Published in final edited form as:
Presence (Camb). 2011 August ; 20(4): 289–308.

A Physics-driven Neural Networks-based Simulation System (PhyNNeSS) for multimodal interactive virtual environments involving nonlinear deformable objects

Suvranu De, Sc.D., **Dhannanjay Deo, PhD**, **Ganesh Sankaranarayanan, PhD**, and **Venkata S. Arikatla, MS**

Suvranu De is a professor and Director of the Center for Modeling, Simulation and Imaging in Engineering, Rensselaer Polytechnic Institute, Troy, USA

Dhannanjay Deo is an employee of Kitware Inc., USA

Ganesh Sankaranarayanan is a research associate at the Center for Modeling, Simulation and Imaging in Engineering, Rensselaer Polytechnic Institute, Troy, USA

Venkata S. Arikatla is a graduate student in the Department of Mechanical, Aerospace and Nuclear Engineering at the Rensselaer Polytechnic Institute, Troy, USA

Abstract

Background—While an update rate of 30 Hz is considered adequate for real time graphics, a much higher update rate of about 1 kHz is necessary for haptics. Physics-based modeling of deformable objects, especially when large nonlinear deformations and complex nonlinear material properties are involved, at these very high rates is one of the most challenging tasks in the development of real time simulation systems. While some specialized solutions exist, there is no general solution for arbitrary nonlinearities.

Methods—In this work we present PhyNNeSS - a Physics-driven Neural Networks-based Simulation System - to address this long-standing technical challenge. The first step is an off-line pre-computation step in which a database is generated by applying carefully prescribed displacements to each node of the finite element models of the deformable objects. In the next step, the data is condensed into a set of coefficients describing neurons of a Radial Basis Function network (RBFN). During real-time computation, these neural networks are used to reconstruct the deformation fields as well as the interaction forces.

Results—We present realistic simulation examples from interactive surgical simulation with real time force feedback. As an example, we have developed a deformable human stomach model and a Penrose-drain model used in the Fundamentals of Laparoscopic Surgery (FLS) training tool box.

Conclusions—A unique computational modeling system has been developed that is capable of simulating the response of nonlinear deformable objects in real time. The method distinguishes itself from previous efforts in that a systematic physics-based pre-computational step allows training of neural networks which may be used in real time simulations. We show, through careful error analysis, that the scheme is scalable, with the accuracy being controlled by the number of neurons used in the simulation. PhyNNeSS has been integrated into SoFMIS (Software Framework for Multimodal Interactive Simulation) for general use.

Corresponding author: Suvranu De, Address: JEC 5002, Department of Mechanical, Aerospace and Nuclear Engineering, Rensselaer Polytechnic Institute, 110 8th Street, Troy, NY 12180, USA, des@rpi.edu, Office: +1-518-276-6096, Fax: +1-518-276-6025.

1. Background

Real-time physics-based interactive simulation is computationally very demanding (C. Basdogan et al., 2004). While update rates of at least 30 Hz are necessary for real time graphical display, real time display of interaction forces through a haptic interface device such as a PHANToM, requires a much higher update rate of about 1 kHz. To support virtual environments that include interactions with deformable objects, efficient modeling schemes must be used (C. Basdogan et al., 2004). Geometry-based models developed by Delp, Loan, & Cagatay Basdogan (1997); Ho, Cagatay Basdogan, & Mandayam A. Srinivasan (1999) tend to sacrifice the physics of the deformation process to achieve real time performance. Physics-based techniques that take into account the underlying mechanics of soft tissue deformation using mass spring networks (Kuehnapfel & Neisius, 1993) or finite elements (Bro-Nielsen, 1998) have been developed. Neural networks mimicking mass-spring dynamics are used by Radetzky, Nürnberger, & Pretschner (1998)

Mass-spring systems are notorious for their instabilities and the necessity to “tweak” thousands of parameters. Finite element-based techniques are more accurate, very stable, yet computationally expensive. Techniques such as condensation (Bro-Nielsen, 1998) and precomputation (Picinbono, Delingette, & Ayache, 2003) have been used to accelerate finite element computations. Meshfree methods such as the point associated finite field (PAFF) offer yet another alternative to finite element techniques and several methods have been presented to reduce computational costs (Suvranu De, Lim, Manivannan, & Mandayam A. Srinivasan, 2006). Hardware acceleration has also been used to achieve real time rates (Liu & Suvranu De, 2008).

The presence of nonlinearities in the response of the deformable objects further complicates the problem as the simulation must now be iterative (Lim & Suvranu De, 2007). Nonlinearities may be introduced due to large deformations, when the strain is a nonlinear function of the displacement field, and when the material properties are nonlinear, e.g., hyper elastic or inelastic. Of course, possible solutions pursued by us and others include faster hardware, clever computational algorithms, and use of massively parallelized systems. A detailed review of simulation of nonlinear response of deformable objects in real time maybe found in work by Lim & Suvranu De (2007). As the capability of computing resources has increased over the years, so have the complexities of the simulation scenarios. Using an example from surgical simulation, a typical surgical scene involves much more than just tool-tissue interaction. Simulations of multiple organs, medical devices, sutures, coupled fluid flows due to bleeding, and modeling of physiological consequences of surgical procedures are now becoming common (M. Harders et al., 2006). With increasing complexity of surgical simulation scenarios, it is now clear that a straightforward technique of solving partial differential equations will not be sufficient – at least on commodity hardware.

In this paper we present a very general approach – the Physics-driven Neural Networks-based Simulation System (PhyNNeSS) (D. Deo & S. De, 2009) for real time simulation of linear and nonlinear response of deformable objects. The first step in PhyNNeSS is to develop geometric models of the deformable objects, ascribe them with realistic material properties (linear or nonlinear) which may be obtained from actual physical experiments and simulate them with carefully applied loads. An accurate physics-based computational scheme such as the finite element or meshfree method may be used in this simulation. This is a pre-processing step and need not be computationally efficient. This step generates a massive database which is used to train a set of radial basis function neural networks. During real time computations, these neural networks are used to compute the deformation fields and the reaction forces.

The significance of the method is that linear and nonlinear simulations may be performed with almost the same operational complexity. Additionally, the quality of the real time computations may be easily controlled by scaling the number of neurons used in the computations. This system provides a unique platform to leverage the computational speed and scalability of soft computation methods for real time interactive simulations. Neural networks, however, have been previously used in surgical simulation. In the following paragraphs we will review existing literature in this area.

Neural networks have been utilized in surgery simulation in specialized applications. For simulation of cutting involving liver, a neuro-fuzzy system was developed by Weiguo & Kui (2005). To obtain the training data, a specialized scalpel was designed for the acquisition of cutting parameters. However, very few parameters may be recorded in such experiments sacrificing the fidelity of the simulations. A geometry-based mesh deformation algorithm was proposed by De Boer, van der Schoot, & Bijl (2007) which interpolates displacements of the boundary nodes to interior modes using radial basis functions (RBF's).

Neural networks have also been used in energy-based simulation of deformation (Zhong, Shirinzadeh, Alici, & Smith, 2006). But this approach is not truly physics-based as they draw an analogy between a cellular neural network (CNN) and elastic deformation. The potential energy stored in an elastic body as a result of a deformation caused by an external force is propagated among mass points by a nonlinear CNN. Another model has been proposed for animation by Grzeszczuk, Terzopoulos and Hinton (1998), in which large neural networks are trained to emulate a simple physical system. This recent approach has not been proven practical for large coupled systems of nonlinear partial differential equations.

Simulation of structural systems in virtual reality applications using neural networks trained from finite element simulations was performed by Hambli, Chamekh, & Bel Hadj Salah (2006). In the field of design optimization response surfaces have been constructed using radial basis functions. Radial basis function-based meta models have been developed by Mullur and Messac (2006) and they have been used for representation of static geometrical surfaces obtained from point clouds by Carr et al., (2001). Neural networks have also been used by Omar, Eskandarian and Bedew, (1998) to approximate the failure surface in vehicle crash modeling. In mechanics, neural networks have been used to simplify the responses of models (Kallassy, 2003) also the validity of using neural networks for approximations of simple mechanical systems is presented in. In finite element analysis neural networks have also been used for the approximation of complex visco-plastic constitutive relations for soil and cement mechanics (Jung & Ghaboussi, 2006). A review of applications of neural networks in structural engineering is provided by Waszczyszyn and Ziemianski (2001).

A learning algorithm to train a linear 2D mass-spring-damper system that behaves similarly to a high-fidelity nonlinear finite element (FE) model is presented by Bianchi, Solenthaler,Gábor Székely and Matthias Harders (2004). In work by Peterlik and Matyska (2007) a neural network based approach is presented which involves a pre-computation phase much like PhyNNeSS, however, they use state-space based methods to interpret this data, unlike PhyNNeSS. In the work by Höver, Kósa, Gábor Székely, and Matthias Harders (2009) a data driven haptic rendering method was proposed for rendering of forces during the interaction with visco-elastic materials. This method utilizes RBF for interpolating force data based on positions and velocities. Unlike PhyNNeSS, this method cannot be generalized for complex geometries.

Acquiring a computer model by directly recording deformation and force feedback information is presented in work by Pai et al. (2001). Acquired deformations were used to capture spatially varying strain-stress relationships in work by Bickel et al. (2009) to compute nonlinear deformations but their approach does not reflect true physics of the objects involved.

In work by Barbič and James (2007), the authors presented an approach to provide 6-DoF force feedback for reduced deformable body. Saupin, Duriez, and Cotin (2008)'s work also focuses on contact handling for haptic rendering of non-linear organ models during surgery simulation.

The organization of this paper is as follows. We introduce PhyNNeSS in section 2, followed by its application to interactive surgical simulation in section 3 and some concluding remarks in section 4.

2. Methods

Figure 1 presents the three major steps of PhyNNeSS. The first stage is data generation. This is a pre-computation step in which geometric models of the deformable models are generated and are ascribed with boundary conditions and realistic mechanical properties, possibly based on physical experiments. The models may be discretized either using finite elements or by using a distribution of nodal points for meshfree analysis (Suvranu De et al., 2006). In the present work we have chosen finite elements. Each model is then simulated by prescribing carefully chosen displacements at each node. The response, in terms of the reaction forces, is computed using finite elements, and stored in a large database. The second step of PhyNNeSS consists of machine learning in which the data is vastly condensed and stored as a set of coefficients describing neurons of a Radial Basis Function network (RBFN) (Park & Sandberg, 1991). The third and final step of PhyNNeSS is real time simulation in which the neural networks are used to reconstruct the deformation fields as well as the reaction forces.

Using a combination of hard and soft computing methods, PhyNNeSS is able to reduce the solution of nonlinear problems to almost the same runtime complexity as solving linear problems. This technique is not only extremely rapid, but also scalable – which implies that we have a ‘control knob’ which can be turned up or down to control the accuracy of the solution. The scalability is controlled by the choice of the number of neurons in the RBFN. More neurons may be chosen for higher fidelity but slower simulation while fewer neurons may be chosen for coarser, but more rapid simulation. In the following subsections we present details of each of the three steps of PhyNNeSS.

2.1. Data generation

The data generation step involves simulating the organ models and solving the equilibrium equations of mechanics accurately using finite elements. In this section we will briefly recapitulate the equations governing the deformation of a general nonlinear continuum and its finite element analysis (Bathe, 1996). The nonlinearity may be due to large deformations (geometric nonlinearities) or nonlinear stress-strain relationships (material nonlinearity) or a combination of both.

