**Evaluating magnetocaloric effect in magnetocaloric materials: a novel approach based on indirect measurements using artificial neural networks**

Angelo Maiorino\*, a, Manuel Gesù Del Ducaa, Jaka Tušekb, Urban Tomcb, Andrej Kitanovskib and Ciro Apreaa

*a Department of Industrial Engineering, Università di Salerno, Via Giovanni Paolo II, 132, 84084 Fisciano, Salerno, Italy*

*b University of Ljubljana, Faculty of Mechanical Engineering, Aškerčeva 6, 1000 Ljubljana, Slovenia*

\* Corresponding author

e-mail: [amaiorino@unisa.it](mailto:amaiorino@unisa.it)

**Contents**

[Overview 1](#_Toc5617118)

[Repository 2](#_Toc5617119)

[System requirements (Hardware and Software) 2](#_Toc5617120)

[Installation guide 2](#_Toc5617121)

[User information 2](#_Toc5617122)

[Instructions for use with provided data 4](#_Toc5617123)

[File dependencies 5](#_Toc5617124)

[Reproducibility 6](#_Toc5617125)

[Pseudocodes 7](#_Toc5617126)

# Overview

Here, we present a new procedure to evaluate thermodynamic properties of magnetocaloric materials based on Artificial Neural Networks (ANNs). This methodology has been developed to speed up the characterisation of new magnetocaloric materials and facilitate the design process for a magnetic refrigerator. Indeed, it requires a small amount of experimental data to characterise an MCM accurately, in comparison with other commonly-used techniques.

Three different codes are provided to run the procedure within the MATLAB environment. The first code is related to the training of the ANN starting from the experimental data. The second code allows to use the evaluation method within an AMR numerical simulation model. The last code provides a database of the thermodynamic properties of magnetocaloric materials, based on the learning capabilities of the Artificial Neural Networks.

To run the codes, execute the following m-files:

* ‘*MAIN\_for\_ANN\_training.m*’, for training an ANN;
* ‘*MAIN\_Adiabatic\_temperature\_change.m*’, for the evaluation of the adiabatic temperature change and the specific heat at a specific point;
* ‘*MAIN\_MagProp.m*’, for evaluate magnetocaloric properties of a specific material over the desired range.

# Repository

* <Code>: functions package;
* <Data>: data provided to test the codes;
* <Results>: some example of outputs.

# System requirements (Hardware and Software)

The proposed codes require only a standard computer with enough RAM to support the operations within MATLAB environment. No specific hardware is required. According to the release, a minimum value of 2 GB or 4 GB of RAM is suggested, for MATLAB R2017b and MATLAB R2018a, respectively. For better performance, values more than 4 GB and 8 GB are recommended for MATLAB R2017b and MATLAB R2018a, respectively. There are no minimum specifications for the performance of the CPU (<https://it.mathworks.com/support/sysreq/previous_releases.html>).

The codes have been tested on the following two systems:

1. OS: Windows 10 Home (version 1803)

CPU: 6 cores, 2.20 GHz/core, max. 4.10 GHz/core. RAM 16 GB

1. OS: Windows 10 Home

CPU: 2 cores, 1.70 GHz/core, max. 2.4 GHz/core. RAM 4 GB

About the software version, the following releases have been tested:

1. MATLAB R2018a (9.4);
2. MATLAB R2017b (9.3).

No specific toolbox are required to run the proposed codes.

# Installation guide

No specific software installation is required. Just MATLAB must be installed on your computer to run the codes. The time required to install this software is strongly related to the number of packages or toolboxes required for the execution of a code. In this case, this time should be the lowest possible since the codes here presented don’t need any specific additional package to be run.

