Piotr Breitkopf

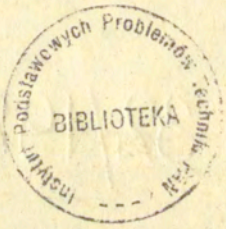
EMENTY SZTUCZNEJ INTELIGENCJI W NUMERYCZNEJ ANALIZIE KONSTRUKCJI

WARSZAWA 1987

ISSN 0208-5658

P r a c a d o k t o r s k a

Praca wpłynęła do Redakcji dnia 25 września 1987 r.



Instytut Podstawowych Problemów Techniki PAN

Nakład 170 egz. Ark.wyd.8,55 Ark.druk.12
Oddano do drukarni w październiku 1987 r.
Nr zamówienia 565/87.

Warszawska Drukarnia Naukowa, Warszawa,
ul. Śniadeckich 8

Piotr Breitkopf
Zakład Teorii Konstrukcji
IPPT PAN

ELEMENTY SZTUCZNEJ INTELIGENCJI W NUMERYCZNEJ ANALIZIE KONSTRUKCJI

Wstęp: zakres i cel pracy

Zastosowanie komputerów do numerycznej analizy zagadnień mechaniki jest obecnie tak powszechne, że trudno sobie wyobrazić praktyczne rozwiązanie złożonego zadania bez ich użycia. W miarę wzrostu stopnia złożoności zarówno rozpatrywanych zagadnień jak i wymaganych do ich rozwiązania algorytmów, coraz silniej dają się jednak odczuć ograniczenia tradycyjnie pojętych metod komputerowych, które nie tylko nie wyczerpują możliwości zastosowania komputera lecz także wymagają coraz większego zasobu wiedzy i doświadczenia od użytkownika. Równolegle z metodami numerycznymi rozwijała się inna dziedzina informatyki - sztuczna inteligencja, która wytworzyła jakościowo odmienne metody programowania mające na celu zwiększenie użyteczności komputerów, wykraczające poza zakres możliwy do osiągnięcia tradycyjnymi, algorytmicznymi metodami. Nowe metody, które możemy określić ogólnie mianem metod symbolicznych mają już pierwsze sukcesy w rozwiązywaniu szeregu zagadnień, zaś ich dotychczasowe ograniczenia związane z wymaganiami odnośnie zasobów komputerowych stają się w związku z niesłabnącym tempem rozwoju sprzętu komputerowego, coraz mniej istotne. Wydaje się rzeczą celową i na czasie podjęcie próby zastosowania nowych technik programowania w numerycznej analizie konstrukcji. Praca niniejsza jest próbą skonkretyzowania możliwości zastosowania tych technik poprzez zakreślenie ogólnych możliwości i realizację szczegółowych koncepcji. Ze względu na stwierdzony w trakcie realizacji obenej pracy potencjał nowych metod, praca ma wstępny charakter i powinna być kontynuowana w przyszłości. O kierunkach tego rozwoju wspominamy w podsumowaniu pracy.

ROZDZIAŁ 1

TRADYCYJNY SYSTEM METODY ELEMENTÓW SKOŃCZONYCH I

Spośród wielu metod numerycznej analizy konstrukcji, największą popularnością cieszy się metoda elementów skończonych (MES). Wynika to z faktu jednolitego podejścia do całej gamy zagadnień spotykanych w praktyce inżynierskiej i badawczej, co pozwala na stosowanie standardowego oprogramowania. MES jest na tyle ogólna, że niektórzy autorzy [106] postulują koncepcję tzw. uogólnionej MES zawierającej inne metody aproksymacyjne, takie jak metoda różnic skończonych (MRS) oraz metoda elementów brzegowych (BEM - od ang. Boundary Element Method) jako szczególne przypadki.

Obszarem zastosowania tradycyjnego systemu MES jest etap analizy następujący po przyjęciu przez użytkownika modelu fizycznego rozpatrywanego procesu mechanicznego. Rola takiego systemu kończy się w momencie uzyskania wyników obliczeń. Zadaniem użytkownika jest ocena tych wyników pod kątem zgodności z przyjętym modelem i podjęcie decyzji o ewentualnym ponowieniu analizy ze zmodyfikowanymi danymi.

Pierwszym etapem rozwiązania zadania tradycyjnym systemem MES jest dyskretyzacja obszaru za pomocą siatki elementów skończonych. Zakładając, że układ znajduje się w równowadze dynamicznej w chwili t, możemy zapisać równania ruchu węzłów siatki w chwili $t + \Delta t$

$$\mathbf{M}\ddot{\mathbf{r}}^{(t+\Delta t)} + \mathbf{C}\dot{\mathbf{r}}^{(t+\Delta t)} + \mathbf{K}_T \Delta \mathbf{r}^{(t,t+\Delta t)} = \mathbf{R}^{(t+\Delta t)} - \mathbf{F}^t \tag{1.1}$$

gdzie

$\mathbf{M}$ – macierz mas,

$\mathbf{C}$ – macierz tłumienia,

$\mathbf{K}_T$ – macierz sztywności stycznej,

$\mathbf{R}^{(t+\Delta t)}$ – wektor sił zewnętrznych w chwili $t + \Delta t$,

$\mathbf{F}^t$ – wektor uogólnionych sił węzłowych odpowiadających stanowi naprężenia w chwili t. t,

$\Delta \mathbf{r}^{(t,t+\Delta t)}$ – przyrost wektora przemieszczeń w przedziale $(t, t + \Delta t)$,

$\dot{\mathbf{r}}^{(t+\Delta t)}$ – wektor prędkości,

$\ddot{\mathbf{r}}^{(t+\Delta t)}$ – wektor przyspieszeń.

1.1 nieliniowa statyka

Pominięcie w równaniu (1.1) efektów inercyjnych oraz efektów tłumienia prowadzi do przyrostowego równania statyki nieliniowej

$$\mathbf{K}_T \Delta \mathbf{r}^{(t,t+\Delta t)} = \mathbf{R}^{(t+\Delta t)} - \mathbf{F}^t \tag{1.2}$$

W zagadnieniach statyki czas pełni rolę parametru obciążenia, w związku z czym możemy go zastąpić numerem kroku przyrostowego

$$\mathbf{K}_T \Delta \mathbf{r}^{(i,i+1)} = \mathbf{R}^{i+1} - \mathbf{F}^i(\mathbf{r}^i) \tag{1.3}$$

W związku z nieliniowym charakterem powyższego związku, zachodzi konieczność iteracyjnego poprawiania uzyskanego rozwiązania $\Delta \mathbf{r}^{(i+1)}$. Podstawowym algorytmem iteracyjnego powrotu na ścieżkę równowagi jest tzw. algorytm Newtona-Raphsona (N-R), który można wyprowadzić rozwijając zależność (1.2) w szereg Taylora wokół punktu $\Delta \mathbf{r}^{(i+1)}$. Kolejne przybliżenia wektora przyrostu przemieszczenia oznaczamy $\Delta \mathbf{r}^{(i+1|k)}$, i uzyskujemy metodą akumulacji poprawek $\delta \mathbf{r}^{(i+1|k)}$

$$\Delta \mathbf{r}^{(i+1|k)} = \Delta \mathbf{r}^{(i+1|k-1)} + \delta \mathbf{r}^{(i+1|k)} \tag{1.4}$$

gdzie i jest jak poprzednio numerem kroku, zaś k numerem iteracji. Poprawki uzyskujemy rozwiązując układ równań algebraicznych

$$\mathbf{K}_T{}^{(i+1|k-1)} \delta \mathbf{r}^{(i+1|k)} = \mathbf{R}^{(i+1)} - \mathbf{F}^{(i+1)}(\mathbf{r}^{(i+1|k-1)}) \tag{1.5}$$

gdzie $\mathbf{K}_T{}^{(i+1|k-1)}$ jest zgodnie z przyjętą konwencją oznaczeń, $k-1$-ym przybliżeniem macierzy stycznej na $i+1$-ym kroku. Kryterium zakończenia iteracji przyjmuje się często w postaci

$$\frac{\|\delta \mathbf{r}^{(i+1|k)}\|}{\|\mathbf{r}^{(i+1|k)}\|} < \varepsilon \tag{1.6}$$

gdzie ε jest żądaną tolerancją rozwiązania, zaś $\|.\|$ pewną normą wektorową. Przejście do kolejnego kroku wymaga jeszcze aktualizacji wektora przemieszczeń

$$\mathbf{r}^{(i+1|k)} = \mathbf{r}^{(i+1|k-1)} + \delta\mathbf{r}^{(i+1|k)} = \mathbf{r}^{i} + \Delta\mathbf{r}^{(i+1|k)} \tag{1.7}$$

Praktyczna przydatność algorytmu N-R jest ograniczona z następujących względów:

- czasochłonnego etapu linearyzacji i rozwiązywania układu równań przy każdej iteracji

- przyrosty obciążenia dobiera się metodą prób i błędów

- przechodzenie przez punkty maximum obciążenia wymaga zmian kierunku przykładanego wektora przyrostu obciążenia

- stała wartość przyrostu obciążenia na kroku może nie prowadzić do zbieżności

- przy silnych zmianach sztywności układu, pierwsze przybliżenie wektora przyrosttu przemieszczeń może leżeć daleko od ścieżki równowagi

- ze względu na kumulację stanów nie wpełni zrównoważonych wynik może zależeć od drogi obciążenia.

Zmodyfikowanymi algorytmami N-R nazywamy te algorytmy, które polegają na ograniczeniu częstotliwości aktualizacji macierzy stycznej. W tradycyjnej, pełnej metodzie N-R aktualizacja ta ma miejsce przy każdym kroku obciążenia i przy każdej iteracji. Spośród nieograniczonej liczby metod zmodyfikowanych możliwych do uzyskania przez zastosowanie różnych kryteriów aktualizacji macierzy stycznej, wyróżnia się zwyczajowo dwie: metodę początkowej i stałej sztywności.

W metodzie początkowej sztywności zwanej też metodą początkowych naprężeń, do obliczenia wszystkich przyrostów wektora przemieszczeń używana jest początkowa macierz sztywności $\mathbf{K}_T^0$.

$$\mathbf{K}_T^{(0)}\delta\mathbf{r}^{(i+1|k)} = \mathbf{R}^{(i+1)} - \mathbf{F}^{(i+1)}(\mathbf{r}^{(i+1|k-1)}) \tag{1.8}$$

W metodzie stałej sztywności macierz styczna jest aktualizowana na początku kolejnego kroku i pozostawiana stała w trakcie iteracji.

$$\mathbf{K}_T^{(i+1|0)}\delta\mathbf{r}^{(i+1|k)} = \mathbf{R}^{(i+1)} - \mathbf{F}^{(i+1)}(\mathbf{r}^{(i+1|k-1)}) \tag{1.9}$$

W metodach Newtonowskich kierunek poszukiwania poprawionego rozwiązania jest kierunkiem stycznym do ścieżki równowagi. Metody quasi-newtonowskie oparte są na zastąpieniu warunku stycznej warunkiem siecznej. Macierz sieczną dla iteracji $k+1$ oblicza się przez dodanie do macierzy siecznej $\dot{\mathbf{K}}^{(i|k)}$ z poprzedniego kroku iteracyjnego (v pierwszej iteracji możemy posłużyć się macierzą styczną $\mathbf{K}_T^0$), pewnej macierzy rędu pierwszego tak dobranej by jej wektor własny u był równoległy do wektora sił nezrównoważonych. Korzyść z zastosowania metody polega na możliwości wykorzystana wzorów analitycznych, za pomocą których możemy zaktualizować macierz odwrotną. Jżeli poprawka do macierzy jest dana przez

$$\dot{\mathbf{K}}^{(i|k+1)} = \dot{\mathbf{K}}^{(i|k)} + G\mathbf{u}\mathbf{u}^T \tag{1.10}$$

t macierz odwrotna do macierzy poprawionej dana jest wzorem Shermana-Morrisona

$$\dot{\mathbf{K}}^{(i|k+1)^{-1}} = \dot{\mathbf{K}}^{(i|k)^{-1}} - \frac{\dot{\mathbf{K}}^{(i|k)^{-1}}\mathbf{u}\mathbf{u}^T\dot{\mathbf{K}}^{(i|k)^{-1}}}{\frac{1}{G} + \mathbf{u}^T\dot{\mathbf{K}}^{(i|k)^{-1}}\mathbf{u}} \tag{1.11}$$

Omijamy w ten sposób zasadniczą niedogodność algorytmu N-R polegającą na potrzebie każdorazowego składania i odwracania macierzy stycznej.

Modyfikacja n-tego rzędu macierzy sztywności polega na dodaniu do macierzy stycznej obliczonej na początku kroku, pewnej macierzy rzędu n-tego (macierzy o n niezerowych wartościach własnych). Modyfikacja pierwszego rzędu jest dana przez:

$$\dot{\mathbf{K}}^{(i|k+1)} = \dot{\mathbf{K}}^{(i|k)} + \frac{(\gamma^{(i|k)} - \dot{\mathbf{K}}^{(i|k)}\delta^{(i|k)})(\gamma^{(i|k)} - \dot{\mathbf{K}}^{(i|k)}\delta^{(i|k)})^T}{\delta^{(i|k)^T}(\gamma^{(i|k)} - \dot{\mathbf{K}}^{(i|k)}\delta^{(i|k)})} \tag{1.12}$$

przy czym za kierunek modyfikacji przyjmuje się kierunek siły niezrównoważonej z z ostatniej iteracji. Zmodyfikowana macierz odwrotna, uwzględniając (1.11) ma postać

$$\hat{K}^{(i|k+1)^{-1}} = \hat{K}^{(i|k)^{-1}} + \frac{(\delta^{(i|k)} - \hat{K}^{(i|k)^{-1}}\gamma^{(i|k)})(\delta^{(i|k)} - \hat{K}^{(i|k)^{-1}}\gamma^{(i|k)})^T}{\gamma^{(i|k)^T}(\delta^{(i|k)} - \hat{K}^{(i|k)^{-1}}\gamma^{(i|k)})} \qquad (1.13\beta)$$

Jak widać, modyfikacja pierwszego rzędu może zawieść, gdy wektory $\gamma^{(i|k)}$ oraz $(\delta^{(i|k)} - \hat{K}^{(i|k)^{-1}})$ będą prawie ortogonalne. Z tego względu w praktyce stosujemy ko-orekcję rzędu wyższego, z reguły drugiego.

Poprawka rzędu drugiego do macierzy sztywności jest dana wzorem DFP (od nazwisk Davidon-Fletcher-Powell)

$$\hat{K}^{(i|k+1)} = \left(I - \frac{\gamma^{(i|k)}\delta^{(i|k)^T}}{\gamma^{(i|k)^T}\delta^{(i|k)}}\right)\hat{K}^{(i|k)}\left(I - \frac{\delta^{(i|k)}\gamma^{(i|k)^T}}{\gamma^{(i|k)^T}\delta^{(i|k)}}\right) + \frac{\gamma^{(i|k)}\gamma^{(i|k)^T}}{\delta^{(i|k)^T}\gamma} \qquad (1.14\delta)$$

lub BFGS (od nazwisk Broyden-Fletcher-Goldfarb-Shanno)

$$\hat{K}^{(i|k+1)} = \hat{K} + \frac{\gamma^{(i|k)}\gamma^{(i|k)^T}}{\delta^{(i|k)^T}\gamma^{(i|k)}} - \frac{\hat{K}^{(i|k)}\delta^{(i|k)}\delta^{(i|k)^T}\hat{K}^{(i|k)}}{\delta^{(i|k)^T}\hat{K}\delta^{(i|k)}} \qquad (1.15\delta)$$

gdzie pierwszy człon odpowiada zmianie sztywności w kierunku siły niezrównoważonej j z ostatniej iteracji, zaś drugi człon zmianie sztywności w kierunku siły niezrównoważonej j z przedostatniej iteracji. Macierze odwrotne do tak zmodyfikowanych macierzy sztywności otrzymujemy przez dwukrotne zastosowanie (1.11) odpowiednio dla metody DFP

$$\hat{K}^{(i|k+1)^{-1}} = \hat{K}^{(i|k)^{-1}} + \frac{\delta^{(i|k)}\delta^{(i|k)^T}}{\delta^{(i|k)^T}\gamma^{(i|k)}} - \frac{\hat{K}^{(i|k)^{-1}}\gamma^{(i|k)}\gamma^{(i|k)^T}(\hat{K}^{(i|k)^{-1}})^T}{\gamma^{(i|k)^T}\hat{K}^{(i|k)^{-1}}\gamma^{(i|k)}} \qquad (1.16\delta)$$

oraz BFGS

$$K^{(i|k+1)^{-1}} = \left(I - \frac{\delta^{(i|k)}\gamma^{(i|k)^T}}{\delta^{(i|k)^T}\gamma^{(i|k)}}\right)\hat{K}^{(i|k)^{-1}}\left(I - \frac{\gamma^{(i|k)}\delta^{(i|k)^T}}{\delta^{(i|k)^T}\gamma^{(i|k)}}\right) + \frac{\delta^{(i|k)}\delta^{(i|k)^T}}{\delta^{(i|k)^T}\gamma^{(i|k)}} \qquad (1.17\delta)$$

Dalsze warianty wyznaczania macierzy siecznej można znaleźć w pracach [12, 41, 42, 61, 73] W związku z tym, że sprawność poszczególnych algorytmów zależy od różnych czynników, między innymi od liczby stopni swobody, nie udało się dotychczas ustalić najepszej metody.

1.1.1 skalowanie długości kroku obciążenia

współczynnik sztywności

Zdefiniujmy bieżący parametr sztywności S_p [15]

$$S_p = \frac{\|\Delta \mathbf{R}^{(i,i+1)}\|^2}{\Delta \mathbf{r}^{(i,i+1)^T} \Delta \mathbf{R}^{(i,i+1)}} \tag{1.18}$$

o następujących własnościach:

- początkowo S_p jest równy jedności
- dla lokalnych ekstremów obciążenia S_p wynosi zero
- S_p jest dodatni dla układów umacniających się
- S_p jest ujemny dla układów słabnących
- prędkość zmian S_p ma związek z nieliniowością

Jeżeli proces linearyzacji rozumiemy jako przybliżenie otrzymane w wyniku rozwinięcia pełnej nieliniowej zależności w szereg Taylora, to błąd zaokrąglenia na kolejnym kroku jest zależny od przyrostu obciążenia i miary drugiej pochodnej wektora przemieszczenia liczonej względem obciążenia

$$\delta = c\alpha\|\mathbf{F}\| \tag{1.19}$$

gdzie

δ –błąd zaokrąglenia

c -stała

α -przyrost parametru obciążenia

Ponieważ S_p jest aproksymacją pierwszej pochodnej przemieszczeń liczonej względem obciążenia, to aproksymacji drugiej pochodnej będzie odpowiadać iloraz S_p przezz przyrost parametru α. Przyrost parametru obciążenia będziemy mogli wyrazić teraz wv funkcji przyrostu na poprzednim kroku, następującym wzorem

$$\alpha^{(i+1)} = \alpha^i c / \Delta S_p^i \qquad (1.20))$$

gdzie

$\alpha^{(i+1)}$ -parametr obciążenia na kroku $i+1$,

ΔS_p^i -zmiana S_p na kroku i.

Metoda ta prowadzi do małych przyrostów w przypadku silnej nieliniowości i do dużych kroków w obszarach prawie liniowych.

skalowanie oparte na liczbie iteracji

Z wyjątkiem pierwszego kroku, dla którego wielkość przyrostu parametru sterowania zakładamy arbitralnie, długość każdego kolejnego kroku jest dobierana wg wzoru:

$$\alpha^{(i+1)} = \alpha^i I / I^i \qquad (1.21))$$

gdzie

I^i - liczba iteracji, która była potrzebna do zbieżności na kroku i,

α^i - przyrost parametru sterowania na kroku i,

I - przewidywana liczba iteracji na kroku $i+1$.

W wypadku dużej liczby iteracji na danym kroku, zmniejszana jest w ten sposób długość kolejnego kroku.

skalowanie oparte na normie wektora sił niezrównoważonych

Tego rodzaju skalowanie może mieć miejsce przy każdorazowym wyznaczaniu stanu

układu. Polega ono na wykonaniu następujących kroków:

1. dobór próbnego przyrostu parametru obciążenia α_0^i
2. rozwiązanie układu równań
3. wyznaczenie wektora sił niezrównoważonych
4. jeżeli norma $\|\mathbf{F}^i\|$ wektora sił niezrównoważonych jest większa od zadanej tolerancji ϵ, to należy zmniejszyć przyrost parametru obciążenia α^i

$$\alpha_0^i = \alpha^i \frac{\epsilon}{\|\mathbf{F}^i\|} \tag{1.22}$$

i powrócić do punktu 2

5 jeżeli mniejsza przejść do fazy iteracyjnej.

optymalizacja sekwencji przyrostów obciążenia

Dla obciążeń jednoparametrowych, wektory przyrostu i całkowitego obciążenia możemy zapisać na kroku następująco

$$\Delta \mathbf{R}^i = \lambda^i \mathbf{R}_{ref}$$

$$\mathbf{R}^i = \Delta \mathbf{R}^1 + \Delta \mathbf{R}^2 + \cdots + \Delta \mathbf{R}^i = \Lambda^i \mathbf{R}_{ref} \quad (i = 1, 2, \ldots, N) \tag{1.23}$$

$\mathbf{R}_{ref}$ jest pewnym wektorem odniesienia, z reguły utożsamianym z końcowym stanem obciążenia. W takiej sytuacji suma parametrów przyrostowych spełnia następujący warunek

$$\Lambda^N = \lambda^1 + \lambda^2 + \cdots + \lambda^N = 1 \tag{1.24}$$

Z globalnego punktu widzenia, funkcjonał energii potencjalnej Π może być zapisany jako suma funkcjonałów Π^i na poszczególnych krokach $i = 1, \ldots, N$

$$\Pi = \Pi^1 + \cdots + \Pi^N \tag{1.25}$$

z warunkiem stacjonarności względem $\Delta \mathbf{r}^i$ dla $i = 1, \ldots, N$. Optymalizacja sekwencji przyrostów $\lambda_1, \ldots, \lambda_N$ polega [98] na uwzględnieniu parametrów λ_i w zmodyfikowanym funkcjonale $\tilde{\Pi}$ poprzez wprowadzenie mnożnika Lagrange'a μ.

$$\tilde{\Pi} = \tilde{\Pi}(\Delta \mathbf{r}^i, \lambda^i) = \Pi + \mu(\lambda^1 + \cdots + \lambda^N - 1) \tag{1.266}$$

Wówczas

$$\delta\tilde{\Pi} = \sum_{i=1}^{N} \delta\Delta\mathbf{r}^{iT}\{\mathbf{K}_T(\mathbf{r}^{i-1})\Delta\mathbf{r}^i - \Delta\mathbf{R}^i\} + \sum_{i=1}^{N} \delta\lambda^i\left\{-\Delta\mathbf{r}^{iT}\left(\frac{\delta\Delta\mathbf{R}^i}{\delta\lambda^i}\right) + \mu\right\} + \delta\mu(\lambda^1 + \cdots + \lambda^N - 1) \tag{1.277}$$

z warunkami stacjonarności

$$\mathbf{K}_T\mathbf{r}^{(i-1)}\Delta\mathbf{r}^i = \Delta\mathbf{R}^i = \lambda^i\mathbf{R}_{ref}$$

$$-\Delta\mathbf{r}^{iT}\mathbf{R}_{ref} + \mu = 0 \qquad (i = 1, \ldots, N) \tag{1.288}$$

$$\lambda_1 + \cdots + \lambda_N = 1$$

Z warunku optymalności wynika związek pomiędzy parametrem λ_i a mnożnikiem Lagrange'a μ

$$\mu = \lambda_i\mathbf{R}_{ref}^T\mathbf{K}_T{}^{-1}(\mathbf{r}^{(i-1)})\mathbf{R}_{ref} \tag{1.299}$$

oraz intepretacja mnożnika μ jako gęstości przyrostowej energii potencjalnej względem parametru λ_i.

$$\frac{\Pi^i}{\lambda^i} = \frac{1}{2}\mu = const. \qquad (i = 1, \ldots, N) \tag{1.300}$$

1.1.2 sterowanie przemieszczeniowe

Sterowanie przyrostami wektora obciążenia staje się nieefektywne w otoczeniu punktów maksimum obciążenia. Przejście na sterowanie przemieszczeniowe wiąże się z doborem miary wektora przemieszczenia, którą zamierzamy kontrolować.

Najprostsza koncepcja polega na iteracji przy zachowaniu założonej wartości przemieszczenia wybranego stopnia swobody układu. Na każdym kolejnym kroku wartość ta jest zwiększana o pewien określony przyrost, co prowadzi do niezrównoważenia układu. Nieznany przyrost $\Delta R'$ wektora obciążenia znajdujemy w następujący sposób:

1. rozwiązujemy równania równowagi niezależnie dla siły niezrównoważonej

$$\mathbf{K}_T \Delta \mathbf{r}_F = \mathbf{F} \tag{1.31}$$

oraz dla przemieszczenia wywołanego arbitralnym przyrostem obciążenia

$$\mathbf{K}_T \Delta \mathbf{r}_R = \mathbf{R} \tag{1.32}$$

2. przemieszczenie $\Delta \mathbf{r}$ sterowanego stopnia swobody układu i uzyskujemy jako:

$$\Delta \mathbf{r} = (\Delta \mathbf{r}_F)_i + \alpha_E (\Delta \mathbf{r}_R)_i \tag{1.33}$$

skąd

$$\alpha_E = (\Delta \mathbf{r} - (\Delta \mathbf{r}_F)_i)/(\Delta \mathbf{r}_R)_i \tag{1.34}$$

Uogólnienie powyższej metody polega na sterowaniu pewnym skalarnym przemieszczeniem w kierunku określonym przez dobór wektora jednostkowego $\mathbf{b}$. Ogólniejsza postać wyprowadzonych wzorów prowadzi do

$$\Delta \bar{r} = b^T \Delta r = b^T \Delta r_F + \alpha_E b^T \Delta r_R \tag{1.335}$$

a następnie

$$\alpha_E = \frac{\Delta \bar{r} - b^T \Delta r_F}{b^T \Delta r_R} \tag{1.336}$$

W szczególnym przypadku jako wektor b możemy wybrać sam przyrost Δr wektora przemieszczenia. Prowadzi to zachowania normy euklidesowej wektora przyrostu przemieszczenia

$$\|\Delta r\|_E = (\Delta r^T \Delta r)^{\frac{1}{2}} \tag{1.337}$$

przy czym przyrost obciążenia dany jest przez

$$\alpha_E = \frac{-\Delta r_R^T \Delta r_F \pm \sqrt{(\Delta r_R^T \Delta r_F)^2 - (\Delta r_R^T \Delta r_R)(\Delta r_F^T \Delta r_F)}}{\Delta r_R^T \Delta r_R} \tag{1.338}$$

1.1.3 kroki w przestrzeni przemieszczeniowo-naprężeniowej

Rozwiązanie kompromisowe pomiędzy sterowaniem przemieszczeniowym a obciążeniowym polega na uwzględnieniu w parametrze sterowania zarówno przemieszczenia jak i obciążenia, t.j. na kontrolowaniu pewnej wartości s, zdefiniowanej w przestrzeni przemieszczeniowo - naprężeniowej następująco

$$s = (\alpha_E^2 + \Delta r^T \Delta r)^{\frac{1}{2}} \tag{1.339}$$

gdzie

Δr –przyrost przemieszczenia

α_E –przyrost parametru obciążenia

Przyrost s tak zdefiniowanego parametru sterowania prowadzi do wyrażenia na przyrost wektora obciążenia α_E

$$\alpha_E = \frac{-\Delta r_R^T \Delta r_F \pm \sqrt{(\Delta r_R^T \Delta r_F)^2 - (1 + \Delta r_R^T \Delta r_R)(\Delta r_F^T \Delta r_F - s^2)}}{1 + \Delta r_R^T \Delta r_R} \tag{1.40}$$

Dla konstrukcji podatnych dominującą rolę w sterowaniu odgrywa przemieszczenie, zaś dla struktur sztywnych - obciążenie. Problematyczne wydaje się jednak wykorzystywanie w zależności (1.39) wielkości o różnych mianach. Długość kroku zależy od doboru jednostek, a ponadto uwzględnienie w mierze wszystkich stopni swobody prowadzi do zagubienia lokalnych nieliniowości.

1.1.4 dobór kierunku kroku

Zarówno w wypadku sterowania przemieszczeniem jak i obciążeniem możemy osiągnąć na ścieżce równowagi punkt zwrotny. W sytuacji tego rodzaju należy zmienić znak przykładanych przyrostów parametru sterowania. Przewidywanie takich sytuacji jest ułatwione, jeżeli kontrolujemy zdefiniowany wyżej (1.18) parametr sztywności układu. Na pierwszym kroku wartość parametru sztywności jest równa jedności i wykonujemy krok dodatni. W miarę zmniejszania się sztywności konstrukcji parametr maleje. Gdy parametr sztywności zmienia znak przechodząc przez zero, dobieramy ujemne przyrosty obciążenia. Dodatkową korzyścią jest stosowanie jednego parametru zarówno do doboru kierunku jak i wartości obciążenia.

1.1.5 modyfikacje N-R w fazie iteracyjnej

W otoczeniu punktów krytycznych możemy realizować procedurę N-R bez iteracji. Na każdym kroku przykładany jest przyrost obciążenia oraz siła niezrównoważona z poprzedniego kroku. Kryterium zawieszenia iteracji może być osiąganie wartości bliskich zeru przez współczynnik sztywności S_p.

Iteracja przy stałym obciążeniu może prowadzić do rozbieżności. Zakładając stałość przemieszczenia (pewnej normy wektora Δr) otrzymujemy sytuację w której obciążenie będzie zmienne w trakcie iteracji. Unikamy wtedy problemów związanych z doborem

znaku przyrostu obciążenia charakterystycznych dla struktur z osłabieniem dla których i obciążenie będzie się zmniejszać.

wyznaczanie stanu niezależnie od ścieżki obciążenia

W tradycyjnym algorytmie N-R stan układu jest aktualizowany przy każdej iteracji... Jeżeli, jak to ma miejsce w wypadku niektórych równań konstytutywnych, stosujemy w przypadku odciążania zależności inne niż dla obciążania, to kumulacja stanów niewpełnia zrównoważonych może prowadzić do nie mających w rzeczywistości miejsca odciążeń i w konsekwencji do stabilizacji procesu iteracyjnego w punkcie odległym od ścieżki równowagi. Modyfikacja algorytmu N-R polegająca na przyjmowaniu w kolejnych iteracjach jako konfiguracji odniesienia stanu zrównoważonego z początku kroku pozwalaa wyznaczać kolejne stany równowagi niezależnie od drogi obciążenia.

aproksymacja liniowa

Każda z procedur analizy nieliniowej zawiera element rozwiązania układu równań liniowych, w wyniku którego otrzymujemy kolejne przybliżenie $\Delta r^{(i+1|k+1)}$ wektoraa przyrostu przemieszczenia. Poszukujemy takiej wartości mnożnika β aby otrzymać minimalne niezrównoważenie. Algorytm można przedstawić w dwóch krokach:

1. Obliczamy pracę obciążenia niezrównoważonego na początku kroku ($\beta = 0$) i dla wyliczonego przyrostu przemieszczenia ($\beta = 1$) :

$$W(\beta) = \Delta r^T \mathbf{F}(\beta) \tag{1.411}$$

Jeżeli uzyskane wartości są przeciwnych znaków, to przechodzimy do drugiej fazyy, w przeciwnym razie podwajamy długość kroku i wracamy do punktu 1

2. Wyliczamy wartośc β metodą aproksymacji liniowej

$$\beta = \beta_1 + \frac{W(\beta 1)(\beta_2 - \beta_1)}{W(\beta_2) - W(\beta_1)} \tag{1.422}$$

Jeżeli osiągnęliśmy zbieżność to przechodzimy do kolejnego przyrostu obciążenia, jeżeli nie, to iterujemy powtarzając od punktu 1 do uzyskania zbieżności

metoda kierunków sprzężonych

Rozszerzając ideę aproksymacji liniowej o modyfikację nie tylko długości przemieszczenia, lecz też jego kierunku możemy sformułować metodę kierunków sprzężonych.

1. Rozwiązujemy układ równań liniowych

$$\Delta r^{(i+1)} = K_T^{i-1}(R^{(i+1)} - F^i) \tag{1.43}$$

tylko raz, na początku kroku.

2. Znajdujemy wektor Δr_c, sprzężony z wektorem pochodzącym z poprzedniej iteracji.

$$\Delta r^{(i+1)}{}_c = \Delta r^{(i+1)} - c\Delta r_c^i \tag{1.44}$$

gdzie c jest stałą określoną tak aby

$$(\Delta r^{(i+1)}{}_c)^T K_T \Delta r_c^i = 0 \tag{1.45}$$

3. Minimalizujemy niezrównoważenie metodą aproksymacji liniowej. Powtarzamy ostatnie dwa kroki aż do spełnienia kryterium zbieżności.

specyficzne strategie przy silnych niezrównoważeniach

Obliczamy przyrost przemieszczenia dla arbitralnej wartości obciążenia. Sprawdzamy, czy nie nastąpiła przewidywana zmiana stanu elementów (uplastycznienie, zamknięcie szczeliny etc.). Skalujemy długość kroku tak, by maksymalnie zbliżyć się do punktu ścieżki równowagi odpowiadającego jednemu z przewidywanych zjawisk. W ten sposób wykorzystujemy pełną analizę N-R jedynie w otoczeniu punktów, w których jest to konieczne.

Wygodne jest archiwizowanie stanów zbieżności. Po wykonaniu pierwszych kilku iteracji możemy spróbować przyśpieszyć zbieżność. Jeżeli zbieżność wymaga

zbyt dużej liczby iteracji lub jest w ogóle niemożliwa, powracamy do zachowanego stanu równowagi.

1.2 analiza liniowa

Przyjęcie liniowej postaci związku (1.2) prowadzi do analizy zagadnień liniowych. Atrakcyjną cechą analizy liniowej jest możliwość adaptacyjnego poprawiania zbieżności [39, 40, 71, 72, 73, 96, 105, 107, 108].

Zasadniczym zagadnieniem numerycznej analizy konstrukcji metodą elementów skończonych jest ocena dokładności uzyskanych wyników, a zarazem problem uzyskania największej dokładności przy ograniczeniu czasu obliczeń. Prowadzący obliczenia musi więc uzyskawszy wyniki dla założonego modelu skończenie elementowego, udzielić odpowiedzi na pytanie, jak zmodyfikować model, aby z jednej strony poprawić rozwiązanie, z drugiej zaś nie przekroczyć dostępnych zasobów komputerowych.

Znane metody *h* oraz *p* poprawiania zbieżności wymagają znajomości lokalnych oszacowań błędu. Metoda *h* polega na zwiększaniu liczby stopni swobody poprzez zagęszczanie podziału na elementy w obszarach gdzie nie uzyskano wymaganej dokładności. Metoda *p* polega odpowiednio na zwiększaniu liczby stopni swobody przy zachowaniu dotychczasowego podziału poprzez podnoszenie stopnia wielomianów aproksymacyjnych. Znacznie mniej popularna metoda *t* stopniowo dostosowuje topologię podziału obszaru na elementy przy zachowaniu zarówno ich liczby jak i liczby stopni swobody.

Wybór właściwej metody adaptacyjnej wymaga uwzględnienia szeregu przesłanek natury heurystycznej. Aproksymacja skończenieelementowa dotyczy nie tylko rozwiązania, lecz przede wszystkim geometrii obszaru. Z tego względu metody *p* oraz *t* bazujące na stałej liczbie elementów mogą okazać się niewystarczające w obszarach nieregularnych, gdzie wzrost dokładności wymaga poprawy odwzorowania geometrii poprzez wzrost liczby elementów, czyli metodą *h*. Z kolei metoda *h* lub *p* może dawać zbieżność znacznie słabszą niż adaptacja topologii metodą *t*. Tak więc wybór metody zależy od szczególnego zadania, ograniczenia czasu obliczeń a tym samym kosztów i wymaga każdorazowo starannego rozważenia.

1.3 analiza dynamiczna

Dobór odpowiedniego algorytmu jest szczególnie istotny w wypadku uwzględnienia pełnej postaci równania (1.1). Analiza nawet niewielkich zagadnień dynamicznych wymaga poważnych zasobów komputerowych zaś efektywność algorytmów silnie zależy od szczególnego problemu. Pomimo istnienia całego szeregu metod, rozwiązanie wielu zadań jest niemożliwe z uwagi na niewystarczającą moc obliczeniową dostępnego sprzętu.

Rozpatrzmy liniową postać zależności (1.1) w której dla uproszczenia pominięto macierz tłumienia C.

$$\mathbf{M}\ddot{\mathbf{r}}^{(t)} + \mathbf{K}_T\mathbf{r}^{(t)} = \mathbf{R}^{(t)} \tag{1.46}$$

Rozpatruje się dwa rodzaje macierzy mas: tzw. macierz konsystentną o postaci niediagonalnej, oraz diagonalną macierz zredukowaną. Macierz konsystentna daje dokładniejsze częstości dla konstrukcji prętowych, lecz korzyść ta zmniejsza się ze wzrostem długości fali. W ogólności wymagany rodzaj macierzy mas zależy w metodach bezpośredniego całkowania równań od algorytmu całkowania względem czasu. Należy jedynie zaznaczyć, że w wypadku zastosowania macierzy diagonalnej otrzymujemy oszacowanie częstości drgań z dołu, zaś w wypadku macierzy pełnej otrzymujemy oszacowanie z góry.

Dla zagadnień dynamicznych opracowano dwie zasadnicze strategie: metodę całkowania bezpośredniego i superpozycji modalnej. Wybór metody będzie zależał od częstotliwości wymuszenia i od tego, czy zagadnienie dotyczy dynamiki konstrukcji, czy też propagacji fal. W wypadku analizy zagadnień propagacji fal odpowiedniejsze są metody bezpośredniego całkowania z uwagi na potrzebę uwzględnienia dużej ilości postaci drgań.

1.3.1 metoda superpozycji modalnej

W metodzie superpozycji modalnej diagonalizuje się macierze sztywności mas celem rozprzężenia układu równań przez rozwiązanie zagadnienia własnego

$$K_T \Phi = \omega^2 M \Phi \qquad (1.447)$$

Wartości własne ω i wektory Φ odpowiadają częstościom i postaciom własnym drgań konstrukcji. W praktyce wystarcza niewielka ilość najmniejszych wartości własnych.

W metodzie superpozycji modalnej najbardziej czasochłonnym etapem jest rozwiązanie zagadnienia własnego. W programach MES wykorzystuje się metody transformacyjne oraz wektorowe metody iteracyjne.

Metody transformacyjne pozwalają na wyznaczenie wszystkich wartości własnych dzięki przekształceniu macierzy do postaci trójdiagonalnej (metody Hauseholdera i GGivensa) lub diagonalnej (metoda Jacobiego). Implementacje tych metod w istniejących programach z reguły nie wykorzystują charakterystycznej dla MES pasmowości macierzy. W związku z tym, w dużych zadaniach proces wyznaczenia wartości własnych poprzeddza się tzw. statyczną kondensacją macierzy (zastąpienie macierzy K_T oraz M odpowieednimi macierzami o mniejszym wymiarze). Zwykle eliminuje się obrotowe stopnie swwobody oraz te, którym nie przyporządkowano bezwładności.

Wariacje klasycznej metody potęgowej prowadzą do wektorowych metod iteracyyjnych. Jedną z nich jest metoda iteracji podprzestrzeni w której iteracje wykonuje się na grupie wektorów próbnych co prowadzi do wyznaczenia wielu wektorów własnychh w każdym cyklu iteracji. Metoda ta jest szczególnie efektywna gdy potrzebujemy wyyznaczyć dużą liczbę wartości własnych. Ponadto wykorzystuje ona pasmowość macierrzy.

