is not to replicate exactly how a dragonfly's wing is but draw inspiration from transportation networks like a dragonfly wing.

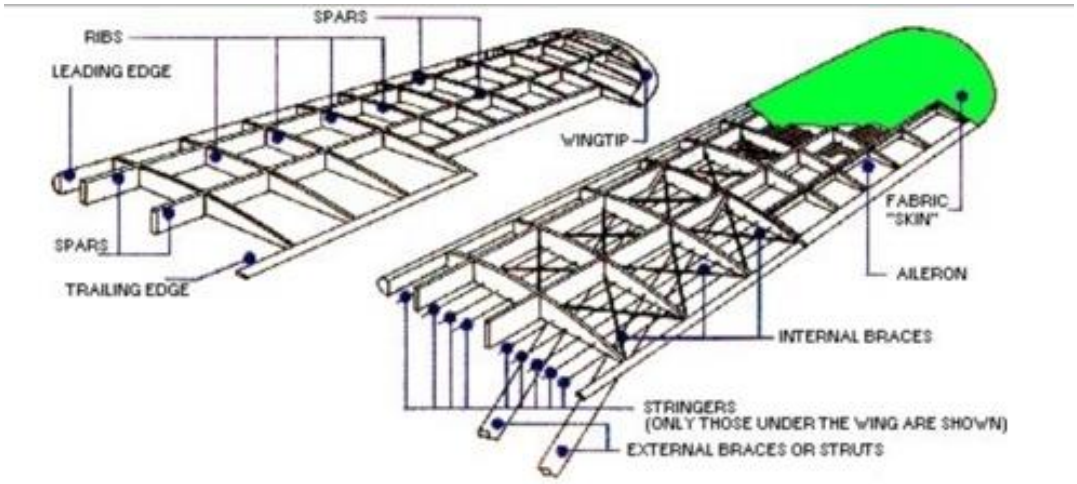

Figure 1: The structure of the wings of the airplane

ISSN 2616-5775 Vol. 4, Issue 1: 35-43, DOI: 10.25236/AJCIS.2021.040106

Figure 2: The shape of global hawk

To achieve this, we must find out the dynamics of the evolution of the wings of the dragonfly.

As the technology of unmanned aerial vehicles (drones) develops, more and more of them are used in daily tasks such as transporting packages and aerial photography. However, drones tasked with carry heavier objects small drones can't are currently very ineffective and inefficient. As shown on the right, the best-unmanned aerial vehicle currently is the DJI, which can only transports light objects like food, cellphones, and so on. But even for its relatively small size, a change or renewal in power is still needed every 30 minutes. This is very inconvenient and needs a constant replacement of batteries. But if we want to transport a greater amount of or heavier objects like washing machines, we need large unmanned aircraft like the Global Hawk. Even though the Global Hawk can fly very far, it is notably heavy; requiring large batteries only the military has steady access to. If we can redesign the wings (shown in the cross-sectional examination photo above) for large-style unmanned aircraft successfully, we can make the drone wings significantly lighter. This would allow for more accessible materials and sources of energy to be able to power the Global Hawk and other large drones like it more effectively. Doing so, that would not only make using large heavy drones more viable, but also allow for more efficient transporting.

The idea of my research comes inspired by dragonfly wings comes from the transportation network evolution of slime molds, the vessel system, and that of leaves. Even though the transportation network is different from the elastic network I will be using in my research to construct unmanned drone wings, they both have similar physical dynamics like energy optimization. The application of energy optimization on the transportation network shows that there must be a very simple and direct way to get the ideal network structure, even for elastic networks (I will explain that in the introduction of the energy optimization later in the research plan).