# User information

The first code aims to train an Artificial Neural Network to predict the value of magnetisation and specific heat as a function of absolute temperature and magnetic field. It could be used when a new characterisation is needed or when one wants to improve the prediction performance of previous ANNs. To run this code, it is needed to enter the following command in the command window:

>> [Wh,Wo,map\_i,map\_t]=MAIN\_for\_ANN\_training()

The input are loaded after the code is launched from two .txt files. The name of these files can be changed in ‘*MAIN\_for\_ANN\_training.m*’ at line 27-28. The latter must have a structure like that in ‘*mag\_table\_Specimen1.txt*’ and ‘*cp\_table\_Specimen1.txt*’. The name of these files can be changed in ‘*MAIN\_for\_ANN\_training.m*’ at line 27-28. During the training procedure, the results of each trial are showed in the command window. After the training is completed, its performance are saved in a .txt file, as well as the parameters of the network, i.e. the synaptic weights and the input-target mapping. At line 44-48 of ‘*MAIN\_for\_ANN\_training.m*’ is possible to change the name of these files. The synaptic weights, the input-output mapping and the performance of the ANN training represent the output of the code. Some changes can be made according to needs. For example, the range of investigation of the hidden layer size can be modified at line 40-41 of ‘*MAIN\_for\_ANN\_training.m*’, as well as the maximum reasonable Mean Absolute Percentage Error (MAPE) at line 50. In ‘*Data\_clustering.m*’ is possible to change the percentages of data that compose the train, validation and test set whereas the training parameters can be modified in ‘*set\_train\_parameters.m*’. The running time of this code strongly depends on the parameters used to train the ANN. With the system used so far, it is in the range between 50 *s* and 480 *s*.

The second code could be integrated with an AMR numerical model for the evaluation of the adiabatic temperature change (*dTad*) and the specific heat (*cp*) based on the use of an ANN. It allows to calculate them considering the previous value of the magnetic field (*Hprev*), the new one (*H*) and the previous value of the temperature (T*prev*) of the MCM. To run this code, it is needed to enter the following command in the command window:

*>> [dTad,cp]=MAIN\_Adiabatic\_temperature\_change(Hprev,H,Tprev)*

To calculate the output variables, the parameters of the ANN, i.e. synaptic weights and input-output mapping, are loaded from four .txt file, which are defined at line 21-24 in ‘*MAIN\_Adiabatic\_temperature\_change.m*’. These files are provided as output by the execution of ‘*MAIN\_for\_ANN\_training.m*’. Their name must be changed according to the MCM, as well as the value of density at line 34. The running time of this code is about 0.04 *s* with the system 1 in System requirements Section.

The third code allows to point out the main thermodynamic properties of a MCM, based on the use an ANN. After a new MCM is characterised by ANN (running ‘*MAIN\_for\_ANN\_training*’), i.e. synaptic weights and input-output mapping are identified, one can execute this code to evaluate the behaviour of the MCM over the desired range. To run this code, it is needed to enter the following command in the command window:

*>> [T\_diagram,H\_array,mag,cp,dSiso,Stot\_th,Stot\_mag,dTad\_mag,dTad\_demag,dTad\_mag\_max,dTad\_demag\_max,Tpeak\_mag,Tpeak\_demag]=MAIN\_MagProp()*

The code loads the needed parameters from the .txt files where they have been previously saved as output of ‘*MAIN\_for\_ANN\_training*’. After launching the code, the program asks which kind of material you want to evaluate the properties. The already available materials are listed in ‘*Available Materials.txt*’, where it is possible to get the codes required to enter the query. Then, all the main magnetocaloric properties (magnetisation, specific heat, isothermal entropy change, adiabatic temperature change, peak temperatures) are calculated as functions of the absolute temperature and the magnetic field over the desired range. Some keyboard input are needed to define the calculation procedure, such as: temperature step, magnetic field step and maximum magnetic field. In the end, it is possible to save the data into a worksheet for further analysis and investigation and to compare the predicted values with the experimental results. Enter *‘y’* or *‘n’* for saving the data or not, respectively. The running time depends on the input entered from the keyboard during the execution of the code. Hence, it cannot be defined a priori.