1.3.2 bezpośrednie całkowanie równań ruchu

W metodach bezpośredniego całkowania wyróżniamy dwie grupy: jawne i niejawne całkowanie względem czasu. Obydwie metody wykorzystują operatory różniccowe aproksymujące związki pomiędzy przemieszczeniami $\mathbf{r}$ a prędkościami $\dot{\mathbf{r}}$ i przyspieszeniami $\ddot{\mathbf{r}}$. Przykładowo, w metodzie Newmarka przyjmujemy

$$\dot{\mathbf{r}}^{(t+\Delta t)} = \dot{\mathbf{r}}^{(t)} + \Delta t(\ddot{\mathbf{r}}^{(t)} + \ddot{\mathbf{r}}^{(t+\Delta t)}) \qquad (1.448a)$$

$$\mathbf{r}^{(t+\Delta t)} = \mathbf{r}^{(t)} + \Delta t \dot{\mathbf{r}}^{t} + \Delta t^2((\frac{1}{2} - \beta)\ddot{\mathbf{r}}^{(t)} + \beta\ddot{\mathbf{r}}^{(t+\Delta t)}) \tag{1.48b}$$

Powyższe wzory możemy wyprowadzić dla $\beta = \frac{1}{4}$ przy założeniu stałego przyspieszenia w przedziale czasu $(t, t + \Delta t)$ równego średniemu przyspieszeniu na krańcach przedziału. Dla $\beta = \frac{1}{6}$ otrzymujemy związki dla odcinkowo liniowego pola przyspieszeń, natomiast $\beta = 0$ odpowiada impulsom przyspieszenia w odstępach Δt. Można ponadto pokazać, że metoda Newmarka przy $\beta = 0$ odpowiada wzorom metody różnic centralnych

$$\dot{\mathbf{r}}^{(t+\frac{\Delta t}{2})} = \dot{\mathbf{r}}^{(t-\frac{\Delta t}{2})} + \Delta t \ddot{\mathbf{r}}^{(t)} \tag{1.49a}$$

$$\mathbf{r}^{(t+\Delta t)} = \mathbf{r}^{t} + \Delta t \dot{\mathbf{r}}^{(t+\frac{\Delta t}{2})} \tag{1.49b}$$

przy założeniu, że

$$\dot{\mathbf{r}}^{(t+\frac{\Delta t}{2})} = \dot{\mathbf{r}}^{t} + \frac{1}{2}\Delta t \ddot{\mathbf{r}}^{(t)} \tag{1.50}$$

Spośród całej gamy możliwości wymienimy jeszcze wzory Houboldta oparte na odcinkowo sześciennej interpolacji przyspieszeń i metodę Wilsona. W metodzie niejawnej otrzymujemy dla powyższych metod algorytmy bezwarunkowo stabilne, natomiast dla metody Newmarka wymagane jest aby $\beta \geq 4$. Tym samym długość kroku czasowego w metodach niejawnych jest ograniczona przez wymagania zbieżności. Metoda różnic centralnych jest jedynie warunkowo stabilna, w związku z czym używa się jej jedynie w procedurach całkowania jawnego. Długość kroku czasowego powinna się mieścić w granicach stabilności

$$\Delta t^2 \leq 4/\lambda \tag{1.51}$$

gdzie λ jest kwadratem najwyższej częstości drgań układu. Dla siatek elementów skończonych o liniowym polu przemieszczeń translacyjnych i sześciennym polu obrotów λ może być szacowane następująco

$$\lambda = \frac{4c^2}{l^2} \quad , \lambda = 12\frac{D}{lm} \tag{1.52}$$

gdzie l jest długością elementu, c prędkością fali sprężystej, D sztywnością zgięciową (EI) dla zagadnień sprężystych), zaś m zredukowaną bezwładnością obrotową. Należy użyć większej z wyznaczonych wartości. Jednakowoż, z powodu błędów zaokrągleń i innych niewyjaśnionych powodów, w praktyce należy używać od 0.5 do 0.8 wartości wyznaczonej z równania (1.52).

Odrębnym zagadnieniem jest rozpoznanie przez użytkownika przekroczenia granic stabilności rozwiązania. Wprawdzie niestabilne rozwiązanie prowadzi szybko do błędu nadmiaru w komputerze, to jednak w zagadnieniach z materiałami posiadającymi zdolność dyssypacji energii (n.p. sprężysto-plastycznych) niestabilność może ulec tłumieniu i wyniki nie będą zawierały ewidentnych błędów. Zjawisko dyssypowanej niestabilności może zostać wykryte przy sprawdzaniu bilansu energetycznego układu, tym niemniej większość programów nie daje takiej możliwości. Tak więc stosowanie algorytmów warunkowo stabilnych wymaga zachowania szczególnej ostrożności.

niejawne całkowanie

Wzory niejawnego całkowania otrzymujemy eliminując z (1.46) dla chwili $t + \Delta t$ przyspieszenia przez podstawienie (1.48). Dla metody Newmarka otrzymujemy

$$\tilde{\mathbf{K}}\mathbf{r}^{(t+\Delta t)} = \tilde{\mathbf{R}} \tag{1.53}$$

gdzie

$$\tilde{\mathbf{K}} = \mathbf{M} + \beta\Delta t^2\mathbf{K}_T$$

$$\tilde{\mathbf{R}} = \beta\Delta t^2\mathbf{R}^{(t+\Delta t)} + \mathbf{M}((\frac{1}{2} - \beta)\Delta t^2\ddot{\mathbf{r}}^{(t)} + \Delta t\dot{\mathbf{r}}^{(t)} + \mathbf{r}^{(t)}) \tag{1.54}$$

Tym samym metoda niejawna prowadzi na każdym kroku czasowym do rozwiązania układu równań na wektor przemieszczeń $\mathbf{r}$. W związku z tym efektywność obliczeń

będzie określona przede wszystkim przez procedurę rozwiązywania układu równań algebraicznych. W zagadnieniach liniowych macierz $\bar{K}_t$ jest triangularyzowana raz (tabela 1.2). Dla zagadnień o n stopniach swobody, przy szerokości półpasma n_b należy wykonać nn_b^2 mnożeń. Równania (1.52) rozwiązujemy wówczas przez pojedyncze podstawienie wektora prawych stron, co wymaga $2nn_b$ mnożeń na każdym kroku czasowym. Tabela (1.1) zawiera porównanie liczby operacji dla różnych metod bezpośredniego całkowania równań ruchu. Metody całkowania niejawnego nie są tak szybkie na kroku czasowym jak metody jawne, tym niemniej ich bezwarunkowa stabilność pozwala na dobór dłuższych kroków czasowych. Można stwierdzić, że metody niejawne dają lepsze rezultaty w zagadnieniach liniowych przy niewielkiej szerokości półpasma.

metoda	zagadnienie	liczba mnożeń na kroku czasowym
	liniowe	$n_N n_C n_D^2$
jawna		
	nieliniowe	$2(m_B + m_M)en_I$
	liniowe	$n_D n_N (2n_B + n_D n_C) + n_B^2 n_D n_N$
niejawna		
	nieliniowe	$n_D n_N (n_B^2 + 2n_B) + (2m_B + m_M)en_I +$ *obliczenie macierzy sztywności*

e – liczba elementów

n_I – liczba punktów całkowania w elemencie

m_B – liczba mnożeń w równaniach przemieszczenie –odkształcenie

m_M – liczba mnożeń w równaniu konstytutywnym

n_C – 3, 9, 27 odpowiednio dla zagadnień jedno- dwu- i trójwymiarowych

n_D – liczba stopni swobody w węźle

n_N – liczba węzłów

Tabela 1.1

Oszacowanie liczby operacji w bezpośrednim całkowaniu.

Liczba możliwych podejść ogromnie zwiększa się przy dopuszczeniu nieliniowości w procesach dynamicznych. Procedury niejawnego całkowania równań ruchu prowadzą do rozwiązywania układów równań algebraicznych zbliżonych do tych z jakimi mamy do czynienia w nieliniowej statyce, tak więc możemy wykorzystać omówione wyżej procedury typu Newtonowskiego. Dodatkowe trudności wynikają z konieczności wyboru kon-

figuracji odniesienia, a tym samym stacjonarnego lub uaktualnianego opisu Lagrangige'a lub współrzędnych konwekcyjnych. Dla zagadnień związanych z dużymi ugięciami należy ponadto uwzględnić w macierzy stycznej macierze początkowych naprężeń i sztywności geometrycznej. Macierz styczną należy liczyć i triangularyzować na każdym kroroku, co prowadzi do znacznej liczby operacji algebraicznych. Ponadto, dla materiałów o własnościach zależnych od drogi obciążenia, długość kroku czasowego jest dodatkkowo ograniczona co prowadzi do znacznie wyższych kosztów obliczeń niż w metodzie jawwnej.

jawne całkowanie

Przyspieszenia na każdym kroku znajdowane są z

$$\ddot{r} = M^{-1}(R - K_T r) \qquad (1(1.55)$$

Przemieszczenia i prędkości wynikają wówczas z (1.48). Dla metody Newmarka a jest to możliwe jedynie przy $\beta = 0$. Dla diagonalnej macierzy M metoda wymagaga jedynie mnożenia macierzy, bez ich odwracania. Jak wspomniano powyżej, użycie e diagonalnych macierzy bezwładności w metodach jawnych jest pożądane również z uwagi na dokładność. Mnożenie macierzy sztywności przez wektor przemieszczeń może be być wykonywane za pomocą technik opracowanych dla macierzy rzadkich i tym samym li liczba operacji zależeć będzie jedynie liniowo od liczby stopni swobody i nie będzie mieć związku z szerokością pasma. Tabela (1.1) zawierająca oszacowanie liczby operacji wskazuuje, że korzyści ze stosowania metody jawnej wzrastają z szerokością pasma. Jednakowoż, dla długotrwałych obciążeń metoda jawna staje się nieekonomiczna z uwagi na krótkie i kroki czasowe.

Metoda jawna jest szczególnie korzystna w zagadnieniach nieliniowych, gdyż i nieliniowości nie wprowadzają zasadniczego wzrostu kosztów. W jawnej metodzie r różnic centralnych rozwiązujemy równanie

$$\ddot{r} = M^{-1}(R - F) \qquad (1.56)$$

w którym macierz styczna wogóle nie występuje. Liczba operacji zależy głównie od liczby elementów, niezależy natomiast od szerokości pasma. Wprawdzie krok czzasowy musi być dobrany zgodnie z ograniczeniem stabilności metody, to z reguły jego długość

będzie zgodna z wymaganą przez adekwatny opis historii obciążenia dla materiałów sprężysto- plastycznych. Należy jednak podkreślić, że w wielu zagadnieniach użycie metody niejawnej będzie konieczne z uwagi na zbieżność obliczeń.

podsumowanie

W rozdziale powyższym chcieliśmy podkreślić uderzający kontrast pomiędzy prostymi i eleganckimi równaniami teorii MES z jednej strony a wysoką złożonością oprogramowania z drugiej. Wynika on z faktu, że równania MES reprezentują wysoki poziom abstrakcji w odniesieniu do istotnych parametrów problemu. Np. wymiar i gęstość wypełnienia macierzy mające pierwszorzędne znaczenie z punktu widzenia programowania są zdeterminowane przez szczególne zagadnienie i nie pojawiają się jawnie w równaniach macierzowych. Poza sformułowaniami macierzy elementowych, praktycznie wszystkie podstawowe równania macierzowe słuszne są dla dowolnego rodzaju konstrukcji, niezależnie od tego, czy mamy do czynienia z ośrodkiem dwu lub trójwymiarowym, strukturą cienkościenną czy konstrukcją fundamentu reaktora. Gdy dochodzi do implementacji programu numerycznego, nie możemy pozostać na tym samym poziomie abstrakcji. Wprost przeciwnie, im większy udział specyfiki zadania w programie, tym łatwiejsze i skuteczniejsze zastosowanie.

W środowisku badawczym, kandydaci do stopni naukowych wykonują ogromną ilość tradycyjnego, fortranowskiego oprogramowania celem rozwiązania konkretnych, wąskich zagadnień. Każda zmiana struktury problemu powoduje konieczność ingerencji w oprogramowanie. Jest to powód dla którego programy rozwijane jako narzędzia badawcze szczególnego przeznaczenia nie kwalifikują się do bezpiecznego użycia w warunkach eksploatacyjnych przez kogokolwiek z wyjątkiem autora. Obecnie istnieją, i wciąż przybywają ogromne ilości pakietów o takiej właśnie genezie - drastyczna wskazówka, że problem narzędzi badawczych MES nie jest opanowany. W rozdziale drugim przedstawiamy pewne koncepcje programowania wywodzące się z metod tzw. sztucznej inteligencji, które w dalszej części pracy posłużą nam do sformułowania ogólnej koncepcji usprawnienia tradycyjnego środowiska obliczeń numerycznych zagadnień mechaniki.

ROZDZIAŁ 2
WPROWADZENIE DO ZAGADNIEŃ SZTUCZNEJ INTELIGENCJI.

Równolegle z metodami numerycznymi rozwijała się inna dziedzina informatyki - sztuczna inteligencja [2.1]. Wywodzi się ona z rozbudzonej nastaniem ery maszyn cyfrowych aspiracji człowieka do stworzenia doskonalszego odwzorowania swojego własnego umysłu: - w latach pięćdziesiątych pisano często o koncepcji "mózgu elektronowego". Pogląd na istotę AI ewoluował jednak wraz ze zbieraniem kolejnych doświadczeń. Fiasko koncepcji GPS [2.2], programu który miał rozwiązywać dowolne zadania, nieudane próby automatycznego przekładu, uświadomiły badaczom istnienie wspólnej bariery którą napotykają wszelkie poważne projekty AI. Rozpatrzmy bowiem pewien chwilowy stan jakiegoś abstrakcyjnego świata, powiedzmy pozycję na szachownicy i spróbujmy dobrać optymalną kontynuację, czyli najlepsze posunięcie jednego z grających. We wczesnym okresie rozwoju AI modelowano właśnie gry i problemy matematyczne [87,93,100], które dla stosunkowo niewielkich struktur wyjściowych, przy niewielkim wysiłku poświęconym na programowanie generują sytuacje o największym stopniu złożoności. Jeżeli zaczniemy teraz przeglądać kolejne warianty, to natrafimy na zjawisko eksplozji kombinatorycznej polegające na niepohamowanym wzroście liczby kombinacji już przy niewielkim oddaleniu od sytuacji wyjściowej. Jak zjawisko to jest istotne świadczy, że nawet najszybsze współczesne komputery przegrywają z arcymistrzami szachowymi.

Istota przewagi człowieka nad maszyną polega między innymi na jego zdolności identyfikowania obiecujących koncepcji i skupiania na nich uwagi przy jednoczesnym odrzucaniu pozostałych możliwości. Wymaga to wiedzy specyficznej, właściwej dla konkretnej dziedziny, wynikającej z doświadczenia w rozwiązywaniu podobnych zagadnień. Wiedza taka jest z natury swojej heurystyczna, nie zawsze daje się wyartykułować i często nie posiada racjonalnych podstaw - gracze szachowi nierzadko podejmują decyzje kierując się przesłankami estetycznymi. Obecny etap rozwoju AI dotyczy uchwycenia właśnie tego rodzaju wiedzy. Zajmuje się tym inżynieria wiedzy [2.3], poddziedzina AI związana z budową tzw. systemów ekspertowych [2.4], t.j. programów komputerowych mających spełniać rolę eksperta w konkretnej, wąskiej dziedzinie. K.E. jest praktycznie jedyną

[2.1] ang. Artifitial Itelligence, AI
[2.2] ang. General Problem Solver
[2.3] ang. Knowledge Engineering, KE
[2.4] ang. Expert System, ES

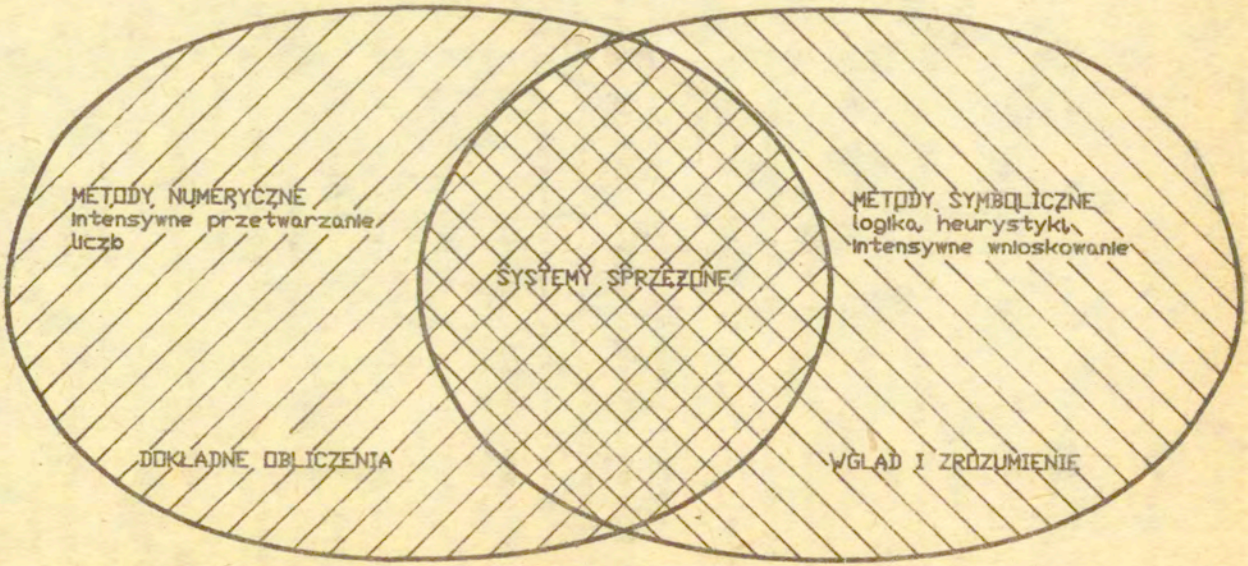
dziedziną AI znajdującą praktyczne, nie akademickie, laboratoryjne zastosowania.

AI stale ewoluując nie posiada i nie może posiadać ścisłej definicji. Można jedynie ogólnie powiedzieć, że AI jest dziedziną informatyki zajmującą się budową systemów komputerowych naśladujących procesy przetwarzania informacji przez umysł ludzki. Praktyczniej będzie wyszczególnić aktualne obszary zastosowań AI [65,55]:

- przetwarzanie języka naturalnego
- wyprowadzanie informacji z baz danych
- automatyczne dowodzenie twierdzeń
- uczenie maszyn
- automatyczne programowanie
- zadania kombinatoryczne
- zagadnienia robotyki
- inżynieria wiedzy.

AI bazuje na zasadniczo odmiennej niż metody numeryczne koncepcji użycia maszyny cyfrowej. O ile metody numeryczne zmierzają do maksymalnego wykorzystania mocy komputera w celu przetwarzania liczb, to AI bazuje na przetwarzaniu symboli. Podstawowe różnice pomiędzy koncepcją programowania AI a tradycyjnym rozumieniem programowania można przedstawić następująco:

PROGRAMOWANIE AI	*PROGRAMOWANIE TRADYCYJNE*
Interpretacja danych	*Przetwarzanie danych*
symbolicznych.	*numerycznych.*
Modelowanie jakościowe	*Modelowanie ilościowe*
heurystyki	*statystyka i prawdopodobieństwo*
Deklaratywny opis wiedzy o	*Algorytmiczny opis wiedzy.*
strukturze symbolicznej.	
niedeterministyczne	*deterministyczny charakter*
sterowanie przebiegiem programu	*obliczeń*
Preferencja interaktywnych	*Preferencja przetwarzania*
środowisk rozwoju i	*sekwencyjnego lub wsadowego,*
zastosowań oprogramowania	*co utrudnia uzasadnianie*



Rys. 2.1

Metody komputerowe.

umożliwia tłumaczenie *działania programu.*
działania programu.
Programowanie poprzez budowę *Programowanie w oparciu o*
i uściślanie prototypów. *specyfikacje.*

AI posługuje się odmiennymi niż metody numeryczne językami programowania. Zamiast FORTRANu lub PASCALa stosuje się LISP, PROLOG lub SMALLTALK. Inny charakter będą miały wymagania odnośnie zasobów komputerowych: pamięć operacyjna, sposób interakcji wreszcie architektura wewnętrzna maszyny. W chwili obecnej trudno mówić o sensownym programowaniu AI na sprzęcie nie posiadającym conajmniej 4MB pamięci operacyjnej. Dedykowane komputery, tak zwane maszyny lispowe i komputerowe stanowiska pracy [2.5] charakteryzują się z reguły kolorową grafiką o wysokiej rozdzielczości (rzędu 1024*1024), pracą w wielu oknach równocześnie i wszystkie wyposażone są w mysz. Zamiast procesora wektorowego lub macierzowego pożądana jest struktura pamięci odpowiadająca listom LISPowym lub tak jak to ma miejsce w japońskim projekcie V generacji, implementacja maszynowa PROLOGu.

2.1 Systemy sprzężone.[2.6]

Możemy wymienić dwie główne przyczyny naszego zainteresowania sprzężeniem obliczeń numerycznych z symbolicznymi. Pierwsza to potrzeba wspomożenia osób używających złożonych algorytmów i programów numerycznych. Celem rozwiązania wielu zagadnień naukowych i technicznych wymagany jest wgląd w istotę problemu i precyzja obliczeń. W wielu wypadkach otrzymanie wyników zależy od wiedzy o procesie rozwiązania, w innych wiedzy wymaga interpretacja wyników obliczeń. W jeszcze innych użytkownicy potrzebują doradztwa i pomocy w użyciu narzędzi, którymi dysponują. Tradycyjne obliczenia zapewniają wprawdzie dużą dozę dokładności, lecz nie dają żadnego z powyższych rodzajów zrozumienia. Użytkownicy zamierzający zastosować metody komputerowe lub ocenić wyniki programów numerycznych pozostawieni są z reguły samym sobie. Systemy sprzężone zapowiadają integrację tłumaczenia i zdolności rozwiązywania zadań właściwych dla systemów ekspertowych, z dokładnością tradycyjnych obliczeń numerycznych.

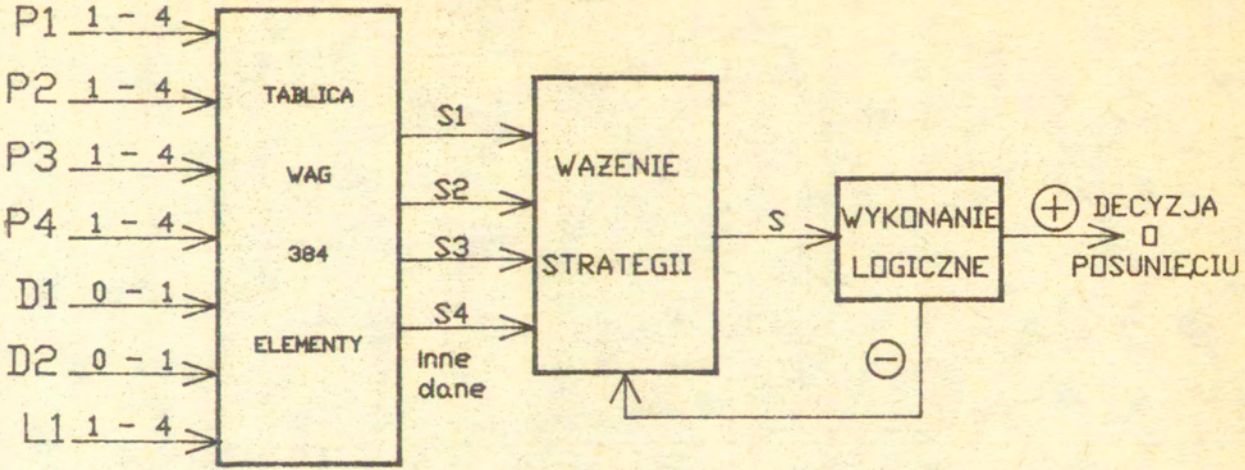
[2.55] ang. workstations
[2.66] ang. Coupled Systems, CS

Naukowcy i inżynierowie na codzień zajmują się zagadnieniami związanymi z wieloznacznymi, sprzecznymi i niedokładnymi danymi. Nawet jeżeli wiele części całej analizy i procesu projektowania może być modelowanych matematycznie lub obliczonych przy użyciu metod numerycznych, to wiele zagadnień zawiera elementy niedostatecznie zdefiniowane lub zrozumiane by mogły być sprowadzone do tradycyjnych technik algorytmicznych lub symbolicznych. W tych wypadkach wymagane jest bardziej efektywne niż tradycyjnie środowisko rozwiązywania zadań. Tak więc drugą przyczyną sprzęgania technik symbolicznych z numerycznymi jest potrzeba narzędzi obliczeniowych o wzrastającej mocy i użyteczności, zdolnych przekroczyć ograniczenia tradycyjnych środowisk. Wierzymy, że integracja formalnych metod matematycznych z metodami opartymi na wiedzy symbolicznej, to znaczy sprzężenie metod numerycznych z symbolicznymi (rys 2.1) jest kluczem do rozwinięcia narzędzi obliczeniowych zdolnych rozwiązać przynajmniej część problemów uważanych dotychczas za niedostępne.

2.2 Reprezentacja wiedzy.

Sztuczna inteligencja zajmuje się budową programów rozwiązujących zadania, których podjęcie przez człowieka wymaga szeroko rozumianej wiedzy. Wprawdzie epistemologia - nauka o wiedzy fascynuje człowieka nieomal od początków jego świadomego istnienia, to jednak po dzień dzisiejszy brak jest jednoznacznej definicji istoty wiedzy. Praktyczne badania sztucznej inteligencji wywodzić się zatem muszą z pewnej pragmatycznej koncepcji rozumienia tego pojęcia. Zamiast prowadzić scholastyczne spory, większość badaczy zmierza do doskonalenia metod reprezentacji wiedzy: takich struktur danych i mechanizmów sterujących, które prowadzą do potocznie przyjmowanego za inteligentne zachowania się programu.

Jedną z zasadniczych kwestii do roztrzygnięcia przy określeniu pożądanego formalizmu dla konkretnego zastosowania staje się wybór pomiędzy deklaratywnym a proceduralnym zapisem wiedzy. Poprawny formalizm powinien zapewniać odwzorowanie dostatecznie szerokiego zakresu wiedzy na odpowiednim poziomie szczegółowości. Ponadto, wygodnie jest uwzględnić pewne dodatkowe postulaty, w szczególności dotyczące modularności zapisu, jego przejrzystości, zrozumiałości.



Rys. 2.2

Przepływ informacji w systemie z tablicą decyzyjną.

przestrzeń stanów

We wczesnym okresie rozwoju AI posługiwano się grami takimi jak warcaby [87] lub domino [93] celem modelowania sytuacji wymagających podejmowania decyzji. Zbiór sytuacji opisywano przez wartości odpowiednio dobranych parametrów. Wszelkie legalne posunięcia klasyfikowano poprzez określenie zbioru strategii. Dla każdej sytuacji określano preferowane stategie przez odpowiedni dobór wag. Przykładowo, dla gry w domino [93] podano następujące strategie:

$S1$ - gra na maksymalny zapis,

$S2$ - gra na blokowanie następnego w kolejności gracza,

$S3$ - pomoc partnerowi,

$S4$ - "otwieranie" własnej ręki.

Przestrzeń stanów opisano przez zbiór kombinacji następujących parametrów:

$P1, P2, P3, P4$ - kolejne numery graczy którzy mają odpowiednio coraz mniej kości,

$D1 = 1$, jeżeli maszyna ma o 2 kości więcej niż gracz P1, przeciwnym razie $D1 = 0$,

$D2 = 1$, jeżeli program ma con. o 2 kości więcej niż gracz $P2$, w przeciwnym razie $D2 = 0$,

$L1$ = liczba kości gracza $P1$, lecz nie więcej niż 4.

Przepływ informacji w tak określonym systemie przedstawiony jest na rys(2.2).Wagi dla poszczególnych strategii dobierano doświadczalnie przez premiowanie strategii wygrywających i dyskryminowanie przegrywających po zakończeniu każdego rozdania. Rozdania były generowane automatycznie i rozgrywane samodzielnie przez komputer, można więc mówić o automatycznym uczeniu się programu. Program osiągnął poziom mistrzowski po rozegraniu 10000 partii.

Ograniczenie metody przestrzeni stanów wynika z opisanego wcześniej zjawiska eksplozji kombinatorycznej. Ponadto, w związku z niejawną reprezentacją wiedzy trudno jest zrealizować moduł tłumaczący działania systemu.

reprezentacja regułowa

Najczęściej spotykane systemy regułowe składają się z trzech elementów: bazy reguł, interpretera i pewnej struktury danych zwanej kontekstem [2.7]. Reguły zapisuje się w postaci:

$$JEŻELI \quad (warunek_1, warunek_2, \ldots, warunek_n)$$

$$TO \quad (akcja_1, akcja_2, \ldots, akcja_n).$$

W trakcie pracy systemu, jeżeli zachodzi ciąg warunków, t.j. lewa strona danego prawa jest spełniona, to wykonywane są akcje zawarte po prawej stronie. Reguły wypisane są w sposób jawny, ich liczba może być znaczna. Wykonywaniem reguł steruje interpreter. Interpreter buduje w trakcie pracy kontekst - zapis wszelkich danych dotyczących aktualnie rozwiązywanego zagadnienia. Początkowo kontekst zawiera wyjściowe informacje z którymi system przystąpił do pracy. Wykonanie kolejnej akcji polega na rozbudowaniu kontekstu o nowy fakt. Kontekst często implementowany jest jako sieć semantyczna lub drzewo. W tym ostatnim przypadku, w węzłach znajdują się kolejno uzyskiwane fakty, gałęzie odpowiadają regułom je wiążącym zaś liście drzewa oznaczają konkluzje. Interpreter działa w cyklu:

- wyszukanie reguł aktywnych w danym kontekście,
- rozwiązanie konfliktu [2.8],
- wykonanie.

Reguł aktywnych, t.j. o spełnionych lewych stronach może wystąpić kilka. Rozwiązanie konfliktu, czyli wybór reguły do wykonania wymaga przyjęcia pewnej strategii. Liczba możliwych strategii jest praktycznie nieograniczona. Wymienimy jedynie najczęściej stosowane:

- wykonuje się pierwszą znalezioną regułę,
- wykonuje się regułę o największej liczbie warunków,
- wykonuje się regułę najbardziej rozbudowującą kontekst,
- wykonuje się wszystkie znalezione reguły równolegle,

[2.7] ang. context

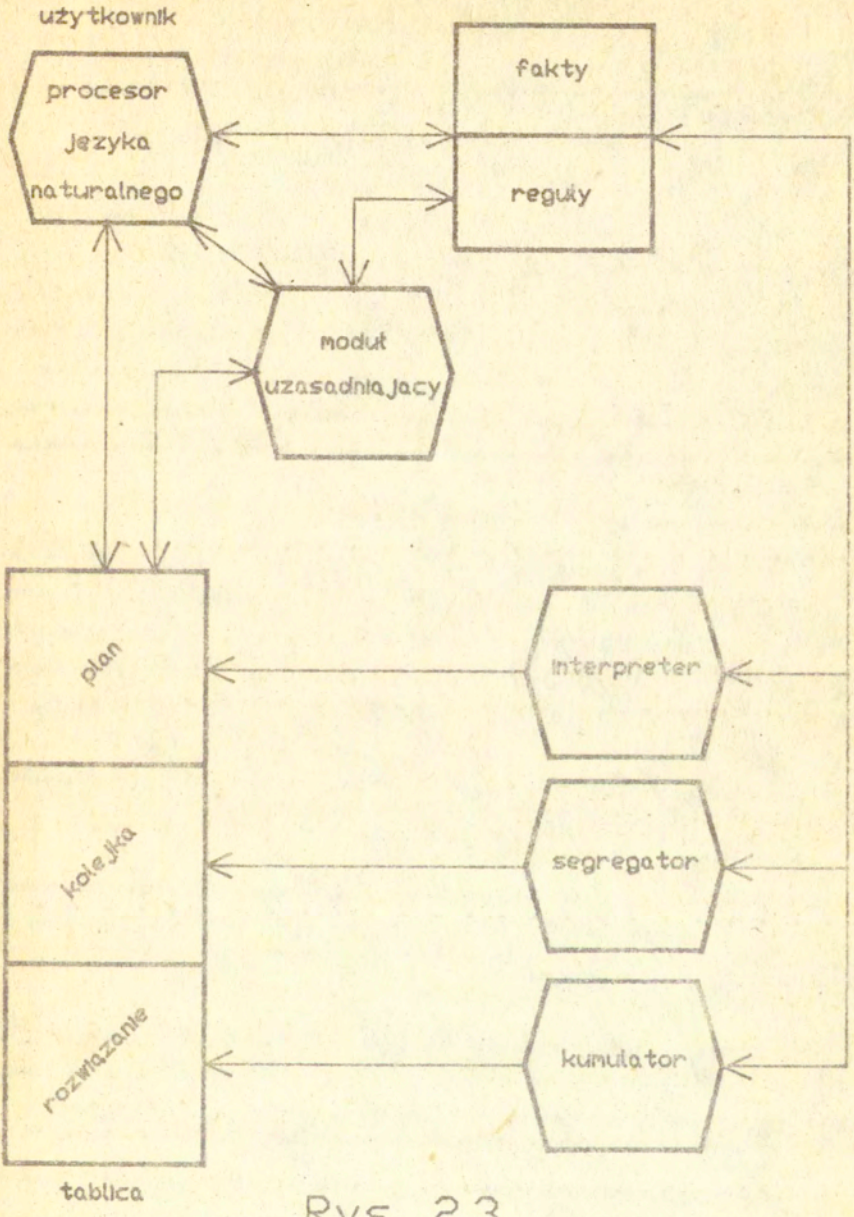
[2.8] ang. conflict resolution

W zależności od przyjętej strategii wnioskowania wyróżnia się dwa rodzaje interpreterów: wnioskujące wprzód i wnioskujące wstecz. Interpretery działające wprzód wychodzą ze zbioru znanych faktów, w kolejnych cyklach wykonują wszelkie znalezione reguły i kończą działanie gdy zapas aktywnych reguł zostanie wyczerpany, lub spełniony zostanie inny warunek zakończenia. Strategia "wprzód" stosowana jest z powodzeniem w systemach analizy danych doświadczalnych i w ogólności we wszelkich zagadnieniach w których mamy do czynienia z ograniczoną liczbą danych znanych już w momencie rozpoczęcia pracy systemu.

Strategia "wstecz" polega na określeniu pewnych faktów jako celów działania systemu i sprawdzaniu prawych stron reguł, w odróżnieniu od strategii "wprzód", w której sprawdza się lewe strony. W momencie znalezienia reguły zawierającej po prawej stronie cel główny, przechodzi się do analizy lewej strony. Jeżeli nie wszystkie warunki po lewej stronie są jeszcze sprawdzone, to stają się one aktualnymi celami i procedurę wywołuje się rekurencyjnie. Strategia "wstecz" umożliwia w naturalny sposób sterowanie dialogiem z użytkownikiem. Pytania zadawane są gdy dla kolejnego celu niższego rzędu brakuje reguły, która mogłaby być spełniona. Można też nałożyć ograniczenie na głębokość rekurencji lub przypisać poszczególnym faktom atrybuty określające, czy system ma o nie pytać, czy też próbować wpierw określić je z innych praw.

System reguł produkcji niewątpliwie spełnia postulat modularności reprezentacji wiedzy: wymiana lub dodanie nowej reguły nie zakłóca działania pozostałych, gdyż poszczególne reguły komunikują się pomiędzy sobą pośrednio poprzez kontekst. Zaletą jest też jednolitość reprezentacji i jej naturalność - w całym szeregu zagadnień człowiek posługuje się przepisami postępowania zbliżonymi do tego typu reguł.

Systemy regułowe nie są wolne jednakowoż od wad: składanie długich ciągów postępowania z elementów okazuje się często nieefektywne i pożądane jest wówczas skrócenie procesu wnioskowania przez przeskok do konkluzji za pomocą z góry określonych sekwencji reguł. Równoprawność reguł w "czystych" systemach reguł produkcji utrudnia zapis algorytmicznych elementów wiedzy, zaradzić temu można poprzez wprowadzenie hierarchii reguł i umożliwienie wywoływania procedur i programów przez prawe strony. Wszelkie tego typu modyfikacje prowadzą jednak do niejawnego zawarcia pewnych elementów sterowania w samej organizacji bazy reguł i tym samym pozostają w sprzeczności



Rys. 2.3

Wyidealizowana struktura systemu ekspertowego.

z deklaratywnym charakterem zapisu.

2.3 Systemy ekspertowe.

Zastosowania systemów ekspertowych, pierwotnie ograniczone do dziedzin wojskowości i medycyny są obecnie bardzo różnorodne. Można powiedzieć, że systemy ekspertowe wykorzystuje się wszędzie tam, gdzie intensywnie wykorzystuje się wiedzę ludzką i gdzie dostęp do tej wiedzy jest utrudniony. Przykładowe zastosowania metodologii ES sięgają od zaganień interpretacji danych (PROSPECTOR [29] - identyfikacja formacji geologicznych) poprzez diagnozowanie (MYCIN [91] - zakażenia krwi), kontrolowanie procesów (Ventillation Manager [33] - sterowanie sztucznym oddychaniem), planowanie (MOLGEN [95] - planowanie eksperymentów w genetyce molekularnej) do projektowania (R1 [62] - konfigurowanie komputerów VAX).

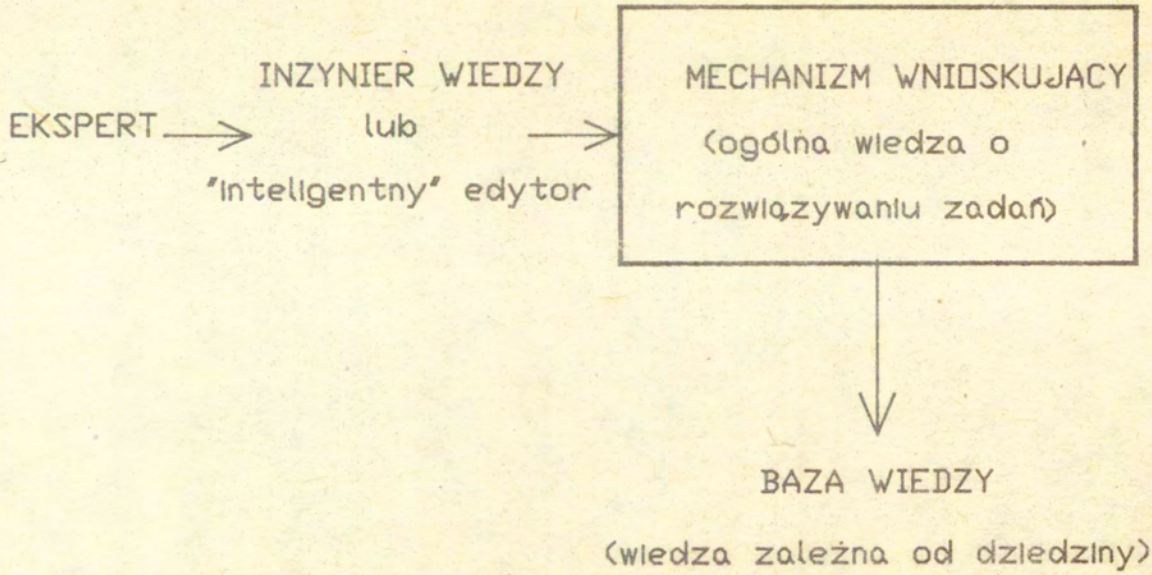
Kluczowymi i w znacznym stopniu niezależnymi zagadnieniami technologii ES są: akwizycja, reprezentacja i przetwarzanie wiedzy. To rozdzielenie zadań znajduje odzwierciedlenie w strukturze zdecydowanej większości zrealizowanych ES prowadząc jednocześnie do wyraźnego rozróżnienia pomiędzy ES a programami tradycyjnymi. Podstawowe pojęcia terminologii ES to: baza wiedzy, mechanizm wnioskujący i akwizycja wiedzy.

Bazą wiedzy nazywamy część programu zawierającą zgodny z przyjętym formalizmem zapis wiedzy .

Część logiczną programu, której zadaniem jest przetwarzanie wiedzy nazywamy mechanizmem wnioskującym. Pobieranie i kodowanie wiedzy odbywa się przy pomocy modułu akwizycji wiedzy, realizowanego z reguły pod postacią odrębnego programu.

Na rysunku (2.3)[23]pokazano wyidealizowaną strukturę systemu ekspertowego. W prawdzie żaden z istniejących systemów nie zawiera wszystkich wyróżnionych w schemacie elementów, tym niemniej w każdym ES występuje przynajmniej jeden z nich.

Baza wiedzy zawiera wiedzę pasywną - fakty, proceduralną - prawa heurystyczne oraz metawiedzę, t.j. wiedzę o wykorzystaniu faktów i praw.



Rys. 2.4
Schemat tradycyjnego procesu akwizycji wiedzy.

