Machine Learning Techniques to Model Highly Nonlinear Multi-field Dynamics

Ruxandra Barbulescu[1(✉)], Gabriela Ciuprina[2], Anton Duca[2], and L. Miguel Silveira[3]

[1] INESC-ID, Rua Alves Redol 9, 1000-029 Lisbon, Portugal
ruxi@inesc-id.pt
[2] Politehnica University of Bucharest, Spl. Independentei 313, 060042 Bucharest, Romania
gabriela@lmn.pub.ro, anton.duca@upb.ro
[3] INESC-ID and IST Técnico Lisboa, Universidade de Lisboa, Av. Rovisco Pais 1, 1049-001 Lisbon, Portugal
lms@inesc-id.pt

Abstract. Modelling the dynamics of the membrane displacement in a micromachined beam fixed at both ends for different applied voltages is important for real applications. The strong nonlinearities involved and the interaction between multiple physical fields make this task challenging for classical modelling and model reduction approaches. In this work we search for a simplified, yet accurate, data-driven models, based on different recurrent neural network architectures, using only peripheral input-output information of the original system. The main goal is to find the most suitable neural network architecture having the smallest number of hidden units that provides low error of the minimum gap dynamics for different applied voltages. We show that these black-box models, with only 4 hidden units, are able to accurately reproduce the original system's response to a variety of different stimuli, and a strategy to make them parameter aware is proposed.

1 Introduction

Since their creation in research labs in the 1950's, Micro-Electro-Mechanical Systems (MEMS) and their Radio-Frequency (RF) variety have seen a wide range of applications, from sensors to switches, vehicle controls, pacemakers and even games. The basic structure of many RF-MEMS switches is based on a beam suspended like a bridge across a substrate, which is pulled down by a force (such as an electrostatic force) and eventually contacts a dielectric on the substrate thus blocking the signal (Fig. 1). The pull-in voltage and the response time are some of the most important parameters of electrostatically-actuated MEMS switches. Both the response time and the force needed to pull-in the membrane depend nonlinearly on its displacement and this dependence is the result of coupled electro-mechanical-fluid systems interaction.

One of the devices synthesising this mechanism is a micromachined beam fixed at both ends, often used as a benchmark for model reduction algorithms [1] and even studied as a pressure sensor [2]. A physics-aware model reduction approach is proposed in [3], where the low-order model is built by progressively adding physical phenomena. The dynamics of the bridge benchmark is described in detail in Sect. 2. The reduced

M. van Beurden et al. (Eds.): SCEE 2022, MI 43, pp. 125–132, 2024.
https://doi.org/10.1007/978-3-031-54517-7_14

model in [3] reproduces with high-fidelity the dependence between the pull-in voltage and the membrane displacement as well as the dynamic behaviour but, in the latter case, only a few basic input stimuli are considered in the modelling and reduction processes.

In all these approaches, the difficulty of modelling and producing simplified representations comes from the nonlinearity of the system and the interaction of more than one physical field. In particular, adding the air damping phenomena makes the system highly nonlinear. Physics-awareness can be both a plus and a minus, while gaining specificity and physical interpretation, one sacrifices generalization.

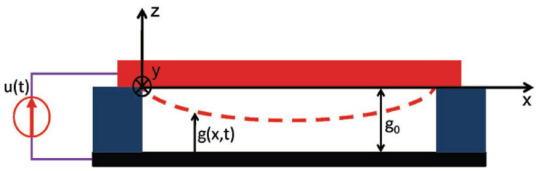

Fig. 1. The bridge benchmark (extracted from [3]).

In recent years, machine learning type models have been successfully used in various fields to tackle strongly nonlinear problems with considerable success. In this work, we use machine learning techniques to model the dynamics of the membrane displacement in the bridge benchmark, using only input-output information of the original full-order system. We generate data-driven, black-box models assuming no prior knowledge of the original system's structure and constitutive equations, which can further be explained using specific interpretation techniques for neural networks [4]. We train recurrent neural networks on datasets representing the system's response (membrane's minimum gap) to different input voltage signals (different shapes and magnitudes). We compare these architectures in terms of their properties and accuracy in reproducing the output of the original system. We show that with a recurrent layer of only 4 hidden units, it is possible to accurately reproduce the original system's response to a variety of different stimuli. We further show that we can generate parameter-aware models, which are able to predict with fidelity the system's behaviour for different values of specific parameters.