# Instructions for use with provided data

To test the first code, it is just needed to run ‘*MAIN\_for\_ANN\_training.m*". The program will read the experimental data for training an ANN from the provided files related to the Specimen 1, ‘*mag\_table\_Specimen1.txt*’ and ‘*cp\_table\_Specimen1.txt*’. In the end, one can see the results by four different pictures, as follows:

* (Figure 1): values of MAPE as a function of different sizes of the hidden layer;
* (Figure 2): the evolution of mse (mean square error) over the epochs for the train, validation and test set;
* (Figure 3): the comparison of the predicted vs experimental magnetisation as a function of the absolute temperature at the maximum magnetic field;
* (Figure 4): the comparison of the predicted vs experimental specific heat as a function of the absolute temperature at the maximum magnetic field;

Furthermore, the performance of the training will be saved in ‘*Train\_results.txt*’ whereas the parameters of the trained ANN, i.e. synaptic weights linking input and hidden layer, synaptic weights linking hidden and output layer, input and target mapping, will be saved in '*Wh.txt*', '*Wo.txt*', '*map\_i.txt*' and '*map\_t.txt*', respectively.

To test the second code, it is needed to enter the following command in the command window:

*>> [dTad,cp]=MAIN\_Adiabatic\_temperature\_change(0,1,272)*

where 0 is the value of *Hprev* in T, 1 is the value of *H* in T and 272 is the value of previous temperature in K. One can change these values as it likes. The code will read the parameters of the ANN from the same files where they are saved after the execution of the training. Hence, in this case, the second code will provide the prediction of the adiabatic temperature change and the specific heat for the Specimen 1. These values will be displayed in the command window.

To test the third code, it is just needed to run ‘*MAIN\_MagProp.m*’. Before doing this, it is recommended to open ‘*Available Materials.txt*’, where the MCMs, for which the ANN parameters are provided, are listed with the respective code item needed to run the program. Once this file is open, the user can decide of which MCM wants to predict the properties. Then, the program will ask to set the temperature and the magnetic field step for the evaluation. It is recommended to choose between 0.001 *K* and 1 *K* for the temperature step, and between 0.1 *T* and 0.5 *T* for the magnetic field. This is for ensuring a good clarity of the figures. However, a smaller step of magnetic field can be selected. Then, the user must set the maximum magnetic field, that should be 1 *T*. In the end, it will be possible to save the data in a .xlsx file for further investigations. Enter *‘y’* or *‘n’* for saving the data or not, respectively.

# File dependencies

In this Section, file dependencies of each main code are clarified. They are shown in the following tables.

|  |  |  |
| --- | --- | --- |
| **Code 1: MAIN\_for\_ANN\_training.m** | | |
| INPUT | DATA FILE | SUB-FUNCTION |
| *No one* | *‘mag\_table\_Specimen1.txt’* | *ANN\_model.m* |
|  | *‘cp\_table\_Specimen1.txt’* | *ANN\_training.m* |
|  |  | *ANN\_training\_codeBP.m* |
|  |  | *batch\_training.m* |
|  |  | *Data\_clustering.m* |
|  |  | *data\_division.m* |
|  |  | *denormalize.m* |
|  |  | *Determination\_coefficient.m* |
|  |  | *Error\_calculation.m* |
|  |  | *Error\_calculation\_Target.m* |
|  |  | *Format\_mapping.m* |
|  |  | *initialize\_weights.m* |
|  |  | *normalize.m* |
|  |  | *Optimize\_network.m* |
|  |  | *random\_test.m* |
|  |  | *random\_train.m* |
|  |  | *random\_val.m* |
|  |  | *set\_train\_parameters.m* |
|  |  | *sorting\_data.m* |
|  |  | *tanlin\_ANN.m* |
|  |  | *Test\_normalization.m* |
|  |  | *testing.m* |
|  |  | *Training.m* |
|  |  | *validation.m* |
|  |  | *write\_parameters.m* |
|  |  | *write\_results.m* |

