ANYTIME SOFT COMUTING METHODS FOR INTELLIGENT MEASUREMENT, DIAGNOSIS AND CONTROL

Orsolya Takács, Annamária R. Várkonyi-Kóczy

Dept. of Measurement and Information Systems
Budapest University of Technology and Economics
Műegyetem rkp. 9., Budapest, H-1521 Hungary
Phone: +36 1 463 2057, Fax: +36 1 463 4112
takacs@mit.bme.hu, koczy@mit.bme.hu

Abstract: In systems, where the available time and resources are limited, and even could change during the operation of the system, special algorithms, computing methods, the so called anytime techniques could be used advantageously. While different soft computing methods are wide-spreadly used in nearly every area of engineering, their usability is limited, because of their exponentially increasing complexity. The SVD-based reduction gives a way for the elimination of the redundancy of these systems, and furthermore makes possible to construct several rule-bases or neural networks for the same problem with different complexity and accuracy, so wider use of soft computing methods becomes possible.

Keywords: real-time AI, fuzzy systems, neural networks, anytime systems, complexity reduction

1. INTRODUCTION

In modern system monitoring and diagnostic systems which are able to supervise complex industrial processes, and determine the appropriate actions in case of some failure or deviation from the optimal operation mode, several tasks - measurement, signal processing, fault detection and diagnosis, system control - must be carried out "on-line", with a guaranteed response-time. Often not only the available time, but the resources are also limited, so computing methods with lower complexity are needed.

Moreover, the amount of available time and resources could be not only limited, but could also change during the working of the system. Consequently, so called "anytime" systems are needed, which are able to provide short response time, and maintain the information processing even in case of temporary shortage of time or computational power (Várkonyi-Kóczy *et al.*, 2000).

In monitoring systems for the diagnosis the model of the faultless systems is used, and the evaluation of the model must be carried out on-line, so the model must be not only correct, but also computable with limited resources/time. Fuzzy or neural models could be well used for modeling and fault diagnosis, and they are usable even, when the exact mathematical description of the industrial process is not known, but several measured data and/or symbolic expert knowledge is available. Unfortunately fuzzy and neural models often has a high complexity, because the necessary number of antecedent fuzzy sets or neurons for the desirable accuracy could not be exactly determined.

This paper shows a possible method for the application of fuzzy inference systems and special neural networks in anytime systems. Section 2. summarizes the main expectations against tools, which could be used in anytime monitoring systems. Section 3. describes the basic principles of the Singular Value Decomposition

(SVD) based complexity reduction method, and its use for reducing fuzzy rule-bases. In Section 4. a short description of the generalized neural-network and its reduction could be found, while Section 5. raises some additional questions (error expansion, dynamic systems). Section 6. summarizes the results and proposes further research directions.

2. ANYTIME MONITORING SYSTEMS

In monitoring and diagnostic systems the computations must be carried out with a guaranteed response time, and the system must also be flexible in respect to the available input data, time and computational power. This flexibility makes able these systems to operate in changing circumstances, as for example the overloading of the system with a fault-diagnosis task, during which the evaluation of the model of the industrial process should be carried on. Naturally, while the information processing could be maintained, the complexity of some modules must be reduced, which in most cases leads to less accurate results (Zilberstein, 1996).

The group of usable algorithms/computing methods is limited by the following expectations:

- Guaranteed response time and accuracy: the accuracy of the results, and the needed computing time/resources must be known in advance.
- Low complexity: redundant calculations must be omitted.
- Changeable response time/computational need: to get a flexible system, the complexity of the used computing methods must be easily changeable.
- Known error: the error, originating from the non-exact complexity-reduction must also be known in advance, to find the optimal or at least acceptable solution during the given circumstances, and to compute the resultant error of the outputs.

Iterative algorithms are popular tools in anytime systems, because their complexity could be flexibly changed. These algorithms usually gives some - not accurate - result within a very short time, and more accurate results could be obtained, if the calculations are continued. The available time/resources need not be estimated in advance: the calculations could be continued until the results are needed, therefore always in the given conditions achievable most accurate results could be obtained.

However, the usability of iterative algorithms are limited, because in several cases there is not at all an effective iterative evaluation method. In other cases, there exists an iterative algorithm for the problem, but the accuracy of the results is not known in advance: the algorithm gives more and more accurate results, but it is not known, how much time is needed to achieve a given accuracy, or what will be the error, if the calculations are stopped at a given point.

