THE MONITORING OF A LATHE USING AN ARTIFICIAL NEURAL NETWORK – 2ND PART (THE TRAINING OF AN ARTIFICIAL NEURAL NETWORK and THE EXPERIMENTAL SETUP)

George C. BALAN, Alexandru EPUREANU, Floriana POPESCU

Universitatea *Dunarea de Jos* din Galati
Str. Domneasca nr. 47 Galati – 800 008 ROMANIA
George.BALAN@ugal.ro

The study of machine-tool dynamic is realized here as "monitoring", meaning checking and improving the functioning of the machine. The state of processing is followed by certain sensors whose signs are processed inside the computer and then it takes the decision of monitoring, meaning the identification of a class from the set of classes (process conditions).
In this part of the paper they are presented the batch training of an artificial neural network with high performance algorithm (Levenberg-Marquardt) and the experimental setup for components of cutting forces.

Keywords: monitoring, lathe, artificial neural networks

1. INTRODUCTION

In the first part of this paper is presented: -the classes (tool conditions) for monitoring in turning; -the artificial neural networks – **ANN**s (a neuron with a single R-element input vector, the transfer functions, an one-layer network, a three-layer network and the creation of an ANN with the function *newff*).

In addition to the "current stage" from [2]:

From [4] we quote: Since no exact and reliable mathematical models exist for the cutting process which are able to predict tool wear, tool breakage, cutting temperature and forces, the development of tool condition monitoring systems are highly requested by industry, especially in recent years. The purpose of this research is to develop a tool condition monitoring and a real-time tool life detection system in turning processes. The software and hardware generated for this research is based on the characteristic variations of the cutting forces. When a tool wears or breaks, cutting forces increase slightly right after the tool breakage and then decrease sharply. The change of cutting forces can itself be a good indicator to detect the tool failure. The performance of the system has been tested experimentally for certain machining conditions and the experimental results showed that the system was being successful in tool breakage detection

In [9] we remark: An effective wear-monitoring system for machine tool inserts could yield significant cost savings for manufacturers. One difficulty with collecting sensory information from machine tools is that the signal-to-noise ratio of useful information about the tool wear is extremely poor. This problem can be overcome by using advanced signal-processing methods and also by fusing the information obtained from numerous sensors into a single modeling or decision-making scheme such as neural networks. Neural networks are known for their capacity to solve problems effectively in cases where theoretical/analytical models cannot be established. Furthermore, neural networks can handle noisy and incomplete data such as that typically obtained from machining operations. In this paper a cost-effective and reliable tool condition monitoring system was developed, utilizing the advantages of neural networks for a typical industrial machining operation. The operation considered is interrupted turning (facing and boring) of Aluminum alloy components for the automotive industry.

An on-line tool condition monitoring system using a single-chip microcomputer for detecting tool breakage during cutting process is discussed in [12]. Conventionally, PC-based monitoring systems are used in most research works. To reduce costs, the tool condition monitoring system was built with an Intel 8051 single-chip microprocessor. The 8051 tool monitoring system uses a strain gauge for measuring cutting force; according to the force feature, the tool monitoring system can easily recognize the breakage of the cutting tool with its tool breakage algorithm.

In [7] acoustic emission and motor power sensors were used to detect the tool breakage in turning. Time–frequency analysis was used to process different acoustic emission signals emitted from the cutting process (normal cutting condition, tool breakage, chip fracture, etc.). Four types of power signal variation were observed in experiments when tool breakage occurred, which suggest that the change of power signals in the time domain was stochastic. Delayed variance is proposed to extract features from the power signals. The tool condition can be recognized through a neural network based on adaptive resonance theory.

[5] focuses on the two of the techniques, namely design of experiments and the neural network for predicting tool wear. In this work, flank wear, surface finish and cutting zone temperature were taken as response (output) variables measured during turning and cutting speed, feed and depth of cut were taken as input parameters. Predictions for all the three response variables were obtained with the help of empirical relation between different responses and input variables using design of experiments and also through neural network program. Predicted values of the responses by both techniques were compared with the experimental values and their closeness with the experimental values was determined. Relationship between the surface roughness and the flank wear and also between the temperature and the flank wear were found out for indirect measurement of the flank wear through surface roughness and cutting zone temperature.