Let R_0 denote a deforming continuum at time t=0 (the ‘reference’ configuration) as shown in Figure 2, which at time ‘t’ occupies R_t (the ‘current’ configuration). We employ a Lagrangian description in which a point $\mathbf{X} = \{X_1, X_2, X_3\}$ in the reference configuration (known as ‘material point’) deforms to the point $\mathbf{x} = \{x_1, x_2, x_3\}$ in the current configuration (known as ‘spatial point’) using the map

$$x=x\,(X,\mathbf{t}) \quad (1)$$

The displacement vector at any point is

$$u=x-X \quad (2)$$

The deformation gradient tensor **F** is defined as

$$F=\nabla x=I+\nabla u \quad (3)$$

where the gradient is taken with respect to **X**.

The material or Green-Lagrange strain is

$$E=\frac{\mathbf{1}}{\mathbf{2}}\left(F^{T}F-I\right) \quad (4)$$

where **I** is the identity tensor.

In finite element analysis, the following weak form of the governing differential equations is solved (Bathe, 1996) to compute the displacement field

$$\int_{\Omega_0}\delta E^{T}:S\,\mathbf{d\Omega}=\int_{\mathbf{\Omega_0}}\delta u^{\mathbf{T}}b^{\mathbf{0}}\mathbf{d\Omega}+\int_{\mathbf{\Gamma_t^0}}\delta u^{\mathbf{T}}t^{-\mathbf{0}}\mathbf{d\Gamma} \quad (5)$$

satisfying prescribed displacements on portion Γ_u^0 of the boundary. In this equation, '**S**' is the second Piola-Kirchhoff stress tensor; $\mathbf{b}^0$ is the body force per unit reference volume (including inertia forces) and $\mathbf{t}^{-0}$ is the surface traction on the boundary Γ_t^0. The operator δ represents a first order variation of the respective mathematical entities.

In this paper, we will consider hyper-elastic materials only, i.e., we will assume that the stress (S) may be obtained from a scalar strain energy density function using the following relationship:

$$S=\frac{\partial \mathbf{W}}{\partial E} \quad (6)$$

While there are multiple expressions of 'W' available in the literature describing the response of deformable materials, we will use the following models (Bonet & Wood, 1999) in the examples presented in this paper:

(1) St. Venant-Kirchhoff

$$W=\frac{1}{2}\lambda(tr(E)^{2}+\mu.tr(E^{2}) \quad (7)$$

where μ and λ are Lame constants and **E** is the Green strain tensor.

(2) Neo-Hookean

$$W=\frac{\mu}{2}(\mathrm{I}_c-3)-\mu.\ln J+\frac{\lambda}{2}(\ln J)^2 \quad (8)$$

Where, $J = \det(\mathbf{F})$

(3) Mooney-Rivlin

$$W=\mu_{10}(I_{\mathrm{C}}-3)-\frac{1}{2}\mu_{01}(I^2{}_{\mathrm{C}}-II_{\mathrm{C}}-6) \quad (9)$$

Where, $I_{\mathbf{C}}$ and $II_{\mathbf{C}}$ are the first and second invariants of the right Cauchy-Green tensor $\mathbf{C} = \mathbf{F}^T\mathbf{F}$.

The weak form in equation 5 is solved numerically using finite element discretization (Bathe, 1996) where the displacement field (**u**) is approximated in terms of the nodal point displacements (**U**) as

$$\mathrm{u}(\mathrm{X},\mathbf{t})=\mathrm{H}(\mathrm{X})\mathrm{U}(\mathbf{t}) \quad (10)$$

Where, **H** is the matrix of nodal shape functions expressed in terms of the reference configuration. After imposing the displacement boundary conditions, the solution of equation 5 proceeds in an iterative manner. The most commonly employed technique is the Newton-Raphson method. For a nonlinear analysis, it is customary to divide the load into smaller load steps and allow the iterations to proceed in an incremental manner from one load step to the other.

To completely characterize the response of any nonlinear model and store it in a database, ideally, each finite element node must be deformed in all possible directions to all possible magnitudes of displacements and the displacements at all other nodes in all three Cartesian directions and the reaction forces at the simulated node must be recorded. Such an exercise is not only infeasible, but also not necessary. In our technique, we fix a few of the nodes to apply fixed boundary conditions. Each of the remaining nodes is then displaced by a predetermined amount δ along one of only 26 directions (Figure 3), which we are going to refer to as “exploratory directions”. We will refer to the node which is being displaced as the “active node”. The 26 exploratory directions correspond to 26 unit outward normals to a unit sphere, denoted by $\{\mathbf{e}_i\}_{i=1,\ldots,26}$, as shown in Figure 3, centered at each active node. The result of each simulation is a “response set” – that includes the coordinates and labels of the active node, the direction and magnitude of the imposed displacement, the converged values of the reaction force, and a list of all other deformable nodes with their coordinates and the computed displacements. Typically, each active node is displaced ‘n’ times successively along each exploratory direction so that the total displacement at that node is Δ = nδ (see Algorithm 1). This entire data generation step is automated based on a script written in Python. It should be noted that for linear analysis, it is sufficient to sample data along the three Cartesian directions. However, since linear superposition does not hold in nonlinear analysis, we have chosen the 26 directions. The choice is not unique. More directions can, of course, be chosen along normals to the unit sphere. However, this will increase computational costs and storage requirements. Besides, as we show in our results, sufficient accuracy is achieved using the 26 exploratory directions.

Algorithm 1: Data generation using finite element analysis

```
Let N=total number of nodes in the model
```

```
Apply fixed boundary conditions to $N^{fixed}$ ($<N$) nodes
**for** *each remaining active node* $J=1,..,N^{active}$ ($=N-N^{fixed}$)
  **for** *each exploratory direction* $\{\mathbf{e}_i\}_{i=1,\ldots,26}$
    $\Delta = \delta$.
    **while** $\Delta \le n\delta$
      apply displacement $\Delta\mathbf{e}_i$ to node $J$
      solve nonlinear finite elements
      compute and store reaction force $\mathbf{f}_J$ at node $J$
      **for** $I=1,\ldots,N^{active}$, $I \ne J$
        store displacements $\mathbf{u}_I$
      **end for**
      $\Delta \mathrel{+}= \delta$
    **end while**
  **end for**
**end for**
```

All the data generated using finite element analysis is not necessary for training the neural networks (see section 2.2). It has been shown in Kuroe and Kawakami (2007) that a few vectors from a vector field are sufficient for generalizing the response using a network. For most effective approximation of input and output mapping using RBFNs, all data columns must be mapped to a uniform variation. All data fields are centered and scaled to lie between −1 and +1. The values for centering and scaling are also recorded so that same transformations may be applied to inputs during simulation.

2.2. Machine learning

The data generated as part of the finite element preprocessing step is used to train the RBFN (Park & Sandberg, 1993). The basic topology of a RBFN is shown in Figure 4. Two kinds of RBFNs are designed – a ‘force network’ and a ‘deformation network’. The imposed displacement vector at the active node $\delta\mathbf{e}_i$ corresponding to each displacement increment (δ) of each of the 26 exploratory directions ($\mathbf{e}_i$) serves as the common input to both these networks. However, the force network is trained to compute the three components of the reaction force at the active node while the deformation network is trained to compute the displacements of all nodes where the displacements are not prescribed. In addition to the displacements at the active node, the input to the displacement network includes the nodal coordinates of the other nodes. Thus each data point (input – output pair) of deformation network has 6 components of input and 3 components of output, and each data point of force network has 3 components of input and 3 components of output.

Neural networks are black box functions capable of performing a non-linear mapping between a vector input $\mathbf{X} \in \mathbf{R}^{m}$ and a vector output $\mathbf{Y} \in \mathbf{R}^{m}$. We use a feed-forward neural network with a single hidden layer of radial basis function neurons which are capable of approximating nonlinear functions (Abe & Iguni, 2006). The activation level of each neuron in the hidden layer is computed using a radial basis function of the following form:

$$h_i(\mathrm{x}) = \exp\left(\frac{\|\mathrm{x} - \mathrm{c}_\mathbf{i}\|^2}{\mathbf{r}_\mathbf{i}^2}\right) \quad (11)$$

where ‖•‖ represents the Euclidean norm, $\boldsymbol{c_i} \in \mathbf{R}^{m}$ is a vector of centers and the ‘r_i’ s are the scalar spread parameters. The output is obtained by a weighted (linear) combination of the outputs of the hidden layer neurons and a bias. The ‘jth’ component of the output vector is computed as

$$y_j = w_{j0} + \sum_{i=1}^{m} w_{ji} h_i(\mathrm{x}) \quad (12)$$

Where, w_{j0} is the bias for j^{th} output and w_{ji} are scalar weights. Another possibility is to use weighted averages, which has higher computational complexity.

The goal of the RBFN training is to determine the parameters of the hidden layer (i.e., the centers $\mathbf{c}_i$, and the spread parameters r_i) and obtain the output layer weights in such a manner that optimizes the fit between the RBFN output and the training data obtained from finite element simulations. The fit is evaluated in terms of the difference between RBFN output ($\mathbf{y}$) and the finite element output ($\hat{\mathrm{y}}$)

$$\mathrm{e} = \widehat{\mathrm{y}} - \mathrm{y} \quad (13)$$

e is then used to define the mean squared error (MSE)

$$MSE = \frac{\mathrm{e}^T \mathrm{e}}{n} \quad (14)$$

Though the network topology allows for different ‘r_i’ (spread parameter) for each of the hidden neurons, it has been shown that a uniform spread parameter is sufficient for arbitrarily accurate approximation of any function (Park & Sandberg, 1993). In this work, we have used a single value of spread parameter for all the neurons. The optimal value of the spread parameter depends on the model being simulated and may be found by performing a parametric study as outlined in section 3 (Figure 9).

RBFNs are usually trained in a supervised manner, i.e., a user selected set of training data points (input-output pairs) is provided which is used to compute the necessary parameters. The ability of a neural network to efficiently approximate a nonlinear function depends on the selection the centers of the hidden layer (Orr, 1995; Valdes, Biscay, & Jimenez, 1999). We have adopted an orthogonal least square selection method which is recommended when the training set is large (Sherstinsky & Picard, 1996). In this method a Gram-Schmidt type approach is employed. The network is initiated with empty (zero neurons) hidden layer. The network is then systematically grown by addition of one radial basis neuron in the hidden layer during each iteration. The center of each new RBF neuron is selected to coincide with the data point, which generated the highest error in the previous iteration (Chen, Billings, & Luo, 1989; Yu, Gomm, & Williams, 1997). Once the centers of the hidden layer are chosen, the weights and offsets in equation 12 are determined using a least squares minimization of the error between desired and simulated outputs. The training process is continued until the MSE is less than a desired value, or the hidden layer reaches a maximum number of neurons.

2.3. Real time simulation

The coefficients obtained from the RBFN training are exported to text files with mesh information to use in real time simulation. During real time computation, an efficient collision detection routine determines the node/s to be displaced. The force neural network is used to compute the reaction forces. The deformation neural network is used to compute the deformation of the nodes at which displacements are not specified.

During runtime, the deformation and reaction force at each movable node are computed according to equations 11 and 12. If there are ‘m’ inputs and the hidden layer consists of η neurons, then the total number of floating point operations per output is estimated as $O(\eta(3m + E+2))$ where E operations are necessary for calculating an exponential operation. The computation of the terms in equation 11 requires $O(3m + E)$ operations which must be repeated for η neurons. To accomplish the computations in equation 12, $O(2\eta)$ operations must be performed. Hence the cost of deforming all the movable nodes of the mesh in response to an interaction thus scales linearly with the number of neurons in hidden layer (η).

