Articole

SISTEM PENTRU SIMULARE ŞI CONTROL, BAZAT PE CUNOŞTINŢE

dr. Florin Stănciulescu
mat. Victor Popa

Institutul de Cercetări în Informatică

Rezumat: Se prezintă un sistem pentru modelare, simulare şi control, bazat pe cunoştinţe, în faza de prototip. Sistemul, denumit HYBRID, se bazează pe o metodă şi o tehnică de calcul hibrid, care îmbină avantajele modelului de simulare numerică cu cele ale unui model de control euristic, aflat într-o bază de cunoştinţe (model matematico-euristic). Un algoritm de simulare şi control, bazat pe prelucrarea cunoştinţelor, ca şi o teoremă de compatibilitate (între modelul matematic şi cel euristic) asigură suportul toretic al sistemului. Instrumentele informatice, care au stat la baza realizării prototipului sistemului HYBRID sunt produsul/limbajul de simulare Mathcad şi un expert system shell original-Inferno (scris în Prolog), aflat la cea de a 2-a versiune, care asigură achiziţionarea cunoştinţelor de la expert, construcţia bazei de cunoştinţe şi prelucrarea acestora şi interfaţa cu utilizatorul. Experimentele au fost efectuate cu ajutorul unui model de simulare standard, constituit din 30 ecuaţii de stare, din care 6 ecuaţii de stare cu timp-discret, neliniare şi 24 ecuaţii algebrice neliniare, care descriu procesele (naturale) de circulaţie, stocare şi împrospătare a apei într-o reţea hidrologică, formată din 6 lacuri mari, interconectate prin canale şi şenale (din Delta Dunării). Modelul euristic de control este constituit din 510 reguli de tip expert (reguli de comportare, de control şi de decizie), aflate într-o bază de cunoştinţe. Rezultatele obţinute prin simulare pe un calculator PC-386 sub MS-DOS 6.0, au fost interpretate de către ecologi şi hidrologi şi folosite la fundamentarea propriilor lor decizii.

Cuvinte cheie: Model matematico-euristic, simulare bazată pe cunoştinţe, control bazat pe cunoştinţe, bază de cunoştinţe, motor inferenţial, prototip sistem.

1. Introducere

1.1. Scopul, obiectivele şi caracteristicile sistemului

Această lucrare are ca scop prezentarea sistemului pentru simularea şi controlul bazat pe cunoştinţe, obiectivul fiind realizarea unui prototip al acestui sistem, pornind de la un sistem experimental [1].

Ideea de modelare şi simulare bazată pe cunoştinţe a apărut în deceniul '80, ca o alternativă la modelarea şi simularea numerică, care deşi au avantajul de a fi riguroase, tind să fie mult mai precise decît este necesar şi de aceea devin mari consumatoare de timp şi de resurse de calcul. Pe de altă parte, în cazul sistemelor mari, complexe, metodele de modelare şi simulare numerică au o aplicabilitate restrînsă şi de cele mai multe ori implică o reducere/simplificare drastică a modelului. Acestea sunt principalele raţiuni care au condus la apariţia modelării şi simulării bazată pe cunoştinţe [4, 5, 8 - 10, 15], a simulării calitative [6, 7], şi a simulării flexibile [5].

Aceleaşi raţiuni au condus şi la controlul bazat pe cunoştinţe [2, 3, 12, 13].

Aşadar, scopul şi obiectivele prototipului sistemului pentru modelarea, simularea şi controlul bazat pe cunoştinţe sunt acelea de a permite:

- realizarea de experimente privind modelarea şi simularea unor sisteme de complexitate medie, atît în ceea ce priveşte starea actuală a sistemului simulat, cît şi predicţia evoluţiei stărilor acestuia şi-nu în ultimul rînd-determinarea comenzilor capabile să readucă mărimile de stare în intervalele de (sub) optimalitate atunci cînd acestea au ieşit din limitele specificate de către expert (controlul sistemului);
- îmbunătăţirea funcţiunilor, caracteristicilor şi performanţelor produsului realizat, în urma experimentelor efectuate asupra prototipului;
- concluzionarea asupra posibilităţilor de a dezvolta-în viitor-un nou sistem pentru modelarea şi simularea bazată pe cunoştinţe, mai performant în ceea ce priveşte portanţa sistemului (dimensionalitate şi complexitate mai mare).

Trebuie menţionat că toate aceste obiective au fost atinse, aşa cum va reieşi din lucrare şi din rezultatele experimentelor la care prototipul a fost supus.