Table 1. File dependencies of the code for ANN training

|  |  |  |
| --- | --- | --- |
| **Code 2: MAIN\_Adiabatic\_temperature\_change.m** | | |
| INPUT | DATA FILE | SUB-FUNCTION |
| *Hprev* | *‘Wh.txt’* | *ANN\_model.m* |
| *H* | *‘Wo.txt’* | *denormalize.m* |
| *Tprev* | *‘map\_i.txt’* | *tanlin.m* |
|  | *‘map\_t.txt’* | *Test\_normalization.m* |

Table 2. File dependencies of the code for specific heat and adiabatic temperature change calculation

Note that, for the second code, one must change also the value of the density according to the material (at line 34).

|  |  |  |
| --- | --- | --- |
| **Code 3: MAIN\_MagProp.m** | | |
| INPUT | DATA FILE | SUB-FUNCTION |
| *No one* | *‘material\_Wh.txt’* | *ANN\_model.m* |
|  | *‘material\_Wo.txt’* | *ANN\_prediction.m* |
|  | *‘material\_mapi.txt’* | *denormalize.m* |
|  | *‘material\_mapt.txt’* | *MCM\_database.m* |
|  |  | *Peak\_temperature.m* |
|  |  | *Plot\_results.m* |
|  |  | *save\_data.m* |
|  |  | *tanlin\_ANN.m* |
|  |  | *Test\_normalization.m* |

Table 3. File dependencies of the code for the evaluation of thermodynamic properties of magnetocaloric materials

# Reproducibility

All the data reported in the paper “Magnetocaloric effect in Gadolinium and LaFe13\_x\_yCoxSiy alloys: a novel approach based on indirect measurements using artificial neural networks” by Angelo Maiorino, Manuel Gesù Del Duca, Jaka Tušek, Urban Tomc, Andrej Kitanovski and Ciro Aprea can be reproduced using the third code described above (‘*MAIN\_MagProp.m*’). All the parameters of the four ANNs developed and used for the four MCMs investigated are provided with the main code. Hence, to reproduce the same results, it is needed to run ‘*MAIN\_MagProp.m*’ four times, changing the material code each time according to ‘*Available Materials.txt*’. Furthermore, the temperature and the magnetic field step must be set to 0.1 *K* and 0.25 *T*, respectively. The upper limit of the magnetic field must be set to 1 *T*.

# Pseudocodes

|  |
| --- |
| **Pseudocode 1 (‘MAIN\_for\_ANN\_training’)**: A new Artificial Neural Network can be defined for new materials or to improve the performance of an existing ANN for known materials. During the training procedure, the results of each trial are showed in the command window. After the training is completed, its performance are saved in a .txt file, as well as the parameters of the network, i.e. the synaptic weights and the input-target mapping. The latter could be used as input of the codes for the calculation of the magnetocaloric properties. The results are showed by four plots. |
| **Input:** no input.  **Output:** the matrix of synaptic weights linking hidden layer and input layer (*Wh*), the matric of synaptic weights linking output and hidden layer (*Wo*), minimum and maximum values of inputs (*map\_*i) and minimum and maximum values of targets (*map\_t*).   1. **function** MAIN\_for\_ANN\_training 2. **read** experimental M **and** cp 3. **get size** M matrix 4. **determine** T\_array **and** M\_array 5. **determine** M matrix **and** cp matrix 6. **input** Ni **and** No 7. **call function** Format\_mapping **with** T\_array, M\_array, M, cp, Ni, No 8. **return** input **and** target 9. **end function** 10. **input** Nh\_min **and** Nh\_max 11. **compute** Nh\_array 12. **input** file: **string** to store Wh, Wo, map\_i, map\_t 13. **input** maximum\_MAPE 14. **initialize** net\_success, z **to** 0 15. **do while** net\_success = 0 16. **call function** ANN\_training **with** input, target, No, Nh 17. **return** ANN parameters and performance 18. **end function** 19. **compute** MAPE average 20. **if** MAPE average < maximum\_MAPE\*(1+z/10) 21. **print** 'MAPE value is lower than target. Its value is equal to:' 22. **print** MAPE average 23. **set** net\_success **to** 1 24. **plot** MAPE average vs Nh 25. **plot** mse vs epochs 26. **call function** write\_parameters **with** ANN parameters **and** files 27. **return** 28. **end function** 29. **call function** write\_results **with** performance 30. **return** 31. **end function** 32. **else** 33. **print** 'MAPE is too large. Its value is: ' 34. **print** MAPE average 35. **increment** z 36. **end if** 37. **end do** 38. **call function** ANN\_model **with** ANN parameters 39. **return** output 40. **end function** 41. **set** counter **to** 1 42. **initialize** M\_predicted **and** cp\_predicted 43. **for** each H in H\_array 44. **for** each T in T\_array 45. **set** M\_predicted (H, T) = output (1, counter) 46. **set** cp\_predicted (H, T) = output (1, counter) 47. **increment** counter 48. **end for** 49. **end for** 50. **plot** M\_predicted vs T at H maximum 51. **plot** cp\_predicted vs T at H maximum 52. **return** 53. **end function** |

