ELSEVIER

Electric Power Systems Research 32 (1995) 195–202

ELECTRIC POWER SYSTEMS RESEARCH

On CAD/CAM of microwave garnets: a codicil

Rodica Puflea-Ramer [a], Arthur Ramer [b], Tamas Gedeon [b], John Fagan [c]

[a] *School of Electrical Engineering, University of New South Wales, PO Box 1, Kensington, NSW 2033, Australia*
[b] *School of Computer Science, University of New South Wales, PO Box 1, Kensington, NSW 2033, Australia*
[c] *School of Electrical Engineering, University of Oklahoma, Norman, OK 73019, USA*

Received 10 October 1994

Abstract

Our earlier work demonstrated the feasibility of CAD for certain types of ferrimagnetic garnets. Experimental results and the theoretical model built thereupon were presented, along with a detailed discussion of numerical processing and computer implementation. Here we expand our discussion of the numerical processing, and suggest future use of software oriented towards symbolic and graphical processing. We follow with a proposal for applying neural networks, especially in the data analysis and model building phases. In the last part we present additional experimental data and their graphical analysis. The results fully agree with our earlier predictive model, based on the linear pattern of change of the magnetic parameters.

Keywords: Magnetic susceptibility; Microwave garnets; CAD/CAM of magnetic materials

1. Numerical and symbolic processing

Our earlier analyses [1,2] and the rudimentary CAD were implemented entirely in MATLAB [3]. The choice was motivated firstly by the wide availability of the package and its relatively modest demands on computing power. The almost universal familiarity with the package within the electrical engineering community was the other main reason for attempting such implementation; some of the resulting programs were presented in the Appendix to our earlier paper [1]. These codes, if viewed more closely (which we do not particularly recommend) would reveal several programming obscurities. They were caused by the intrinsic limitations of MATLAB. Simply, MATLAB was designed only for a limited audience, requiring 'quick and dirty' matrix manipulations for signal processing problems. It was not designed with much forethought as a more general programming language. For example, it has no facilities for representing three- and higher dimensional arrays, even though many vector and matrix problems naturally lead to such structures. The control and looping facilities are primitive, about on the level of earlier FORTRAN's. (This should not be construed as criticism — it permits very easy implementation of its basic functions, thus gaining in simplicity at the cost of certain sophistication.)

By way of illustration we will comment here on three troublesome problems encountered while using MATLAB. The first difficulty came at the data acquisition stage. Unpredictably, MATLAB has severe limitations on the length of input vectors; it also does not permit easy input of two- or higher dimensional data. MAPLE [4] and MATHEMATICA [5] have no difficulty in handling arbitrary-length inputs, whether as a single stream of numbers, or in a structured form. The latter refers to the option of entering data as a sequence of sequences, thus explicitly entering the matrix in a row-major fashion.

The second difficulty, much more significant, was the presentation of three-dimensional data. Ostensibly MATLAB can handle it, and its newest release (which was not available at the time of the original work) promised a reasonable support for the graphical display of data points and surfaces dependent on two parameters. In practice, attempts to use this facility failed. It was virtually impossible to present, for example, a three-dimensional plot, with the coordinate axes, and make the result at all comprehensible. MAPLE handles such problems quite easily, and MATHEMATICA is almost spectacular; management of graphical displays is one of the strongest points in its favor. Options are seemingly unlimited, with all sorts of rotations and changes of viewpoints, shading and coloring, and all kinds of scaling.

The third and last difficulty we discuss here falls entirely beyond the scope of MATLAB (to reiterate —

it is not a criticism, as the package was intended for a different audience and applications). It is the absence of symbolic computing. We performed basic curve fitting using certain least-squares based routines of MATLAB. Owing to the physical consideration of change of phase, we fitted curves separately in the adjacent temperature regions. Already, combining these three curves would have been handled better using their symbolic representations. More serious difficulty ensued with attempts to discover patterns and regularities as the chemical composition of the materials was varied. One of our major results was the discovery of an approximately linear behavior of a number of key magnetic parameters. Additional work would be needed to refine this model and allow for recognition of second-order (nonlinear) changes. To accomplish this efficiently, simple numerical processing of data is not likely to be sufficient. One possible direction is to use a symbolic representation for the magnetization curves. Another would be to dispense with 'hard' coefficients and attempt a form of pattern recognition using neural networks. They have been used in the past, with good success, for several difficult applications [6].