Besides the iterative algorithms, a wide-range of other type of computing methods/algorithms could be used in anytime systems, as well, if a modular architecture is used (Fig. 1.). The flexibility of this method is lower, then in case of iterative algorithms, and it needs some extra planning and consideration, but could be used in cases, when an adequate iterative algorithm could not be found.

In this case, each module of the system, implementing a certain task, is realized in more ways: for the same task, there are more units, which have the same inputs, outputs and solve the same problem, but have different complexity and accuracy, etc. At a given time, in the knowledge of the current conditions (tasks to complete, achievable time/resources, needed accuracy, etc.), an expert system chooses the adequate configurations, i. e. the units, which will be used.

The expert system needs knowledge about the current conditions and about the units. Every unit could be described by several parameters:
- its place in the system: the realized module
- complexity: time and resource demand
- accuracy: absolute error(s) on the output(s), etc.

Soft computing methods have some advantages, which makes them very useful in system modeling, diagnosis and control. Neural networks could be used in every case, when enough measured data is available from the industrial process, even, if the exact mathematical formula of the process is not known, so the classical methods are not usable.

Fuzzy systems are able to handle inprecise data and inexactly formulated concepts, which could not be handled by classical tools. The representation of information is close to human thinking, so fuzzy systems are usually easy to survey, and knowledge, originated from human experts could be easily built into them. With the use of fuzzy-neural systems, the accuracy of fuzzy systems could further be improved by the use of sample data.

Unfortunately, the usability of these soft computing methods is limited because of their high computational complexity. There is not any universal method for the estimation of the necessary number of antecedent fuzzy sets or neurons to achieve the desired accuracy, so in practice, their numbers are usually overestimated, which yields to large and redundant rule-bases/neural networks.

If a method for the elimination of redundancy of rule-bases and neural networks could be found, which is also able for further complexity reduction considering the allowable error, the use of soft computing methods in anytime systems would become possible with the above described modular architecture.

3. SVD-BASED COMPLEXITY REDUCTION OF FUZZY INFERENCE SYSTEMS

Several complexity-reduction algorithm are known for fuzzy inference systems, but some of them could be used only for rule-bases with special properties or modify the inference algorithm. The SVD-based reduction algorithm could be used for a wide range of fuzzy systems (Product-Sum-Gravity and Takagi-Sugeno inference systems), and it changes only the rule-base, not the inference algorithm. Its main advantage is that it makes possible not only the automatic elimination of the redundancy of the rule-base, but also further, non-exact reduction considering the allowable error. Namely, in case of higher allowable error, the result will be a less complex fuzzy inference system, with a smaller rule-base. The complexity and inaccuracy of the resulted rule-bases could be always easily estimated.

3.1. SVD-based complexity reduction algorithm

The SVD-based complexity reduction algorithm is based on the decomposition of the F matrix, describing the mapping:

$$F_{(n_1 \times n_2)} = A_{1,(n_1 \times n_1)} B_{(n_1 \times n_2)} A^T_{2,(n_1 \times n_2)}, \qquad (1)$$

where A_k are orthogonal matrixes ($A_k A_k^T = E$), and B contains the λ_i singular values of F in decreasing order. The maximum number of the nonzero singular values is $n_{SVD} = \min(n_1, n_2)$. The singular values indicate the significance of the corresponding columns of A_k. Let be partitioned the matrixes in the following way (Yam, 1997):

$$A_k = \left| A^r_{k,(n_k \times n_r)} \quad A^d_{k,(n_k \times n_d)} \right| \text{ and}$$