In [14] is presented a new method for tool wear monitoring in precision hard turning is presented in this paper. The flank wear of a cutting tool is monitored by feature parameters extracted from the measured passive force, by the use of a force dynamometer. The feature parameters include the passive force level, the frequency energy and the accumulated cutting time. An ANN model was used to integrate these feature parameters in order to obtain more reliable and robust flank wear monitoring.

From [11] we quote: The supervision of tool wear is the most difficult task in the context of tool condition monitoring for metal-cutting processes. Based on a continuous acquisition of signals with multi-sensor systems it is possible to estimate or to classify certain wear parameters by means of neural networks. However, despite of more than a decade of intensive scientific research, the development of tool wear monitoring systems is an on-going attempt. 138 publications dealing with on-line and indirect tool wear monitoring in turning by means of artificial neural networks are evaluated in this paper.

[3] describes an application of three artificial intelligence methods to estimate tool wear in lathe turning. The first two are "conventional" artificial intelligence methods—the feed forward back propagation neural network and the fuzzy decision support system. The third is a new artificial neural network based-fuzzy inference system with moving consequents in if–then rules. Tool wear estimation is based on the measurement of cutting force components.

[6] describes a tool-wear monitoring procedure in a metal turning operation using vibration features. Machining of EN24 was carried out using coated grooved inserts, and on-line vibration signals were obtained. The measured tool-wear forms were correlated to features in the vibration signals in the time and frequency domains. Analysis of the results suggested that the vibration signals' features were effective for use in cutting tool-wear monitoring and wear qualification.

In [10] the implementation of a monitoring system utilizing simultaneous vibration and strain measurements on the tool tip, is investigated for the wear of synthetic diamond tools which are specifically used for the manufacturing of aluminum pistons. Data from the manufacturing process was recorded with two piezoelectric strain sensors and an accelerometer, each coupled to a DSPT Siglab analyzer. A large number of features indicative of tool wear were automatically extracted from different parts of the original signals. These included features from the time and frequency domains, time-series model coefficients (as features) and features extracted from wavelet packet analysis. A correlation coefficient approach was used to automatically select the best features indicative of the progressive wear of the diamond tools. The self-organizing map (SOM) was employed to identify the tool state. The SOM is a type of neural network based on unsupervised learning. A near 100% correct classification of the tool wear data was obtained by training the SOM with two independent data sets, and testing it with a third independent data set.

In [8] the sound pressure at 0.5 m from the cutting zone during the turning was measured by a condenser microphone and analyzed in frequency domain from 0 to 22 kHz. The measurement arrangement as well as the possibilities for on-line sensing of the tool wear are presented. The work piece material was carbon steel and the cutting insert was made of cermet without coating. The depth of cut was the constant, whereas the cutting speed, feed rate and flank wear of the tool were variables. The results showed that an increase in tool wear correlates with an increase in the amplitude of the recorded sound between 6 and 20 kHz. Similarly, an increase in feed rate resulted in an increase of sound intensity between 2 and 19 kHz. In contrast with these findings, the cutting speed influenced the recorded signals considerably less.

In [13] the influence of the wear's tool on the created surface is presented. Based on in-process acquisition of digital image of the surface it was possible to estimate or classify wear parameters by means of neural networks.

In this part of the paper they are presented the batch training of an ANN with high performance algorithm (Levenberg-Marquardt) and the experimental setup for components of cutting forces.

2. THE TRAINING OF AN A.N.N.

Several different training algorithms for feed-forward networks are described below. All of these algorithms use the gradient of the performance function E to determine how to adjust the weights to minimize performance. The gradient is determined using a technique called **back-propagation**, which involves performing computations backwards through the network.