2. Neural networks

Neural networks are complexly interconnected collections of small processing elements. They are named in analogy with the human nervous system.

Biological neurons form a vast network in the nervous system, forming elaborate structures with extremely complex interconnections. The network receives inputs from receptors ranging from rods and cones in the retina to stretch receptors in muscles. These receptors convert stimuli into patterns of electrical activity in the network. After some processing, any outputs are expressed in terms of electrical activity to effectors such as muscles and glands. It has been estimated that there are of the order of 10^{10} neurons in the human nervous system, each of which is connected to 10^4 other neurons on average.

The connections between neurons are called synapses, and are chemical in nature. There is a synaptic gap which is bridged by the release of various chemicals called neurotransmitters. A neurotransmitter released into a synapse may be either excitatory, in that it increases the likelihood that the cell on the output side of the synapse will fire, or inhibitory in that its likelihood of firing is decreased. The release of chemicals from many synapses happens continuously. When the concentration of excitatory neurotransmitters diffusing into a cell from all its synapses is sufficiently higher than the concentration of inhibitory neurotransmitters, the cell will experience a chain reaction of electrical activity, which is called *firing*, and the pulse travelling down the axon will cause the release of neurotransmitters at all of its connections to other cells. The cell now has a refractory period during which it is unable to fire again, while the ions which leaked out to produce the electric pulse are pumped back in again. For the cell to fire, its threshold of excitation must be exceeded within some strict temporal neighborhood because, while it takes some time for the chemicals to diffuse across the synapse, if too long a period of time elapses after the release of the neurotransmitters from a particular synapse, they will diffuse away. This is called the period of latent summation. In the human brain the time from the arrival of impulses to firing and the refractory period is on the order of one thousandth of a second. From the point of view of modern digital computers, this is a very slow clock cycle, yet we manage to perform extremely complicated tasks using this architectural base.

The computational power of neural networks derives from simple local computations within individual elements, and their interactions with other processing elements, all operating in parallel. This is in contrast with the more familiar von Neumann computer architecture we find in modern digital computers, with their large, fast central processing unit which performs large numbers of calculations in sequence. It is worth pointing out that, while hardware to implement the requirements of truly parallel neural networks exists, and is becoming cheaper, most work and utilization of neural network methods still rely on simulations of the neural network on von Neumann machines. This emphasizes an important aspect, that the new paradigm of neural computation enables us to perform tasks which could not be readily programmed using conventional programming techniques.

2.1. Neural network training

A major advantage of neural computation is the ability of neural networks to learn from examples. In supervised learning, for example, networks are generally initiated with random weights on their connections. This means that the initial functionality is random. Thus, when the first input pattern is presented to the network, the result is not likely to be correct. Once the network has produced an output, the difference from the expected result can be calculated, and used to tweak the weights in the network slightly so as to increase the likelihood that a subsequent presentation of the same input pattern will produce the correct output. If a sufficiently large number of input patterns is presented to the network and its weights adjusted each time, eventually the network will *learn* the collection of input patterns and produce appropriate responses.

The essential procedure for training a feedforward neural network using the error back-propagation algorithm can be summarized in a few steps [7].

• Create input patterns from the raw data. A neural network input pattern for a supervised learning algorithm consists of a number of input values, and the expected, or correct, output value associated with those input values. These values should be normalized to a range from 0.1 to 0.9 for quicker learning by the neural network. Any well-known preprocessing steps which improve the information quality of the data should also be done, again to speed up learning.

• Separate data into training and test sets. The network should be trained on some 70% of the available input patterns, and intermittently tested on the remaining data without modifying weight connections during testing.

• Initialize the connection weights in the network to small random values. This provides some small random initial functionality which is necessary to break the all-zeros symmetry of the network. Without random initialization a synchronous simulation of a neural network will never learn complex examples.

• Present each input pattern in the training set to the network, and propagate the activations through successive layers until the output layer produces results.

• Compare the result with the desired output, and calculate an error measure. This error is used to modify the weights leading into the output layer of neurons to modify its output in the correct direction (either increase or decrease its value) for the current pattern. The modifications of the weights connecting to units in previous layers are used as a signal indicating the degree of correctness of the results in previous layers, and their input weights can also be adjusted. This propagates backwards until the weights from the input units are reached. With sufficiently small weight adjustments, this process approximates a true gradient descent on an error surface in the input pattern space.

