SIMULATION OF SELF ORGANIZING STRUCTURES USING NEURO MECHANICAL NETWORKS

MAGNUS SETHSON
Dept. of Mechanical Engineering
Linköping University
Sweden

PETTER KRUS
Dept. of Mechanical Engineering
Linköping University
Sweden

MATTS KARLSSON
Dept. of Biomedical Engineering
Linköping University
Sweden

Abstract: The neuro mechanical network consists of a large number of one-dimensional elements connected into a topological graph of intelligent actuators in 2 or 3 dimensions. This forms a self actuating mechanical network that can be trained to perform certain tasks. In the analysis and training of such networks the time domain simulation of the network performance becomes important. Even though the basic components hardly exist in hardware at present, the study of such networks gives us interesting models to design and analysis the mechanisms of the near future using current technologies and engineering tools. The neuro mechanical network has a meaning also at a micro or even macro level in order to realize highly robust flexible actuator systems. Another potential use is for design of more conventional system, requiring a minimum of components. Furthermore it can be used as an explanatory model for some of the mechanics found in very complex biological systems, e.g. heart muscles. The key to success in design such networks will be the training of the neurons handling the information propagation through the structure. To be able to evaluate its dynamic behaviour, time domain simulation techniques are used. Some preliminary results of such simulations and their general implementation are presented in this paper.

Keywords: Simulation, neuro mechanical networks, large scale systems
Address: Magnus Sethson, Campus Valla Bld. A, IKP/MekSys, SE-58183 Linköping, SWEDEN, magse@ikp.liu.se

1. Introduction

By studying the body tissues of humans and animals one get a clear view of its cellular structure. Also muscle fibres reveals a ordered pattern of fibres, beneficial for its primary function or load direction. The structures in nature are built up by small basic elements that form a larger functional component. This might become the concept of future manufacturing technologies as well. Layered manufacturing and 3D-printing machinery is examples of that.

Several different types of basic elements might be used to form a working structure and by the mixture of different kinds of elements one get multi functional components that reflect the basic characteristics of the foundation elements. This is indeed an interesting approach to future mechanical design. However, it is still difficult to analysis the properties of a system or network of such elements, even if the overall structure has a general behaviour that is easy to model. The internal interaction among all elements may be hard to solve in detail. Computer tools for such analysis are needed. In fact the computer selection, simulation and optimization tools are essential to the development of selforganizing structures. In the same way that nature has evolved the designs of all spices, future computer tools will be able to select and evolve the interior of self actuating structures of advanced complex robots and machinery. This scenario is sometimes analogues to VLSI design within the field of electronics, where most of the present highly integrated designs would not exist without computational tools for both analysis and synthesis.

1.1. Project Presentation.

This project, called Neuro Mechanical Networks (NMN), tries to evaluate the properties of such networks for use in engineering systems design, [3][8]. In figure 1 is a simple layout of a neuro mechanical network presented. In general, the used approach introduce an actuating element containing several energy conversion processes and sensing capabilities, see section 2.3. By establishing software for simulation, selection and training of the neuro mechanical network one get an engineering tool that support the future study of the application of the network, its topology, and the characteristics of the building elements.

It is worth to emphasize that this work relates to the computational synthesis of systems of large numbers of small elements. Similar structures have been studied in statistical mechanical since the late eighties [4][6]. However, in this work entropy studies of the general behaviour of the network of elements is not studied, even though that is most likely interesting for the understanding of neuro mechanical networks.

Proceedings 17th European Simulation Multiconference
 ISBN 3-936150-25-7

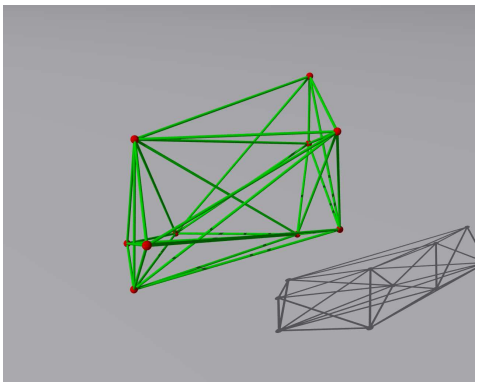

Figure 1: This is a typical layout of a neuro mechanical network. The spheres represent the nodes and the rods represent actuators. The topology is never changed during simulation.

