**MECHANIKA TEORETYCZNA
I STOSOWANA
Journal of Theoretical
and Applied Mechanics
4, 32, 1994**

INFORMATION FLOW BASED ALGORITHM OF MODELLING

WITOLD GRABYSZ

Engineering Mechanics Department

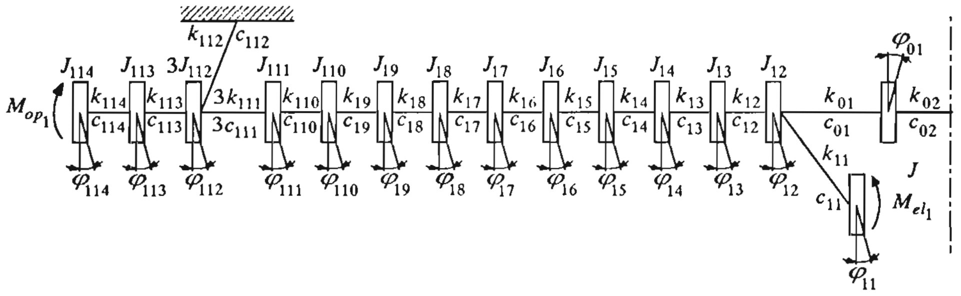
Silesian Technical University

A new concept of numerical modelling of discrete mechanical systems is presented in the paper. The method is based on a different organisation of information exchange between system elements than that commonly used. The idea was implemented by means of the object oriented programming. The way of overcoming the problem of numerical instability is discussed. Some examples are presented, where the values of eigenfrequencies obtained with the help of new method are compared with those obtained analytically.

1. Introduction

One of the methods of investigation of complex mechanical system dynamics is to consider such a system as a finite set of stiff masses, springs and dampers. Instead of an object with infinitely many degrees of freedom we get in this way a discrete model, being characterised by a finite number of degrees of freedom. An example of such a system is shown in Fig.1 (cf Świtoński et al. (1993)).

Formulation of the discrete model is only the first step on the way of investigation of dynamics. In order to obtain a desirable information (e.g. eigenfrequencies or compliances) we should assemble – basing on the model – differential equations describing dynamics of the system. We can apply the Newton's, Lagrange's or Hamilton's formalisms, respectively. However, a minor change made to the system structure (e.g. its development) can result in a fundamental change in the form of these equations. It is why such a procedure is very awkward as far as the investigation of influence of a system's structure on its output parameters is concerned. In order to overcome



k_{112}/c_{112}
J_{114} J_{113} $3J_{112}$ J_{111} J_{110} J_{19} J_{18} J_{17} J_{16} J_{15} J_{14} J_{13} J_{12}
M_{op_1}
k_{114} k_{113} $3k_{111}$ k_{110} k_{19} k_{18} k_{17} k_{16} k_{15} k_{14} k_{13} k_{12} k_{01} k_{02}
c_{114} c_{113} $3c_{111}$ c_{110} c_{19} c_{18} c_{17} c_{16} c_{15} c_{14} c_{13} c_{12} c_{01} c_{02}
φ_{114} φ_{113} φ_{112} φ_{111} φ_{110} φ_{19} φ_{18} φ_{17} φ_{16} φ_{15} φ_{14} φ_{13} φ_{12}
φ_{01}
k_{11} c_{11} J M_{el_1}
φ_{11}

Fig. 1. Dynamic model of a driving system of gearheads of the KGS-300 shearer

this problem some attempts of computer automation of differential equations assembly process for assumed models were made (cf Maißer et al. (1993); Otter et al. (1993)). The graph formalisms, particularly bond graphs, involve an interesting approach. There are computer programs able to generate the appropriate system of differential equations for a given model of a system (Głowacki et al. (1993)).

The third stage of investigation of dynamics is an analysis (solving) of the differential equations, often possible to carry out only numerically.