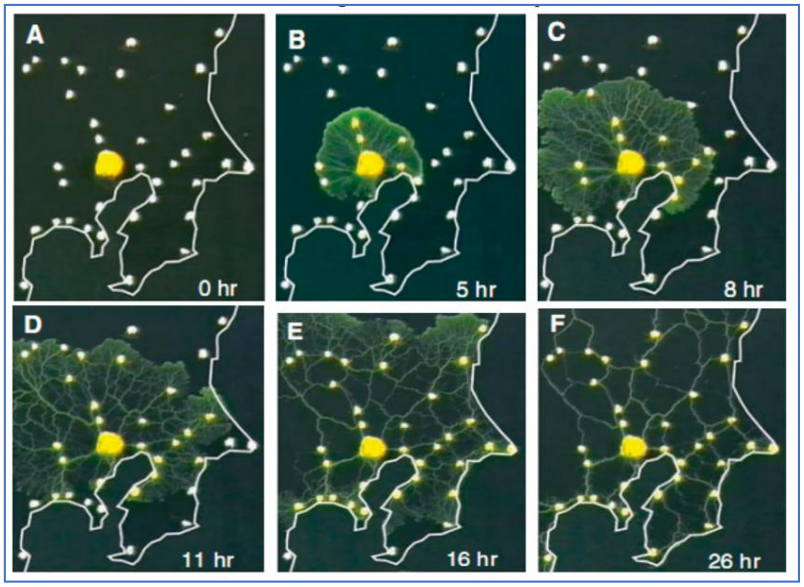

Figure 3: The network formed by the slime

ISSN 2616-5775 Vol. 4, Issue 1: 35-43, DOI: 10.25236/AJCIS.2021.040106

Previously, scientists designed a miniature version of the Tokyo subway network. The slime mold is placed in the center of Tokyo, and the food is placed in various subway stations in Tokyo with heavy traffic. Over time, the slime mold gradually evolved a transportation network similar to the Tokyo subway network. These phenomena show that for low-level single-celled organisms, there is a very simple mechanism that allows them to evolve network structures aimed at a certain goal, which is similar to the function of more intelligent organisms.

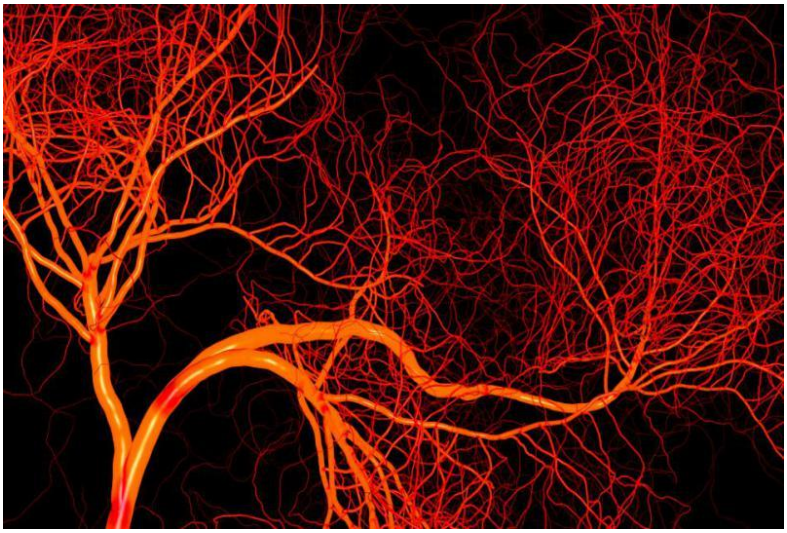

Figure 4: The structure of the blood system

The diagram on the right describes this process. The largest yellow dot in Figure A is the city center where the slime mold is located. The small yellow dots are the subway stations where food is placed. Over time, slime bacteria will evolve into a transportation network similar to the Tokyo subway network.

As for the human body, a more intelligent organism, there is a blood vessel network. Blood flows out of the arteries of the heart, through the organs and branches of the arteries, transporting nutrients (oxygen, protein, etc.) to corresponding organs and at the same time, taking away the waste materials of cell metabolism (carbon dioxide, etc.). The structural basis for the successful operation of this mechanism is the blood vessel network in the human body.

As is shown on the right, Arteries carry oxygenated blood away from the heart.

They’re tough on the outside but contain a smooth interior layer of epithelial cells that allows blood to flow easily. Arteries also contain a strong, muscular middle layer that helps pump blood through the body.

For both low and high intelligence level organisms, transportation network evolution occurs. Yet how are these vascular networks formed, such as the human blood vessel system? Some previous molecular biology studies believed that genes control the growth of blood vessel networks, but this explanation is not convincing because in the process of human growth, there will always be some accidents (such as falls, or genetic mutations), which are impossible to receive genetic control. Once a blood vessel is injured in the process of growth and development, organisms will always grow new blood vessels to replace the original blood vessels. Genes cannot control these risks. There are some other studies that believe that neural networks control the growth of blood vessels, because neural networks control many functions during the growth and development of the human body. But this is also not convincing. Facts have proved that the basic blood vessel network forms before the emergence of neural networks. Therefore, neural networks cannot control the initial stage of blood vessel growth and evolution.

