Mathematical Modeling of M_2 Tetramer Ion Channel using Numerical Analysis

Reuben D. Budiarda
Advisor: Professor Carl S. Helrich

23rd May 2002

Abstract

The M_2 protein ion channel regulates the pH of cell infected by influenza A virus. From previous studies, it is known that M_2 protein has a tetramic arrangement. It is also known that the current flow through the channel in the virus is 10,000 as large as expected from a single tetramer. Thus, a collective behavior was suspected. A model is then developed to see if this collective behavior is possible theoretically. A mozaic approach was undertaken, and a numerical solution was obtain to get a describtion of motion in the dynamic of each particle in tetramic arrangement. The numerical solution also revealed that a collective behavior with five tetramers is a possibility, which result in a larger channel.

1 Introduction

M_2 protein ion channel is a known pH regulator during the process of infestation by influenza A virus [2]. It is known the channel is formed by a tetramic arrangement of M_2 proteins linked by disulfide bonds [3]. It has been estimated that the a single channel can only pass current no larger than 0.005 pA. From bilayer experimentation however, current as large as 10,000 times as large as expected were observed. Thus, a collective behaviour of multiple channel were suspected.

The previous work for modeling of this collective behaviour includes a collisional model [4] and mozaic model [1]. The collisional model did not seem to represent the real data from observation. The mozaic approach agreed more with the data. In this paper, the mozaic approach is revisited. The original model used a simplification to try to get an analytical result. This model will include the non-linearity of the mozaic that was left out and then use numerical solution to get a result. The model seek to see if a natural mode in the cooperation of tetramers allow the possibility of a formation of a greater channel.

2 The Model

The M_2 ion channel tetramic arrangement can be represented by four particle. The disulfide bonds between the tetramers are represented by springs that has a quadratic potential with a spring constant k. Figure 1 shows this representation.

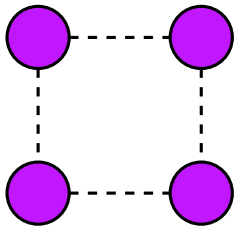

Figure 1: Tetramer representation used in the model. Each particle is connected by a spring that has the same spring constant k. The dashes line represent a spring.

Hamiltonian of the physical situation was obtained. The Hamiltonian is the total of kinetic and potential energy, since it is believed that in this situation the energy is conserved. The Hamiltonian is as follows.

$$\begin{aligned} H &= \frac{1}{2}\frac{\sum_{i=1}^{4} Px_i^2 + \sum_{i=1}^{4} Py_i^2}{m} + k\left(x_1^2 + x_2^2 + x_3^2 + x_4^2 + y_1^2 + y_2^2 + y_3^2 + y_4^2\right) \\ & k\left(x_1x_2 + x_2x_3 + x_3x_4 + x_4x_1 + y_1y_2 + y_2y_3 + y_3y_4 + y_4y_1\right) + 2 \cdot k \cdot L_0^2 \\ & k \cdot L_0\left(\sqrt{x_1^2 + x_2^2 + y_1^2 + y_2^2 - 2x_1x_2 - 2y_1y_2} + \sqrt{x_2^2 + x_3^2 + y_2^2 + y_3^2 - 2x_2x_3 - 2y_2y_3}\right) + \\ & k \cdot L_0\left(\sqrt{x_3^2 + x_4^2 + y_3^2 + y_4^2 - 2x_3x_4 - 2y_3y_4} + \sqrt{x_4^2 + x_1^2 + y_4^2 + y_1^2 - 2x_4x_1 - 2y_4y_1}\right) \end{aligned}$$

with x_μ and y_μ is the position of the μ^{th} particle, k is the spring constant, and L_0 is the length of the spring at equilibrium. The position (x_μ, y_μ) and momentum (Px_μ) was considered to be a function of time. From the Hamiltonian, the cannonical equations were obtained using the following equations:

$$\begin{aligned} \frac{\partial x_\mu(t)}{\partial t} &= \frac{\partial H}{\partial Px_\mu(t)} \\ \frac{\partial Px_\mu(t)}{\partial t} &= \frac{\partial H}{\partial x_\mu(t)} \end{aligned}$$

which gives us a system of differential equation. Since the differential equation and Hamiltonian is non-linear, numerical solution was used to obtain the position and momentum of each particle at any given time t by solving the system of differential equation. By putting in initial condition, one can observe if a suspected mode is really an natural mode for the vibration of the particles. If

the vibration dies off or differs from the initial condition, then it is not a natural mode, but if it continues, then it is a natural mode. From previous work [1], it is suspected that the initial conditions shown in figure 2 is a natural mode.

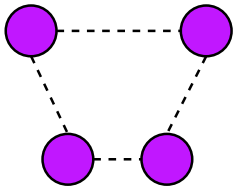

Figure 2: Initial condition for a suspected natural mode of vibration.

The numerical result from Maple as the position in x and y coordinate for each particle is then used to feed a program that gives a visualization of the movement of the particle. The source code of the program is included in the appendix. The following are some snapshot from the visualization program. The opening of the channel can be seen in the first and third picture in the snapshots below.

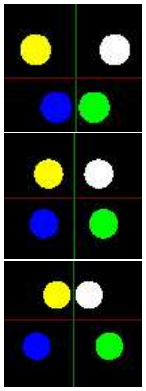

Figure 3: Snapshots of the vibration of the tetramer as time increases

The same procedure was then used to get a describtion of motion from 5 tetramers. Since it was suspected that the five tetramers has a collective behaviour, there should be a bond between each tetramers. Each tetramers was then connected using a spring with a spring constant k_2. The spring constant for each particle in a tetramer was renamed to k_1. k_2 is smaller than k_1, since the bond between tetramers is naturally weaker than the bond within particles in a tetramer. Figure 4 is a representation of the physical condition. The Hamiltonian and cannonical equation for this condition is included in the Maple file print out as an appendix.

The initial condition that was used is shown in figure 5. One can see that this initial condition gives a big opening of the channel through the middle of

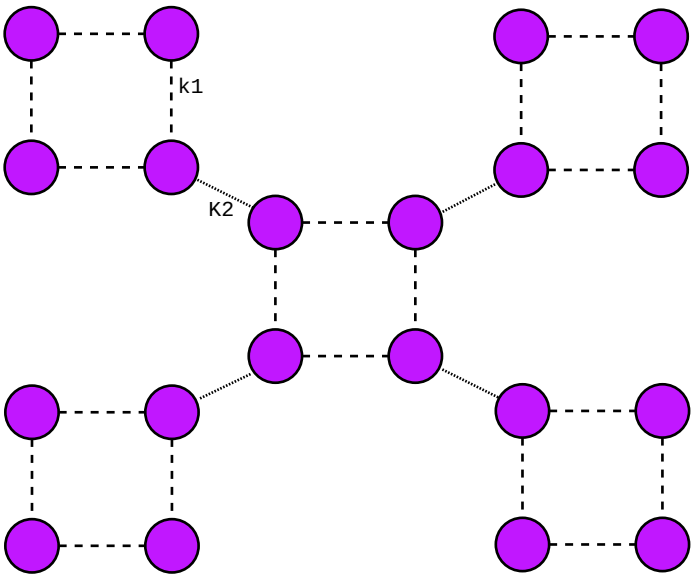

Figure 4: 5 tetramers connected together.

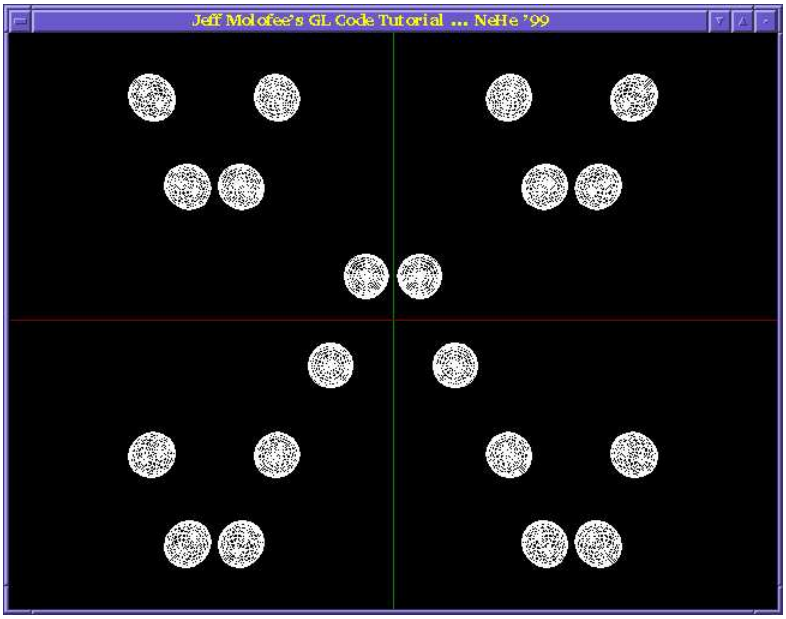

Figure 5: Initial condition with five tetramers. A big opening happens through the middle of the tetramers.

the tetramers. From the calculation, it was observed that this initial condition is a natural mode of the channel. Figure 6 is snapshots from the visualization program.