• Continue presenting input patterns until the whole training set has been processed. The presentation of the whole training set in this way is called a training epoch.

• Test the network using the test set every 100 or so training epochs. The network weights are not adjusted during testing.

• Repeat the cycle of training and testing until training should be stopped. The training of the network should be stopped when the error on the test set stops decreasing and begins to increase. At this point the error on the training set usually continues to decrease, however the increasing error on the test set indicates that the network is overfitting the training set. If the data are of known high quality, or the number of data points is low, more sophisticated methods can be used to determine the point at which to stop training [8].

• The network can now be used on new input data for which no desired output is available. An input pattern is formed as above, and presented to the network. The network output can then be interpreted as the answer with the level of significance estimated from the success on the test set when training was stopped.

The back-propagation network has been used successfully in a number of domains, ranging from image compression [9] to financial bond rating.

The repeated presentation of all of the input patterns in the training set with only small modifications of the weights allows the network to learn a generalization of the training set without being unduly influenced by particular input patterns which may be outliers. These neural networks are particularly resistant to noisy data, and continue to perform well in such situations.

2.2. *Neural networks versus statistical methods*

Statistical methods always require the assumption of some particular functional form for relating dependent variables to independent variables. When this functional form does not reflect the reality of the relationship between the variables, the statistical technique can only confirm this, but cannot predict the correct functional form.

Feedforward neural networks provide a more general framework for determining relationships in the training data and do not require the specification of a particular functional form. The network forms internal representations which automatically learn an appropriate functional form at the same time as fitting the data to the functional form.

A secondary disadvantage of the practical use of functional models is that they are not readily updated when new data become available. They may need to be rebuilt with the newer data from scratch. Again, this is less problematic with feedforward neural networks, as extra training cycles incorporating the new data can be used to modify the internal representations formed by the process of back-propagation training to include any changes. Thus the previously trained network can be trained further, without losing the functionality represented by the weights on the connections between neurons.

2.3. *Modeling of microwave materials*

Input patterns are produced from the raw data which consist of the signals measuring various properties of the material, such as magnetization $4\pi M_s$ susceptibility χ, etc.

An example of typical output features could be the values of magnetic susceptibility versus temperature [10]. In our previous work [1] it was shown that separate single polynomial curves, of degree between 6 and 9, could be fitted to the data points. Polynomial curves of degree 7 were shown to be relatively good in all cases and were used in general to automate the process. These curves were also fitted piecewise to the data in