Perhaps Mulberry leaves can hint at how a transportation network similar to that of a human’s blood vessel system is constructed. Mulberry leaves do not have advanced neural networks, but they also evolve a tree-like network structure like human blood vessels do during the growth process. The growth of a tree-like network structure at the initial stage of the growth process indicates that there is a simple and universal way to allow organisms to evolve a tree-like network structure. Our goal is to find a universal mechanism to simulate the process of biological evolution of the network structure, so that (if possible) a new network structure based on elasticity can be designed for the wing of an unmanned

ISSN 2616-5775 Vol. 4, Issue 1: 35-43, DOI: 10.25236/AJCIS.2021.040106

aircraft where the wings can be sturdy and reasonably light.

2. Research questions

1) How do we find the dynamic equation (expressed as a differential equation) satisfied by evolution using minimum energy principle?

2) How do we create a method of evolution of the network based on the differential equation?

3) How do we accelerate the computation of solving the equations obtained in the previous steps?

4) How can we get the elastic mechanism-based network using the previous work?

3. Goals and expected outcomes

The goal of this project is to establish a method to simulate the evolution of biological network structure so as to have a deeper understanding of the adaptive process in nature. In this project, we want to try a new method by applying the process of network optimization to see if the initial network can evolve into a tree structure (in nature, tree-shaped network structure is the most common). Because the unique branch structure of a tree network structure is able to adapt to the function of network transportation of substances to each small monomer or sub-network, if we can accomplish the above tasks, then we can design the most reasonable networked wing (light enough but hard enough) based on the weight and speed of the unmanned aircraft. We expect to create a dragonfly-like (in terms of visual design) elastic network of a drone wing with minimum elastic potential energy and gravity potential energy.

4. Numerical Simulation of Constrained Problem

4.1 The gradient flow solution method of the constraint problem

Recall the model of the constraint problem:

Minimize $f = \sum_{(i,j)} \frac{Q_{ij}(C_{ij})^2}{C_{ij}}$

St. $\sum_{(i,j)} C_{ij}^{\gamma} = K$, $Q_{ij}\left(C_{ij}\right)$ meas Q_{ij} is function of all of the C_{ij}, and is determined by

$Q_{ij} = C_{ij}(p_i - p_j)$.

First, let's introduce the principle of transforming general optimization problems into gradient flow problems.

Theorem: If there is a multivariate function $f(x_1, x_2, ..., x_n)$, use $\vec{x}$ represent vector $(x_1, x_2, ..., x_n)$ and consider a negative gradient flow system $\frac{d\vec{x}}{dt} = -\left(\nabla f(\vec{x})\right)$:. Then along any set of solutions of the system, the function $\varphi(t) = f(x_1(t), x_2(t), ..., x_n(t))$ is a decreasing function, and the equilibrium point $f(x_1, x_2, ..., x_n)$ of the system is the minimum point.

Proof: According to the chain rule of derivation, $\varphi'(t) = \left(\nabla f(\vec{x})\right)\frac{d\vec{x}}{dt} = -\left(\nabla f(\vec{x})\right)'\left(\nabla f(\vec{x})\right)$, $\varphi'(t) \le 0$, so $\varphi(t)$ is a decreasing function. And, if there is at a certain point, x_0, $\frac{d\vec{x_0}}{dt} = 0$, then $\nabla f(\vec{x_0}) = 0$, cause $\varphi(t)$ is a decreasing function, the equilibrium point can only be a minimum

ISSN 2616-5775 Vol. 4, Issue 1: 35-43, DOI: 10.25236/AJCIS.2021.040106

point, not a maximum point, and the proof is complete.

According to this theorem, we define Lagrange function

$$F_\lambda(C_{ij}) = \sum_{(i,j)} \frac{Q_{ij}(C_{ij})^2}{C_{ij}} + \lambda\left(\sum_{(i,j)} C_{ij}^{\ \gamma} - K\right),$$

because $\sum_{(i,j)} C_{ij}^{\ \gamma} - K = 0$, for a random value λ, $F_\lambda(C_{ij}) = f(C_{ij})$, so for any value λ, as long as $\sum_{(i,j)} C_{ij}^{\ \gamma} - K = 0$ and decrease $F_\lambda(C_{ij})$, we can get the point.