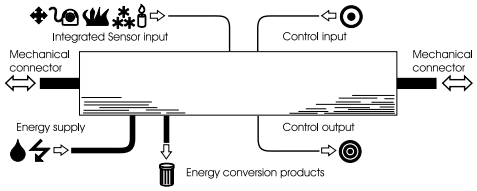

Figure 2: An idiomatic figure of a general actuator. It contain signal processing, sensing capabilities and energy conversation processes. Even though this is an element of the future it is most interesting to study for the understanding of more conventional designs.

The computational tools for this development process may use formal algebraic methods or optimizing functionality to fulfil the goal of the design. This is similar to electronics design tools making use of both analytical tools for circuit layout and different kinds of optimizing strategies for handling non-linearity phenomena and dynamics. The creation of mechanical actuation systems may in the future be dependent on small engineering elements in large numbers and computational synthesis tools. Such tools need to be able to configure the neuro mechanical network both in terms of topology and dynamical behaviour.

2. The Geometry of Neuro Mechanical Networks

The neuro mechanical network consists of a simple actuating element that performs either positional actuation, like a position servo, or a force actuation similar to a spring. In [3] a position servo approach has been adopted. This paper describes the force actuating approach.

Each actuator can be looked upon as a spring with and actuating element. Also a damping element is included. By tuning the basic properties of each actuators such as stiffness, actuation and damping one get a tool for creating more actuated bone-like structures with internal damping domains that dissipative energy from the system. The general layout of such actuator is shown in figure 2. The tuning process becomes very complex and computer power demanding. A tuning process that is similar to the selections of the fittest in nature is currently under study. There are several possible ways of tuning the individual actuator behaviours. In this work we use neural networks in every node controlling the actuation of every attached actuator. The actuator itself also has a neural network to balance the signals from each of its ends. The inputs to the neural network in the nodes are the actual actuation displacement of each actuator. The signal propagation velocity then becomes the same as the actuation velocity. The signal propagates with the same speed through the network as a mechanical wave of displacements in the actuators. This approach gives us one important benefit: It improves the numerical stability in the time domain simulation of the system. Another interesting approach uses the forces in the actuators as inputs to the nodes. This provides us with the ability to tune the actual rim or border of the neuro mechanical network. An actuator that has been tuned to no longer provide any force, having a low stiffness coefficient, does not transfer signals anymore. Therefore, the signal routes become consistent with the active mechanical structure and cavities in the structure may emerge in a natural way. The flow of signals is schematically shown in figure 3. By combining both displacement and force of the actuators into the neural networks of the nodes further generalization may be achieved. It is important that the time-domain simulation technique used supports all the proposed synapses of the neural network. That includes actuation length, sensors, position of elements and general signal fields. The trapezoidal integration rule for numerical integration seems to provide a good foundation for such requirement. In its most general form it may be formulated as in equation 1.

(1)
$$F\left[(n+1)T\right] = F\left[nT\right] + \frac{T}{2}\left(f\left[(n+1)T\right] + f\left[nT\right]\right)$$

Notice that the actuating and damping element most likely produces energy conversion products like heat or gases. These rest products need to be

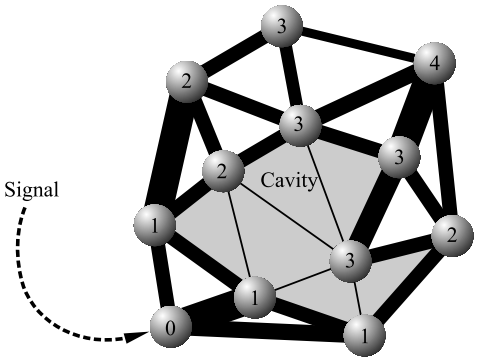

Figure 3: The general signal flow for controlling the neuro mechanical network. A signal is provided at the 0 node. The number in the node represents the time delay of the signal. Some of the actuators have just collapsed into a small line, unable to support any forces. Notice that no signals are routed through them.