2 The Bridge Benchmark Dynamics

The bridge benchmark is a polysilicon beam of length $l = 610\,\mu\text{m}$, width $w = 40\,\mu\text{m}$ and height $h = 2.2\,\mu\text{m}$ suspended like a bridge over a silicon substrate. The initial gap is $g_0 = 2.3\,\mu\text{m}$. The mechanism is described by the strongly coupled 1D Euler's beam equation (1) and 2D Reynolds' squeeze film damping equation (2):

$$EI\frac{\partial^4 g}{\partial x^4} - S\frac{\partial^2 g}{\partial x^2} = F_{\text{elec}} - \rho\frac{\partial^2 g}{\partial t^2} + F_{\text{air}}, \tag{1}$$

$$\text{div}\left(\left(1+6\frac{\lambda}{g}\right)g^3 p\,(\text{grad}(p))\right) = 12\mu\frac{\partial(pg)}{\partial t}, \tag{2}$$

where $g(x,t)$ is the unknown gap (the displacement is $g_0 - g(p_m,t)$, where p_m is the middle point of the membrane $p_m = l/2$), $E = 149\,\text{GPa}$ is the Young modulus, $I = wh^3/12$ is the inertial moment, $S/(hw) = -3.7\,\text{MPa}$ is the initial stress, ρ is the mass per unit of length $\left(\rho/(hw) = 2330\,\text{kg/m}^3\right)$, $F_{\text{elec}}(x,t) = -\varepsilon_0 w v(t)/\left(2g^2(x,t)\right)$ is the electric force per unit of length (ε_0 is the air permittivity), $F_{\text{air}} = \int_0^w (p - p_a)\text{d}y$ is the damping force per unit of length, $p(x,y,t)$ is the unknown pressure, $p_a = 1.013 \cdot 10^5\,\text{Pa}$ is the environment pressure, $\lambda = 0.064\,\mu\text{m}$ is the mean free path of air and $\mu = 1.82 \cdot 10^{-5}\,\text{kg}/(\text{m}\cdot\text{s})$ is the air viscosity.

Since $l \gg w$, the deflection is assumed uniform across the width. Moreover, the deformation is symmetrical along the length of the membrane, therefore we can consider as quantity of interest the middle point $p_m = l/2$, where the gap is minimum. This point would also be the first touching the dielectric in case the membrane is pulled down completely. Figure 2 shows the membrane's displacement at different moments in time and the minimum gap for a periodic impulse.

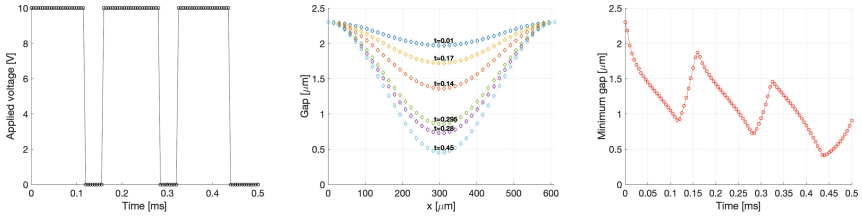

Fig. 2. Example of membrane displacement, generated with the original code from [3]). Left: Input – applied voltage $v(t)$. Middle: Displacement $g(x,t)$ at different moments in time (t in [ms]). Right: Output – minimum gap $g(p_m,t)$ in time.

3 Neural Networks Models

Recurrent Neural Networks (RNNs) [5–7] are a family of neural networks used for processing sequential data. Compared to their predecessors – the Feedforward Neural Networks (FFNNs) – the RNNs allow neurons in a given layer to form connections among themselves, thus being particularly adept to processing sequences of values $\mathbf{x}_1, \ldots, \mathbf{x}_t$ of equal or variable length. In this work we create models based on three architectures: the simple RNN [6], the Long Short-Term Memory (LSTM) unit [8,9] and the Gated Recurrent Unit (GRU) [10]. The structures of their cells are presented comparatively in Fig. 3.