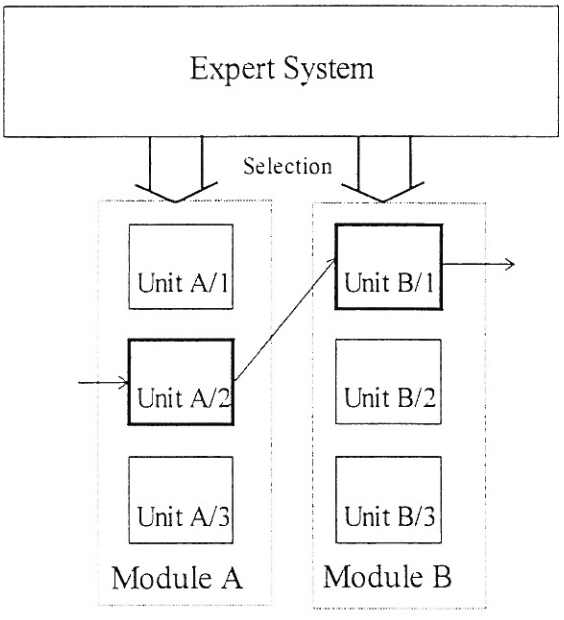

Fig. 1. Anytime system with a modular architecture

$$B_k = \begin{vmatrix} B^r_{(n_r \times n_r)} & 0 \\ 0 & B^d_{((n_1 - n_r) \times (n_2 - n_r))} \end{vmatrix},$$

where r denotes "reduced" and $n_r \le n_{SVD}$.

If B^d contains only zero singular values, then B^d and A^d_k could be dropped: $F = A^r_1 B^r A^{rT}_2$.

If B^d contains nonzero singular values, as well, then the $F' = A^r_1 B^r A^{rT}_2$ matrix is only an approximation of F, and the maximal difference between the values of F and F' (Yam *et al.*, 1999):

$$E_{RSVD} = |F - F'| \le \left(\sum_{i=n_r+1}^{n_{SVD}} \lambda_i \right) 1_{(n_1 \times n_2)}. \qquad (2)$$

3.2 Reduction of fuzzy rule-bases with SVD

Consider a so called PSGS (Product-Sum-Gravity-Singleton) fuzzy rule base with two inputs. The antecedent fuzzy sets are in Ruspini-partition and the consequence fuzzy sets are singletons:

$R_{i,j}$: If x_1 is $A_{1,i}$ and x_2 is $A_{2,j}$ then $y = y_{i,j}$.

During the inference, singleton fuzzification, sum S-norm and product T-norm are used. The result of the inference in case of the (x_1^*, x_2^*) input values:

$$y^* = \sum_{i_1, i_2} y_{i_1, i_2} \mu_{A_{1,i_1}}(x_1^*) \mu_{A_{2,i_2}}(x_2^*). \qquad (3)$$

Let F be a matrix, containing the $y_{i,j}$ elements, then apply the above described procedures to obtain $F \approx F' = A_1 B A_2^T$, where A_1 and A_2 are SN (Sum-Normalised: the sum of each row equals to one) and NN (Non-Negative).

Then the new rule-base will be:

$R'_{i,j}$: If x_1 is $A'_{1,i}$ and x_2 is $A'_{2,j}$ then $y = y'_{i,j}$,

where $i = 1 \ldots n_1^r$, $j = 1 \ldots n_2^r$, $y'_{i,j}$ are the elements of B, and the new membership functions could be obtained as:

$$\mu_{A'_{k,j}}(x_k) = \sum_j \mu_{A_{k,j}}(x_k) A_{k,j,i}. \qquad (4)$$

The reduced rule-base contains only n_r^2 rules instead of $n1*n2$ rules, and the error is the sum of the discarded singular values (Eq. (2)).

The method could be extended to n-dimension cases, as follows. In this case the reduction could be made in n steps, in every step one dimension of the F_1 matrix, containing the $y_{i_1, \ldots, i_n}$ consequences, is reduced. The i step will be:

1. Spreading out the n-dimensional F_l matrix (size: $n_1^r \times \ldots \times n_{l-1}^r \times n_l \times \ldots \times n_n$) into a two-dimensional S_l ($n_l \times (n_1^r * \ldots * n_{l-1}^r * n_{l+1} * \ldots * n_n)$) matrix. F_l contains the $y_{l_1 \ldots l_n}$ consequences, the following F_l-s will be generated by the algorithm.

2. Reduction of S_l: $S_l \approx A_l B A_l^T = A_l S_l^*$, where the size of A_l is $n_l \times n_l^r$ and the size of S_l^* is $n_l^r \times (n_1^r * \ldots * n_{l-1}^r * n_{l+1} * \ldots * n_n)$.

3. Re-stacking of S_l^* into the F_{l-1} n-dimensional matrix (size $n_1^r \times \ldots \times n_l^r \times n_{l-1} \times \ldots \times n_n$), and continuing with step 1. for F_{l-1}.

