Static and Dynamic Modelling of Materials Forging

Coryn A.L. Bailer-Jones, David J.C. MacKay
Cavendish Laboratory, University of Cambridge
email: calj@mrao.cam.ac.uk; mackay@mrao.cam.ac.uk

Tanya J. Sabin and Philip J. Withers
Department of Materials Science and Metallurgy, University of Cambridge
email: tjs1005@cus.cam.ac.uk; pjw12@cus.cam.ac.uk

ABSTRACT

The ability to model the thermomechanical processing of materials is an increasingly important requirement in many areas of engineering. This is particularly true in the aerospace industry where high material and process costs demand models that can reliably predict the microstructures of forged materials. We analyse two types of forging, cold forging in which the microstructure develops statically upon annealing, and hot forging for which it develops dynamically, and present two different models for predicting the resultant material microstructure. For the cold forging problem we employ the Gaussian process model. This probabilistic model can be seen as a generalisation of feedforward neural networks with equally powerful interpolation capabilities. However, as it lacks weights and hidden layers, it avoids ad hoc decisions regarding how complex a 'network' needs to be. Results are presented which demonstrate the excellent generalisation capabilities of this model. For the hot forging problem we have developed a type of recurrent neural network architecture which makes predictions of the time derivatives of state variables. This approach allows us to simultaneously model multiple time series operating on different time scales and sampled at non-constant rates. This architecture is very general and likely to be capable of modelling a wide class of dynamic systems and processes.

Australian Journal on Intelligent Information Processing Systems, 5(1), 10–16, 1998

1. Introduction

The problem in the modelling of materials forging can be broadly stated as follows: Given a certain material which undergoes a specified forging process, what are the final properties of this material? Typical final properties in which we are interested are the microstructural properties, such as the mean grain size and shape and the extent of grain recrystallisation. Relevant forging process control variables are the strain, strain rate and temperature, all of which may be functions of time.

A trial-and-error approach to solving this problem has often been taken in the materials industry, with many different forging conditions attempted to achieve a given final product. The obvious drawbacks of this approach are large time and financial costs and the lack of any reliable predictive capability. Another method is to develop a parameterised, physically-motivated model, and to solve for the parameters using empirical data [2]. However, the limitation with this approach is that in terms of the physical theory the microstructural evolution depends upon several "intermediate" microscopic variables which have to be measured in order to apply the model. Some of these variables, such as dislocation density, are difficult and time-consuming to measure, making it impracticable to apply such an approach to large-scale industrial processes.

Our approach to the prediction of forged microstructures is therefore to develop an empirical model in which we define a parameterised, non-linear relationship between the microstructural variables of interest and those easily measured process variables. Such a model could be implemented, for example, as a neural network with the hidden nodes essentially playing a role analogous to the "intermediate" microscopic variables.

2. Materials Forging

When a material is deformed, potential energy is put into the system by virtue of work having been done to move crystal planes relative to one another. The material is therefore not in equilibrium and has a tendency to lower its potential energy by atomic rearrangement, through the competing processes of recovery, recrystallisation and grain growth. These processes are encouraged by raising the temperature of the material (annealing). Forge deformation processes can be divided into two classes. In *cold working* the recrystallisation rate is so low that recrystallisation essentially does not occur during forging. Recrystallisation is subsequently

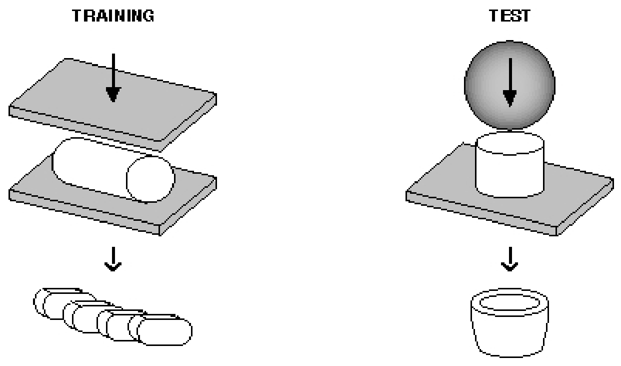

Fig. 1: Deformation geometries. (a) (left) Plane-strain diametrical compression. The workpiece is subsequently sectioned into many nominally identical specimens which are annealed at different combinations of temperature and time. The compression gives rise to a non-linear distribution of strains across the specimen (see Figure 2b). These allow us to obtain many input training vectors, $\mathbf{x}$, for our model using a single compression test. (b) (right) Axisymmetric axial compression.