The simplest implementation of back-propagation learning updates the network weights and biases in the direction in which the performance function decreases most rapidly - the negative of the gradient. One iteration of this algorithm can be written: $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha_k \boldsymbol{g}_k$ (1)
where $\boldsymbol{x}_k$ is a vector of current weights and biases, $\boldsymbol{g}_k$ is the current gradient, and α_k is the learning rate.

There are two different ways in which this gradient descent algorithm can be implemented: incremental mode and batch mode. In the incremental mode, the gradient is computed and the weights are updated after each input is applied to the network. In the batch mode all of the inputs are applied to the network before the weights are updated.

BATCH TRAINING (*train*). In batch mode the weights and biases of the network are updated only after the entire training set has been applied to the network. The gradients calculated at each training example are added together to determine the change in the weights and biases.

Batch Gradient Descent (*traingd*). The batch steepest descent training function is *traingd*. There are seven training parameters associated with *traingd*: *epochs, show, goal, time, min_grad, max_fail,* and *lr*. The learning rate *lr* is multiplied times the negative of the gradient to determine the changes to the weights and biases. The training status is displayed for every *show* iteration of the algorithm. The training stops if the number of iterations exceeds *epochs*, if the performance function drops below *goal*, if the magnitude of the gradient is less than *mingrad*, or if the training time is longer than *time* seconds. The following code creates a training set of inputs *p* and targets *t*. For batch training, all of the input vectors are placed in one matrix.

```
p = [-1 -1 2 2;0 5 0 5];
t = [-1 -1 1 1];
```

Next, we create the feed-forward network:

```
net=newff(minmax(p),[3,1],{'tansig','purelin'},'traingd');
```

Here we use the function *minmax* to determine the range of the inputs to be used in creating the network: *pr = minmax (p)*, where if *p = (R x Q)* matrix, then *pr = (R x 2)* matrix of min. and max. values for each row of *p*.

We are ready to train the network: [net,tr]=train(net,p,t);

Now the trained network can be simulated to obtain its response to the inputs in the training set:

```
a = sim(net,p)
a =
    -1.0010   -0.9989    1.0018    0.9985
```

Note that since we reinitialized the weights and biases before training (by calling *newff* again), we obtain a different mean square error than we did using *traingd*. The random choice of initial weights and biases will affect the performance of the algorithm. If you want to compare the performance of different algorithms, you should test each using several different sets of initial weights and biases. You may want to use *net=init(net)* to reinitialize the weights, rather than recreating the entire network with *newff*.

In this section we discuss several high performance algorithms that can converge from ten to one hundred times faster than the algorithm discussed previously. All of the algorithms in this section operate in the batch mode and are invoked using *train*. These faster algorithms use standard numerical optimization techniques for neural network training: quasi-Newton (*trainbfg*), and Levenberg-Marquardt (*trainlm*).

QUASI-NEWTON ALGORITHMS (BFGS Algorithm - *trainbgf*) Newton's method is an alternative to the conjugate gradient methods for fast optimization. The basic step of Newton's method is:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \boldsymbol{A}_k^{-1}\,\boldsymbol{g}_k \tag{2}$$

where $\boldsymbol{A_k}$ is the Hessian matrix (second derivatives) of the performance index at the current values of the weights and biases. Newton's method often converges faster than conjugate gradient methods. Unfortunately, it is complex and expensive to compute the Hessian matrix for feed-forward neural networks. There is a class of algorithms that is based on Newton's method, but which doesn't require calculation of second derivatives. These are called quasi-Newton methods. They update an approximate Hessian matrix at each iteration of the algorithm. The update is computed as a function of the gradient. The quasi-Newton method that has been most in published studies is the Broyden, Fletcher, Goldfarb, and Shanno (BFGS) update. This algorithm has been implemented in the *trainbfg* routine.