Let's calculate $\nabla F_\lambda(C_{ij})$:

$$\frac{\partial F_\lambda(C_{ij})}{\partial C_{ij}} = -\frac{Q_{ij}(C_{ij})^2}{C_{ij}^{\ 2}} + \sum_{(i,j)} \frac{2Q_{ij}(C_{ij})}{C_{ij}} \frac{\partial Q_{ij}}{\partial C_{ij}} + \lambda\gamma C_{ij}^{\ \gamma-1}$$

$$= -(p_i - p_j)^2 + 2\sum_{(i,j)} (p_i - p_j) \frac{\partial Q_{ij}}{\partial C_{ij}} + \lambda\gamma C_{ij}^{\ \gamma-1}$$

$$= -(p_i - p_j)^2 + \lambda\gamma C_{ij}^{\ \gamma-1} + 2\sum_i \sum_j p_i \frac{\partial Q_{ij}}{\partial C_{ij}} - 2\sum_i \sum_j p_j \frac{\partial Q_{ij}}{\partial C_{ij}}$$

$$= -(p_i - p_j)^2 + \lambda\gamma C_{ij}^{\ \gamma-1} + 2\sum_i \sum_j p_i \frac{\partial Q_{ij}}{\partial C_{ij}} - 2\sum_i \sum_j p_i \frac{\partial Q_{ji}}{\partial C_{ji}}$$

$$= -(p_i - p_j)^2 + \lambda\gamma C_{ij}^{\ \gamma-1} + 4\sum_i p_i \sum_j \frac{\partial Q_{ij}}{\partial C_{ij}}$$

For all of the value of i, $\sum_j Q_{ij}$ is a constant, so $\sum_j \frac{\partial Q_{ij}}{\partial C_{ij}} = 0$, and then $\frac{\partial F_\lambda(C_{ij})}{\partial C_{ij}} = -(p_i - p_j)^2 + \lambda\gamma C_{ij}^{\ \gamma-1}$.

We define d as a $\left(3N^2 + 3N + 1\right) \times \left(3N^2 + 3N + 1\right)$ matrix. Its row i and column j is $d_{ij} = (p_i - p_j)^2 - \lambda\gamma C_{ij}^{\ \gamma-1}$, so we have $\frac{\partial F_\lambda(C_{ij})}{\partial C_{ij}} = -d_{ij}$, and then we can find $\nabla F_\lambda(C_{ij})$. On the bases, if we multiply d_{ij} by a factor $C_{ij}^{\ \beta}$, thus we use $d_{ij} C_{ij}^{\ \beta} = \left((p_i - p_j)^2 - \lambda\gamma C_{ij}^{\ \gamma-1}\right) C_{ij}^{\ \beta}$ replace $d_{ij} = (p_i - p_j)^2 - \lambda\gamma C_{ij}^{\ \gamma-1}$, use the therom $\varphi'(t) = -\sum_{(i,j)} d_{ij}^{\ 2} C_{ij}^{\ \beta} \le 0$, is decreasing , so we can get the $d_{ij} = 0$. The reason for this treatment here is that we want to find an optimal gradient flow of this form, that is, when the value is taken, the energy of the equilibrium point obtained by the gradient flow is the lowest. If there is no other special instructions in the following, all will be taken.

In summary, we transform the original minimum problem into the following negative gradient flow problem.

Minus Gradient Problem: Solve the system of ordinary differential equations

ISSN 2616-5775 Vol. 4, Issue 1: 35-43, DOI: 10.25236/AJCIS.2021.040106

$$\frac{dC_{ij}}{dt} = d_{ij} = \left(-(p_i - p_j)^2 + \lambda\gamma C_{ij}^{\gamma-1}\right) C_{ij}^{\beta} \quad \cdots\cdots(1)$$

$$st. \sum_{(i,j)} C_{ij}^{\gamma} - K = 0 \qquad \sum_{(i,j)} \left(C_{ij}^{(n)}\right)^{\gamma} - K = 0 \qquad \sum_{(i,j)} \left(C_{ij}^{(n)} + ud_{ij}^{(n)}\right)^{\gamma} - K = 0$$

use Taylor expansion

$$\sum_{(i,j)} \left(C_{ij}^{(n)} + ud_{ij}\right)^{\gamma} = \sum_{(i,j)} \left(C_{ij}^{(n)}\right)^{\gamma} + ud_{ij}\gamma\left(C_{ij}^{(n)}\right)^{\gamma-1} + O(u^2) = K \text{。}$$