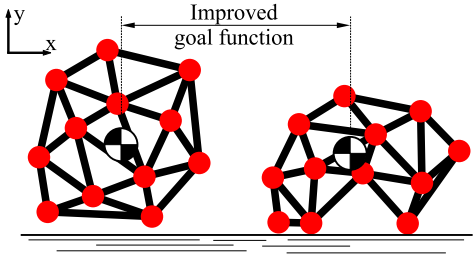

Figure 4: Example of a goal function and its direct relation to the structure. The network is trained to move along the x-axis. The wimple objective then becomes to maximise the x-coordinate of the centre of gravity for the structure.

transported away from the area. At the same time energy is required to be supplied into the system. These requirements for transport may either be inside the structure or parallel to it. This finding indicates a clear geometric dependency of the design problem. The actuator may not be modelled as a scalar symbolic element. Instead it needs to have length, thickness and position in space. The most general approach to an element finite in space will be the one-dimensional actuator connected at its ends towards other actuators. The connection may be free of friction and not supporting torque. The connecting joints will also host some of the controlling electronics for the neural signal propagation. By forming these simple actuators into a network we get a mechanical structure with large number degrees of freedom. The tuning of the actuators then becomes an engineering task making use of many traditional engineering disciplines in a new integrated way: the task is to make all these actuators respond to the environment and input signals in order to achieve some predefined objective. The time domain simulation technique becomes a natural choice for predicting the behaviours of a certain neuro mechanical network. One may compare this to our arms, which contain thousands of muscle fibres and still only have a few degrees of freedom.

2.1. **Time domain simulations and optimizations.** The application of time domain simulations for training the neurons is studied in this work. We also assume that the network perform some dynamic task like actuation or movement. Any static or quasi-static performance of the network will not be analyzed here. However, such analysis may very well be efficient and beneficial to the engineering process of tuning the neuro mechanical network. Such tuning will most likely rely upon algebraic methods and not relying on numerical algorithms.

A goal function is needed for designating a figure of merit on the actual configuration of a network. Since there are so many degrees of freedom, the goal function is formulated closely to the desired task of the network and not as an objective for each actuator. An example of that is shown in 4 where a desired movement along a certain direction is represented by the x-coordinate for the centre of gravity for the whole structure. The goal function then becomes very easy to formulate, it is simply to maximise the x-coordinate of the system. However, for large numbers of degrees of freedom this results in a huge set of possible solutions. There are several possible ways of dealing with this problem. One previously studied is the dimensional reduction of geometric models, especially triangular meshes [7]. Hierarchical geometric models created by applying simple rule sets may also be used. This is similar to cellular automata. In this work genetic algorithms have been selected in the first test designs of neuro mechanical networks. Its general characteristics seem to be beneficial to the design process at the current stage of the project. However this leads to a need for very large computational power. Other techniques may very well be used for network training. Still, in all possible approaches for training the network, a short simulation time becomes essential since each selection step may require thousands of simulations.

2.2. **Neurons.** The neurons of the network, both in the actuator and in the connecting nodes, are defined by the well known sigmoid function [2][1], see equations 2 and 3. s_0 in eq. 3 represent the input offset.

$$f(S_N) = \frac{2}{1 + e^{-\lambda(S_N - \Theta)}} - 1 \tag{2}$$

$$S_N = \sum_{n=0}^{N} s_n w_n \tag{3}$$

Since the mapping of the neural network is directly related to the topology of the mechanical actuator network we often get a neural network that is recurrent, signal may very well propagate in a cyclic way through the network. This could lead to standing waves in the structure, a kind of muscular limit cycle. The limitation of signal propagation speed then becomes important for the stability of the network.