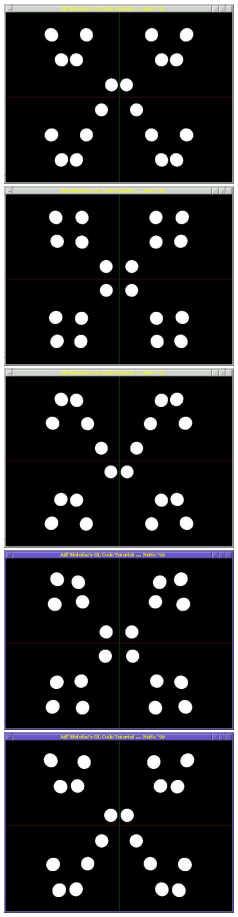

Figure 6: Five tetraters motion in the order of increasing time. The vibration continues.

3 Conclusion

From this result, one can observe that collective behavior is theoretically a possibility. This will allow the channel to pass a much larger current through than a single tetramer will. The model can be extended to have more than five tetramers theoretically, although the mathematics will become unbearably long if one were to obtain numerical solution.

4 Acknowledgement

The author gratefully acknowledge the help of Reynard Hilman in building the initial program structure that was then modified and used to visualize the vibration of tetramers.

A Appendix

A.1 Maple Worksheet for Five Tetramers Numerical Solution

```
> restart;
   > # V1 is the potential energy due to the interaction in the tetramers, and
have the spring constants
   > # k1
   > V1 := k1*(x1^2+x2^2+x3^2+x4^2+y1^2+y2^2+y3^2+y4^2)
   >     -k1*(x1*x2+x2*x3+x3*x4+x4*x1+y1*y2+y2*y3+y3*y4+y4*y1)
   >     +2*k1*L0^2
   >     -k1*L0*(sqrt(x1^2+x2^2+y1^2+y2^2-2*(x1*x2+y1*y2))
   >           +sqrt(x2^2+x3^2+y2^2+y3^2-2*(x2*x3+y2*y3))
   >           +sqrt(x3^2+x4^2+y3^2+y4^2-2*(x3*x4+y3*y4))
   >           +sqrt(x4^2+x1^2+y4^2+y1^2-2*(x4*x1+y4*y1)))  +
   >
   >      k1*(x5^2+x6^2+x7^2+x8^2+y5^2+y6^2+y7^2+y8^2)
   >     -k1*(x5*x6+x6*x7+x7*x8+x8*x5+y5*y6+y6*y7+y7*y8+y8*y5)
   >     +2*k1*L0^2
   >     -k1*L0*(sqrt(x5^2+x6^2+y5^2+y6^2-2*(x5*x6+y5*y6))
   >           +sqrt(x6^2+x7^2+y6^2+y7^2-2*(x6*x7+y6*y7))
   >           +sqrt(x7^2+x8^2+y7^2+y8^2-2*(x7*x8+y7*y8))
   >           +sqrt(x8^2+x5^2+y8^2+y5^2-2*(x8*x5+y8*y5)))   +
   >
   >      k1*(x9^2+x10^2+x11^2+x12^2+y9^2+y10^2+y11^2+y12^2)
   >    -k1*(x9*x10+x10*x11+x11*x12+x12*x9+y9*y10+y10*y11+y11*y12+y12*y9)
   >     +2*k1*L0^2
   >     -k1*L0*(sqrt(x9^2+x10^2+y9^2+y10^2-2*(x9*x10+y9*y10))
   >           +sqrt(x10^2+x11^2+y10^2+y11^2-2*(x10*x11+y10*y11))
   >           +sqrt(x11^2+x12^2+y11^2+y12^2-2*(x11*x12+y11*y12))
   >           +sqrt(x12^2+x9^2+y12^2+y9^2-2*(x12*x9+y12*y9)))  +
   >
   >      k1*(x13^2+x14^2+x15^2+x16^2+y13^2+y14^2+y15^2+y16^2)
   >    -k1*(x13*x14+x14*x15+x15*x16+x16*x13+y13*y14+y14*y15+y15*y16+y16*y13)
   >     +2*k1*L0^2
   >     -k1*L0*(sqrt(x13^2+x14^2+y13^2+y14^2-2*(x13*x14+y13*y14))
   >           +sqrt(x14^2+x15^2+y14^2+y15^2-2*(x14*x15+y14*y15))
```

```
>          +sqrt(x15^2+x16^2+y15^2+y16^2-2*(x15*x16+y15*y16))
>          +sqrt(x16^2+x13^2+y16^2+y13^2-2*(x16*x13+y16*y13)))  +
>
>     k1*(x17^2+x18^2+x19^2+x20^2+y17^2+y18^2+y19^2+y20^2)
>   -k1*(x17*x18+x18*x19+x19*x20+x20*x17+y17*y18+y18*y19+y19*y20+y20*y17)
>    +2*k1*L0^2
>    -k1*L0*(sqrt(x17^2+x18^2+y17^2+y18^2-2*(x17*x18+y17*y18))
>          +sqrt(x18^2+x19^2+y18^2+y19^2-2*(x18*x19+y18*y19))
>          +sqrt(x19^2+x20^2+y19^2+y20^2-2*(x19*x20+y19*y20))
>          +sqrt(x20^2+x17^2+y20^2+y17^2-2*(x20*x17+y20*y17))):
>
> # V2 is the potential energy due to the interaction between tetramers,
and have the spring constants
> # k2
> V2 := 1/2*k2*(x4^2+x6^2+x7^2+x9^2+x8^2+x14^2+x5^2+x19^2+
>              y4^2+y6^2+y7^2+y9^2+y8^2+y14^2+y5^2+y19^2)
>    -k2*(x4*x6+x7*x9+x8*x14+x5*x19+y4*y6+y7*y9+y8*y14+y5*y19)
>     +2*k2*J0^2
>     -k2*J0*(sqrt(x4^2+x6^2+y4^2+y6^2-2*(x4*x6+y4*y6))
>          +sqrt(x7^2+x9^2+y7^2+y9^2-2*(x7*x9+y7*y9))
>          +sqrt(x8^2+x14^2+y8^2+y14^2-2*(x8*x14+y8*y14))
>          +sqrt(x5^2+x19^2+y5^2+y19^2-2*(x5*x19+y5*y19))):
>
> T := 1/(2*m) * (Px1^2+Px2^2+Px3^2+Px4^2+Px5^2+Px6^2+Px7^2+Px8^2+Px9^2+Px10^2
>          +Px11^2+Px12^2+Px13^2+Px14^2+Px15^2+Px16^2+Px17^2+Px18^2+Px19^2+Px20^2
>          +Py1^2+Py2^2+Py3^2+Py4^2+Py5^2+Py6^2+Py7^2+Py8^2+Py9^2+Py10^2
>          +Py11^2+Py12^2+Py13^2+Py14^2+Py15^2+Py16^2+Py17^2+Py18^2+Py19^2+Py20^2):
> H := T + V1 + V2:
>  -diff(H, x1);
>  -diff(H, x2);
>  -diff(H, x3);
>  -diff(H, x4);
>  -diff(H, x5);
>  -diff(H, x6);
>  -diff(H, x7);
>  -diff(H, x8);
>  -diff(H, x9);
>  -diff(H, x10);
>  -diff(H, x11);
>  -diff(H, x12);
>  -diff(H, x13);
>  -diff(H, x14);
>  -diff(H, x15);
>  -diff(H, x16);
>  -diff(H, x17);
>  -diff(H, x18);
```

```