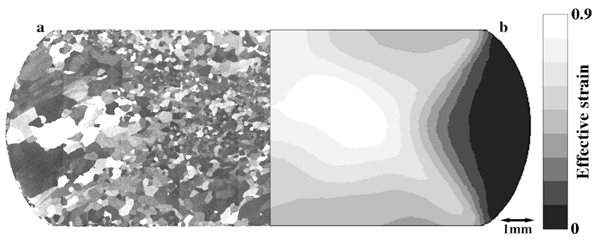

Fig. 2: (a) The left half of this diagram shows the microstructure of half of a sectioned specimen which has been deformed under a plane-strain compression. The material has been annealed at $350°C$ for 30 mins producing many recrystallised grains. (b) The right half of this diagram is the corresponding strain contour map produced by the Finite Element model. Note that the areas of high strain in (b) correspond to small grains in (a).

achieved *statically* by annealing. In contrast, *hot working* refers to the high temperature forging of materials in which recrystallisation occurs *dynamically* during forging. This process is considerably more complex than cold working as now the final microstructure of the material is generally a path-dependent function of the history of the process variables. This is particularly true of the Aluminium–Magnesium alloy considered here, which have a relatively long 'memory' of the process, thus necessitating a model which keeps track of the history of the material. We shall consider a model for this dynamic process in Section 4.

The ultimate goal of forge modelling is the inverse problem: Given a set of desired final properties for a component, what is the optimal material and forging process which will realise these properties? This is a considerably harder problem since there may be a one-to-many mapping between the desired properties and the necessary forging process. This problem will not be addressed in this paper.

3. Static Modelling

Cold forging can in general be modelled with the equation

$$v = \mathcal{F}(\mathbf{x}) \tag{1}$$

where v is a microstructural variable, $\mathbf{x}$ is the set of process variables and $\mathcal{F}$ is some non-linear function. In our particular implementation we are interested in predicting a single microstructural variable, namely grain size, in a given material (an Al-1%Mg alloy) as a function of the total strain, ε, annealing temperature, T, and annealing time, t. The experimental set-up for obtaining these data is as follows. A workpiece of the material is compressed in plane-strain compression at room temperature, as shown in Figure 1a. After the specimen has been annealed, it is etched and the grain sizes measured with an optical microscope. The local strain experienced at each point in the material is evaluated using a Finite Element (FE) model, the parameters of this model being determined by the known material properties, forging geometries, friction factors and so on. Figure 2b shows an example of an FE map. Many grain sizes within a single small area are averaged to give a mean grain size. Thus we now have a set of model inputs, ε, T and t, associated with a single mean grain size which can be used to develop a static microstructural model of forging. Further details of the experimental procedure can be found in Sabin et al. [9].

3.1. The Gaussian Process Model

The Gaussian process model [4] [10] assumes that the prior joint probability distribution of a set of any N observations is given by an N-dimensional Gaussian, i.e.

$$P(\mathbf{v}_N|\{\mathbf{x}_N\}, \boldsymbol{\mu}, \mathbf{C}_N) \tag{2}$$

$$\propto \exp\left(-\frac{1}{2}(\mathbf{v}_N - \boldsymbol{\mu})^T \mathbf{C}_N^{-1} (\mathbf{v}_N - \boldsymbol{\mu})\right) \quad , \tag{3}$$

where $\mathbf{v}_N = (v_1(\mathbf{x}_1), v_2(\mathbf{x}_2), \ldots, v_N(\mathbf{x}_N))$ is the set N of observations corresponding to the set of N input vectors, $\{\mathbf{x}_N\} \equiv \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$. $\boldsymbol{\mu}$ and $\mathbf{C}_N$, respectively the mean and covariance matrix for the distribution, parameterise this model. The elements of the covariance matrix are specified by the covariance function, which is a function of the input vectors, $\{\mathbf{x}_N\}$, and a set of *hyperparameters*. A typical form of the covariance function is

$$C_{ij} = \theta_1 \exp\left[-\frac{1}{2}\sum_{l=1}^{l=L} \frac{(x_i^{(l)} - x_j^{(l)})^2}{r_l^2}\right] + \theta_2 + \theta_3 \delta_{ij} \quad . \tag{4}$$