Ważnym elementem współczesnych ES jest tablica [2.9]. Jest to struktura danych rejestrująca w sposób jawny hipotezy i pośrednie rezultaty podlegające przetwarzaniu. Tablica składa się z trzech elementów: planu, kolejki i rozwiązania. Elementy planu określają ogólną strategię rozwiązania danego zagadnienia t.j. koncepcje, aktualne cele i konteksty. Kolejka [2.10] zawiera listę akcji gotowych do wykonania. Rozwiązanie zawiera zbiór wygenerowanych konkluzji i hipotez wraz ze wzajemnymi powiązaniami.

Do obsługi tablicy służą następujące programy: segregator, interpreter i kumulator. Segregator [2.11] zawiaduje kolejką poprzez nadawanie priorytetów poszczególnym działaniom na zasadzie ogólnych reguł wnioskowania i w powiązaniu z planem. Interpreter realizuje wybrany element kolejki poprzez wykonanie odpowiedniego prawa z bazy wiedzy. Kumulator [2.12] podtrzymuje spójność wyprowadzanych rozwiązań, dokonuje rewizji hipotez gdy pojawiają się nowe dane.

Moduł uzasadniający [2.13] tłumaczy użytkownikowi przesłanki działań podejmowanych przez system. Procesor językowy umożliwia komunikowanie się z systemem poprzez pewien podzbiór języka naturalnego.

Przyjęte na wstępie określenie ES spełnia szersza klasa systemów, w tym również programy bez wyróżnionej w sposób jawny bazy wiedzy jak np. zaawansowany system manipulacji symbolicznej MACSYMA.

2.4 Akwizycja wiedzy.

Jedną z zasadniczych trudności występujących przy budowie systemów ekspertowych jest akwizycja wiedzy. Przyjmuje się najczęściej, że źródłem wiedzy jest ekspert - osoba, która osiągnęła dużą sprawność w rozwiązywaniu zagadnień z danej dziedziny. Można ponadto posłużyć się danymi empirycznymi, szczegółowymi opracowaniami lub innymi źródłami wiedzy pierwotnymi w stosunku do wiedzy eksperta. Proces przetwarzania wiedzy źródłowej do postaci zgodnej z przyjętym formalizmem bazy wiedzy odbywa się przy pomocy metod inżynierii wiedzy lub automatycznie przez program rys(2.4)[23].

2.9 ang. blackboard
2.10 ang. agenda
2.11 ang. scheduler
2.12 ang. consistency enforcer
2.13 ang. justifier

Przy przyjęciu takiego modelu procesu akwizycji wiedzy ekspert spełnia rolę dedukcyjną: z ogólnych zasad (np. prawa Newtona) wyprowadza zasady szczegółowe (np. prawa ruchu planet Keplera). Fakty i reguły postulowane przez eksperta w trakcie akwizycji wiedzy z natury rzeczy odnoszą się do sytuacji wyidealizowanych, przywoływanych przez eksperta z pamięci, lub hipotetycznych. Inżynier wiedzy musi więc być świadomy, że:

- pobrana wiedza może nie być adekwatna w szczególnym kontekście, aktualnym w chwili jej wykorzystania,

- mogą wystąpić problemy z przewidywaniem a priori sytuacji wyjątkowych,

- łatwiej jest rozpoznać błędną decyzję w sytuacji rzeczywistej niż postulować ją hipotetycznie,

- eksperci mają z reguły trudności z określaniem numerycznych wartości współczynników pewności.

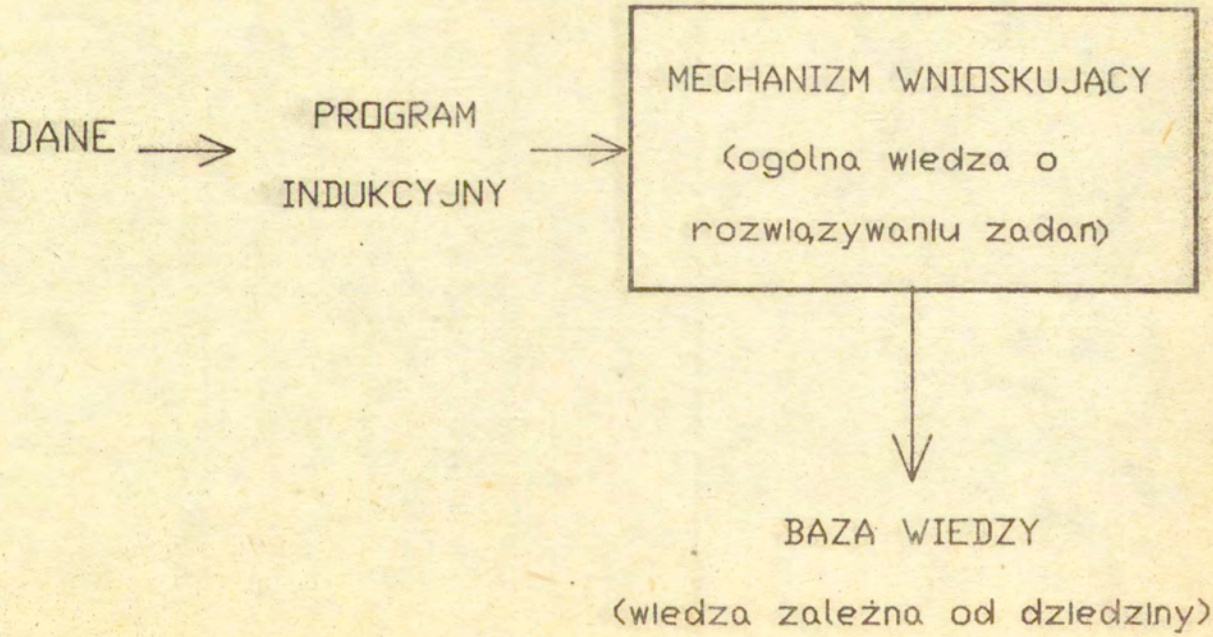
Metody przezwyciężania trudności powyższego typu wypracowane przez inżynierię wiedzy polegają m.in. na:
- odpowiednim doborze strategii przepytywania eksperta,
- precyzyjnym definiowaniu celów każdego etapu procesu akwizycji wiedzy,
- użyciu wyspecjalizowanych narzędzi akwizycji wiedzy,
- rejestracji działań eksperta w trakcie wykonywania rzeczywistej pracy.

Ponadto, najsłabszym ogniwem metody dedukcyjnej jest niewątpliwie sama osoba eksperta. W literaturze [23],[63],[65] szczegółowo rozważane są takie problemy jak:

- trudności z ustaleniem kompetentnego eksperta,

- podtrzymanie zaangażowania emocjonalnego eksperta przez cały okres realizacji projektu ES

- zła wola wynikająca m.in. z niechęci eksperta do rozpowszechniania jego wiedzy,



Rys. 2.5
Schemat zautomatyzowanego procesu akwizycji wiedzy.

– problemy z nawiązaniem wspólnego języka pomiędzy inżynierem wiedzy a ekspertem,

i wiele innych.

W związku z trudnościami występującymi przy metodzie dedukcyjnej poszukuje się alternatywnych, możliwie zautomatyzowanych rozwiązań. Duże nadzieje wiąże się z podejściem indukcyjnym. Polega ono na odwróceniu kierunku procesu logicznego metody dedukcyjnej. W przeciwieństwie do niej, zamiast przechodzić od praw o charakterze ogólnym do kontekstowo zależnych reguł, indukcja polega na poszukiwaniu praw rządzących danym zjawiskiem na podstawie zbioru przykładów. Najistotniejszą cechą indukcji jest możliwość jej automatyzacji. Symulacja procesu poszukiwania pewnych ogólnych zasad opisujących szczególne przypadki prowadzi do modelu jak na rys(2.5)[23].

Znajdowanie regularności w dużych bazach danych wymaga istotnej wiedzy podstawowej z rozpatrywanej dziedziny oraz technik statystycznych. Trudno w związku z tym liczyć na zupełne wyeliminowanie roli eksperta poprzez pełną automatyzację procesu akwizycji wiedzy. Tym niemniej zrealizowane zostały programy (AQ11, META - DENDRAL, EXTRAN) oparte na metodzie indukowania praw ocharakterze ogólnym w oparciu o przykłady testowe. Istota automatycznej indukcji srowadza się do wyszukiwania takich cech wspólnych dla przykładów pozytywnych, które nie występują w przykładach negatywnych.

Uzasadnienie potrzeby sprzężenia metod symbolicznych z numerycznymi.

Zautomatyzowanie projektu skomplikowanego urządzenia, przykładowo, samolotu wymaga integracji metod obliczeń numerycznych takich jak obliczeniowa dynamika płynów, rachunek prawdopodobieństwa, statystyka i modelowanie podzespołów z metodami manipulacji symbolicznej takimi jak zarządzanie bazami danych, generacja testów, propagacja ograniczeń i planowanie. Proces projektowania nie może być wpełni zautomatyzowany bez efektywnego sprzężenia programów symbolicznych z numerycznymi.

Wiele zadań mniej złożonych jak np. rozwiązywanie nieliniowych układów równań może się okazać nieosiągalne tradycyjnymi metodami symbolicznymi lub numerycznymi.

Ograniczenia poszczególnych metod numerycznych można często przezwyciężyć, jeżeli kilka różnych metod zostanie zastosowanych we właściwej kolejności. W wielu wypadkach tego rodzaju zdolność może zostać zawarta w systemie z bazą wiedzy zdolnym zarządzać technikami numerycznymi.

Często sprzężenie nie jest konieczne, lecz może ono udoskonalić zarówno proces symboliczny jak i numeryczny. Dla przykładu można obyć się bez inteligentnych interfejsów lecz ich wykorzystanie może wielce poprawić użyteczność programów numerycznych. Programy numeryczne mogą zostać ulepszone przez zastosowanie niederministycznych strategii rozwiązania, zdolności tłumaczenia oraz przez użycie technik programowania takich jak propagacja ograniczeń, sieci semantyczne i ramy. Obliczenia symboliczne mogą być czasem udoskonalone przez zakodowanie wiedzy proceduralnej pod postacią algorytmów numerycznych (algorytmów zdolnych zastąpić obliczeniowo intensywne procedury przeszukiwania przestrzeni stanów) oraz przez większą dokładność procesów numerycznych.

podsumowanie

Obserwacja tendencji i tempa rozwoju sprzętu i oprogramowania narzędziowego sugeruje, że sprzężenie metod symbolicznych z numerycznymi jest nieuchronnym kierunkiem rozwoju. Kierunkiem tym bardziej oczywistym, że zgodnym z potocznym pojęciem o roli komputera. Ani metody symboliczne ani metody numeryczne oddzielnie nie mogą rozwiązać wszystkich zagadnień projektowania i analizy. Złożone zadania nie mogą być rozwiązane przy pomocy metod czysto symbolicznych czy też numerycznych oddzielnie. W takich sytuacjach uzyskanie rozwiązania wymaga metod mieszanych.

Nawet jeżeli posiadamy pełen model matematyczny sytuacji, to może on w wielu zadaniach okazać się sam przez się niewystarczający, ponieważ interpretacja wartości numerycznych poszczególnych zmiennych problemu wymaga rozumowania jakościowego. Tak więc rozwiązanie złożonych zagadnień często zmusza nas do przełączania się na zmianę pomiędzy analizą formalną i wnioskowaniem jakościowym.

ROZDZIAŁ 3
PRZEGLĄD ZASTOSOWAŃ METOD SZTUCZNEJ INTELIGENCJI WE WSPÓŁCZESNEJ MECHANICE.

3.1 Manipulacja symboliczna

Najwcześniejszym obszarem zastosowań metod związanych ze sztuczną inteligencją do zagadnień mechaniki była automatyzacja przekształceń algebraicznych. Pierwsze prace dotyczyły różniczkowania analitycznego przy pomocy maszyny cyfrowej. Następnym, ważnym kierunkiem rozwoju stała się budowa wyspecjalizowanych języków manipulacji symbolicznej. Obejmowały one on m.in. przekształcenia układów współrzędnych, rachunek tensorowy, całkowanie i różniczkowanie symboliczne oraz inne operacje przydatne przy wyprowadzaniu zależności algebraicznych. Tabela 3.1 [69] wymienia najważniejsze spośród udokumentowanych zastosowań systemów manipulacji symbolicznej do zagadnień mechaniki.

Zastosowanie	*Użyty system*	*Bibliografia*
Generacja macierzy elementów skończonych	*FORMAC*	*(27,75,76)*
	MACSYMA	*(4,5,8,9,53,67,68)*
	CONFORM	*(103)*
	INNE	*(24,46,59)*
Zastosowanie metody Rayleigha-Ritza do:		
Statycznej analizy laminowanych powłok osiowo symetrycznych	*FORMAC*	*(101)*
Lokalnej i globalnej utraty stateczności cienkościennych kolumn walcowych	*MACSYMA*	*(25)*
Analizy drgań laminowanych, eliptycznych płyt kompozytowych	*MACSYMA*	*(6,7)*
Sprzężenie symetrycznego i	*REDUCE*	*(56,57)*

i niesymetrycznego wyboczenia powłok cylindrycznych w warunkach pełzania		
Zależne od częstości macierze mas i sztywności tarcz	*MACSYMA*	*(47)*
perturbacyjna procedura analizy stanów pokrytycznych konstrukcji sprężystych	*REDUCE*	*(88)*

Tabela 3.1

Spis udokumentowanych zastosowań skomputeryzowanej manipulacji symbolicznej do zagadnień teorii konstrukcji.

MACSYMA opracowany w MIT jest dużym systemem (wymaga min. 2 Mb pamięci operacyjnej) praktycznie wykorzystywanym w codziennej pracy przez setki osób poprzez międzyuczelnianą sieć komputerową. Mikrokomputerową namiastką MACSYMy jest system MUMATH opracowany w języku MUSIMP będącym wersją LISP-u na komputery osobiste IBM. Wykonuje on operacje rachunku macierzowego, różniczkowego i całkowego w zakresie odpowiadającym programowi szkoły średniej.

Z punktu widzenia mechaniki numerycznej szczególnie interesujące są zastosowania systemów automatycznych przekształceń symbolicznych do generacji macierzy sztywności elementów skończonych.

3.1.1 Macierze elementów skończonych na symbolach ogólnych

Zagadnienie obliczania macierzy sztywności elementów skończonych na symbolach ogólnych jest szczególnie istotne w kontekście :

- oszczędności czasu i pamięci maszyny,
- dokładności obliczeń
- zaufania użytkownika do oprogramowania.

Główną atrakcję automatyzacji operacji symbolicznych, zwłaszcza przy stosowaniu mikrokomputerów, stanowi zredukowanie liczby operacji zmiennoprzecinkowych, związane z faktem, że całkownie macierzy sztywności w tradycyjnych systemach ma miejsce niezależnie dla każdego elementu konstrukcji i jest ponawiane w każdym przebiegu pro-

gramu dla aktualnej geometrii. Tymczasem całkowanie na symbolach ogólnych można wykonać jednokrotnie dla danego typu elementu. Współrzędne węzłów i parametry materiałowe stają się zmiennymi w uzyskanych wzorach analitycznych. Tak opisane macierze składowane są w pamięci zewnętrznej maszyny w postaci odpowiednich nakładek programowych (overlay-ów).

Dodatkową korzyścią z całkowania analitycznego w odniesieniu do całkowania numerycznego, jest wzrost dokładności wyników końcowych. Tym niemniej, czas generacji macierzy symbolicznej może być znaczny i w przeciwieństwie do operacji algebraicznych jest z reguły silnie zależny od danych wejściowych a w tym od możliwości uproszczenia ułamków, stopnia złożoności wyrażeń pośrednich i innych elementów związanych z efektywnością przekształceń analitycznych.

W związku z powyższym rozpatruje się [69] możliwość łączenia metody analitycznej z numeryczną. Na najbardziej czasochłonnym etapie generacji macierzy sztywności jakim jest całkowanie, rozpatruje się możliwość wprowadzenia całkowania numerycznego na symbolach ogólnych, zamiast pierwotnej koncepcji całkowania analitycznego. Autorzy [69] postulują, jako dalszy kierunek badań, automatyzację przejścia od ogólnego opisu tensorowego zjawiska do postaci wykorzystywanej w programie. W przypadku generacji macierzy np. elementu powłokowego, na wejściu należało by podać:

- funkcjonał w notacji tensorowej,
- równania powierzchni odniesienia dla powłoki,
- kształt i geometrię elementu skończonego,
- funkcje aproksymacyjne,
- równania konstytutywne.

zaś na wyjściu należało by się spodziewać efektywnej procedury (automatyczne programowanie) do obliczania macierzy charakterystycznych elementu powłokowego.

3.2 przegląd zrealizowanych systemów ekspertowych w mechanice

3.2.1 system HI-RISE

Utworzenie wstępnego projektu wysokiego budynku jest zadaniem systemu HI-RISE

[94]. Sformułowanie wstępnego projektu budynku następuje w wyniku typowego procesu heurystycznego, wymagającego przyjęcia całego szeregu uproszczających założeń. Projekt taki oparty jest na przybliżonych rachunkach oraz regułach wynikających z uogólnienia doświadczeń nabytych przy projektowaniu typowych budowli i służy za punkt wyjścia do bardziej precyzyjnych obliczeń. Powodzenie późniejszej optymalizacji w dużej mierze zależy od trafnego doboru tej pierwszej postaci projektu .

Autorzy przyjmują, że w momencie uruchomienia systemu etap planowania przestrzennego jest zakończony i wszelkie późniejsze zmiany w tym zakresie pozostawiają projektantowi. Pozwala to ograniczyć pracę systemu do analizy samej konstrukcji. Rozwiązanie prezentowane użytkownikowi przez system ma postać na tyle szczegółowego opisu wykonalnych struktur przestrzennych, by umożliwić dalsze modelowanie wybranej alternatywy za pomocą tradycyjnego systemu numerycznej analizy konstrukcji.

HI-RISE został zbudowany na bazie języka reprezentacji wiedzy PSRL [86] funkcjonującego w środowisku języka LISP. Baza danych systemu składa się z praw produkcji zgrupowanych po 2 do 16 w 28 zbiorach praw, zestawu 30 wzorców schematów oraz pewnej liczby funkcji w języku LISP. Przykładowa reguła nakładająca ograniczenia na modelowanie konstrukcji trójwymiarowej:

JEŻELI

liczba pięter > 40,

oraz

struktura przestrzenna opisana jest za pomocą ortogonalnych ram płaskich

TO

zmień koncepcję struktury przestrzennej.

ma w składni przyjętego formalizmu OPS8 następującą postać:

```
(p 3D-elim::2D-elim
   {<goal> (goal ^name "3D-elim" ^grid-item <3D-sys> ^status active)}
   {<3D-sys> (3D-lateral ^3D-description sub-2D $stories > 10)}
   -->
   (modify <goal> ^status eliminated))
```

Zakres zastosowań HI-RISE jest ograniczony do relatywnie niewielkiej klasy budowli gdyż uwaga autorów systemu skoncentrowana była na analizie formalizmów potrzebnych do rozwoju prototypowego systemu projektującego. W odróżnieniu od zagadnień reprezentacji, problematyka akwizycji wiedzy do systemów ekspertowych wspomagających projektowanie wstępne nie była przez autorów rozważana. Do wypełnienia bazy wiedzy posłużono się heurystykami opracowanymi na podstawie podręczników inżynierskich.

W niniejszej pracy szczególną uwagę poświęcimy systemom ekspertowym związanym z analizą konstrukcji za pomocą tradycyjnego systemu MES. Jedną z przeszkód w upowszechnianiu tradycyjych programów MES jest silna zależność efektywności obliczeń od kwalifikacji użytkownika. Przygotowanie parametrów nieliniowej analizy nawet niezbyt złożonej konstrukcji wymaga poza znajomością MES, doświadczenia w eksploatacji konkretnego programu a także intuicji. Wiedza tego rodzaju jest często subiektywna, fragmentaryczna. W związku z powyższym podjęto próby wyposażenia niektórych systemów MES, m. in. MARC, NASTRAN i SESAM 69 w systemy ekspertowe służące jako rodzaj inteligentnej instrukcji obsługi.

3.2.3 system SACON

System SACON [13] był pierwszym chronologicznie przykładem zastosowania do budowy nowego systemu ekspertowego gotowego mechanizmu wnioskującego, w tym wypadku modułu EMYCIN pochodzącego z systemu MYCIN. Zadaniem SACON-a jest ułatwienie obsługi systemu MARC [60].

MARC jest ogólnym programem MES służącym do analizy geometrycznie i fizy-

cznie nieliniowych zadań mechaniki, umożliwia uwzględnienie efektów termicznych oraz anizotropowych własności materiału. Mnogość opcji obliczeniowych sprawia, że biegłe posługiwanie się systemem wymaga ok. 2-3 lat szkolenia i praktyki [13].

Wiedza w systemie SACON jest reprezentowana przez zbiór praw produkcji złożonych z trójek (obiekt - atrybut wielowartościowy - wartość) z przypisanym współczynnikiem pewności. Przykładowe prawo [14] ma postać:

Rule050

Premise:

($AND ($SAME CNTXT MATERIAL (LIST OF METALS))

(BETWEEN CNTXT ERROR 5 30)*

(GREATERP CNTXT NO-STRESS 0.9)*

(BETWEEN CNTXT CYCLES 1000 100000)*

ACTION:

(CONCLUDE CNTXT SS-STRESS FATIGUE TALLY 1.0)

co jest formalnym zapisem związku:

JEŻELI

1.Materiał konstrukcji jest jednym z metali,

oraz

2.Procentowy błąd obliczeń zawiera się pomiędzy 5 a 30,

oraz

3.Bezwymiarowe naprężenie przekracza 0.9,

oraz

4.Liczba cykli przykładanego obciążenia zawiera się pomiędzy 1000 a 100000

WÓWCZAS

stwierdza się z całą pewnością (współczynnik 1.0), że

należy uwzględnić zmęczenie materiału.

Prawa są zorganizowane w grupach odnoszących się do obciążenia, materiału i t.d. Kontekst w formie drzewa jest budowany dynamicznie w trakcie konsultacji. Mechanizm wnioskujący działa wstecz (backward chaining). System rekomenduje strategię analizy w oparciu o wiedzę o możliwych rodzajach analizy, dopuszczalnym błędzie i ekstremalnych naprężeniach.

3.2.3 system SESCON [37,38]

Podobnie jak w przypadku systemu SACON moduł EMYCIN został wykorzystany do zbudowania systemu SESCON za pomocą którego użytkownik systemu SESAM 69 może uzyskać w trybie interaktywnym informacje o optymalnym doborze modułów programowych do konkretnego zadania.

Pakiet SESAM 69 zawiera programy analizy statycznej oraz drgań swobodnych i wymuszonych konstrukcji w zakresie liniowej teorii sprężystości. Z uwagi na zastosowania w projektowaniu dużych i skomplikowanych konstrukcji takich jak okręty i platformy wiertnicze SESAM 69 jest szczególnie dostosowany do techniki analizy wielopoziomej metodą superelementów.

SESCON został wyposażony w niewielką bazę danych liczącą około 35 praw dla 20 parametrów. Prawa mogą być wprowadzne do bazy wiedzy bezpośrednio w języku LISP lub w bardziej przystępnym języku ARL (Abbreviated Rule Language). Na wyjściu prawo może być dodatkowo przetłumaczone na język naturalny (angielski). Przykładowo

reguła wyrażona w języku naturalnym:

JEZELI

[Prawo ma zastosowanie do obciążeń i służy do doboru rodzaju analizy dla konstrukcji]

1) obszarem zadania jest continuum

2) dwuwymiarowy stan naprężenia

3) obciążenie powoduje zginanie

4) istotne naprężenia ścinające

TO

teoria płyt średniej grubości jest jedną z dopuszczalnych dla zadania.

w zapisie ARL przybierze postać:

IF

1) Construction=continuum

2) Dimension=2

3) Bending, and

4) Shear

THEN:

Class = thick - shell (1.0)

co odpowiada zapisowi w języku LISP:

PREMISE:

($AND(SAME CNTXT CONSTRUCTION CONTINUUM)

(SAME CNTXT DIMENSION 2)

(SAME CNTXT BENDING)

(SAME CNTXT SHEAR)

ACTION:

(CONCLUDE CNTXT CLASS THICK-SHELL TALLY 1000)

3.2.4 Inny preprocesor systemu MARC

Na nieco prostszych zasadach oparty jest opisany w pracy [80] system tworzenia zbiorów danych wejściowych i interpretacji wyników końcowych uzyskanych za pomocą pakietu MARC.

Interaktywny tryb pracy na etapie wprowadzania danych oraz automatyczny dobór wymaganego rodzaju analizy z pewnego podzbioru opcji dostępnych w systemie MARC umożliwia dokonywanie stosunkowo skomplikowanych obliczeń użytkownikom nie związanym na codzień z pracownią obliczeniową.

Baza wiedzy została utworzona z myślą o automatyzacji procesów:

- określenia rodzaju analizy

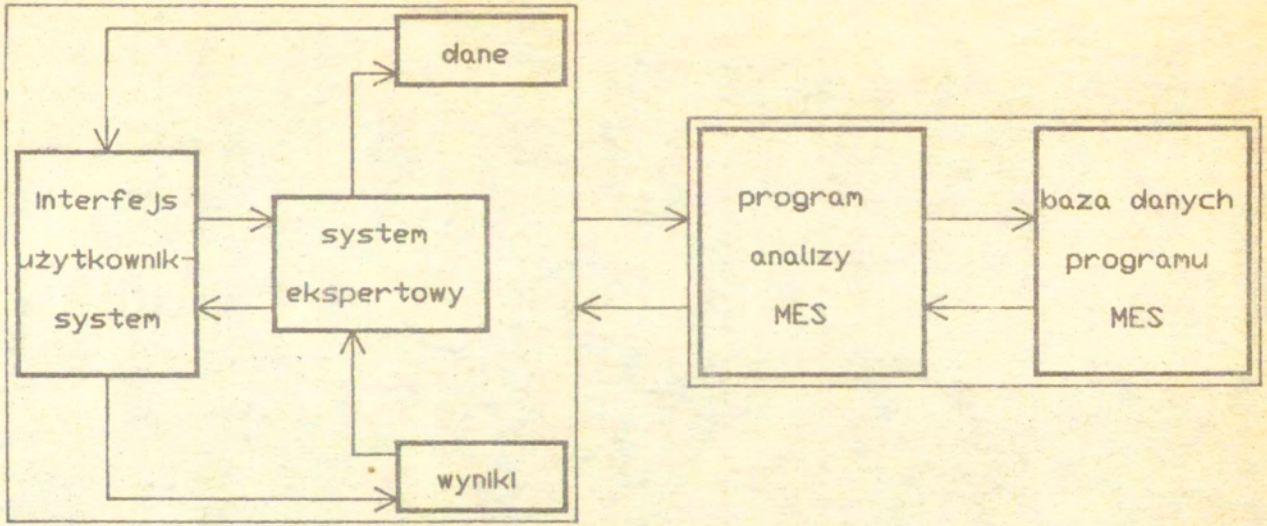
- przygotowania zbioru danych w postaci wymaganej przez system MARC

- oceny odpowiedzi konstrukcji na przyłożone obciążenia pod kątem ewentualnej ponownej analizy ze zmodyfikowanym modelem procesu fizycznego

W ramach przyjętej reprezentacji wiedzy, o składni pozostającej pod istotnym wpły-

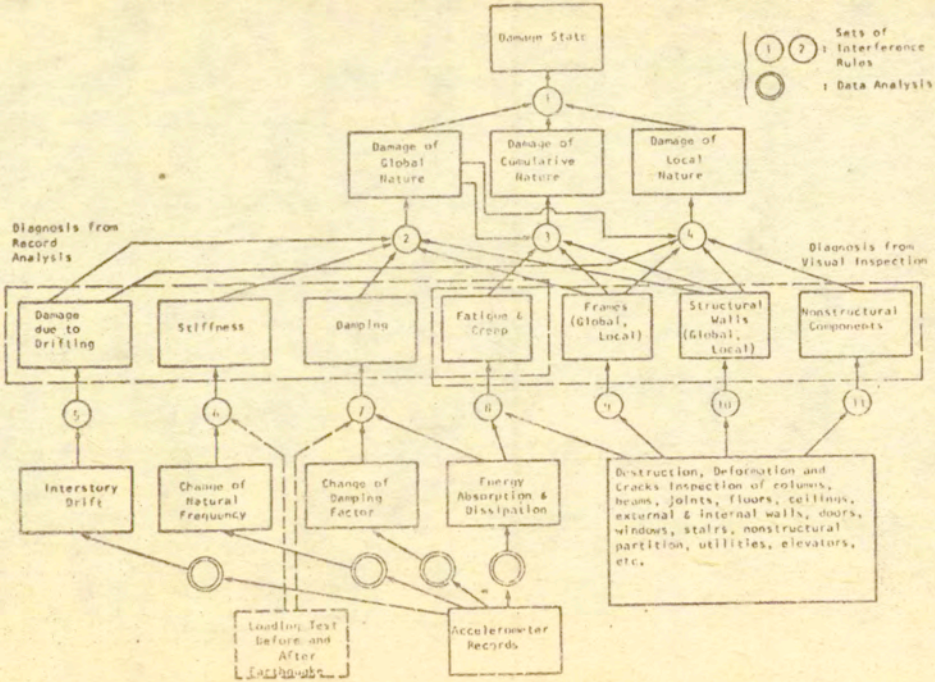
wem języka programowania FORTRAN, typowe prawo [80] zapisujemy w postaci:

```
RULE 213
IF TYPE IS 'THERMO-MECHANICAL'
IF TIME IS 'INDEPENDENT'
IF TEMPERATURE IS 'HIGH'
WRITE 0 :
FORMAT ('WRITE CREEP CARD ON INPUT2')
WRITE 2 :
FORMAT ('CREEP')
WRITE 2 :
FORMAT ('  0  -1  0  0     1.075E-26'/'  5.5')
CONCLUDE M_CARD_THREE IS 'GIVEN'
```



Rys. 3.1

Przepływ informacji w rozbudowanym systemie MARC.



Rys. 3.2 struktura reprezentacji wiedzy systemu SPERIL

co można zapisać nieformalnie:

JEŻELI

typ obciążenia jest termo-mechaniczny,

oraz

obciążenie jest stałe w czasie,

oraz

temperatura jest wysoka

TO

napisz na konsoli, że przewidujesz pełzanie materiału,

oraz

dopisz do zbioru danych wiersz 'pełzanie',

oraz

dopisz do zbioru danych wiersz z parametrami prawa konstytutywnego,

oraz

uwzględnij wykonane czynności w kontekście.

Baza wiedzy liczy ok. 75 takich praw. Zakłada się, że wszystkie prawa są jednakowo słuszne i wszystkie posiadają współczynnik pewności równy jeden. Mechanizm wnioskujący realizuje prostą strategię typu "backward chaining". Ze względu na potencjalne odwołania do istniejącego oprogramowania numerycznego, mechanizm wnioskujący został całkowicie napisany w FORTRANie.

Na rysunku (3.1)zobrazowany jest [94] przepływ informacji w rozbudowanym systemie MARC. Z punktu widzenia systemu ekspertowego, system MARC wraz z jego bazą danych został potraktowany jako tzw. czarna skrzynka z którą użytkownik komunikuje się jedynie za pośrednictwem systememu ekspertowego.

3.2.5 System SPERIL [50]

System ekspertowy oceny stopnia uszkodzenia budynków poddanych trzęsieniu zie-

mi na podstawie zapisów akcelerometrycznych, badań doświadczalnych oraz danych z inspekcji wzrokowej poszczególnych fragmentów konstrukcji. Wiedza eksperta jest przedstawiona w postaci sieci praw. Przyjęte podejście polega na dekompozycji złożonego zagadnienia na szereg prostszych podproblemów. Relacje pomiędzy podproblemami są hierarchiczne (dziedziczenie) lub równoległe. Dodatkowo, celem ułatwienia akwizycji wiedzy struktura podziału na podproblemy odpowiada podziałowi kompetencji pomiędzy ekspertami. Schemat reprezentacji wiedzy przedstawiony jest na rys (3.2).Każdy z numerowanych węzłów odpowiada zbiorowi praw, węzły zaznaczone podójnymi kółkami odpowiadają procesom analizy danych akcelerometrycznych.

przykładowe prawo:

JEŻELI:

obserwacja stanu ścian nośnych wskazuje na poważne uszkodzenia

oraz

konstrukcja jest wykonana ze zbrojonego betonu

TO:

zachodzi istotne prawdopodobieństwo,

że uszkodzenie ogólne konstrukcji jest znaczne.

zapisuje się następująco:

RULE #207

IF:

1) WAG is severe, and

2) MAT is reinforced concrete,

THEN:

there is considerable indication (0.5) that GLO is severe.

Ze względu na trudności związane z przedstawieniem wiedzy w sposób ścisły wyko-

rzystano teorię zbiorów rozmytych. Zdefiniowano stopień uszkodzenia konstrukcji w skali od 0 do 10. W związku z nieścisłym charakterem klasyfikacji wprowadzono interpretację stopnia uszkodzenia przez zmienne lingwistyczne lekki, umiarkowany, poważny, niszczący. Każda z werbalnie określonych klas jest związana z rekomendacjami i kosztem reperacji konstrukcji. Dla każdej z funkcji określono funkcję przynależności. Zbiór rozmyty "poważny stopień uszkodzenia" możemy przedstawić następująco:

$$\Omega_{poważny} = \{0|0.0, 1|0.05, 2|0.1, 3|0.4, 4|0.8, 5|1.0, 6|0.8, 7|0.4, 8|0.1, 9|0.05, 10|0.0\}$$

Zapis powyższy jest rozszerzeniem booleowskiej funkcji przynależności, która jest stosowana w klasycznej teorii zbiorów i oznacza, że 0 należy do zbioru Ω w stopniu 0.0, 1 należy w stopniu 0.05 itd.

Różnica pomiędzy teorią zbiorów rozmytych a rachunkiem prawdopodobieństwa polega na sposobie traktowania zdarzeń. O ile w rachunku prawdopodobieństwa zdarzenie jest pewne, natomiast poszukuje się miary pewności jego wystąpienia, to w teorii zbiorów rozmytych samo zdarzenie jest niepewne i może jednocześnie należeć do różnych klas. Z formalnego punktu widzenia różnica pomiędzy obu teoriami sprowadza się do odmiennych definicji operatorów koniunkcji i alternatywy.

podsumowanie

W dotychczasowych realizacjach systemów ekspertowych przeznaczonych dla użytkowników MES nie brano pod uwagę możliwości sprzężenia oprogramowania symbolicznego z numerycznym. Systemy ekspertowe działają niezależnie od programów MES, często na odmiennym sprzęcie przez co ich rola ogranicza się do pełnienia funkcji doradczej a praktyczne wykorzystanie, jeżeli w ogóle brane pod uwagę (systemy tego typu mają z reguły charakter eksperymentalny) może okazać się uciążliwe. Z tego względu, pomimo 10-letniej historii rozwoju zapoczątkowanej systemem SACON (r.1978), preprocesory ekspertowe nie znalazły szerszego, praktycznego zastosowania.

ROZDZIAŁ 4
OGÓLNA KONCEPCJA WYPOSAŻENIA SYSTEMU MES W ELEMENTY SZTUCZNEJ INTELIGENCJI

Pierwszy rozdział niniejszej pracy zawiera przegląd algorytmów rozwiązywania zagadnień brzegowo początkowych tradycyjnym systemem MES. Algorytmy te zapisane w formaliźmie rachunku macierzowego mają przejrzystą postać i dają się łatwo programować. Schemat przepływu informacji w tradycyjnym systemie MES przedstawia rys.4.1. Możemy wyróżnić trzy zasadnicze grupy procedur:

- bibliotekę elementów skończonych
- pakiet operacji na macierzach rzadkich
- pakiet przetwarzania danych.

Pod pojęciem biblioteki elementów skończonych rozumiemy zbiór wszystkich procedur operujących na pojedyńczych elementach. Pakiet operacji na macierzach rzadkich zawiera procedury składające, rozwiązujące układy równań liniowych i nieliniowych oraz wyznaczające wartości i wektory własne. Pakiet przetwarzania danych zarządza zasobami sprzętowymi: pamięcią operacyjną i zbiorami dyskowymi. Szczególnie istotne z punktu widzenia użytkownika są jego dwa elementy: preprocesor i postprocesor. Preprocesor jest to program służący do rozkodowywania zbiorów danych przygotowanych przez użytkownika. Preprocesor ma również za zadanie graficzną prezentację danych a także wyszukanie ewentualnych błędów. Postprocesor służy do prezentacji wyników końcowych.

O ile można mówić o standardowej metodologii programowania elementów pełniących funkcje numeryczne, to wiedza o prowadzeniu obliczeń poddaje się algorytmizacji w niewielkim stopniu. Najważniejsze wady istniejących systemów związane z ograniczeniem do tradycyjnych technik programowania to:

- duże i stale rosnące wymagania w stosunku do użytkownika
- ukrycie struktur sterowania w kodzie programu
- brak możliwości uwzględnienia zmieniających się koncepcji.

Kryteria podejmowania typowych dla procesu obliczeniowego decyzji o:

- modelowaniu
  - doborze prawa konstytutywnego
  - modelowaniu geometrii (podziale na elementy, zagęszczaniu podziału)
  - modelowaniu warunków brzegowych
  - modelowaniu obciążenia
  - doborze elementu (sposobu aproksymacji)
- doborze programów obliczeń liniowych
- wyborze algorytmu obliczeń nieliniowych
- doborze precyzji obliczeń
- sterowaniu procesem obliczeń nieliniowych
- doborze parametrów do restartu po przerwaniu obliczeń
- analizie dokładności wyników końcowych
- podejmowaniu decyzji o ponowieniu obliczeń.

są nieprecyzyjne i co więcej zmieniają się wraz ze zbieraniem doświadczenia i opracowywaniem nowych metod.

Koncepcja wyposażenia tradycyjnego systemu MES w elementy sztucznej inteligencji polega na sprzężeniu oprogramowania numerycznego z systemem ekspertowym zawierającym wiedzę o powyższych zagadnieniach. W odróżnieniu od systemów opisanych w rozdziale trzecim, które z reguły realizowane były pod postacią niezależnych programów i których rola ograniczała sie do funkcji doradczej proponowany system steruje całokształtem procesu obliczeniowego. W rozdziale piątym na przykładzie prototypu tego typu systemu przedstawiamy następujące korzyści wynikające z zastosowania metod

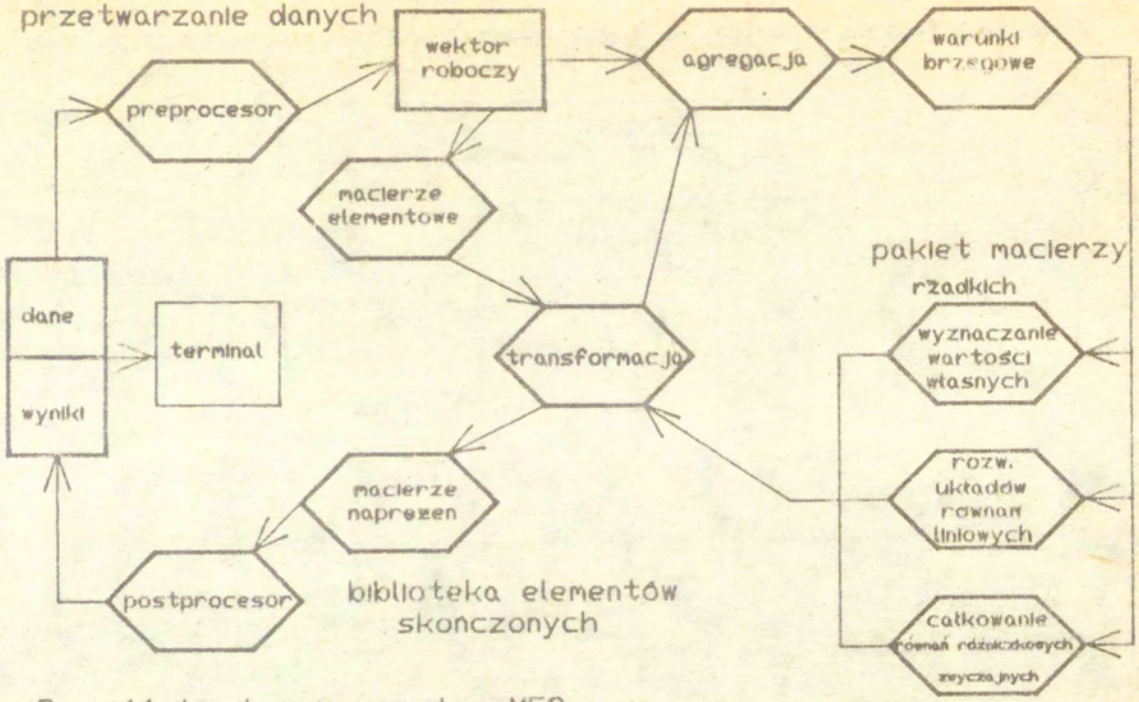
programowania AI do budowy modułu nadzorczego dla istniejącego systemu MES

- praktyczne eksperymenty z bazą wiedzy są możliwe już przy niewielkich bazach wiedzy (w granicach 50 reguł)
- narzędzia AI są przystosowane do przetwarzania wiedzy niepełnej i niejednoznacznej
- dzięki jawnej reprezentacji wiedzy program może ewoluować wraz ze wzrostem liczby rozwiązanych zadań
- istnieją możliwości wyjaśniania przez system przesłanek podejmowanych decyzji
- modyfikacja programu może się odbywać przez ingerencję użytkownika w zawartość bazy wiedzy lub automatycznie przez program.