The consequences of the reduced rule-base will be the elements of F_n, and the new membership functions could be obtained according to Eq. (4).

The reduced rule base will contain only $n_1^r * \ldots * n_n^r$ rules, instead of $n_1 * \ldots * n_n$ and the maximum error of the reduction - at any point - will be less or equal then the sum of the discarded singular values.

There also exist extensions of the SVD based reduction to non-singleton (PSGN) (Baranyi and Yang, 1997) Takagi-Sugeno (Baranyi and Yam, 1997; Baranyi *et al.*, 1999a) and extremely large rule-bases, where the size of the rule-base is greater then the available operational memory (Baranyi *et al.*, 1999b).

4. Reduction of a Generalized Neural Network

The classical multilayer neural network could be generalised, if the non-linear transfer function is moved from the nodes into the links. It results in neurons, which apply only a sum operation to the input values, and links, which are characterised by a possibly non-linear weighting function, instead of a simple constant weight.

Let us focus on two neighbouring layers l and $l+1$ of a forward model. Let the neurons be denoted as $N_{l,i}$, $i = 1..n_l$ in layer l, where n_l is the number of neurons. Further, let input values of $N_{l,i}$ be $x_{l,i,k}$, $k = 1..n_{l-1}$ and its output $y_{l,i}$. The connection between layers l and $l+1$ can be defined by the $f_{l,j,i}(y_{l,i})$ weighting functions ($j = 1..n_{l+1}$). Thus

$$x_{l+1,j,i} = f_{l,j,i}(y_{l,i}) \qquad (6)$$

Therefore, the output of neuron $N_{l+1,j}$ will be

$$y_{l+1,j} = \sum_{i=1}^{n_l} f_{l,j,i}(y_{l,i}) . \qquad (7)$$

The weighting functions can also be changed during the training: the unknown weighting functions are approximated with linearly combined known functions, where only the linear combination must be trained. For this approximation the above described PSGS fuzzy systems could be used, with one input and one output:

$$y_{l+1,j} = \sum_{i=1}^{n_l} f_{l,j,i}(y_{l,i}) = \sum_{i=1}^{n_l} \sum_{t=1}^{m_{l,i}} \mu_{l,i,t}(y_{l,i}) b_{l,j,i,t} \qquad (8)$$

To reduce the size of a generalised neural network the SVD-based complexity reduction could be used. Equation (8) can always be transformed into the following form:

$$y_{l+1,j} = \sum_{z=1}^{n_{l-1}^r} a_{l,j,z} \sum_{i=1}^{n_l} \sum_{t=1}^{m_{l,i}^r} \mu_{l,i,t}^r(y_{l,i}) b_{l,z,i,t}^r \qquad (9)$$

where „r" denotes „reduced", further $n_{l+1}^r \le n_{l-1}$ and $\forall i : m_{l,i}^r \le m_{l,i}$.

The reduced form is represented as a neural network with an extra inner layer between layers l and $l-1$. Between the original layer l and the new layer the weighting functions are approximated from the reduced PSGS fuzzy systems, and layer $l-1$. simply computes the weighted sum ($a_{l,j,z}$) of the outputs of the new layer.

The reduction means the reduction of the $B = \left[b_{l,j,i,t}\right]$ three-dimensional matrix in two steps. In the first step, the first dimension is reduced, and the $a_{l,j,z}$ values are determined, in the second step the third dimension is reduced, and the new membership functions are determined. The detailed description of the algorithm could be found in (Baranyi *et al.*, 2000a,b).

Because of the new layer, in some cases the resulted neural network could have higher complexity, then the original. But the size of the new layer is usually considerably small, so the number of links - and with it the complexity - will be reduced. However the determination of the degree of complexity-reduction needs more considerations, then in case of fuzzy systems.

The maximal error of the resulted neural network could be computed from the discarded singular values, but the singular values, discarded in the first step counts n_l times (Takács and Nagy, 2000).

5. Error Expansion

The temporary reduction of complexity cause the reduction of accuracy, as well. While in case of SVD-based complexity reduction this error could be estimated, to obtain the so called resultant error (the error of the whole system) further computations must be made, considering the errors of the different modules, and the path of data and error through the modules.