This equation gives the covariance between any two values v_i and v_j with corresponding L-dimensional input vectors $\mathbf{x}_i$ an $\mathbf{x}_j$ respectively, and is capable of implementing a wide class of functions, $\mathcal{F}$, that could appear in equation 1. (The Gaussian process model has a scalar 'output', v; to model several microstructural variables we would use several independent models.) The first term in equation 4 expresses our belief that the function we are modelling is smoothly varying, where r_l is the length scale over which the function varies in the l^{th} input dimension. The second term allows the functions to have a constant offset and the third is a noise term: this particular form is a model for input independent Gaussian noise. The hyperparameters, r_l ($l = 1 \ldots L$), θ_1, θ_2, θ_3, specify the function, and are generally inferred from a set of training data in a fashion analogous to training a neural network. They are called *hyperparameters* rather than *parameters* because they explicitly parameterise a probability distribution rather than the function itself. This distinguishes them from weights in a neural network, which are rather "arbitrary", in that adding another hidden node could change the weights yet leave the input–output mapping essentially unaltered.

Once the hyperparameters are known, the probability distribution of a new predicted value, v_{N+1}, corresponding to a new 'input' variable, $\mathbf{x}_{N+1}$, is

$$P(v_{N+1}|\mathbf{v}_N, \{\mathbf{x}_N\}, \mathbf{x}_{N+1}, \mathbf{C}_{N+1}) \tag{5}$$

$$\propto \exp\left(-\frac{(v_{N+1} - \hat{v}_{N+1})^2}{2\sigma^2_{\hat{v}_{N+1}}}\right) , \tag{6}$$

i.e. a one-dimensional Gaussian, where $\hat{v}_{N+1}$ and $\sigma_{\hat{v}_{N+1}}$ are evaluated in terms of the covariance function and the training data. We would typically report our prediction as $\hat{v}_{N+1} \pm \sigma_{\hat{v}_{N+1}}$. These errors reflect both the noise in the data (third term in equation 4) and the model uncertainty in interpolating the training data. The fact that the Gaussian process model naturally produces confidence intervals on its predictions is important in the materials industry where material properties must often be specified within certain tolerances.

Our model assumes that the measurement noise and the prior probability of the unknown function can be described by a Gaussian distribution. In our application it is more sensible to assume that it is the *logarithm* of grain sizes which are distributed as a Gaussian, rather than the grain sizes themselves. This is because uncertainties in measuring grain size scale with the mean grain size, and are therefore more appropriately expressed as a fraction of the mean grain size rather than a fixed absolute grain size. Moreover, empirical evidence suggests that grain size distributions are well described by a log normal distribution [7].

3.2. Model Predictions

A Gaussian process model was trained using a set of 46 data pairs obtained from the plane-strain geometry, with $0.08 < \varepsilon < 0.79$, $325°\text{C} < T < 375°$ C, 1 mins $< t <$ 60 mins as the inputs. Once trained, the model was used to produce predictions of grain sizes for a range of the input variables. These predictions, shown in Figure 3, agree well with metallurgical expectations.

One of the assumptions implicit in our model of cold forging is that given the local strain conditions, the microstructure is independent of the material shape and forge geometry. In other words, we assume that predictions can be obtained given only the local accumulated strain (and annealing conditions). This is an important requirement as it means that a single model could be applied to a range of industrial forging geometries, provided that the local strains could be obtained (e.g. with an FE model). We tested the validity of this assumption by using the Gaussian process model trained on plane-strain data to predict grain sizes in a material compressed using a different geometry, namely an axial compression (Figure 1b). As before, after compression the material was annealed, sectioned and grain sizes measured. A new FE model gave the concomitant local strains. These process inputs were then used to obtain predictions of the grain sizes using the previous Gaussian process model. Figure 4 plots these predictions against the measurements. We see remarkable agreement—well within the predicted errors—thus validating our modelling approach. A practical application of our model is to produce diagrams such as that shown in Figure 5, a map of the grain sizes. Such a map is important for engineers who need to know the grain sizes at different points in the material, and can thus assess its resistance to phenomena such as creep and fatigue.