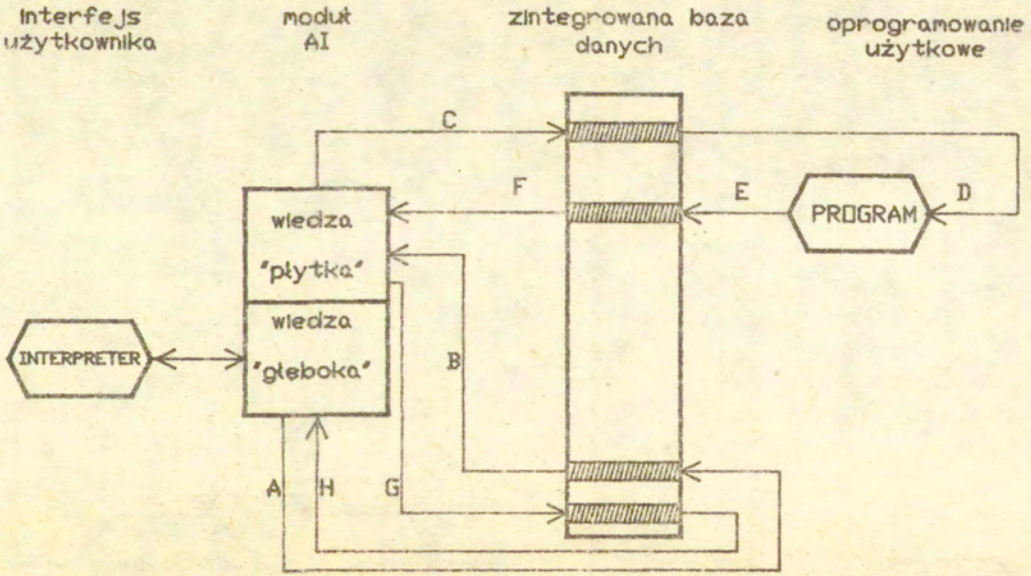
O tej ostatniej, szczególnie obiecującej możliwości piszemy w drugiej części niniejszego rozdziału poświęconej akwizycji wiedzy.

4.1 Zakres reprezentowanej wiedzy

Reprezentacja wiedzy proponowanego systemu wymaga modelu hybrydowego wiedzy o dwóch poziomach. Poziom płytki wiedzy obejmuje składnię zbiorów danych tradycyjnych programów MES i CAD oraz przepisy ich wywoływania. Z każdym obsługiwanym programem stowarzyszony jest zbiór parametrów sterujących i obserwowanych. Dla przykładu parametrami sterującymi programu obliczeń liniowych będą dane wejściowe (topologia podziału, stałe materiałowe etc.), natomiast parametrem obserwowanym może być pewna miara dokładności obliczeń. Na tym poziomie programy obliczeniowe traktowane są jako "czarne skrzynki", których funkcja nie jest reprezentowana jawnie. Przykładowa reguła płytkiego poziomu wiedzy dla programu analizy liniowej może mówić, że jeżeli zapisane są już fragmenty zbioru danych dotyczące postaci równania konstytutywnego, topologii podziału obszaru na elementy skończnoe, dyskretyzacji warunków brzegowych i obciążeń to należy dopisać do zbioru danych wiersz *KONIEC* i uruchomić



Rys. 4.1 tradycyjny system MES



Rys. 4.2 Przepływ informacji w systemie MES z elementami sztucznej inteligencji

odpowiedni program MES:

JEŻELI

ZAPIS (topologia O.K.)
ZAPIS (materiał O.K.)
ZAPIS (warunki brzegowe O.K.)
ZAPIS (obciążenia O.K.)

TO

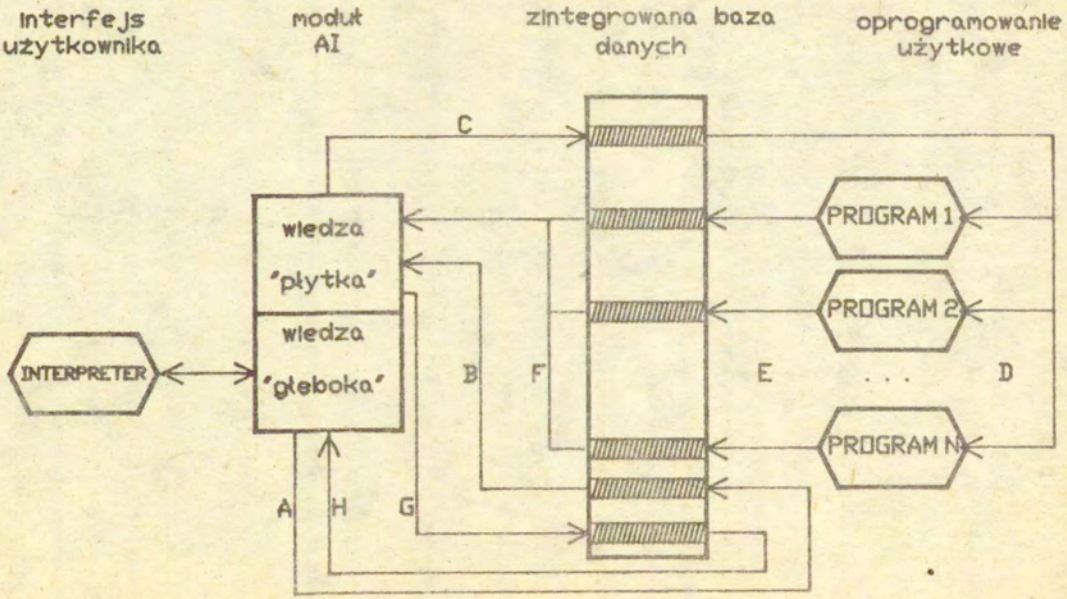
DOPISZ (DANE:KONIEC)
RUN (MES2S).

Wystąpienie nazwy programu po prawej stronie wykonywanej reguły, w tym wypadku MES2S jest nazwą programu analizy liniowej, oznacza zawieszenie działania mechanizmu wnioskującego systemu ekspertowego, przekazanie sterowania do programu numerycznego i wznowienie wnioskowania po zakończeniu obliczeń.

Poziom głębszy wiedzy to zbiór reguł korzystania z poszczególnych programów, a przede wszystkim wiedza o ustalaniu sekwencji niezależnych programów tak by realizować konkretne algorytmy. Dla każdej metody, na poziomie głębszym zapisane są przesłanki jej stosowania. Wiedza ta służy do interpretacji stanu analizy i doboru strategii dalszych obliczeń. Na tym poziomie funkcja poszczególnych programów i ograniczenia ich użycia reprezentowane są jawnie. O ile wiedza płytka dotyczy składni zbiorów danych i wyników obliczeń, to wiedza głęboka odnosi się do ich semantyki.

4.2 przepływ informacji w systemie

Przepływ informacji w systemie przedstawiony jest na rys.(4.2).Celem nadania bardziej przejrzystej formy graficznej w schemacie posłużyliśmy się dodatkowo koncepcją zintegrowanej bazy danych. Idea takiej bazy danych polega na wyodrębnieniu z programów numerycznych pakietów przetwarzania danych i nadaniu im wspólnej postaci. Dzięki temu poszczególne programy mogą operować na fizycznie tych samych obiektach. Realizacja tej koncepcji wymaga jednak przebudowy oprogramowania numerycznego i może mieć miejsce w praktyce jedynie przy opracowywywaniu nowego systemu od podstaw podczas gdy zaletą obecnej ogólnej koncepcji wyposażenia systemu MES w elementy AI jest nieingerencja w istniejące oprogramowanie.



Rys. 4.3 schemat przepływu informacji w wieloprocesorowym systemie sprzężonym

Litery na schemacie odpowiadają w porządku alfabetycznym kolejności wykonywanych działań. Użytkownik komunikuje się z systemem interakcyjnie poprzez interpreter głębszego poziomu wiedzy. Interpreter ten spełnia rolę powłoki ekspertowej nałożonej na zbiór programów poprzez zintegrowaną bazę danych. Jego sprawność będzie zależeć od ogólności i efektywności środowiska w jakim zostanie zbudowany. O ile do budowy zintegrowanej bazy danych potrzebny jest z uwagi na efektywność przetwarzania obiektów tradycyjny język programowania typu FORTRANu, to do budowy interfejsu użytkownika pożądane jest zastosowanie odpowiedniego systemu reprezentacji wiedzy zbudowanego na bazie jednego z języków AI - PROLOG, LISP, SMALLTALK - przy czym wybór konkretnego narzędzia powinien dotyczyć kompilatora - języki AI są najczęściej interpretowane - dającego możliwość łączenia z FORTRANem. Użycia FORTRANu pomimo jego znanych wad nie da się prawdopodobnie wykluczyć z uwagi na znaczną liczbę transmisji pomiędzy bazą danych a oprogramowaniem numerycznym wymaganych między innymi w skomplikowanych procesach iteracyjnych .

Na etapie formułowania zadania (na schemacie etap A) tworzony jest obiekt opisujący metodę rozwiązania zadania. Etap B polega na doborze konkretnych programów numerycznych realizujących zadanie sformułowane w ogólnych kategoriach. Wiedza płytka wykorzystywana jest do zapisu zbiorów danych w formacie wymaganym przez poszczególne programy oraz relizacji sekwencji rozkazów wymaganych do ich wywołania. Te wszystkie informacje zawarte zostają w obiekcie generowanym na etapie C. Na schemacie dla uproszczenia przedstawiono sytuację w której wystarcza użycie jednego programu. Program zostaje wywołany na etapie D. W trakcie przebiegu programu tworzone mogą być dodatkowe obiekty, które po zakończeniu przebiegu są kasowane lub pozostawiane jako skutki uboczne (macierze sztywności elementów itp.). Z punktu widzenia modułu AI istotny jest obiekt tworzony w fazie E działania systemu. Zawiera on informacje o sposobie zakończenia programu (przebieg normalny, przebieg z błędami itp.) oraz parametry do interpretacji przy pomocy wiedzy płytkiej na etapie F. Efektem interpretacji jest nowy obiekt powstający na etapie G zawierający sformułowanie wyników w znormalizowanej postaci nadającej się do przetworzenia przy pomocy wiedzy głębokiej na etapie H do postaci przedstawianej użytkownikowi. Kolejne cykle wywoływane są w miarę modyfikowania pierwotnych założeń przez użytkownika.

Kolejny rysunek (4.3)przedstawia pełniejszy schemat przepływu informacji. Litery

oznaczają etapy cyklu odpowiadające etapom z rys.(4.2). Różnica polega na tym, że teraz wiedza płytka jest wykorzystywana do zarządzania całym zbiorem programów. Decyzja o doborze konkretnego programu do realizacji określonego etapu analizy podejmowana jest w oparciu o moduł wiedzy głębokiej. Wiedza płytka służy do ustalania kolejności wykonywanych zadań, tworzenia zbiorów komunikacyjnych pomiędzy poszczególnymi programami, sygnalizacji błędów. Przykładowo, wiedza głęboka potrzebna jest do wyboru jednego spośród kilku programów wyznaczania wartości własnych. Wiedza płytka potrzebna będzie wówczas do przekodowania zapisu macierzy właściwego dla programu składającego z macierzy elementowych macierz globalną, do postaci wymaganej przez wybrany program numeryczny.

Porównanie schematów 4.1 z 4.2 oraz 4.3 wskazuje, że przetwarzanie w tradycyjnym systemie MES ma charakter sekwencyjny. Wszystkie ewentualności prowadzenia obliczeń równolegle muszą być przewidziane i zaprogramowane z góry. System z nadzorczym modułem ekspertowym ma odmienny charakter. Decyzja o uruchomieniu równoległych procesów może zostać podjęta w każdym momencie gdy okoliczności po temu sprzyjają. Może mieć to przykładowo miejsce gdy wykonywany jest czasochłonny etap generowania macierzy elementowych, zaś w sieci komputerowej znajdują się nieobciążone procesory. Wówczas system ekspertowy może obciążyć poszczególne procesory proporcjonalnie do ich możliwości i synchronizować przebieg obliczeń.

Poza ogólnymi zaletami wymienionymi na wstępie, proponowana koncepcja umożliwia wykorzystanie zbioru oprogramowania powstałego w środowisku akademickim, co było w tradycyjnych systemach komputerowych niemożliwe z uwagi na specyfikę tego oprogramowania, najczęściej powstającego dla celów konkretnej pracy i do własnego użytku autora, w wyniku czego charakteryzującego się z reguły następującymi niekorzystnymi cechami:

- z natury rzeczy bardzo ograniczonym zakresem zastosowania,
- szczególnie wysokimi wymaganiami odnośnie kwalifikacji użytkownika,
- nieprzewidzianym zachowaniem w wypadku zastosowania niezgodnie z nie zawsze jasno określonymi intencjami autora,
- "nieprzyjaznością dla użytkownika", w szczególności brakiem grafiki, generatorów siatek i t.d.,
- brakiem dokumentacji.

4.3 akwizycja wiedzy

Moduł akwizycji wiedzy powinien spełniać następujące postulaty:

integracja różnych źródeł wiedzy:

(i) wiedza książkowa z MES: wiedza ta jest nieodzowna przy generowaniu metod w oparciu o istniejące oprogramowanie, przy formułowaniu reguł określających miary efektywności poszczególnych metod.

automatyzacja czynności uciążliwych, wymagających specjalistycznego przygotowania:

(ii) instrukcje wprowadzania danych przedstawione w postaci drzew decyzyjnych: Wybór metody będzie się wiązał z reguły z uruchomieniem programu numerycznego. Programy numeryczne wymagają sformułowania zbiorów danych o ściśle określonym formacie. Zbiory danych mogą mieć najróżnorodniejszą postać zależną najczęściej od indywidualnych rozstrzygnięć autorów programu. W związku z tym zalecamy przedstawienie instrukcji wprowadzania danych w postaci drzewa decyzyjnego, a następnie zapis regułowy takiego drzewa.

formalizacja wiedzy eksperta:

(iii) konsultacje z ekspertem: Z reguły zakres ES będzie ograniczony przez dostępne oprogramowanie. Nie jest więc celowe podejmowanie próby zakodowania całej wiedzy eksperta odnoszącej się do MES. Celem konsultacji jest uzyskanie poglądu eksperta na stosowanie poszczególnych metod.

Z uwagi na specyfikę zagadnienia, trudności związane z pozyskaniem ekspertów do współpracy oraz wymierność kompetencji ekspertów w całym szeregu zagadnień należy również zapewnić możliwość automatycznego uczenia się programu.

4.3.1 akwizycja wiedzy przez automatyczną indukcję

Zrealizowane procesy numeryczne opisujemy przez podanie zbioru wartości atry-

butów. Dodatkowy atrybut określa stopień powodzenia zastosowanej metody. Algorytm automatycznej indukcji służy do generowania minimalnych drzew decyzyjnych wystarczających do opisu dowolnego zbioru C obiektów powyższych klas. Algorytmom wyznaczania drzew powyższego typu poświęcona jest praca [63]. Drzewa te, opisujące w najbardziej zwięzły sposób rejestrowane związki pomiędzy wartościami poszczególnych atrybutów danej klasy obiektów a wartością atrybutu klasyfikującego mogą być wykorzystane do rozbudowy bazy wiedzy głębszego rodzaju. Mechanizm algorytmu jest następujący:

1. Jeżeli C jest zbiorem pustym, (nie wygenerowano żadnego obiektu) to stowarzyszamy go arbitralnie z dowolną klasą. Zabieg ten ma charakter czysto formalny i służy do zakończenia procesu rekurencyjnego uruchamianego w punkcie trzecim.

2. Jeżeli wszystkie elementy zbioru C są tej samej klasy, wówczas otrzymujemy liść drzewa decyzyjnego i opisujemy go przez podanie nazwy klasy. Jest to oczywiste: jeżeli we wszystkich wypadkach zastosowania danej metody otrzymaliśmy wyniki pozytywne, lub we wszystkich negatywne, to wartość metody jest jednoznacznie określona z dokładnością do reprezentatywności wygenerowanych obiektów.

3. Jeżeli C zawiera elementy różnych klas, to dobieramy atrybut względem którego rozbijemy C na rozłączne zbiory obiektów $C_1, C_2, \ldots, C_n$, *gdzie* C_i zawiera elementy zbioru C odpowiadające $i-tej$ wartości wybranego atrybutu a następnie wywołujemy algorytm kolejno dla każdego podzbioru C_i. Atrybut wybieramy w taki sposób aby budowane drzewo klasyfikacyjne było minimalne.

Wynikiem procedury jest drzewo decyzyjne, którego każdy liść opisany jest przez wartość atrybutu klasyfikacyjnego, natomiast każdy węzeł wewnętrzny określa zarówno atrybut do przetestowania jak i gałęzie odpowiadające poszczególnym wartościom atrybutu.

Dla ilustracji przyjmijmy, że rozwiązujemy pewną liczbę zadań testowych pewnym hipotetycznym programem MES o którym wiemy jedynie, że ma potencjalnie nieograniczone możliwości obliczeniowe. Przykłady zaczerpnięte z rzeczywistej praktyki obliczeniowej zamieszczamy w następnym rozdziale. Do opisu procesu numerycznego przyjmijmy uproszczony generator obiektów o trzech atrybutach:

LSS – liczba stopni swobody;

wartości: duża, mała,

WYMIAR – liczba stopni swobody w węźle;

wartości: 1, 2 lub 3,

RODZAJ – rodzaj zadania;

wartości: *L* – liniowe, *N* – nieliniowe.

Dla kolejnych przykładów test zaliczamy w wypadku wykonania poprawnych obliczeń. Definiujemy więc dodatkowo atrybut klasyfikacyjny *WYNIK*:

WYNIK – atrybut klasyfikacyjny;

wartości: +, −.

Niech wyniki dla poszczególnych zadań przedstawiają się następująco:

1: kratownica płaska, 28 elementów, 20 węzłów, 40 stopni swobody, zadanie liniowe, wynik zgodny z rozwiązaniem dokładnym,

2: tarcza, 841 elementów, 900 węzłów, 1800 stopni swobody, deformacyjna teoria plastyczności, przekroczenie pamięci operacyjnej,

3: rozkład temperatur w próbce osiowosymetrycznej, 1423 elementy, 981 stopni swobody, liniowe równanie przewodnictwa, wynik zgodny w 92% z danymi doświadczalnymi,

4: kratownica przestrzenna, 30 elementów, 12 węzłów, 24 stopnie swobody, liniowa teoria sprężystości (LTS), błąd przebiegu programu,

5: zapora na rzece X, 18 elementów przestrzennych, 222 węzły, 666 stopni swobody, LTS, przekroczenie dostępnej pamięci operacyjnej,

6: tarcza z otworami, 400 elementów, 362 węzły, 724 stopnie swobody, LTS, wynik wiarygodny,

7: ruszt, 320 elementów 240 węzłów, 720 stopni swobody, plastyczność, wynik błędny,

8: pojedyńczy element tarczowy, 8 węzłów, 16 stopni swobody, nieliniowa sprężystość, błąd przebiegu programu.

Zgodnie z przyjętym zbiorem atrybutów możemy powyższe przykłady zapisać w postaci kolejnych obiektów:

NR	LSS	WYMIAR	RODZAJ	WYNIK
1.	MAŁA	2	L	+
2.	DUŻA	2	N	-
3.	DUŻA	1	L	+
4.	MAŁA	3	L	-
5	DUŻA	3	L	
6.	DUŻA	2	L	+
7.	DUZA	3	N	-
8.	MAŁA	2	N	+

Celem uzyskania minimalnego drzewa klasyfikacyjnego posłużymy się podejściem teorii informacyjnej pozwalającym określić oczekiwaną ilość informacji z zależności:

$$I_c = -\sum_{i=1}^{n} p(i) * log_2(p(i)),$$

gdzie:

I_c – zawartość informacyjna (w bitach)

n – liczba klas

$p(i)$ – prawdopodobieństwo wystąpienia i-tej klasy.

W związku z tym, że nie znamy wartości prawdopodobieństw, zastępujemy je względnymi częstościami. W naszym zbiorze obiektów mamy cztery przykłady klasyfikacji pozytywnej i cztery negatywnej:

$$I_c = \frac{-4}{8} log_2\left(\frac{4}{8}\right) - \frac{4}{8} log_2\left(\frac{4}{8}\right) = 1.0$$

Teraz określamy udział I_a kolejnych atrybutów w I_c. Do rozbicia na podzbiory wybierzemy atrybut dający największą wartość różnicy $(I_c - I_a)$.

Dla atrybutu "LSS" mamy:

LSS

gałąź DUŻA: $-\frac{2}{5} log_2(\frac{2}{5}) - \frac{3}{5} log_2(\frac{3}{5}) = 0.971$

gałąź MAŁA $-\frac{2}{3} log_2(\frac{2}{3}) - \frac{1}{3} log_2(\frac{1}{3}) = 0.918$

DUŻA	MAŁA
5 obiektów	3 obiekty
2 +	2 +
3 -	1 -

$IA = \frac{5}{8} * 0.971 + \frac{3}{8} * 0.918 = 0.951$

Informacja uzyskana z LSS: $1.0 - 0.951 = 0.049$

Dla atrybutu "WYMIAR" mamy:

WYMIAR

gałąź 3: $I_3 = 0.0$

gałąź 1: $I_1 = 0.0$

gałąź 2: $I_2 = 0.813$

3	1	2
3 obiekty	1 obiekt	4 obiekty
3-	1 +	3 +
		1 -

$I_a = \frac{3}{8} * 0 + \frac{1}{8} * 0 + \frac{4}{8} * 0.8133 = 0.405$

Informacja uzyskana z WYMIARu: $1.0 - 0.405 = 0.595$

Testowanie tą samą techniką atrybutu RODZAJ daje 0.049. Tak więc największą korzyść uzyskamy wybierając atrybut WYMIAR. Otrzymujemy drzewo jednopoziomowe:

WYMIAR

1	2	3
DUŻA L +	MAŁA L +	DUŻA N -
	DUŻA N -	MAŁA L -
	DUŻA L +	DUŻA L -
	MAŁA N +	

Obiekty gałęzi odpowiadającej wartości "1" atrybutu WYMIAR są jednej klasy ("+"), tak samo dla wartości "3" ("- "). Otrzymaliśmy więc dwa liście drzewa decyzyjnego. Dla wartości "2" mamy ciągle dwie klasy, więc powtarzamy procedurę i rozbijamy zbiór obiektów względem atrybutu RODZAJ. Otrzymujemy:

WYMIAR

1	2	3
+		- .
	L	N
	MAŁA +	DUŻA -
	DUŻA +	MAŁA -

I ostatecznie:

WYMIAR

1	2	3
+		-
	L	N
	+ MAŁA	DUŻA
	+	-

co jest graficznym odpowiednikiem sformułowania:

"Badany pakiet MES zachowuje się poprawnie w zadaniach jednowymiarowych, w zadaniach dwuwymiarowych zawodzi jedynie dla dużych zagadnień nieliniowych natomiast nie nadaje się zupełnie do zagadnień trójwymiarowych."

Powyższe drzewo możemy zapisać w postaci regułowej:

PRAWO 1:

Jeżeli WYMIAR = 1 To +.

PRAWO 2:

Jeżeli WYMIAR = 2 To

Jeżeli RODZAJ = L To +,

Jeżeli RODZAJ = N To

Jeżeli LSS = MAŁA To +

Jeżeli LSS = DUŻA To -.

PRAWO 3:

Jeżeli WYMIAR = 3 To -.

podsumowanie

Analiza procesów mechanicznych składa się z wielu etapów, spośród których automatyzacji pod postacią systemów komputerowych podlega w tradycyjnym ujęciu jeden - obliczenia numeryczne. Wprawdzie jest to etap najbardziej pracochłonny, w wypadku złożonych zagadnień wręcz niemożliwy do zrealizowania bez użycia komputera, to jednak wartość jego wyników zależy w decydującym stopniu od decyzji podjętych na etapie ustalania modelu dyskretnego.

Zastosowanie metod AI umożliwia komputeryzację nienumerycznych etapów modelowania procesów mechanicznych a także daje możliwość budowy modułów nadzorczych łączących różnorodne oprogramowanie inżynierskie. Koncepcja wykorzystania technik automatycznej akwizycji wiedzy prowadzi do systemów MES, które w kolejnych przebiegach będą dawać przy tych samych danych wejściowych coraz lepsze wyniki.

ROZDZIAŁ 5
REALIZACJA WYBRANYCH FRAGMENTÓW KONCEPCJI OGÓLNEJ

W niniejszej części pracy przedstawiamy próbę realizacji tych fragmentów koncepcji ogólnej, które uznaliśmy za kluczowe dla realizacji całości przedsięwzięcia. Eksperymentalne programy NANCON i INDUKTOR opracowane zostały celem zilustrowania i weryfikacji najistotniejszych stwierdzeń i postulatów zawartych w części trzeciej.

5.1 System ekspertowy NANCON.

Prototypowy system NANCON zarządza zbiorem programów liniowej i nieliniowej analizy konstrukcji w cyklu:

- wprowadzanie danych,
- wykonanie,
- analiza wyników końcowych,
- modyfikacja założeń,

oraz wykonuje niezbędne funkcje systemu operacyjnego przede wszystkim w zakresie obsługi pamięci zewnętrznej (zbiorów dyskowych). W trakcie pracy NANCON przekazuje sterowanie do programów zewnętrznych, pozostając w pamięci operacyjnej. Po zakończeniu pracy wywoływanego programu, NANCON automatycznie odzyskuje kontrolę zgłaszając się w stanie aktualnym w momencie wywołania programu zewnętrznego. W zakresie liniowej analizy sprężystej konstrukcji w płaskim stanie naprężenia lub odkształcenia wykorzystywany jest program MES2S zaś do analizy nieliniowej programy pakietu NANZAM.

5.1.1 reprezentacja wiedzy.

NANCON wykorzystuje reprezentację wiedzy w postaci reguł typu:

JEŻELI

$(warunek_{[1,1]}, warunek_{[1,2]}, \ldots, warunek_{[1,N]})$,

LUB

$(warunek_{[2,1]}, warunek_{[2,2]}, \ldots, warunek_{[2,N]})$,

...

...

...

LUB

$(warunek_{[M,1]}, warunek_{[M,2]}, \ldots, warunek_{[M,N]})$,

TO :

$(konkluzja_1, konkluzja_2, \ldots, konnkluzja_N)$,

W PRZECIWNYM RAZIE :

$(alternatywa_1, alternatywa_2, \ldots, alternatywa_N)$,

PONIEWAŻ :

(uzasadnienie),

NA PODSTAWIE :

(źródło uzasadnienia).

W warunkach, konkluzjach i alternatywach wykorzystywane są zmienne typu symbolicznego, n.p.:

dostępne typy analizy to:

1. statyka,
2. dynamika,
3. stateczność;

zmienne liczbowe (nie jest czynione rozgraniczenie pomiędzy liczbami całkowitymi a rzeczywistymi) n.p.: liczba elementów; zmienne tekstowe n.p.: identyfikator zadania. Część zmiennych jest inicjowana w momencie wywołania systemu.

Warunki symboliczne wymagają dokonania wyboru z przedstawionej listy możliwości. Warunki z wykorzystaniem zmiennych liczbowych buduje się przy pomocy operatorów <, >, <=, >=, = . Do zmiennych liczbowych stosuje się podstawowe operatory

algebraiczne $(+, -, *, /)$, funkcje trygonometryczne i logarytmiczne. Warunki typu tekstowego wymagają wprowadzenia łańcucha znaków. Zmienna reprezentowana jest poprzez nazwę, rodzaj, tekst określający jej znaczenie, wzorzec pytania o wartość zmiennej oraz wartość początkową, jeżeli taka jest zakładana.

W części wykonawczej, konkluzje poza nadawaniem wartości zmiennym mogą dotyczyć wywoływania programów zewnętrznych.

5.1.2 mechanizm wnioskujący

NANCON wykorzystuje dwuwartościową, monotoniczną logikę boolowską. Przeszukiwanie przestrzeni rozwiązań odbywa się poprzez kombinację jawnej strategii BACKWARD CHAINING i niejawnej FORWARD CHAINING. Strategia niejawna polega na automatycznym wykonywaniu wszelkich reguł, których warunki są w danej chwili spełnione. Strategia wykonywana jawnie jest podporządkowana aktualnemu celowi, którym jest początkowo pozytywne przeprowadzenie obliczeń numerycznych. Wywoływane są reguły, które ten cel zawierają w części wykonawczej. Jeżeli nie wszystkie warunki są jeszcze spełnione w wyniku działania innych reguł, początkowych wartości zmiennych i działania programów zewnętrznych, to warunki niespełnione stają się aktualnymi celami i procedura jest powtarzana rekurencyjnie.

Po zakończeniu obliczeń użytkownik może zrewidować dowolne spośród podjętych decyzji i uruchomić mechanizm wnioskujący na nowo, przy czym będzie pytany jedynie o brakujące dane, potrzebne do realizacji nowej ścieżki. Jeżeli przykładowo, po zakończeniu pierwszej analizy dotyczącej stanu sprężystego, użytkownik zmieni model materiału na plastyczny, to zapytany zostanie dodatkowo tylko o granicę plastyczności, wartość modułu stycznego i charakter nieliniowości (przykład 1). Nowy zbiór danych NANZAMu zostanie wygenerowany w oparciu o zaktualizowane dane, kontrola zostanie czasowo przekazana do pakietu numerycznego i nowe wyniki wyświetlone na ekranie.

Uzasadnianie zadawanych pytań i podejmowanych decyzji polega na wyświetlaniu na żądanie aktualnie wykonywanych reguł, ich części objaśniającej i źródłowej, aktualnej hierarchii celów i reguł do których spełnienia są one potrzebne.

5.1.3 przykładowy dialog.

Poniższy dialog przeprowadzony dla zadania z jednym elementem skończonym (rys. 5.1) celem demonstracji cyklu obliczeniowego sterowanego systemem ekspertowym. Odpowiedzi użytkownika wyróżniono pismem pochyłym.

SYSTEM SPRZĘŻONY
ANALIZY KONSTRUKCJI METODĄ ELEMENTÓW SKOŃCZONYCH
PIOTR BREITKOPF, kwiecień 1987

Naciśnij dowolny klawisz:

Podaj Nazwa zbioru danych

: *DATA*

Podaj Identyfikator zadania

: *TARCZA PSN/1 - 1*

TARCZA PSN/1 - 1 :

1 Nowe zadanie

2 Restart

1

Podaj numer(y) wartości, DLACZEGO by uzyskać informację o regule,

typ analizy :

1 statyka

2 dynamika

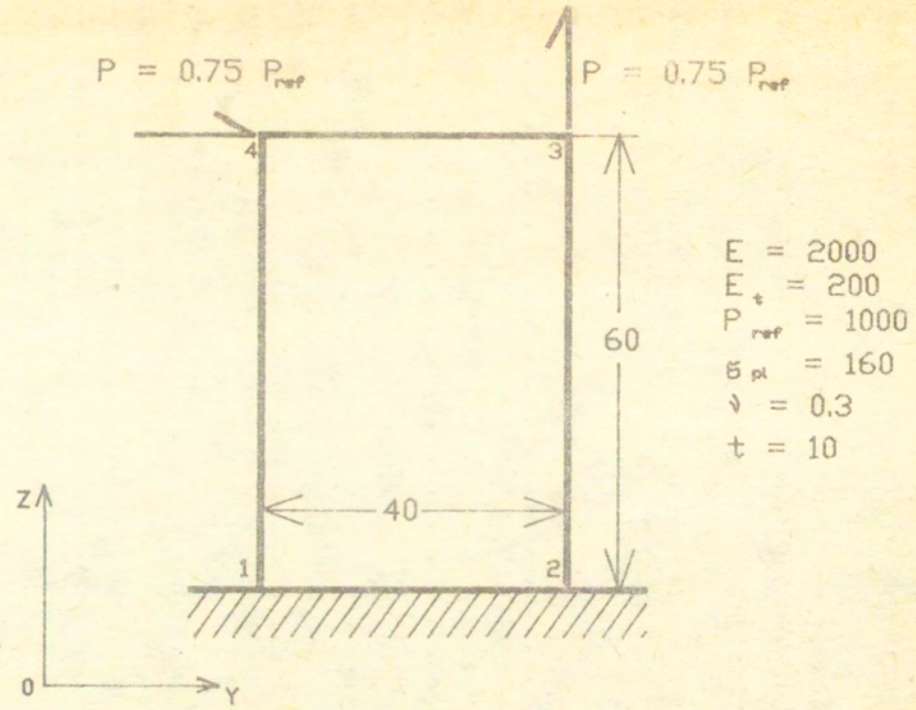
3 stateczność

DLACZEGO?

Podaj numer(y) wartości, DLACZEGO by uzyskać informację o regule,

Regula numer: 5

JEŻELI:



Rys. 5.1 test pojedyńczego elementu PSN

(1) typ analizy: statyka

oraz (2) analiza: NIE liniowa

oraz (3) algorytm: BFGS

TO:

run(writeln [Zbiór] STATYKA BFGS /C)

orazNagłówek 3 - Prawdopodobieństwo=01

W PRZECIWNYM RAZIE:

run(writeln [Zbiór] STATYKA NR /c)

orazNagłówek 3 - Prawdopodobieństwo=01

PONIEWAŻ: ponieważ NANZAM pozwala jedynie na nieliniową

analizę, więc analiza liniowa jest modelowana jako

szczególny przypadek nieliniowej z jednym krokiem

i bez iteracji; jest wówczas nieistotne, czy

wybierzemy algorytm N-R, czy BFGS; pytanie to jest

istotne w przypadkach nieliniowych

Nr linii: wyprowadzenie, <K>-znane dane, <C>-wybór *3*

Jeszcze nie wiem czy to prawda.

Nr linii: wyprowadzenie, <K>-znane dane, <C>-wybór *K*

1 TARCZA PSN/1 - 1 to Nowe zadanie

2 Zmienna [Zbiór] = DATA

3 Zmienna [identyfikator zadania] = TARCZA PSN/1 - 1

Podaj numer wiersza dla dodatkowej informacji,

typ analizy:

1 statyka

2 dynamika

3 stateczność

1

Podaj numer(y) wartości, DLACZEGO by uzyskać informację o regule,

rodzaj analizy:

1 liniowo sprężysta izotropowa

2 liniowo sprężysta ortotropowa

3 Plastyczność Misesa

4 Plastyczność Druckera

5 Nieliniowo sprężysta

1

Podaj numer(y) wartości, DLACZEGO by uzyskać informację o regule,

Warunki to

1 Płaski stan odkształcenia

2 Płaski stan naprężenia

3 Osiowa symetria

1

Podaj numer(y) wartości, DLACZEGO by uzyskać informację o regule,

Podaj liczbę elementów w płaskim stanie odkształcenia

: *1*

Generacja siatki

1 Ręczna

2 Generacja w obszarze prostokątnym

3 Generacja w obszarze kołowym

4 odwzorowanie

2

Podaj numer(y) wartości, DLACZEGO by uzyskać informację o regule,

Podaj liczbę węzłów w kieruneku Y

: *2*

Podaj liczbę węzłów w kierunku Z

: *2*

Podaj odstęp siatki w kierunku osi Y

: *40*

Podaj odstęp siatki w kierunku osi Z

: *60.0*

Podaj listę węzłów podpartych w kierunku Y

: *1 2*

Podaj listę węzłów podpartych w kierunku Z

: *1 2*

Podaj Obciążenie

: *1000,0*

Podaj odpowiednie wartości obciążenia do przyłożenia w

kierunku Y

: *0.75*

Podaj numery węzłów z przyłożonym obciążeniem w kierunku Y

: *3*

Podaj odpowiednie wartości obciążenia do przyłożenia w

kierunku Z

: *0.75*

Podaj numery węzłów z przyłożonym obciążeniem w kierunku Z

: *4*

Podaj Moduł Younga

: *2000*

Podaj Liczba Poissona

: *0.3*

Podaj Grubość

: *10*

NAZWA ZADANIA

* TARCZA PSN/1 - 1 *

WYKONANIE

STATYKA NR

TYPY PSO 1

WEZLY 4

WSPOLRZEDNE WEZLOW

L (1,2) 0.00 (0,60.00)

L (3,4) 40.00 (0,60.00)

WARUNKI BRZEGOWE

ZAMOCOWANY Y 1 2

ZAMOCOWANY Z 1 2

PARAMETRY STERUJACE

LKROK 1 DT 1 LKAKT 2 LKITE 2

FUNKCJE OBCIAZENIA

FUNKCJA 1

0 0 100 1000.000000

OBCIAZENIE WEZLOWE

Y 0.750 3

Z 0.750 4

OPIS ELEMENTOW

TYP PSO GL LSI

WLASNOŚĆI

E V GRUBOŚĆ WSZYSTKIE

2000 0.3 10

UKLAD ELEMENTOW

1 4 2 1 3

KONIEC

JEŻELI dane są poprawne to naciśnij dowolny klawisz...

system pracuje

ładowanie SPN1.exe

system pracuje

ładowanie SPN2.exe

system pracuje

ładowanie SPN3.exe

system pracuje

ładowanie SPN5.exe

system pracuje

*** PROGRAM SPN15(W.WK1PS/R.WK2PS)

*** ROZWIAZANIE PROBLEMU: TARCZA PSN/1 - 1

***KROK NR 1 WARTOSC PARAMETRU STEROWANIA .1000$E+01$

BEZ ITERACJI ROWNOWAGI

UAKTUALNIONO MACIERZ SZTYWNOSCI

***PRZEMIESZCZENIA

NR WEZLA X Y Z

1 .000000 .000000 .000000

2 .000000 .000000 .000000

3 .000000 .133466E-02 .934031E-02

4 .000000 .463043E-02 .146251E-02

*** KONIEC OBLICZEN PROBLEMU TARCZA PSN/1 - 1

1 typ analizy : statyka

2 TARCZA PSN/1 - 1 : Nowe zadanie

3 Przyjęte warunki to : Plaski stan odkształcenia

4 Generacja siatki : Generacja w obszarze prostokątnym

5 rodzaj analizy : liniowo sprężysta izotropowa

6 Zmienna [Zbiór] = DATA

7 Zmienna [identyfikator zadania] = TARCZA PSN/1 - 1

8 Zmienna [NOY] = 2.000000

9 Zmienna [NOZ] = 2.000000

10 Zmienna [DY] = 40.000000

11 Zmienna [DZ] = 60.000000

12 Zmienna [Węzły] = 1 2

13 Zmienna [Węzły Z] = 1 2

14 Zmienna [Obciążenie] = 1000.000000

15 Zmienna [Obciążone Węzły] = 4

16 Zmienna [Wartości] = 0.75

17 Zmienna [Wartości Z] = 0.75

18 Zmienna [Obciążone Węzły Z] = 3.000000

19 Zmienna [E] = 2000

20 Zmienna [V] = 0.3

21 Zmienna [GRUBOŚĆ] = 10

Numer zmienianej linii, <O> Pierwotne dane, <R> Powrót do konsultacji,

H informacja, dowolny inny klawisz - więcej danych: *5*

rodzaj analizy :

1 liniowo sprężysta izotropowa

2 liniowo sprężysta ortotropowa

3 Plastyczność Misesa

4 Plastyczność Druckera

5 Nieliniowo sprężysta

3

Numer(y) wartości, Q - zrzut danych na dysk, <H> informacja

1 typ analizy : statyka

2 [Identyfikator] : Nowe zadanie

3 Przyjęte warunki : Plaski stan odkształcenia

4 Generacja siatki : Generacja w obszarze prostokątnym

5 rodzaj analizy : Plastyczność Misesa

6 Zmienna [Zbiór] = DATA

7 Zmienna [identyfikator zadania] = TARCZA PSN/1 - 1

8 Zmienna [NOY] = 2.000000

9 Zmienna [NOZ] = 2.000000

10 Zmienna [DY] = 40.000000

11 Zmienna [DZ] = 60.000000

12 Zmienna [Węzły] = 1 2

13 Zmienna [Węzły Z] = 1 2

14 Zmienna [Obciążenie] = 1000.000000

15 Zmienna [Obciążone Węzły] = 4

16 Zmienna [Wartości] = 0.75

17 Zmienna [Wartości Z] = 0.75

18 Zmienna [Obciążone Węzły Z] = 3.000000

19 Zmienna [E] = 2000

20 Zmienna [V] = 0.3

21 Zmienna [GRUBOŚĆ] = 10

Numer zmienianej linii, <O> Pierwotne dane, <R> Powrót do konsultacji,

H informacja, dowolny inny klawisz - więcej danych: *R*

algorytm :

1 N - R

2 BFGS

1

Podaj numer(y) wartości, DLACZEGO by uzyskać informację o regule,

Nieliniowość jest:

1 słaba

2 średnia

3 silna

4 Chciałbyś wprowadzić parametry samemu

1

Podaj numer(y) wartości, DLACZEGO by uzyskać informację o regule,

Podaj Granica plastyczności

: *160*

Podaj Moduł styczny

: *200*

NAZWA ZADANIA

* TARCZA PSN/1 - 1 *

WYKONANIE

STATYKA NR

TYPY PSO 1

WEZLY 4

WSPOLRZEDNE WEZLOW

L (1,2) 0.00 (0,60.00)

L (3,4) 40.00 (0,60.00)

WARUNKI BRZEGOWE

ZAMOCOWANY Y 1 2

ZAMOCOWANY Z 1 2

PARAMETRY STERUJACE

LKROK 1 DT 100 ITEMAX 10 LKITE 0

FUNKCJE OBCIAZENIA

FUNKCJA 1

0 0 100 1000.000000

OBCIAZENIE WEZLOWE

Y 0.750 1 4

Z 0.750 1 3

OPIS ELEMENTOW

TYP PSO UL MISES

WLASNOŚĆI

E V SIGMA ET GRUBOŚĆ WSZYSTKIE

2000 0.3 160 200 10

UKLAD ELEMENTOW

1 4 2 1 3

KONIEC

JEŻELI dane są poprawne to naciśnij dowolny klawisz...

system pracuje

ładowanie SPN1.exe

system pracuje

ładowanie SPN2.exe

system pracuje

ładowanie SPN3.exe

system pracuje

ładowanie SPN5.exe

system pracuje

*** PROGRAM SPN15(W.WK1PS/R.WK2PS)

*** ROZWIAZANIE PROBLEMU: TARCZA PSN/1 - 1

***KROK NR 1 WARTOSC PARAMETRU STEROWANIA .1000E+03

WYKONANO 1 ITERACJI ROWNOWAGI

UAKTUALNIONO MACIERZ SZTYWNOSCI

***PRZEMIESZCZENIA

NR WEZLA X Y Z

1 .000000 .000000 .000000

2 .000000 .000000 .000000

3 .000000 .147588 .923789

4 .000000 .467874 .161122

*** KONIEC OBLICZEN PROBLEMU TARCZA PSN/1 - 1

5.1.4 reguły użyte w przykładowym dialogu.