The information generated in the precomputation stage corresponds to a single point interaction with the finite element mesh. However, real time simulations are not restricted to a single point. If deformations $\mathbf{U}_I$; $I = 1,2,\ldots M$ are applied at ‘M’ nodes with position vectors $\mathbf{x}_I$; $I = 1,2,\ldots,M$, then we compute the resultant deformation at any point $\mathbf{x}$ as

$$u(x)=\sum_{\mathbf{I=1}}^{\mathbf{M}}\boldsymbol{\Theta}_{\mathbf{I}}\tilde{\mathrm{U}}_{\mathbf{I}} \quad (15)$$

Where, $\tilde{\mathrm{U}}_I$ is the displacement vector computed at $\mathbf{x}$ due $\mathbf{U}_I$ and the weights

$$\Theta_I=\frac{\frac{1}{\|x-x_{\mathbf{I}}\|}}{\sum_{J=1}^{M}\frac{1}{\|x-x_{\mathbf{J}}\|}} \quad (16)$$

For interaction with M nodes using this technique, the operation count is $O(\eta M(3m + E + 2) + 10.M^2 + 10M - 1)$ which shows that the computations grow linearly with the number of hidden layer neurons.

PhyNNeSS is integrated into our custom developed software framework for multimodal interactive simulation (SoFMIS) (Halic et al., 2011).

3. Results

While PhyNNeSS is a very general algorithm applicable to any interactive environment, in this paper we will discuss the application of PhyNNeSS to real time surgical simulation. We have used two models of varying complexity: a liver model (Figure 5a) and a Penrose drain model (Figure 5b), which is a soft rubber tube placed in a wound to drain fluids. The liver model is discretized using 2341 tetrahedral volumetric finite elements. The Penrose drain model is a hollow tube, with a slit on the top and 436 shell elements are used. The following four simulation scenarios have been considered:

CASE 1

The liver model is simulated within the linear small deformation regime using a St. Venant-Kirchhoff material model (equation 7) with Young's modulus (E) = 20 and Poisson's ratio (μ) = 0.45. In literature, the Young's modulus of liver has been reported to be between 8 and 48 kPa with (Nava, Mazza, Furrer, Villiger, & Reinhart, 2008) reporting 20 kPa, hence the choice of these material constants is realistic. Case 1 corresponds to linear elastic material and small deformations, i.e., linear geometric model.

CASE 2

The liver model is simulated for large deformation loading using a St. Venant Kirchhoff material model (equation 7) with E = 20 kPa and $\mu = 0.45$. This case corresponds to linear material model but nonlinear geometric model.

CASE 3

The liver model is simulated for large deformation loading using a neo-Hookean material model (equation 8) with the following model parameters: $C_{10} = 250$Pa and bulk modulus = 100 kPa ($D_1 = 0.00001$) based on work by B. Ahn and Jung Kim (2010). The neo-Hookean material model is widely used for soft tissue simulation (Kauer, Vuskovic, Dual, Szekely, & Bajka, 2001; Miller, 2005; G Székely et al., 2000). This case corresponds to nonlinear material and nonlinear geometric model.

CASE 4

The Penrose drain model is simulated for large deformation loading using a Mooney-Rivlin material model (equation 9) with the following model parameters: C_{01} =0.2 Pa, C_{10}=2 Pa and bulk modulus = 100 kPa ($D_1 = 0.00001$). Penrose drains are made up of rubber for which the Mooney-Rivlin model is appropriate (Edwards & Tabor, 1976; Lake & Thomas, 1967). This case corresponds to nonlinear material and nonlinear geometric model.

With reference to these four cases we will discuss the errors in the computation and how the training parameters may be used to control these errors. A Pentium 4, 3 GHz single core workstation with 2GB RAM and two connected PHANToM Omni force feedback devices was used for pre-computations and real-time computations. Table 1 provides a summary of the model specifications, resource requirements and computational times. We can see that the vast data generated by the finite element computations is condensed into coefficients of the RBFNs with storage requirements as low as a few Mbs, and still can rapidly reproduce the response in real-time.

To demonstrate the ability of PhyNNeSS to accurately model response for nodal interactions, we have graphically plotted the errors between the response recorded from finite elements and as approximated by the RBFN. Let $\mathbf{y}_{\boldsymbol{i}} \in \mathbf{R}^3$ be the three components of the output (force or displacement) corresponding to the ith test point computed using the neural network and $\hat{\mathbf{y}}_{\boldsymbol{i}} \in \mathbf{R}^3$ be the corresponding finite element solution. We define the ith component of the 'relative error' as

$$\text{Rel\_ error}_i = \frac{\|e_i\|}{\max_j \left(\|y_j\|\right)} \text{with}\, e_i = y_i - \widehat{y}_i \qquad (17)$$

The 'mean relative error' is computed as the average of all the relative errors. The neural network should be capable of simulating results for the data points which were not used in training. Hence all the error computations are done with data points which are different from those used in training.

In Figure 6 we plot the force displacement relationships for all three cases of the liver model (Cases 1-3 above) comparing the predictions of PhyNNeSS with that of finite elements when 50, 75 and 100 neurons are used in the network. The neural networks were trained with data for tool displacement of up to 2.8 cm. However, the model predictions are for a maximum of 4 cm tool displacement. These plots clearly show that predictions using PhyNNeSS compare well with finite element predictions beyond their range of calibration. This is true for both linear and nonlinear predictions, irrespective of whether the nonlinearity arises from the material model or due to large deformations. It is also clear that the approximation accuracy improves with increasing number of neurons in the network. The only exception is for Case 2 where the 100 neuron network seems to be performing inferior to the 75 neuron case. A possible explanation could be due to over fitting of the data (Samarasinghe, 2006). Also, the convergence may be non-monotonic with respect to the number of neurons, an issue that requires further analysis to resolve. In the remainder of this section we will present results for Cases 3 and 4 only. Figure 7 presents a histogram of the percentage relative error in the displacement for Case 3 using 3000 interaction cases (data points) of deformation computations and different sizes of the network. As the network is grown with more neurons in the hidden layer, the approximation improves shifting the peak of the error histogram towards the left, i.e. for the 20 neuron case the peak is at 5% relative error which is 2% for the 100 neuron and < 2% for the 300 neuron network. For the 300 neuron network, most of very few test points have errors beyond 6%. In Figure 8, we plot the % mean relative error as a function of the number of neurons in the hidden layer for both Cases 3 (liver) and 4 (Penrose drain model). 5000 input-output pairs were used for training to generate each of the networks and errors were computed using 3000 pairs not used in the training. The errors reduce super linearly as the neurons in the hidden layer are increased. However, the rate of decrease slows down with increase in the number of neurons indicating that a very large number of neurons may not be necessary for an acceptable level of error based on the “just noticeable difference” or JND of the human sensory system which for force perception at human fingertips is about 10% (Allin, Matsuoka, & Klatzky, 2002; Pongrac, Färber, Hinterseer, Kammerl, & Steinbach, 2006; Tan et al., 1994)

Figure 9 shows the percentage mean relative error as a function of the spread parameter (r_i) in equation 11 where a single spread parameter has been assumed for all the neurons. We observe that a spread parameter of about 0.75 results in the minimum mean relative error in the liver model, however a spread parameter of 1 is considered more appropriate for the Penrose drain model.

Another important variable in our study is the effect of the size of the training data set on the accuracy of the displacement prediction. In Figure 10, we have plotted the percentage mean error with increase in data points for training. For this study, we have used a network of 200 neurons and 3000 input-output pairs (different from those used in training) were used to validate the resulting neural network. In general, the results indicate that while larger training datasets reduce error, the advantage of using very large sets becomes less significant. A point to note is that the reduction of relative error is not strictly monotonic which explains the slight increase in error for the Penrose drain model beyond 4000 data points. Using graphs like these, one may decide the size of the training set for a desired level of accuracy.

In Figure 11 we present the percentage mean relative error in the force computation as a function of the number of neurons in the hidden layer. The force RBFN has much less data points than the deformation RBFN. Hence we have used 100 data points for training the networks and the error is observed at 30 test cases which are different from the data points used in the training.

In Figure 12 we show the percentage max relative error in the force computation as a function of the number of neurons in the hidden layer. In the figure on can see that the % max relative error is below the JND with 87 neurons for liver mode l and 63 neurons for the Penrose drain model.

Next we address the issue of how the computational time scales with increase in mesh resolution. Figure 13 shows three meshes of increasing resolution derived from the original mesh model of the liver using successive subdivision based on the Loop subdivision algorithm (Loop, n.d.). Details of the higher resolution meshes are given in Table 2. Pre-computed neural networks used in the simulation of the three higher resolution meshes are the same as that for the lower resolution mesh in Figure 13a. Figure 14 shows that the performance of PhyNNeSS scales linearly with the number of vertices in the mesh.

The final example concerns interactions of the liver model with two tools. In this experiment, two nodes are selected on the surface of the liver model and displaced randomly along two independent directions chosen from among the 26 exploratory directions in Figure 3. The results are reported for maximum of the two displacements. The histogram of errors in the displacements computed using equation 15 and finite element simulations is presented in Figure 15. As expected, the error increases with increase in the depth of indentation, however, the maximum error for the majority of the points is within 2-4%.

4. Concluding remarks

In this paper we have presented a novel method (PhyNNeSS) for real-time physics-based interactive simulation involving deformable objects which may have arbitrary nonlinearities. A database is first generated by systematically probing a finite element model of the deformable objects, which is then utilized to train radial basis function neural networks. The networks are then used in real time computations. We have analyzed the error in computing the displacements and forces when the number of neurons as well as the training data set is increased.

PhyNNeSS results in substantial reduction in runtime computational costs and increased accuracy by leveraging the use of offline computations leading to enriched visual experience on commodity hardware. It is important to note that the neural network simulation is a relatively rapid process with the number of operations scaling linearly with the number of neurons in the hidden layer. We have observed that the number of neurons required for a desired level of accuracy depends on the complexity of the model, e.g. for the liver model, 100-300 neurons result in deformation fields which are visually indistinguishable from fields generated using much larger number of neurons as the mean relative error lies between 2-3%. For this model the performance scales from 154 frames per second for 300 neurons to about 700 frames per second for 100 neurons, with the force feedback running at haptic rates of 1000 Hz.

PhyNNeSS is not only extremely rapid, but also scalable – which implies that we have a ‘control knob’ which we can turn up or down to control the accuracy of the solution with little effort. The scalability is controlled by the choice of the number of neurons in the RBF networks. More neurons may be chosen for higher fidelity but slower simulation while fewer neurons may be chosen for coarser, but more rapid simulation. We have applied PhyNNeSS in a collaborative interactive setup taking advantage of this scalability (Sankaranarayanan, Dhanannjay Deo, & Suvranu De, 2009).

In this paper we have primarily focused on application of PhyNNeSS for simulation of vector fields, namely deformation and force fields. However, the method may be easily extended to simulation of additional degrees of freedom. For example, in the case of electrosurgery simulation, where (Lister & Desai, 2008; Maciel & Suvranu De, 2008) temperature changes are responsible for coagulation/ cauterization of soft tissues, the temperature field may be an additional variable that can be pre-computed and simulated using a RBFN.

In PhyNNeSS deformation or force at a single node is computed independently once the network has been trained. This makes PhyNNeSS ideal for implementing on single instruction multiple data (SIMD) architectures available on current nowadays graphical processing units. This makes PhyNNeSS suitable for CUDA (NVIDIA Corporation, n.d.) architecture, where shared memory is relatively larger (suitable for storing trained neural networks for complete mesh) and memory of each pipeline is limited, but it can easily accommodate intermediate data structures for execution of the trained neural network.

In this paper, we have focused attention to static problems. Extension of PhyNNeSS to dynamics would require a much larger set of data to be stored. Since PhyNNeSS utilizes precomputed data to train the RBFN, changes of topology, e.g., surgical cutting, may be a challenge. However, we anticipate developing techniques where local updates may be applied to a globally computed solution using PhyNNeSS to tackle such problems. In this paper we have discussed nonlinearities that may be introduced due to nonlinear material deformations and nonlinear constitutive equations. A third type of nonlinearity may be introduced due to contact conditions, when multiple organs are in contact and the contact areas change as a function of loading. This is a nontrivial problem and is left for future work.