LEVENBERG-MARQUARDT (*trainlm*) Like the quasi-Newton methods, the Levenberg-Marquardt algorithm was designed to approach second-order training speed without having to compute the Hessian matrix. When the performance function has the form of a sum of squares (as is typical in training feed-forward networks), then the Hessian matrix can be approximated as: $\mathbf{H} = \mathbf{J}^{\mathbf{T}}\mathbf{J}$, and the gradient can be computed as: $\mathbf{g} = \mathbf{J}^{\mathbf{T}}\,\mathbf{e}$, where **J** is the Jacobian matrix that contains first derivatives of the network errors with respect to the weights and biases, and **e** is a vector of network errors. The Jacobian matrix can be computed through a standard back-propagation technique (that is much less complex than computing the Hessian matrix). The Levenberg-Marquardt algorithm uses this approximation to the Hessian matrix in the following Newton-like update:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - [\boldsymbol{J}^T \boldsymbol{J} + \mu\,\boldsymbol{I}]^{-1}\,\boldsymbol{J}^T \boldsymbol{e} \tag{3}$$

When the scalar μ is zero, this is just Newton's method, using the approximate Hessian matrix. When μ is large, this becomes gradient descent with a small step size. Newton's method is faster and more accurate near an error minimum, so the aim is to shift towards Newton's method as quickly as possible. Thus, μ is decreased after each successful step (reduction in performance function) and is increased only when a tentative step would increase the performance function. In this way, the performance function will always be reduced at each iteration of the algorithm.

In the following code, we reinitialize our previous network and retrain it using the Levenberg-Marquardt algorithm. The training parameters for *trainlm* are *epochs, show, goal, time, min_grad, max_fail, mu, mu_dec, mu_inc, mu_max, mem_reduc*. We have discussed the first six parameters earlier. The parameter *mu* is the initial value for μ. This value is multiplied by *mu_dec* whenever the performance function is reduced by a step. It is multiplied by *mu_inc* whenever a step would increase the performance function. If *mu* becomes larger than *mu_max*, the algorithm is stopped. The parameters show and epoch are set to 5 and 300, respectively.

```
p = [-1 -1 2 2;0 5 0 5];
t = [-1 -1 1 1];
net=newff(minmax(p),[3,1],{'tansig','purelin'},'trainlm');
net.trainParam.show = 5;
net.trainParam.epochs = 300;
net.trainParam.goal = 1e-5;
[net,tr]=train(net,p,t);
  TRAINLM, Epoch 0/300, MSE 2.7808/1e-05, Gradient 7.77931/1e-10
  TRAINLM, Epoch 4/300, MSE 3.67935e-08/1e-05, Gradient 0.000808272/1e-10
  TRAINLM, Performance goal met.
a = sim(net,p)
a =
  -1.0000   -1.0000    1.0000    0.9996
```

This algorithm appears to be the fastest method for training moderate-sized feedforward neural networks (up to several hundred weights).

3. THE EXPERIMENTAL SETUP

3.1.- COMPONENTS OF CUTTING FORCE

Forces F_y and F_z will be calculated on the basis of the experimental study of the lathe cutting-tool bending and with the help of two strain gauges *1* and *2* (fig.1), stuck on the lathe cutting-tool and connected to SPIDER [2].