Reguła numer: 1

JEŻELI:

TO:

RUN(REWRITE [Zbiór] NAZWA ZADANIA /C)

*oraz RUN(WRITELN [Zbiór] * [identyfikator zadania] * /C)*

oraz Nagłówek 1 - Prawdopodobieństwo=1

PONIEWAŻ:

ta reguła nie ma części "JEŻELI", więc jest zawsze

wykonywana jako pierwsza i służy do inicjalizacji zbioru

danych

NA PODSTAWIE

por. wydruki programów REWRITE.PAS i WRITELN.PAS

Reguła numer: 2

JEŻELI:

[Identyfikator] IS Nowe zadanie

TO:

RUN(WRITELN [Zbiór] WYKONANIE /C)

oraz Nagłówek 2 - Prawdopodobieństwo=1

W PRZECIWNYM RAZIE:

RUN(WRITELN [Zbiór] RESTART /C)

oraz Nagłówek 2 - Prawdopodobieństwo=1

Reguła numer: 3

JEŻELI:

rodzaj analizy :

Liniowo sprężysta izotropowa

LUB Liniowo sprężysta ortotropowa

TO:

analiza jest: liniowa

Reguła numer: 4

JEŻELI:

rodzaj analizy :

Plastyczność Misesa LUB Plastyczność Druckera

LUB Nieliniowa sprężystość

TO:

analiza jest: nieliniowa

Reguła numer: 5

JEŻELI:

typ analizy is statyka

oraz analiza jest: NOT liniowa

oraz algorytm is BFGS

TO:

RUN(WRITELN [Zbiór] STATYKA BFGS /C)

oraz Nagłówek 3 - Prawdopodobieństwo=1

W PRZECIWNYM RAZIE:

RUN(WRITELN [Zbiór] STATYKA NR /c)

oraz Nagłówek 3 - Prawdopodobieństwo=1

PONIEWAŻ: NANZAM pozwala jedynie na nieliniową analizę, więc analiza liniowa jest modelowana jako szczególny przypadek nieliniowej z jednym krokiem i bez iteracji; jest wówczas nieistotne, czy wybierzemy algorytm N-R, czy BFGS; pytanie to jest istotne w przypadkach nieliniowych

Reguła numer: 6

JEŻELI:

Przyjęte warunki to : Płaski stan naprężenia

TO:

[Stany płaskie] Otrzymuje wartość : "PSN"

oraz RUN(WRITELN [Zbiór] TYPY [Stany płaskie] [Elementy PSN] /C) -

oraz Warunki - Prawdopodobieństwo=1

Reguła numer: 7

JEŻELI:

Przyjęte warunki to : Płaski stan odkształcenia

TO:

[Stany płaskie] Otrzymuje wartość : "PSO"

oraz RUN(WRITELN [Zbiór] TYPY [Stany płaskie] [Elementy PSO] /C)

oraz Warunki - Prawdopodobieństwo=1

Reguła numer: 8

JEŻELI:

Generacja siatki Rzezna

TO:

RUN(WS [Zbiór] /e)

oraz węzły - Prawdopodobieństwo=1

JEŻELI:

Generacja siatki Rzezna

TO:

RUN(WS [Zbiór] /c)

oraz węzły - Prawdopodobieństwo=1

Reguła numer: 9

JEŻELI:

Generacja siatki Generacja w obszarze prostokątnym

TO:

RUN(PROSTOKĄT [Zbiór] Węzły [NOY] [NOZ] 4 [DY] [DZ] /C)

oraz węzły - Prawdopodobieństwo=1

Reguła numer: 10

JEŻELI:

węzły = 1

TO:

RUN(WRITELN [Zbiór] WARUNKI BRZEGOWE /C)

oraz RUN(WRITELN [Zbiór] ZAMOCOWANY Y [Węzły] /C)

oraz RUN(WRITELN [Zbiór] ZAMOCOWANY Z [Węzły Z] /C)

oraz Podparcie - Prawdopodobieństwo=1

Reguła numer: 11

JEŻELI

typ analizy is statyka

oraz analiza jest: liniowa

TO:

RUN(WRITELN [Zbiór] PARAMETRY STERUJACE /C)

oraz RUN(WRITELN [Zbiór] LKROK 1 DT 1 LKAKT 2 LKITE 2 /C)

oraz RUN(WRITELN [Zbiór] FUNKCJE OBCIAZENIA /C)

oraz RUN(WRITELN [Zbiór] FUNKCJA 1 /C)

oraz RUN(WRITELN [Zbiór] 0 0 100 [Obciążenie] /C)

oraz Parametry - Prawdopodobieństwo=1

Regula numer: 12

JEŻELI:

typ analizy is statyka

oraz analiza jest: nieliniowa

oraz Nieliniowość jest: słaba

TO:

RUN(WRITELN [Zbiór] PARAMETRY STERUJACE /C)

oraz RUN(WRITELN [Zbiór] LKROK 1 DT 100 ITEMAX 10 LKITE 0 /C)

oraz RUN(WRITELN [Zbiór] FUNKCJE OBCIAZENIA /C)

oraz RUN(WRITELN [Zbiór] FUNKCJA 1 /C)

oraz RUN(WRITELN [Zbiór] 0 0 100 [Obciążenie] /C)

oraz Parametry - Prawdopodobieństwo=1

Reguła numer: 13

JEŻELI:

Parametry = 1

TO:

RUN(WRITELN [Zbiór] OBCIAZENIE WEZLOWE /C)

oraz RUN(LOADINPT [Zbiór] Y [Wartości]

Wartości [Obciążone węzły y] /C)

oraz RUN(LOADINPT [Zbiór] Z [Wartości z]

Wartości [Obciążone węzły z] /c)

oraz obciążenia - Prawdopodobieństwo=1

Reguła numer: 14

JEŻELI:

obciążenia = 1

TO:

RUN(WRITELN [Zbiór] OPIS ELEMENTOW /C)

oraz Nagłówek opisu elementów - Prawdopodobieństwo=1

Reguła numer: 15

JEŻELI:

Przyjęte warunki :

Płaski stan odkształcenia LUB Płaski stan naprężenia

oraz Rodzaj analizy :

Liniowo sprężysta izotropowa

TO:

RUN(WRITELN [Zbiór] TYP [Stany płaskie] GL LSI /C)

oraz RUN(WRITELN [Zbiór] WLASNOŚĆI /C)

oraz RUN(WRITELN [Zbiór] E V GRUBOŚĆ WSZYSTKIE /C)

oraz RUN(WRITELN [Zbiór] [E] [V] [GRUBOŚĆ] /C)

oraz Opis elementów - Prawdopodobieństwo=1

Reguła numer: 16

JEŻELI:

Przyjęte warunki :

Płaski stan odkształcenia LUB Płaski stan naprężenia

oraz Rodzaj analizy :

liniowo sprężysta ortotropowa

TO:

run(writeln [Zbiór] TYP [Stany płaskie] GL LSO /C)

oraz RUN(WRITELN [Zbiór] WLASNOŚĆI /C)

oraz RUN(WRITELN [Zbiór] E1 E2 E3 V12 V23 G12 BETA GRUBOŚĆ WSZYSTKIE

/C)

oraz RUN(WRITELN [Zbiór] [E1] [E2] [E3] [V12] [V23] [G12] [BETA]

[GRUBOŚĆ] /C)

oraz Opis elementów - Prawdopodobieństwo=1

Reguła numer: 17

JEŻELI:

Przyjęte warunki :

Płaski stan odkształcenia LUB Płaski stan naprężenia

oraz Rodzaj analizy :

Plastyczność Misesa

TO:

run(WRITELN [Zbiór] TYP [Stany płaskie] UL MISES /C)

oraz RUN(WRITELN [Zbiór] WLASNOŚĆI /C)

oraz RUN(WRITELN [Zbiór] E V SIGMA ET GRUBOŚĆ WSZYSTKIE /C)

oraz RUN(WRITELN [Zbiór] [E] [V] [SIGMA] [ET] [GRUBOŚĆ] /C)

oraz Opis elementów - Prawdopodobieństwo=1

Reguła numer: 18

JEŻELI:

Opis elementów = 1

TO:

Opis elementów - Prawdopodobieństwo=1

Reguła numer: 18

JEŻELI:

Opis elementów = 1

TO:

run(PROSTOKĄT [Zbiór] ELEMENTY [NOY] [NOZ] [DY] [DZ] /C)

oraz Połączenia - Prawdopodobieństwo=1

Reguła numer: 19

JEŻELI:

Nagłówek 1 = 1

oraz Nagłówek 2 = 1

oraz Nagłówek 3 = 1

oraz Warunki = 1

JEŻELI:

Nagłówek 1 = 1

oraz Nagłówek 2 = 1

oraz Nagłówek 3 = 1

oraz Warunki = 1

oraz węzły = 1

oraz Podparcie = 1

oraz Parametry = 1

oraz obciążenia = 1

oraz Nagłówek opisu elementów = 1

oraz Podparcie = 1

oraz Parametry = 1

oraz obciążenia = 1

oraz Nagłówek opisu elementów = 1

oraz Opis elementów = 1

oraz Połączenia = 1

TO:

RUN(WRITELN [Zbiór] KONIEC /C)

oraz RUN(DISPLAY [Zbiór] /c)

oraz DISPLAY - Prawdopodobieństwo=1

oraz RUN(\NANZAM\ SPN1.EXE)

oraz RUN(\NANZAM\ SPN2.EXE)

oraz RUN(\NANZAM\ SPN8.EXE)

oraz RUN(\NANZAM\ SPN5.EXE > CON /C)

oraz RUN(DISPLAY RFN5 /C)

5.2 Moduł akwizycji wiedzy INDUKTOR.

Praktyczne zastosowanie algorytmu indukcyjnego zademonstrujemy na dwóch przykładach: zagadnienia generacji praw dotyczących modelowania konstrukcji wspornikowych oraz generacji reguł dotyczących efektywności alternatywnych algorytmów numerycznych. Obliczenia zostały wykonane na mikrokomputerach typu IBM PC/XT przy pomocy programu INDUKTOR opracowanego w języku PASCAL (por. Dodatek 1).

5.2.1 indukcja reguł modelowania konstrukcji wspornikowych

Płaski wspornik modelowany przy pomocy różnych rodzajów i układów elementów poddawany jest obciążeniu zginającemu (rys 5.3). Zakładamy, że spełnione są warunki płaskiego stanu naprężenia. Atrybuty dobrano następująco:

L wydłużenie wspornika, (wartości całkowite, rys.5.2),

MESH rodzaj użytej siatki elementów, (wartości symboliczne, rys.5.4),

DOF liczba stopni swobody, (wartości całkowite)

EL rodzaj elementu, (wartości symboliczne:

A - prostokąt z liniową funkcją przemieszczeń,

B - prostokąt o stałym ścinaniu, z liniową funkcją przemieszczeń,

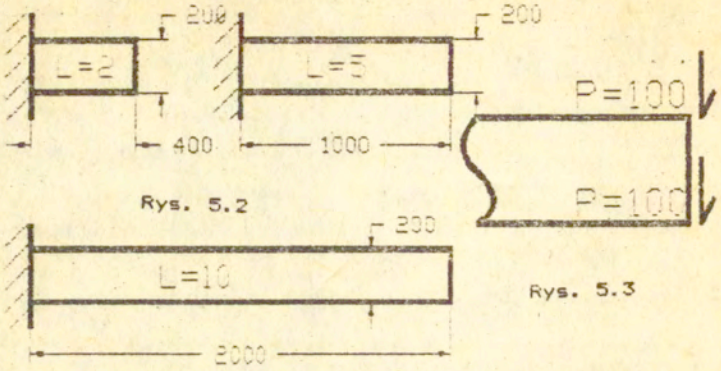
C - czworokąt izoparametryczny,

D - czworokąt izoparametryczny o stałym ścinaniu,

E - trójkąt o liniowych funkcjach kształtu,

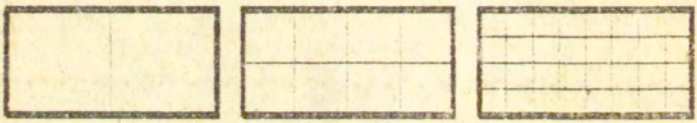
F - trójkąt mający stopnie swobody typu obrotowego,

RESULT stosunek obliczonego ugięcia swobodnego końca



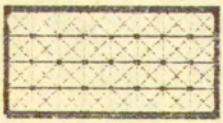
Rys. 5.2

Rys. 5.3



MESH I MESH II MESH III

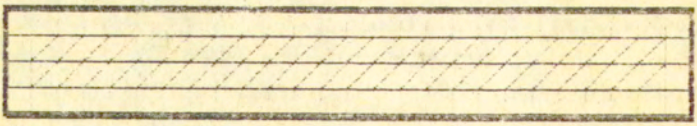
MESH IV MESH V MESH VI



Rys. 5.4

MESH VII

MESH VIII



wspornika do znanego rozwiązania analitycznego w przyjętego

do celów porównawczych w problemach wzorcowych [55] następująco:

$$w = \frac{Pl^3}{3EJ} + \frac{1.5Pl}{AG}$$

gdzie: $P = 200, E = 2.1E6, G = 8.1E5$.

Wyniki obliczeń mogą być przedstawione w postaci poniższej listy przykładów:

```
((EL L MESH DOF RESULT)
(A 2 I 8 0.64)
(A 2 II 24 0.81)
(A 2 III 80 0.90)
(B 2 I 8 0.82)
(B 2 II 24 0.88)
(B 2 III 80 0.92)
(C 2 I 8 0.64)
(C 2 II 24 0.81)
(C 2 III 80 0.90)
(D 2 I 8 0.82)
(D 2 II 24 0.88)
(D 2 III 80 0.92)
(E 2 IV 8 0.32)
(E 2 V 12 0.57)
(E 2 VI 24 0.61)
(E 2 VII 80 0.82)
(F 2 I 8 0.64)
(F 2 II 24 0.81)
(F 2 III 80 0.90)
(A 3 I 20 0.67)
(A 3 II 60 0.87)
(A 3 III 200 0.95)
(B 3 I 20 0.89)
(D 3 I 20 0.89)
(D 3 II 60 0.95)
(D 3 III 200 0.98)
(D 3 VIII 200 0.91)
(E 3 IV 20 0.27)
(E 3 V 30 0.58)
(E 3 VI 60 0.62)
(E 3 VII 200 0.86)
(F 3 I 20 0.67)
(F 3 II 60 0.87)
(F 3 III 200 0.95)
(A 10 I 40 0.67)
(A 10 II 120 0.89)
(A 10 III 400 0.95)
(B 10 I 40 0.90)
(B 10 II 120 0.97)
(B 10 III 400 0.99)
(C 10 I 40 0.67)
(C 10 II 120 0.89)
(C 10 III 400 0.96)
(D 10 I 40 0.90)
(D 10 II 120 0.97)
(D 10 III 400 0.99)
```

(B 3 II 60 0.95)
(B 3 III 200 0.98)
(C 3 I 40 0.67)
(C 3 II 60 0.87)
(C 3 III 200 0.95)
(C 3 VIII 200 0.87)
(E 10 IV 40 0.29)
(E 10 V 60 0.58)
(E 10 VI 120 0.62)
(F 10 I 40 0.67)
(F 10 II 120 0.88)
(F 10 III 400 0.92)),

gdzie pierwsza lista:

(EL L MESH DOF RESULT)

zawiera nazwy atrybutów, a kolejne listy opisujące przykłady składają się z wartości atrybutów, n.p.:

(A 2 I 8 0.64)

oznacza:

elementy typu A

wydłużenie L/H = 2

rodzaj siatki I

liczba stopni swobody 8

wynik wl/w = 0.64.

Atrybutowi *RESULT* o wartościach rzeczywistych przyporządkowano na podstawie subiektywnej oceny zbiór wartości dyskretnych:

gorzej niż 50% : *BAD*,

pomiędzy 50% a 75% : *POOR*,

pomiędzy 75% a 95% : *GOOD*,

lepiej niż 95% : *EXCELLENT*.

W wyniku działania programu indukcyjnego otrzymano następujące prawa:

RULE 1:

IF MESH = I THEN IF EL = A OR C OR F
THEN POOR
ELSE IF EL = B OR D
THEN GOOD.

RULE 2:

IF MESH = II THEN IF EL = A OR C OR F
THEN GOOD
ELSE IF EL = B OR D
THEN IF L = 2 OR 3
THEN GOOD
ELSE IF L = 10
THEN EXCELLENT.

RULE 3:

IF MESH = III THEN IF EL = A OR F
THEN GOOD
ELSE IF EL = B OR C OR D
THEN IF L = 2
THEN GOOD
ELSE IF L = 3 OR 10
THEN EXCELLENT.

RULE 4:

IF MESH = IV THEN BAD.

RULE 5:

IF MESH = V OR VI THEN POOR.

RULE 6:

IF MESH = VII OR VIII THEN GOOD.

Interpretacja powyższych praw nie nastręcza trudności i można ją sformułować następująco:

"Dla konstrukcji typu wspornikowego dokładność obliczeń MES jest uzależniona w pierwszym rzędzie od rodzaju przyjętej siatki elementów, podczas gdy dla pewnych typów elementów wyniki poprawiają się ze wzrostem wydłużenia".

Co jest szczególnie istotne, liczba stopni swobody oraz użycie odkształconych elementów (przynajmniej w stopniu dopuszczonym w siatce typu VIII) mają drugorzędne znaczenie podczas gdy pewne siatki zostały zdyskwalifikowane dla wszystkich rodzajów elementów.

Powyższe wnioski mogą wydawać się doświadczonemu użytkownikowi oczywiste, tym niemniej sposób ich automatycznego uzyskiwania wydaje się być atrakcyjny, wskazuje bowiem na potencjał opracowanego algorytmu.

5.2.2 Indukcja reguł określających efektywność algorytmów

Szczególnie interesującym obszarem zastosowania algorytmów indukcyjnych jest automatyczne generowanie regul określających efektywność alternatywnych algorytmów numerycznych. Zbiór przykładów zastosowań różnych algorytmów rozwiązywania układów liniowych równań algebraicznych otrzymano [102] dla zadania

$$u_{xx} + 2u_{yy} = 4$$

rozwiązywanego metodą Galerkina przy jednorodnych warunkach brzegowych Dirichleta i Neumanna. Wzięto pod uwagę dwa rodzaje obszarów: w kształcie litery L oraz kwadrat, różnej gęstości niejednorodne siatki elementów trójkątnych i czworokątnych przy zastos aniu liniowych i kwadratowych funkcji kształtu klasy C^0 i C^1. Zadanie rozwiązano numerycznie przy wykorzystaniu różnych sposobów składowania macierzy współczynników dla algorytmów bezpośrednich oraz różnych algorytmów iteracyjnych.

Dane dla algorytmu indukcyjnego zapisano dobierając 6 następujących atrybutów o wartościach symbolicznych:

DOF- liczba stopni swobody;

poniżej 200, $DOF = VLOW$;

pomiędzy 200 a 500, $DOF = LOW$;

pomiędzy 500 a 950, $DOF = MID$;

pomiędzy 950 a 1450, $DOF = HIGH$;

powyżej 1450, $DOF = VHIGH$,

EL- rodzaj elementów:

elementy trójkątne, $EL = T$;

elementy czworokątne, $EL = Q$,

ORD - rząd funkcji kształtu:

funkcje liniowe, $ORD = L$;

funkcje kwadratowe, $ORD = Q$,

$CLASS$ - klasa ciągłości funkcji kształtu:

funkcje klasy $C0$, $CLASS = C0$;

funkcje klasy $C1$, $CLASS = C1$;

$METH$- rodzaj metody rozwiązywania układu równań:

metoda bezpośrednia, $METH = DIR$;

metoda iteracyjna, $METH = ITER$,

ALG- rodzaj algorytmu:

metoda nadrelaksacji, $ALG = SOR$;

metoda symetrycznej nadrelaksacji połączona z metodą gradientów sprzężonych, $ALG = SSOR - CG$;

metoda Jacobiego połączona z metodą gradientów sprzężonych, $ALG = J - CG$;

metod macierzy rzadkich, $ALG = SPARSE$;

metoda pasmowa z numeracją stopni swobody według odwrotnego algorytmu Cuthil-McKnee [101], $ALG = BAND - RCM$;

metoda pasmowa z naturalną numeracją stopni swobody, $ALG = BAND - N$;

metoda "skyline", $ALG = ENV$

Relatywną efektywność metod scharakteryzowano atrybutem *PERF*. Wartości atrybutu *PERF* otrzymano szeregując wyniki otrzymane dla każdego przykładu według czasu wykonania. Najlepszemu algorytmowi przypisano wartość $PERF = 1$, następnym algorytmom przypisano wartości *PERF* od jednego do siedmiu. Jeżeli algorytm spowodował przekroczenie dostępnej pamięci operacyjnej, to atrybut *PERF* otrzymuje wartość *OVF*.

Listę kolejnych 128 przykładów zapisano zgodnie z powyższym określeniem atrybutów następująco:

((DOF EL ORD CLASS METH ALG PERF)
(VLOW T L C0 DIR ENV 1)
(LOW T L C0 DIR ENV 1)
(MID T L C0 DIR ENV 1)
(HIGH T L C0 DIR ENV 1)
(VHIGH T L C0 ITER J-CG 1)
(VLOW T Q C0 DIR ENV 1)
(LOW T Q C0 DIR ENV 1)
(MID T Q C0 DIR ENV 1)
(HIGH T Q C0 DIR ENV 1)
(VHIGH T Q C0 ITER J-CG 1)
(VLOW Q Q C0 DIR ENV 1)
(LOWQ Q Q C0 DIR BAND-N 4)
(MID Q Q C0 DIR BAND-N 4)
(HIGH Q Q C0 DIR SPARSE 4)
(VLOW Q Q C1 ITER SOR 4)
(LOW Q Q C1 ITER ENV 4)
(MID Q Q C1 ITER SSOR-CG 4)
(VLOW T L C0 ITER SOR 5)
(LOW T L C0 DIR SPARSE 5)
(MID T L C0 DIR SPARSE 5)
(HIGH T L C0 DIR SPARSE 5)
(VLOW T Q C0 ITER SOR 5)

(LOW Q Q C0 ITER J-CG 1)
(MID Q Q C0 ITER J-CG 1)
(HIGH Q Q C0 ITER J-CG 1)
(VLOW Q Q C1 ITER J-CG 1)
(LOW Q Q C1 ITER J-CG 1)
(MID Q Q C1 ITER J-CG 1)
(HIGH Q Q C1 ITER J-CG 1)
(VLOW T L C0 DIR BAND-N 2)
(LOW T L C0 DIR BAND-RCM 2)
(MID T L C0 ITER J-CG 2)
(HIGH T L C0 ITER J-CG 2)
(VHIGH T L C0 ITER SSOR-CG 2)
(VLOW T Q C0 DIR BAND-N 2)
(LOW T Q C0 ITER J-CG 2)
(MID T Q C0 ITER J-CG 2)
(HIGH T Q C0 ITER J-CG 2)
(VHIGH T Q C0 ITER SSOR-CG 2)
(VLOW Q Q C0 ITER J-CG 2)
(LOW Q Q C0 DIR ENV 2)
(MID Q Q C0 DIR ENV 2)
(HIGH Q Q C0 ITER SSOR-CG 2)
(VLOW Q Q C1 DIR BAND-N 2)
(LOW Q Q C1 ITER SOR 2)
(MID Q Q C1 ITER SOR 2)
(HIGH Q Q C1 ITER SOR 2)
(VLOW T L C0 ITER J-CG 3)
(LOW T L C0 ITER J-CG 3)
(MID T L C0 DIR BAND-RCM 3)
(HIGH T L C0 DIR BAND-RCM 3)
(VHIGH T L C0 ITER SOR 3)
(VLOW T Q C0 ITER J-CG 3)
(LOW T Q C0 DIR BAND-RCM 3)
(MID T Q C0 DIR SPARSE 3)
(HIGH T Q C0 DIR SPARSE 3)
(LOW T Q C0 ITER SOR 5)
(MID T Q C0 DIR BAND-N 5)
(HIGH T Q C0 ITER SOR 5)
(VLOW Q Q C0 DIR BAND-RCM 5)
(LOW Q Q C0 ITER SOR 5)
(MID Q Q C0 DIR SPARSE 5)
(VLOW Q Q C1 ITER SSOR-CG 5)
(LOW Q Q C1 ITER SSOR-CG 5)
(MID Q Q C1 ITER SPARSE 5)
(VLOW T L C0 DIR SPARSE 6)
(LOW T L C0 ITER SOR 6)
(MID T L C0 ITER SSOR-CG 6)
(HIGH T L C0 ITER SSOR-CG 6)
(VLOW T Q C0 DIR SPARSE 6)
(LOW T Q C0 DIR BAND-N 6)
(MID T Q C0 ITER SOR 6)
(VLOW Q Q C0 DIR SPARSE 6)
(LOW Q Q C0 DIR SPARSE 6)
(MID Q Q C0 DIR SPARSE 6)
(VLOW Q Q C1 DIR BAND-RCM 6)
(LOW Q Q C1 DIR BAND-RCM 6)
(VLOW T L C0 ITER SSOR-CG 7)
(LOW T L C0 ITER SSOR-CG 7)
(MID T L C0 ITER SOR 7)
(HIGH T L C0 ITER SOR 7)
(VLOW T Q C0 ITER SSOR-CG 7)
(LOW T Q C0 ITER SSOR-CG 7)
(MID T Q C0 ITER SSOR-CG 7)
(VLOW Q Q C0 DIR BAND-N 7)
(LOW Q Q C0 DIR BAND-RCM 7)
(VLOW Q Q C1 DIR SPARSE 7)
(LOW Q Q C1 DIR SPARSE 7)
(VHIGH T L C0 DIR BAND-N OVF)
(VHIGH T L C0 DIR BAND-RCM OVF)

(VHIGH T Q C0 ITER SOR 3)
(VLOW Q Q C0 ITER SSOR-CG 3)
(LOW Q Q C0 ITER SSOR-CG 3)
(MID Q Q C0 ITER SSOR-CG 3)
(HIGH Q Q C0 ITER SOR 3)
(VLOW Q Q C1 DIR ENV 3)
(LOW Q Q C1 DIR BAND-N 3)
(MID Q Q C1 DIR BAND-N 3)
(HIGH Q Q C1 ITER SSOR-CG 3)
(VLOW T L C0 ITER J-CG 4)
(LOW T L C0 DIR BAND-N 4)
(MID T L C0 DIR BAND-N 4)
(HIGH T L C0 DIR BAND-N 4)
(VLOW T Q C0 DIR BAND-RCM 4)
(LOW T Q C0 DIR SPARSE 4)
(MID T Q C0 DIR BAND-RCM 4)
(HIGH T Q C0 ITER SSOR-CG 4)
(VLOW Q Q C0 ITER SOR 4)
(HIGH Q Q C1 DIR SPARSE OVF))
(VHIGH T L C0 DIR ENV OVF)
(VHIGH T L C0 DIR SPARSE OVF)
(HIGH T Q C0 DIR BAND-N OVF)
(HIGH T Q C0 DIR BAND-RCM OVF)
(VHIGH T Q C0 DIR BAND-N OVF)
(VHIGH T Q C0 DIR BAND-RCM OVF)
(VHIGH T Q C0 DIR ENV OVF)
(VHIGH T Q C0 DIR SPARSE OVF)
(MID Q Q C0 DIR BAND-RCM OVF)
(HIGH Q Q C0 DIR BAND-N OVF)
(HIGH Q Q C0 DIR BAND-RCM OVF)
(HIGH Q Q C0 DIR ENV OVF)
(HIGH Q Q C0 DIR SPARSE OVF)
(MID Q Q C1 DIR BAND-RCM OVF)
(MID Q Q C1 DIR ENV OVF)
(HIGH Q Q C1 DIR BAND-N OVF)
(HIGH Q Q C1 DIR BAND-RCM OVF)
(HIGH Q Q C1 DIR ENV OVF)

W wyniku działania programu indukcyjnego otrzymano siedem reguł:

```
RULE 1:
    IF ALG=ENV THEN
        IF DOF=VLOW THEN
            IF CLASS=C0 THEN 1 ELSE
            IF CLASS=C1 THEN 3 ELSE
        IF DOF=LOW THEN
            IF EL=T THEN 1 ELSE
            IF EL=Q THEN
                IF CLASS=C0 THEN 2 ELSE
                IF CLASS=C1 THEN 4 ELSE
        IF DOF=MID THEN
            IF EL=T THEN 1 ELSE
            IF EL=Q THEN
                IF CLASS=C0 THEN 2 ELSE
                IF CLASS=C1 THEN OVF ELSE
        IF DOF=HIGH THEN
            IF EL=T THEN 1 ELSE
            IF EL=Q THEN OVF ELSE
        IF DOF=VHIGH THEN OVF
```

```
RULE 2:
    IF ALG=J-CG THEN
        IF DOF=VLOW THEN
            IF EL=T THEN 3 ELSE
            IF EL=Q THEN
                IF CLASS=C0 THEN 2 ELSE
                IF CLASS=C1 THEN 1 ELSE
        IF DOF=LOW THEN
            IF EL=T THEN
                IF ORD=L THEN 3 ELSE
                IF ORD=Q THEN 2 ELSE
            IF EL=Q THEN 1 ELSE
        IF DOF=MID THEN
            IF EL=T THEN 2 ELSE
            IF EL=Q THEN 1 ELSE
        IF DOF=HIGH THEN
            IF EL=T THEN 2 ELSE
            IF EL=Q THEN 1 ELSE
        IF DOF=VHIGH THEN 1
```

```
RULE 8:
    IF ALG=BAND-N THEN
        IF DOF=VLOW THEN
            IF EL=T THEN 2 ELSE
            IF EL=Q THEN
                IF CLASS=C0 THEN 7 ELSE
                IF CLASS=C1 THEN 2 ELSE
        IF DOF=LOW THEN
            IF CLASS=C0 THEN
                IF EL=T THEN
                    IF ORD=L THEN 4 ELSE
                    IF ORD=Q THEN 6 ELSE
                IF EL=Q THEN 4 ELSE
            IF CLASS=C1 THEN 3 ELSE
        IF DOF=MID THEN
            IF CLASS=C0 THEN
                IF EL=T THEN
                    IF ORD=L THEN 4 ELSE
                    IF ORD=Q THEN 5 ELSE
                IF EL=Q THEN 4 ELSE
            IF CLASS=C1 THEN 3 ELSE
        IF DOF=HIGH THEN
            IF ORD=L THEN 4 ELSE
            IF ORD=Q THEN OVF ELSE
        IF DOF=VHIGH THEN OVF
```

```
RULE 4:
    IF ALG=BAND-ROM THEN
        IF DOF=VLOW THEN
            IF EL=T THEN 4 ELSE
            IF EL=Q THEN
                IF CLASS=C0 THEN 5 ELSE
                IF CLASS=C1 THEN 6 ELSE
        IF DOF=LOW THEN
            IF EL=T THEN
                IF ORD=L THEN 2 ELSE
                IF ORD=Q THEN 3 ELSE
            IF EL=Q THEN
                IF CLASS=C0 THEN 7 ELSE
                IF CLASS=C1 THEN 6 ELSE
        IF DOF=MID THEN
            IF EL=T THEN
                IF ORD=L THEN 3 ELSE
                IF ORD=Q THEN 4 ELSE
            IF EL=Q THEN OVF ELSE
        IF DOF=HIGH THEN
            IF ORD=L THEN 3 ELSE
            IF ORD=Q THEN OVF ELSE
        IF DOF=VHIGH THEN OVF
```

```
RULE 5:
    IF ALG=SSOR-CG THEN
        IF DOF=VLOW THEN
            IF EL=T THEN 7 ELSE
            IF EL=Q THEN
                IF CLASS=C0 THEN 3 ELSE
                IF CLASS=C1 THEN 5 ELSE
        IF DOF=LOW THEN
            IF EL=T THEN 7 ELSE
            IF EL=Q THEN
                IF CLASS=C0 THEN 3 ELSE
                IF CLASS=C1 THEN 5 ELSE
        IF DOF=MID THEN
            IF EL=T THEN
                IF ORD=L THEN 6 ELSE
                IF ORD=Q THEN 7 ELSE
            IF EL=Q THEN
                IF CLASS=C0 THEN 3 ELSE
                IF CLASS=C1 THEN 4 ELSE
        IF DOF=HIGH THEN
            IF EL=T THEN
                IF ORD=L THEN 6 ELSE
                IF ORD=Q THEN 4 ELSE
            IF EL=Q THEN
                IF CLASS=C0 THEN 2 ELSE
                IF CLASS=C1 THEN 3 ELSE
        IF DOF=VHIGH THEN 2
```

```
RULE 6:
    IF ALG=SOR THEN
        IF DOF=VLOW THEN
            IF EL=T THEN 5 ELSE
                IF EL=Q THEN 4 ELSE
        IF DOF=LOW THEN
            IF ORD=L THEN 6 ELSE
            IF ORD=Q THEN
                IF CLASS=C0 THEN 5 ELSE
                IF CLASS=C1 THEN 2 ELSE
        IF DOF=MID THEN
            IF EL=T THEN
                IF ORD=L THEN 7 ELSE
                IF ORD=Q THEN 6 ELSE
            IF EL=Q THEN 2 ELSE
        IF DOF=HIGH THEN
            IF EL=T THEN
                IF ORD=L THEN 7 ELSE
                IF ORD=Q THEN 5 ELSE
            IF EL=Q THEN
                IF CLASS=C0 THEN 3 ELSE
                IF CLASS=C1 THEN 2 ELSE
        IF DOF=VHIGH THEN 3
```

```
RULE 7:
    IF ALG=SPARSE THEN
        IF DOF=VLOW THEN
            IF CLASS=C0 THEN 6 ELSE
            IF CLASS=C1 THEN 7 ELSE
        IF DOF=LOW THEN
            IF EL=T THEN
                IF ORD=L THEN 5 ELSE
                IF ORD=Q THEN 4 ELSE
            IF EL=Q THEN
                IF CLASS=C0 THEN 6 ELSE
                IF CLASS=C1 THEN 7 ELSE
        IF DOF=MID THEN
            IF EL=T THEN
                IF ORD=L THEN 5 ELSE
                IF ORD=Q THEN 3 ELSE
            IF EL=Q THEN 5 ELSE
        IF DOF=HIGH THEN
            IF EL=T THEN
                IF ORD=L THEN 5 ELSE
                IF ORD=Q THEN 3 ELSE
            IF EL=Q THEN OVF ELSE
        IF DOF=VHIGH THEN OVF.
```

Wprawdzie otrzymane reguły można wykorzystać w bazie wiedzy systemu ekspertowego, to jednak ich jakościowa interpretacja bez użycia komputera może nastręczać pewne trudności z uwagi na dość duży stopień złożoności warunków. Zauważmy więc na razie tylko, że precyzyjna miara efektywności poszczególnych algorytmów w ostatecznym rozrachunku zależy w powyższych prawach od atrybutów ORD, CLASS, EL. Jak wiadomo, rodzaj elementu skończonego, rząd i klasa funkcji kształtu mają decydujący wpływ na stopień wypełnienia macierzy sztywności elementami niezerowymi. Atrybuty: ORD, CLASS i EL zastąpimy więc, jedynie dla celów interpretacyjnych nowym atrybutem MATRIX określonym następująco:

elementy trókątne, liniowe funkcje kształtu klasy $C0, MATRIX = SPARSE$;

elementy trójkątne, funkcje kształtu rzędu drugiego

klasy $C0, MATRIX = DENSE$;

elementy czworokątne, funkcje kształtu rzędu drugiego klasy $C0, MATRIX = VDENSE$;

elementy czworokątne, funkcje kształtu rzędu drugiego klasy $C1, MATRIX = SATURATED$.

Zredukujmy też zakres wartości atrybutu $PERF$. Niech dotychczasowym wartościom $PERF$:

1, 2 odpowiada $PERF1 = 1$

3, 4, 5 odpowiada $PERF1 = 2$

6,7 odpowiada $PERF1 = 3$

OVF bez zmiany, $PERF1 = OVF$.