Acknowledgments

The authors gratefully acknowledge the support of this work by NIH/NIBIB through grant # R01 EB005807.

References

Abe Y, Iiguni Y. Interpolation capability of the periodic radial basis function network. Iee Proceedings-Vision Image and Signal Processing. 2006; 153(6):785–794.

Ahn B, Kim J. Measurement and characterization of soft tissue behavior with surface deformation and force response under large deformations. Medical Image Analysis. 2010; 14(2):138–148.10.1016/j.media.2009.10.006 [PubMed: 19948423]

Allin, S.; Matsuoka, Y.; Klatzky, R. Measuring just noticeable differences for haptic force feedback: implications for rehabilitation. Haptic Interfaces for Virtual Environment and Teleoperator Systems, 2002 HAPTICS 2002 Proceedings 10th Symposium on; Presented at the Haptic Interfaces for Virtual Environment and Teleoperator Systems, 2002 HAPTICS 2002 Proceedings 10th Symposium on; 2002. p. 299-302.

Barbič J, James DL. Time-critical distributed contact for 6-DoF haptic rendering of adaptively sampled reduced deformable models. Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on computer animation, SCA'07. 2007:171–180.

Basdogan C, De S, Kim J, Muniyandi Manivannan, Kim H, Srinivasan M. Haptics in minimally invasive surgical simulation and training. Computer Graphics and Applications IEEE. 2004; 24(2): 56–64.

Bathe, K. Finite element procedures. Englewood Cliffs, NJ: Prentice Hall; 1996. Unabridged paperback reprint ed

Bianchi G, Solenthaler B, Székely G, Harders M. Simultaneous Topology and Stiffness Identification for Mass-Spring Models Based on FEM Reference Deformations. Medical Image Computing and Computer-Assisted Intervention-MICCAI (2, 2004). 2004:293–301.

Bickel B, Bächer M, Otaduy MA, Matusik W, Pfister H, Gross M. Capture and modeling of non-linear heterogeneous soft tissue. ACM Transactions on Graphics. 2009; 28(3):1–9.

de Boer A, van der Schoot MS, Bijl H. Mesh deformation based on radial basis function interpolation. Computers & Structures. 2007; 85(11-14):784–795.

Bonet, J.; Wood, RD. Nonlinear continuum mechanics for finite element analysis (Repr). Cambridge university: Cambridge Univ. Press; 1999.

Bro-Nielsen M. Finite element modeling in surgery simulation. Proceedings of the IEEE. 1998; 86(3): 490–503.

Carr, JC.; Beatson, RK.; Cherrie, JB.; Mitchell, TJ.; Fright, WR.; McCallum, BC.; Evans, TR. Proceedings of the 28th annual conference on Computer graphics and interactive techniques, SIGGRAPH'01. New York, NY, USA: ACM; 2001. Reconstruction and representation of 3D objects with radial basis functions; p. 67-76.

Chen S, Billings SA, Luo W. Orthogonal Least-Squares Methods and Their Application to Non-Linear System-Identification. International Journal of Control. 1989; 50(5):1873–1896.

De S, Lim Y, Manivannan M, Srinivasan MA. Physically realistic virtual surgery using the point-associated finite field (PAFF) approach. Presence: Teleoperators and Virtual Environments. 2006; 15(3):294–308.

Delp S, Loan P, Basdogan C. Surgical Simulation: An Emerging Technology for Training in Emergency Medicine. Presence: Teleoperators and Virtual Environments. 1997; 6(2):147–159.

Deo, D.; De, S. PhyNeSS: A Physics-driven Neural Networks-based Surgery Simulation system with force feedback. proceedings of the third joint EuroHaptics conference and the Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems; World Haptics. 2009. p. 30-34.

Edwards S, Tabor D. The Theory of Rubber Elasticity [and Discussion]. Proceedings of the Royal Society of London Series A Mathematical and Physical Sciences. 1976; 351(1666):397–406.

Grzeszczuk, R.; Terzopoulos, D.; Hinton, G. NeuroAnimator: fast neural network emulation and control of physics-based models. Proceedings of the 25th annual conference on Computer graphics and interactive techniques (SIGGRAPH '98); ACM, New York, NY, USA. 1998.

Halic T, Arikatla VS, Sankaranarayanan G, Lu Z, Ahn W, De S. A Software Framework for Multimodal Interactive Simulations (SoFMIS). Studies in Health Technology and Informatics. 2011; 163:213–217. [PubMed: 21335791]

Hambli R, Chamekh A, Bel Hadj Salah H. Real-time deformation of structure using finite element and neural networks in virtual reality applications. Finite Elements in Analysis and Design. 2006; 42(11):985–991.

Harders M, Bajka M, Spaelter U, Tuchschmid S, Bleuler H, Szekely G. Highly-realistic, immersive training environment for hysteroscopy. Studies in Health Technology and Informatics. 2006; 119:176–81. [PubMed: 16404040]

Ho C, Basdogan C, Srinivasan MA. Efficient Point-Based Rendering Techniques for Haptic Display of Virtual Objects. Presence: Teleoperators and Virtual Environments. 1999; 8(5):477–491.

Höver R, Kósa G, Székely G, Harders M. Data-Driven Haptic Rendering—From Viscous Fluids to Visco-Elastic Solids. IEEE Transactions on Haptics. 2009; 2:15–27.10.1109/TOH.2009.2

Jung S, Ghaboussi J. Neural network constitutive model for rate-dependent materials. Computers & Structures. 2006; 84(15-16):955–963.

Kallassy A. A new neural network for response estimation. Computers & Structures. 2003; 81(26-27): 2417–2429.

Kauer, M.; Vuskovic, V.; Dual, J.; Szekely, G.; Bajka, M. Inverse Finite Element Characterization of Soft Tissues. In: Niessen, W.; Viergever, M., editors. Medical Image Computing and Computer-Assisted Intervention – MICCAI 2001, Lecture Notes in Computer Science. Vol. 2208. Springer Berlin; Heidelberg: 2001. p. 128-136.

Kuehnapfel UG, Neisius B. CAD-based graphical computer simulation in endoscopic surgery. Endoscopic Surgery and Allied Technologies. 1993; 1(3):181–4. [PubMed: 8055320]

Kuroe Y, Kawakami H. Vector Field Approximation by Model Inclusive Learning of Neural Networks. In Artificial Neural Networks – ICANN 2007. 2007:717–726.

Lake GJ, Thomas AG. The Strength of Highly Elastic Materials. Proceedings of the Royal Society of London Series A, Mathematical and Physical Sciences. 1967; 300(1460):108–119.

Lim Y, De S. Real time simulation of nonlinear tissue response in virtual surgery using the point collocation-based method of finite spheres. Computer Methods in Applied Mechanics and Engineering. 2007; 196(31-32):3011–3024.

Lister K, Desai JP. Soft-tissue characterization during monopolar electrocautery procedures. Studies in Health Technology and Informatics. 2008; 132:254–256. [PubMed: 18391298]

Liu Y, De S. CUDA-based real time surgery simulation. Studies in Health Technology and Informatics. 2008; 132:260–2. [PubMed: 18391300]

Loop, C. Smooth subdivision surfaces based on triangles (Master's thesis). University of Utah:

Maciel A, De S. Physics-based real time laparoscopic electrosurgery simulation. Studies in Health Technology and Informatics. 2008; 132:272–274. [PubMed: 18391303]

Miller K. Method of testing very soft biological tissues in compression. Journal of Biomechanics. 2005; 38(1):153–158.10.1016/j.jbiomech.2004.03.004 [PubMed: 15519351]

Mullur AA, Messac A. Metamodeling using extended radial basis functions: a comparative approach. Engineering with Computers. 2006; 21(3):203–217.

Nava A, Mazza E, Furrer M, Villiger P, Reinhart W. In vivo mechanical characterization of human liver. Medical Image Analysis. 2008; 12(2):203–216. [PubMed: 18171633]

NVIDIA_Corporation. (n.d.). CUDA Zone. NVIDIA Corporation; 2009.

Omar T, Eskandarian A, Bedewi N. Vehicle crash modelling using recurrent neural networks. Mathematical and Computer Modelling. 1998; 28(9):31–42.

Orr MJL. Regularization in the Selection of Radial Basis Function Centers. Neural Computation. 1995; 7(3):606–623.

Pai, DK.; Doel, KVD.; James, DL.; Lang, J.; Lloyd, JE.; Richmond, JL.; Yau, SH. Proceedings of the 28th annual conference on Computer graphics and interactive techniques. ACM; New York, NY, USA: 2001. Scanning physical interaction behavior of 3D objects; p. 87-96.SIGGRAPH '01

Park J, Sandberg IW. Universal approximation using radial-basis-function networks. Neural Computation. 1991; 3(2):246–257.

Park J, Sandberg IW. Approximation and Radial-Basis-Function Networks. Neural Computation. 1993; 5(2):305–316.

Peterlik, I.; Matyska, L. proceedings of the second joint EuroHaptics conference and the Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems. Vol. 2007. World Haptics; 2007. An Algorithm of State-Space Precomputation Allowing Non-linear Haptic Deformation Modelling Using Finite Element Method; p. 231-236.

Picinbono G, Delingette H, Ayache N. Non-linear anisotropic elasticity for real-time surgery simulation. Graphical Models. 2003; 65(5):305–321.

Pongrac, H.; Färber, B.; Hinterseer, P.; Kammerl, J.; Steinbach, E. In Human-Centered Robotics Systems 2006. Munich, Germany: 2006. Limitations of Human 3D Force Discrimination.

Radetzky A, Nürnberger A, Pretschner DP. Simulation of elastic tissues in virtual medicine using neuro-fuzzy systems. Medical imaging: Image Display Proceedings of the International Society for Optics and Photonics (SPIE). 1998; 3335:399–409.

Samarasinghe, S. Neural Networks for Applied Sciences and Engineering. Boston, MA, USA: Auerbach Publications; 2006.

Sankaranarayanan G, Deo D, De S. Hybrid network architecture for interactive multi-user surgical simulator with scalable deformable models. Studies in Health Technology and Informatics. 2009; 142:292–294. [PubMed: 19377171]

Saupin G, Duriez C, Cotin S. Contact Model for Haptic Medical Simulations. In Biomedical Simulation. 2008:157–165.

Sherstinsky A, Picard RW. On the efficiency of the orthogonal least squares training method for radial basis function networks. Ieee Transactions on Neural Networks. 1996; 7(1):195–200. [PubMed: 18255570]

Székely G, Brechbühler C, Hutter R, Rhomberg A, Ironmonger N, Schmid P. Modelling of soft tissue deformation for laparoscopic surgery simulation. Medical Image Analysis. 2000; 4(1):57–66.10.1016/S1361-8415(00)00002-5 [PubMed: 10972321]

Tan H, Radcliffe J, Ga BN, Tan HZ, Eberman B, Srinivasan MA, Cheng B. Human Factors For The Design Of Force-Reflecting Haptic Interfaces. proceedings of the American Society of Mechanical Engineers (ASME) Dynamic Systems and Control. 1994; 55(1):353–359.

Valdes JL, Biscay R, Jimenez JC. Geometric selection of centers for radial basis function approximations involved in intensive computer simulations. Mathematics and Computers in Simulation. 1999; 48(3):295–306.

Waszczyszyn Z, Ziemianski L. Neural networks in mechanics of structures and materials - new results and prospects of applications. Computers & Structures. 2001; 79(22-25):2261–2276.

Weiguo, S.; Kui, Y. Haptic modeling for liver cutting based on fuzzy neural network. Presented at the Advanced Intelligent Mechatronics. Proceedings, 2005 IEEE/ASME International Conference on; 2005. p. 1216-1220.