În esenţă, caracteristicile prototipului sistemului HYBRID, pentru modelare şi simulare bazată pe cunoştinţe sunt următoarele:

- este un sistem hibrid, care utilizează concomitent metode şi tehnici specifice simulării numerice, cu metode şi tehnici de prelucrare a cunoştinţelor euristice (deci metode calitative);
- modelul pe care se bazează sistemul HYBRID este un model de simulare matematico-euristică, alcătuit dintr-un model diferenţial cu timp-discret, neliniar, şi un model euristic compus din reguli deduse din cunoaşterea expertă (aflate în baza de cunoştinţe pentru simulare);
- sistemul utilizează un algoritm de simulare original, denumit HYBRID;
- suportul informatic al sistemului HYBRID este compus dintr-un program de simulare (care utilizează produsul Mathcad) şi un shell denumit Inferno, care asigură achiziţionarea cunoştinţelor, construcţia bazei de cunoştinţe pentru simulare şi procesul de inferare (forward sau backward) al cunoştinţelor;
- prototipul a fost testat cu ajutorul unui model matematico-euristic, compus din 30 de ecuaţii de stare, 510 reguli euristice şi 25 de fapte;
- limitele (portanţa) prototipului sunt cel puţin egale cu cele menţionate la punctul anterior, dar avem toate motivele să credem că ele sunt, în realitate, peste aceste cifre.

1.2. Justificarea soluţiei

Soluţia aleasă de noi pentru a elabora prototipul sistemului pentru modelarea şi simularea bazată pe

cunoştinţe - HYBRID, se bazează pe următoarele entităţi:

(a) Modelul de simulare matematico - euristică;

(b) Algoritmul de simulare hibridă;

(c) Instrumente informatice adecvate (Mathcad, Inferno).

Din punct de vedere teoretic la baza realizării prototipului au stat o serie de cercetări multi- şi inter - disciplinare (efectuate în anul 1992 şi primul semestru din 1993), care au inclus următoarele domenii de cercetare: teoria sistemelor, analiza sistemelor, analiza matematică, logica matematică, teoria bazelor de cunoştinţe, teoria algoritmilor, teoria mulţimilor vagi şi altele.

Soluţia finală, aleasă pentru prototipul sistemului, aceea de sistem pentru modelare şi simulare bazată pe cunoştinţe utilizînd un model matematico-euristic şi un algoritm de simulare hibridă se justifică, atît din punct de vedere teoretic, cît şi practic:

a) spre deosebire de simularea calitativă sau de simularea flexibilă ş.a, care renunţă la simularea numerică, simularea hibridă (bazată pe model matematico-euristic) nu renunţă la unele dintre avantajele simulării numerice, pentru a le completa cu cele ale simulării euristice/calitative; aşadar, cele două metode sunt considerate ca fiind complementare şi sunt tratate ca atare;

b) spre deosebire de simularea calitativă sau de simularea flexibilă ş.a, care oferă numai soluţii calitative, simularea hibridă (bazată pe model matematico-euristic) oferă şi soluţii cantitative-cînd expertul cere astfel de soluţii - ca de exemplu: pentru fundamentarea unor soluţii de proiectare prin simulare, pentru asistarea factorului de decizie, atunci cînd acesta trebuie să ia decizii cantitative etc.

Două exemple din experienţa colectivului pot fi concludente:

În **sistemele de comandă**, şi vom lua ca exemplu sistemele de comandă electro- hidraulică, se pune problema ca amplitudinea mărimilor de stare (ex. debitul de fluid Q şi presiunea P), ca şi timpii de răspuns (tr) la anumite comenzi, să se afle în limitele intervalelor de suboptimalitate: $Q \in [Q_1, Q_2]$, $P \in [P_1, P_2]$, $t_r \in [t_1, t_2]$. Aceste limite sunt precizate de către expert (proiectant) foarte exact, depăşirea lor conducînd la urmări nedorite (ex. reproiectarea elementelor hidraulice).

Or, numai modelarea şi simularea hibridă asigură gradul de acurateţe cerut de astfel de aplicaţii.

În **sistemele naturale**, şi vom lua ca exemplu sistemul ecologic/hidrologic Delta Dunării, mărimile de stare trebuie, de asemenea să aparţină unor intervale de suboptimalitate, date de către expert, depăşirea limitelor acestora aducînd mari prejudicii ecologice şi economice acestui sistem natural. Astfel, factorul de înprospătare a apei W, trebuie să aparţină intervalului $[W_1, W_2]$, concentraţia de fosfor $CP \in [CP_1, CP_2]$ etc.