The author proposes a different scheme named the "information flow based algorithm of modelling". The name itself is perhaps not very fortunate, however it gives a "pronounceable" acronym: IFBAM. IFBAM is an algorithm which enables one to get numerically courses of displacements (velocities, forces) existing in the modelled, discrete system. A clever (in author's opinion) method for expressing interactions among system's elements is the kernel of the algorithm. Producing of the computer program taking advantage of the discussed algorithm would not be practically feasible without object oriented programming.

2. Basics of the object oriented programming

In object oriented programs – in contrast to traditional ones – related variables and procedures are merged into software packets called **objects**. An important feature of an object is that to a large extent it is separated from remaining components of the program. This feature – **encapsulation** – enables one to avoid undesirable interactions among them. It is evident, however, that in some situations the exchange of information must take place.

This exchange is realized by **messages**. The object sending a message is called a **sender**; the receiving object is called a **receiver**. In C++ – a popular object oriented programming language – a message takes the following form

```
NameOfReceiver->NameOfMessage(ParameterList);
```

As a matter of fact the name of a message is the name of receiver procedure the sender has the right to execute. This procedure in turn has the access to all (omitting some details) procedures and variables of its object.

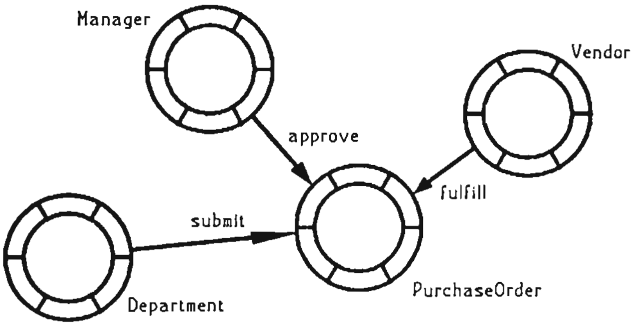


Fig. 2. Automation of office works implemented by means of object oriented programming (cf Taylor (1993))

Taylor (1993) presents the idea of **enterprise modelling**, i.e. formulating models of economic organisations. A particular organisation unit of the enterprise is represented in these models by an object. Relationships among the units are modelled by means of sending messages. The role of such models is to automatize many tasks of the organisation (mainly concerning the flow of documents). Objects called `Manager`, `Vendor`, `Department`, `PurchaseOrder` and messages called `submit`, `approve` and `fulfil` are shown in Fig.2. In order not to duplicate an object definition e.g. for a few purchase orders having the same structure, the concept of a **class** was introduced. A class is a template for creation of the objects (instances of the class). For example, `PurchaseOrder1`, `PurchaseOrder2`, `PurchaseOrder3` can be instances of class called `PurchaseOrder`. The base of IFBAM consists in an approach similar to the enterprise modelling.

3. The concept of IFBAM

Let us consider a simple mechanical system shown in Fig.3.

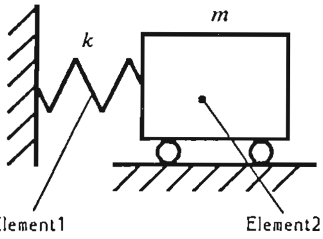


Fig. 3. System having one degree of freedom

Let us also define two classes: Spring and Mass as shown in Fig.4.

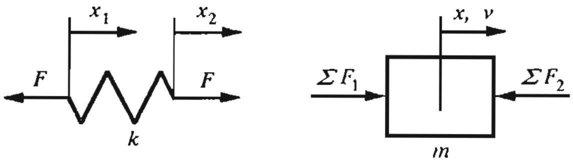


Fig. 4. Classes called Spring and Mass

Let Element1 be an instance of the Spring class and Element2 be an instance of the Mass class. The exchange of information between instances representing elements of the model is the heart of IFBAM. This exchange takes a form of "dialogue".

In the considered simple case, the dialogue looks as follows:

Element1:	"I act with the force $F = F^{(1)}$"
Element2:	"As a result of being treated by the force $F^{(1)}$ I have shifted myself in time τ to the final position $x = x^{(1)}$"
Element1:	"Because of the displacement of the right end to the position $x^{(1)}$ the force F has changed its value to $F^{(2)}$"
Element2:	"The action of the force $F^{(2)}$ results in my new position $x^{(2)}$"

etc.