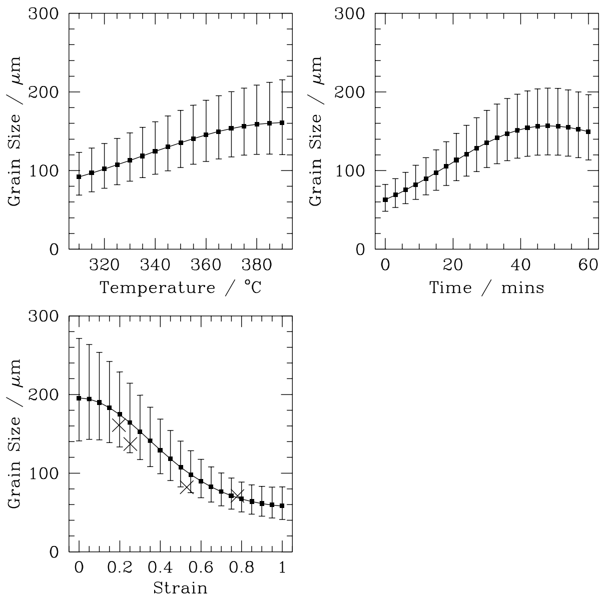

Fig. 3: Grain size predictions obtained with the Gaussian process model trained on data from the plane-strain compression geometry. In each of the three plots, two of the input variables are held constant and the other varied. When not being varied, the inputs were held constant at: $T = 350°$C; $t = 30$ mins; $\varepsilon = 0.5$. The crosses in the strain plot are data from the training set. As the Gaussian process is an interpolation model, predictions at any values of the inputs are constrained by the entire training set.

It should be noted that this method contains other implicit assumptions. The final material microstructure is very strongly dependent upon the material composition. It is well known that even small changes in the fractions of the alloying constituents (and by extension, impurities) can have a strong effect on the thermome-

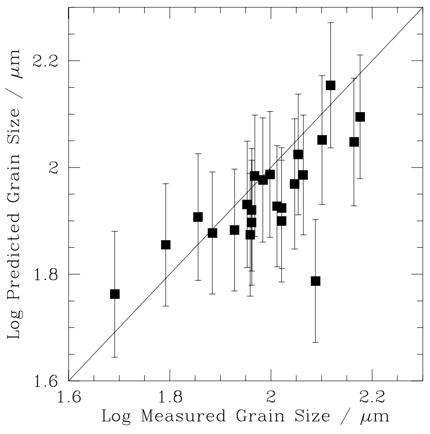

Fig. 4: Gaussian process model predictions compared with measured values. The Gaussian process model was trained on data from one compressional geometry (plane-strain) and its performance evaluated using data from another geometry (axial-compression) which was not seen during training. The $y = x$ line is to guide the eye. Note that not even a perfect model would produce predictions on this line due to finite noise in the data.

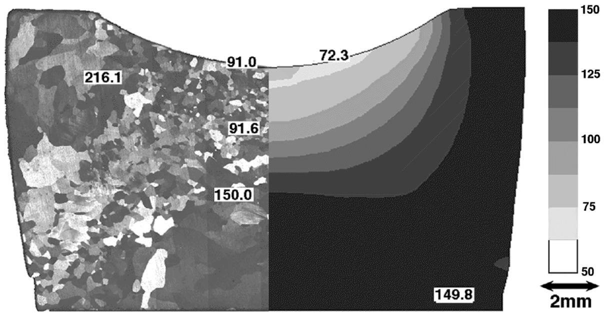

Fig. 5: The left half is an image of the microstructure in the axially compressed specimen. The right half is the corresponding grain size predictions from the Gaussian process model shown as a contour map.

chanical processing of the material. One way forward is to include further input variables corresponding to composition [3]. A second implicit assumption has been the constancy of the initial microstructure. Depending upon the material and the degree of thermomechanical processing, the final microstructure may retain some 'memory' of its initial microstructure, thus necessitating a model which has "initial conditions" as additional input variables.