> -diff(H, x19);
> -diff(H, x20);
> -diff(H, y1);
> -diff(H, y2);
> -diff(H, y3);
> -diff(H, y4);
> -diff(H, y5);
> -diff(H, y6);
> -diff(H, y7);
> -diff(H, y8);
> -diff(H, y9);
> -diff(H, y10);
> -diff(H, y11);
> -diff(H, y12);
> -diff(H, y13);
> -diff(H, y14);
> -diff(H, y15);
> -diff(H, y16);
> -diff(H, y17);
> -diff(H, y18);
> -diff(H, y19);
> -diff(H, y20);
>

> cannonical :=
> diff(x1(t),t) = Px1(t)/m,
> diff(x2(t),t) = Px2(t)/m,
> diff(x3(t),t) = Px3(t)/m,
> diff(x4(t),t) = Px4(t)/m,
> diff(x5(t),t) = Px5(t)/m,
> diff(x6(t),t) = Px6(t)/m,
> diff(x7(t),t) = Px7(t)/m,
> diff(x8(t),t) = Px8(t)/m,
> diff(x9(t),t) = Px9(t)/m,
> diff(x10(t),t) = Px10(t)/m,
> diff(x11(t),t) = Px11(t)/m,
> diff(x12(t),t) = Px12(t)/m,
> diff(x13(t),t) = Px13(t)/m,
> diff(x14(t),t) = Px14(t)/m,
> diff(x15(t),t) = Px15(t)/m,
> diff(x16(t),t) = Px16(t)/m,
> diff(x17(t),t) = Px17(t)/m,
> diff(x18(t),t) = Px18(t)/m,
> diff(x19(t),t) = Px19(t)/m,
> diff(x20(t),t) = Px20(t)/m,
> diff(y1(t),t) = Py1(t)/m,
```

```
> diff(y2(t),t) = Py2(t)/m,
> diff(y3(t),t) = Py3(t)/m,
> diff(y4(t),t) = Py4(t)/m,
> diff(y5(t),t) = Py5(t)/m,
> diff(y6(t),t) = Py6(t)/m,
> diff(y7(t),t) = Py7(t)/m,
> diff(y8(t),t) = Py8(t)/m,
> diff(y9(t),t) = Py9(t)/m,
> diff(y10(t),t) = Py10(t)/m,
> diff(y11(t),t) = Py11(t)/m,
> diff(y12(t),t) = Py12(t)/m,
> diff(y13(t),t) = Py13(t)/m,
> diff(y14(t),t) = Py14(t)/m,
> diff(y15(t),t) = Py15(t)/m,
> diff(y16(t),t) = Py16(t)/m,
> diff(y17(t),t) = Py17(t)/m,
> diff(y18(t),t) = Py18(t)/m,
> diff(y19(t),t) = Py19(t)/m,
> diff(y20(t),t) = Py20(t)/m,
>
> diff(Px1(t),t) = -2*k1*x1(t)+k1*(x2(t)+x4(t))+k1*L0*(1/2/(x1(t)^2+x2(t)^2+y1(t)^2+y2(t)^2-
2*x1(t)
> *x2(t)-2*y1(t)*y2(t))^(1/2)*(2*x1(t)-2*x2(t))+1/2/(x4(t)^2+x1(t)^2+y4(t)^2+y1(t)
> ^2-2*x4(t)*x1(t)-2*y4(t)*y1(t))^(1/2)*(2*x1(t)-2*x4(t))),
>
> diff(Px2(t),t)=
> -2*k1*x2(t)+k1*(x1(t)+x3(t))+k1*L0*(1/2/(x1(t)^2+x2(t)^2+y1(t)^2+y2(t)^2-
2*x1(t)
> *x2(t)-2*y1(t)*y2(t))^(1/2)*(2*x2(t)-2*x1(t))+1/2/(x2(t)^2+x3(t)^2+y2(t)^2+y3(t)
> ^2-2*x2(t)*x3(t)-2*y2(t)*y3(t))^(1/2)*(2*x2(t)-2*x3(t))),
>
> diff(Px3(t),t)=
> -2*k1*x3(t)+k1*(x2(t)+x4(t))+k1*L0*(1/2/(x2(t)^2+x3(t)^2+y2(t)^2+y3(t)^2-
2*x2(t)
> *x3(t)-2*y2(t)*y3(t))^(1/2)*(2*x3(t)-2*x2(t))+1/2/(x3(t)^2+x4(t)^2+y3(t)^2+y4(t)
> ^2-2*x3(t)*x4(t)-2*y3(t)*y4(t))^(1/2)*(2*x3(t)-2*x4(t))),
>
> diff(Px4(t),t)=
> -2*k1*x4(t)+k1*(x1(t)+x3(t))+k1*L0*(1/2/(x3(t)^2+x4(t)^2+y3(t)^2+y4(t)^2-
2*x3(t)
> *x4(t)-2*y3(t)*y4(t))^(1/2)*(2*x4(t)-2*x3(t))+1/2/(x4(t)^2+x1(t)^2+y4(t)^2+y1(t)
> ^2-2*x4(t)*x1(t)-2*y4(t)*y1(t))^(1/2)*(2*x4(t)-2*x1(t)))-k2*x4(t)+k2*x6(t)+1/2*
> k2*J0/(x4(t)^2+x6(t)^2+y4(t)^2+y6(t)^2-2*x4(t)*x6(t)-2*y4(t)*y6(t))^(1/2)*(2*x4(t
> )-2*x6(t)),
>
> diff(Px5(t),t)=
```

```
> -2*k1*x5(t)+k1*(x6(t)+x8(t))+k1*L0*(1/2/(x5(t)^2+x6(t)^2+y5(t)^2+y6(t)^2-
2*x5(t)
> *x6(t)-2*y5(t)*y6(t))^(1/2)*(2*x5(t)-2*x6(t))+1/2/(x8(t)^2+x5(t)^2+y8(t)^2+y5(t)
> ^2-2*x8(t)*x5(t)-2*y8(t)*y5(t))^(1/2)*(2*x5(t)-2*x8(t)))-k2*x5(t)+k2*x19(t)+1/2*
> k2*J0/(x5(t)^2+x19(t)^2+y5(t)^2+y19(t)^2-2*x5(t)*x19(t)-2*y5(t)*y19(t))^(1/2)*(2
> *x5(t)-2*x19(t)),
>
> diff(Px6(t),t)=
> -2*k1*x6(t)+k1*(x5(t)+x7(t))+k1*L0*(1/2/(x5(t)^2+x6(t)^2+y5(t)^2+y6(t)^2-
2*x5(t)
> *x6(t)-2*y5(t)*y6(t))^(1/2)*(2*x6(t)-2*x5(t))+1/2/(x6(t)^2+x7(t)^2+y6(t)^2+y7(t)
> ^2-2*x6(t)*x7(t)-2*y6(t)*y7(t))^(1/2)*(2*x6(t)-2*x7(t)))-k2*x6(t)+k2*x4(t)+1/2*
> k2*J0/(x4(t)^2+x6(t)^2+y4(t)^2+y6(t)^2-2*x4(t)*x6(t)-2*y4(t)*y6(t))^(1/2)*(2*x6(t
> )-2*x4(t)),
>
> diff(Px7(t),t)=
> -2*k1*x7(t)+k1*(x6(t)+x8(t))+k1*L0*(1/2/(x6(t)^2+x7(t)^2+y6(t)^2+y7(t)^2-
2*x6(t)
> *x7(t)-2*y6(t)*y7(t))^(1/2)*(2*x7(t)-2*x6(t))+1/2/(x7(t)^2+x8(t)^2+y7(t)^2+y8(t)
> ^2-2*x7(t)*x8(t)-2*y7(t)*y8(t))^(1/2)*(2*x7(t)-2*x8(t)))-k2*x7(t)+k2*x9(t)+1/2*
> k2*J0/(x7(t)^2+x9(t)^2+y7(t)^2+y9(t)^2-2*x7(t)*x9(t)-2*y7(t)*y9(t))^(1/2)*(2*x7(t
> )-2*x9(t)),
>
> diff(Px8(t),t)=
> -2*k1*x8(t)+k1*(x5(t)+x7(t))+k1*L0*(1/2/(x7(t)^2+x8(t)^2+y7(t)^2+y8(t)^2-
2*x7(t)
> *x8(t)-2*y7(t)*y8(t))^(1/2)*(2*x8(t)-2*x7(t))+1/2/(x8(t)^2+x5(t)^2+y8(t)^2+y5(t)
> ^2-2*x8(t)*x5(t)-2*y8(t)*y5(t))^(1/2)*(2*x8(t)-2*x5(t)))-k2*x8(t)+k2*x14(t)+1/2*
> k2*J0/(x8(t)^2+x14(t)^2+y8(t)^2+y14(t)^2-2*x8(t)*x14(t)-2*y8(t)*y14(t))^(1/2)*(2