2.3. **Actuator.** Here, the actuator is described by a rod having a certain stiffness K, parallel to an activation element F_A that can introduce a force in either direction. The relative motion between the ends of the rod is damped by a damping element B. The stiffness is separated into two springs, one attached at each nodes of the actuator. However, in this work the stiffness and damping properties are moved into the nodes for numerical reasons. This simplifies the simulation algorithm used.

The actual displacement of the actuators is used as neuron inputs to the neural networks in the nodes. There is no limitation in actuation length of the actuators. However, to support the neural network with a well defined input signal from the actuators, displacements, are normalized against a predefined individual nominal actuation length. Typically, values of 80% to 100% of the initial length of the actuator have been used. This means that an actuator is supposed, but not limited, to work only in the range of 50% to 150% of the initial length.

2.4. **Nodes.** The nodes contain the major part of the simulation task and it represents the mass of the system. All forces from the actuators are summed up to form a total force vector for each node. This vector is then integrated twice to get the new updated position of the node. The trapezoidal rule is used for integration, see equation 1. The trapezoidal rule for integration has a special interpretation in transmission line modelling [5] in that it represent the time delay for a wave travelling from one end to the other of a inertial element. In this case it can be simplified even more if all summation is done using the same T of equation 1. We assume that all signal propagations between nodes takes the same time, independent of distance. The assumption is motivated by the fact that the simulations are not primarily used for obtaining the most accurate physical behaviour, but used merely for selection and comparison between different simulations in an optimization scheme.

As in [5] the boundary conditions for the wave propagation elements is represented by characteristics of the form shown in equation 4 and 5. Normally the c and Z parameters are provided into the simulation component and the effort e and flow f are then solved for. In this work, however, the actuators are just calculating their length. The force acting upon the node is calculated by the node itself by using its current velocity and position. This approach gives us a numerical integration scheme that is computer memory linear. All calculations are done from two sequential lists of primitives, the actuators and nodes. This will improve the computational speed of the application. The memory storage requirement is also reduced since there is no need to save the c and Z variables, which simplifies even further the proposed numerical scheme. However, one should remember that the proposed simulation scheme does not represent an accurate physical model since the dynamics of the system becomes dependent upon the number of nodes and simulation time step.

$$e_1(t) = Z\left[f_1(t) + f_2(t-T)\right] + p_2(t-T) \tag{4}$$

$$e_2(t) = Z\left[f_2(t) + f_1(t-T)\right] + p_1(t-T) \tag{5}$$

Therefore the simple integration of the velocity and positional states of the nodes becomes as in equation 6 and 7.

$$\dot{x}_n = \dot{x}_{n-1}\frac{T}{2}\left[\ddot{x}_n + \ddot{x}_{n-1}\right] \tag{6}$$

$$x_n = x_{n-1}\frac{T}{2}\left[\dot{x}_n + \dot{x}_{n-1}\right] \tag{7}$$

The acceleration $\ddot{x}$ is defined by the mass m of each node and the summed force F_n.

2.5. **Numerical stability.** As said before, it is beneficial for the computational efficiency if the simulation algorithm has a linear memory layout. This can be achieved almost completely by introducing some approximations to the time domain simulation model. Let assume that the node are described by the following equation of momentum.

$$F_A - Kx - B\dot{x} = m\ddot{x} \tag{8}$$

This may be transformed into the frequency s-plane.

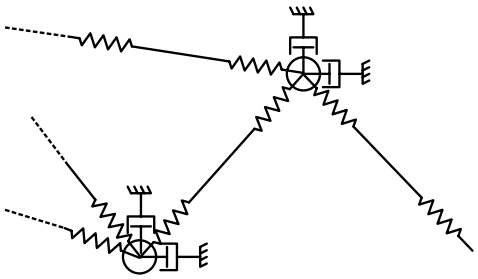

Figure 5: The general layout of actuators and nodes in the proposed simulation algorithm. Notice that the actuators do not include any dynamics.