In the simplest form of RNNs (Fig. 3-left), the prediction at a certain time point $\hat{\mathbf{y}}_t$ depends on the hidden state of the cell at the current time point $\mathbf{h}_t$, which in turn depends of the hidden state at the previous time point $\mathbf{h}_{t-1}$. In the following equations $\mathbf{V}$, $\mathbf{W}$ and $\mathbf{U}$ are matrices of weights, $\mathbf{b}$ and $\mathbf{c}$ arrays of biases and ϕ is an activation function, usually the hyperbolic tangent:

$$\mathbf{h}_t = \phi\left(\mathbf{V}\mathbf{x}_t + \mathbf{U}\mathbf{h}_{t-1} + \mathbf{c}\right), \qquad \hat{\mathbf{y}}_t = \mathbf{W}\mathbf{h}_t + \mathbf{b}.$$

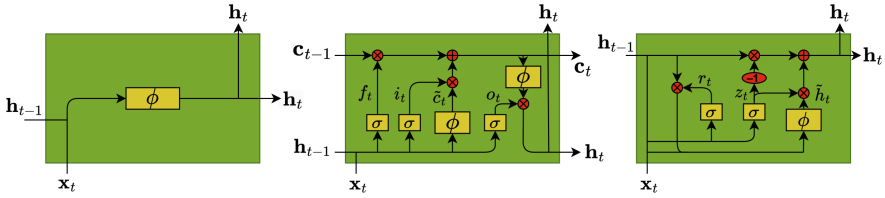

Fig. 3. Structure of a cell in the three architectures: simple RNN, LSTM, GRU (adapted from [11]).

The simple RNNs however are known to suffer from various issues, a delicate one being the vanishing gradient [12], which happens when long term components go exponentially fast to norm 0, making it impossible for the model to learn the correlation between temporally distant events. In our case, for a faithful reproduction of the dynamics, the simulations of the original model require the use of fine time steps, leading to datasets with long sequences. This in turn implies that the response at a given time will depend on values which are far back in the sequence. This situation, however unavoidable, may lead the RNN to experience difficulties in learning the data dependencies, resulting in a model with unacceptable error (usually measured in terms of Root-Mean-Squared Error – RMSE).

One solution to the vanishing gradient problem is the Long Short-Term Memory (LSTM) unit [8,9]. A LSTM consists of three main gates: the input gate $\mathbf{i}_t \in (0,1)^h$ that controls whether the cell state is updated or not, where h is the number of hidden units, the forget gate $\mathbf{f}_t \in (0,1)^h$ defining how the previous memory cell affects the current one and the output gate $\mathbf{o}_t \in (0,1)^h$, which controls how the hidden state is updated. The usage of gates is a major difference from the simple RNNs, since besides the hidden state $\mathbf{h}_t \in (-1,1)^h$, the LSTM also outputs a cell state $\mathbf{c}_t \in \mathbb{R}^h$ to the next LSTM unit, as shown in Fig. 3-center. The computation of the cell state is based on the candidate cell state $\tilde{\mathbf{c}}_t \in (-1,1)^h$. The vanishing gradient problem is partially solved by the LSTM units by allowing gradients to also flow *unchanged*. The LSTM mechanism is described by the following equations, where the learned parameters are the weights $\mathbf{W}_* \in \mathbb{R}^{h \times d}$ and $\mathbf{U}_* \in \mathbb{R}^{h \times h}$, and the biases $\mathbf{b}_* \in \mathbb{R}^h$, where d is the number of input features:

$$\begin{aligned}
\mathbf{i}_t &= \sigma\left(\mathbf{W}_i \mathbf{x}_t + \mathbf{U}_i \mathbf{h}_{t-1} + \mathbf{b}_i\right), & \tilde{\mathbf{c}}_t &= \phi\left(\mathbf{W}_c \mathbf{x}_t + \mathbf{U}_c \mathbf{h}_{t-1} + \mathbf{b}_c\right), \\
\mathbf{f}_t &= \sigma\left(\mathbf{W}_f \mathbf{x}_t + \mathbf{U}_f \mathbf{h}_{t-1} + \mathbf{b}_f\right), & \mathbf{c}_t &= \mathbf{f}_t \circ \mathbf{c}_{t-1} + \mathbf{i}_t \circ \tilde{\mathbf{c}}_t, \\
\mathbf{o}_t &= \sigma\left(\mathbf{W}_o \mathbf{x}_t + \mathbf{U}_o \mathbf{h}_{t-1} + \mathbf{b}_o\right), & \mathbf{h}_t &= \mathbf{o}_t \circ \phi\left(\mathbf{c}_t\right).
\end{aligned}$$

σ and ϕ are the logistic sigmoid and the hyperbolic tangent activation functions, respectively. The operator $\circ$ denotes the Hadamard product (element-wise product).