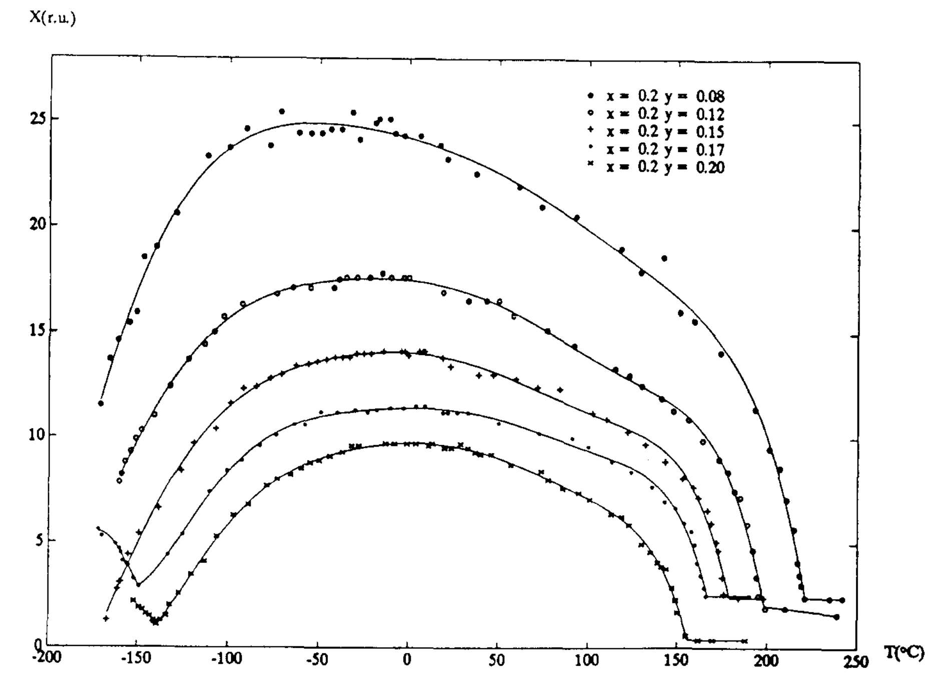

Fig. 1. Susceptibility vs. temperature of Gd- and Al-substituted YIG, $Y_{3-3x}Gd_{3x}Fe_{5-5y}Al_{5y}O_{12}$, with $x = 0.2$ and $y = 0.08$–0.20.

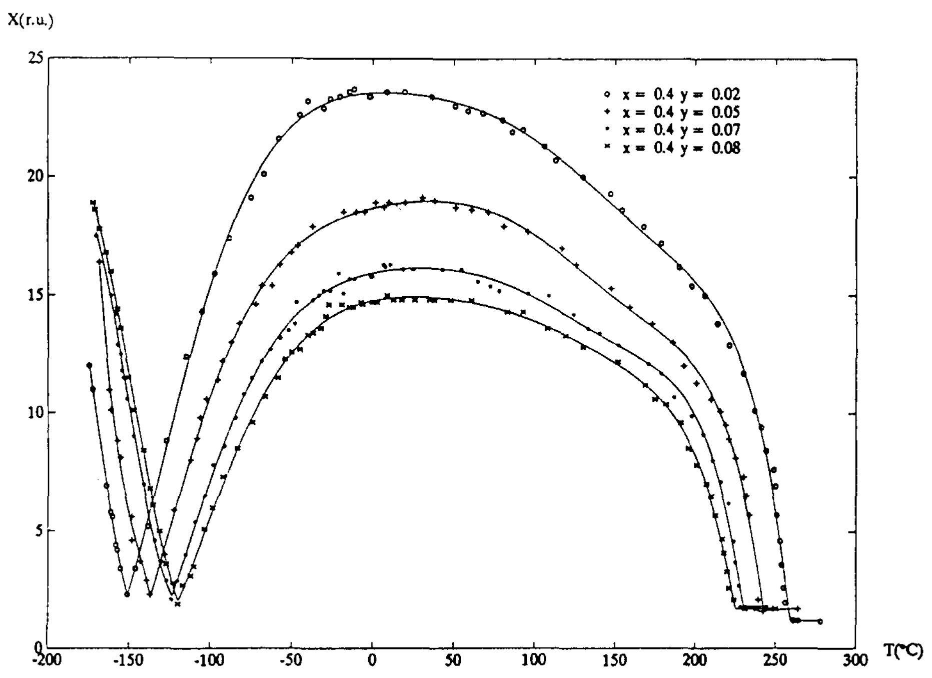

Fig. 2. Susceptibility vs. temperature of Gd- and Al-substituted YIG, $Y_{3-3x}Gd_{3x}Fe_{5-5y}Al_{5y}O_{12}$, with $x = 0.4$ and $y = 0.02$–0.08.

three sections — below the compensation temperature, in between the compensation and Curie temperatures, and above the Curie temperature. It was also indicated that for specific materials the polynomials of degree 7 did not provide the best fit.

Using the feedforward neural network trained using the error back-propagation model, we believe that the process of fitting these curves can be combined within a single neural network of appropriate complexity. The neural network must be sufficiently complex to be able to simulate the fitting of a polynomial of up to degree 9. In practice, it is known that for neural networks to learn quickly, excess functionality is initially required which may be removed later by the elimination of redundant neurons [11].

The raw data are used to produce training and test sets as described above. The network is then trained. The internal representation formed in the weight connections between neurons implicitly contains the curve fitted to the data points. For visual display, this can be