A problem occurs when we have to deal with more complicated system, e.g. as in Fig.5.

For example, after Element3 hands over the message on the force value to Element2, should the latter answer first to Element3 or rather should it send the appropriate message to Element1? In order to solve this problem we will take advantage of concept used in construction of flip-flop circuits, namely we will divide the cycle of the IFBAM algorithm into two stages:

- Stage called **master**, during which only the exchange of information is carried out
- Stage called **slave**, during which particular elements calculate their output parameters to be handed over in the master stage of the next cycle.

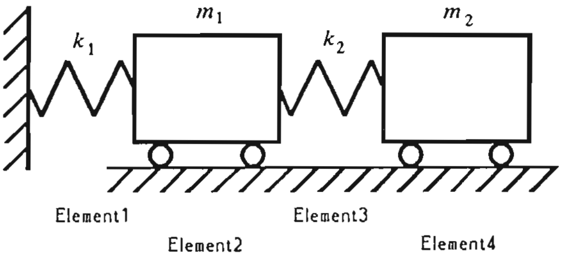


Fig. 5. System with two degrees of freedom

For the sake of unification let us define in the `Spring` class as well as in the `Mass` class three functions: `MasterFromLeft`, `MasterFromRight` and `Slave`. Functions called `MasterFromLeft` and `MasterFromRight` transmit the appropriate messages to the object located on the left and the right sides of the sender, respectively. The identifier of the object which the information should be sent to is a parameter of these functions. The `Slave` function makes the suitable calculations. A fragment of the C++ program being an implementation of the IFBAM idea for the case shown in Fig.3 can be written as[1]:

```
Element1->MasterFromLeft(Element2);
Element2->MasterFromRight(Element1);

Element1->Slave();
Element2->Slave();
```

Repetition of the fragment listed above in a loop in connection with monitoring of the required variable (e.g. displacement of the mass) allows one to obtain a course of this variable. Of course it is necessary to assume a time step τ required in every numerical integration.

[1]Somewhere earlier in the program the objects `Element1`, `Element2` must be declared and initialised

4. Problems with numerical stability

Let us have a closer look at `Slave` functions. In order to calculate velocity we will use the central difference method

$$v^{i+1} = v^{i-1} + 2\frac{F_1 - F_2}{m}\tau \tag{4.1}$$

and the displacement in the next step we will approximate by the backward difference

$$x^{i+1} = x^i + v^{i+1}\tau \tag{4.2}$$

Superscripts denote a moment of time. As far as the function called `Slave` of the `Spring` class is concerned, we will calculate force at every step according to the following formula

$$F = (x_1 - x_2)k \tag{4.3}$$

Tracing displacement of the mass in the system shown in Fig.3 we obtain the course shown in Fig.6.