A simpler unit composed of only two gates is the Gated Recurrent Unit (GRU) [10], proposed in 2014. The GRU is described by:

$$\begin{aligned}
\mathbf{z}_t &= \sigma\left(\mathbf{W}_z \mathbf{x}_t + \mathbf{U}_z \mathbf{h}_{t-1} + \mathbf{b}_z\right), & \hat{\mathbf{h}}_t &= \phi\left(\mathbf{W}_h \mathbf{x}_t + \mathbf{U}_h\left(\mathbf{r}_t \circ \mathbf{h}_{t-1}\right) + \mathbf{b}_h\right), \\
\mathbf{r}_t &= \sigma\left(\mathbf{W}_r \mathbf{x}_t + \mathbf{U}_r \mathbf{h}_{t-1} + \mathbf{b}_r\right), & \mathbf{h}_t &= \left(\mathbf{1} - \mathbf{z}_t\right) \circ \mathbf{h}_{t-1} + \mathbf{z}_t \circ \hat{\mathbf{h}}_t,
\end{aligned}$$

where the weights $\mathbf{W}_* \in \mathbb{R}^{h\times d}$ and the biases $\mathbf{b}_* \in \mathbb{R}^h$ are learned parameters. The GRU (Fig. 3-right) is only composed of two gates, the update gate $\mathbf{z}_t \in (0,1)^h$ and the reset gate $\mathbf{r}_t \in (0,1)^h$. The update gate controls how much of the past information needs to be passed along to the future, while the reset gate is used to decide how much information the model should forget. The GRU only outputs the hidden state $\mathbf{h}_t \in \mathbb{R}^h$ computed based on the candidate hidden state $\hat{\mathbf{h}}_t \in (-1,1)^h$.

4 Results

Our first objective is, from a model reduction perspective, to find the most suitable architecture and the smallest neural network (in terms of hidden units) that provides good predictions. We therefore search for a nonlinear approximation $\mathscr{F}$ of the minimum gap $\hat{g}(p_m,t)$, for different applied voltages $v(t)$: $\hat{g}(p_m,t) = \mathscr{F}(v(t))$.

Data. We simulate the high-fidelity model from [3] described in Sect. 2 for 0.5 ms, with a time step of 5 μs and different input signals to generate snapshots of the system's dynamical behaviour. Each snapshot contains pairs of input/output (I/O) data for 100 points, namely the input voltage's variation in time $v(t)$ and the membrane's minimum gap $g(p_m,t)$ (where $p_m = l/2$ is the middle point of the membrane along the length), which takes values in the range $[0,2.3]$ μm. In the cases when the membrane is totally pulled down, the minimum gap is set to 0 for the remaining simulation time. We use scaled values (the gap is in μm, time is in ms), since the supervised learning uses an absolute error metric (the RMSE – Root-Mean-Squared Error), whose very low values might misdirect the training process.

We generate 40 snapshots, each with I/O values for 100 time moments, therefore amounting to 4000 pairs of I/O values, and divide them into three sets of data: the training set 50%, the validation set 25% and the test set 25%. The first two sets are used to compute the learned parameters, then the model is evaluated against the test set, containing data unseen before. The number of snapshots as well as their nature were chosen as to sufficiently sample the training and test distributions. Numerical tests showed that this choice was suitable for this benchmark.

Table 1. The average RMSE out of ten simulations, for RNN with 16 and 64 hidden units and for LSTM and GRU with 8 and 32 hidden units, for the iteration with the smallest validation loss.