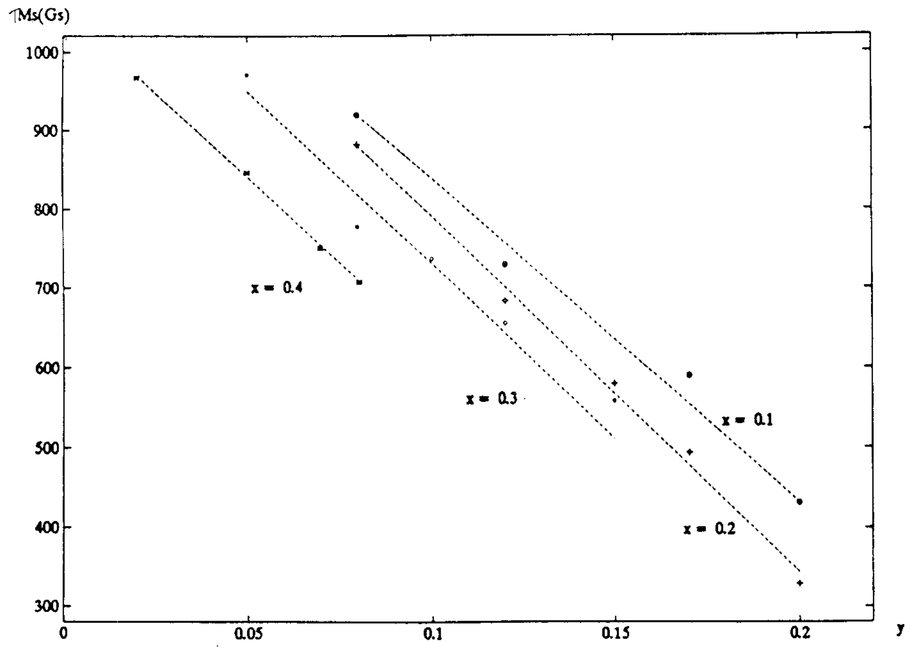

Fig. 3. Saturation magnetization vs. Al content of Gd- and Al-substituted YIG, $Y_{3-3x}Gd_{3x}Fe_{5-5y}Al_{5y}O_{12}$, with $x = 0.1$–0.4 and $y = 0.02$–0.20.

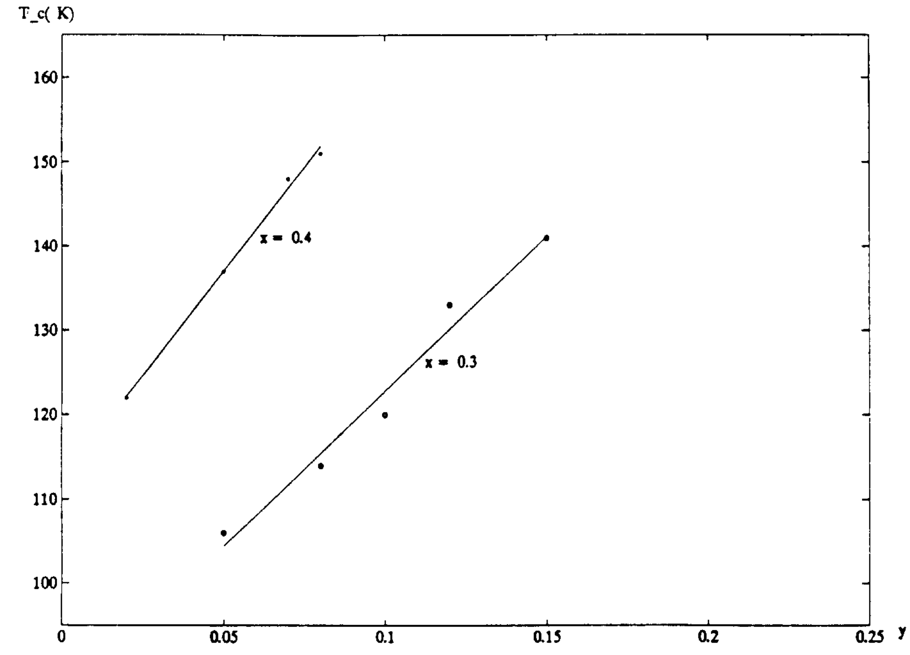

Fig. 4. Compensation temperature vs. Al content of Gd- and Al-substituted YIG, $Y_{3-3x}Gd_{3x}Fe_{5-5y}Al_{5y}O_{12}$, with $x = 0.3$ and 0.4 and $y = 0.02$–0.15.

extracted by running the trained neural network on all of the original data and plotting the results. For the training and test data, clearly the results will be close to the expected (measured) values, however, the noise in individual measurements will have been filtered out. It is now also possible to input intermediate patterns and use the network to produce results for them, rather than interpolating between actual measurement points.