> *x8(t)-2*x14(t)),
>
> diff(Px9(t),t)=
> -2*k1*x9(t)+k1*(x10(t)+x12(t))+k1*L0*(1/2/(x9(t)^2+x10(t)^2+y9(t)^2+y10(t)^2-
2*
> x9(t)*x10(t)-2*y9(t)*y10(t))^(1/2)*(2*x9(t)-2*x10(t))+1/2/(x12(t)^2+x9(t)^2+y12(t
> )^2+y9(t)^2-2*x12(t)*x9(t)-2*y12(t)*y9(t))^(1/2)*(2*x9(t)-2*x12(t)))-k2*x9(t)+k2
> *x7(t)+1/2*k2*J0/(x7(t)^2+x9(t)^2+y7(t)^2+y9(t)^2-2*x7(t)*x9(t)-2*y7(t)*y9(t))^(
> 1/2)*(2*x9(t)-2*x7(t)),
>
> diff(Px10(t),t)=
> -2*k1*x10(t)+k1*(x9(t)+x11(t))+k1*L0*(1/2/(x9(t)^2+x10(t)^2+y9(t)^2+y10(t)^2-
2*
> x9(t)*x10(t)-2*y9(t)*y10(t))^(1/2)*(2*x10(t)-2*x9(t))+1/2/(x10(t)^2+x11(t)^2+y10(
> t)^2+y11(t)^2-2*x10(t)*x11(t)-2*y10(t)*y11(t))^(1/2)*(2*x10(t)-2*x11(t))),
>
> diff(Px11(t),t)=
```

```
> -2*k1*x11(t)+k1*(x10(t)+x12(t))+k1*L0*(1/2/(x10(t)^2+x11(t)^2+y10(t)^2+y11(t)^2-

> 2*x10(t)*x11(t)-2*y10(t)*y11(t))^(1/2)*(2*x11(t)-2*x10(t))+1/2/(x11(t)^2+x12(t)^
> 2+y11(t)^2+y12(t)^2-2*x11(t)*x12(t)-2*y11(t)*y12(t))^(1/2)*(2*x11(t)-
2*x12(t))),
>
> diff(Px12(t),t)=
> -2*k1*x12(t)+k1*(x9(t)+x11(t))+k1*L0*(1/2/(x11(t)^2+x12(t)^2+y11(t)^2+y12(t)^2-
2
> *x11(t)*x12(t)-2*y11(t)*y12(t))^(1/2)*(2*x12(t)-2*x11(t))+1/2/(x12(t)^2+x9(t)^2+
> y12(t)^2+y9(t)^2-2*x12(t)*x9(t)-2*y12(t)*y9(t))^(1/2)*(2*x12(t)-2*x9(t))),
>
> diff(Px13(t),t)=
> -2*k1*x13(t)+k1*(x14(t)+x16(t))+k1*L0*(1/2/(x13(t)^2+x14(t)^2+y13(t)^2+y14(t)^2-

> 2*x13(t)*x14(t)-2*y13(t)*y14(t))^(1/2)*(2*x13(t)-2*x14(t))+1/2/(x16(t)^2+x13(t)^
> 2+y16(t)^2+y13(t)^2-2*x16(t)*x13(t)-2*y16(t)*y13(t))^(1/2)*(2*x13(t)-
2*x16(t))),
>
> diff(Px14(t),t)=
> -2*k1*x14(t)+k1*(x13(t)+x15(t))+k1*L0*(1/2/(x13(t)^2+x14(t)^2+y13(t)^2+y14(t)^2-

> 2*x13(t)*x14(t)-2*y13(t)*y14(t))^(1/2)*(2*x14(t)-2*x13(t))+1/2/(x14(t)^2+x15(t)^
> 2+y14(t)^2+y15(t)^2-2*x14(t)*x15(t)-2*y14(t)*y15(t))^(1/2)*(2*x14(t)-
2*x15(t)))-
> k2*x14(t)+k2*x8(t)+1/2*k2*J0/(x8(t)^2+x14(t)^2+y8(t)^2+y14(t)^2-
2*x8(t)*x14(t)-2
> *y8(t)*y14(t))^(1/2)*(2*x14(t)-2*x8(t)),
>
> diff(Px15(t),t)=
> -2*k1*x15(t)+k1*(x14(t)+x16(t))+k1*L0*(1/2/(x14(t)^2+x15(t)^2+y14(t)^2+y15(t)^2-

> 2*x14(t)*x15(t)-2*y14(t)*y15(t))^(1/2)*(2*x15(t)-2*x14(t))+1/2/(x15(t)^2+x16(t)^
> 2+y15(t)^2+y16(t)^2-2*x15(t)*x16(t)-2*y15(t)*y16(t))^(1/2)*(2*x15(t)-
2*x16(t))),
>
> diff(Px16(t),t)=
> -2*k1*x16(t)+k1*(x13(t)+x15(t))+k1*L0*(1/2/(x15(t)^2+x16(t)^2+y15(t)^2+y16(t)^2-

> 2*x15(t)*x16(t)-2*y15(t)*y16(t))^(1/2)*(2*x16(t)-2*x15(t))+1/2/(x16(t)^2+x13(t)^
> 2+y16(t)^2+y13(t)^2-2*x16(t)*x13(t)-2*y16(t)*y13(t))^(1/2)*(2*x16(t)-
2*x13(t))),
>
> diff(Px17(t),t)=
> -2*k1*x17(t)+k1*(x18(t)+x20(t))+k1*L0*(1/2/(x17(t)^2+x18(t)^2+y17(t)^2+y18(t)^2-
```

```
    > 2*x17(t)*x18(t)-2*y17(t)*y18(t))^(1/2)*(2*x17(t)-2*x18(t))+1/2/(x20(t)^2+x17(t)^
    > 2+y20(t)^2+y17(t)^2-2*x20(t)*x17(t)-2*y20(t)*y17(t))^(1/2)*(2*x17(t)-
2*x20(t))),
    >
    > diff(Px18(t),t)=
    > -2*k1*x18(t)+k1*(x17(t)+x19(t))+k1*L0*(1/2/(x17(t)^2+x18(t)^2+y17(t)^2+y18(t)^2-

    > 2*x17(t)*x18(t)-2*y17(t)*y18(t))^(1/2)*(2*x18(t)-2*x17(t))+1/2/(x18(t)^2+x19(t)^
    > 2+y18(t)^2+y19(t)^2-2*x18(t)*x19(t)-2*y18(t)*y19(t))^(1/2)*(2*x18(t)-
2*x19(t))),
    >
    > diff(Px19(t),t)=
    > -2*k1*x19(t)+k1*(x18(t)+x20(t))+k1*L0*(1/2/(x18(t)^2+x19(t)^2+y18(t)^2+y19(t)^2-

    > 2*x18(t)*x19(t)-2*y18(t)*y19(t))^(1/2)*(2*x19(t)-2*x18(t))+1/2/(x19(t)^2+x20(t)^
    > 2+y19(t)^2+y20(t)^2-2*x19(t)*x20(t)-2*y19(t)*y20(t))^(1/2)*(2*x19(t)-
2*x20(t)))-
    > k2*x19(t)+k2*x5(t)+1/2*k2*J0/(x5(t)^2+x19(t)^2+y5(t)^2+y19(t)^2-
2*x5(t)*x19(t)-2
    > *y5(t)*y19(t))^(1/2)*(2*x19(t)-2*x5(t)),
    >
    > diff(Px20(t),t)=
    > -2*k1*x20(t)+k1*(x17(t)+x19(t))+k1*L0*(1/2/(x19(t)^2+x20(t)^2+y19(t)^2+y20(t)^2-

    > 2*x19(t)*x20(t)-2*y19(t)*y20(t))^(1/2)*(2*x20(t)-2*x19(t))+1/2/(x20(t)^2+x17(t)^
    > 2+y20(t)^2+y17(t)^2-2*x20(t)*x17(t)-2*y20(t)*y17(t))^(1/2)*(2*x20(t)-
2*x17(t))),
    >
    > diff(Py1(t),t)=
    > -2*k1*y1(t)+k1*(y2(t)+y4(t))+k1*L0*(1/2/(x1(t)^2+x2(t)^2+y1(t)^2+y2(t)^2-
2*x1(t)
    > *x2(t)-2*y1(t)*y2(t))^(1/2)*(2*y1(t)-2*y2(t))+1/2/(x4(t)^2+x1(t)^2+y4(t)^2+y1(t)
    > ^2-2*x4(t)*x1(t)-2*y4(t)*y1(t))^(1/2)*(2*y1(t)-2*y4(t))),
    >
    > diff(Py2(t),t)=
    > -2*k1*y2(t)+k1*(y1(t)+y3(t))+k1*L0*(1/2/(x1(t)^2+x2(t)^2+y1(t)^2+y2(t)^2-