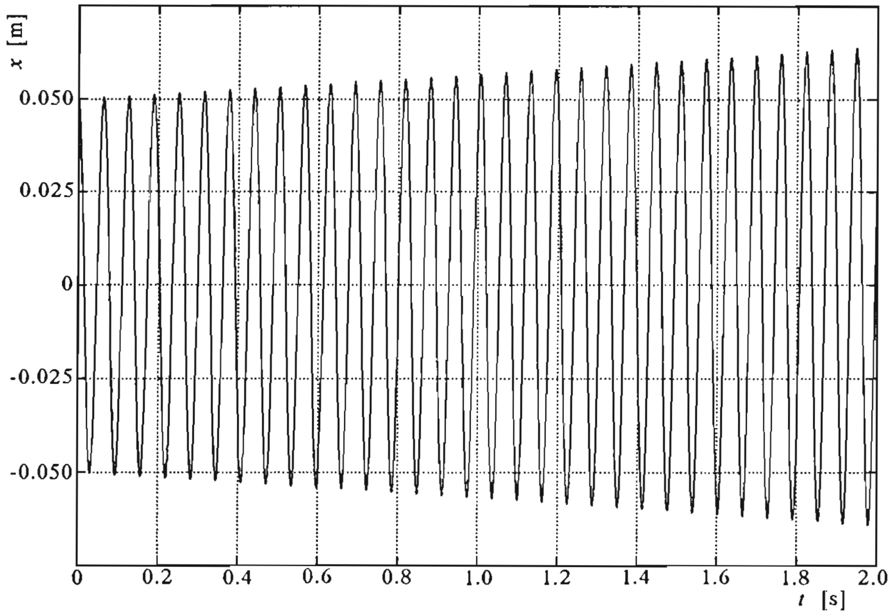


Fig. 6. Original course of displacement

It is noticeable, that amplitude is increasing with time which should not take place. Let us apply the improvement based on the fact that the function should return mean value of force in the interval $\langle t, t+\tau\rangle$, namely

$$F = \frac{\int\limits_{\Delta^i}^{\Delta^{i+1}} f(\Delta)\, d\Delta}{\Delta^{i+1} - \Delta^i} \tag{4.4}$$

where Δ denotes compressing (extending) of the spring and $f(\Delta)$ is the characteristic of the spring. On the assumption of the linear characteristic we get

$$F = \frac{k}{2}(\Delta^{i+1} + \Delta^i) \tag{4.5}$$

so it is necessary to estimate the values of displacement at the next step. We will use the formulae

$$\begin{aligned} \tilde{x}_1^{i+1} &= 2x_1^i - x_1^{i-1} \\ \tilde{x}_2^{i+1} &= 2x_2^i - x_2^{i-1} \end{aligned} \tag{4.6}$$

Then Eq (4.3) can be rewritten as

$$F = \frac{k}{2}(x_1^i + \tilde{x}_1^{i+1} - x_2^i - \tilde{x}_2^{i+1}) \tag{4.7}$$

After these changes we obtain the course shown in Fig.7. Notice that now the course is stable.

In the calculations we have assumed the following values of system's parameters

$$k = 10\,000 \frac{\mathrm{N}}{\mathrm{m}} \qquad m = 1\mathrm{kg}$$

It means that

$$T = 2\pi\sqrt{\frac{m}{k}} = 2\pi \cdot 10^{-2}\mathrm{s}$$

In the table below results for different time steps are presented.

τ [s]	$5T$ [s]
0.000025	0.3141125
0.00005	0.314075
0.0001	0.314
0.0002	0.3138
0.0004	0.3134
0.001	0.312

In the following we will assume $\tau = 0.00005$ s.