3. Supplementary experimental results

3.1. Measurements and computed results

Following our earlier papers, we offer additional numerical evidence, further strengthening our thesis on the industrial feasibility of CAD/CAM for microwave garnets. We first present two additional families of susceptibility versus temperature curves (Figs. 1 and 2).

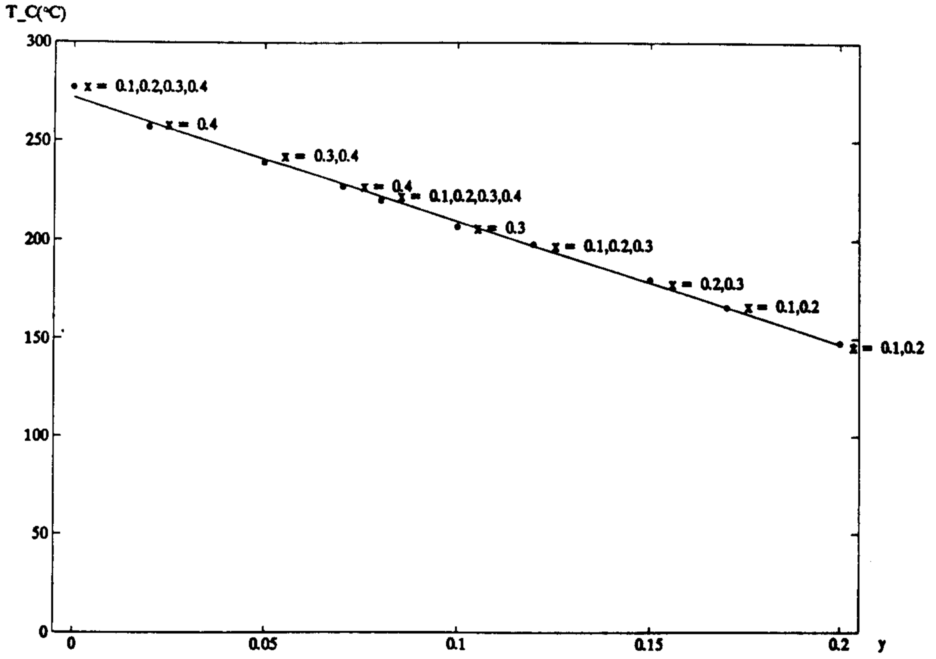

Fig. 5. Curie temperature vs. Al content of Gd- and Al-substituted YIG, $Y_{3-3x}Gd_{3x}Fe_{5-5y}Al_{5y}O_{12}$, with $x = 0.0$–0.4 and $y = 0.02$–0.20.

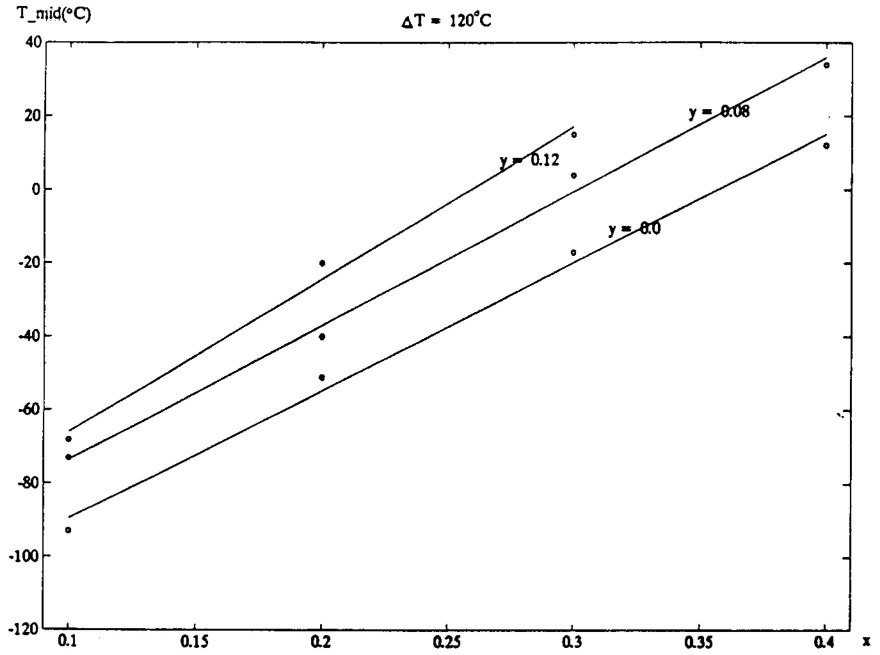

Fig. 6. The middle temperature of the stability zone vs. Gd content, for Gd- and Al-substituted YIG, $Y_{3-3x}Gd_{3x}Fe_{5-5y}Al_{5y}O_{12}$, with $x = 0.1$–0.4 and $y = 0.00$–0.12.