Zmodyfikowana lista przykładów przedstawia się następująco:

((DOF MATRIX METH ALG PERF1)
(VLOW SPARSE DIR ENV 1)
(LOW SPARSE DIR ENV 1)
(MID SPARSE DIR ENV 1)
(HIGH SPARSE DIR ENV 1)
(VHIGH SPARSE ITER J-CG 1)
(VLOW DENSE DIR ENV 1)
(LOW DENSE DIR ENV 1)
(MID DENSE DIR ENV 1)
(HIGH DENSE DIR ENV 1)
(VHIGH DENSE ITER J-CG 1)
(LOW VDENSE DIR BAND-N 2)
(MID VDENSE DIR BAND-N 2)
(HIGH VDENSE DIR SPARSE 2)
(VLOW SATURATED ITER SOR 2)
(LOW SATURATED ITER ENV 2)
(MID SATURATED ITER SSOR-CG 2)
(VLOW SPARSE ITER SOR 2)
(LOW SPARSE DIR SPARSE 2)
(MID SPARSE DIR SPARSE 2)
(HIGH SPARSE DIR SPARSE 2)

(VLOW VDENSE DIR ENV 1)
(LOW VDENSE ITER J-CG 1)
(MID VDENSE ITER J-CG 1)
(HIGH VDENSE ITER J-CG 1)
(VLOW SATURATED ITER J-CG 1)
(LOW SATURATED ITER J-CG 1)
(MID SATURATED ITER J-CG 1)
(HIGH SATURATED ITER J-CG 1)
(VLOW SPARSE DIR BAND-N 1)
(LOW SPARSE DIR BAND-RCM 1)
(MID SPARSE ITER J-CG 1)
(HIGH SPARSE ITER J-CG 1)
(VHIGH SPARSE ITER SSOR-CG 1)
(VLOW DENSE DIR BAND-N 1)
(LOW DENSE ITER J-CG 1)
(MID DENSE ITER J-CG 1)
(HIGH DENSE ITER J-CG 1)
(VHIGH DENSE ITER SSOR-CG 1)
(VLOW VDENSE ITER J-CG 1)
(LOW VDENSE DIR ENV 1)
(MID VDENSE DIR ENV 1)
(HIGH VDENSE ITER SSOR-CG 1)
(VLOW SATURATED DIR BAND-N 1)
(LOW SATURATED ITER SOR 1)
(MID SATURATED ITER SOR 1)
(HIGH SATURATED ITER SOR 1)
(VLOW SPARSE ITER J-CG 2)
(LOW SPARSE ITER J-CG 2)
(MID SPARSE DIR BAND-RCM 2)
(HIGH SPARSE DIR BAND-RCM 2)
(VHIGH SPARSE ITER SOR 2)
(VLOW DENSE ITER J-CG 2)
(LOW DENSE DIR BAND-RCM 2)
(MID DENSE DIR SPARSE 2)
(VLOW DENSE ITER SOR 2)
(LOW DENSE ITER SOR 2)
(MID DENSE DIR BAND-N 2)
(HIGH DENSE ITER SOR 2)
(VLOW VDENSE DIR BAND-RCM 2)
(LOW VDENSE ITER SOR 2)
(MID VDENSE DIR SPARSE 2)
(VLOW SATURATED ITER SSOR-CG 2)
(LOW SATURATED ITER SSOR-CG 2)
(MID SATURATED ITER SPARSE 2)
(VLOW SPARSE DIR SPARSE 3)
(LOW SPARSE ITER SOR 3)
(MID SPARSE ITER SSOR-CG 3)
(HIGH SPARSE ITER SSOR-CG 3)
(VLOW DENSE DIR SPARSE 3)
(LOW DENSE DIR BAND-N 3)
(MID DENSE ITER SOR 3)
(VLOW VDENSE DIR SPARSE 3)
(LOW VDENSE DIR SPARSE 3)
(MID VDENSE DIR SPARSE 3)
(VLOW SATURATED DIR BAND-RCM 3)
(LOW SATURATED DIR BAND-RCM 3)
(VLOW SPARSE ITER SSOR-CG 3)
(LOW SPARSE ITER SSOR-CG 3)
(MID SPARSE ITER SOR 3)
(HIGH SPARSE ITER SOR 3)
(VLOW DENSE ITER SSOR-CG 3)
(LOW DENSE ITER SSOR-CG 3)
(MID DENSE ITER SSOR-CG 3)
(VLOW VDENSE DIR BAND-N 3)
(LOW VDENSE DIR BAND-RCM 3)
(VLOW SATURATED DIR SPARSE 3)
(LOW SATURATED DIR SPARSE 3)
(VHIGH SPARSE DIR BAND-N OVF)

(HIGH DENSE DIR SPARSE 2)
(VHIGH DENSE ITER SOR 2)
(VLOW VDENSE ITER SSOR-CG 2)
(LOW VDENSE ITER SSOR-CG 2)
(MID VDENSE ITER SSOR-CG 2)
(HIGH VDENSE ITER SOR 2)
(VLOW SATURATED DIR ENV 2)
(LOW SATURATED DIR BAND-N 2)
(MID SATURATED DIR BAND-N 2)
(HIGH SATURATED ITER SSOR-CG 2)
(VLOW SPARSE ITER J-CG 2)
(LOW SPARSE DIR BAND-N 2)
(MID SPARSE DIR BAND-N 2)
(HIGH SPARSE DIR BAND-N 2)
(VLOW DENSE DIR BAND-RCM 2)
(LOW DENSE DIR SPARSE 2)
(MID DENSE DIR BAND-RCM 2)
(HIGH DENSE ITER SSOR-CG 2)
(VLOW VDENSE ITER SOR 2)
(HIGH SATURATED DIR SPARSE OVF))
(VHIGH SPARSE DIR BAND-RCM OVF)
(VHIGH SPARSE DIR ENV OVF)
(VHIGH SPARSE DIR SPARSE OVF)
(HIGH DENSE DIR BAND-N OVF)
(HIGH DENSE DIR BAND-RCM OVF)
(VHIGH DENSE DIR BAND-N OVF)
(VHIGH DENSE DIR BAND-RCM OVF)
(VHIGH DENSE DIR ENV OVF)
(VHIGH DENSE DIR SPARSE OVF)
(MID VDENSE DIR BAND-RCM OVF)
(HIGH VDENSE DIR BAND-N OVF)
(HIGH VDENSE DIR BAND-RCM OVF)
(HIGH VDENSE DIR ENV OVF)
(HIGH VDENSE DIR SPARSE OVF)
(MID SATURATED DIR BAND-RCM OVF)
(MID SATURATED DIR ENV OVF)
(HIGH SATURATED DIR BAND-N OVF)
(HIGH SATURATED DIR BAND-RCM OVF)
(HIGH SATURATED DIR ENV OVF)

Otrzymane reguły mają bardziej czytelną postać i przedstawiają się następująco:

```
RULE 1:
    IF ALG = ENV THEN
        IF DOF = VLOW OR LOW THEN
            IF MATRIX = SPARSE OR DENSE OR VDENSE THEN 1 ELSE
            IF MATRIX = SATURATED THEN 2 ELSE
        IF DOF = MID THEN
            IF MATRIX = SPARSE OR DENSE OR VDENSE THEN 1 ELSE
            IF MATRIX = SATURATED THEN OVF ELSE
        IF DOF = HIGH
            THEN IF MATRIX = SPARSE OR DENSE THEN 1 ELSE
            IF MATRIX = VDENSE OR SATURATED THEN OVF ELSE
        IF DOF = VHIGH THEN OVF

RULE 2:
    IF ALG = J-CG THEN
        IF DOF = VLOW THEN
            IF MATRIX = SPARSE OR DENSE THEN 2 ELSE
            IF MATRIX = VDENSE OR STURATED THEN 1 ELSE
        IF DOF = LOW THEN
            IF MATRIX = SPARSE THEN 2 ELSE
            IF MATRIX = DENSE OR VDENSE OR SATURATED THEN 1 ELSE
        IF DOF = MID OR HIGH OR VHIGH THEN 1
```

```
RULE 3:
    IF ALG = BAND-N THEN
        IF DOF = VLOW THEN
            IF MATRIX = SPARSE OR DENSE OR SATURATED THEN 1 ELSE
            IF MATRIX = VDENSE THEN 3 ELSE
        IF DOF = LOW THEN
            IF MATRIX = SPARSE OR VDENSE OR SATURATED THEN 2 ELSE
            IF MATRIX = DENSE THEN 3 ELSE
        IF DOF = MID THEN 2 ELSE
        IF DOF = HIGH THEN
            IF MATRIX = SPARSE THEN 2 ELSE
            IF MATRIX = DENSE OR VDENSE OR SATURATED THEN OVF ELSE
        IF DOF = VHIGH THEN OVF

RULE 4:
    IF ALG = BAND-RCM THEN
        IF DOF = VLOW THEN
            IF MATRIX = DENSE OR VDENSE THEN 2 ELSE
            IF MATRIX = SATURATED THEN 3 ELSE
        IF DOF = LOW THEN
            IF MATRIX = SPARSE THEN 1 ELSE
            IF MATRIX = DENSE OR VDENSE OR SATURATED THEN 3 ELSE
        IF DOF = MID THEN
            IF MATRIX = SPARSE OR DENSE THEN 2 ELSE
            IF MATRIX = VDENSE OR SATURATED THEN OVF ELSE
        IF DOF = HIGH THEN
            IF MATRIX = SPARSE THEN 2 ELSE
            IF MATRIX = DENSE OR VDENSE OR SATURATED THEN OVF ELSE
        IF DOF = VHIGH THEN OVF
```

RULE 5:

IF ALG = SSOR-CG THEN
IF MATRIX = SPARSE THEN
IF DOF = VLOW OR LOW OR MID OR HIGH THEN 3 ELSE
IF DOF = VHIGH THEN 1 ELSE
IF MATRIX = DENSE THEN
IF DOF = VLOW OR LOW OR MID THEN 3 ELSE
IF DOF = HIGH THEN 2 ELSE
IF MATRIX = SPARSE THEN
IF DOF = VLOW OR LOW OR MID OR HIGH THEN 3 ELSE
IF DOF = VHIGH THEN 1 ELSE
IF MATRIX = DENSE THEN
IF DOF = VLOW OR LOW OR MID THEN 3 ELSE
IF DOF = HIGH THEN 2 ELSE
IF MATRIX = VDENSE THEN
IF DOF = VLOW OR LOW OR MID THEN 2 ELSE
IF DOF = HIGH THEN 1 ELSE
IF MATRIX = SATURATED THEN 2

RULE 6:

IF ALG = SOR THEN
IF MATRIX = SPARSE THEN
IF DOF = VLOW OR VHIGH THEN 2 ELSE
IF DOF = LOW OR MID OR HIGH THEN 3 ELSE
IF MATRIX = DENSE THEN
IF DOF = VLOW OR LOW OR VHIGH OR VHIGH THEN 2 ELSE
IF DOF = MID THEN 3 ELSE
IF MATRIX = VDENSE THEN 2 ELSE
IF MATRIX = SATURATED THEN
IF DOF = VLOW THEN 2 ELSE
IF DOF = LOW OR MID OR HIGH THEN 1

RULE 7:

IF ALG = SPARSE THEN
IF DOF = VLOW THEN 3 ELSE
IF DOF = LOW THEN
IF MATRIX = SPARSE OR DENSE THEN 2 ELSE
IF MATRIX = VDENSE OR SATURATED THEN 3 ELSE
IF DOF = MID THEN 2
IF DOF = HIGH THEN
IF MATRIX = SPARSE OR DENSE THEN 2 ELSE
IF MATRIX = SATURATED THEN OVF ELSE
IF DOF = VHIGH THEN OVF

Przede wszystkim zauważmy, że w obu przypadkach nie udało się otrzymać generalnej, bezpośredniej zależności pomiędzy metodami iteracyjnymi a bezpośrednimi (atrybut *METH*).

Reguła pierwsza mówi, że algorytm typu *ENV* daje najlepsze wyniki spośród metod bezpośrednich, lecz już przy średnich liczbach stopni swobody zaczyna zagrażać niebezpieczeństwo przekroczenia dostępnej pamięci operacyjnej.

Według reguły drugiej algorytm *J – CG* dla niewielkich liczb stopni sobody zachowuje się lepiej przy bardziej wypełnionych macierzach współczynników zaś jego efektywność wzrasta wraz ze wzrostem liczby równań i staje się niezależna od gęstości macierzy. Algorytm *J – CG* jest niewątpliwie najbardziej efektywnym spośród rozważanych algorytmów.

Regułę trzecią można interpretować następująco: algorytm *BAND – N* staje się coraz mniej efektywny zarówno ze wzrostem stopnia wypełnienia macierzy jak i liczby równań prowadząc przy większych zadaniach do błędu nadmiaru.

Według prawa czwartego charakter zachowania algorytmu *BAND – RCM* jest podobny jak *BAND – N* przy nieznacznie niższej efektywności.

Kolejne dwie reguły: piąta i szósta dotyczą algorytmów iteracyjnych odpowiednio *SSOR – CG* i *SOR* i różnią sie od poprzednich tym, że podstawowym kryterium klasy-

fikacji jest stopień wypełnienia macierzy a nie, jak poprzednio - liczba równań. Efektywność obu metod jest zbliżona i wzrasta wraz ze wzrostem liczby równań i gęstością macierzy.

Ostatnia reguła dotyczy algorytmu *SPARSE* i wskazuje na wysokie ryzyko wystąpienia błędu nadmiaru zaś efektywność spada ze wzrostem gęstości wypełnienia.

Ogólność powyższych wniosków jest ograniczona do rozpatrywanego zbioru przykładów. Szczególnie istotne jest, że wszystkie porównania miały miejsce dla zadań płaskich. Należy się spodziewać, że w zagadnieniach jednowymiarowych metody bezpośrednie dawałyby lepsze wyniki podczas gdy nie ulega wątpliwości, że w zadaniach przestrzennych przewaga metod iteracyjnych zaznaczyła by się jeszcze wyraźniej. Ponadto, z uwagi na możliwość wykorzystania zasady superpozycji przez jednokrotną triangularyzację macierzy dla różnych wariantów prawych stron (tzw. "schematów obciążenia"), metody bezpośrednie są preferowane w zadaniach liniowych. Ostatnia uwaga dotyczy faktu, że powyższe porównania dotyczą nie tyle algorytmów co ich implementacji w konkretnych programach. Ma to niewątpliwą zaletę w praktycznym wykorzystaniu, tym niemniej wpływa na ogólność uzyskanych wyników.

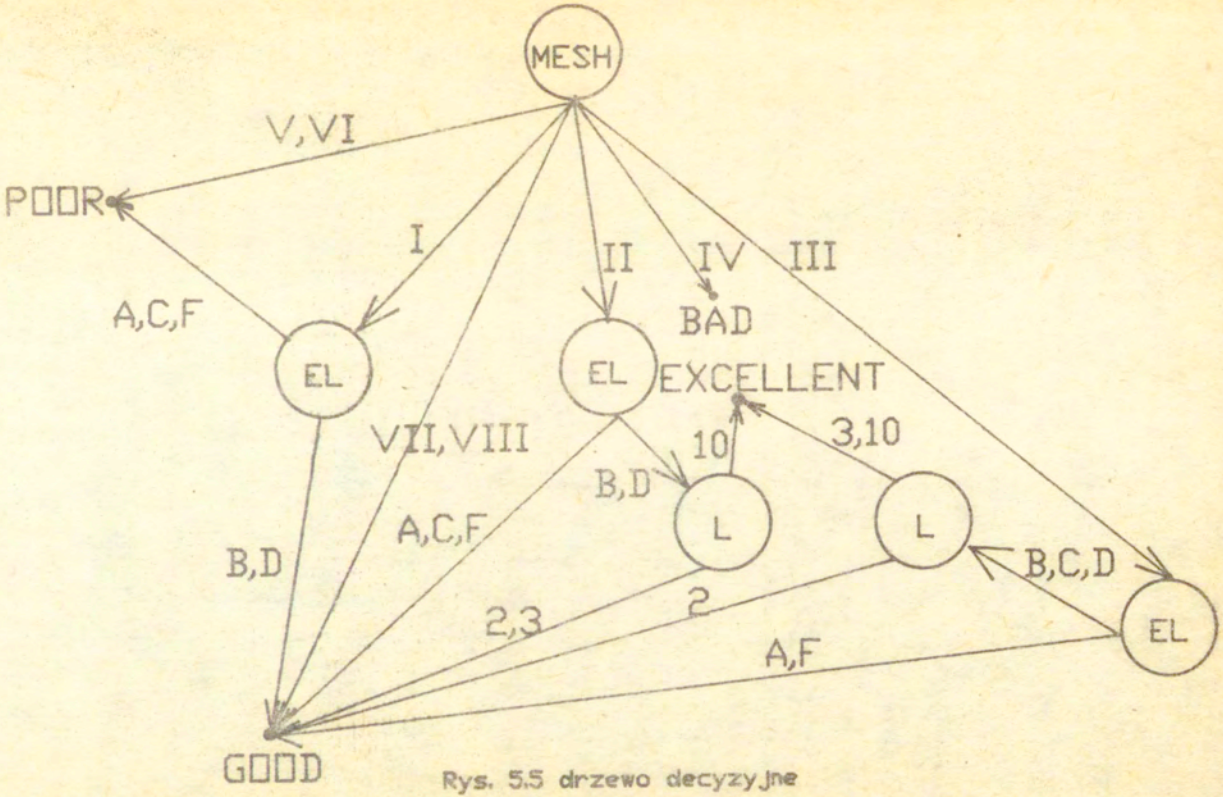
Algorytm indukcyjny znacznie ułatwia akwizycję wiedzy, tym niemniej warunkiem włączenia generowanych reguł do bazy wiedzy jest dostarczenie przez eksperta uzasadnienia każdej reguły tego rodzaju. Algorytm indukcyjny nie dostarcza uzasadnienia reguł gdyż operuje jedynie na poziomie syntaktycznym. Bezkrytyczne stosowanie indukowanych reguł może więc prowadzić do utraty kontroli nad systemem, gdyż poszczególne reguły mogą wynikać z pewnych przypadkowych prawidłowości.

5.3 Praktyczne wykorzystanie otrzymanych wyników.

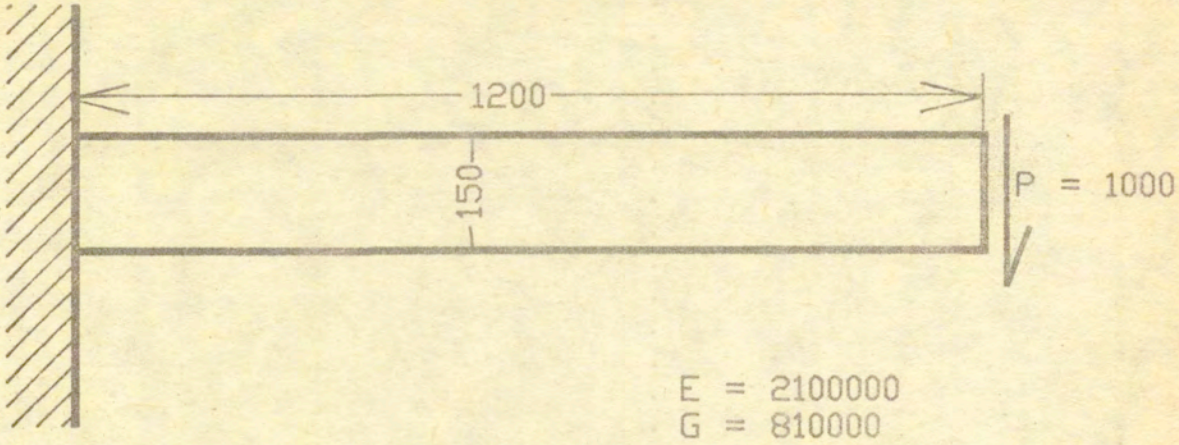
W pierwszych dwóch częściach niniejszego rozdziału przedstawiliśmy realizację koncepcji systemu ekspertowego sterującego nieliniowymi obliczeniami MES oraz przykłady automatycznej generacji reguł opisujących specyficzne własności pewnych typowych elementów procesu numerycznego rozwiązywania zagadnień mechaniki. Poniższy przykład dotyczy oceny praktycznej przydatności reguł otrzymanych w punkcie 5.2.1 w wyniku zastosowania programu indukcyjnego, które przedstawiamy na rys. 5.5 w postaci drzewa decyzyjnego. Z drzewa tego skorzystamy do przygotowania zbioru danych programu MES2S liniowej analizy płaskich zagadnień teorii sprężystości [18], a więc innego programu niż ten, którym posłużyli się autorzy problemów wzorcowych zamieszczonych w pracy [51]. W szczególności MES2S dysponuje jedynie dwoma typami elementów skończonych. Wydruk programu w języku PASCAL wraz z zapisem zbiorów danych i wyników końcowych dla poniższego przykładu zamieszczamy w końcowej części pracy.

Rys. 5.6 przedstawia dźwigar o wydłużeniu $L = 8$. Celem wstępnego, symbolicznego etapu analizy jest zapewnienie największej dokładności obliczeń poprzez odpowiedni dobór parametrów przy pomocy drzewa decyzyjnego. Rys. 5.7 przedstawia kolejne etapy wnioskowania "wst cz", poczynając od celu głównego, którym jest z początku otrzymanie wartości *EXCELLENT* atrybutu *RESULT*. W wypadku załamania procesu wnioskowania przyjmiemy za cel główny uzyskanie przynajmniej wartości *GOOD* itd.

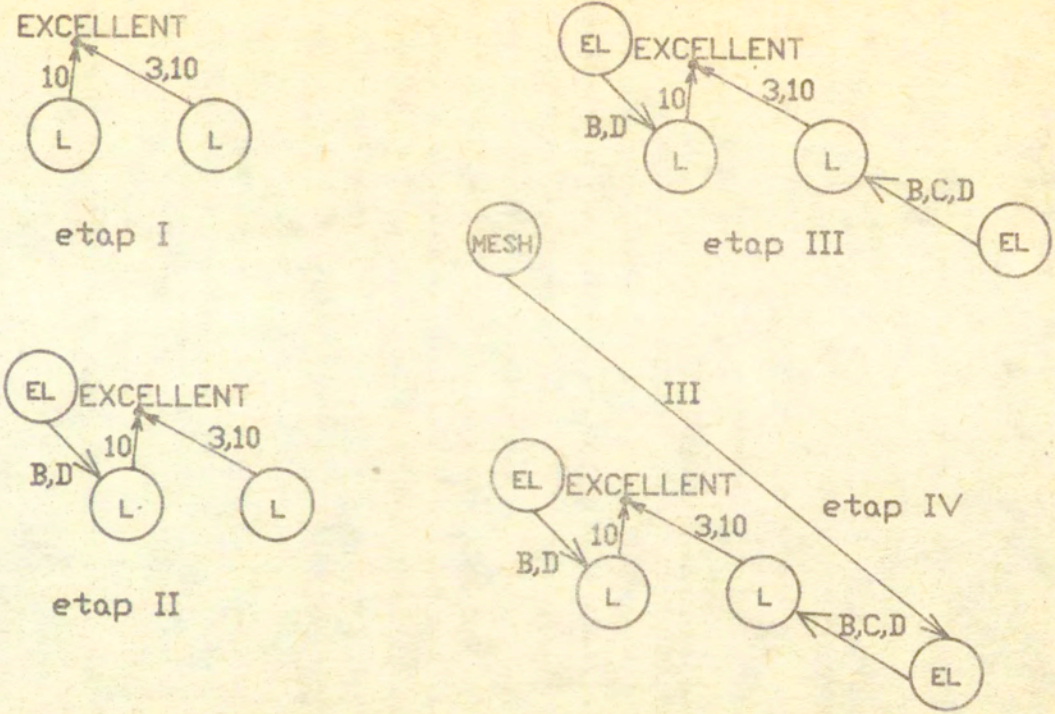
Do liścia odpowiadającego wartści *EXCELLENT* prowadzą dwie gałęzie (etap *I* na rys. 5.7). Przyjmujemy, że wartość L wynosząca w naszym przypadku 8 jest zbliżona do wartości 10. Wybieramy lewą gałąź. Do węzła w którym testowany jest atrybut L wchodzi tylko jedna gałąź odpowiadająca wartościom B oraz D atrybutu EL. Kontynuacja przeszukiwania drzewa tą drogą nie jest możliwa, gdyż program MES2S nie dysponuje odpowiednimi typami elementów skończonych. Proces wnioskowania zawraca więc do ostatniego punktu, w którym istniała możliwość wyboru, w tym wypadku spowrotem do liścia *EXCELLENT* (etap *II* na rys. 5.7). Wybieramy gałąź 3,10 atrybutu L i kontynuujemy (etap *III* na rys. 5.7) przez węzeł EL, gałęzią odpowiadającą typom B, C, D elemntów skończonych. W programie MES2S istnieje element typu C, więc proces wnioskowania postępuje dalej gałęzią odpowiadającą wartości *III* atrybutu



Rys. 5.5 drzewo decyzyjne



Rys. 5.6 dźwigar o wydłuzeniu L=8



Rys. 5.7 kolejne etapy wnioskowania 'wstecz'

MESH (etap *IV* na rys. 5.7). Przygotowujemy zbiór danych programu MES2S z wykorzystaniem elementu typu *C* i siatki typu *III*. W wyniku obliczeń otrzymujemy wartość 0.375 przemieszczenia końca wspornika, co po uwzględnieniu dyskretyzacji atrybutu *RESULT* daje istotnie wartość *EXCELLENT*.

Dla porównania przeprowadzono obliczenia dla elementu trójkątnego o stałym naprężeniu i siatki typu *IV*. Otrzymana wartość 0.520 ugięcia w prowadzi do wartości *POOR* atrybutu *RESULT*, zgodnie z drzewem decyzyjnym z rys. 5.5. Tak więc reguły otrzymane w wyniku zastosowania procesu indukcyjnego do zbioru problemów wzorcowych mogą służyć do automatycznej oceny użytkowych pakietów MES.

podsumowanie

Wiele rzeczywistych zagadnień spotykanych w nauce i przemyśle wymaga technik łączących metody AI z konwencjonalnymi metodami obliczeniowymi. Z reguły z tych problemów wyodrębnić można pewne główne podproblemy sprowadzalne do formalnego modelowania jakościowego i numerycznych metod obliczeniowych takich jak: analiza numeryczna, statystyka i symulacja jakościowa. Często jednak ogólny proces rozwiązania wymaga głębszego wejrzenia i rozumowania jakościowego celem otrzymania rozwiązania lub interpretacji wyników procesu obliczeniowego. Tak więc należy oczekiwać, że warunkiem rozwiązania złożonych zagadnień jest sprzężenie w systemach ekspertowych metod symbolicznych z numerycznymi. Podkreślamy, że nie należy oczekiwać, że obliczenia symboliczne pod postacią konwencjonalnych systemów ekspertowych mogą przewyższyć lub wyeliminować istniejące formy obliczeń numerycznych. Rozsądniejsze podejście polega na stopniowym łączeniu numerycznych i symbolicznych narzędzi obliczeniowych by tworzyć nową, sprawniejszą generację oprogramowania narzędziowego. Te nowe narzędzia, wykorzystujące architektury systemów z bazą wiedzy dostarczą nowych zdolności rozwiązywania problemów i pomogą użytkownikowi łatwiej i z większym zrozumieniem użyć metod analizy ilościowej.

Wykorzystanie metod A.I. wskazuje drogę uzyskania narzędzi działających na różnych poziomach abstrakcji. Na najwyższym poziomie abstrakcji oprogramowanie numeryczne pozostaje do końca ukryte przed użytkownikiem, nacisk położony jest na automatyczny dobór metod spośród mnogości obsługiwanego oprogramowania. Prze-

chodząc do niższych poziomów abstrakcji zaawansowany użytkownik będzie mógł programując z pomocą systemu brakujące partie kodu generować własne metody, zgodnie ze zmieniającym się opisem problemu.

DODATEK 1
PROGRAMOWANIE MODUŁÓW SZTUCZNEJ INTELIGENCJI NA MIKROKOMPUTERACH IBM PC/XT/AT

Budowa systemów sprzężonych nakłada określone wymagania na oprogramowanie narzędziowe, które musi zapewnić właściwą szybkość przetwarzania informacji przez moduł AI tak, aby nie spowalniać wywoływanych procesów numerycznych. Niewiele spośród dostępnych na IBM PC powłok systemów ekspertowych posiada zdolność wywoływania oprogramowania zewnętrznego. Efektywność tych narzędzi podobnie zresztą jak i efektywność mikrokomputerowych implementacji klasycznych języków AI - LISPu i PROLOGu ogranicza praktyczny zakres ich zastosowania do budowy interfejsów użytkownika. Wynika to z faktu, że interfejs zużywa najwięcej czasu czekając na odpowiedź użytkownika przy względnie niewielkiej liczbie wymienianych informacji, podczas gdy procesy numeryczne przebiegają szybko i wymagają intensywnej komunikacji ze zintegrowaną bazą danych.

Właściwymi językami implementacyjnymi mikrokomputerowych systemów sprzężonych powinny być z powyższych względów tradycyjne języki wyższego rzędu: FORTRAN, PASCAL, C. Wybór języka w konkretnym zastosowaniu jest sprawą subiektywną, w niniejszej pracy decyzję o wyborze języka PASCAL podjęto kierując się łatwością z jaką poszczególne języki pozwalają budować charakterystyczne dla AI struktury danych oraz pewnymi dodatkowymi względami wynikającymi z porównania dostępnych kompilatorów: efektywnością środowiska operacyjnego, wielkością modułu wynikowego i szybkością przetwarzania.

Podstawową strukturą danych w operacjach symbolicznych leżącą u podstaw języka LISP są wyrażenia kropkowe: atomy i listy. Wyrażenia kropkowe definiuje się rekurencyjnie:

1. *atomy są wyrażeniami kropkowymi*

2. *para wyrażeń kropkowych jest wyrażeniem kropkowym.*

Poniższy fragment programu napisanego w języku PASCAL definiuje przy pomocy rekordu wariantowego zarówno atomy gdy przełącznik *rodzaj* przybiera wartość *jeden*

jak i listy, gdy *rodzaj* przyjmuje wartość *dwa*.

```
type w = ^w1;
     w1 =record
              case rodzaj : (jeden, para) of
                   jeden : (printname : string80; property : w);
                   para : (a : w; b : w);
         end;
```

Atom charakteryzuje się nazwą *printname* i listą własności *property*. Wyróżniamy pewien szczególny atom *NIL*, który odgrywa ważną rolę w operacjach na wyrażeniach kropkowych. W szczególności oznacza on listę pustą oraz służy do zakańczania list. Przy powoływaniu kolejnych atomów funkcją new_atom:

```
function new_atom(symbol : string80) : w;
var v : w;
begin
     new(v);
     v^.rodzaj := jeden;
     v^.printname := symbol;
     v^.property := NIL;
     new_atom := v
end; {function  new_atom(symbol : string80) : w}
```

nadana zostaje nazwa przekazana przez parametr *symbol* zaś lista własności inicjowana jest jako *NIL*.

Elementy składowe list - kontrolki tworzymy funkcją cons:

```
function cons(x1,x2 : w) : w;
var y : w;
begin
    new(y);
    with y^ do
            begin
                rodzaj := para;
                a := x1;
                b := x2;
            end;
    cons := y
end; {function  cons(x1,x2 : w) : w}
```

przy każdorazowym wywołaniu funkcji cons tworzone są dwie komórki wskazujące odpowiednio na pierwszy i drugi argument funkcji. Każdy z argumentów może być atomem lub listą.

dod1.1 podstawowe operacje na listach

Dwie podstawowe funkcje car i cdr umożliwiają przemieszczanie się po listach. Funkcja car zwraca pierwszy argument listy:

```
function car(x : w) : w;
begin
    car := x^.a
end;{function  car(x : w) : w}
```

zaś funkcja cdr zwraca resztę listy, tj. listę bez jej pierwszego argumentu:

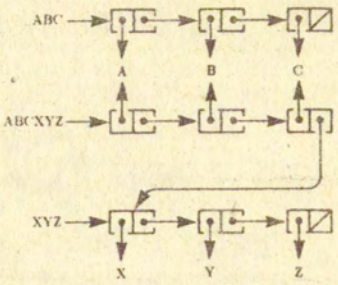
```
function cdr(x : w) : w;
begin
    cdr := x^.a
end; {function  cdr(x : w) : w}
```

dod1.2 podstawowe predykaty

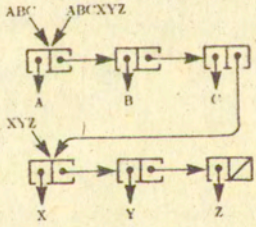
Zdefiniowane zostały predykaty - funkcje dające w wyniku wartość logiczną. Zasadnicze z nich sprawdzają czy argument nie ma wartości *NIL* (funkcja **null**), czy jest atomem (funkcja **atom**), czy też parą kropkową (funkcja **consp**).

```
function null(x : w) : boolean;
begin
    null := x = nil
end; {function  null(x : w)}

function atom(x : w) : boolean;
begin
    if null(x) then atom := true
               else atom := x.rodzaj = jeden
end; {function  atom(x : w) : boolean}

function consp(x : w) : boolean;
begin
    consp := x.rodzaj = para
end; {function  consp(x : w) : boolean}
```

Funkcja **equal** sprawdza tożsamość dwóch obiektów. Opiszemy ją bardziej szczegółowo, gdyż jest ona zbudowana w charakterystyczny, rekurencyjny sposób, typowy dla większości funkcji i procedur programu. Wprawdzie operacje rekurencyjne są mniej efektywne, to jednak program staje się krótszy i bardziej czytelny. W pierwszym kroku **equal** bada pierwszy argument. Jeżeli jest on atomem to badany jest drugi argument. Jeżeli on też jest atomem, to porównuje się nazwy obu atomów, w przeciwnym razie porównanie daje wartość negatywną. Jeżeli pierwszy argument jest listą, a drugi atomem to funkcja daje wartość ujemną. Gdy obydwa argumenty są listami to porównuje się rekurencyjnie tj. przy pomocy kopii tej samej funkcji **equal** pierwsze elementy i reszty list. Tak więc **equal** równolegle z porównaniem wartości sprawdza tożsamość struktur obiektów.



Rys. dod1.1 funkcja APPEND



Rys. dod1.2 funkcja NCONC

```
function equal(x, y : w) : boolean;
begin
    if atom(x) then
      if atom(y) then equal := (x = y)
        else equal := false
    else
      if atom(y) then equal := false
        else
          if equal(car(x), cdr(y))
            then equal := equal(car(x), cdr(y))
            else equal := false
end; {function  equal(x, y : w)}
```

dod1.3 niedestrukcyjne i destrukcyjne operacje na listach

W programie występują dwa rodzaje funkcji. Jedne z nich działają na zasadzie tworzenia nowych obiektów inne na zasadzie modyfikacji istniejących. Dwie funkcje append i nconc dają dla takich samych argumentów ten sam wynik. Tym niemniej ich działanie jest różne i prowadzi do zasadniczo odmiennych skutków ubocznych.

Funkcja append łączy dwie listy. Rys.(dod1.1) przedstawia działanie procedury. W pierwszym etapie kopiowany jest pierwszy argument. Następnie funkcja poszukuje rekurencyjnie końca kopii i podstawia drugi argument w miejsce *NIL*.

```
function append(x, y : w) : w;
begin
    if null(x) then append := y
    else append := cons(car(x), append(cdr(x), y))
end; {function  append(x, y : w)}
```

Funkcja nconc rwnież w wyniku daje sumę dwóch list, tym niemniej nie jest tworzony nowy obiekt. Suma list uzyskiwana jest przez zmianę wskaźników w efekcie czego ulega również zmianie wartość pierwszego argumentu, który w wyniku działania proce-

dury przybiera wartość połączonych list. Procedura **nconc** działa szybciej niż **append**, nie zużywa przy tym pamięci – pierwszy argument nie jest kopiowany. Użycie procedury jest jednak niebezpieczne gdyż zmienia ona wszystkie obiekty zawierające wskaźniki do pierwszego argumentu (rys.dod1.2).

```
function nconc(x, y : w) : w;
begin
    nconc := x;
    while not null(cdr(x)) do x := cdr(x);
    x.b := y
end; {function  nconc(x, y : w)}
```

dod1.4 program INDUKTOR

Program INDUKTOR służy do automatycznego generowania reguł klasyfikacyjnych dla zbiorów przykładów opisanych listami wartości atrybutów (por. rozdziały 4 i 5). Program otrzymuje na wejściu listę przykładów. Pierwszym elementem listy jest lista atrybutów. Kolejne elementy to przykłady klasyfikacji opisane przez listy wartości atrybutów. Każda z przykładowych list wartości powinna mieć taką samą długość jak lista atrybutów. Liczba przykładów ograniczona jest jedynie przez dostępną pamięć komputera. Dane do programu indukcyjnego INDUKTOR zapisujemy w pamięci zewnętrznej w dowolnym zbiorze. Przykładowe dane:

((weather stay wet car umbrella
(dry inside not not no
(wet outside not not yes
(bluster outside not yes no
(wet inside not not no))

możemy zapisać w zbiorze *weather*.

Po uruchomieniu programu użytkownik pytany jest nazwę zbioru danych. Program wyświetla na ekranie listę danych a następnie w kolejnych wierszach atrybuty i ich zakresy wartości oraz przewidywaną zawartość informacyjną generowanego zbioru reguł. Jeżeli użytkownik nie zauważy błędu w danych to podaje nazwę zbiory wynikowego, w przeciwnym razie może przerwać działanie programu i poprawić dane. Podanie nazwy *con* spowoduje wyświetlenie wyników na ekranie. Przy dużych zbiorach danych i długotrwałym przetwarzaniu korzystniej jest zapisywać wyniki na dysku przez podanie nazwy zbioru dyskowego lub na drukarce przez podanie nazwy *prn*. Poniżej zamieszczamy zapis przebiegu programu:

data in file ... *weather*

((weather stay wet car umbrella)

(dry inside not not no)

(wet outside not not yes)

(bluster outside not yes no)

(wet inside not not no))

wet ... ((RANGE not))

stay ... ((RANGE outside inside))

umbrella ... ((RANGE yes no))

weather ... ((RANGE bluster wet dry))

car ... ((RANGE yes not))

info_content of examples : 0.8113

output tree to... *con*

RULE 1:IF weather =dry THEN no

RULE 2:IF weather =wet THEN

IF stay =inside THEN no

ELSE IF stay =outside THEN yes

RULE 3:IF weather =bluster THEN no

gdzie odpowiedzi użytkownika zostały zaznaczone pismem pochyłym.

Zbiór danych *weather* i powyższa ilustracja może posłużyć do sprawdzenia poprawności uruchamiania programu z wydruku.

dod1.5 Prosty mechanizm wnioskujący.

Program DEDUKTOR służy do demonstracji idei wnioskowania wstecz i wprzód. Operuje on na uproszczonej bazie wiedzy zapisanej w postaci listy reguł. Reguły zapisane są również w postaci list: pierwszym elementem jest lista warunków reszta listy to wnioski. Tak więc reguła

((*a b c*) *d e*)

oznacza:

JEŻELI
a ORAZ b ORAZ c
TO
d ORAZ e.

Warunek jest spełniony, jeżeli użytkownik określi jego współczynnik pewności lub jeżeli współczynnik pewności zostanie nadany w wyniku zadziałania reguły w której rozpatrywany warunek występuje po stronie konkluzji. Współczynnik pewności to liczba z przedziału (0,1). Program rozpoczyna działanie od żądania określenia celu konsultacji. Należy podać dowolną z konkluzji. Następnie uruchamiany jest mechanizm wnioskowania wstecz (procedura *bckwd*) naprzemiennie z mechanizmem wnioskowania wprzód (procedura *fwd*). Reguła jest wykonywana jeżeli jej współczynnik pewności przekracza wartość progową (stała *threshold*) ustaloną w programie na 0.2. Wykonanie reguły polega na nadaniu współczynnikom pewności wszystkich jej konkluzji wartości wyliczonej dla lewej strony. Wartość współczynnika pewności reguły wyznacza się jako minimum wartości współczynników pewności przyporządkowanych poszczególnym warunkom reguły. Użytkownik może odmówić podania współczynnika pewności warunku odpowiadając *unk*. Wówczas program próbuje sam wyznaczyć potrzebną wartość umieszczając rozpatrywany warunek na stosie celów.

Kontekst implementowany jest w postaci listy własności przyporządkowanej kolejno rozpatrywanym zmiennym. Wyświetlenie aktualnego kontekstu można uzyskać na

każdym etapie konsultacji, poprzez wczytanie znaku ?. Na kontekst składają się wszystkie rozpatrzone warunki i powzięte konkluzje. Jeżeli zmienna ma przyporządkowaną wartość współczynnika pewności to jest on stowarzyszony z informacją skąd ta wartość pochodzi. Jeżeli została ona nadana przez użytkownika to atrybut *SOURCE* otrzymuje wartość *USER*, jeżeli zaś została wyliczona przez program, atrybutowi *SOURCE* zostaje przyporządkowana wykorzystana reguła. Zmienne, które stają się celami kolejnych rzędów w rekurencyjnym procesie wnioskowania wstecz i nie mają nadanych bądź wyliczonych wartości współczynników pewności opatrywane są atrybutem *GOAL*. Wartością atrybutu *GOAL* jest dla celu głwnego *MAIN*, zaś dla celu niższego rzędu reguła dla której spełnienia jest on potrzebny.