For the calculation of the resultant error the error transfer functions of the modules must also be known. If module B uses the results of module A, then, if the accuracy of module A reduces, the accuracy of the outputs of B will reduce, as well. The $y=f_E(x)$ error transfer function means, that if the input of the module has an absolute error x, then the output of the module will have an additional absolute error y. The error along the data-path are cumulative. It is supposed, that the internal error of the modules, originating from inexact computations, noise, etc., could be modeled as an additive error component in the output of the module. Thus in the example above if the module A has an error E_A and the module B has an error E_B, then the resultant error on the output of B will be: $f_{e,B}(E_A)+E_B$, where $f_{e,B}$ is the error expansion function of module B.

The error transfer function could be determined in a given x_0 point:

$$f_E(x_e)=\max\{\max\{f(x)\,|\,x\in[x_0-x_e,x_0+x_e]\}-f(x_0),\ f(x_0)-\min\{f(x)\,|\,x\in[x_0-x_e,x_0+x_e]\}\} \quad (10)$$

where $f()$ is the transfer function of the given module/unit.

While this formula is accurate, it also is practically uncomputable, so simpler methods are needed. If $f()$ is monotone, then the error transfer function is:

$$f_E(x_e)=\max\{|f(x_0)-f(x_0-x_e)|,|f(x_0+x_e)-f(x_0)|\} \quad (11)$$

This formula still needs the calculation of $f()$ two times for every x_0.

If $f()$ is linear or nearly linear, the error transfer function could be estimated as:

$$f_E(x_e)\approx x_e * f'(x_0). \quad (12)$$

While this gives an easily computable estimation, it could cause the underestimation of the resultant error, if $f()$ is non-linear.

A certain overestimation of the resultant error could be got according to the followings:

$$f_E(x_e)\approx x_e * \max\{|f'(x)|\,|\,x\in[x_0-x_e,x_0+x_e]\} \quad (13)$$

While this formula also needs a lot of calculations, if $f'()$ has a global maximum, then the $f_E()$ function could be computed off-line, and it will be valid for the whole domain. To improve the accuracy of the estimation, the domain could be divided into more, overlapping intervals, and different $f_E()$ functions could be determined to the intervals.

With this last formula, only the derivative of the $f()$ must be known. In case of the above described fuzzy inference systems and generalized neural networks this could be easily computed.

In dynamic systems, further considerations must be made. In this case, the error could spread not only in space, but also in time in the system, namely, the temporary reduction of accuracy could effect the operation of the system even after the restoration of the original accuracy. If the system contains additive memory elements, then the error theoretically will never disappear from the system.

If the $f_E()$ error expansion function of the feed-back module is bounded as:

$$f_E(x_e)\le k * x_e, \quad (14)$$

then the error in the $k-1$. step could be estimated from the k. step (it is supposed, that the "original" accuracy was restored before the first step):

$$x_e^{k+1}\le a * x_e^k \le a^2 * x_e^{k-1} \le \ldots \le a^{k+1} * x_e^0. \quad (15)$$

This means, that if the absolute value of the $f_E()$ error expansion function is always less then one, then the error will sooner or later disappear from the system, but if its absolute value is greater or equal than one, then the effect of a temporal accuracy-reduction will influence all of the later results (Dorf, 1987).

This property of dynamic systems demands a great deal of precaution and planning from the engineers. The error expansion functions of the feed-back modules must always be estimated in advance (e. g. from the derivatives of the system), and if they do not have the wanted properties (i. e. they are not bounded), additional error-compensation methods must be used.

6. Conclusion

In modern diagnostic and monitoring systems the measurement, signal processing, evaluation of system model, fault diagnosis, and control must be carried out on-line. This demands the use of anytime techniques, which are able to provide a guaranteed response-time, and are flexible in respect to the available time and computational power.

Soft computing methods could be well used for systems modeling and control, even in cases, when classical tools are unusable, because the exact mathematical description is unavailable. However, their usability is limited because of their high complexity. There is not any universal method for the exact estimation of the necessary number of rules or neurons, so in the practice they are usually overestimated, which results in a huge and redundant rule base or neural network.

With the help of the SVD-based reduction not only the redundancy of the rule-bases or generalized neural networks could be eliminated, but further reduction could be made, considering the allowable error. This property of SVD-based reduction makes possible the use of fuzzy inference systems in anytime systems.