Or, în cazul în care una dintre mărimile de stare iese din limitele mai sus menţionate, factorul de decizie sau expertul implicat este interesat să ştie ce acţiune trebuie să întreprindă (în limbajul teoriei sistemelor: cum trebuie să modifice o variabilă de comandă), astfel încît să readucă în limitele dorite variabila (sau variabilele) de stare respectivă. Dar nu calitativ!

În rezumat, prototipul sistemului HYBRID, pentru modelarea şi simularea bazată pe cunoştinţe asigură rezolvarea a trei probleme majore, din domeniul asistării deciziei (manageriale, expert, cercetător):

- simularea stării(lor) sistemului studiat şi afişarea grafică (în 2D) sau numerică a rezultatelor; aceasta se face cu ajutorul unor programe de simulare, utilizînd produsul Mathcad (sau Matlab, sau Simulink), modificat cu introducerea verificării dacă variabila de stare $x_i \in [x_{i1}, x_{i2}]$, $(i=1,2,..., n)$;
- predicţia evoluţiei stării(lor) sistemului studiat, pe un interval de prognoză determinat;
- sinteza (evaluarea) comenzii (comenzilor) ce trebuiesc aplicate sistemului simulat, pentru ca variabilele de stare $x_i \notin (x_{i1}, x_{i2})$ să reintre în intervalele de tip expert; pentru aceasta se utilizează în plus şi produsul de tip shell, Inferno.

Subliniem faptul că, în unele cazuri, sistemul poate furniza expertului mai mult de o soluţie (2 sau chiar 3), în care caz acesta va trebui să aleagă. În ori ce caz, datorită celui de al treilea punct (privitor la sinteza comenzii), prototipul sistemului HYBRID este nu numai un sistem de modelare şi simulare, ci şi un **sistem de control bazat pe prelucrarea cunoştinţelor.**

2. Arhitectura prototipului sistem

Arhitectura prototipului sistem a suferit modificări faţă de arhitectura sistemului experimental, nu numai în privinţa numărului de componente, suferind modificări şi unele module componente. De aceea prezentăm această schemă (figura 1) cu precizările de rigoare. Prototipul sistem are în componenţa sa următoarele module:

2.1. Modulul de achiziţionare cunoştinţe expert

Acest modul facilitează achiziţionarea, de la expert, a cunoştinţelor care au ca sursă experienţa (expertului) în domeniile căruia îi aparţine sistemul simulat. Cunoştinţele respective se achiziţionează sub formă de reguli euristice de comportare, de control şi de decizie, pentru a căror achiziţie s-a proiectat un sistem care prezintă expertului, ecrane standard, acesta neavînd

altceva de făcut decît să completeze de la tastatură, elementele care compun regulile euristice, mai precis premisele (denumite "conditions") și concluziile (denumite "actions"). Cum arată aceste ecrane se poate vedea în figura 2. De notat faptul că acest modul face parte dintr-un shell, denumit Inferno, care este scris în Turbo Prolog și care va fi amplu prezentat în capitolul 5, alineatul 5-2.

Modulul a fost modificat pentru a accepta nu numai cunoștințe simbolice sau scalare, ci și elemente de matrice sau vectori.

2.2 Baza de cunoștințe (pentru simulare)

Cunoștințele achiziționate de la expert, cu ajutorul modulului (1) sunt stocate în baza de cunoștințe pentru simulare sub forma regulilor euristice de comportare (cod Bik), a regulilor euristice de control (cod Cik) și a regulilor euristice de decizie (cod Dik). În codurile regulilor, i reprezintă o literă de ordine al regulii (a,b,c,d...), în timp ce k reprezintă pasul de simulare/ momentul de timp-discret ($k = 1,2,3,4,...k_f$).

Remarcăm faptul că baza de cunoștințe pentru simulare reprezintă o bază de cunoștințe specializată (ea conține cunoștințe euristice necesare simulării sistemului studiat). De asemenea, structura ei este suficient de versatilă, pentru a putea spori cunoștințele achiziționate de la expert cu noi reguli euristice, dar și pentru a modifica unele reguli euristice.

Pentru a exemplifica modul cum arată baza de cunoștințe (reguli) pentru simulare, am prezentat cîteva ecrane în figura 3.

Baza de cunoștințe pentru simulare (prototip) a fost modificată, față de versiunea anterioară pentru a putea include și elemente de matrici sau vectori.