Next we show a composite diagram (Fig. 3) of several saturation magnetization curves, all very clearly linear. We follow with three diagrams (Figs. 4–6) showing the patterns of change of key temperature values. Two final diagrams (Figs. 7 and 8) show a novel verification of our CAD/CAM model [1,2].

3.2. Data analysis

Figs. 1 and 2 present changes of magnetic susceptibility as a function of temperature. Both diagrams show several superimposed curves; in each we keep the contents of yttrium and gadolinium constant ($x = 0.2$ and

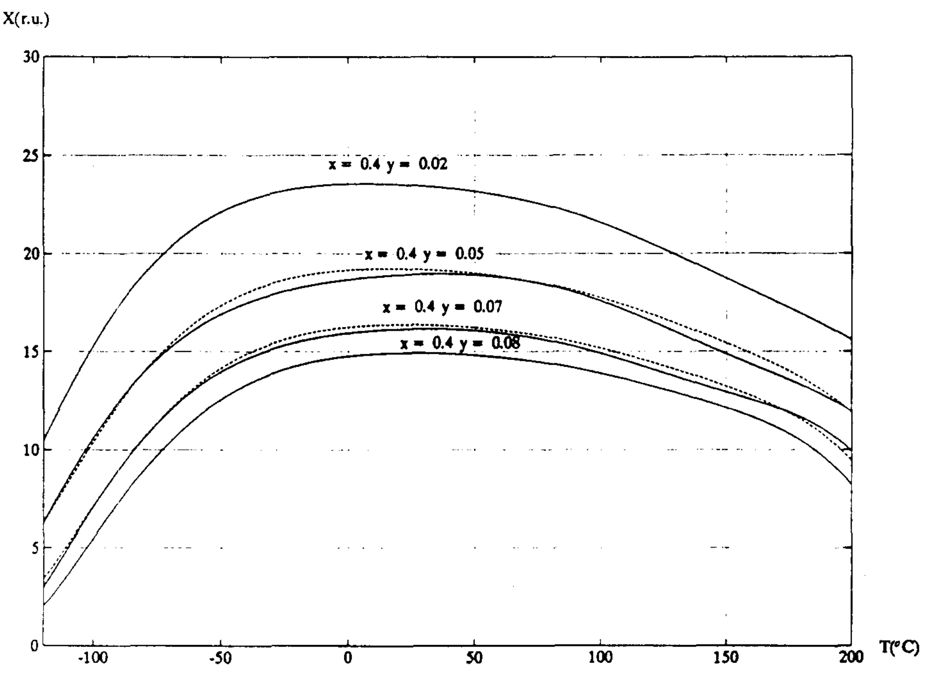

Fig. 7. Verification of the linear interpolation (broken line) of susceptibility vs. temperature of Gd- and Al-substituted YIG, $Y_{3-3x}Gd_{3x}Fe_{5-5y}Al_{5y}O_{12}$, for $x = 0.4$, $y = 0.05$ and $x = 0.4$, $y = 0.07$.