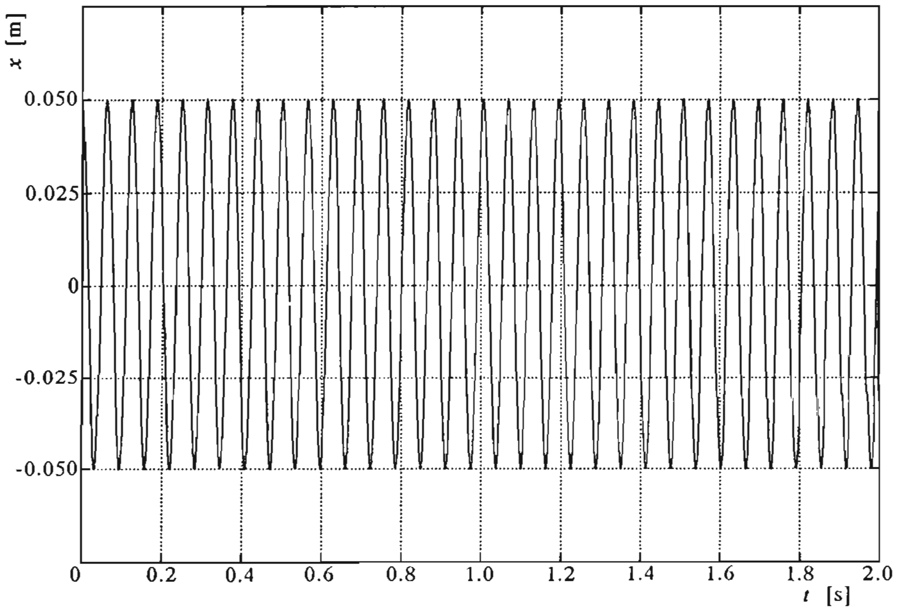


Fig. 7. Improved course of displacement

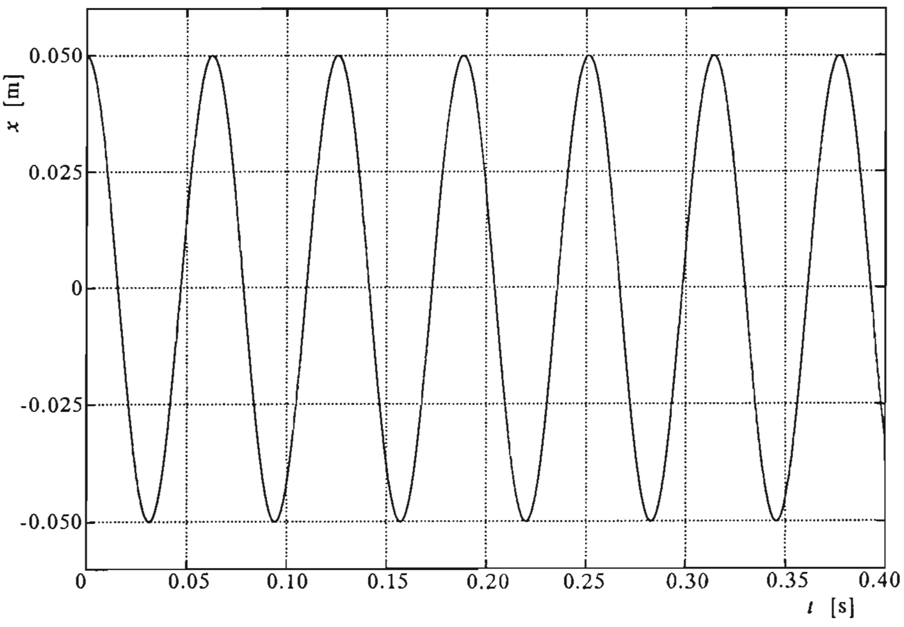


Fig. 8. Zoomed course of displacement

5. Examples calculations

In the previous paragraph it was demonstrated that the IFBAM works properly in the case of vibration of a simple system with one degree of freedom. Let us try now to find the eigenfrequencies of the system shown in Fig.5. A program implementing the idea of IFBAM, which fragment was shown above, must be expanded as follows:

```
Element1->MasterFromLeft(Element2);
Element2->MasterFromRight(Element1);

Element2->MasterFromLeft(Element3);
Element3->MasterFromRight(Element2);
Element3->MasterFromLeft(Element4);
Element4->MasterFromRight(Element3);

Element1->Slave();
⋮
Element4->Slave();
```

It must be admitted that the modifications which had to be done were very simple. After having made this modifications to the program as well as after having assumed the values of parameters

$$k_1 = 10\,000\frac{\mathrm{N}}{\mathrm{m}} \qquad m_1 = 1\mathrm{kg}$$

$$k_2 = 40\,000\frac{\mathrm{N}}{\mathrm{m}} \qquad m_2 = 1\mathrm{kg}$$

we get courses of required variables. Let us perform the Fast Fourier Transform of the displacement of mass m_1. The obtained spectrum – solid line in Fig.9 – reveals two maxima. They correspond to the eigenfrequencies. Their values are the same as the values obtained analytically