2.3 Baza de fapte

Baza de fapte nu a suferit nici o modificare în versiunea prototip față de sistemul experimental. Un fapt are un nume codificat (ex. Fk) și următoarea formă standard: Cod, Atribut, Valoare, Validitate. (vezi fig. 3)

2.4 Baza de date

La acest punct au intervenit elemente substanțial îmbunătățite față de faza anterioară (sistem experimental). Așa cum rezultă și din figura 1, care prezintă noua arhitectură (a prototipului sistem), prototipul sistemului pentru modelarea și simularea bazată pe cunoștințe poate lucra cu oricare din bazele de date mai jos specificate: o bază de date Mathcad, o bază de date Matlab, o bază de date Excel, o bază de date relațională (ex. creată cu ajutorul Dbaze), o bază de date obiecte (ex. creată cu limbajul C++).

Aceste baze de date, create de către utilizator, furnizează datele problemei/modelului numeric de simulare și permit stocarea rezultatelor obținute prin simulare numerică. Subliniem că programul (de tip shell) Inferno, poate căuta în orice bază de date (matriceală, vectorială, scalară) și poate utiliza datele găsite pentru activarea programului de inferare a regulilor euristice, din baza de cunoștințe.

2.5 Modulul de prelucrare a cunoștințelor

Acest modul este constituit din produsul-program Inferno (versiunea V.1.) care include facilități pentru: achiziționarea cunoștințelor, construirea bazei de cunoștințe, inferarea cunoștințelor (motorul inferențial). Programul Inferno va fi descris în detaliu în alineatul 5.2. Aici vom descrie numai modificările survenite față de versiunea V.0. (utilizată în sistemul experimental).

Așadar, în noua versiune V.1., shell-ul Inferno poate infera (în mod forward sau backward), și cunoștințe care includ elemente de matrice (rezultate din simulare), cît și elemente vectoriale (parametri). Această facilitate permite conexiunea directă dintre acest Shell Inferno și orice produs-program de simulare (ex. Mathcad, Matlab, Simulink etc.).

2.6 Modulul de construire a modelului de simulare numerică

Nu au intervenit modificări în acest modul prototip, față de cel utilizat în faza de sistem experimental. El utilizează facilitățile de construire a modelului de simulare numerică, oferite de limbajul Mathcad. Subliniem însă faptul că, sistemul HYBRID poate lucra și cu alte programe de simulare (ex. Matlab, Simulink etc).

2.7 Modulul pentru rezolvare probleme numerice

Nici în acest modul nu au intervenit modificări față de faza de sistem experimental, el utilizînd, de asemenea, facilitățile produsului Mathcad.

2.8 Modulul de simulare numerică

Acest modul utilizează, de asemenea, facilitățile produsului Mathcad, dar, în contextul prototipului sistemului HYBRID, el poate conlucra cu shell-ul Inferno, sub forma matriceală sau vectorială, prin intermediul bazei de date "Baza". Acest lucru asigură un grad mai mare de integrare a procesului: simulare numerică - simulare calitativă (euristică), absentă în faza de sistem experimental.

2.9 Modulul de reprezentare grafică

Acest modul este același cu cel utilizat în faza de sistem experimental și este inclus în Mathcad.

2.10 Modulul de afișare acțiuni (comenzi, control)

Acest modul a suferit unele modificări, nu atît ca ecrane, cît ca informații pe care le poate afișa. În adevăr, în cazul prototipului, acest modul poate afișa și modificări în elemente ale unor vectori (de comandă), ceea ce constituie un element esențial față de sistemul experimental. Acțiunea se exprimă prin: Atribut, valoare (element de vector), Grad de certitudine, Grad de incertitudine, C_f (certitudine - incertitudine).