Yu DL, Gomm JB, Williams D. A Recursive Orthogonal Least Squares Algorithm for Training RBF Networks. Neural Processing Letters. 1997; 5(3):167–176.

Zhong YM, Shirinzadeh B, Alici G, Smith J. A cellular neural network methodology for deformable object simulation. Ieee Transactions on Information Technology in Biomedicine. 2006; 10(4):749–762. [PubMed: 17044409]

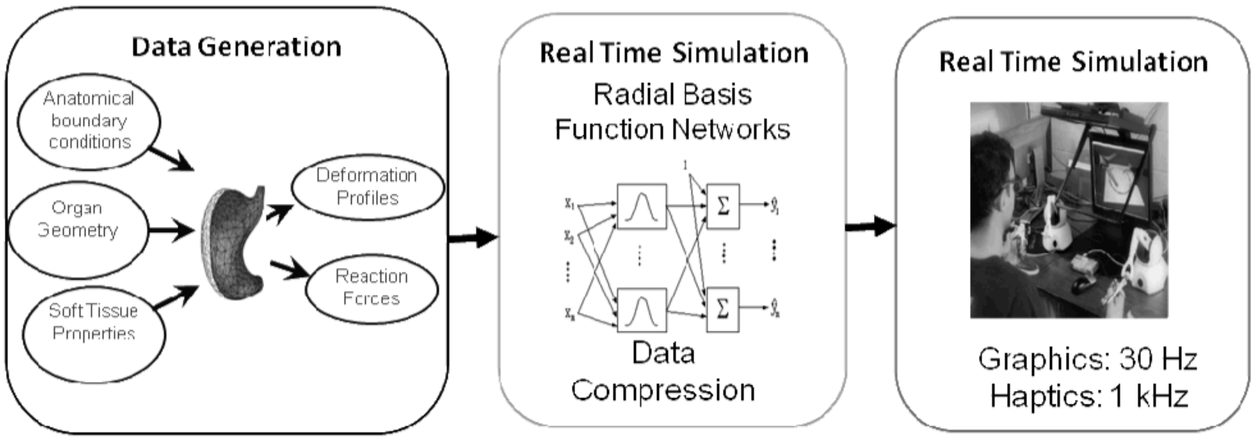


Figure 1. Schematic Diagram of PhyNNeSS

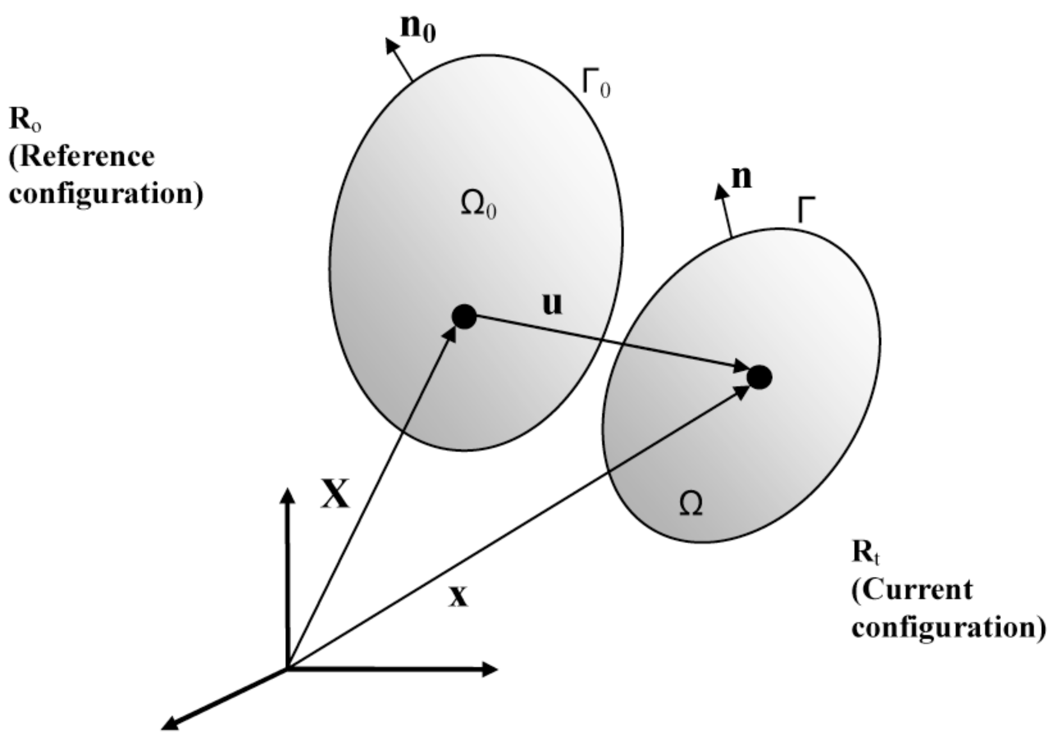


Figure 2. Deforming continuum

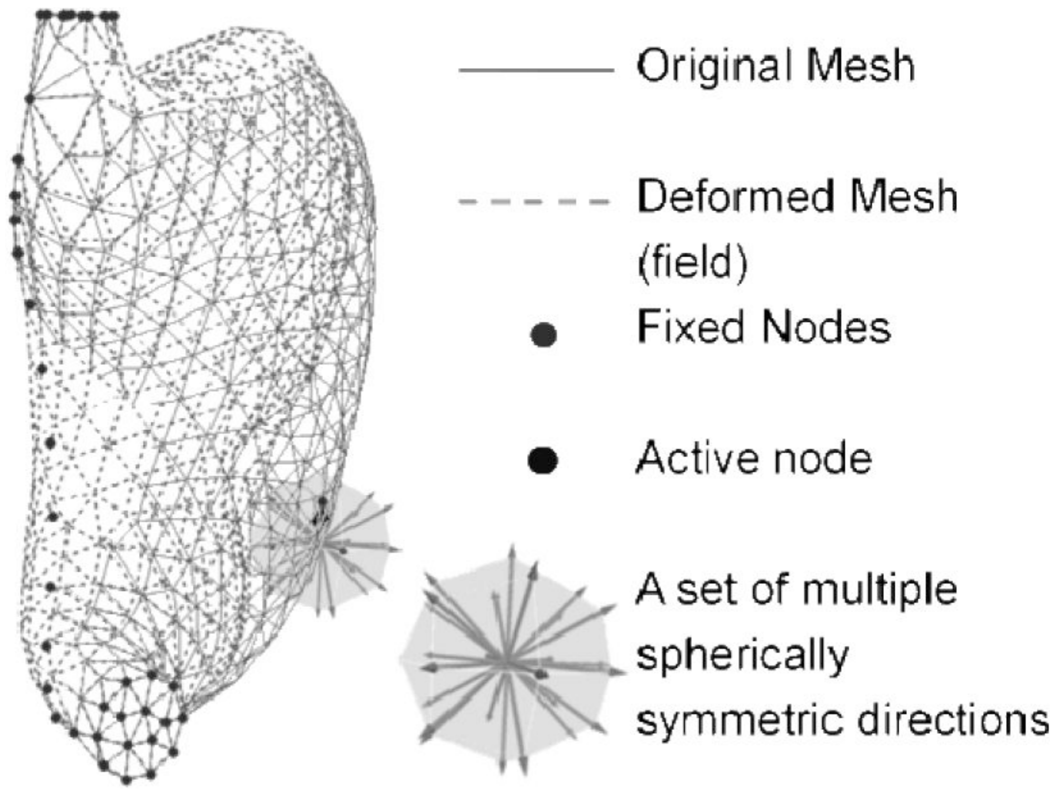


Figure 3. The pre-computation step - A finite element model with the active node being displaced along one of 26 predetermined directions

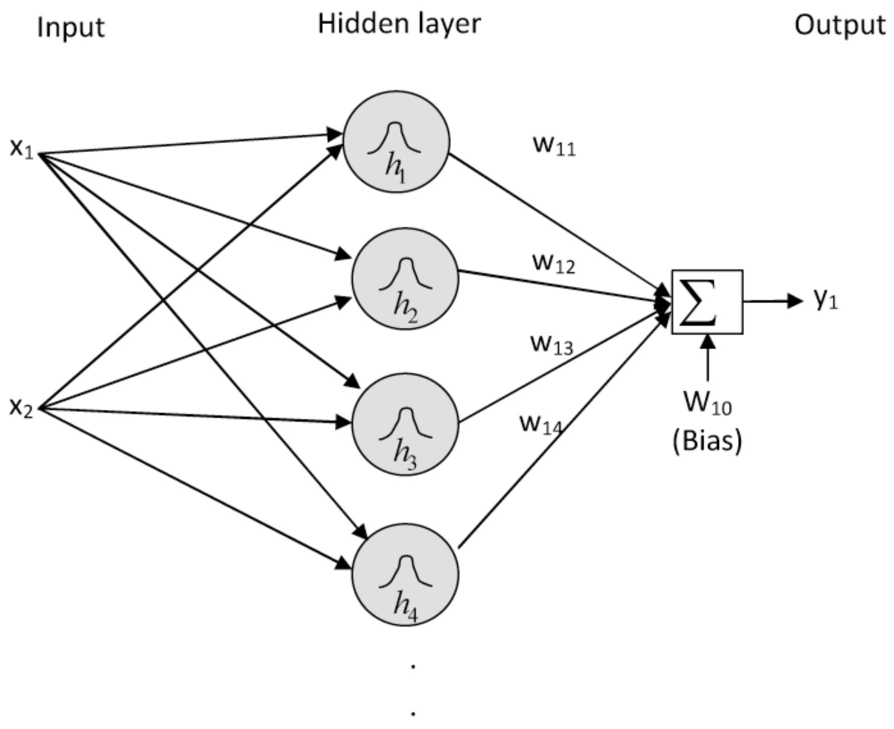


Figure 4.
Basic topology of a radial basis function neural network (RBFN) with two inputs and a single output.