2*x1(t)
    > *x2(t)-2*y1(t)*y2(t))^(1/2)*(2*y2(t)-2*y1(t))+1/2/(x2(t)^2+x3(t)^2+y2(t)^2+y3(t)
    > ^2-2*x2(t)*x3(t)-2*y2(t)*y3(t))^(1/2)*(2*y2(t)-2*y3(t))),
    >
    > diff(Py3(t),t)=
    > -2*k1*y3(t)+k1*(y2(t)+y4(t))+k1*L0*(1/2/(x2(t)^2+x3(t)^2+y2(t)^2+y3(t)^2-
2*x2(t)
    > *x3(t)-2*y2(t)*y3(t))^(1/2)*(2*y3(t)-2*y2(t))+1/2/(x3(t)^2+x4(t)^2+y3(t)^2+y4(t)
    > ^2-2*x3(t)*x4(t)-2*y3(t)*y4(t))^(1/2)*(2*y3(t)-2*y4(t))),
    >
```

```
> diff(Py4(t),t)=
> -2*k1*y4(t)+k1*(y1(t)+y3(t))+k1*L0*(1/2/(x3(t)^2+x4(t)^2+y3(t)^2+y4(t)^2-
2*x3(t)
> *x4(t)-2*y3(t)*y4(t))^(1/2)*(2*y4(t)-2*y3(t))+1/2/(x4(t)^2+x1(t)^2+y4(t)^2+y1(t)
> ^2-2*x4(t)*x1(t)-2*y4(t)*y1(t))^(1/2)*(2*y4(t)-2*y1(t)))-k2*y4(t)+k2*y6(t)+1/2*
> k2*J0/(x4(t)^2+x6(t)^2+y4(t)^2+y6(t)^2-2*x4(t)*x6(t)-2*y4(t)*y6(t))^(1/2)*(2*y4(t
> )-2*y6(t)),
>
> diff(Py5(t),t)=
> -2*k1*y5(t)+k1*(y6(t)+y8(t))+k1*L0*(1/2/(x5(t)^2+x6(t)^2+y5(t)^2+y6(t)^2-
2*x5(t)
> *x6(t)-2*y5(t)*y6(t))^(1/2)*(2*y5(t)-2*y6(t))+1/2/(x8(t)^2+x5(t)^2+y8(t)^2+y5(t)
> ^2-2*x8(t)*x5(t)-2*y8(t)*y5(t))^(1/2)*(2*y5(t)-2*y8(t)))-k2*y5(t)+k2*y19(t)+1/2*
> k2*J0/(x5(t)^2+x19(t)^2+y5(t)^2+y19(t)^2-2*x5(t)*x19(t)-2*y5(t)*y19(t))^(1/2)*(2
> *y5(t)-2*y19(t)),
>
> diff(Py6(t),t)=
> -2*k1*y6(t)+k1*(y5(t)+y7(t))+k1*L0*(1/2/(x5(t)^2+x6(t)^2+y5(t)^2+y6(t)^2-
2*x5(t)
> *x6(t)-2*y5(t)*y6(t))^(1/2)*(2*y6(t)-2*y5(t))+1/2/(x6(t)^2+x7(t)^2+y6(t)^2+y7(t)
> ^2-2*x6(t)*x7(t)-2*y6(t)*y7(t))^(1/2)*(2*y6(t)-2*y7(t)))-k2*y6(t)+k2*y4(t)+1/2*
> k2*J0/(x4(t)^2+x6(t)^2+y4(t)^2+y6(t)^2-2*x4(t)*x6(t)-2*y4(t)*y6(t))^(1/2)*(2*y6(t
> )-2*y4(t)),
>
> diff(Py7(t),t)=
> -2*k1*y7(t)+k1*(y6(t)+y8(t))+k1*L0*(1/2/(x6(t)^2+x7(t)^2+y6(t)^2+y7(t)^2-
2*x6(t)
> *x7(t)-2*y6(t)*y7(t))^(1/2)*(2*y7(t)-2*y6(t))+1/2/(x7(t)^2+x8(t)^2+y7(t)^2+y8(t)
> ^2-2*x7(t)*x8(t)-2*y7(t)*y8(t))^(1/2)*(2*y7(t)-2*y8(t)))-k2*y7(t)+k2*y9(t)+1/2*
> k2*J0/(x7(t)^2+x9(t)^2+y7(t)^2+y9(t)^2-2*x7(t)*x9(t)-2*y7(t)*y9(t))^(1/2)*(2*y7(t
> )-2*y9(t)),
>
> diff(Py8(t),t)=
> -2*k1*y8(t)+k1*(y5(t)+y7(t))+k1*L0*(1/2/(x7(t)^2+x8(t)^2+y7(t)^2+y8(t)^2-
2*x7(t)
> *x8(t)-2*y7(t)*y8(t))^(1/2)*(2*y8(t)-2*y7(t))+1/2/(x8(t)^2+x5(t)^2+y8(t)^2+y5(t)
> ^2-2*x8(t)*x5(t)-2*y8(t)*y5(t))^(1/2)*(2*y8(t)-2*y5(t)))-k2*y8(t)+k2*y14(t)+1/2*
> k2*J0/(x8(t)^2+x14(t)^2+y8(t)^2+y14(t)^2-2*x8(t)*x14(t)-2*y8(t)*y14(t))^(1/2)*(2
> *y8(t)-2*y14(t)),
>
> diff(Py9(t),t)=
> -2*k1*y9(t)+k1*(y10(t)+y12(t))+k1*L0*(1/2/(x9(t)^2+x10(t)^2+y9(t)^2+y10(t)^2-
2*
> x9(t)*x10(t)-2*y9(t)*y10(t))^(1/2)*(2*y9(t)-2*y10(t))+1/2/(x12(t)^2+x9(t)^2+y12(t
> )^2+y9(t)^2-2*x12(t)*x9(t)-2*y12(t)*y9(t))^(1/2)*(2*y9(t)-2*y12(t)))-k2*y9(t)+k2
> *y7(t)+1/2*k2*J0/(x7(t)^2+x9(t)^2+y7(t)^2+y9(t)^2-2*x7(t)*x9(t)-2*y7(t)*y9(t))^(
```

```
> 1/2)*(2*y9(t)-2*y7(t)),
>
> diff(Py10(t),t)=
> -2*k1*y10(t)+k1*(y9(t)+y11(t))+k1*L0*(1/2/(x9(t)^2+x10(t)^2+y9(t)^2+y10(t)^2-
2*
> x9(t)*x10(t)-2*y9(t)*y10(t))^(1/2)*(2*y10(t)-2*y9(t))+1/2/(x10(t)^2+x11(t)^2+y10(
> t)^2+y11(t)^2-2*x10(t)*x11(t)-2*y10(t)*y11(t))^(1/2)*(2*y10(t)-2*y11(t))),
>
> diff(Py11(t),t)=
> -2*k1*y11(t)+k1*(y10(t)+y12(t))+k1*L0*(1/2/(x10(t)^2+x11(t)^2+y10(t)^2+y11(t)^2-

> 2*x10(t)*x11(t)-2*y10(t)*y11(t))^(1/2)*(2*y11(t)-2*y10(t))+1/2/(x11(t)^2+x12(t)^
> 2+y11(t)^2+y12(t)^2-2*x11(t)*x12(t)-2*y11(t)*y12(t))^(1/2)*(2*y11(t)-
2*y12(t))),
>
> diff(Py12(t),t)=
> -2*k1*y12(t)+k1*(y9(t)+y11(t))+k1*L0*(1/2/(x11(t)^2+x12(t)^2+y11(t)^2+y12(t)^2-
2
> *x11(t)*x12(t)-2*y11(t)*y12(t))^(1/2)*(2*y12(t)-2*y11(t))+1/2/(x12(t)^2+x9(t)^2+
> y12(t)^2+y9(t)^2-2*x12(t)*x9(t)-2*y12(t)*y9(t))^(1/2)*(2*y12(t)-2*y9(t))),
>
> diff(Py13(t),t)=
> -2*k1*y13(t)+k1*(y14(t)+y16(t))+k1*L0*(1/2/(x13(t)^2+x14(t)^2+y13(t)^2+y14(t)^2-

> 2*x13(t)*x14(t)-2*y13(t)*y14(t))^(1/2)*(2*y13(t)-2*y14(t))+1/2/(x16(t)^2+x13(t)^
> 2+y16(t)^2+y13(t)^2-2*x16(t)*x13(t)-2*y16(t)*y13(t))^(1/2)*(2*y13(t)-
2*y16(t))),
>
> diff(Py14(t),t)=
> -2*k1*y14(t)+k1*(y13(t)+y15(t))+k1*L0*(1/2/(x13(t)^2+x14(t)^2+y13(t)^2+y14(t)^2-

> 2*x13(t)*x14(t)-2*y13(t)*y14(t))^(1/2)*(2*y14(t)-2*y13(t))+1/2/(x14(t)^2+x15(t)^
> 2+y14(t)^2+y15(t)^2-2*x14(t)*x15(t)-2*y14(t)*y15(t))^(1/2)*(2*y14(t)-
2*y15(t)))-
> k2*y14(t)+k2*y8(t)+1/2*k2*J0/(x8(t)^2+x14(t)^2+y8(t)^2+y14(t)^2-
2*x8(t)*x14(t)-2