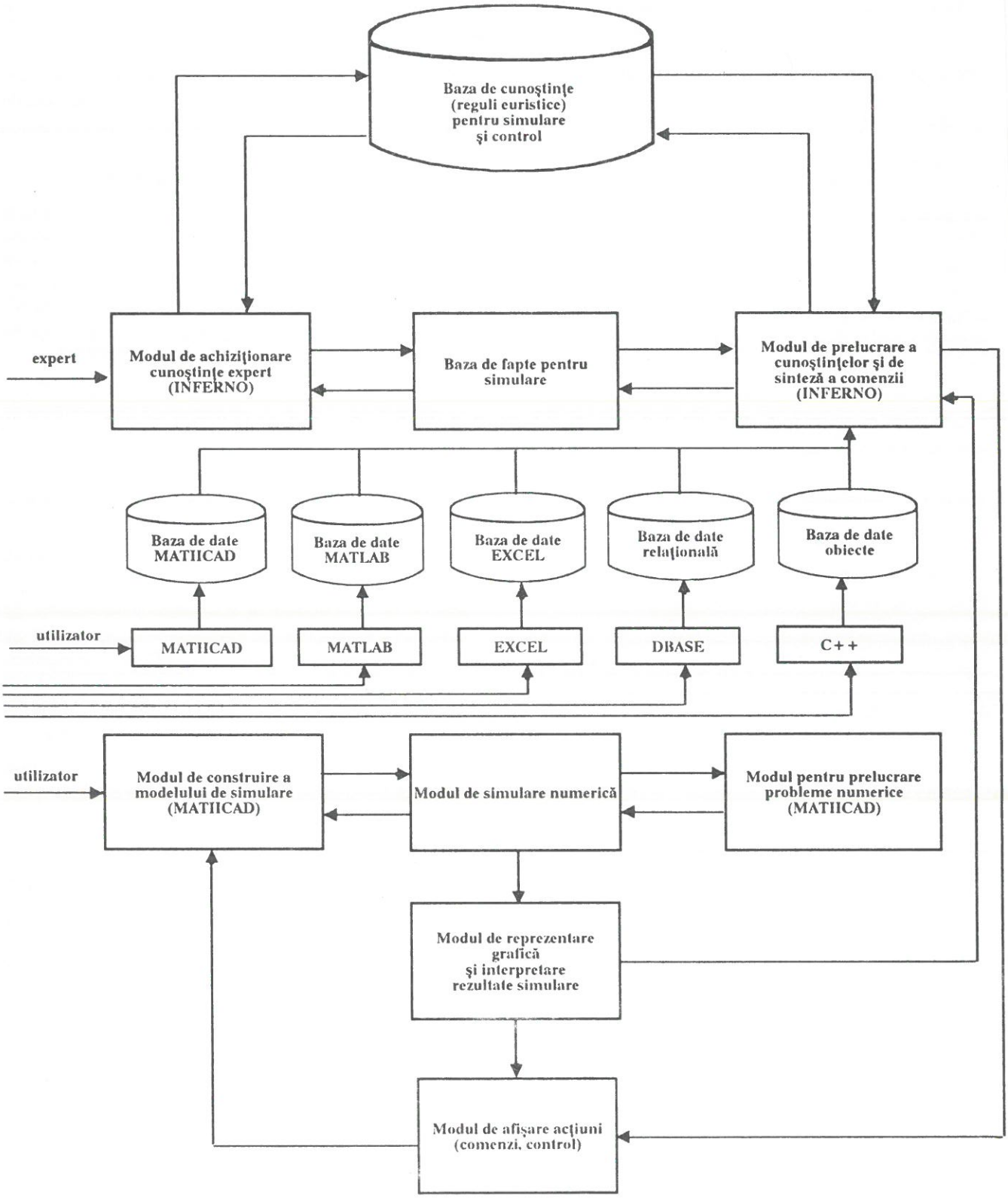


Figura 1. Arhitectura prototip sistem pentru modelarea, simularea și controlul bazat pe cunoștințe

3. Baza de cunoştinţe pentru simulare

Baza de cunoştinţe pentru simulare a prototipului constituie une dintre piesele principale ale sistemului pentru modelare şi simulare bazat pe cunoştinţe. Ea are două componente principale:

Baza de reguli euristice

Aşa cum am arătat mai sus, în baza de cunoştinţe includem numai acele cunoştinţe care pot fi scrise sub forma unor reguli euristice, de o formă standard, pe care o vom prezenta în cele ce urmează (folosind limbajul Prolog standard). Fără a mai relua unele detalii prezentate în [1], vom arăta că am clasificat regulile euristice în 3 clase:

- reguli de comportare-care arată cum se modifică starea unui sistem, la modificări ale parametrilor acestuia;
- reguli de control - care arată cum se modifică starea unui sistem, la modificări ale variabilei (lor) de comandă;
- reguli de decizie - care arată **cum trebuie modificate** variabilele de comandă, pentru a readuce în intervalele de (sub) optimalitate acele stări care au ieşit din limite.

Forma clasică a regulilor euristice este:

<Dacă (condiţie), Atunci (concluzie)> (1)

Cu aceste precizări, prezentăm mai jos structura regulilor euristice din baza de cunoştinţe pentru simulare şi control a prototipului.