4. A Recurrent Neural Network for Dynamic Process Modelling

4.1. Model Description

For the hot working problem, we assume that there are two sets of variables which are relevant in describing the behaviour of the dynamical system. The first, $\mathbf{x}$, are external variables which influence the behaviour of the system, such as the strain, strain rate and temperature. It is assumed that all of these can be measured. The second set of variables, $\mathbf{v}$, are the state variables which describe the system itself. These are split into

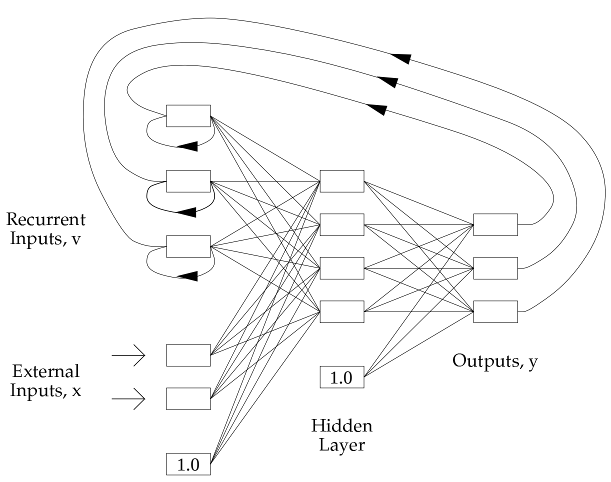

Fig. 6: A recurrent neural network architecture ('dynet') for modelling dynamical systems. The outputs, y, from the network are the time derivatives of the state variables of the dynamical system. The recurrent inputs, v, are these state variables. The values of v at the next time step are evaluated (using equation 9) from the outputs (via the recurrent connections) and the previous values of v. All connections and the two bias nodes are shown.

two categories. The first are measured, such as grain size, and the second are unmeasured, such as dislocation density. Note that the unmeasured variables are not intrinsically unmeasurable: this is simply a category for all of the state variables which we believe to be relevant but which, for whatever reason, we do not measure.

Both $\mathbf{x}$ and $\mathbf{v}$ are functions of time. A general dynamical equation which describes the temporal evolution of the state variables in response to the external variables is

$$\frac{\partial \mathbf{v}(t)}{\partial t} = \mathbf{F}(\mathbf{v}(t), \mathbf{x}(t)) \quad , \tag{7}$$

where $\mathbf{F}$ is some non-linear function. To a first-order approximation, we can write

$$\mathbf{v}(t + \delta t) = \mathbf{v}(t) + \frac{\partial \mathbf{v}(t)}{\partial t}\delta t \quad . \tag{8}$$

This dynamical system can be modelled with the recurrent neural network architecture shown in Figure 6. This is a discrete time network in which the input data are provided as a discrete list of values separated by known time intervals. The input–output mapping of this network implements equation 7 directly: Rather than producing the state variables at the output of the network, as is often the case with recurrent networks (e.g. [8]), we produce the *time derivatives* of the state variables, for reasons that are expounded on below. The hidden nodes compute a non-linear function of both the external and the recurrent inputs with a sigmoid function (e.g. tanh), as conventionally used in feedforward networks. A linear hidden–output function is used to allow for an arbitrary scale of the outputs. The recurrent part of the dynamical system, viz. equation 8, is implemented with the recurrent loops shown in Figure 6, by setting the weights of these recurrent loops to the size of the time step, δt, between successive epochs. Explicity, the k^{th} recurrent input at time step τ is given by

$$v_k(\tau) = v_k(\tau - 1) + y_k(\tau - 1)\delta t(\tau) \tag{9}$$

where $y_k(\tau - 1) = \partial \mathbf{v}(\tau - 1)/\partial t$ and $\delta t(\tau)$ is the time between epoch $(\tau - 1)$ and epoch (τ).