> *y8(t)*y14(t))^(1/2)*(2*y14(t)-2*y8(t)),
>
> diff(Py15(t),t)=
> -2*k1*y15(t)+k1*(y14(t)+y16(t))+k1*L0*(1/2/(x14(t)^2+x15(t)^2+y14(t)^2+y15(t)^2-

> 2*x14(t)*x15(t)-2*y14(t)*y15(t))^(1/2)*(2*y15(t)-2*y14(t))+1/2/(x15(t)^2+x16(t)^
> 2+y15(t)^2+y16(t)^2-2*x15(t)*x16(t)-2*y15(t)*y16(t))^(1/2)*(2*y15(t)-
2*y16(t))),
>
> diff(Py16(t),t)=
```

```
> -2*k1*y16(t)+k1*(y13(t)+y15(t))+k1*L0*(1/2/(x15(t)^2+x16(t)^2+y15(t)^2+y16(t)^2-

> 2*x15(t)*x16(t)-2*y15(t)*y16(t))^(1/2)*(2*y16(t)-2*y15(t))+1/2/(x16(t)^2+x13(t)^
> 2+y16(t)^2+y13(t)^2-2*x16(t)*x13(t)-2*y16(t)*y13(t))^(1/2)*(2*y16(t)-
2*y13(t))),
>
> diff(Py17(t),t)=
> -2*k1*y17(t)+k1*(y18(t)+y20(t))+k1*L0*(1/2/(x17(t)^2+x18(t)^2+y17(t)^2+y18(t)^2-

> 2*x17(t)*x18(t)-2*y17(t)*y18(t))^(1/2)*(2*y17(t)-2*y18(t))+1/2/(x20(t)^2+x17(t)^
> 2+y20(t)^2+y17(t)^2-2*x20(t)*x17(t)-2*y20(t)*y17(t))^(1/2)*(2*y17(t)-
2*y20(t))),
>
> diff(Py18(t),t)=
> -2*k1*y18(t)+k1*(y17(t)+y19(t))+k1*L0*(1/2/(x17(t)^2+x18(t)^2+y17(t)^2+y18(t)^2-

> 2*x17(t)*x18(t)-2*y17(t)*y18(t))^(1/2)*(2*y18(t)-2*y17(t))+1/2/(x18(t)^2+x19(t)^
> 2+y18(t)^2+y19(t)^2-2*x18(t)*x19(t)-2*y18(t)*y19(t))^(1/2)*(2*y18(t)-
2*y19(t))),
>
> diff(Py19(t),t)=
> -2*k1*y19(t)+k1*(y18(t)+y20(t))+k1*L0*(1/2/(x18(t)^2+x19(t)^2+y18(t)^2+y19(t)^2-

> 2*x18(t)*x19(t)-2*y18(t)*y19(t))^(1/2)*(2*y19(t)-2*y18(t))+1/2/(x19(t)^2+x20(t)^
> 2+y19(t)^2+y20(t)^2-2*x19(t)*x20(t)-2*y19(t)*y20(t))^(1/2)*(2*y19(t)-
2*y20(t)))-
> k2*y19(t)+k2*y5(t)+1/2*k2*J0/(x5(t)^2+x19(t)^2+y5(t)^2+y19(t)^2-
2*x5(t)*x19(t)-2
> *y5(t)*y19(t))^(1/2)*(2*y19(t)-2*y5(t)),
>
> diff(Py20(t),t)=
> -2*k1*y20(t)+k1*(y17(t)+y19(t))+k1*L0*(1/2/(x19(t)^2+x20(t)^2+y19(t)^2+y20(t)^2-

> 2*x19(t)*x20(t)-2*y19(t)*y20(t))^(1/2)*(2*y20(t)-2*y19(t))+1/2/(x20(t)^2+x17(t)^
> 2+y20(t)^2+y17(t)^2-2*x20(t)*x17(t)-2*y20(t)*y17(t))^(1/2)*(2*y20(t)-
2*y17(t)));


> initial_condition :=
> x1(0)=1.3, y1(0)=2.5,
> x2(0)=2.7, y2(0)=2.5,
> x3(0)=2.3, y3(0)=1.5,
> x4(0)=1.7, y4(0)=1.5,
> x5(0)=-0.3,y5(0)=0.5,
> x6(0)=0.3, y6(0)=0.5,
> x7(0)=0.7, y7(0)=-0.5,
```

```
> x8(0)=-0.7,y8(0)=-0.5,
> x9(0)=1.3,  y9(0)=-1.5,
> x10(0)=2.7, y10(0)=-1.5,
> x11(0)=2.3, y11(0)=-2.5,
> x12(0)=1.7, y12(0)=-2.5,
> x13(0)=-2.7, y13(0)=-1.5,
> x14(0)=-1.3, y14(0)=-1.5,
> x15(0)=-1.7, y15(0)=-2.5,
> x16(0)=-2.3, y16(0)=-2.5,
> x17(0)=-2.7, y17(0)=2.5,
> x18(0)=-1.3, y18(0)=2.5,
> x19(0)=-1.7, y19(0)=1.5,
> x20(0)=-2.3, y20(0)=1.5,
> Px1(0)=0,Px2(0)=0,Px3(0)=0,Px4(0)=0,
> Py1(0)=0,Py2(0)=0,Py3(0)=0,Py4(0)=0,
> Px5(0)=0,Px6(0)=0,Px7(0)=0,Px8(0)=0,
> Py5(0)=0,Py6(0)=0,Py7(0)=0,Py8(0)=0,
> Px9(0)=0,Px10(0)=0,Px11(0)=0,Px12(0)=0,
> Py9(0)=0,Py10(0)=0,Py11(0)=0,Py12(0)=0,
> Px13(0)=0,Px14(0)=0,Px15(0)=0,Px16(0)=0,
> Py13(0)=0,Py14(0)=0,Py15(0)=0,Py16(0)=0,
> Px17(0)=0,Px18(0)=0,Px19(0)=0,Px20(0)=0,
> Py17(0)=0,Py18(0)=0,Py19(0)=0,Py20(0)=0;


> # defining constant
> k1 := 1;
> k2 := 0.5;
> L0 := 1;
> J0 := evalf(sqrt(2));
> m  := 1;


> dsol := dsolve({cannonical,initial_condition}, numeric, method=classical[rk4],
output=listprocedure);

> fx1 := eval(x1(t),dsol);
> fx2 := eval(x2(t),dsol);
> fx3 := eval(x3(t),dsol);
> fx4 := eval(x4(t),dsol);
> fx5 := eval(x5(t),dsol);
> fx6 := eval(x6(t),dsol);
> fx7 := eval(x7(t),dsol);
> fx8 := eval(x8(t),dsol);
> fx9 := eval(x9(t),dsol);
> fx10 := eval(x10(t),dsol);
```

```
> fx11 := eval(x11(t),dsol);
> fx12 := eval(x12(t),dsol);
> fx13 := eval(x13(t),dsol);
> fx14 := eval(x14(t),dsol);
> fx15 := eval(x15(t),dsol);
> fx16 := eval(x16(t),dsol);
> fx17 := eval(x17(t),dsol);
> fx18 := eval(x18(t),dsol);
> fx19 := eval(x19(t),dsol);
> fx20 := eval(x20(t),dsol);
>
> fy1 := eval(y1(t),dsol);
> fy2 := eval(y2(t),dsol);
> fy3 := eval(y3(t),dsol);
> fy4 := eval(y4(t),dsol);
> fy5 := eval(y5(t),dsol);
> fy6 := eval(y6(t),dsol);
> fy7 := eval(y7(t),dsol);
> fy8 := eval(y8(t),dsol);
> fy9 := eval(y9(t),dsol);
> fy10 := eval(y10(t),dsol);
> fy11 := eval(y11(t),dsol);
> fy12 := eval(y12(t),dsol);
> fy13 := eval(y13(t),dsol);
> fy14 := eval(y14(t),dsol);
> fy15 := eval(y15(t),dsol);
> fy16 := eval(y16(t),dsol);
> fy17 := eval(y17(t),dsol);
> fy18 := eval(y18(t),dsol);
> fy19 := eval(y19(t),dsol);
> fy20 := eval(y20(t),dsol);
>
>
```