	RNN-16	LSTM-8	GRU-8	RNN-64	LSTM-32	GRU-32
Training	0.0856	0.0376	0.0304	0.1728	0.0243	0. 0505
Validation	0.1335	0.0759	0.0896	0.1928	0.1781	0.1378

Implementation. The implementation is done in Python's libraries Keras and Tensorflow. We use a Normalization layer that shifts and scales the inputs into a distribution centered around 0 with standard deviation of 1, with the mean and variance adapted to the data. For a consistent comparison, we fixed the *hyperparameters* for all three

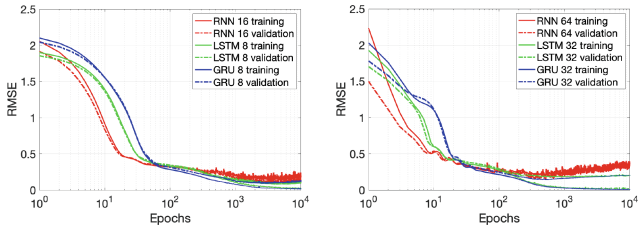

Fig. 4. Training and validation RMSE averaged over 10 runs, for different architectures and sizes.

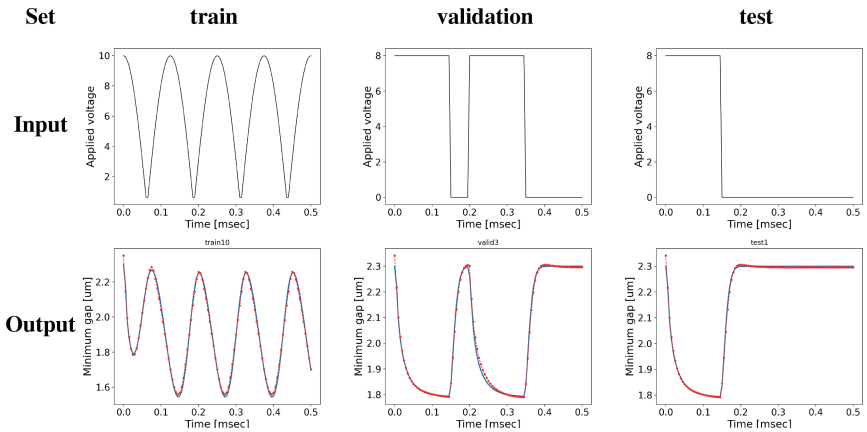

Fig. 5. Real (blue) and predicted (red) minimum gap for the GRU model with 8 hidden units extracted the training, validation and test sets.

architectures to the same previously optimized values, as follows: 1) We ordered the hyperparameters decreasingly based on their expected influence on the model's performance; 2) We set the most important one and so on; to set one, we kept fixed all the other hyperparameters and trained with different values for it. For the hyperparameters that are interdependent – for example batch size with sequence length, we did a grid search for different combinations of these. Our search in the hyperparameters space led to the following: the optimizer was set to Adam, the loss function is the RMSE, and the learning rate is 0.005. We trained the models for 10000 epochs (the number of passes of the training dataset through the algorithm), choosing the number of hidden units (neurons in the recurrent layer) so that the total number of parameters is comparable between models, e.g. a RNN with 16 hidden units (305 parameters) corresponds to a LSTM with 8 (328 parameters) and a GRU with also 8 hidden units (249 parameters).

Figure 4 shows the variation of the RMSE over the epochs, comparatively. Despite the lower complexity, the GRU performed best overall (see Table 1). In fact, with a recurrent layer as small as 4 hidden units, the RMSE is 0.0345 μm for the training and 0.0597 μm for the validation set. Figure 5 shows examples from the three sets, the input and the corresponding real and predicted outputs for the GRU with 8 units.

Parameter-Aware Models. A second objective is parameter-awareness, i.e. the ability of the neural network model to take into account geometrical characteristics and other parameters of the system that impact the output. We identified three important parameters: the membrane length l and width w, and the air viscosity μ. A series of potential values for each are listed in Table 2. We are now looking for an approximation that takes into account these parameters, of the form $\hat{g}(p_m,t) = \mathscr{F}(v(t),l,w,\mu)$.

Table 2. Different parameter values. The air viscosity is dependent on the temperature.