|  |
| --- |
| **Pseudocode 2 (‘MAIN\_Adiabatic\_temperature\_change’)**: This code could be integrated with an AMR numerical model for the evaluation of the adiabatic temperature change and the specific heat based on the use of an Artificial Neural Network (ANN). It allows to calculate them considering the previous value of the magnetic field (Hprev), the new one (H) and the previous value of the temperature of the MCM. Furthermore, the code must load the parameters of the trained ANN, which depends on the specific MCM. |
| **Input:** value of the magnetic field at the previous time step (*Hprev*), value of the magnetic field at the actual time step (*H*), value of MCM temperature at the previous time step (*Tprev*).  **Output:** actual specific heat value (*cp*) and adiabatic temperature change (*dTad*).   1. **function** MAIN\_Adiabatic\_temperature\_change 2. **read** ANN parameters 3. **determine** Tmin, Tmax, Hmin, Hmax, Mmin, Mmax, Nh, Ni 4. **input** density 5. **determine** T\_array 6. **set** H\_array **to** 0 7. **initialize** input 8. **set** k **to** 1 9. **for** each T in T\_array 10. **set** input (k, 1) = H 11. **set** input (k, 2) = T\_array (T) 12. **increment** k 13. **end for** 14. **call function** ANN\_model **with** ANN parameters and input 15. **return** output 16. **end function** 17. **set** cp **=** output (2) 18. **initialize** s\_th 19. **set** s\_th (1) **to** 0 20. **for** each T in (T\_array-1) 21. **compute** s\_th (T) according to equation 26 22. **end for** 23. **set** H\_array = [Hprev H] 24. **initialize** ds\_iso 25. **compute** B according to equation 24 26. **for** each H in H\_array 27. **for** each T in T\_array 28. **compute** ds\_iso according to equations 23 and 25 29. **end for** 30. **end for** 31. **initialize** s\_mag 32. **for** each H in H\_array 33. **for** each T in T\_array 34. **compute** s\_mag according to equation 27 35. **end for** 36. **end for** 37. **initialize** Tmag 38. **initialize** dTad 39. **for** each T in (T\_array-1) 40. **call function** interp1 **with** s\_mag, T\_array, s\_mag (1, T) 41. **return** Tmag (T) 42. **end function** 43. **compute** dTad according to equation 28 44. **end for** 45. **set** input = [H; Tprev+dTad] 46. **call function** ANN\_model **with** ANN parameters and input 47. **return** output 48. **end function** 49. **set** cp = output (2) 50. **return** cp **and** dTad 51. **end function** |