A.2 C++ Source Code for Visualization Program

```
//
    //
    //
    #ifdef WIN32
    #define WIN32_LEAN_AND_MEAN
    #include <windows.h>
    #endif
    #if defined(__APPLE__) && defined(__MACH__)
    #include <OpenGL/gl.h>  // Header File For The OpenGL32 Library
```

```
#include <OpenGL/glu.h> // Header File For The GLu32 Library
#else
#include <GL/gl.h>      // Header File For The OpenGL32 Library
#include <GL/glu.h>     // Header File For The GLu32 Library

#endif
#include "SDL.h"

#include <math.h>
#include <GL/glut.h>

#include <iostream.h>
#include <fstream.h>

/* rotation angle for the triangle. */
float rtri = 0.0f;

/* rotation angle for the quadrilateral. */
float rquad = 0.0f;

/* variables added by reynardmh */
int t = 0;
float xpos = 0.0f;
float ypos = 0.0f;
float zpos = 0.0f;

int const t1 = 5;

float x1dat[50];
float y1dat[50];
float x2dat[50];
float y2dat[50];
float x3dat[50];
float y3dat[50];
float x4dat[50];
float y4dat[50];
float x5dat[50];
float y5dat[50];
float x6dat[50];
float y6dat[50];
float x7dat[50];
float y7dat[50];
float x8dat[50];
float y8dat[50];
float x9dat[50];
float y9dat[50];
```

```
float x10dat[50];
float y10dat[50];
float x11dat[50];
float y11dat[50];
float x12dat[50];
float y12dat[50];
float x13dat[50];
float y13dat[50];
float x14dat[50];
float y14dat[50];
float x15dat[50];
float y15dat[50];
float x16dat[50];
float y16dat[50];
float x17dat[50];
float y17dat[50];
float x18dat[50];
float y18dat[50];
float x19dat[50];
float y19dat[50];
float x20dat[50];
float y20dat[50];


float tx = 0.0f;
float ty = 0.0f;
float tz = 0.0f;
int light_on = 0;

float cam_y = 0;
/* end additional variables by reynardmh */

/* global function definition */

void readData (void);
float getx1data(int);
float gety1data(int);
float getx2data(int);
float gety2data(int);
float getx3data(int);
float gety3data(int);
float getx4data(int);
float gety4data(int);

/* function to read data and put it in array xdat and ydat */
```

```
void readData (void)
  {

  float num;
  int i=0;

  ifstream infile_x_1("x1Data.lst");
  while(infile_x_1 >> num)
    {
    x1dat[i] = num;
    i++;
    }
  i=0;

  ifstream infile_y_1("y1Data.lst");
  while(infile_y_1 >> num)
    {
    y1dat[i] = num;
    i++;
    }
  i=0;

  ifstream infile_x_2("x2Data.lst");
  while(infile_x_2 >> num)
    {
    x2dat[i] = num;
    i++;
    }
  i=0;

  ifstream infile_y_2("y2Data.lst");
  while(infile_y_2 >> num)
    {
    y2dat[i] = num;
    i++;
    }
  i=0;

  ifstream infile_x_3("x3Data.lst");
  while(infile_x_3 >> num)
    {
    x3dat[i] = num;
    i++;
    }
  i=0;
```

```
ifstream infile_y_3("y3Data.lst");
while(infile_y_3 >> num)
  {
  y3dat[i] = num;
  i++;
  }
i=0;

ifstream infile_x_4("x4Data.lst");
while(infile_x_4 >> num)
  {
  x4dat[i] = num;
  i++;
  }
i=0;

ifstream infile_y_4("y4Data.lst");
while(infile_y_4 >> num)
  {
  y4dat[i] = num;
  i++;
  }
i=0;

ifstream infile_x_5("x5Data.lst");
while(infile_x_5 >> num)
  {
  x5dat[i] = num;
  i++;
  }
i=0;

ifstream infile_y_5("y5Data.lst");
while(infile_y_5 >> num)
  {
  y5dat[i] = num;
  i++;
  }
i=0;

ifstream infile_x_6("x6Data.lst");
while(infile_x_6 >> num)
  {
  x6dat[i] = num;
  i++;
  }
```

```
i=0;

ifstream infile_y_6("y6Data.lst");
while(infile_y_6 >> num)
  {
  y6dat[i] = num;
  i++;
  }
i=0;

ifstream infile_x_7("x7Data.lst");
while(infile_x_7 >> num)
  {
  x7dat[i] = num;
  i++;
  }
i=0;

ifstream infile_y_7("y7Data.lst");
while(infile_y_7 >> num)
  {
  y7dat[i] = num;
  i++;
  }
i=0;

ifstream infile_x_8("x8Data.lst");
while(infile_x_8 >> num)
  {
  x8dat[i] = num;
  i++;
  }
i=0;

ifstream infile_y_8("y8Data.lst");
while(infile_y_8 >> num)
  {
  y8dat[i] = num;
  i++;
  }
i=0;

ifstream infile_x_9("x9Data.lst");
while(infile_x_9 >> num)
  {
  x9dat[i] = num;
```

```
  i++;
  }
i=0;

ifstream infile_y_9("y9Data.lst");
while(infile_y_9 >> num)
  {
  y9dat[i] = num;
  i++;
  }
i=0;

ifstream infile_x_10("x10Data.lst");
while(infile_x_10 >> num)
  {
  x10dat[i] = num;
  i++;
  }
i=0;

ifstream infile_y_10("y10Data.lst");
while(infile_y_10 >> num)
  {
  y10dat[i] = num;
  i++;
  }
i=0;

ifstream infile_x_11("x11Data.lst");
while(infile_x_11 >> num)
  {
  x11dat[i] = num;
  i++;
  }
i=0;

ifstream infile_y_11("y11Data.lst");
while(infile_y_11 >> num)
  {
  y11dat[i] = num;
  i++;
  }
i=0;

ifstream infile_x_12("x12Data.lst");
while(infile_x_12 >> num)
```

```
  {
  x12dat[i] = num;
  i++;
  }
i=0;

ifstream infile_y_12("y12Data.lst");
while(infile_y_12 >> num)
  {
  y12dat[i] = num;
  i++;
  }
i=0;

ifstream infile_x_13("x13Data.lst");
while(infile_x_13 >> num)
  {
  x13dat[i] = num;
  i++;
  }
i=0;

ifstream infile_y_13("y13Data.lst");
while(infile_y_13 >> num)
  {
  y13dat[i] = num;
  i++;
  }
i=0;

ifstream infile_x_14("x14Data.lst");
while(infile_x_14 >> num)
  {
  x14dat[i] = num;
  i++;
  }
i=0;

ifstream infile_y_14("y14Data.lst");
while(infile_y_14 >> num)
  {
  y14dat[i] = num;
  i++;
  }
i=0;
```

```
ifstream infile_x_15("x15Data.lst");
while(infile_x_15 >> num)
  {
  x15dat[i] = num;
  i++;
  }
i=0;

ifstream infile_y_15("y15Data.lst");
while(infile_y_15 >> num)
  {
  y15dat[i] = num;
  i++;
  }
i=0;

ifstream infile_x_16("x16Data.lst");
while(infile_x_16 >> num)
  {
  x16dat[i] = num;
  i++;
  }
i=0;

ifstream infile_y_16("y16Data.lst");
while(infile_y_16 >> num)
  {
  y16dat[i] = num;
  i++;
  }
i=0;

ifstream infile_x_17("x17Data.lst");
while(infile_x_17 >> num)
  {
  x17dat[i] = num;
  i++;
  }
i=0;

ifstream infile_y_17("y17Data.lst");
while(infile_y_17 >> num)
  {
  y17dat[i] = num;
  i++;
  }
```

```
i=0;

ifstream infile_x_18("x18Data.lst");
while(infile_x_18 >> num)
  {
  x18dat[i] = num;
  i++;
  }
i=0;

ifstream infile_y_18("y18Data.lst");
while(infile_y_18 >> num)
  {
  y18dat[i] = num;
  i++;
  }
i=0;

ifstream infile_x_19("x19Data.lst");
while(infile_x_19 >> num)
  {
  x19dat[i] = num;
  i++;
  }
i=0;

ifstream infile_y_19("y19Data.lst");
while(infile_y_19 >> num)
  {
  y19dat[i] = num;
  i++;
  }
i=0;

ifstream infile_x_20("x20Data.lst");
while(infile_x_20 >> num)
  {
  x20dat[i] = num;
  i++;
  }
i=0;

ifstream infile_y_20("y20Data.lst");
while(infile_y_20 >> num)
  {
  y20dat[i] = num;
```

```
  i++;
  }
 i=0;
 }


 /* A general OpenGL initialization function.  Sets all of the initial parameters. */
 void InitGL(int Width, int Height)              // We call this right after our OpenGL window is created.
 {
  GLfloat mat_specular[] = { 1.0, 1.0, 1.0, 1.0 };
  GLfloat mat_shininess[] = { 100.0 };
  GLfloat light_position[] = { 1.0, 1.0, 1.0, 0.0 };
  GLfloat light_position2[] = { -1.0, -1.0, 1.0, 0.0 };

  //GLfloat white_light[] = {1, 1, 1, 1};
  //GLfloat lmodel_ambient[] = {0.1, 0.1, 0.1, 1.0};

  glViewport(0, 0, Width, Height);
  glClearColor(0.0f, 0.0f, 0.0f, 0.0f);       // This Will Clear The Background Color To Black
  glClearDepth(1.0);                          // Enables Clearing Of The Depth Buffer
  glDepthFunc(GL_LESS);                       // The Type Of Depth Test To Do
  glEnable(GL_DEPTH_TEST);                    // Enables Depth Testing
  glShadeModel(GL_SMOOTH);                    // Enables Smooth Color Shading


  glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
  glMaterialfv(GL_FRONT, GL_SHININESS, mat_shininess);
  glLightfv(GL_LIGHT0, GL_POSITION, light_position);
  // glLightfv(GL_LIGHT1, GL_POSITION, light_position2); // 2 lights does not work ???

  //glLightfv(GL_LIGHT0, GL_DIFFUSE, white_light);
  //glLightfv(GL_LIGHT0, GL_SPECULAR, white_light);
  //glLightModelfv(GL_LIGHT_MODEL_AMBIENT, lmodel_ambient);

  glEnable(GL_LIGHT0);
  // glEnable(GL_LIGHT1);

  glMatrixMode(GL_PROJECTION);
```

```
    //glMatrixMode(GL_ORTHOGONAL);