Parameter	Air viscosity μ $\left[\frac{kg}{m \cdot s}\right]$	Membrane length l [μm]	Membrane width w [μm]
Reference value	$1.82 \cdot 10^{-5}$ (15°)	610	40
Other values considered	$1.73 \cdot 10^{-5}$ (0°)	590	36
	$1.78 \cdot 10^{-5}$ (10°)	600	38
	$1.85 \cdot 10^{-5}$ (25°)	620	42
	$1.90 \cdot 10^{-5}$ (35°)	630	44

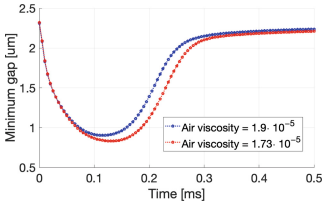

Fig. 6. Predicted minimum gap for two cases with the same input and fixed membrane length and width, but different air viscosity.

We generated new datasets containing the original 40 input-output data and random combinations of these values in a total of 500 examples (out of 5000 possible combinations) divided in 300 for training, 100 for validation and 100 for test). The 500 examples took more than $30h$ to generate with the original code. Using this data, we trained a GRU and re-optimized the hyperparameters.

The RMSEs obtained are of the same order of magnitude as for the previous case. Figure 6 shows the predicted output in two cases where the input is the same, as well as two of the parameters, the length and the width, but the air viscosity is different. The model successfully captures the difference in the output for the different values of air viscosity.

5 Conclusions

In this work we create data-driven models for the dynamics of the minimum gap in the bridge benchmark, using different recurrent neural network architectures. We show that

a GRU layer with only 4 hidden units accurately reproduces the output for various different stimuli, and we further propose a strategy to make the model parameter-aware. The main advantage of this model is the ability to accurately predict the response to various different stimuli and for different parameters. Moreover, once the neural network is trained, the prediction is done instantaneously. The source code, datasets and results are publicly available at https://github.com/ruxandrab/beam. Our next focus is to model the second half of the mechanism – the opening of the switch, as well as look into physics-informed neural networks, by embedding physical constraints that would allow both feature preservation and subsequent interpretation of the low-order models.

Acknowledgements. This work was supported by European Funds "Recovery and Resilience Plan - Comp. 5" included in the NextGenerationEU program, under project n° 62 - "Responsible AI" and Portuguese national funds, under projects UIDB/50021/2020, PTDC/EEI-EEE/31140/2017.

References

1. Rewienski, M., White, J.: A trajectory piecewise-linear approach to model order reduction and fast simulation of nonlinear circuits and micromachined devices. IEEE Trans. Comput.-Aided Des. Integrated Circ. Syst. **22**(2), 155–170 (2003)
2. Gupta, R.J., Senturia, S.D.: Pull-in time dynamics as a measure of absolute pressure. In: Proceedings IEEE the Tenth Annual International Workshop on MEMS. An Investigation of Micro Structures, Sensors, Actuators, Machines and Robots, pp. 290–294. IEEE (1997)
3. Ciuprina, G., Ioan, D., Lup, A.S., Silveira, L.M., Duca, A., Kraft, M.: Simplification by pruning as a model order reduction approach for RF-MEMS switches. COMPEL- Int. J. Comput. Math. Electr. Electron. Eng. **39**(2), 511–523 (2019)
4. Ismail, A.A., Gunady, M., Bravo, H.C., Feizi, S.: Benchmarking deep learning interpretability in time series predictions. arXiv preprint arXiv:2010.13924 (2020)
5. Elman, J.L.: Finding structure in time. Cogn. Sci. **14**(2), 179–211 (1990)
6. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. Nature **323**, 533–536 (1986)
7. Werbos, P.J.: Generalization of backpropagation with application to a recurrent gas market model. Neural Netw. **1**(4), 339–356 (1988)
8. Gers, F.A., Schmidhuber, J., Cummins, F.: Learning to forget: continual prediction with LSTM. Neural Comput. **12**(10), 2451–2471 (2000)
9. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
10. Cho, K., van Merrienboer, B., Gülçehre, Ç., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using RNN encoder-decoder for statistical machine translation. CoRR, abs/1406.1078 (2014)
11. Barbulescu, R., Mestre, G., Oliveira, A., Silveira, L.M.: Learning the dynamics of realistic models of *C. elegans* nervous system with RNNs. Sci. Rep. **13**(467) (2023)
12. Bengio, Y., Simard, P., Frasconi, P.: Learning long-term dependencies with gradient descent is difficult. IEEE Trans. Neural Netw. **5**(2), 157–166 (1994)