Celem wyjaśnienia mechanizmu działania programu posłużymy się przykładowym zestawem reguł, który może również posłużyć jako zbiór danych testowych.

```
((a b)c d)
((f i) e)
((e f)e r)
((e)g)
((e)pp)
((a)f)
((a)b)).
```

Po uruchomieniu programu wczytujemy nazwę zbioru danych, w tym wypadku *IN*. Przykładowy dialog z użytkownikiem może przedstawiać się następująco:

data in file... *in*

enter main goal... *r*

found rule ... ((e f)e r)

checking premise <CF> : e ... *unk*

checking premise <CF> : f ... ?

c ... ((STATUS: GOAL)(SOURCE:(c f)e r))

r ... ((STATUS: GOAL)(SOURCE: MAIN))

checking premise <CF> : f ... *unk*

found rule ... ((a b)c d)

checking premise <CF> : a ... ?

c ... ((STATUS: GOAL)(SOURCE:(c f)e r))

f ... ((STATUS: GOAL)(SOURCE:(c f)e r))

r ... ((STATUS: GOAL)(SOURCE: MAIN))

checking premise <CF> : a ... 0.25

checking premise <CF> : b ... ?

a ... ((CF 0.25)(SOURCE: USER))

c ... ((STATUS: GOAL)(SOURCE:(c f)e r))

f ... ((STATUS: GOAL)(SOURCE:(c f)e r))

r ... ((STATUS: GOAL)(SOURCE: MAIN))

checking premise <CF> : b ... *unk*

FIRE ! ((a)f)with cf...2.50E-01

FIRE ! ((a)b)with cf...2.50E-01

found rule ... ((a b)c d)

FIRE ! ((a b)c d)with cf...2.50E-01

FIRE ! ((c f)e r)with cf...2.50E-01

FIRE ! ((e)g)with cf...2.50E-01

FIRE ! ((c)pp)with cf...2.50E-01

found rule ... NIL

pp ... ((CF 2.50E-01)(STATUS: DEDUCED)(SOURCE:(c)pp))

a ... ((CF 0.25)(SOURCE: USER))

b ... ((CF 2.50E-01)(STATUS: DEDUCED)(SOURCE:(a)b))

c ... ((CF 2.50E-01)(STATUS: DEDUCED)(SOURCE:(a b)c d))

d ... ((CF 2.50E-01)(STATUS: DEDUCED)(SOURCE:(a b)c d))

e ... ((CF 2.50E-01)(STATUS: DEDUCED)(SOURCE:(c f)e r))

f ... ((CF 2.50E-01)(STATUS: DEDUCED)(SOURCE:(a)f))

g ... ((CF 2.50E-01)(STATUS: DEDUCED)(SOURCE:(e)g))

r ... ((CF 2.50E-01)(STATUS: DEDUCED)(SOURCE:(c f)e r))

gdzie odpowiedzi użytkownika wyróżniono jak poprzednio pismem pochyłym.

Wyświetlane są kolejno rozpatrywane reguły. Jeżeli reguła jest wykonywana, to wyświetlana jest po słowie *FIRE* : wraz z wyliczoną wartością współczynnika pewności.

Wydruk programu INDUKTOR, strona 1 09/12/87

```
 1: const
 2:      threshold=0.2;
 3:      hash_table_size=50;
 4:      hts_sub1=49;
 5: type
 6:      string80=string[80];
 7:      w=^w1;
 8:      w1=record
 9:           case rodzaj:(jeden,para) of
10:             jeden:(printname:string80;property:w);
11:             para:(a:w;b:w);
12:    end;
13: var  data,examples,classes,attributes:w;
14:      NameOfFile,OutputFile,symbol:string80;
15:      i,sea,no_attributes,no_examples:integer;
16:      dev,tree:text;
17:      hash_table:array[0..hts_sub1] of w;
18:      info_tree : real;
19:      RANGE:w;
20: procedure our_str(i:real;var ch:string80);
21: begin
22:      str(i,ch);while ch[1]=' ' do ch := copy(ch,2,length(ch))
23: end;
24:
25: function null(x:w): boolean;
26: begin
27:      null := x=nil
28: end; { function null(x:w) }
29:
30: function car(x:w):w;
31: begin
32:      car := x^.a
33: end; { function car(x:w):w }
34:
35: function cdr(x:w):w;
36: begin
37:      cdr := x^.b
38: end; { function cdr(x:w):w }
39:
40: function caar(x:w):w;
41: begin
42:      caar := car(car(x))
43: end; { function caar(x:e) }
44:
45: function cdar(x:w):w;
46: begin
47:      cdar := cdr(car(x))
48: end; { function cdar(x:e) }
49:
50: function n_th(x:w;n:integer):w;
51: var i:integer;
52: begin
53:      for i := 1 to n - 1 do x := cdr(x);
54:      n_th := car(x)
55: end; { function n_th(x:w;n:integer) }
```

```
56:
57: function last(x:w):w;
58: var y:w;
59: begin
60:     y := cdr(x);
61:     if null(y) then last := car(x) else last := last(y)
62: end; {function last(x:w)}
63: procedure rplacd(x,y:w);
64: begin
65:     x^.b:=y
66: end; { procedure rplacd(x,y:w) }
67:
69: function atom(x:w):boolean;
69: begin
70:     if null(x) then atom := true else atom := x^.rodzaj=jeden
71: end; { function atom(x:w):boolean }
72:
73: function dottedp(x:w):boolean;
74: begin
75:     if atom(x) then dottedp := false
76:                else if null(cdr(x)) then dottedp := false
77:                else if atom(car(x)) and atom(cdr(x))
78:                     then dottedp := true
79:                     else dottedp := false
80: end;
81:
82: function getpname(v:w): string80;
83: begin
84:     if null(v) then getpname := 'NIL' else getpname := v^.printname
85: end; { function getpname(v:w) }
86:
87:
88: function consp(x:w):boolean;
89: begin
90:     consp := x^.rodzaj=para
91: end; { function consp(x:w):boolean }
92:
93: procedure writlist(x:w);
94: begin
95:  if x=nil then write('NIL ')
96:  else if atom(x) then write(x^.printname,' ')
97:  else
98:   begin
99:     write('(');
100:    while not atom(x) do
101:      begin
102:        writlist(car(x));
103:        x := cdr(x);
104:      end;
105:
106:    if not null(x) then write(' . ',x^.printname);
107:       write(')');
108:  end
109: end; { procedure writlist(x:w) }
110:
```

```
111: procedure wrilist(tekst:string80;lista:w);
112: begin
113:   writeln;
114:   write(tekst,char(32));
115:   writlist(lista)
116: end; { procedure wrilist(tekst:string80,lista:w) }
117:
118: function cons(x1,x2:w):w;
119:  var y:w;
120: begin
121: { if maxavail < 100 then writeln('less then 100 paragraphs left'); }
122:   new(y);
123:   y^.rodzaj := para;
124:   y^.a := x1;
125:   y^.b := x2;
126:   cons := y
127: end; { function cons(x1,x2:w):w }
128:
129: function nconc(x1,x2:w):w;
130: begin
131:   nconc := x1;
132:   while not null(cdr(x1)) do x1 := cdr(x1);
133:   x1^.b := x2;
134: end; { function nconc(x1,x2:w)  }
135:
136: function nowyatom(symbol:string80):w;
137: var v:w;
138: begin
139:    new(v);
140:    v^.rodzaj := jeden;
141:    v^.printname := symbol;
142:    v^.property := NIL;
143:    nowyatom := v
144: end; { function nowyatom(symbol:string80):w }
145:
146:
147: function equal(x,y:w): boolean;
148: begin
149:      if atom(x) then
150:           if atom(y) then equal := x=y else equal := false
151:      else
152:           if atom(y) then equal := false
153:      else
154:           if equal(car(x),car(y)) then equal := equal(cdr(x),cdr(y))
155:      else
156:           equal := false
157: end; { function equal(x,y:w) }
158:
159: function assoc(x,y:w):w;
160: begin
161:  if null(y) then assoc := nil
162:  else if equal(x,caar(y)) then assoc := car(y)
163:  else assoc := assoc(x,cdr(y))
164: end; { function assoc(x,y:w) }
165:
```

```
166: function pop(var stack,x:w):boolean;
167: begin
168:   if null(stack) then pop := false
169:          else begin
170:                 pop := true;
171:                 x := car(stack);
172:                 stack := cdr(stack)
173:               end
174: end; { function pop(var stack,x:w):boolean }
175:
176: procedure push(var stack:w;x:w);
177: begin
178:   stack := cons(x,stack)
179: end; { procedure push(var stack,x:w) }
180:
181: procedure init_hash_table;
182: var i:integer;
183: begin
184:   for i := 0 to hts_subl do hash_table[i] := nil
185: end; { procedure init_hash_table }
186:
187: function hash(st:string80): integer;
188: var i,imax,tmp:integer;
189: begin
190:   tmp := 0;
191:   imax := length(st);
192:   for i := 1 to imax do tmp:=tmp+ord(st[i]);
193:   hash:=tmp mod hash_table_size
194: end; { function hash(st:string80) }
195:
196: function intern(nowy:string80): w;
197: var x,znany:w;index:integer;
198: label et;
199: begin
200:      index := hash(nowy);
201:      x := hash_table[index];
202:      while pop(x,znany) do if getpname(znany)=nowy then goto et;
203:      znany := nowyatom(nowy);
204:      push(hash_table[index],znany);
205:  et: intern := znany
206:  end; { function intern(nowy:string80):w }
207:
208: { remove all occurences of a from all sublists of x }
209: function remove(a,x:w): w;
210: begin
211:  if atom(x) then remove := x
212:   else if equal(a,car(x)) then remove := remove(a,cdr(x))
213:   else  remove := cons(remove(a,car(x)),remove(a,cdr(x)))
214: end; { function remove(a,x:w) }
215:
216: function getprop(at,prop:w):w;{ funkcja do sprawdzenia}
217: var temp:w;
218: begin
219:      temp:=assoc(prop,at^.property);
220:      if null(temp) then getprop:=nil else getprop := cdr(temp)
```

```
221: end; { function getprop(at,prop:w) }
222:
223: procedure putprop(at,prop,value:w);
224: label et;
225: var temp:w;
226: begin
227:     temp := at^.property;
228:     while not null(temp) do
229:       begin
230:         if caar(temp)=prop then
231:           begin
232:             rplacd(car(temp),value);
233:             goto et
234:           end;
235:         temp := cdr(temp)
236:       end;
237:     at^.property := cons(cons(prop,value),at^.property);
238:     et:
239: end; { procedure putprop(var at:w;prop,value:w) }
240:
241: function checkstack(x,stack:w):boolean;
242: var y:w;
243: begin
244:   checkstack := false;
245:   while pop(stack,y) do if equal(x,y) then checkstack := true
246: end; {function checkstack(x:w;var stack:w)}
247:
248: function append(x,y:w):w;
249: begin
250:     if null(x) then append := y
251:                else append := cons(car(x),append(cdr(x),y))
252: end; { function append(x,y:w) }
253:
254: function lenlist(x:w): integer;
255: begin
256:     if null(x) then lenlist := 0 else lenlist := succ(lenlist(cdr(x)))
257: end;
258:
259: procedure initstack(var stack:w);
260: begin
261:   stack := nil
262: end; { procedure initstack(var stack:w) }
263:
264: function reverse(x:w):w;
265: var y,z:w;
266: begin
267:     initstack(y);
268:     while pop(x,z) do push(y,z);
269:     reverse := y
270: end; { function reverse(x:w) }
271:
272: function remove_n_th(x:w;n:integer):w;
273: var y,z:w;
274:     i:integer;
275: begin
```

```
276:      initstack(y);
277:      i := 0;
278:      while pop(x,z) do
279:           begin
280:                i := succ(i);
281:                if i <> n then push(y,z)
282:           end;
283:      remove_n_th := reverse(y)
284: end; { function remove_n_th(x:sw) }
285:
286: procedure suaput(var dev:text;var symbol:string80);
287: var ch:char;
288: begin
289:   symbol := '';
290:   repeat
291:     read(dev,ch);write(ch);
292:     if ((ord(ch)=13) or (ord(ch)=10)) then
293:     else symbol := concat(symbol,ch);
294:   until ch in ['(',')',' ',',']
295: end; { procedure suaput(var dev:text;symbol:string80) }
296:
297: function dopisz(var dev:text):sw;
298: begin
299:  suaput(dev,symbol);
300:   case symbol[1] of
301:             '(': dopisz := cons(dopisz(dev),dopisz(dev));
302:             ')': dopisz := nil;
303:             else dopisz := cons(intern(symbol),dopisz(dev));
304:   end { case symbol }
305: end; { function dopisz(dev):sw) }
306:
307: function czytaj(var dev:text):sw;
308: var x:sw;
309: begin
310:   suaput(dev,symbol);
311:   if symbol='(' then x := dopisz(dev)
312:                 else x := intern(symbol);
313:   czytaj := remove(intern(' '),x)
314: end; { function czytaj(var dev:text):sw }
315:
316: function czyt(var dev:text;tekst:string80):sw;
317: begin
318:      writeln;
319:      write(tekst,' --> ');
320:      czyt := czytaj(dev)
321: end; { function czyt(tekst:string80;var dev:text) }
322:
323: function member(x,y:sw): sw;
324: begin
325:  if null(y) then member := nil
326:  else if  equal(x,car(y)) then member := y
327:  else member := member(x,cdr(y))
328: end; { function member(x,y:sw) }
329:
330: procedure show;
```

```
331: var list,tmp,x:w;i:integer;
332: begin
333:      writeln;
334:      for i:=0 to hts_sub1 do
335:       begin
336:        list := hash_table[i];
337:        if not null(list) then
338:         repeat
339:           tmp:=car(list);
340:           if not null(tmp^.property)
341:             then
342:               begin
343:                 write(getpname(tmp):6);write('... ');writlist(tmp^.property);
344:                 writeln
345:               end;
346:           list:=cdr(list)
347:         until null(list)
348:       end;
349:     writeln
350: end;
351:
352: function TextFileExists(var filvar:text;NameOfFile:string80):Boolean;
353: begin
354: assign(filvar,NameOfFile);
355: {$I-}reset(filvar);{$I+}
356: TextFileExists := (ioresult=0);
357: end; {TextFileExists}
358:
359: procedure outp(rule:w);
360: var temp,ne:w;
361: begin
362:      write(tree,
363:       'IF ',getpname(caar(rule)),'=',getpname(cdr(car(rule))),' THEN ');
364:      if atom(cdr(rule)) then write(tree,getpname(cdr(rule)))
365:                         else
366:                           begin
367:                             temp := cdr(rule);
368:                             while pop(temp,ne) do
369:                               begin
370:                                 outp(ne);
371:                                 if not null(temp) then writeln(tree,' ELSE ')
372:                               end;
373:                           end;
374: end; (procedure outp(rule:w) )
375:
376: procedure output(var tree:text;x:w);
377: var rule:w;
378:     i:integer;
379: begin
380:      rewrite(tree);
381:      i := 0;
382:      while pop(x,rule) do
383:           begin
384:               i := succ(i);
385:               write(tree,'RULE ',i,':');
```

```
386:                outp(rule);
387:                writeln(tree);
388:            end;
389:        flush(tree)
390: end; { procedure output(var tree:text;x:w) }
391:
392: function same_class(lista:w;var class:w): boolean;
393: var x,example:w;
394: begin
395:     same_class := true;
396:     x := last(car(lista));
397:     while pop(lista,example) do
398:                 if x <> last(example) then same_class := false;
399:     class := x
400: end;
401:
402: function l2(a:real):real;
403: begin
404:     if a = 0.0 then l2 := 0.0 else l2 := ln(a)/ln(2)
405: end;
406:
407:
408: function tree_info_content(classes:w;var lista:w):real;
409: var a:real;
410:     p,t:integer;
411:     pom,class,example:w;
412: begin
413:     t := lenlist(lista);
414:     a := 0.0;
415:     if t > 0 then
416:       while pop(classes,class) do
417:           begin
418:               p := 0;
419:               pom := lista;
420:               while pop(pom,example) do
421:                     if last(example) = class then p := p+1;
422:               a := a - p/t * l2(p/t)
423:           end;
424:     tree_info_content := a
425: end;
426:
427: function info_content(test:w;var lista:w;no:integer):real;
428: var a:real;
429:     i,t:integer;
430:     sublist,example,value,temp:w;
431: begin
432:     t := lenlist(lista);
433:     a := 0.0;
434:     while pop(test,value) do
435:      begin
436:          temp := lista;
437:          initstack(sublist);
438:          while pop(temp,example) do
439:               if n_th(example,no) = value then push(sublist,example);
440:          a := a + tree_info_content(classes,sublist) * lenlist(sublist)/t;
```

```
441:      end;
442:    info_content := a;
443: end;
444:
445: function best_attribute(lista,attributes:w;var no:integer):w;
446: var attribute:w;
447:    n:integer;
448:    x,y:real;
449: begin
450:    x := 0;
451:    n := 0;
452:    while pop(attributes,attribute) do
453:          begin
454:             n := n + 1;
455:             y := info_content(getprop(attribute,RANGE),lista,n);
456:             if (info_tree - y) > x then
457:                begin
458:                   best_attribute := attribute;
459:                   x := info_tree - y;
460:                   no := n
461:                end
462:          end
463: end;
464:
465: function branch(lista,value:w;no:integer):w;
466: var ts,example:w;
467: begin
468:    initstack(ts);
469:    while pop(lista,example) do
470:          if n_th(example,no) = value then push(ts,remove_n_th(example,no));
471:    branch := ts;
472: end;
473:
474: function process(lista,attributes:w):w;
475: var x,values,value,temp,class,attribute,at:w;
476:    no:integer;
477: begin
478:    if same_class(lista,class) then process := class
479:    else
480:       begin
481:          attribute := best_attribute(lista,attributes,no);
482:          values := getprop(attribute,RANGE);
483:          initstack(temp);
484:          while pop(values,value) do
485:           push(temp,cons(cons(attribute,value),
486:            process(branch(lista,value,no),remove(attribute,attributes))));
487:          process := temp;
488:       end;
489: end;
490:
491: function put_values(attributes:w):w;
492: var attribute,value,x:w;
493:    number:integer;
494: begin
495:    number := 0;
```

```
496:      while pop(attributes,attribute) do
497:           begin
498:                number := number + 1;
499:                initstack(x);
500:                for i := 1 to lenlist(examples) do
501:                    begin
502:                         value := n_th(n_th(examples,i),number);
503:                         if null(member(value,x)) then push(x,value)
504:                    end;
505:                putprop(attribute,RANGE,x);
506:           end;
507:      put_values := x;
508: end; { function values(attributes:w) }
509:
510: begin
511:   init_hash_table;
512:   RANGE := intern('RANGE');
513:   write('data in file ... ');
514:   readln(NameOfFile);
515:   if not textfileexists(dev,NameOfFile) then halt;
516:   clrscr;
517:   write(char(240),' ');
518:   data := czytaj(dev);
519:   attributes := car(data);
520:   examples := cdr(data);
521:   classes := put_values(attributes);
522:   SHOW;
523:   attributes := remove(last(attributes),attributes);
524:   no_attributes := lenlist(attributes);
525:   writeln;
526:   info_tree := tree_info_content(classes,examples);
527:   writeln('info_content of examples : ',info_tree:4:4);
528:   write('output tree to... ');readln(OutputFile);
529:   assign(tree,OutputFile);
530:   output(tree,process(examples,attributes));
531: end.
```

Wydruk programu DEDUKTOR, strona 1 09/12/87

```
 1: const
 2:      threshold=0.2;
 3:      hash_table_size=50;
 4:      hts_sub1=49;
 5: type
 6:      string80=string[80];
 7:      w=^w1;
 8:      w1=record
 9:           case rodzaj:(jeden,para) of
10:             jeden:(printname:string80;property:w);
11:             para:(a:w;b:w);
12:    end;
13: var  rules,eaingoal:w;
14:      NameOfFile,symbol:string80;
15:      sem:integer;
16:      dev:text;
17:      hash_table:array[0..hts_sub1] of w;
18:      CF,DEDUCED,STATUS,GOAL,SOURCE,MAIN,USER:w;
19:
20: procedure our_str(i:real;var ch:string80);
21: begin
22:      str(i,ch);while ch[1]=' ' do ch := copy(ch,2,length(ch))
23: end;
24:
25: function null(x:w): boolean;
26: begin
27:      null := x=nil
28: end; ( function null(x:w) )
29:
30: function car(x:w):w;
31: begin
32:      car := x^.a
33: end; ( function car(x:w):w )
34:
35: function cdr(x:w):w;
36: begin
37:      cdr := x^.b
38: end; ( function cdr(x:w):w )
39:
40: function caar(x:w):w;
41: begin
42:      caar := car(car(x))
43: end; ( function caar(x:w) )
44:
45: function cdar(x:w):w;
46: begin
47:      cdar := cdr(car(x))
48: end; ( function cdar(x:w) )
49:
50: procedure rplacd(x,y:w);
51: begin
52:      x^.b:=y
53: end; ( procedure rplacd(x,y:w) )
54:
55: function atom(x:w):boolean;
```

```
 56: begin
 57:     if null(x) then atom := true else atom := x^.rodzaj=jeden
 58: end; { function atom(x:w):boolean }
 59:
 60:
 61: function getpname(v:w): string80;
 62: begin
 63:     if null(v) then getpname := 'NIL' else getpname := v^.printname
 64: end; { function getpname(v:w) }
 65:
 66:
 67: function consp(x:w):boolean;
 68: begin
 69:     consp := x^.rodzaj=para
 70: end; { function consp(x:w):boolean }
 71:
 72: procedure writlist(x:w);
 73: begin
 74:  if x=nil then write('NIL ')
 75:  else if atom(x) then write(x^.printname)
 76:  else
 77:   begin
 78:     write('(');
 79:     while not atom(x) do
 80:       begin
 81:         writlist(car(x));
 82:         x := cdr(x);
 83:       end;
 84:     if not null(x) then write(' . ',x^.printname);
 85:     write(')')
 86:   end
 87: end; { procedure writlist(x:w) }
 88:
 89: procedure wrilist(tekst:string80;lista:w);
 90: begin
 91:   writeln;
 92:   write(tekst,char(32));
 93:   writlist(lista)
 94: end; { procedure wrilist(tekst:string80,lista:w) }
 95:
 96: function cons(x1,x2:w): w;
 97:  var y:w;
 98: begin
 99:   new(y);
100:   y^.rodzaj := para;
101:   y^.a := x1;
102:   y^.b := x2;
103:   cons := y
104: end; { function cons(x1,x2:w):w }
105:
106: function nconc(x1,x2:w):w;
107: begin
108:   nconc := x1;
109:   while not null(cdr(x1)) do x1 := cdr(x1);
110:   x1^.b := x2;
```

```
111: end; { function nconc(x1,x2:w)  }
112:
113: function nowyatom(symbol:string80):w;
114: var v:w;
115: begin
116:    new(v);
117:    v^.rodzaj := jeden;
118:    v^.printname := symbol;
119:    v^.property := NIL;
120:    nowyatom := v
121: end; { function nowyatom(symbol:string80):w }
122:
123:
124: function equal(x,y:w): boolean;
125: begin
126:      if atom(x) then
127:          if atom(y) then equal := x=y else equal := false
128:      else
129:          if atom(y) then equal := false
130:      else
131:          if equal(car(x),car(y)) then equal := equal(cdr(x),cdr(y))
132:      else
133:          equal := false
134: end; { function equal(x,y:w) }
135:
136: function assoc(x,y:w): w;
137: begin
138:  if null(y) then assoc := nil
139:  else if equal(x,caar(y)) then assoc := car(y)
140:  else assoc := assoc(x,cdr(y))
141: end; { funtion assoc(x,y:w) }
142:
143: function pop(var stack,x:w): boolean;
144: begin
145:   if null(stack) then pop := false
146:         else begin
147:                 pop := true;
148:                 x := car(stack);
149:                 stack := cdr(stack)
150:              end
151: end; { function pop(var stack,x:w):boolean }
152:
153: procedure push(var stack:w;x:w);
154: begin
155:   stack := cons(x,stack)
156: end; { procedure push(var stack,x:w) }
157:
158: procedure init_hash_table;
159: var i:integer;
160: begin
161:   for i := 0 to hts_subi do hash_table[i] := nil
162: end; { procedure init_hash_table }
163:
164: function hash(str:string80): integer;
165: var i,ioax,tmp:integer;
```

```
166: begin
167:   tmp := 0;
168:   iaax := length(st);
169:   for i := 1 to iaax do tmp:=tmp+ord(st[i]);
170:   hash:=tmp mod hash_table_size
171: end; { function hash(st:string80) }
172:
173: function intern(nowy:string80): w;
174: var x,znany:w;index:integer;
175: label et;
176: begin
177:      index := hash(nowy);
178:      x := hash_table[index];
179:      while pop(x,znany) do if getpname(znany)=nowy then goto et;
180:      znany := nowyatom(nowy);
181:      push(hash_table[index],znany);
182:  et: intern := znany
183:  end; { function intern(nowy:string80):w }
184:
185: { remove all occurences of a from all sublists of x }
186: function remove(a,x:w): w;
187: begin
188:  if atom(x) then remove := x
189:   else if equal(a,car(x)) then remove := remove(a,cdr(x))
190:   else  remove := cons(remove(a,car(x)),remove(a,cdr(x)))
191: end; { function remove(a,x:w) }
192:
193: function getprop(at,prop:w):w;{ funkcja do sprawdzania}
194: var temp:w;
195: begin
196:      temp:=assoc(prop,at^.property);
197:      if null(temp) then getprop:=nil else getprop := cdr(temp)
198: end; { function getprop(at,prop:w) }
199:
200: procedure putprop(at,prop,value:w);
201: label et;
202: var temp:w;
203: begin
204:      temp := at^.property;
205:      while not null(temp) do
206:        begin
207:          if car(temp)=prop then
208:            begin
209:              rplacd(car(temp),value);
210:              goto et
211:            end;
212:          temp := cdr(temp)
213:        end;
214:      at^.property := cons(cons(prop,value),at^.property);
215:      et:
216: end; { procedure putprop(var at:w;prop,value:w) }
217:
218: function checkstack(x,stack:w):boolean;
219: var y:w;
220: begin
```

```
221:   checkstack := false;
222:   while pop(stack,y) do if equal(x,y) then checkstack := true
223: end; (function checkstack(x:w;var stack:w))
224:
225: function append(x,y:w):w;
226: begin
227:     if null(x) then append := y
228:                else append := cons(car(x),append(cdr(x),y))
229: end; ( function append(x,y:w) )
230:
231: function lenlist(x:w): integer;
232: begin
233:     if null(x) then lenlist := 0 else lenlist := succ(lenlist(cdr(x)))
234: end;
235:
236: procedure initstack(var stack:w);
237: begin
238:   stack := nil
239: end; ( procedure initstack(var stack:w) )
240:
241: procedure sunput(var dev:text;var symbol:string80);
242: var ch:char;
243: begin
244:   symbol := '';
245:   repeat
246:     read(dev,ch);write(ch);
247:     if ((ord(ch)=13) or (ord(ch)=10)) then
248:     else symbol := concat(symbol,ch);
249:   until ch in ['(',')',' ',','];
250: end; ( procedure sunput(var dev:text;symbol:string80) )
251:
252: function dopisz(var dev:text):w;
253: begin
254:  sunput(dev,symbol);
255:   case symbol[1] of
256:             '(': dopisz := cons(dopisz(dev),dopisz(dev));
257:             ')': dopisz := nil;
258:             else dopisz := cons(intern(symbol),dopisz(dev));
259:   end ( case symbol )
260: end; ( function dopisz(dev):w) )
261:
262: function czytaj(var dev:text):w;
263: var x:w;
264: begin
265:   sunput(dev,symbol);
266:   if symbol='(' then x := dopisz(dev)
267:                 else x := intern(symbol);
268:   czytaj := reverse(intern(' '),x)
269: end; ( function czytaj(var dev:text):w )
270:
271: function czyt(var dev:text;tekst:string80):w;
272: begin
273:     writeln;
274:     write(tekst,' --> ');
275:     czyt := czytaj(dev)
```

```
276: end; { function czyt(tekst:string80;var dev:text) }
277:
278: function member(x,y:w):w;
279: begin
280:  if null(y) then member := nil
281:  else if  equal(x,car(y)) then member := y
282:  else member := member(x,cdr(y))
283: end; { funtion member(x,y:w) }
284:
285: procedure show;
286: var list,tmp,x:w;i:integer;
287: begin
288:      writeln;
289:      for i:=0 to hts_sub1 do
290:       begin
291:        list := hash_table[i];
292:        if not null(list) then
293:         repeat
294:           tmp:=car(list);
295:           if not null(tmp^.property)
296:             then
297:                begin
298:                  write(getpname(tmp):6);write('... ');writlist(tmp^.property);
299:                  writeln
300:                end;
301:           list:=cdr(list)
302:         until null(list)
303:       end;
304:      writeln
305: end;
306:
307: function countcf(list:w):real;
308: var poa,cfs:real;
309:     err:integer;
310:     x:w;
311:     ch:string80;
312: begin
313:      poa := 1.0;
314:      while pop(list,x) do
315:           begin
316:                val(getpname(getprop(x,CF)),cfs,err);
317:                if err<>0 then
318:                 writeln('attempt to apply arith. operator to string ',
319:                 getpname(getprop(x,CF)))
320:                else if cfs<poa then poa := cfs;
321:           end;
322:     countcf := poa
323: end; { function countcf(list:w) }
324:
325: function thres(list:w):boolean;
326: var cfs:real;
327:     err:integer;
328:     x:w;
329: begin
330: thres := true;
```

```
331: while pop(list,x) do
332: begin
333:  val(getpname(getprop(x,CF)),cfs,err);
334:   if err=0 then
335:      if cfs < threshold then thres := false
336: end
337: end; { function thres(list:w) }
338:
339: function allprop(list,prop:w):boolean;
340: var x:w;
341: begin
342:      allprop := true;
343:      while pop(list,x) do
344:      if null(getprop(x,prop)) then allprop := false
345: end; { function allprop(list:w) }
346:
347: function foundprop(list,prop,value:w):boolean;
348: var x:w;
349: begin
350:      foundprop := false;
351:      while pop(list,x) do
352:      if getprop(x,prop)=value then foundprop := true
353: end; { function foundprop(list:w) }
354:
355: procedure fire(rules:w);
356: var x,y:w;
357:     cfs:string80;
358: begin
359:      if ( allprop(caar(rules),CF) and thres(caar(rules))
360:           and not allprop(cdar(rules),CF) )
361:      then
362:       begin
363:        our_str(countcf(caar(rules)),cfs);
364:        writlist('FIRE !',car(rules));writeln('with cf...',cfs);
365:        y := cdar(rules);
366:        while pop(y,x) do
367:             begin
368:                  putprop(x,SOURCE,car(rules));
369:                  putprop(x,STATUS,DEDUCED);
370:                  putprop(x,CF,intern(cfs));
371:             end
372:       end;
373:      if not null(cdr(rules)) then fire(cdr(rules))
374: end;
375:
376: function findrule(rules:w):w;
377: begin
378:      if null(rules) then findrule := nil
379:                     else
380:          if ( thres(caar(rules)) and
381:            foundprop(cdar(rules),STATUS,GOAL)
382:              and not foundprop(caar(rules),STATUS,GOAL) )
383:             then
384:               findrule := car(rules)
385:             else
```

```
386:                findrule := findrule(cdr(rules))
387: end; { function findrule(rules:w) }
388:
389: procedure ask(rule,premise:w);
390: var ch:string[80];
391: begin
392:   if null(getprop(premise,CF)) then
393:     begin
394:       repeat
395:         writlist('checking premise <CF> : ',premise);
396:         write('... ');
397:         readln(ch);
398:         if ch='?' then show;
399:       until ch<>'?';
400:       if ch='unk' then
401:                     begin
402:                       putprop(premise,SOURCE,rule);
403:                       putprop(premise,STATUS,GOAL)
404:                     end
405:                   else
406:                     begin
407:                       putprop(premise,SOURCE,USER);
408:                       putprop(premise,CF,intern(ch))
409:                     end
410:     end
411: end;
412:
413: procedure search(var rules:w);
414: var premise,rule,left:w;
415: begin
416:   fire(rules);
417:      rule := findrule(rules);writlist('found rule ...',rule);
418:  if not null(rule) then
419:     begin
420:      left := car(rule);
421:      while pop(left,premise) do ask(rule,premise);
422:      search(rules)
423:    end
424: end; { procedure search(var rules:w) }
425:
426: function TextFileExists(var filvar:text;NameOfFile:string80):Boolean;
427: begin
428: assign(filvar,NameOfFile);
429: {$I-}reset(filvar);{$I+}
430: TextFileExists := (Ioresult=0);
431: end; {TextFileExists}
432:
433: begin
434:      init_hash_table;
435:      DEDUCED := intern('DEDUCED');
436:      STATUS := intern('STATUS:');
437:      SOURCE := intern('SOURCE:');
438:      MAIN := intern('MAIN');
439:      GOAL := intern('GOAL');
440:      USER:=intern('USER');
```

```
441:     CF:=intern('CF');
442:   textbackground(9);textcolor(yellow);
443:   NameOfFile := 'elements';
444:   if not textfileexists(dev,NameOfFile) then halt;
445:   clrscr;
446:   write(char(240),' ');
447:   rules := czytaj(dev);
448:   write(char(240),' ');
449:   maingoal := czytaj(kbd);
450:   putprop(maingoal,SOURCE,MAIN);
451:   putprop(maingoal,STATUS,GOAL);
452:   search(rules);
453:   writeln;
454:   show;
455: end.
```

```
 1: var
 2:    sztyki:array[1..300,1..30] of real;
 3:    obcir:array[1..300] of real;
 4:    przem:array[1..2,1..4] of real;
 5:    wzsp,wzso:array[1..20] of real;
 6:    nzsp,nzso:array[1..20] of integer;
 7:    ibool:array[1..100,1..4] of integer;
 8:    wsx,wsy:array[1..100] of real;
 9:    pwsx,pwsy:array[1..10] of real;
10:    szte:array[1..8,1..8] of real;
11:    qwww:array[1..4,1..2] of real;
12:    c:array[1..3,1..3] of real;
13:    aj,aoj:array[1..2,1..2] of real;
14:    pfil,pfig:array[1..2,1..4] of real;
15:    gaus,beta:array[1..4] of real;
16:    tytul,nane:string[80];
17:    infile:text;
18:    ityp,irod,igen,ndruk,lwee,lpwx,lpwy,lwe,lei,lzsp,lzso,lssu,lsse:integer;
19:    iszer:integer;
20:    i,ii,j,jj,k,l,ll,m,n,ne,ni,nj,ns,nn:integer;
21:    wx,wy,wz,amod,wpois,emiz:real;
22: const
23:    welw:array[1..4,1..2] of real
24:         =((-1.0,-1.0),(1.0,-1.0),(1.0,1.0),(-1.0,1.0));
25: type
26:    string80=string[80];
27:
28: procedure sztyki;
29: var
30: aksi,aeta,aksio,aetao,xlok,ylok,stal:real;
31: i,j,ii,jj,k,ni,nj:integer;
32: fi:array[1..4] of real;
33: r:array[1..4,1..4] of real;
34: rxy,rx,ry:array[1..4,1..4] of real;
35: const
36: gauss:array[1..2] of real=(-0.57735027,0.57735027);
37: waga:array[1..2] of real=(1.0,1.0);
38: begin
39:     for ii:=1 to 4 do
40:     for jj:=1 to 4 do
41:         begin
42:         (inicjalizacja tablic)
43:             r[i,j]:=0.0;
44:             rx[i,j]:=0.0;
45:             ry[i,j]:=0.0;
46:             rxy[i,j]:=0.0;
47:         end;
48:     for ni:=1 to 2 do
49:     for nj:=1 to 2 do
50:         begin
51:             aksi:=gauss[ni];
52:             aeta:=gauss[nj];
53:                 for i:=1 to 4 do
54:                   begin
55:                     xlok:=welw[i,1];
```

```
56:                    ylok:=welw[i,2];
57:                    aksio:=1.0+aksi*xlok;
58:                    aetao:=1.0+aeta*ylok;
59:                    fi[i]:=0.25*aksio*aetao;
60:                    pfil[1,i]:=0.25*xlok*aetao;
61:                    pfil[2,i]:=0.25*ylok*aksio;
62:                  end;
63:              for i:=1 to 2 do
64:              for j:=1 to 2 do
65:                 begin
66:                     aj[i,j]:=0.0;
67:                         for k:=1 to 4 do
68:                            aj[i,j]:=aj[i,j]+pfil[i,k]*gwwe[k,j];
69:                 end;
70:              aman:=aj[1,1]*aj[2,2]-aj[1,2]*aj[2,1];
71:              aoj[1,1]:=aj[2,2]/aman;
72:              aoj[2,2]:=aj[1,1]/aman;
73:              aoj[1,2]:=-aj[1,2]/aman;
74:              aoj[2,1]:=-aj[2,1]/aman;
75:              for i:=1 to 2 do
76:              for j:=1 to 4 do
77:                 begin
78:                     pfig[i,j]:=0.0;
79:                           for k:=1 to 2 do
80:                              pfig[i,j]:=pfig[i,j]+aoj[i,k]*pfil[k,j];
81:                 end;
82:              stal:=aman*waga[ni]*waga[nj];
83:              for i:=1 to 4 do
84:              for j:=1 to 4 do
85:                 begin
86:                     r[i,j]:=r[i,j]+stal*fi[i]*fi[j];
87:                     rx[i,j]:=rx[i,j]+stal*pfig[1,i]*pfig[1,j];
88:                     ry[i,j]:=ry[i,j]+stal*pfig[2,i]*pfig[2,j];
89:                     rxy[i,j]:=rxy[i,j]+stal*pfig[1,i]*pfig[2,j];
90:                 end;
91:          end;
92:          ii:=1;
93:          for i:=1 to 4 do
94:             begin
95:               jj:=1;
96:                 for j:=1 to 4 do
97:                   begin
98:                     szte[ii,jj]:=c[1,1]*rx[i,j]+c[3,3]*ry[i,j];
99:                     szte[ii+1,jj+1]:=c[3,3]*rx[i,j]+c[2,2]*ry[i,j];
100:                    szte[ii,jj+1]:=c[1,2]*rxy[i,j]+c[3,3]*rxy[i,j];
101:                    szte[ii+1,jj]:=c[1,2]*rxy[i,j]+c[3,3]*rxy[i,j];
102:                    jj:=2*j+1;
103:                  end;
104:               ii:=2*i+1;
105:            end;
106: end;
107:
108: procedure sztykt;
109: var
110: b,s:array[1..3,1..6] of real;
```

```
111: bt:array[1..6,1..3] of real;
112: wsx,wsy:array[1..3] of real;
113: var i,j,k:integer;
114: begin
115:     for i:= 1 to 3 do
116:         begin
117:             wsx[i]:=gwwe[i,1];
118:             wsy[i]:=gwwe[i,2];
119:                 for j:=1 to 6 do b[i,j]:=0.0;
120:         end;
121:     for i:=1 to 3 do
122:         begin
123:             j:=i mod 3 + 1;
124:             k:=j mod 3 + 1;
125:             beta[i]:=wsy[j]-wsy[k];
126:             gama[i]:=wsx[k]-wsx[j];
127:         end;
128:     amian:=wsx[1]*(wsy[2]-wsy[3])+
129:              wsx[2]*(wsy[3]-wsy[1])+wsx[3]*(wsy[1]-wsy[2]);
130:     for i:=1 to 3 do
131:         begin
132:             j:=2*i-1;
133:             b[i,j]:=beta[i]/amian;
134:             b[3,j]:=gama[i]/amian;
135:             b[2,j+1]:=gama[i]/amian;
136:             b[3,j+1]:=beta[i]/amian;
137:         end;
138:     for i:=1 to 3 do
139:     for j:=1 to 6 do
140:         begin
141:             bt[j,i]:=b[i,j];
142:             s[i,j]:=0.0;
143:             for k:=1 to 3 do s[i,j]:=s[i,j]+c[i,k]*b[k,j];
144:         end;
145:     for i:=1 to 6 do
146:     for j:=1 to 6 do
147:         begin
148:             szte[i,j]:=0.0;
149:             for k:=1 to 3 do szte[i,j]:=szte[i,j]+0.5*amian*bt[i,k]*s[k,j];
150:         end;
151: end;
152:
153: procedure warbrz;
154: var iszer1:integer;
155: begin
156:     for n:=1 to lzsp do
157:         begin
158:             jj:=nzsp[n];
159:             wz:=wzsp[n];
160:             iszer1:=iszerp-1;
161:             i:=jj-iszerp;
162:                 for ii:=1 to iszer1 do
163:                   begin
164:                     i:=i+1;
165:                     if i >= 1 then
```

```
166:                          begin
167:                            jj:=jj-i+1;
168:                            obcir[i]:=obcir[i]-sztyk[i,j]*wz;
169:                            sztyk[i,j]:=0.0;
170:                          end;
171:                      end;
172:              sztyk[jj,1]:=1.0;
173:             obcir[jj]:=wz;
174:              i:=jj;
175:                 for ii:=2 to iszerp do
176:                    begin
177:                       i:=i+1;
178:                       if i <= lssu then
179:                          begin
180:                            obcir[i]:=obcir[i]-sztyk[jj,ii]*wz;
181:                            sztyk[jj,ii]:=0.0;
182:                          end;
183:                    end;
184:         end;
185: end;
186:
187: procedure rozw;
188: var
189:    lssu1,ipivot,nwie,nkol,icol,jcol:integer;
190:    wspol:real;
191: begin
192:     lssu1:=lssu-1;
193:     for nn:=1 to lssu1 do
194:       begin
195:         ipivot:=nn+1;
196:         ll:=nn+iszerp-1;
197:         if ll > lssu then ll:=lssu;
198:           for nwie:=ipivot to ll do
199:             begin
200:               nkol:=nwie-nn+1;
201:               wspol:=sztyk[nn,nkol]/sztyk[nn,1];
202:                 for nkol:=nwie to ll do
203:                   begin
204:                     icol:=nkol-nwie+1;
205:                     jcol:=nkol-nn+1;
206:                     sztyk[nwie,icol]:=sztyk[nwie,icol]-wspol*sztyk[nn,jcol];
207:                   end;
208:               obcir[nwie]:=obcir[nwie]-wspol*obcir[nn];
209:             end;
210:       end;
211:     for ii:=2 to lssu do
212:     begin
213:     nn:=lssu-ii+2;
214:     obcir[nn]:=obcir[nn]/sztyk[nn,1];
215:     ll:=nn-iszerp+1;
216:     if ll < 1 then ll:=1;
217:     ipivot:=nn-1;
218:     for jj:=ll to ipivot do
219:         begin
220:             nwie:=ipivot-jj+ll;
```

```
221:              nkol:=nn-nwie+1;
222:              wspol:=sztyk[nwie,nkol];
223:              obcir[nwie]:=obcir[nwie]-wspol*obcir[nn];
224:            end;
225:        end;
226:        obcir[i]:=obcir[i]/sztyk[i,i];
227: end;
228:
229: procedure napr;
230: var
231:    wsx,wsy:array[1..3] of real;
232:    uxx,uyy,uxy,uyx,xloc,yloc,sigxx,sigyy,sigxy:real;
233: begin
234:        uxx:=0.0;
235:        uxy:=0.0;
236:        uyx:=0.0;
237:        uyy:=0.0;
238:        if lwee <> 4 then
239:          begin
240:            for i:=1 to lwee do
241:              begin
242:                  wsx[i]:=gwwe[i,1];
243:                  wsy[i]:=gwwe[i,2];
244:              end;
245:            for i:=1 to lwee do
246:              begin
247:                  j:=i+1;
248:                  if j > lwee then j:=j-lwee;
249:                  k:=j+1;
250:                  if k > lwee then k:=k-lwee;
251:                  beta[i]:=wsy[j]-wsy[k];
252:                  gama[i]:=wsx[k]-wsx[j];
253:              end;
254:            amian:=wsx[1]*(wsy[2]-wsy[3])+
255:                   wsx[2]*(wsy[3]-wsy[1])+wsx[3]*(wsy[1]-wsy[2]);
256:          end
257:        else
258:          begin
259:              for i:=1 to 4 do
260:                begin
261:                    xloc:=wnlw[i,1];
262:                    yloc:=welw[i,2];
263:                    pfil[1,i]:=0.25*xloc;
264:                    pfil[2,i]:=0.25*yloc;
265:                end;
266:              for i:=1 to 2 do
267:              for j:=1 to 2 do
268:                begin
269:                    aj[i,j]:=0.0;
270:                        for k:=1 to 4 do
271:                            aj[i,j]:=aj[i,j]+pfil[i,k]*gwwe[k,j];
272:                end;
273:              amian:=aj[1,1]*aj[2,2]-aj[1,2]*aj[2,1];
274:              aoj[1,1]:=aj[2,2]/amian;
275:              aoj[2,2]:=aj[1,1]/amian;
```

```
276:              aoj[1,2]:=-aj[1,2]/amian;
277:              aoj[2,1]:=-aj[2,1]/amian;
278:              for i:=1 to 2 do
279:              for j:=1 to 4 do
280:                 begin
281:                    pfig[i,j]:=0.0;
282:                       for k:=1 to 2 do
283:                          pfig[i,j]:=pfig[i,j]+aoj[i,k]*pfil[k,j];
284:                 end;
285:              for i:=1 to 4 do
286:                 begin
287:                    beta[i]:=pfig[1,i]*amian;
288:                    gama[i]:=pfig[2,i]*amian;
289:                 end;
290:           end;
291:        for i:=1 to lwe do
292:           begin
293:              uxx:=uxx+beta[i]*przem[1,i]/amian;
294:              uyy:=uyy+gama[i]*przem[2,i]/amian;
295:              uxy:=uxy+gama[i]*przem[1,i]/amian;
296:              uyx:=uyx+beta[i]*przem[2,i]/amian;
297:           end;
298:        sigxx:=c[1,1]*uxx+c[1,2]*uyy;
299:        sigyy:=c[1,2]*uxx+c[2,2]*uyy;
300:        sigxy:=c[3,3]*(uxy+uyx);
301:        writeln(ne:10,sigxx:12:5,sigyy:12:5,sigxy:12:5);
302: end;
303:
304: function TextFileExist(var filvar:text;NameOfFile:string80):Boolean;
305: begin
306:  assign(filvar,NameOfFile);
307:  {$I-}reset(filvar);{$I+}
308:  TextFileExists:=(ioresult=0);
309: end; {TextFileExists}
310:
311: procedure gensia(var lex,ley,lwe,lel:integer);
312: var ipa,lwx,lwy:integer;
313: begin
314:      ipa:=3;
315:      lwx:=succ(lex);
316:      lwy:=succ(ley);
317:      lel:=2*lex*ley;
318:      lwe:=lwx*lwy;
319:      ibool[1,1]:=1;
320:      ibool[1,2]:=2;
321:      ibool[1,3]:=lwx+2;
322:      ibool[2,1]:=1;
323:      ibool[2,2]:=lwx+2;
324:      ibool[2,3]:=lwx+1;
325:      for ii:=1 to ley do
326:      begin
327:      l:=ii*2*lex;
328:      m:=(ii-1)*2*lex;
329:      n:=ipa;
330:      if lex<>1 then
```

```
331:     while n <= l do
332:       begin
333:         for i:=1 to 3 do
334:             begin
335:             ibool[n,i]:=ibool[n-2,i]+1;
336:             ibool[n+1,i]:=ibool[n-1,i]+1;
337:           end;
338:       n:=n+2;
339:      end;
340:     if ley<>1 then
341:     for i:=1 to 3 do
342:     begin
343:     ibool[l+1,i]:=ibool[a+1,i]+lwx;
344:     ibool[l+2,i]:=ibool[a+2,i]+lwx;
345:     end;
346:     ipa:=f+3;
347:     end;
348:     l:=0;
349:     wy:=0.0;
350:     for j:=1 to lwy do
351:     begin
352:     wx:=0.0;
353:     for i:=1 to lwx do
354:     begin
355:     l:=l+1;
356:     wsx[l]:=wx;
357:     wsy[l]:=wy;
358:     wx:=wx+pwsx[i];
359:     end;
360:     wy:=wy+pwsy[j];
361:     end;
362: end;
363:
364: begin (program mes2s)
365: for i:=1 to 100 do
366: for j:=1 to 4 do
367: ibool[i,j]:=0;
368:     write('nase of input file ... ');
369:     readln(name);
370:     if TextFileExists(infile,name) then write('opening file ',name,'...');
371:     writeln;
372:     readln(infile,tytul);
373:     writeln(tytul);
374:     readln(infile,ityp,irod,igen,ndruk);
375:     if ((ityp=1) and (igen=1)) then
376:       begin
377:       writeln('w obecnej wersji programu nie mozna generowac czworokatow');
378:       halt
379:       end;
380:     lwee:=ityp+3;
381:     if irod=1 then writeln('theory of elasticity - plain strain');
382:     if irod=0 then writeln('theory of elasticity - plain stress');
383:     if igen=1 then
384:       begin
385:           readln(infile,lpwx,lpwy);
```

```
386:            for i:=1 to lpwx do readln(infile,pwsx[i]);
387:                    for i:=1 to lpwx do writeln('x',pwsx[i]);
388:            for i:=1 to lpwy do readln(infile,pwsy[i]);
389:                    for i:=1 to lpwy do writeln('y',pwsy[i]);
390:            gensia(lpwx,lpwy,lwe,lel);
391:        end
392:            else
393:        begin
394:            readln(infile,lel,lwe);
395:            for n:=1 to lel do
396:              begin
397:                for i:=1 to lwee do read(infile,ibool[n,i]);
398:                readln(infile)
399:              end;
400:            for i:=1 to lwe do
401:            begin
402:            readln(infile,wsx[i],wsy[i]);
403:            end;
404:        end;
405:    readln(infile,amod,wpois);
406:    readln(infile,lzsp);
407:            for i:=1 to lzsp do readln(infile,nzsp[i]);
408:            for i:=1 to lzsp do readln(infile,wzsp[i]);
409:    readln(infile,lzso);
410:            for i:=1 to lzso do readln(infile,nzso[i]);
411:            for i:=1 to lzso do readln(infile,wzso[i]);
412: {koniec wprowadzania danych}
413:        lssu:=lwe*2;
414:        lsse:=lwee*2;
415:        if ityp=0 then writeln('element trojkatny 3w f.ksz. liniowa');
416:        if ityp=1 then writeln('element czworokatny 4w izoparametryczny');
417:        writeln('modul younga: ',amod:10:3,' liczba poissona: ',wpois:10:3);
418:        writeln('elementow: ',lel,' wezlow: ',lwe,' rownan: ',lssu);
419:        writeln('globalne wspolrzedne wezlowe');
420:        for i:=1 to lwe do writeln(i:17,wsx[i]:10:3,wsy[i]:10:3);
421:        writeln('globalne numery wezlow w elementach:');
422:        for i:=1 to lel do
423:          writeln(i:17,ibool[i,1]:18,ibool[i,2]:8,ibool[i,3]:8,ibool[i,4]:8);
424:        writeln('liczba zadanych stopni swobody: ',lzsp);
425:        for i:=1 to lzsp do writeln(nzsp[i]:17,wzsp[i]:10:3);
426:        writeln('liczba zadanych skladowych obciazenia: ',lzso);
427:        for i:=1 to lzso do writeln(nzso[i]:17,wzso[i]:10:3);
428: {szerokosc polpasma}
429:        iszerp:=0;
430:        for n:=1 to lel do
431:        for i:=1 to lwee do
432:        for j:=1 to lwee do
433:          begin
434:              m:=(abs(ibool[n,i]-ibool[n,j])+1)*2;
435:              if iszerp<m then iszerp:=m
436:          end;
437:        writeln('szerokosc polpasma: ',iszerp);
438: {obliczanie macierzy konstytutywnych
439:
440: psn}
```

```
441:        asian:=1.0-wpois*wpois;
442:        c[1,1]:=amod/asian;
443:        c[1,2]:=c[1,1]*wpois;
444:        c[2,2]:=c[1,1];
445:        c[3,3]:=amod/(2.*(1+wpois));
446:        if irod<>0 then {pso}
447:          begin
448:            c[1,1]:=amod*(1.-wpois)/((1.+wpois)*(1.-2.*wpois));
449:            c[1,2]:=c[1,1]*wpois/(1.-wpois);
450:            c[2,2]:=c[1,1];
451:          end;
452:        c[1,3]:=0.0;
453:        c[2,3]:=0.0;
454:        c[2,1]:=c[1,2];
455:        c[3,1]:=c[1,3];
456:        c[3,2]:=c[2,3];
457: {inicjalizacja macierzy sztywnosci i obciazenia}
458:        for i:=1 to lnsu do
459:          begin
460:            obcir[i]:=0;
461:            for j:=1 to iszerp do sztyk[i,j]:=0.0
462:          end;
463: {obliczanie elementowych macierzy sztywnosci}
464:        for n:=1 to lel do
465:          begin
466:            for i:=1 to lwee do
467:              begin
468:                m:=ibool[n,i];
469:                gwee[i,1]:=wsx[m];
470:                gwee[i,2]:=wsy[m]
471:              end;
472:            if ityp=1 then sztyki;
473:            if ityp=0 then sztykt;
474:            if ndruk<>0 then
475:              begin
476:                writeln('macierz sztywnosci elementu: ',n:3);
477:                for i:=1 to lsse do
478:                  begin
479:                    for j:=1 to lsse do write(sztyk[i,j]:9:2,' ');
480:                    writeln
481:                  end
482:              end;
483: {tworzenie globalnej macierzy sztywnosci}
484:            for i:=1 to lwee do
485:              begin
486:                k:=(ibool[n,i]-1)*2;
487:                for ii:=1 to 2 do
488:                  begin
489:                    k:=k+1;
490:                    l:=(i-1)*2+ii;
491:                    for j:=1 to lwee do
492:                      begin
493:                        mm:=(ibool[n,j]-1)*2;
494:                          for jj:=1 to 2 do
495:                            begin
```

```
496:                                  a1:=(j-1)*2+jj;
497:                                  ns1:=nn+jj+1-k;
498:                              if ns>0 then sztyk[k,ns]:=sztyk[k,ns]+szte[i,a];
499:                                end;
500:                            end;
501:                          end;
502:                      end;
503:              end;
504: { utworzenie wektora obciazenia oraz uwzglednienie warunkow brzegowych)
505:          for i:=1 to lzso do
506:            begin
507:              ii:=nzso[i];
508:              obcir[ii]:=wzso[i]+obcir[ii];
509:            end;
510:          if ndruk > 1 then
511: { wydruk macierzy sztywnosci przed wykorzystaniem kinematycznych w.b.)
512:            begin
513:              writeln('globalna macierz sztywnosci przed wyk. war. brz.');
514:              for i:=1 to lssu do
515:                begin
516:                  for j:=1 to iszerp do write(sztyk[i,j]:9:2,' ');
517:                  writeln;
518:                end;
519: { wydruk wektora obciazenia )
520:              writeln('wektor obciazenia');
521:              for i:=1 to lwe do writeln(i,obcir[2*i-1]:10:3,obcir[2*i]:10:3);
522:            end;
523:          if lzsp<>0 then
524:            begin
525:              warbrz;
526:              if ndruk > 2 then
527:                begin
528:                  writeln('globalna macierz sztywn. po wyk. kin. war. brz.');
529:                  for i:=1 to lssu do
530:                    begin
531:                      for j:=1 to iszerp do write(sztyk[i,j]:10:3);
532:                      writeln;
533:                    end;
534:                end;
535:            end;
536:          rozw;
537: { wektor obcir zawiera obecnie rozwiazanie
538:
539:  wydruk rozwiazania - przemieszczenia w wezlach )
540:          writeln('rozwiazanie  nr wezla          ux                uy');
541:          for i:=1 to lwe do writeln(i:17,obcir[2*i-1]:18:4,obcir[2*i]:18:4);
542:          writeln('element          sigmax          sigmay');
543:          for ne:=1 to lel do
544:          begin
545:            for i:=1 to lwee do
546:            begin
547:              a:=ibool[ne,i];
548:              l:=a*2-1;
549:              przem[1,i]:=obcir[l];
550:              przem[2,i]:=obcir[l+1];
```

```
551:            gwwe[i,1]:=wsx[n];
552:            gwwe[i,2]:=wsy[n];
553:          end;
554: { obliczanie wart. naprezen i ich drukowanie }
555:          napr;
556:        end;
557: end.
558: }
```

```
przyklad z rozdzialu 5.4, zbior 2 (elementy trojkatne), L = 4.
    0   0   0   0
   32  27
    1   4   5
    5   2   1
    2   5   3
    5   6   3
    4   7   5
    5   7   8
    5   8   9
    5   9   6
    7  10  11
    7  11   8
    8  11  12
   11  12   9
   10  13  11
   11  13  14
   11  14  15
   11  15  12
   13  16  17
   13  17  14
   14  17  15
   15  17  18
   16  19  17
   17  19  20
   17  20  21
   17  21  18
   19  22  23
   19  23  20
   20  23  21
   23  24  21
   22  25  23
   23  25  26
   23  26  27
   23  27  24
  0.0     0.0
  0.0   100.0
  0.0   200.0
100.0     0.0
100.0   100.0
100.0   200.0
200.0     0.0
200.0   100.0
200.0   200.0
300.0     0.0
300.0   100.0
300.0   200.0
400.0     0.0
400.0   100.0
400.0   200.0
500.0     0.0
500.0   100.0
500.0   200.0
600.0     0.0
600.0   100.0
```

```
56:  600.0       200.0
57:  700.0       0.0
58:  700.0       100.0
59:  700.0       200.0
60:  800.0       0.0
61:  800.0       100.0
62:  800.0       200.0
63: 2100000.0    0.3
64:     6
65:       1
66:       2
67:       3
68:       4
69:       5
70:       6
71:       0.0
72:       0.0
73:       0.0
74:       0.0
75:       0.0
76:       0.0
77:     2
78:       50
79:       54
80:       -100.0
81:       -100.0
```