$$f_1 = 10.9\mathrm{Hz} \qquad f_2 = 46.5\mathrm{Hz}$$

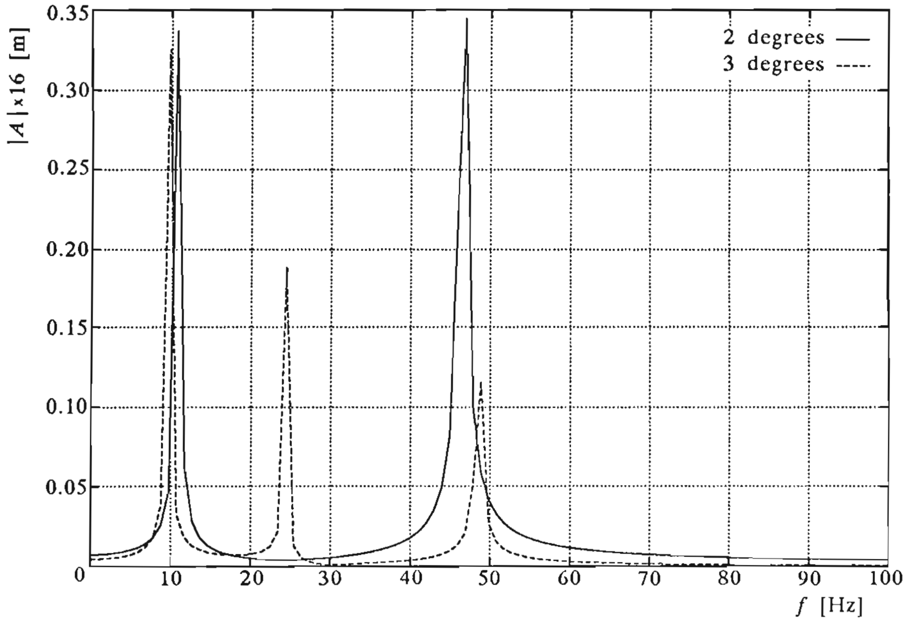


Fig. 9. Spectra of displacement

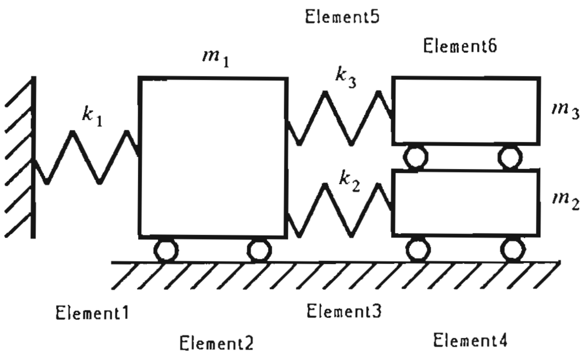


Fig. 10. System having three degrees of freedom

The dashed line in Fig.9 represents the spectrum of displacement of mass m_3 of the system with three degree of freedom shown in Fig.10, where

$$k_1 = 10\,000\frac{\mathrm{N}}{\mathrm{m}} \qquad m_1 = 1\mathrm{kg}$$

$$k_2 = 40\,000\frac{\mathrm{N}}{\mathrm{m}} \qquad m_2 = 1\mathrm{kg}$$

$$k_3 = 10\,000\frac{\mathrm{N}}{\mathrm{m}} \qquad m_3 = 0.5\mathrm{kg}$$

The appropriate fragment of the program takes the form

```
Element1->MasterFromLeft(Element2);
Element2->MasterFromRight(Element1);

Element2->MasterFromLeft(Element3);
Element3->MasterFromRight(Element2);
Element3->MasterFromLeft(Element4);
Element4->MasterFromRight(Element3);

Element2->MasterFromLeft(Element5);
Element5->MasterFromRight(Element2);
Element5->MasterFromLeft(Element6);
Element6->MasterFromRight(Element5);

Element1->Slave();
:
Element6->Slave();
```

The eigenfrequencies obtained analytically have the values

$$f_1 = 9.66\mathrm{Hz} \qquad f_2 = 24.3\mathrm{Hz} \qquad f_3 = 48.5\mathrm{Hz}$$

Tracing displacement and velocity of mass m_3 allows to plot a phase portrait shown in Fig.11. The portrait is presented for its aesthetic rather than scientific values.