|  |
| --- |
| **Pseudocode 3 (‘MAIN\_MagProp’)**: This code allows to point out the main thermodynamic properties of a MCM, based on the use of Artificial Neural Network (ANN). The code loads the needed parameters from the .txt files where they have been previously saved. All the main magnetocaloric properties (magnetisation, specific heat, isothermal entropy change, adiabatic temperature change, peak temperatures) are calculated as functions of the absolute temperature and the magnetic field over the desired range. It is possible to save the data into a worksheet for further analysis and investigation, and to compare the predicted values with the experimental results. |
| **Input:** no input.  **Output:** temperatures (*T\_array*), magnetic fields (*H\_array*), magnetisation values (*mag*), specific heat values (*cp*), isothermal entropy change values (*ds\_iso*), total entropy at zero magnetic field (*s\_th*), total entropy at different magnetic fields (*s\_mag*), adiabatic temperature change during the magnetisation (*dTad\_mag*) and the demagnetisation (*dTad\_demag*) phase, maximum adiabatic temperature change during the magnetisation (*dTad\_mag\_max*) and the demagnetisation (*dTad\_demag\_max*) phase, peak temperatures during the magnetisation (*Tpeak\_mag*) and the demagnetisation (*Tpeak\_demag*) phase. All these outputs are matrixes.   1. **function** MAIN\_MagProp 2. **keyboard input** MCM: **string** 3. **call function** MCM\_database **with** MCM 4. **return** ANN parameters **and** density 5. **end function** 6. **determine** Tmin, Tmax, Hmin, Hmax, Mmin, Mmax, Nh, Ni 7. **keyboard input** stepT, stepH, Hupper 8. **if** Hupper > Hmax 9. **warning print** 'The upper limit of the magnetic field is out of training range (1 T): the predictions may be inaccurate.' 10. **end if** 11. **determine** T\_array **and** H\_array 12. **call function** ANN\_prediction **with** ANN parameters, H\_array and T\_array 13. **return** mag **and** cp 14. **end function** 15. **initialize** s\_th 16. **set** s\_th (1) **to** 0 17. **for** each T in (T\_array-1) 18. **compute** s\_th (T) according to equation 26 19. **end for** 20. **initialize** ds\_iso 21. **compute** B according to equation 24 22. **for** each H in H\_array 23. **for** each T in T\_array 24. **compute** ds\_iso according to equations 23 and 25 25. **end for** 26. **end for** 27. **initialize** s\_mag 28. **for** each H in H\_array 29. **for** each T in T\_array 30. **compute** s\_mag according to equation 27 31. **end for** 32. **end for** 33. **initialize** Tmag, Tdemag, dTad\_mag, dTad\_demag 34. **for** each H in H\_array 35. **for** each T in (T\_array-1) 36. **call function** interp1 **with** s\_mag (H), T\_array, s\_th (T) 37. **return** Tmag (T) 38. **end function** 39. **compute** dTad\_mag (H, T) according to equation 28 40. **call function** interp1 **with** s\_th, T\_array, s\_mag (H, T) 41. **return** Tdemag (T) 42. **end function** 43. **compute** dTad\_demag (H, T) according to equation 28 44. **end for** 45. **end for** 46. **call function** max **with** dTad\_mag 47. **return** dTad\_mag\_max 48. **end function** 49. **call function** min **with** dTad\_demag 50. **return** dTad\_demag\_max 51. **end function** 52. **call function** Peak\_temperature **with** H\_array, T\_array, dTad\_mag, dTad\_demag, dTad\_mag\_max, dTad\_demag\_max 53. **return** Tpeak\_mag **and** Tpeak\_demag 54. **end function** 55. **call function** plot\_results 56. **return** 57. **end function** 58. **call function** save\_data 59. **return** 60. **end function** 61. **return** T\_array, H\_array, mag, cp, ds\_iso, s\_th, s\_mag, dTad\_mag, dTad\_demag, dTad\_mag\_max, dTad\_demag\_max, Tpeak\_mag, Tpeak\_demag 62. **end function** |