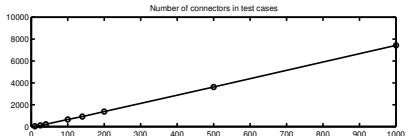

Figure 6: The number of actuators, A_N, scales almost linear with the number of nodes, N.

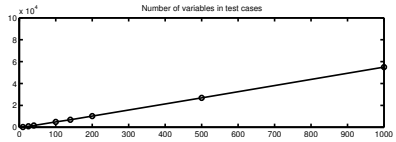

Figure 7: The number of design variables follows the number of connectors quite well, see figure 6. Still it is a function of topology, nodes and connectors.

$$F_A - KX - BsX = ms^2X \tag{9}$$

In solving equation 9 it becomes necessary to handle the numerical stability in a sensible way and at the same time keep the computational efficiency up. The proposed way of doing that tries to estimate an upper limit of the stiffness and damping factors for each node in the neuro mechanical network. The actuator stiffness may be defined in a traditional way.

$$\frac{\Delta F_A}{\Delta X}K = 1 + \frac{B}{K}s + \frac{m}{K}s^2 = 1 + \frac{2\delta}{\omega}s + \frac{s^2}{\omega^2} \tag{10}$$

Assuming optimal damping, that means real-valued roots to the polynomial in 10 we get an expression for B_{opt}.

$$B_{opt} = 2\sqrt{Km} \tag{11}$$

Even though the system might refers to a physical layout we apply a damping factor of B_opt to improve the numerical stability .The damping factor is applied in all dimensions of the nodes coordinates.

The actuators is attached to nodes by springs, see figure 5, that represent the stiffness of the actuator or rod connecting between two nodes. The worst case in terms of stiffness for all possible configurations of actuators and nodes is when all actuators lines up in one direction. A damper is introduced in every nodes origin in each coordinate direction. Normally we get one, two or three dampers in an orthogonal arrangement. The least damped system will be found when all actuators of a node are aligned to a coordinate axis. Notice that the damping is not applied in the actuators themselves but directly at the nodes. The K in eq. 11 refers to the total stiffness of all attached actuators of a node.

Another important factor for stability is the time step T for the numerical integration. As the trapezoidal rule has a complete stability region the expression for the maximum T to marching the state integration becomes as follows.

$$T_{opt} \leq \min_N \left[\pi\sqrt{\frac{m}{K}}\right] \tag{12}$$

The two limiting values of B_{opt} and T_{opt} may preferable be updated before the first time step of each simulation. This requires the stiffness K of each spring in the nodes to remain unchanged during all of the simulation. This might seems a severe limitation, but it only applies to the maximum stiffness achievable in the interior of the actuators. In most cases that is a well known factor. It is also possible to have different B_{opt} and T_{opt} for all nodes. This has not been studied further in this work but may be an interesting approach for further improve the computational speed. In the same way the time step may be varying across the structure. Some parts of the structure are then only updated every second or third time step. Equations 11 and 11 then becomes important for the sectioning of the different numerical domains of the structure.

3. Results

The characteristics of the simulation software is presented as diagrams in figure 6 to 9. The used genetic algorithm typically require a population size that is normally 10 to 50 times the number of design variables, N_d. This means that the computer memory storage requirement has a characteristic quadratic scaling to the number of design variables, $\propto {N_d}^2$. This is clearly seen in figure 8.

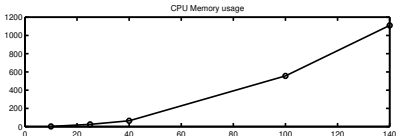

Figure 8: The amount of computer memory needed for the optimizations is growing exponential in a most unpleaseant way. From the diagram one can make the conclusion that on 32-bit machine only systems with less than about 200 nodes can be trained. This underlines the need for a better implementation of the framework.