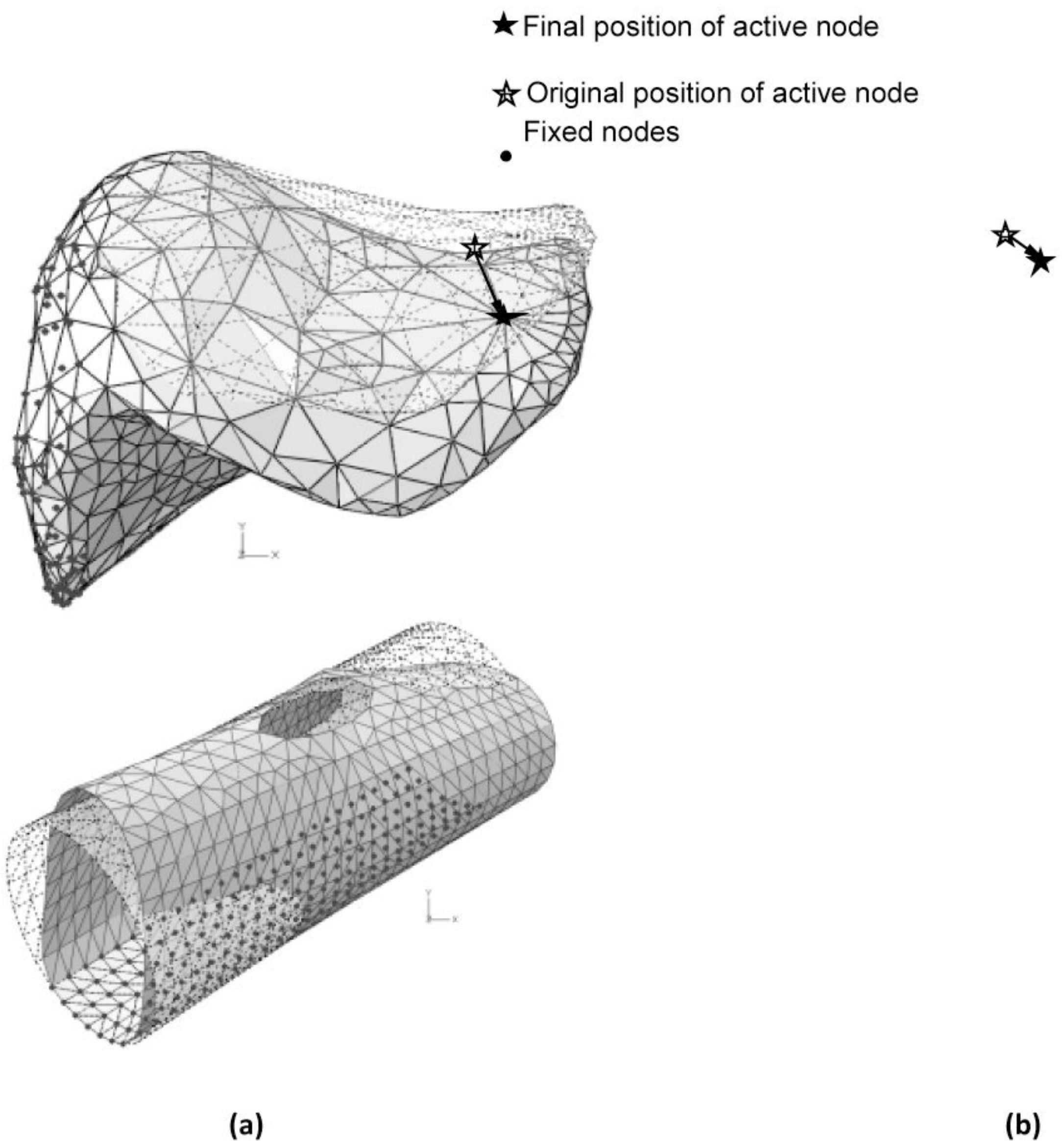

Final position of active node
Original position of active node
Fixed nodes
(a)
(b)

Figure 5. The finite element mesh and deformed configurations computed using finite elements of (a) Liver model and (b) Penrose drain model

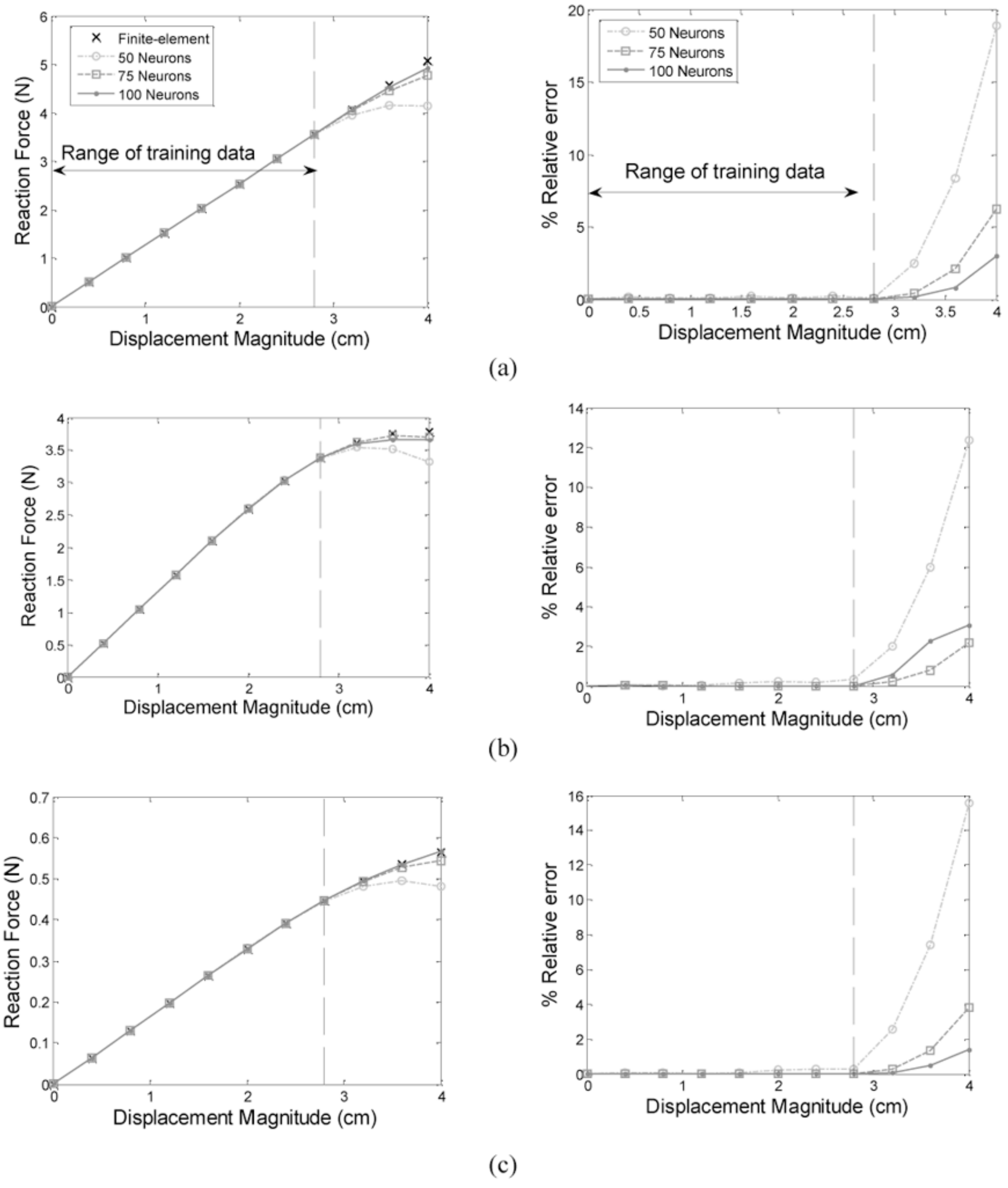
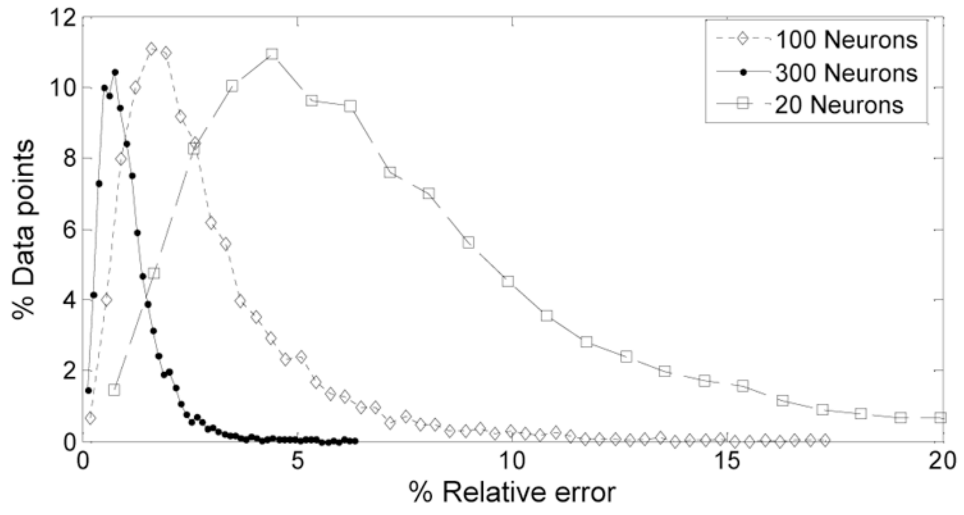
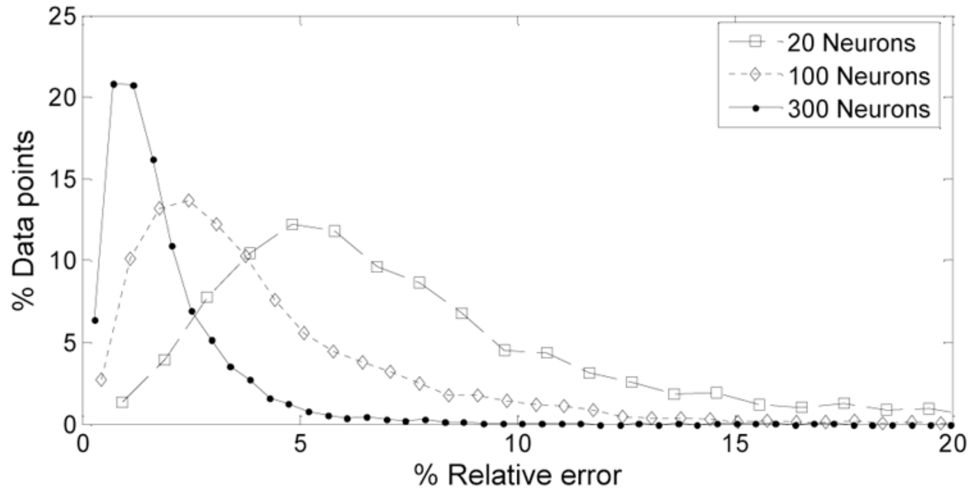


Figure 6.

Force-displacement relationships and their approximation using PhyNNeSS using 50, 75 and 100 neurons in the network for (a) Case 1, (b) Case 2 and (c) Case 3.



(a)



(b)

Figure 7.
Histograms of displacement error distribution for different neural networks for (a) Case 3 and (b) Case 4.

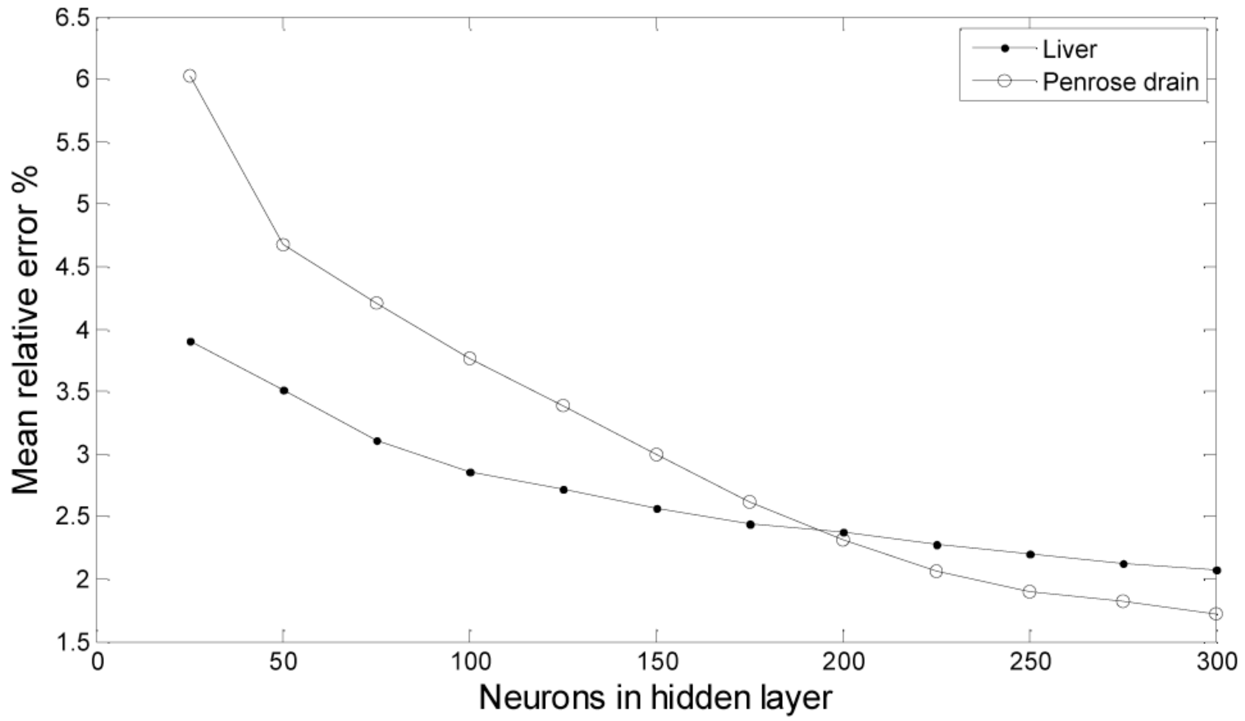


Figure 8. Plot of the % mean relative error as a function of the number of neurons in the hidden layer of the deformation radial basis function neural network (RBFN)