The principal reason for developing a network which predicts the time derivatives of the state variables is that it can be trained on time-series data in which the separations between the epochs, $\delta t(\tau)$, need not be constant: at each epoch τ we simply set the weights of the recurrent feedback loops to $\delta t(\tau)$. Furthermore, the network can be trained on multiple time series in which the time scales for each time series may be very different. This is important in forging applications as the forging of large components would occur over a longer time scale than for small components, whereas the microscopic behaviour of the materials would essentially be the same (for a given material). In such a case we would want to incorporate data from both forgings into the same model, but without having to obtain measurements at the same rate in both cases.

While our network is similar to that of Jordan [5], our architecture has the important attributes that: 1) the outputs are time derivatives of the state variables, and 2) in training the network the error derivatives can be propagated via the recurrent connections to the arbitrarily distant past. Our training algorithm can be seen as a generalisation of the method described by Williams & Zipser [11] extended to multiple time series. Although necessarily only the feedforward weights are trainable, the input–hidden weights, for example, are nonetheless dependent upon the values of the hidden nodes by virtue of the recurrent connections, and this dependency is taken into account. Training proceeds by minimizing an error function, typically the sum of squares error, by gradient descent or a conjugate gradient algorithm. The weights can be updated after each epoch of each time-series (i.e. Real Time Recurrent Learning [11]), after all epochs of all patterns, or at any intermediate point.

We use a Bayesian implementation of the training procedure [6]: multiple weight decay constants regularize the determination of the weights, and a noise term specifies the assumed noise levels in the targets ($\mathbf{v}$). Some form of regularisation is probably essential on account of the many intermediate output values for which their are no targets. The noise and weight decay hyperparameters are currently set by hand, but could in principle be inferred from the data.

To train the network we need at least one target value at at least one epoch. Note that the training algorithm is not restricted to use targets only for the 'outputs': errors can be propagated from any node. Generally we would have values of the state variables (recurrent inputs) for the final epoch. However, in metallurgical applications we may sometimes be able to obtain additional measurements at some intermediate epochs, thus improving the accuracy of the derived input–output function. We will of course not have any target values for the 'unmeasured' state variables. Hence these variables may not even correspond to any physical variables, instead acting as 'hidden' variables which convey some state information not contained in the 'measured' state variables. Nonetheless we may be able to provide some loose physical interpretation for unmeasured variables.

Once trained, the network produces a complete time sequence for all state variables given a sequence of the external inputs, i.e. the details of the forging process.

4.2. Model Demonstration

We now demonstrate the performance of the model on one particular synthetic problem in which there are two external input variables, $x_1(t)$ and $x_2(t)$, and two state variables, $v_1(t)$ and $v_2(t)$, defined by

$$\frac{\partial v_1}{\partial t} = x_1 - 2v_1 + 8v_2 - x_1 v_1 \tag{10}$$

$$\frac{\partial v_2}{\partial t} = x_2 - 5v_1 + v_2 - x_2 v_2 \ . \tag{11}$$

To mimic real processes in which the external inputs are often constrained to be positive (e.g. temperature), the x_1 and x_2 sequences were generated from constrained random walks: at each epoch, x_1 (x_2) changes with a probability of 0.1 (0.5) by a random amount uniformly distributed between -0.5 and $+0.5$ (-1 and $+1$). The modulus of x is taken to ensure a positive sequence. The initial v values were randomly selected from a uniform distribution between -1 and $+1$.

Equations 10 and 11 are not analytically solvable, so we used the first-order Taylor expansion in equation 8 to evaluate the $v_1(t)$ and $v_2(t)$ sequences. A very small value of δt was used to yield an accurate approximation to the true continuous sequence.

One hundred synthetic time series were generated in this fashion, with different sequences for x_1 and x_2 and different initial values of v_1 and v_2. The sequences were generated from $t = 0$ to $t = 8$ inclusive. We chose to sample these sequences at a constant epoch separation of $\delta t = 0.1$, giving a sequence length of 81 epochs. This time extent and value of δt samples the dynamical sequence well: If all x dependence is removed in equations 10 and 11, we are left with damped sinusoidal oscillations in v_1 and v_2, with oscillation period 1 and e^{-1} damping timescale 2. The x terms make the series somewhat more complex, but the overall oscillatory behaviour is retained.