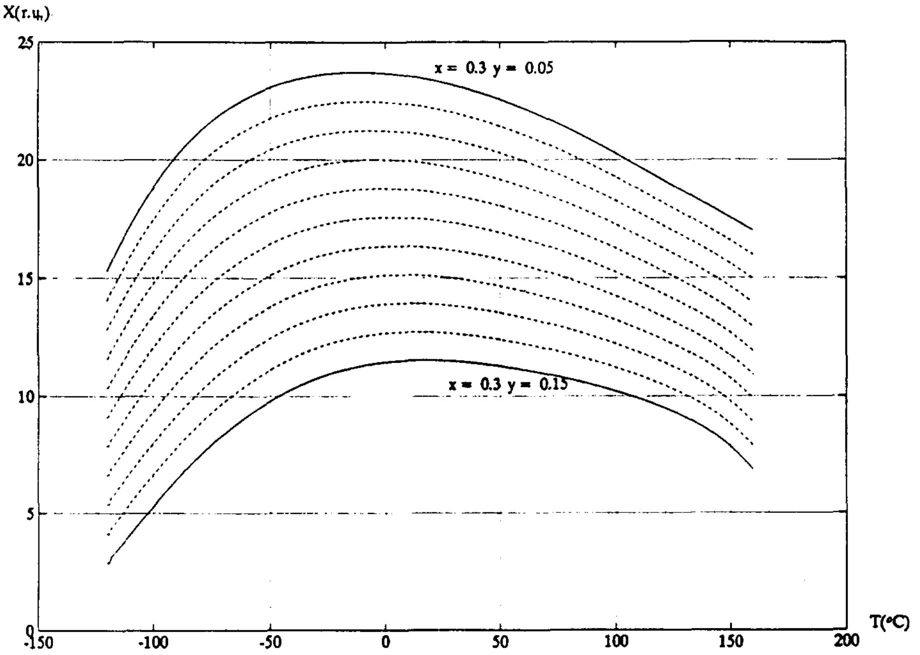

Fig. 8. CAD of the susceptibility vs. temperature of a hypothetical, novel, yet to be synthesized Gd- and Al-substituted YIG, $Y_{3-3x}Gd_{3x}Fe_{5-5y}Al_{5y}O_{12}$, with $x = 0.3$ and $y = 0.05$–0.15.

0.4, respectively), while varying the contents of iron and aluminum (from $y = 0.08$ to 0.20 in Fig. 1, and from $y = 0.02$ to 0.08 in Fig. 2).

Fig. 3 presents several examples of *linearly decreasing* saturation magnetization with increasing Gd and Al contents. Fig. 4 gives two examples of the linearity of the compensation temperatures, while Fig. 5 demonstrates that the Curie temperature remains constant with changes of Gd content, but decreases linearly when Al content is increased. The most significant for engineering applications are the linear patterns shown in Fig. 6. The location of the middle temperature of the stability zone can determine the actual industrial usefulness.

The last two diagrams (Figs. 7 and 8) offer new examples of the correctness of the linear interpolation and the resulting CAD approach to designing new compounds for their susceptibility properties.

References

[1] R. Puflea-Ramer, J. Fagan and A. Ramer, Computer modeling of microwave materials, *Electr. Power Syst. Res., 4* (1992) 141–148.

[2] R. Puflea-Ramer, K.D. McKinstry, J. Fagan and A. Ramer, Predictive analysis of linewidth parameters of microwave garnets, *Electr. Power Syst. Res., 24* (1992) 149–152.

[3] *PC-MATLAB for Personal Computers*, MathWorks, South Natick, MA, 1989.

[4] *Maple V Library Reference Manual*, Springer and Waterloo Maple Publishing, New York, 1991.

[5] S. Wolfram, *MATHEMATICA (A System for Doing Mathematics by Computer)*, Addison-Wesley, Reading, MA, 1991.

[6] S. Dutta and S. Shekhar, Bond rating: a non-conservative application of neural networks. *Proc. IEEE Int. Conf. Neural Networks, San Diego, CA, USA, 1988*, Vol. II, pp. 443–450.

[7] D.E. Rumelhart, G.E. Hinton and R.J. Williams, Learning internal representations by error propagation, in D.E. Rumelhart and J. McClelland (eds.), *Parallel Distributed Processing*, Vol. 1, MIT Press, Cambridge, MA, 1986.

[8] T.D. Gedeon and D. Harris, Hidden units in a plateau, *Proc. 1st Int. Conf. Intelligent Systems, Singapore, 1992*, pp. 391–395.

[9] T.D. Gedeon and D. Harris, Image quality driven compression, *Proc. 2nd Int. Conf. Image Processing, Singapore, 1992*, pp. 426–429.

[10] W.H. von Aulock (ed.), *Handbook of Microwave Ferrite Materials*, Academic Press, New York, 1964.

[11] T.D. Gedeon and D. Harris, Network reduction techniques, *Proc. Int. Conf. Neural Network Methodologies and Applications, San Diego, CA, USA, 1991*, Vol. 1, pp. 119–126.