So we can get

$$\sum_{(i,j)} ud_{ij}\gamma\left(C_{ij}^{(n)}\right)^{\gamma-1} = 0, \quad i.e. \sum_{(i,j)} \left(-(p_i^{(n)} - p_j^{(n)})^2 + \lambda^{(n)}\gamma C_{ij}^{(n)\gamma-1}\right)\left(C_{ij}^{(n)}\right)^{\beta+\gamma-1} = 0$$

$$\lambda^{(n)} = \frac{(p_i^{(n)} - p_j^{(n)})^2 \left(C_{ij}^{(n)}\right)^{\beta+\gamma-1}}{\gamma\left(C_{ij}^{(n)}\right)^{\beta+2\gamma-2}} \quad \cdots\cdots(2)$$

Finally, we give the flow of the numerical solution, as shown in the figure below.

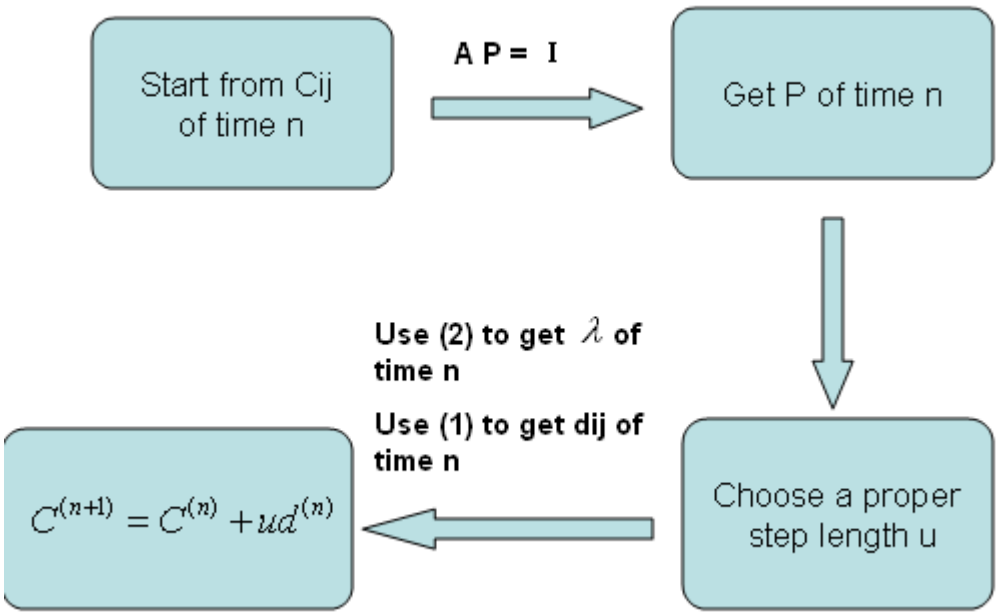

Figure 5: The steps of computation

4.2 Several acceleration methods

The above numerical solution time is mainly consumed in solving the equations AP=I. In order to facilitate the calculation of larger N and the data statistics for different situations, it is necessary to shorten the calculation time of the program. The methods used in this article to shorten the time mainly include The following three:

4.2.1 Sparse matrix

There is a very convenient way to deal with sparse matrices in Matlab. Since the order of A is, the number of elements in A is, but each row has at most 7 non-zero elements, so the number of non-zero elements is, so A is A very sparse matrix. Before using the conjugate gradient method to solve the equation group AP=I in matlab, inputting the A=sparse(A) command will greatly shorten the solving time of the equation.

ISSN 2616-5775 Vol. 4, Issue 1: 35-43, DOI: 10.25236/AJCIS.2021.040106

In matlab, take N=30 to solve AP=I as an example, the code is as follows:

```
N=30;
D=ones(3*N^2+3*N+1);
D=hexagon_initialD(D);
p=zeros(3*N^2+3*N+1,1);
b=-ones(3*N^2+3*N+1,1);
b(1)=0;
A=diag(sum(D))-D;
A(1,1)=1;
    for i=2:(3*N^2+3*N+1)
        A(i,1)=0;
        A(1,i)=0;
    end
A=sparse(A);
tic
    r=A*p-b;
    x=b-A*p;
    while (b-A*p)'*(b-A*p)>0.00001
        y=(r'*r)/(x'*A*x);
        p=p+y*x;
        temp=r+y*A*x;
        x=-temp+x*(temp'*temp)/(r'*r);
        r=temp;
    end
toc
```

The code indicated in red is the command to define A as a sparse matrix. The output results with and without this code are:

```
elapsed_time =
    0.0117s
elapsed_time =
   0.9478s
```

As you can see, the acceleration effect of using the sparse matrix is huge!