The network was trained on 50 of the time series. In doing so, the network was only given the initial and final v values and the complete x sequences for each time series: it did not see the intermediate v values. Once trained, the network was applied to the other 50 sequences, i.e. given the initial v values and the x sequences, the network produces sequences for v. We see from Figure 7 that the network makes excellent predictions of v_1 and v_2 at the final epoch.

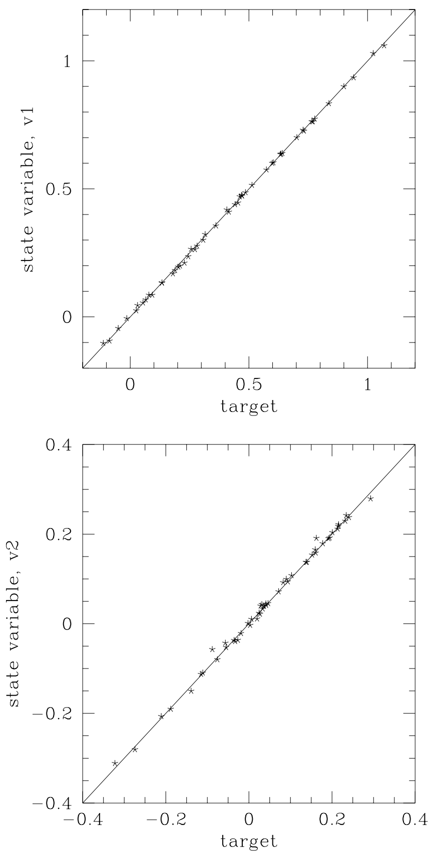

Fig. 7: Network predictions of the final values of the v_1 and v_2 sequences for the 50 time series in the test data set plotted against the target values. The rms errors are 0.0067 and 0.0084 for v_1 and v_2 respectively.

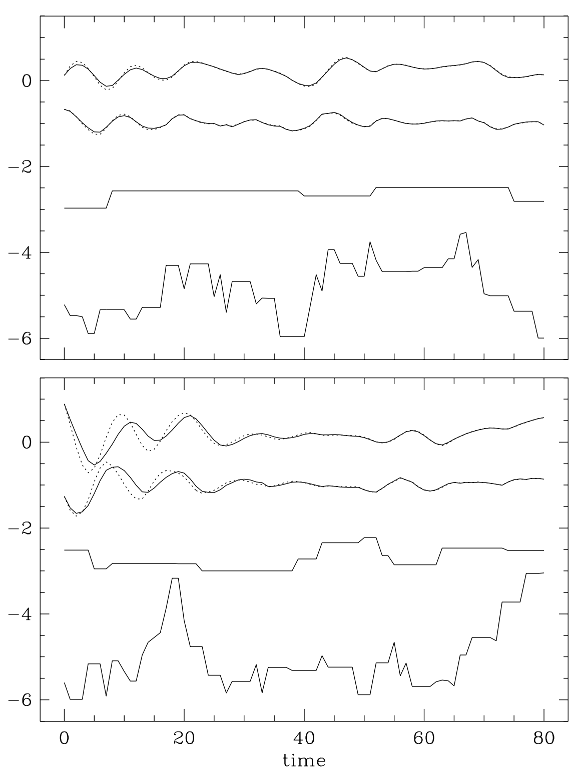

Fig. 8: External input and state variable sequences for two different time series in the test data set. In each plot the solid lines from top to bottom are v_1, v_2, x_1 and x_2. For v_1 and v_2 the solid lines are the nework predictions and the dashed lines the true sequences. For clarity, the sequences for v_2, x_1 and x_2 have been offset from their true positions by -1, -3 and -6 respectively.