First a practically "accurate" fuzzy system or generalized neural network is constructed. For the determination of the rule-bases expert knowledge could also be used, but further improvement could be made with the use of training data and some learning algorithm. For neural networks the back-propagation learning algorithm could be used. This first rule-base or neural network could be large, because the redundancy later could be eliminated.

In the second step with the SVD-based complexity reduction algorithm a reduced, but accurate rule-base or neural network could be generated. To make possible the use of the rule-base/neural network in anytime systems, further variations of the rule-base/neural network must be constructed, with different accuracy and complexity. An alternative rule-base/neural network could be characterized by its complexity and its error (could be estimated from the sum of discarded singular values). The different rule-bases or neural networks could be different units, realizing a given module. The expert system could adaptively change the units, according to the temporarily available time and resources.

Because a complex system consists of several modules, the - temporal - errors of the modules could influence the accuracy of the outputs in several way. The resultant error could be estimated from the individual errors with the help of the error expansion functions of the modules.

In dynamic systems the temporal error could affect the output of the system even after the original accuracy were restored. This property of dynamic systems needs further considerations: the error expansion functions of the feed-backed modules must be strictly bounded, or additional compensation must be used. The reconfiguration of certain modules itself may induce additional problems (e. g. errors) in dynamic systems, because transients will appear after the parameter and/or structure adaptation. This transient behavior of the system also needs further research.

In complex systems, consisting several modules, it could be really hard to determine, which modules must be complexity-reduced to have a minimal resultant error, because both the errors and the error-expansion functions must be considered, and the error-expansion functions could also change during the complexity-reduction. The constructing of these expert systems needs further study.

REFERENCES

Baranyi P., Yang Y. (1997a). Singular value-based approximation with non-singleton support, *Seventh Int. IFSA World Congress*, Prague, June 25-29, 1997., pp.127-132.

Baranyi P., Y. Yam (1997b). Singular Value-Based Approximation with Takagi-Sugeno Type Fuzzy Rule Base, *IEEE Int. Conf. on Fuzzy Systems*, 1997., pp. 265-270.

Baranyi P., Y. Yam, C. T. Yang, A. Várkonyi-Kóczy (1999a). Complexity Reduction of a Rational General Form, *IEEE Int. Fuzzy Systems Conf.*, Aug. 22-25, 1999, Seoul, Korea, pp.366-371.

Baranyi P., Yam Y., Yang C. T., A. R. Várkonyi-Kóczy (1999b). Practical Extension of the SVD Based Reduction Technique for Extremely Large Fuzzy Rule Bases, *IEEE Int. Workshop on Intelligent Signal Processing*, 4-7. Sept., 1999, pp. 29-33.

Baranyi P., Y. Yam, .H. Hashimoto, P. Korondi, P. Michelberger (2000a). Approximation and Complexity Reduction of the Generalized Neural Network, accepted to *IEEE Trans. on Fuzzy Systems*.

Baranyi P., K.F. Lei, Y. Yam (2000b). Complexity Reduction of Singleton Based Neuro-fuzzy Algorithm, accepted in *IEEE Conf on System, Man and Cybernetics 2000*

Dorf R. C. (1987). *Modern Control Systems*, Addison-Wesley Publ. Comp., USA

Takács O., Nagy I. (2000), Error-bound of the SVD-based Neural Networks, *IFAC Symp. on Artificial Intelligence in Real Time Control*, 2-4., Oct., 2000, Budapest, Hungary

Yam Y. (1997). Fuzzy approximation via grid point sampling and singular value decomposition, *IEEE Trans. Syst., Man, Cybern.*, vol. **27**, pp. 933-951, Dec. 1997.

Yam Y., P. Baranyi, C. T. Yang (1999). Reduction of Fuzzy Rule Base Via Singular Value Decomposition, *IEEE Trans. on Fuzzy Systems*, Vol. 7., No. 2., Apr., 1999, pp.120-132.

Zilberstein S. (1996). Using Anytime Algorithms in Intelligent systems, *AI Magazine*, **17(3)**, pp. 73-83.

Várkonyi-Kóczy A. R., Kovácsházy T., Takács O., Benedecsik Cs. (2000). Anytime Algorithms in Intelligent Measurement and Control, *2000 World Automation Congress*, Maui, Hawaii, June 11-16., 2000