4.2.2 Use the solution of the previous step

When the entire system gradually stabilizes, the change of matrix C becomes smaller and smaller at each step, so the difference between the two steps before and after the equation group AP=I becomes smaller and smaller. So if each search with the conjugate gradient method starts from a fixed point, a lot of time will be wasted. In fact, if the solution of the previous step is used as the initial value of the next step to search, then the number of times to search for the equation system for each iteration in the entire iterative process will rapidly decrease, and the time required will gradually decrease. If the system of equations is solved from the same point in each iteration, the time consumed for each solution will gradually stabilize at a higher value.

ISSN 2616-5775 Vol. 4, Issue 1: 35-43, DOI: 10.25236/AJCIS.2021.040106

4.2.3 Runge-Kuta method

Runge-Kutta method is essentially an improvement of Euler's methodIf we write the right end of the above negative gradient flow as a function of C_{ij} $\frac{dC_{ij}}{dt} = h(C_{ij}) = \left(-(p_i - p_j)^2 + \lambda\gamma C_{ij}^{\gamma-1}\right) C_{ij}^{\beta}$, that is, the Euler method is the nth step. If we define the local truncation error as follows:：

Definition: The local truncation error of an algorithm is called, if we assume that the calculated value in the previous step is an accurate value, the magnitude of the error between the value obtained in the next step of the calculation and the accurate value is, where u is Stride.

$$\varepsilon = C_{ij}^{[n]} - C_{ij}^{(n)} = C_{ij}^{[n]} - C_{ij}^{(n-1)} - uh(C_{ij}^{(n-1)}) = C_{ij}^{[n]} - C_{ij}^{[n-1]} - uh(C_{ij}^{[n-1]}) = O(u^2)$$

If a Runge-Kuta method has n-order accuracy, then the method is called the n-1 order Runge-Kuta method.So the first-order Runge-Kuta method is Euler's method.

Below we mainly introduce the second-order Runge-Kuta method:

$$d_1^{(n)} = h(C_{ij}^{(n)}),$$

$$d_2^{(n)} = h(C_{ij}^{(n)} + pud_1), 0 < p < 1$$

$$C_{ij}^{(n+1)} = C_{ij}^{(n)} + u(\lambda_1 d_1 + \lambda_2 d_2)$$

$$\varepsilon = C_{ij}^{[n+1]} - C_{ij}^{(n+1)}$$

$$= C_{ij}^{[n+1]} - C_{ij}^{(n)} - u(\lambda_1 d_1 + \lambda_2 d_2)$$

$$= C_{ij}^{[n+1]} - C_{ij}^{[n]} - u(\lambda_1 h(C_{ij}^{[n]}) + \lambda_2 h(C_{ij}^{[n]} + pud_1))$$

$$= uh(C_{ij}^{[n]}) + \frac{1}{2}u^2 h'(C_{ij}^{[n]})h(C_{ij}^{[n]}) + O(u^3) - u(\lambda_1 h(C_{ij}^{[n]}) + \lambda_2 h(C_{ij}^{[n]} + pud_1))$$

$$= u\left[h(C_{ij}^{[n]}) - \lambda_1 h(C_{ij}^{[n]}) - \lambda_2\left(h(C_{ij}^{[n]}) + pud_1 h'(C_{ij}^{[n]}) + O(u^2)\right)\right] + \frac{1}{2}u^2 h'(C_{ij}^{[n]})h(C_{ij}^{[n]}) + O(u^3)$$

$$= u\left[h(C_{ij}^{[n]}) - \lambda_1 h(C_{ij}^{[n]}) - \lambda_2 h(C_{ij}^{[n]})\right] + u^2\left(\frac{1}{2}h'(C_{ij}^{[n]})d_1 - \lambda_2 pd_1 h'(C_{ij}^{[n]})\right) + O(u^3)$$

$$\begin{cases} \lambda_1 + \lambda_2 = 1 \\ \lambda_2 p = 0.5 \end{cases}$$

It can be seen that the second-order Runge-Kuta method has third-order accuracy, which is more accurate than Euler's format.