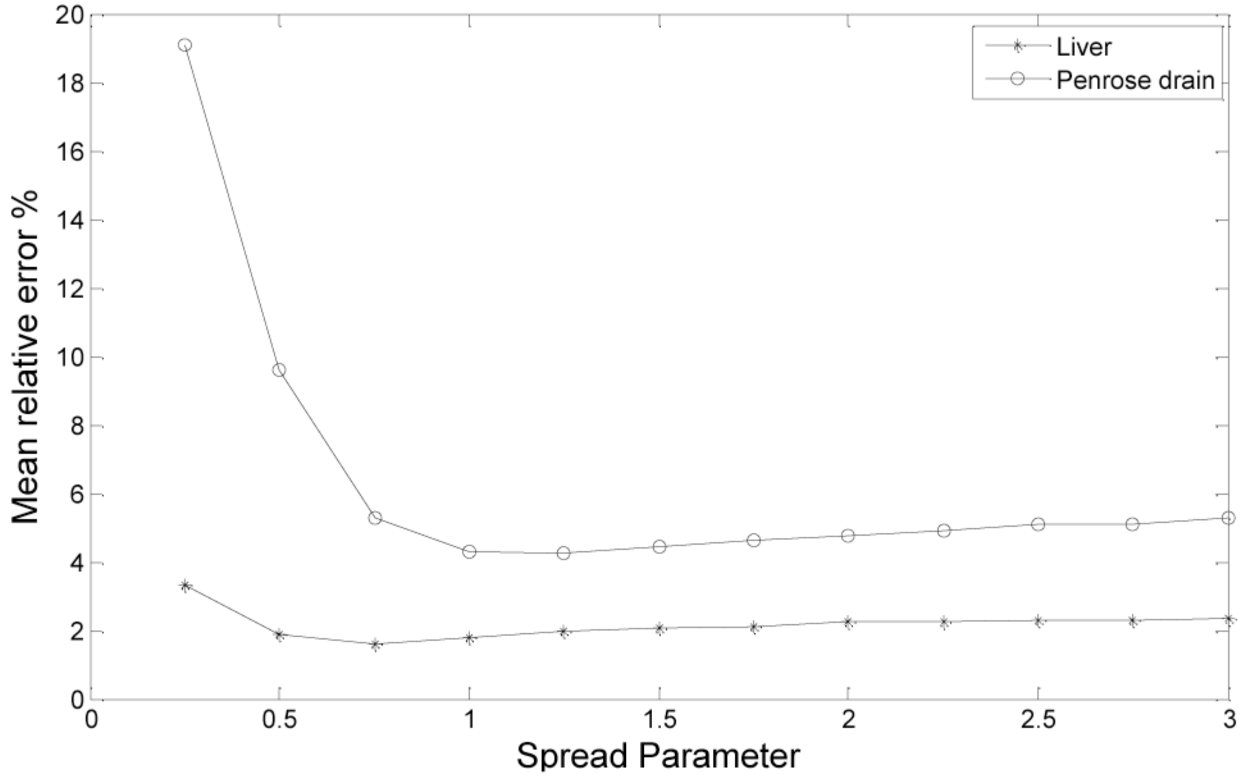


Figure 9. Plot of % mean relative error as a function of spread parameter used in training of neural networks for liver and Penrose drain models

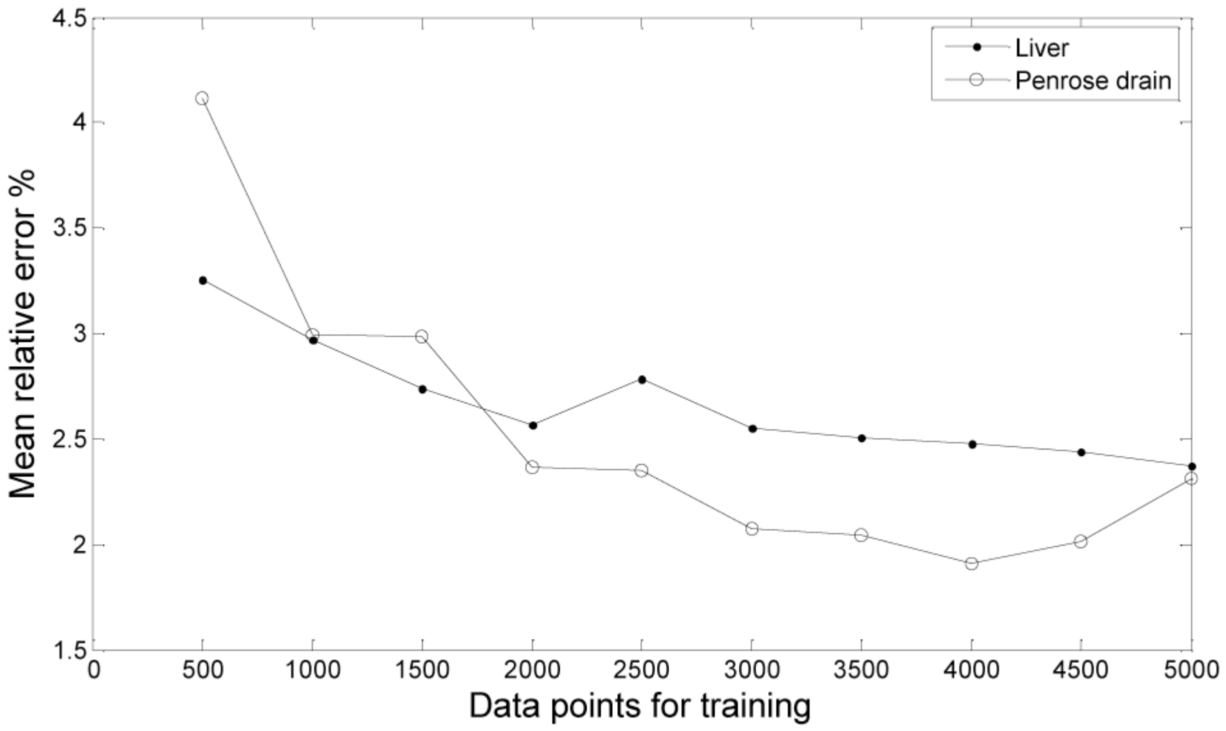


Figure 10. Plot of the % mean relative error with increase in number of data points used in training of the deformation RBFN

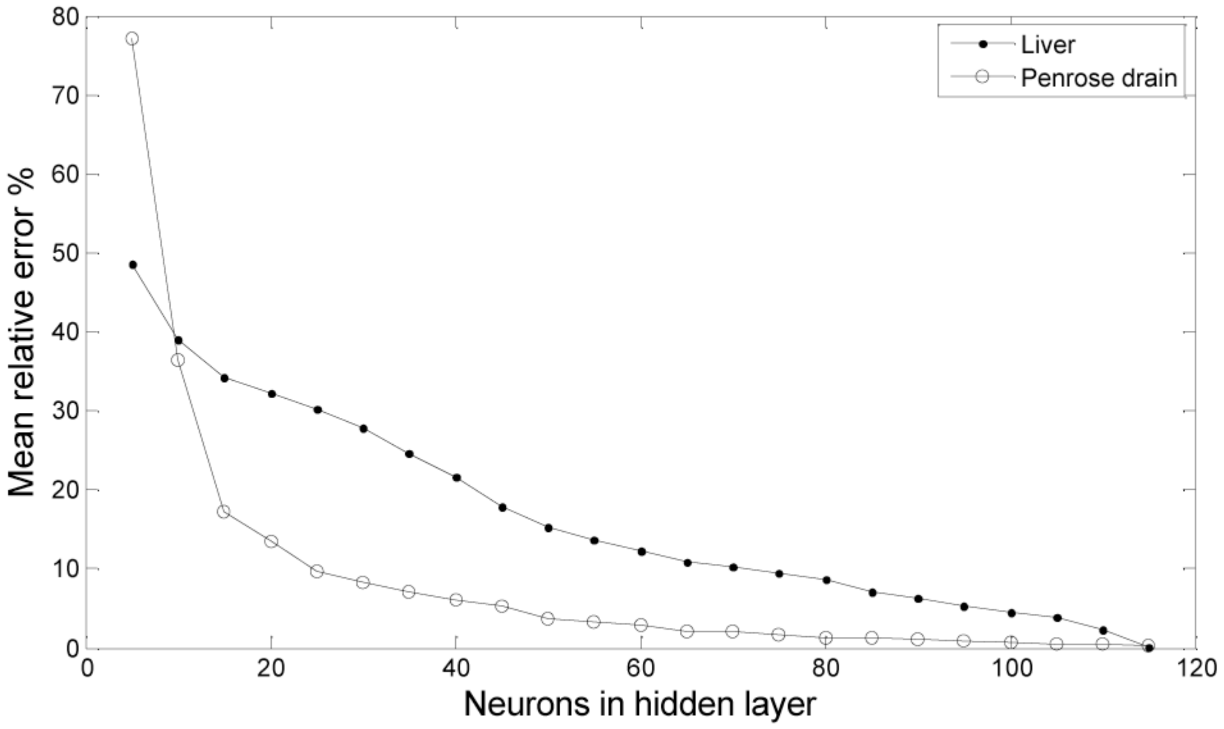


Figure 11. Plot of % mean relative error with increase in number neurons in hidden layer of force RBFN

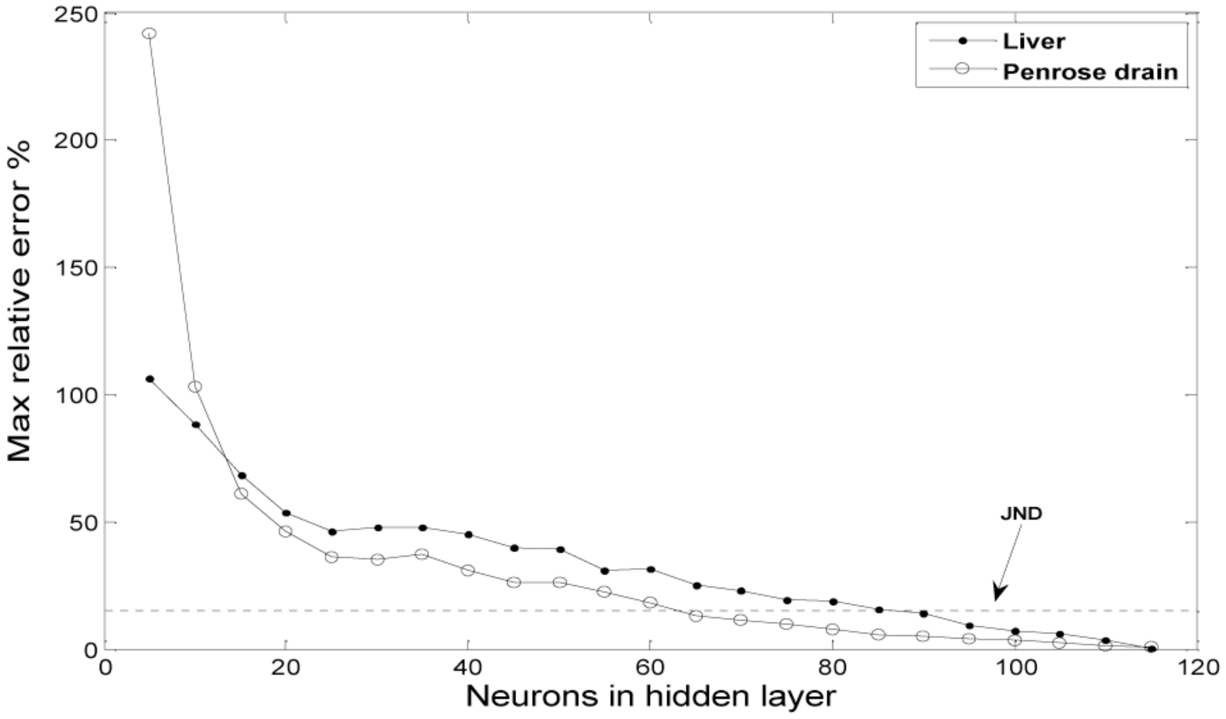


Figure 12. Plot of % Max relative error with increase in number of neurons in hidden layer of RBFN