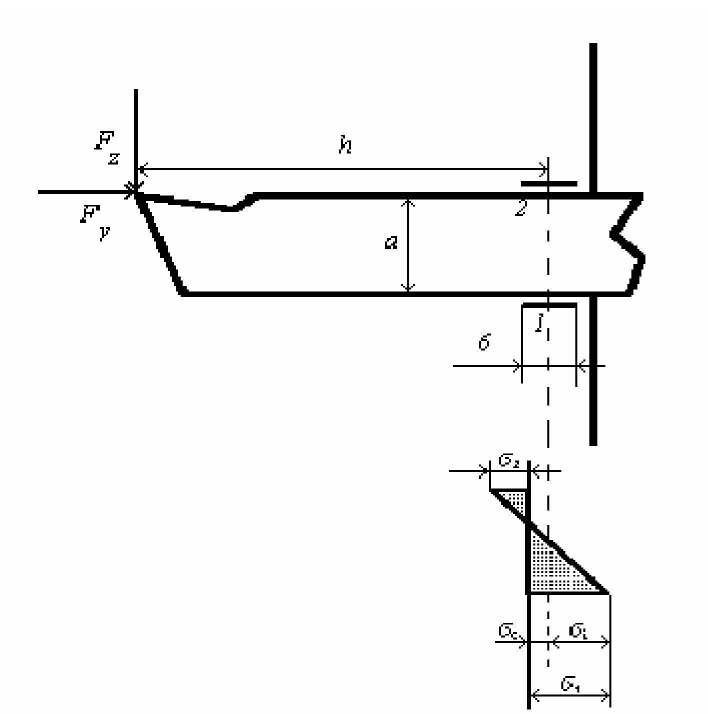

Fig. 1

The gauges are as type HBM (short, R = 120 Ω). The cutting-tool (armed with a cutting plate, whose dimensions are: $h = 37$ mm; $a = 23$; $b = 23$) is forced to bend by force F_z and to compress by F_x. Measuring relative deformations ε_1 and ε_2, the unitary efforts shown in figure 1 may be calculated:

$$\sigma_1 = E\,\varepsilon_1 \quad , \qquad \sigma_2 = E\,\varepsilon_2 \tag{4}$$

The unitary efforts of compressing and bending are:

$$\sigma_c = F_y \,/\, (a\,b) \quad , \qquad \sigma_i = M / W = (F_z\,h - F_y\,a/2)\,/\,(b\,a^2/6)\,. \tag{5}$$

In figure 1: $\sigma_1 = \sigma_i + \sigma_c \quad , \quad \sigma_2 = \sigma_i - \sigma_c$; (6)

it results: $\sigma_i = (\sigma_1 + \sigma_2)/2 \quad , \quad \sigma_c = (\sigma_1 - \sigma_2)/2$. (7)

The result is: $F_y = a\,b\,(\sigma_1 - \sigma_2)/2$. (8)

A tensometer system, identical with that one in figure 1, will be provided on the other cutting-tool, with a view to the thermal compensation.

The stress of cutting-tool bending is produced by F_z and F_y : $\sigma_z = \dfrac{F_z \cdot h}{b\ a^2/6}$; $\sigma_z = \dfrac{F_y \cdot a/2}{b\ a^2/6}$

We know that: $F_y \approx 0{,}3\,F_z$, so $\dfrac{\sigma_y}{\sigma_z} = \dfrac{0{,}3\ F_z \cdot 11{,}5}{F_z \cdot 37} = 0.09 = 9$ % ; therefore, the bending produced by F_y is negligible. The relations (5) become:

$$\sigma_i = \frac{F_z \cdot 37}{23^3/6} = 0{,}018 \cdot F_z \quad ; \quad \sigma_c = \frac{0{,}3 \cdot F_z}{23^2} = 5{,}7 \cdot 10^{-4} \cdot F_z \quad ; \quad \frac{\sigma_c}{\sigma_i} = 0{,}03 = 3 \ \%$$

In conclusion, the strains produced by F_y can be neglected and only the main cutting force is measured by means of tensometer system $F_z = a^2\ b\ \sigma\ /\ (\ 6\ h\ )$, (9)

where: $\sigma = (\sigma_1 + \sigma_2)\ /\ 2$. (10)

The gauges on upper faces of the two knives bound on a channel of SPIDER device, and the two on bottom faces – on other channel, according to *half-bridge* connection in figure 2.

The experiment was carried out on a lathe S N B 400 X 1500.