Below we do not add proof, give the internationally recognized classic fourth-order Runge-Kuta method:

$$d_1^{(n)} = h(C_{ij}^{(n)}),$$

$$d_2^{(n)} = h(C_{ij}^{(n)} + 0.5ud_1)$$

$$d_3^{(n)} = h(C_{ij}^{(n)} + 0.5ud_2)$$

$$d_4^{(n)} = h(C_{ij}^{(n)} + ud_3)$$

$$C_{ij}^{(n+1)} = C_{ij}^{(n)} + \frac{1}{6}u(d_1 + 2d_2 + 2d_3 + d_4)$$

ISSN 2616-5775 Vol. 4, Issue 1: 35-43, DOI: 10.25236/AJCIS.2021.040106

The above fourth-order Runge-Kuta method is the most commonly used and is also used in this article.

Because the Runge-Kuta method has higher precision, we can increase the step length of each step to achieve the purpose of reducing the number of iterations, which greatly accelerates the operation of the program.

5. Statistical results and conclusions

This chapter will introduce the results and conclusions of the numerical simulation. Unless otherwise specified, the following results are iterated from the initial value to the equilibrium state.

It is mentioned in the abstract that the value of γ directly affects the topological structure of the equilibrium state: when γ>1, it is a uniform sheet, and when γ<1, it is a tree structure. And γ=1 is the phase turning point between the two [2]. This section will take the constraint problem as an example, and use the negative gradient flow method and the alternating direction method for numerical verification.Take N=8, use the negative gradient flow method to solve the constraint problem, take γ as 0.5, 0.9, 1.1 and 1.5, respectively, to obtain different final equilibrium topological structures, as shown in Figure 5.1-1 and Figure 5.1-2.

It can be seen that there are many loops when γ=1.1 and 1.5, and when γ=0.5 and 0.9, it is a tree without loops, and the results meet the conclusion.

It can be seen that the ordinates of all points are positive, which means that whether it is a relatively small-scale constraint problem or a relatively large-scale unconstrained problem, the conclusion is almost unanimously obtained: the energy obtained by the negative gradient flow method is It is lower than the energy obtained by the alternating direction method.

In the process of numerical solution, it is found that the network structure of the equilibrium state obtained by the two methods is obviously different: the tree structure obtained by the negative gradient flow method is thick in the middle and diverging to both sides; while the tree structure obtained by the alternating direction method is obtained. The tree-like structure diverges from the beginning, with a thin middle and thick sides. The following is the results.

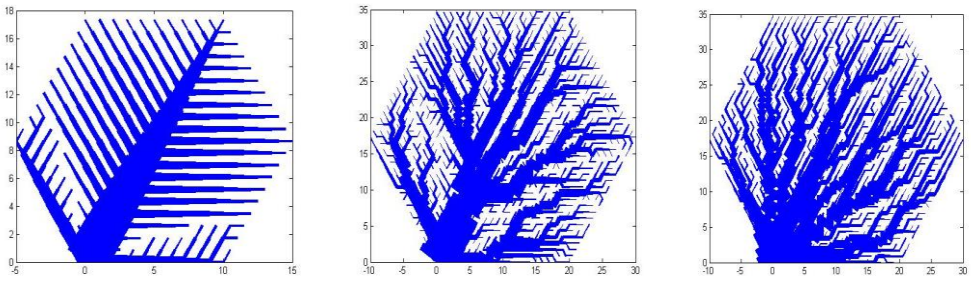

Figure 6: The network generated by numerical simulation

References

[1] Tero A, Takagi S, Saigusa T, et al. Rules for Biologically Inspired Adaptive Network Design[J]. Science, 2010, 327(5964): 439-442.
[2] Katifori E, Szollosi G J, Magnasco M O. Damage and Fluctuations Induce Loops in Optimal Transport Networks[J]. Physical Review Letters, 2010, 104(4): 048704.1-048704.4.
[3] Bohn S, Magnasco M O. Structure, Scaling, and Phase Transition in the Optimal Transport Network[J]. Physical Review Letters, 2007, 98(8): 088702.
[4] Tero A, Kobayashi R, Nakagaki T. A Mathematical Model For Adaptive Transport Network in Path Finding by True Slime Mold[J]. Journal of Theoretical Biology, 2007, 244(4): 553-564.
[5] Kurz H, Burri P H, Djonov V G. Angiogenesis and Vascular Remodeling by Intussusception: From Form to Function[J]. News Physiol Sci, 2003, 18(2): 65-70.