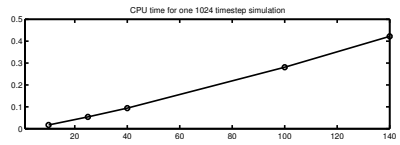

Figure 9: This is the simulation time for one 1024 simulation steps evaluation of the system. It is noticed that the simulation time does not drop even if the amount of used memory is higher than the available in the machine (768 Mb). This gives a hint that the memory problem of figure 8 is solvable.

4. Discussion

There are several aspects of the current implementation of the simulation software for neuro mechanical networks. One is the memory requirement. This will need to be addressed in the future. Since most of computational time is required for the simulation process one can think of several improvements to the current approach. The comparisons between different solutions during the optimization may very well be performed at each time step of the simulation and then it becomes possible to abort a simulation that has already shown a worse goal function value than the currently best one. The use of multi functional optimization is also interesting. The optimization scheme very often requires a population size proportional to the number of design variables times some constant. Typically the memory requirement in these applications follows a $50N_a^2$ scaling.

Another interesting area to improve the simulation time further is to make use of different time step in different parts of the structure. The equations 11 and 12 gives a clear hint of that. At the same time this will most likely be quite difficult to handle if the stiffness of the actuators are one of the objectives of the training or adoption phase.

5. Conclusions

A first attempt to simulate networks of self organizing mechanical structures has been presented. The important feature of linear scaling with the number of actuators reveals a promising future application for this scheme as a synthesis design tool. The exponential requirement for computer memory needs to be further studied. The use of bidirectional elements as the foundation element of the network has proved to give fast and accurate simulation results that can be used for optimization techniques and selection schemes. This forms an evolutionary design process to self-organizing structures. The memory requirement shows that it is possible to train a neuro mechanical network of about 1000 nodes on a regular 32-bit PC.

References

1. Adries P. Engelbrecht, *Computational intelligence, an introduction*, Wiley, University of Pretoria, South Africa, 2002.
2. Simon Haykin, *Neural networks, a comprehensive foundation*, second edition ed., Prentice Hall, 1999.
3. Petter Krus and Matts Karlsson, *Neuro-mechanical networks, self-organising multifunctional systems*, PTMC2002 (Edge, ed.), vol. 1, CRC Press, Sept 2002, pp. 22–44.
4. L.D. Landau and E.M. Lifshitz, *Statistical physics: Part1*, 3rd ed., Pergamon Press, 1980.
5. Jonas Larsson, *User's guid to hopsan, an integrated simulation enviorment*, Department of Mechanical Engineering, Linkoping Institute of Technology, August 2002, Avaliable at hydra.ikp.liu.se.
6. G. Parisi, *Statistical field theory*, Addison-Wesley, Reading,MA, 1988.
7. Magnus Sethson, *Complex cavity analysis : analytical fluid-power models using cad information*, Dissertations, no. 576, Linkoping studies in science and technology. Dissertations, June 1999.
8. Magnus Sethson, Matts Karlsson, and Petter Krus, *Neuro-mechnical networks as an architecture for system design*, Computational Synthesis, AAAI, March 2003.

Magnus Sethson, PhD. Assistant Professor at Division of Engineering Systems, Department of Mechanical Engineering, Linköping University, Sweden. magse@ikp.liu.se. Magnus research and engineering studies originates from his special interest in geometry and mechatronics. After a stay within the military aviation industry as a CAD-systems programmer he started his M.Sc. in the late of the eighties. In 1999 he presented his PhD. thesis on the topic "Complex Cavity Analysis" covering different automatic dimensional reduction schemes for CAD-systems. The current research interest includes evolutionary design of mechatronic systems, multi-actuator systems and their relation to geometric information systems. He is also lecturing in mechatronics and motion control systems. Current research projects focus on autonomous vehicle control and steer by wire systems. He is also a keen programmer within many technical areas and maintains some open source projects within genetic algorithms. In his spare time he is most often found behind the camera as an amateur photographer.