The process developed as follows:

a) - The gauges were stamped charging statically the cutting-tool on the recess face - near the top – with a

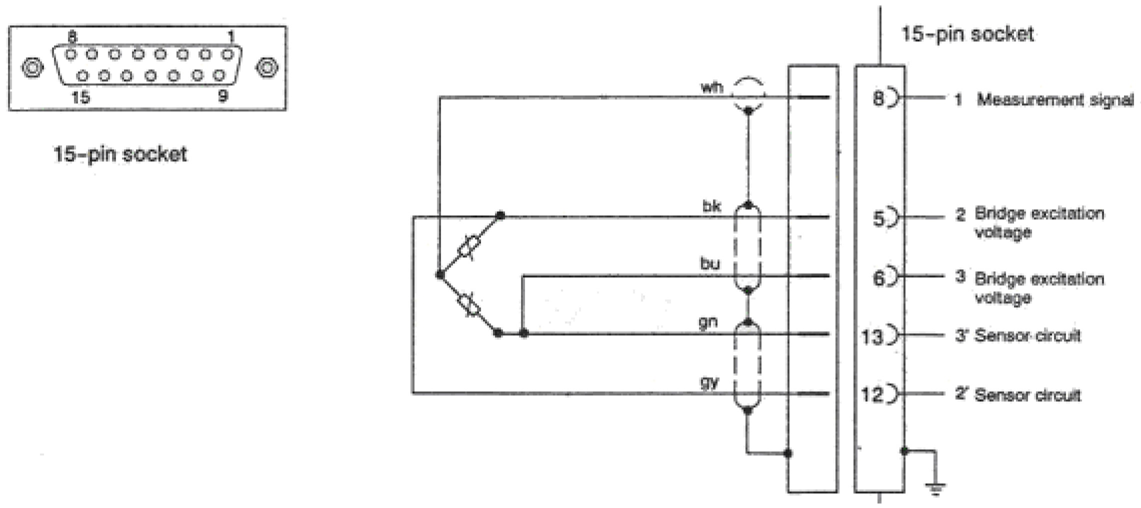

Fig. 2.

weight $G = 10\ daN$. The two acquired signals have values of opposite signs, because the strain gauges are solicited: *1* - at the compression, and *2* - at the stretching:

$$\varepsilon_1^{\,I\,n\,r} = -\ 0{,}0051 \quad ; \quad \varepsilon_2^{\,I\,n\,r} = 0{,}0037.$$

We consider this asymmetry to be due to the non-linearities in the system. On average:

$$\varepsilon_{I\,n\,r\,e\,g} = (\ 5\ ,1 + 3{,}7\ )\ 10^{-3}\ /\ 2 = 4{,}4\ .\ 10^{-3}\ . \quad (11)$$

The weight G provokes a relative stretching:

$$\varepsilon = \sigma / E = M\ /\ (\ E\ W\ ) = G\ h\ /\ (\ E\ b\ a^2\ /\ 6\ ) = 6\ x\ 3{,}7\ /\ (\ 2{,}1\ x\ 10^{6}\ x\ 2{,}3^3)\ x\ G\ ;$$

$$\varepsilon = 0{,}869\ x\ 10^{-6}\ x\ G \ \rightarrow\ \varepsilon_{real} = 8{,}69\ x\ 10^{-6} \quad (12)$$

The relative stretching scale: $K_F = \varepsilon_{r\,e\,a\,l}\ /\ \varepsilon_{I\,n\,r\,e\,g} = 1\ ,957\ x\ 10^{-3}$. (13)

b)- 191 recordings were made during the longitudinal turning of a OLC 45 cylinder (Φ 113, L = 1000), with a lathe cutting-tool with metal carbide P 20 and ℜ = 45°. For example, for the working regime no.101: the turn $n = 250$ rot / min., the longitudinally advance $s = 0{,}2$ mm / rot., the cutting depth $t = 1$ mm, the piece diameter $\varnothing = 107$ mm) oscillograms in figure 3 were obtained, in which the curve with negative coordinates is $\varepsilon_1^{\,I\,n\,r}$ and the one with positive coordinates is $\varepsilon_2^{\,I\,n\,r}$.