It is now interesting to ask whether the network has managed to correctly learn the entire v sequences for the test data, or whether it has learned some simpler input–output function which just gives correct v values at the final epoch. Figure 8 shows the x input sequences, and the predicted v sequences in comparison with the true sequences: The agreement is generally very good. Some of the predicted sequences (such as the lower one shown) deviate from the true sequence at early epochs. This is probably due to the difference between the stationary distributions of the state variables at $t = 8$ and at early epochs: The network is explicity trained to give predictions for the target values of v. These lie in the ranges shown in Figure 8, which is narrower than the range of v values at $t = 0$. Therefore, we would not expect the network to give such good predictions in the non-overlapping part of this latter range. As the dynamical system is damped, the poorer predictions tend to occur earlier on. Nonetheless, we see that the network has learned a good approximation of the true underlying dynamical sequence for the range of v values present in the targets.

The network has been applied to a number of other synthetic problems (including ones with non-linear, e.g. v_1v_2, terms), and a good predicitive capablility is observed. The network has also been able to learn from training data sets in which δt varies within each sequence, and the lengths of the sequences differ [1].

5. Summary

We have introduced two techniques for modelling materials forging, one in the static domain relevant to cold forging, and another in the dynamic domain relevant to hot forging. The Gaussian process model used in the former case has been demonstrated to provide a good predictive capability, even when using a small, noisy data set. Furthermore, the sufficiency of our input variables (for a given material) has been demonstrated from the successful application of the trained Gaussian process model to a different forging geometry.

For the hot forging problem we have introduced a general neural network architecture for modelling dynamical processes. The power of this network was demonstrated on a synthetic problem. Further details of this model, as well as its application to a range of synthetic problems, will be presented in a forthcoming paper [1]. Future work with this model will focus on its application to real data sets obtained from hot forging.

Acknowledgements

The authors are grateful to the EPSRC (grant number GR/L10239), DERA and INCO Alloys Ltd. for financial support and to Mark Gibbs for use of his Gaussian process software.

References

[1] C.A.L. Bailer-Jones, D.J.C. MacKay, in preparation, 1998.

[2] T. Furu, H.R. Shercliff, C.M. Sellars, M.F. Ashby, "Physically-based modelling of strength, microstructure and recrystallisation during thermomechanical processing of Al–Mg alloys", *Materials Sci. Forum*, 217–222, pp. 453–458, 1996.

[3] L. Gavard, H.K.D.H. Bhadeshia, D.J.C. MacKay, S. Suzuki, "Bayesian neural network model for austenite formation in steels", *Materials Sci. Technol.*, vol. 12, pp. 453–463, 1996.

[4] M.N. Gibbs, *Bayesian Gaussian processes for regression and classification*. PhD thesis, University of Cambridge, 1997.

[5] M.I. Jordan, "Attractor dynamics and parallelism in a connectionist sequential machine", *Proc. of the Eight Ann. Conf. of the Cognitive Sci. Soc.*, Hillsdale, NJ: Erlbaum, 1986.

[6] D.J.C. MacKay, "Probable networks and plausible predictions: a review of practical Bayesian methods for supervised neural networks", *Network: Computation in Neural Systems*, vol. 6, pp. 469–505, 1995.

[7] F.N. Rhines, B.R. Patterson, "Effect of the degree of prior cold work on the grain volume distribution and the rate of grain growth of recrystallised aluminium", Metallurgical Transations A, vol. 13A, pp. 985–993, 1982.

[8] A.J. Robinson, F. Fallside, "A recurrent error propagation network speech recognition system", *Computer Speech and Language*, vol. 5, pp. 259–274, 1991.

[9] T.J. Sabin, C.A.L. Bailer-Jones, S.M. Roberts, D.J.C. MacKay, P.J. Withers, "Modelling the evolution of microstructures in cold-worked and annealed aluminium alloy", in T. Chandra, T. Sakai (eds), *Proc. of the Int. Conf. on Thermomechanical Processing*, Pennsylvania, PA: The Minerals, Metals and Materials Society, 1997.

[10] C.K.I. Williams, C.E. Rasmussen, "Gaussian processes for regression", in D.S. Touretzky, M.C. Mozer, M.E. Hasselmo (eds), *Neural Information Processing Systems 8*, Boston, MA: MIT Press, 1996.

[11] R.J. Williams, D. Zipser, "A learning algorithm for continually running fully recurrent neural networks", *Neural Computation*, vol. 1, pp. 270–280, 1989.