Forma standard a regulilor euristice

(1) **Forma standard a regulilor euristice de comportare** (behavioural rules):

Cod regulă	Acţiune (action)	Condiţie (condition)
Bαk	x_{ik}	$C_{i1}(p_1)$ o $C_{i2}(p_2)$ o...o $C_{in}(p_n)$

unde:

α = a,b,c,...,

k = 1,2,...,k_f,

p_i = parametrii de care depinde comportarea lui xi,

o = conectivele logice "and", "or"

(negaţia logică "not" poate, de asemenea, interveni în partea condition a regulii).

Notă: Atunci cînd elementele C_i, din partea Condiţie a regulii, sunt conectate prin conectivul "or", regula respectivă este descompusă în mai multe reguli, avînd aceeaşi parte Acţiune, dar diferind prin Condiţie (aceasta este dictată de modul de lucru al shell- ului).

(2) **Forma standard a regulilor euristice de control** (control rules):

Cod regulă	Acţiune (action)	Condiţie (condition)
Cαk	$x_{i,k+2}$	$C_i(p_1, p_2, ..., p_n)$ o x_{ik} o $[u_{i,k+1} \leftarrow u_{ik} \pm \Delta u_{ik}]$

Formula $[u_{i,k+1} \leftarrow u_{ik} \pm \Delta u_{ik}]$, arată modul de evaluare a noii comenzi, $u_{i,k+1}$; în această formulă Δu_{ik} este un increment (decrement) care este evaluat cu ajutorul cunoştinţelor din baza de cunoştinţe.

(3) **Forma standard a regulilor euristice de decizie** (decision rules):

Cod regulă	Acţiune (action)	Condiţie (condition)
Dαk	$u_{i,k+1} = u_{ik} \pm \Delta u_{ik}$	$(x_i < x_{imin})$ or $(x_i > x_{imax})$

unde : $[x_{imin}, x_{imax}]$ reprezintă intervalul de (sub) optimalitate, al variabilelor de stare, specificat de către expert.

Remarcă : Menţionăm că în regulile de mai sus x_i,p_i sunt elemente de matrice sau de vector, ceea ce reprezintă un progres real al prototipului sistemului HYBRID, faţă de sistemul experimental.

Baza de fapte

Spre deosebire de baza de cunoştinţe prototip (reguli euristice) care a suferit modificări importante (pe care le-am menţionat), baza de fapte nu a suferit modificări esenţiale, astfel încît structura unui fapt rămîne cea prezentată în [1] : Cod, Atribut, Valoare, Validitate.

Menţionăm că, în afară de baza de cunoştinţe (reguli euristice) şi baza de fapte, prototipul mai face apel şi la o bază de date "Baza", în care sunt stocate rezultate de simulare (matrice, vectori) şi parametrii sistemului (vectori).

4. Modelul şi algoritmul hibrid de prelucrare a cunoştinţelor

4.1. Modelul matematico-euristic de simulare

Modelul de simulare care a stat la baza prototipului este practic acelaşi cu modelul utilizat în faza de sistem experimental, adică un model matematico-euristic standard. Diferenţele sunt numai de ordin formal.

(a) **Modelul de simulare numerică**

Acest model este compus, în forma sa standard cu care vom lucra în continuare, dintr-un sistem de ecuaţii cu timp discret şi formule de calcul, derivate din analiza sistemului de simulat; modelul mai include condiţii iniţiale şi intervale (limite) de admisibilitate sau de suboptimalitate, furnizate de către expert.

Forma standard a modelului de simulare numerică este:

$$x_i(k+1) = A_i x_i(k) + B_i u_i(k) + f_i(x_i) + v_i(x) \quad (2)$$

$$x_i(0) = x_{i0} \quad (3)$$

$$v_i(x) = \sum_{j=1, j\neq 1}^{n} g_{ij}(x_j), \quad (i = 1, 2, ..., n) \quad (4)$$

la care se adaugă restricţiile de inegalitate:

$$x_{i1} \le x_i \le x_{i2} \quad (5)$$

$$u_{i1} \le u_i \le u_{i2} \quad (6)$$

În modelul de mai sus, (2) reprezintă ecuații de stare vectoriale (cîte una pentru fiecare subsistem i), în care: A_i și B_i sunt matrice de dimensiuni potrivite, f_i sunt funcții vectoriale care descriu neliniaritățile sistemului simulat, v_i descriu interacțiunile subsistemului i cu celelalte n–1 subsisteme; x_{i1} și x_{i2} reprezintă limitele minimă și maximă ale intervalului în care, în mod normal, trebuie să se afle x_i; în (6) u_{i1} și u_{i2} reprezintă limite admisibile pentru comenzile u_i.