przyklad z rozdzialu 5.4, zbior 2 (elementy trojkatne), L = 4.
theory of elasticity - plain stress
element trojkatny 3w f.ksz. liniowe
modul younga: 2100000.000 liczba poissona: 0.300
elementow: 32 wezlow: 27 rownan: 54
globalne wspolrzedne wezlow:

	1	0.000	0.000
	2	0.000	100.000
	3	0.000	200.000
	4	100.000	0.000
	5	100.000	100.000
	6	100.000	200.000
	7	200.000	0.000
	8	200.000	100.000
	9	200.000	200.000
	10	300.000	0.000
	11	300.000	100.000
	12	300.000	200.000
	13	400.000	0.000
	14	400.000	100.000
	15	400.000	200.000
	16	500.000	0.000
	17	500.000	100.000
	18	500.000	200.000
	19	600.000	0.000
	20	600.000	100.000
	21	600.000	200.000
	22	700.000	0.000
	23	700.000	100.000
	24	700.000	200.000
	25	800.000	0.000
	26	800.000	100.000
	27	800.000	200.000

globalne numery wezlow w elementach:

	1	1	4	5	0
	2	5	2	1	0
	3	2	5	3	0
	4	5	6	3	0
	5	4	7	5	0
	6	5	7	8	0
	7	5	8	9	0
	8	5	9	6	0
	9	7	10	11	0
	10	7	11	8	0
	11	8	11	12	0
	12	11	12	9	0
	13	10	13	11	0
	14	11	13	14	0
	15	11	14	15	0
	16	11	15	12	0
	17	13	16	17	0
	18	13	17	14	0
	19	14	17	15	0
	20	15	17	18	0
	21	16	19	17	0

```
                22          17    19    20     0
                23          17    20    21     0
                24          17    21    18     0
                25          19    22    23     0
                26          19    23    20     0
                27          20    23    21     0
                28          23    24    21     0
                29          22    25    23     0
                30          23    25    26     0
                31          23    26    27     0
                32          23    27    24     0
liczba zadanych stopni swobody: 6
                 1    0.000
                 2    0.000
                 3    0.000
                 4    0.000
                 5    0.000
                 6    0.000
liczba zadanych obciazenia: 2
                50  -100.000
                54  -100.000
szerokosc polpasma: 10
rozwiazanie  nr wezla          ux              uy
                 1          0.0000          0.0000
                 2          0.0000          0.0000
                 3          0.0000          0.0000
                 4         -0.0007         -0.0006
                 5          0.0000         -0.0003
                 6          0.0006         -0.0006
                 7         -0.0013         -0.0015
                 8          0.0000         -0.0015
                 9          0.0012         -0.0015
                10         -0.0017         -0.0033
                11          0.0000         -0.0032
                12          0.0017         -0.0033
                13         -0.0021         -0.0053
                14          0.0000         -0.0053
                15          0.0021         -0.0053
                16         -0.0024         -0.0078
                17         -0.0000         -0.0077
                18          0.0024         -0.0078
                19         -0.0027         -0.0104
                20         -0.0000         -0.0104
                21          0.0027         -0.0104
                22         -0.0028         -0.0132
                23         -0.0000         -0.0132
                24          0.0028         -0.0132
                25         -0.0028         -0.0162
                26         -0.0000         -0.0161
                27          0.0028         -0.0162
element          sigmax          sigmay
      1   -13.56395     1.21895     0.75538
      2     0.20342     0.06103    -2.63947
      3     0.20342     0.06103    -2.63947
      4    13.15710    -1.14422     0.52357
```

111:	5	-12.23947	1.61629	-2.07986
112:	6	0.11914	-0.68455	0.62801
113:	7	0.73651	1.37335	-0.25130
114:	8	11.38382	-1.67620	-2.29685
115:	9	-9.96134	0.17209	-0.57785
116:	10	-0.32385	-0.81745	-2.90614
117:	11	-0.84733	-2.56240	0.40750
118:	12	11.13252	1.03156	-0.92350
119:	13	-7.81863	0.81491	-1.56485
120:	14	-0.50956	-0.68308	-0.00033
121:	15	-0.03463	0.90003	-0.18113
122:	16	8.36283	0.20065	-2.25369
123:	17	-6.23487	0.33539	-0.25293
124:	18	-0.32876	-0.62834	-1.58344
125:	19	0.14617	0.95427	-1.76424
126:	20	6.41746	-0.24762	-0.39940
127:	21	-4.55394	0.83967	-1.42800
128:	22	-0.11875	-0.36748	-0.55891
129:	23	0.10383	0.37446	-0.56388
130:	24	4.56885	-0.80220	-1.44921
131:	25	-2.69417	-0.03766	-0.69433
132:	26	-0.11378	-0.36599	-1.30085
133:	27	0.10880	0.37595	-1.30582
134:	28	2.69915	0.04026	-0.69880
135:	29	-0.81742	0.52536	-1.18223
136:	30	0.24515	0.81777	-0.81742
137:	31	-0.24536	-0.81727	-0.81763
138:	32	0.81763	-0.52419	-1.18273

BIBLIOGRAFIA

1 S. ALAGIC, M. ARBIB "Projektowanie programów poprawnych i dobrze zbudowanych", WNT Warszawa, 1984

2 B. O. ALMROTH, P. STERN, F. A. BORGAN "Future Trends in Nonlinear Structural Analysis" Comp. Struct. vol10(1979) pp369-374

3 B. AMBROSIO "Building expert systems with M1" BYTE vol10(1985)

4 C. M. ANDERSEN "Evaluation of integrals for a ten-node isoparametric tetrahedral finite element" Comput. Mech. Applic. (1978)

5 C. M. ANDERSEN, A. K. NOOR "A computrized symolic integration technique for development of triangular and quadrilateral composite shallow-shell finite element" NASA TND-8067 (1975)

6 C. M. ANDERSEN, A. K. NOOR "Free vibration of laminated composite elliptic plates" Advances in Engng Science vol 2 CP-2001 pp425-438 NASA (1976)

7 C. M. ANDERSEN, A. K. NOOR "Symbolic Manipulation Techniques for Vibration Analysis of Laminated Elliptic Plates" Proc. 1977 MACSYMA Users' Conf. pp161-176 NASA CP-2012 (1977)

8 C. M. ANDERSEN, A. K. NOOR "Use of group-theoretic methods in the development of nonlinear shell finite elements" Proc. Conf. Symmetry, Similarity Group Theoretic Meth. Mech. University of Calgary, Canada (1974)

9 C. M. ANDERSEN, J. T. BOWEN "A computer program for anisotropic shallow shell finite elements using symbolic integration" NASA TMX-3325 (1976)

10 "Artificial Intelligence and Pattern Recognition in Computer Aided Design", Latombe, ed. IFIP, North-Holland Publishing Company, (1978)

11 K. J. BATHE "ADINA A Finite Element Program for Dynamic Incremental Nonlinear Analysis", MIT Report 82448-1, Cambridge, Massachussets (1977)

12 K. J. BATHE, A. P. CIMENTO "Some Practical Procedures for the Solution of

Nonlinear Finite Element Equations" Comp. Mech. Appl. Mech. Engng, vol22(1980), pp55-85

13 J. BENNETT, L. CREARY, R. ENGLEMORE, R. MELOSH "SACON:a knowledge - based consultant for structural analysis" Technical report STAN-CS-78- 699 Stanford University, Sept. 1978

14 J. BENNETT, R. ENGLEMORE, R. MELOSH "SACON:a knowledge-based consultant for structural analysis" Proceedings Sixth IJCAI, 47-49, August 20-23, 1979

15 P. BERGAN et al. , "Solution Techniques for Nonlinear Finite Element Problems", I. J. N. M. E. vol. 12(1978), pp. 1677-1696

16 P. BREITKOPF, M. KLEIBER "Expert system techniques in finite eement structural analysis" w "Microcomputers in Engineering" B. A. Schrefler, R. W. Lewis (eds) Pineridge Press, Swansea (1986)

17 P. BREITKOPF, M. KLEIBER "Knowledge Engineering Enhancement of Finite Element Analysis", Communications in Applied Numerical Methods vol. 3 (1987)

18 P. BREITKOPF, M. KLEIBER "Porównanie kompilatorów PASCALa i FORTRANu na przykładzie dydaktycznego programu MES", mat. konf. "Metody Komputerowe w Mechanice Konstrukcji" Jadwisin, maj 1987

19 P. BREITKOPF, M. KLEIBER "AI Techniques in Structural Mechanics", mat. konf. "Metody Komputerowe w Mechanice Konstrukcji" Jadwisin, maj 1987

20 P. BREITKOPF, M. KLEIBER "Finite Element Knowledge Acquisition for Solid Mechanics Applications" mat. konf. Artificial Intelligence in Engineering", 4-7 sierpnia 1987, Cambridge, MA, U. S. A.

21 P. BREITKOPF, M. KLEIBER "Nonnumerical Computational Techniques in Structural Mechanics", Uspiechi Mechaniki praca złożona do druku

22 P. BREITKOPF, M. KLEIBER "Self Learning Finite Element Program", mat. konf. First World Congress on Computational Mechanics, Austin, wrzesień 1986

23 "Building Expert Systems" F. Hayes-Roth, D. A. Waterman, D. B. Lenat (eds.) Addison – Wesley, 1983

24 M. M. CECCHI, G. LAMI "Automatic Generation of Stiffness Matrices for Finite Element Analysis" IJNME 11, pp396-400 (1977)

25 J. COHEN "Symbolic and numeric computer analysis of the combined local and overall uckling of rectangular thin-walled columns" Comput. Meth. Appl. Mech. Engng. 7 pp17-38 (1976)

26 "Coupling Symbolic and Numerical Computing in Expert Systems" J. S. Kowalik (editor) Elsevier Science Publishers B. V. (North Holland), 1986

27 N. CZEPURNIY, "Evaluation of high-order polynomial triangular finite elements using FORMAC" Proc. 2nd Symp. Symolic Algebraic Manipulation pp. 365-371 (1971)

28 R. DAMRATH, P. J. PAHL "Data and Storage Structures for a System of Finite Element Programs" Formulations and Algorithms in Finite Element Analysis MIT Press Cambridge Mass. 1977 pp140-162

29 P. O DUDA i inni "A Computer-Based Consultant for Mineral Exploration" Final Report SRI Project 6415 SRI International edition, 1979.

30 C. L. DYM "Expert systems:new approaches to computer-aided engineering" Engineering with computers voll(1985) pp9-25

31 R. A. EDMUNDS "The Prentice Hall Standart Glossary of Computer Terminology" Prentice Hall 1985

32 S. C. EISENSTAT, M. H. SCHULTZ, A. H. SHERMAN "oftware for Sparsial Gaussian Elimination with Limited Core Storage" Sparse Matrix Proceedings, I. S. Duff, G. W. Stewart (eds.) 1978, SIAM Philadelphia, pp 135-153

33 L. M. FAGAN i inni "Representation of Dynamic Clinical Knowledge: Measurment Interpretation in the Intensive Care Unit" Proceedings Fifth IJCAI, 1014-1019, 1979

34 C. A. FELIPPA "Architecture of a distributed analasis network for computational mechanics" Comp. Struct. vol13(1981) pp405-414

35 C. A. FELIPPA "Database in Scientific Computing" Part I Comp. Struct. vol10 (1979) pp33-61 Part II Comp. Struct. vol12(1980) pp131-146

36 S. J. FENVES "Computer aided design in civil engineering" proc. IEEE vol69 (1981) pp1240-1248

37 R. FJELLHEIM "An expert system for Sesam-69 program selection", Norwegian Maritime Research No. 3/1984

38 R. FJELLHEIM, P. SYVERSEN "An expert system for SESAM 69 Structure analysis program selection" Technical report CP-83-6010 Division for Data Technology, Computas, Norway, Jan. 1983

39 J. P. GAGO, D. W. KELLY, O. C. ZIENKIEWICZ "Adaptive Finite Elenent Schemes and a-Posteriori Error Analysis, an Evaluation of Alternatives 3rd World Conf. on FEM L. A. California oct. 1981

40 J. P. GAGO, D. W. KELLY, O. C. ZIENKIEWICZ, I. BABUSKA "A Posteriori Error Analysis and Adaptive Processes in the Finite Element Method. Part II Adaptive Mesh Refinement" IJNME vol19 (1983) No11 pp1621-1656

41 M. GERADIN, S. IDELSEN, M. HODGE "Computational Strategies for the Solution of Large Nonlinear Problems via Quasi- Newton Methods", Comp. and Str. vol 13 (1981), pp. 73-81

42 M. GERADIN, S. IDELSEN, M. HODGE "Nonlinear Structural Dynamic via Newton and Quasi-Newton", Nucl. Engng. Design, vol58 (1980), pp. 339-348

43 J. GOLIŃSKI " Adaptacyjne systemy poszukiwania konstrukcji optymalnej ", PWN 1976

44 J. GOLIŃSKI "Metody optymalizacyjne w projektowaniu technicznym" WNT 1974

45 A. K. GRIFFITH "A Comparison of Three Machine Learning Procedures as Applied to the Game of Checkers" Artifitial Intelligence 5(1974) pp137-148

46 R. H. GUNDERSON, A. CETINER "Element stiffness matrix generator" J. Struct. Div. ASCE 97 pp363-375 (1976)

47 A. K. GUPTA, E. W. NG "Symbolic calculations in finite element dynamic analysis" Proc. 1977 MACSYMA Users' Conf. pp151-160 NASA CP-2012 (1977)

48 A. K. GUPTA, B. MOHRAZ "A method of computing numerically integrated stiffness matrices" IJNMEvol5(1972) pp83-90

49 W. E. HAISLER "Development and evaluation of solution procedures for nonlinear structural mechanics", Texas A and M University dec. 1970

50 M. ISHIZUKA, FU K. S. , YAO J. T. P. SPERIL I - Computer Based Structural Damage Assesment System. Technical Report CE - STR - 81 - 36, Purdue University, November, 1981

51 M. KLEIBER, J. JACHIMOWICZ "Problemy wzorcowe", Warszawa 1987

52 M. KLEIBER "Metoda elementów skończonych w nieliniowej mechanice continuum", PWN 1985

53 A. P. KORNCOFF, S. J. FENVES "Symbolic Generation of FEM Stiffness Matrices" Comp. Struct. vol10 (1979) pp119-124

54 J. KUCWAJ, J. ORKISZ "Procedura symbolicznego różniczkowania na EMC", Mechanika i Komputer tom 5 (1982)

55 C. A. KULIKOWSKI 'Expert System Design and Construction: From Prototyping to Knowledge Base Refinement', Summer School on Expert System Design Methodologies and Tools, Udine Sept. 8-12, 1986

56 I. M. LEVI, N. J. HOFF "Interaction between axisymmetric and nonsymmetric creep buckling of circular cyllindrical shells in axial compression" SUDAR Rep. No388 Dept. of Aeronautics and Astronautics Stanford University

57 I. M. LEWI "Symbolic algebra by computer a pplications to structural mechanics" AIAA/ASME 12th Structures, Structural Dynamics, Materials Conf. Anahem, California (1971)

58 J. W. LIU, A. H. SHERMAN "Comparative Analysis of the Cuthill-McKee and the Reverse Cuthill-McKee Ordering Algorithms for Sparse Matrices", SIAM J. Numer. Anal. , 13, 198-213, 1976

59 W. LUFT, J. M. ROESSET, J. J. CONNOR "Automatic generation of finite element matrices" J. Struct. Div. ASCE 97 pp349-362 (1976)

60 "MARC General Purpose Finite Element Program, User Manual", Marc Analysis Research Corporation

61 H. MATTHIES, G. STRANG "The Solution of Nonlinear Finite Element qations", I J. N. M. E. vol14 (1979), pp1613-1626

62 J. McDERMOTT, "Ri A Rule-Based Configurer of Computer Systems" Technical Report CMU-CS-80-119, Carnegie-Mellon University, 1980

63 MICHALSKI " Learning Machines ", Tioga 1983

64 M. MINSKY "Semantic Information Processing" MIT Press, Cambridge Mass. 1968 p. 12

65 N. J. NILSSON "Principles of Artifitial Intelligence" Tioga Publishing co. Palo Alto 1980

66 A. K. NOOR "Survey of computer programs for solution of nonlinear structural and solid mechanics problems" Comp. Struct. vol13(1981) pp425-465

67 A. K. NOOR, C. M. ANDERSEN "Mixed isoparametric elements for Sant-Venant torsion" Comput. Meth. Appl. Mech. Engng. , pp195-218 (1975)

68 A. K. NOOR, C. M. ANDERSEN "Mixed isoparametric finite element mocels for laminated composite shells" Comput. Meth Appl. Mech. Engng. 11 pp255-280 (1977)

69 A. K. NOOR, C. M. ANDERSEN "Computerized Symbolic Manipulation in Structural Mechanics-Progress and Potential" Comp. Struct. vol 10((1979) pp95-118

70 A. K. NOOR, M. ANDERSEN "Computerized Symbolic Manipulation in Nonlinear Finite Element Analysis" Comp. Struct. vol13(1981) pp379-404

71 A. PEANO "Adaptive Techniques of FEM Computations" proc. of IABSE Workshop, Bergamo 1982

72 A. PEANO "Automated Discretization Control in Finite Element Analysis" Proc. Second World Congress on FEM, Bournemouth Dorset oct. 1978

73 A. PEANO "General Purpose Systems Based on Adaptive Finite Element Sowtware Design Considerations" Proc. Fourth World Congress and Exhibition on FEM, Interlaken Switzerland, sept 1984

74 A. PEANO, I. N. KATZ, M. P. ROSSOW "Nodal Variables for Complete Conforming Elements of Arbitrary Polynomial Order" Comp. Math. with Appl.

75 P. PEDERSEN, "On computer-aided analytic element analysis and the similarities of tetrahedron elements" IJNME 11, pp611-622 (1977)

76 P. PEDERSEN, M. M. MEGAHED, "Axisymmetric element analysis using analytical computing", Comp. Str. 5 pp241-247 (1975)

77 D. S. PERAU "Selection of an Appropriate Domain for An Expert System", THE AI MAGAZINE, Summer 1985.

78 A. PICA, E. HINTON "The Quasi-Newton BFGS Method in Large Deflection of Plates", Numerical Meth. for Nonlinear Problems vol. 1, (1980) pp355-365, Pineridge, Swansea

79 W. J. RASDORF, G. C. SALLEY "Generative engineering databases- towards expert systems" Comp. Struct. vol20(1985) pp11-15

80 J. M. RIVLIN, M. B. HSU, P. V. MARCAL "Knowledge based consultation for finite element analysis" Technical report AFWAL-TR-80-3069 Flight Dynamics Labora-

tory (FIBRA), Wright-Patterson Airforce Base, May 1980

81 N. RIZZI A. TATONE 'Symbolic manipulation in buckling and postbuckling analysis', Comp. and Str. vol 21 No 4 pp. 691- 700 (1985)

82 D. ROGULA "Mechanika, Komputer, Człowiek" Mechanika i Komputer t. 2. str9-30

83 D. ROGULA, M. SZTYREN "Analiza logiczna pracy pręta do celów autonatycznego generowania programów inżynierskich" Mechanika i Komputer t. 2. str 187-198

84 M. F. ROONEY, S. E. SMITH "Artificial Intelligence in Engineering Design" Comp. Struct. vol16(1983) pp279-288

85 M. P. ROSSOW, I. N. KATZ "Hierarchial Finite Elements and Precomputed Arrays", IJNME vol. 12(1978) No6 pp977-1000

86 M. D. RYCHENER "PSRL: an SRL-Based Production-Rule System. Reference Manual"

87 A. L. SAMUEL "Some Studies in Machine Learning Using the Game of Checkers" IBM res. dev. 11(Nov. 1967) pp601-618

88 E. SCHREM "Development and maintenance of large finite element systens in" Structural mechanics computer programs University Press of Virginia Charlottesville 1974

89 E. SCHREM "From Program Systems to Programming Systems for Finite Element Analysis" Formulations and Algorithms in Finite Element Analysis MIT Press Cambridge Mass. 1977 pp163-190

90 E. SCHREM "Trends and Aspects of the Development of Large Finite Element Software Systems" Comp. Struct. vol10(1979) pp419-425

91 E. H. SHORTLIFFE "Computer-Based Medical Consultations: MYCIN" American Elsevier, New York, 1976.

92 J. R. SLAGLE, C. D. FARRELL "Experiments in Automatic Learning for a

Multipurpose Heuristic Program" Commun. ACM Feb. 1971 vol13 No2

93 M. H. SMITH 'A learning program which plays partnership dominoes'. Comm. ACM 16(8) August 1973, pp462-467

94 D. SRIRAM, M. L. MAHER, S. J. FENVES "Knowledge-based expert systems in structural design" Comp. Struct. vol20(1985) pp1-9

95 M. STEFIK, "Planning with Constraints (MOLGEN 1)" Artifitial Intelligence 16:111-140, 1981

96 B. A. SZABO, A. MEHTA "P-convergent Finite Element Approximations in Fracture Mechanics" IJNME vol12(1978) pp977-999

97 M. SZTYREN "Automatyczna generacja programów inżynierskich na podstawie danych symbolicznych" Mechanika i Komputer t. 2 str 211- 218

98 M. TANAKA, Y. SEGUCHI, "Optimum adaptive incremental sequence in non-linear analysis", Comp. Mech. vol 1 No 4, 1986 pp. 243-247

99 R. M. THRALL, C. COOMBS, R. L. DAVIS "Decision Processes" Wiley 1954

100 D. A. WATERMAN "Generalisation Learning Techniques for Automating the Learning of Heuristics" Artificial Intelligence 1(1970), pp121-170

101 D. J. WILKINS, JR "A pplications of a symolic algebra manipulation language for composite structures analysis" Comp. 3 Str. 801-807 (1973)

102 P. H. WINSTON, B. HORN "Lisp" 2-nd edition Addison- Wesley 1984

103 A. K. WONG "Symbolic computing" Num. and Comp. Meth. in Structural Mechanics pp459-477 Academic Press New York (1973)

104 A. K. WONG, BUGLIARELLO "Artificial Intelligence in Continuum Mechanics" Journ. of the Eng. Mech. Div. vol96 Dec1970 pp 1239-1265

105 O. C. ZIENKIEWICZ "Iterative Methods and Hierarchical Approaches:a Prospect for the Future of the Finite Element Method" paper for special issue to commem-

orate ASEA 100th Anniversary, Sweden Jan. 1984

106 O. C. ZIENKIEWICZ, K. MORGAN 'Finite Elements and Approximation' Wiley, 1982

107 O. C. ZIENKIEWICZ, D. W. KELLY, J. GAGO, I. BABUSKA "Hierarchical Finite Element Approaches, Error Estimates and Adaptive Refinement" The Mathematics of Finite Elements and Applications IV Academic Press 1982 pp313-346

108 O. C. ZIENKIEWICZ, J. P. GAGO, D. W. KELLY "The Hierarchical Concept in Finite Element Analysis" Comp. Struct. vol16(1983) pp53-65