6. Conclusions

Only linear vibration was the subject of the work, but some results obtained indicate usefulness of the discussed algorithm for non-linear vibration as

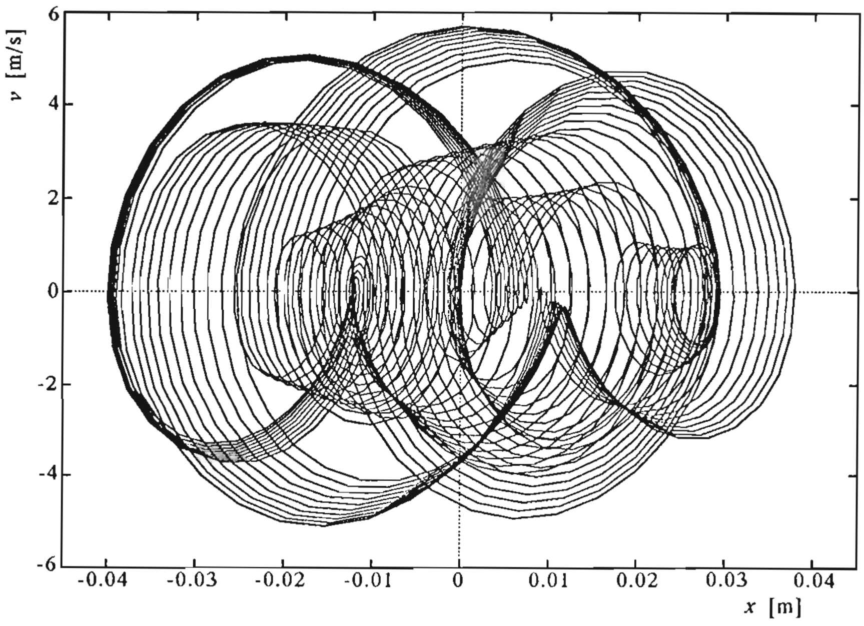


Fig. 11. Phase portrait

well. It probably will be possible to apply the IFBAM to investigate systems not only mechanical but also electrical systems as well as to investigate hybrid (mixed) systems.

The main advantage of the algorithm which should be emphasised is the fact that changes made to the structure of an analysed system are easily translated to the computer program. It makes the IFBAM very suitable for the analysis of influence of the structure on the output parameters of the system.

References

1. GLOWACKI J., GRABOWIECKI K., RYBICKI A., 1993, The Graphic Editor of the Bond Graph, *Proc. of the 1st Conf. "Graphs & Mechanics"*, Gliwice-Ustroń, 65-75

2. MAISSER P. ET AL., 1993, Alaska. Demonstration program manual, Institute of Mechatronics, Chemnitz

3. OTTER M., ELMQVIST H., CELLIER F.E., 1993, Modeling of Multibody Systems with the Object-Oriented Modeling Language Dymola, *NATO-Advanced*

Study Institute, Proc. of the Conf. "Computer aided analysis of rigid and flexible mechanical systems", II, 91-110, Tróia, Portugal

4. ŚWITOŃSKI E., DUBIEL D., RAK Z., 1993, Analysis of the Influence of Plays on Magnitudes of Dynamic Forces in Kinematic Pairs of a Driving System of Gearheads of a Coal Shearer, *Zeszyty Naukowe Pol.Śl., Mechanika,* 113, 407-413, Gliwice
5. TAYLOR D. A., 1993, *Object Oriented Technology: a Manager's Guide,* Addison-Wesley, Reading, MA

Algorytm modelowania oparty na przepływie informacji

Streszczenie

W pracy przedstawiona jest idea nowego sposobu numerycznego modelowania dyskretnych układów mechanicznych. Polega ona na odmiennej – od dotychczas stosowanych – organizacji wymiany informacji pomiędzy elementami układu. Zastosowanym narzędziem jest programowanie obiektowe. Omówiono sposób w jaki przezwyciężono problem niestabilności numerycznej. Przedstawiono przykłady, dla których porównano wartości częstości własnych uzyskanych nową metodą z wartościami analitycznymi.

Manuscript received February 21, 1994; accepted for print May 5, 1994