Un exemplu concret de astfel de model a fost prezentat în capitolul 6. În model sunt incluse și modelele de simulare stocastică [1].

(b) Modelul de simulare euristică

Acest model se află în baza de cunoștințe pentru simulare și este compus atît din reguli euristice (de comportare, de control și de decizie), ca și din fapte.

Forma sa standard este următoarea:

- reguli euristice de **comportare**:

$$<x_i(k) \leftarrow C_{i1}(p_1) \circ C_{i2}(p_2) \circ \ldots \circ C_{in}(p_n)> \quad (i=1, 2, \ldots, n)$$

- reguli euristice de **control**:

$$<x_i(k+2) \leftarrow C_{i1}(p_1) \circ C_{i2}(p_2) \circ \ldots \circ C_{in}(p_n) \wedge (u_i(k+1) \leftarrow u_i(k) \pm \Delta u_i(k)) \quad (i=1, 2, \ldots, n)$$

- reguli euristice de **decizie**:

$$<(u_i(k+1) \leftarrow u_i(k) \pm \Delta u_i(k)) \leftarrow (x_i < x_{i1}) \vee (x_i > x_{i2})> \quad (i=1,2,\ldots,n)$$

- Fapte:

$$<x_i(k)=\alpha_i> \quad \text{(serie timp)} \quad (i=1,2,\ldots,n;\ k=0,1,2,\ldots,k_f)$$

$$<p_i(k)=\beta_i> \quad \text{(serie timp)} \quad (i=1,2,\ldots,m;\ k=0,1,2,\ldots,k_f)$$

De remarcat că, în calculator, atît regulile euristice, cît și faptele sunt reprezentate așa cum se arată în ecranele (screen) din figura 3.

Precizăm faptul că, în regulile euristice de comportare, prin $x_i(k)$ înțelegem faptul că subsistemul S_i se află în starea xi la momentul k; $x_i(k+2)$ trebuie interpretat ca o schimbare în starea subsistemului, sub acțiunea noii comenzii $u_i(k+1)$.

Remarcă : pentru a putea funcționa împreună, cele două părți componente ale modelului matematico-euristic trebuie să fie **compatibile**. Acest lucru se întîmplă numai în anumite condițiuni , pe care le-am stipulat în [10] sub forma unei teoreme de compatibilitate. Datorită caracterului teoretic al acestei probleme, dar cu serioase implicații în aplicații, redăm aici numai, rezultatele obținute (cei ce doresc amănunte le pot găsi în [10]). Așadar, teorema de compatibilitate a celor două părți ale modelului se demonstrează pornind de la formula de evaluare a comenzii:

$$u_i(k+1) = u_i(k) + \Delta u_i(k)$$

pe baza căreia se construiește șirul:

$$u_i(1) = u_i(0) + \Delta u_i(0)$$

$$u_i(2) = u_i(1) + \Delta u_i(1) = u_i(0) + \Delta u_i(0) + \Delta u_i(1)$$

$$\vdots$$

Rezultă evident un astfel de șir și pentru stările xi(k), folosind ecuația de stare (2):

$$x_i(1) = A_i x_i(0) + B_i u_i(0) + f_i(x_i(0)) + v_i(x(0))$$

$$x_i(2) = A_i x_i(1) + B_i u_i(1) + f_i(x_i(1)) + v_i(x(1)) =$$

$$= A_i^2 x_i(0) + (A_i + I) B_i u_i(0) + B_i \Delta u_i(0) +$$

$$+ A_i f_i(x_i(0)) + v_i(x(0)) + f_i(x_i(1)) + v_i(x(1))$$

$$\vdots$$

Utilizînd metoda inducției complete, rezultă în final faptul că, pentru ca $x_i \in [x_{i1}, x_{i2}]$ trebuie îndeplinită condiția:

$$x_{i1} - \Omega_i(k) \le \sum_{j=0}^{k-1} A_i^{k-1} B_i u_i + \sum_{l=0}^{k-2} \sum_{j=0}^{k-i-2} A_i^{k-i-2} B_i \Delta u_i(1) \le x_{i2} - \Omega_i(k), \quad (7)$$

unde:

$$\Omega_i(k) = A_i^k x_{io} + \sum_{j=0}^{k-1} A_i^{k-j-1} \left(f_{ij} + v_{ij} \right)$$