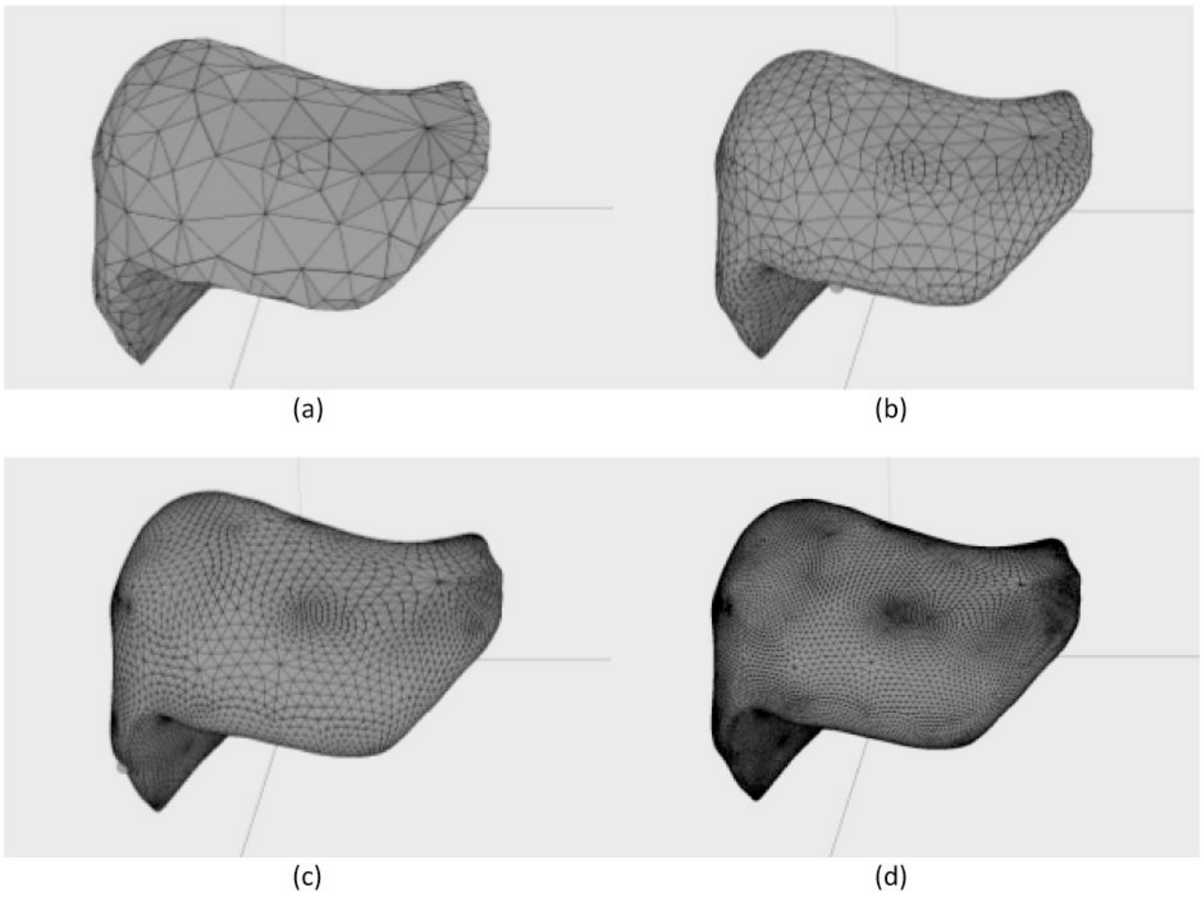


Figure 13.
(a) Original coarse mesh (subdivision level 0) subdivided to increasing levels of refinement in (b), (c) and (d).

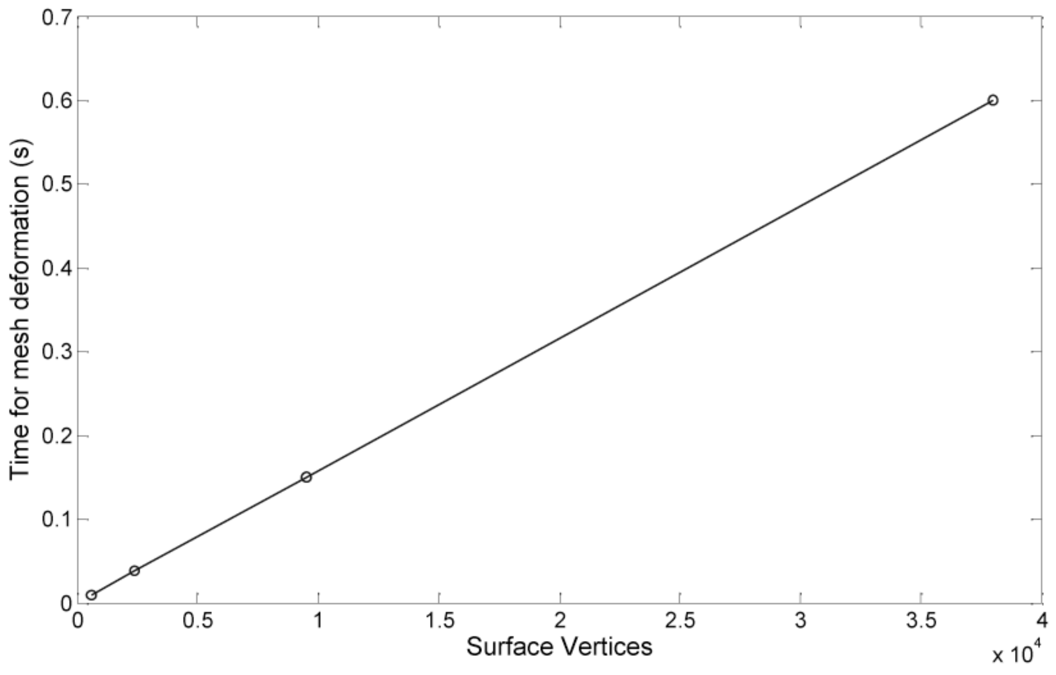


Figure 14. Performance of PhyNNeSS scales linearly with the number of mesh vertices

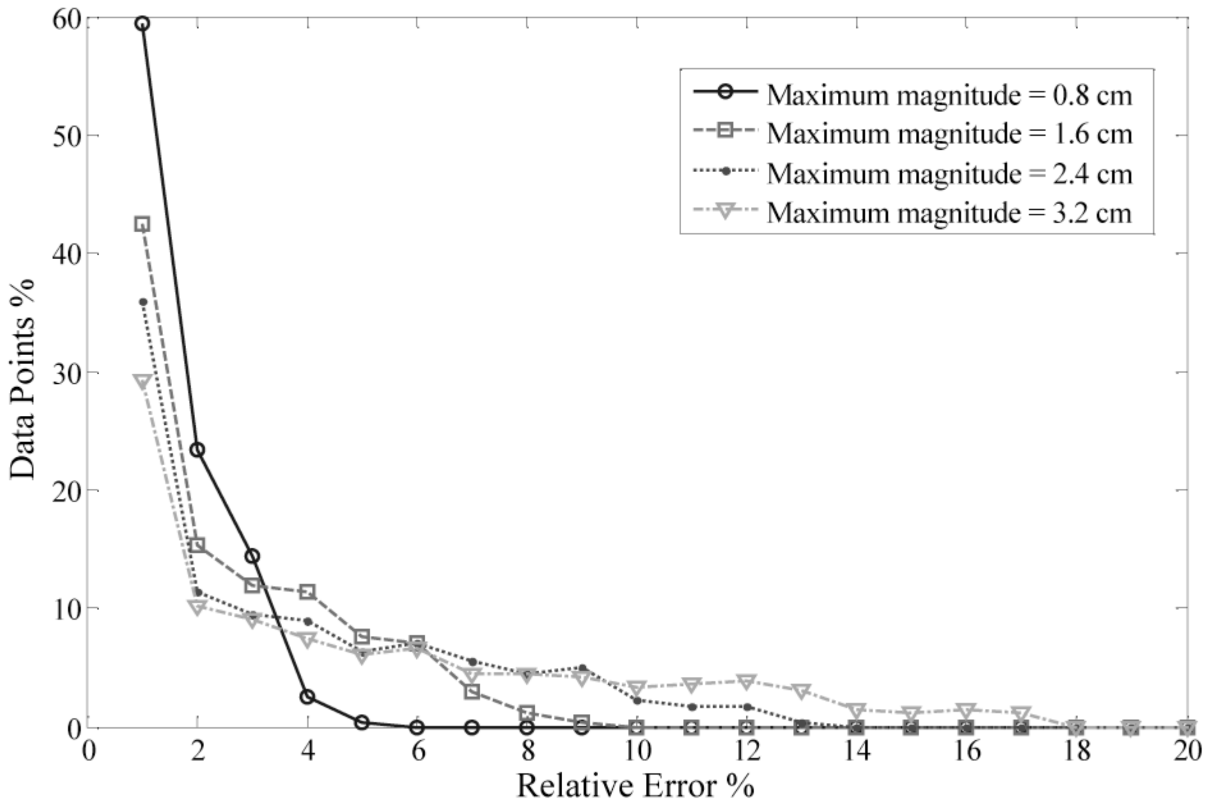


Figure 15. Histogram of percent relative errors for different magnitudes of displacement

Table 1

Model data for the example problems

Parameter	**Liver model**	**Penrose drain model**
Number of nodes (N)	739	239
Number of free nodes (N^{free})	486	181
Type of element	3D tetrahedra	Triangular shell elements
Number of elements	2341	436
Number of precomputation cases solved	12636	4706
CPU Time required for automated script	15 s	8 s
Total data points stored	75660	23520
Number of neurons in network	200	100
Total number of training samples	5000	5000
Average CPU time for training network, including data extraction	280 s	170 s
Total pre-computation time, including data generation and training	87 hrs	18 hrs
Size of data recorded from FEM	8Gb	6Gb
Size of data extracted for training networks	501 Mb	170 Mb
Size of neural networks	5.4Mb	2.02 Mb
Time for deformation computation in real time (average time for computation of deformation at N^{free} nodes when one node is displaced)	6.5×10^{-3} s(154 fps)	2.9×10^{-3} s (345 fps)
Time for force computation in real time (average time to compute the reaction force at the node which is displaced)	1.5×10^{-6}	1.5×10^{-6}

Table 2

Mesh details for successive subdivision

Subdivision Level	Index of selected vertex	Vertices	Triangles	Simulation time
0	102	596	1188	0.00936
1	502	2378	4752	0.03748
2	1497	9506	19008	0.15000
3	4699	38018	76032	0.59940