The numbers on abscissa in this figure and in the similar figures below shows the multiples of the sampling time interval:

$$D\ t = 0{,}5\ /\ 4800 = 1{,}041666\ .\ 10^{-4}\ s. \quad (14)$$

The vibro-recordings qualitative aspect is considered to be adequate, by verifying the relation

$|\ \varepsilon_1\ | > \varepsilon_2$, or $\sigma_1 > \sigma_2$, resulted from expressions (6).

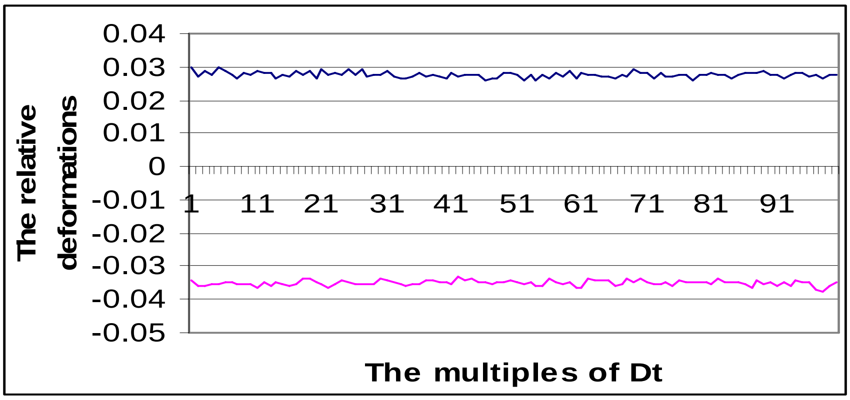

Fig. 3

As a rule, at the beginning and at the end of registering a deviation is noticeable, which is - mainly – due to the thermal non-compensation: the working knife gets warmer and the witness knife remains at the environmental temperature. As each working regime lasted nearly 30 sec., in the first half of the interval the Spider device getting connected for 1 sec., between registerings being (obviously) a break for the changing of the turning regime parameters and for transmitting of the collected data to PC, we will consider the thermal non-compensation as negligible.

The formula (9) becomes, considering (10 and 13) and the fact that $\varepsilon_1 < 0$:

$$F_z = \frac{a^2 b}{6\ \ h} \cdot \frac{\sigma_1 + \sigma_2}{2} = \frac{a^2 b}{6\ \ h} \cdot E \cdot \frac{\varepsilon_2^{real} - \varepsilon_1^{real}}{2} = \frac{a^2 b}{6\ \ h} \cdot E \cdot K \cdot \frac{\varepsilon_2^{inr} - \varepsilon_1^{inr}}{2} =$$

$$= \frac{2,3^3}{12 \cdot 3,7} \cdot 2,1 \cdot 10^6 \cdot 1,975 \cdot 10^{-3} (\varepsilon_2^{inr} - \varepsilon_1^{inr})$$

$$F_z = 1136\,(\varepsilon_2^{Inr} - \varepsilon_1^{Inr})\ \ [\text{daN}], \tag{15}$$

For example, fig. 4 shows the oscillogram $F_z(t)$ in experiment no. 101.

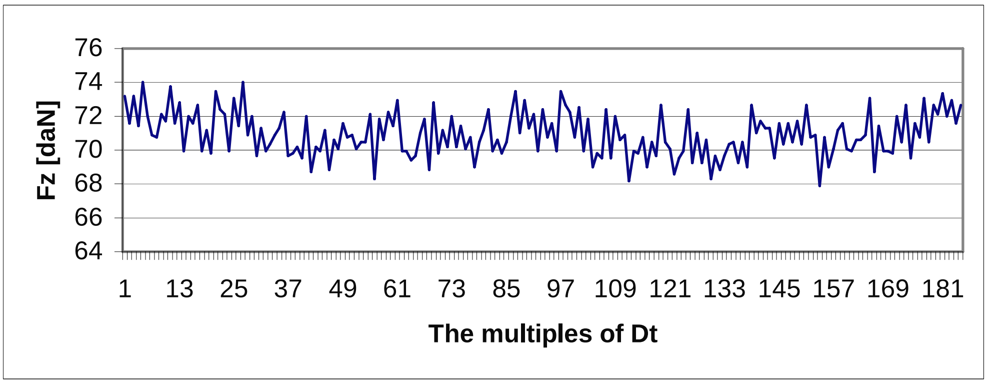

Fig. 4

4. ACKNOWLEDGEMENT

This research was supported by a grant CNCSIS [1].