    glLoadIdentity();                    // Reset The Projection Matrix

     gluPerspective(45.0f,(GLfloat)Width/(GLfloat)Height,0.1f,100.0f);       //
Calculate The Aspect Ratio Of The Window

    glMatrixMode(GL_MODELVIEW);
  }

  /* The main drawing function. */
  void DrawGLScene()
  {
   glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);       //
Clear The Screen And The Depth Buffer

    glLoadIdentity();                   // make sure we're no longer rotated.

    if (light_on)
      glEnable(GL_LIGHTING);
    else
      glDisable(GL_LIGHTING);

    glTranslatef(0,0,0);
    gluLookAt(tx, ty, tz - 15, 0, 0, 0, cam_y, 1, 1);

    //glColor3f(0.5, 0.5, 0.5);
    glBegin(GL_LINES);
      glColor3f (0.5, 0, 0);
      glVertex2f(-50, 0);
      glVertex2f( 50, 0); // x axis

      glColor3f (0, 0.5, 0);
      glVertex2f( 0, -50);
      glVertex2f( 0,  50); // y axis

      glColor3f (0, 0, 0.5);
      glVertex3f( 0, 0, -50);
      glVertex3f( 0, 0,  50); // z axis
    glEnd();

    // Particle 1

    xpos = x1dat[t];
    ypos = y1dat[t];
```

```
glTranslatef(xpos, ypos, 0.0f);
glColor3f(1, 1, 1);
glutWireSphere(0.25f, 100, 100);
glTranslatef(-xpos, -ypos,0);



// Particle 2

xpos = x2dat[t];
ypos = y2dat[t];

glTranslatef(xpos, ypos, 0.0f);
glColor3f(1, 1, 1);
glutWireSphere(0.25f, 100, 100);
glTranslatef(-xpos, -ypos,0);



// Particle 3

xpos = x3dat[t];
ypos = y3dat[t];

glTranslatef(xpos, ypos, 0.0f);
glColor3f(1, 1, 1);
glutWireSphere(0.25f, 100, 100);
glTranslatef(-xpos, -ypos,0);



// Particle 4

xpos = x4dat[t];
ypos = y4dat[t];

glTranslatef(xpos, ypos, 0.0f);
glColor3f(1, 1, 1);
glutWireSphere(0.25f, 100, 100);
glTranslatef(-xpos, -ypos,0);



// Particle 5

xpos = x5dat[t];
```

```
ypos = y5dat[t];

glTranslatef(xpos, ypos, 0.0f);
glColor3f(1, 1, 1);
glutWireSphere(0.25f, 100, 100);
glTranslatef(-xpos, -ypos,0);



// Particle 6

xpos = x6dat[t];
ypos = y6dat[t];

glTranslatef(xpos, ypos, 0.0f);
glColor3f(1, 1, 1);
glutWireSphere(0.25f, 100, 100);
glTranslatef(-xpos, -ypos,0);



// Particle 7

xpos = x7dat[t];
ypos = y7dat[t];

glTranslatef(xpos, ypos, 0.0f);
glColor3f(1, 1, 1);
glutWireSphere(0.25f, 100, 100);
glTranslatef(-xpos, -ypos,0);



// Particle 8

xpos = x8dat[t];
ypos = y8dat[t];

glTranslatef(xpos, ypos, 0.0f);
glColor3f(1, 1, 1);
glutWireSphere(0.25f, 100, 100);
glTranslatef(-xpos, -ypos,0);



// Particle 9
```

```
xpos = x9dat[t];
ypos = y9dat[t];

glTranslatef(xpos, ypos, 0.0f);
glColor3f(1, 1, 1);
glutWireSphere(0.25f, 100, 100);
glTranslatef(-xpos, -ypos,0);



// Particle 10

xpos = x10dat[t];
ypos = y10dat[t];

glTranslatef(xpos, ypos, 0.0f);
glColor3f(1, 1, 1);
glutWireSphere(0.25f, 100, 100);
glTranslatef(-xpos, -ypos,0);



// Particle 11

xpos = x11dat[t];
ypos = y11dat[t];

glTranslatef(xpos, ypos, 0.0f);
glColor3f(1, 1, 1);
glutWireSphere(0.25f, 100, 100);
glTranslatef(-xpos, -ypos,0);



// Particle 12

xpos = x12dat[t];
ypos = y12dat[t];

glTranslatef(xpos, ypos, 0.0f);
glColor3f(1, 1, 1);
glutWireSphere(0.25f, 100, 100);
glTranslatef(-xpos, -ypos,0);
```

```
// Particle 13

xpos = x13dat[t];
ypos = y13dat[t];

glTranslatef(xpos, ypos, 0.0f);
glColor3f(1, 1, 1);
glutWireSphere(0.25f, 100, 100);
glTranslatef(-xpos, -ypos,0);



// Particle 14

xpos = x14dat[t];
ypos = y14dat[t];

glTranslatef(xpos, ypos, 0.0f);
glColor3f(1, 1, 1);
glutWireSphere(0.25f, 100, 100);
glTranslatef(-xpos, -ypos,0);



// Particle 15

xpos = x15dat[t];
ypos = y15dat[t];

glTranslatef(xpos, ypos, 0.0f);
glColor3f(1, 1, 1);
glutWireSphere(0.25f, 100, 100);
glTranslatef(-xpos, -ypos,0);



// Particle 16

xpos = x16dat[t];
ypos = y16dat[t];

glTranslatef(xpos, ypos, 0.0f);
glColor3f(1, 1, 1);
glutWireSphere(0.25f, 100, 100);
glTranslatef(-xpos, -ypos,0);
```

```
// Particle 17

xpos = x17dat[t];
ypos = y17dat[t];

glTranslatef(xpos, ypos, 0.0f);
glColor3f(1, 1, 1);
glutWireSphere(0.25f, 100, 100);
glTranslatef(-xpos, -ypos,0);



// Particle 18

xpos = x18dat[t];
ypos = y18dat[t];

glTranslatef(xpos, ypos, 0.0f);
glColor3f(1, 1, 1);
glutWireSphere(0.25f, 100, 100);
glTranslatef(-xpos, -ypos,0);



// Particle 19

xpos = x19dat[t];
ypos = y19dat[t];

glTranslatef(xpos, ypos, 0.0f);
glColor3f(1, 1, 1);
glutWireSphere(0.25f, 100, 100);
glTranslatef(-xpos, -ypos,0);



// Particle 20

xpos = x20dat[t];
ypos = y20dat[t];

glTranslatef(xpos, ypos, 0.0f);
glColor3f(1, 1, 1);
```

```
    glutWireSphere(0.25f, 100, 100);
    glTranslatef(-xpos, -ypos,0);



    // Updating t
    if(t<t1-1)
      t++;
    else
      t=0;



    //glutSolidSphere(1.0f, 16, 16);
    //glutWireTeapot(2.0f);

    // swap buffers to display, since we're double buffered.
    SDL_GL_SwapBuffers();
  }

  int main(int argc, char **argv)
  {
    int done, i=0;
    SDLMod kmod;


    /* Initialize SDL for video output */
    if ( SDL_Init(SDL_INIT_VIDEO) < 0 ) {
      fprintf(stderr, "Unable to initialize SDL: %s\n", SDL_GetError());
      exit(1);
    }

    /* Create a 640x480 OpenGL screen */
    if ( SDL_SetVideoMode(640, 480, 0, SDL_OPENGL) == NULL ) {
     fprintf(stderr, "Unable to create OpenGL screen: %s\n", SDL_GetError());
      SDL_Quit();
      exit(2);
    }

    /* Set the title bar in environments that support it */
    SDL_WM_SetCaption("Jeff Molofee's GL Code Tutorial ... NeHe '99",
NULL);
    SDL_EnableKeyRepeat(SDL_DEFAULT_REPEAT_DELAY, 10); // re-
peat delay, repeat interval
```

```
// Reading data
readData();


/* Loop, drawing and checking events */
InitGL(640, 480);
done = 0;
while ( ! done ) {
  DrawGLScene();

  /* This could go in a separate function */
  { SDL_Event event;


    while ( SDL_PollEvent(&event) ) {
      if ( event.type == SDL_QUIT ) {
        done = 1;
      }
      kmod = SDL_GetModState();

      //printf("Kmod : %i %i\n", kmod, KMOD_LSHIFT);
      if ( event.type == SDL_KEYDOWN ) {
        if ( kmod - 4096 == KMOD_LSHIFT ) {
          switch ( event.key.keysym.sym ) {
            case SDLK_UP: ty += 0.1f;
                 break;
            case SDLK_DOWN: ty -= 0.1f;
                 break;
            case SDLK_i: cam_y -= 0.1f;
                 break;
          }

        }
        else {
          switch ( event.key.keysym.sym ) {
            case SDLK_ESCAPE: done = 1;
                 break;
            case SDLK_LEFT: tx += 0.2f;
                 break;
            case SDLK_RIGHT: tx -= 0.2f;
                 break;
            case SDLK_UP: tz += 0.2f;
                 break;
            case SDLK_DOWN: tz -= 0.2f;
                 break;
```

```
            case SDLK_i: cam_y += 0.1f;
                break;
            case SDLK_l: light_on = light_on ? 0 : 1;
                break;
          }
        }
      }
    }
  }
  SDL_Quit();
  return 1;
}

float getx1data(int t)
  {
  return x1dat[t];
  }
float gety1data(int t)
  {
  return y1dat[t];
  }
float getx2data(int t)
  {
  return x2dat[t];
  }
float gety2data(int t)
  {
  return y2dat[t];
  }
float getx3data(int t)
  {
  return x3dat[t];
  }
float gety3data(int t)
  {
  return y3dat[t];
  }
float getx4data(int t)
  {
  return x4dat[t];
  }
float gety4data(int t)
  {
  return y4dat[t];
  }
```

References

[1] Breeze, Lara. Mathematical Modeling of Ion Channel Mosaic.

[2] Lamb, Robert A., Leslie J. Holsinger, and Lawrence H. Pinto. (1994). The influenza A virus M_{2} ion channel protein and its role in the influenza virus life cycle. In Cellular Receptors of Animal Virusus, (Cold Spring Harbor Laboratory Press), pp.303-321.

[3] Holsinger, Leslie J. and Robert A. Lamb. (1991). Influenza A virus M_{2} is a homotetramer stabilized by formation of disulfide bonds. Virology 183:32.

[4] Helrich, Carl. Private Communication.