REFERENCES

1. BALAN, G., *Monitorizarea unui strung utilizand o retea neuronala artificiala*, Contract (grant tip A) nr 33 445 / '02, Tema 19, Cod CNCSIS 451.
2. BALAN, G., EPUREANU, A., *The monitoring of a lathe using an artificial neural network – 1-st part (Theoretical basis)*, pe CD-ul "The Annual Symposium of the Institute of Solid Mechanics SISOM 2004, Academia Romana, Bucuresti, May 2004".

3. BALAZINSKI, M., CZOGALA, E., JEMIELNIAK, K., LESKI, J., *Tool condition monitoring using artificial intelligence methods,* Engineering Applications of Artificial Intelligence, Volume 15, Issue 1, February 2002, pp. 73-80.
4. CAKIR, C., ISIK, Y., *Detecting tool breakage in turning AISI 1050 steel using coated and uncoated cutting tools*, Journal of Materials Processing Technology, Volume 159, Issue 2, 30 January 2005, pp.191-198.
5. CHOUDHURY, S.K., BARTARYA, G., *Role of temperature and surface finish in predicting tool wear using neural network and design of experiments*, International Journal of Machine Tools and Manufacture, Volume 43, Issue 7, May 2003, pp.747-753.
6. DIMLA, D. E., *The Correlation of Vibration Signal Features to Cutting Tool Wear in a Metal Turning Operation*, The International Journal of Advanced Manufacturing Technology, Springer-Verlag London Ltd, Volume 19, Number 10, June 2002 , pp. 705 - 713
7. HAILI, W., HUA, S., MING, C., DEJIN, H., *On-line tool breakage monitoring in turning*, Journal of Materials Processing Technology, Volume 139, Issues 1-3, 20 August 2003, pp. 237-242.
8. KOPAC, J., SALI, S., *Tool wear monitoring during the turning process*, Journal of Materials Processing Technology, Volume 113, Issues 1-3, 15 June 2001, pp. 312-316.
9. SCHEFFER, C., HEYNS, P. S., *An industrial tool wear monitoring system for interrupted turning*, Mechanical Systems and Signal Processing, Volume 18, Issue 5, September 2004, pp. 1219-1242.
10. SCHEFFER, C., HEYNS, P. S., W*ear monitoring in turning operations using vibration and strain measurements*, Mechanical Systems and Signal Processing, Volume 15, Issue 6, November 2001, pp. 1185-1202.
11. SICK, B., *On-line and indirect tool wear monitoring in turning with artificial neural networks: a review of more than a decade of research*, Mechanical Systems and Signal Processing, Volume 16, Issue 4, July 2002, pp. 487-546.
12. TSENG, P.-C., TENG, W.-C., *The design of a single-chip tool monitoring system for on-line turning operation*, The International Journal of Advanced Manufacturing Technology, Springer-Verlag London Ltd , Volume 24, Numbers 5-6, September 2004 , pp: 404 - 414
13. ZAWADA-TOMKIEWICZ, A., *Classifying the wear of turning tools with neural networks*, Journal of Materials Processing Technology, Volume 109, Issue 3, 15 February 2001, pp. 300-304.
14. ZHOU, J. M., ANDERSSON, M., STAHL, J. E., *The monitoring of flank wear on the CBN tool in the hard turning process*, The International Journal of Advanced Manufacturing Technology, Springer-Verlag London Ltd , Volume 22, Numbers 9-10,, November 2003, pp. 697 – 702
15. MATLAB 6.5