$$f_{ij} = f_j\left(x_i(j)\right)$$

$$v_{ij} = v_j\left(x_i(j)\right)$$

4.2. Algoritmul de simulare HYBRID. Sinteza comenzii.

Toate cele prezentate în capitolele 1-3 au permis elaborarea unui algoritm de simulare și control hibrid, denumit pe scurt HYBRID, care, utilizînd modelul matematico-euristic, este capabil să simuleze comportarea sistemului analizat, să efectueze o predicție a stărilor sistemului (pe o perioadă bine determinată de timp) și să evalueze corecțiile care trebuie operate asupra comenzilor aplicate sistemului, în cazul în care unele stări x_i ies din intervalele $[x_{i1}, x_{i2}]$, astfel încît ele să reintre în aceste intervale.

Algoritmul HYBRID se prezintă astfel:

Pasul 1: Se REZOLVĂ problema de simulare numerică, descrisă cu ajutorul modelului cu timp discret (2)-(4) și fie $x_i(k)$ valorile obținute prin simulare (pentru $u_i(k)$ dat, $i=1, 2, \ldots, n$; $j=1, 2, \ldots, m$; $k \in [0, k_f]$);

Pasul 2: La fiecare moment de timp k se VERIFICĂ dacă variabilele de stare x_i ($i=1, 2, \ldots, n$), obținute la pasul 1, aparțin intervalului $[x_{i1}, k_{i2}]$:

- dacă NU, se merge la pasul 3!
- dacă DA, se merge la pasul 5!

Pasul 3: Se CAUTĂ o nouă valoare a variabilei de control

u_i (i=1, 2, ..., m), utilizînd cunoştinţele din baza de cunoştinţe (reguli şi fapte) şi mecanismul de prelucrare a cunoştinţelor (din produsul Inferno), evaluînd incrementul $\Delta u_i(k)$ cu care trebuie modificată comanda $u_i(k)$, astfel încît x_i să aparţină $[x_{i1}, x_{i2}]$;

Pasul 4: Se REVINE la pasul 1!

Pasul 5: Se EXAMINEAZĂ tendinţa de evoluţie a variabilelor de stare x_i (i=1, 2, ..., n), obţinute la pasul 1, care aparţin $[x_{i1}, x_{i2}]$ utilizînd acelaşi mecanism de la pasul 3:

dacă

- $(x_i(k+1)<x_i(k)) \wedge (|x_i(k+1)-x_{i1}|<\varepsilon_i) \vee$
  $\vee (x_i(k+1)>x_i(k)) \wedge (|x_{i2}-x_i(k+1)|<\varepsilon_2)$

atunci se merge la pasul 3!

- dacă NU, atunci se merge la pasul 6!

Pasul 6: Se verifică dacă k < kf:

- dacă DA, se revine la pasul 1!
- dacă NU, se merge la pasul 7!

Pasul 7: Se TIPĂRESC sau se TRASEAZĂ graficele mărimilor de stare $x_i(k)$ $(i=1,2,\ldots,n;\ k \in [k_0, k_f])$

Pasul 8: STOP!

Subliniem faptul că algoritmul a fost testat pe mai multe cazuri (vezi capitolul 6) şi a dat rezultate corecte, el fiind validat, atît din punct de vedere logic, cît şi prin comparare cu rezultate experimentale (măsurători).

5. Produse informatice utilizate în cadrul prototip sistem

5.1. Programul de simulare numerică (utilizînd limbajul de simulare Mathcad)

Pentru simulare numerică au fost utilizate facilităţile produsului Mathcad, care are un limbaj de simulare ce oferă utilizatorului o serie de facilităţi privind:

- definirea ecuaţiilor modelului de simulare numerică şi introducerea datelor;
- definirea textelor/comentariilor;
- definirea ploterelor (grafică în 2D);
- definirea şi rezolvarea blocurilor.

Deoarece produsul a fost prezentat în [1] şi de altfel produsul Mathcad are un Help destul de complet, vom arăta aici numai modul cum a fost utilizat el în cadrul prototipului sistem, mai ales în cadrul experimentelor efectuate pentru a demonstra consistenţa soluţiei alese pentru prototip. Astfel pentru simularea numerică, în cadrul prototipului sistem au fost utilizate următoarele facilităţi ale limbajului Mathcad:

(1) Modulul de scriere a modelului de simulare numerică: ecuaţii cu timp-discret, formule de calcul, introducerea condiţiilor iniţiale şi a datelor, restricţii de tip inegalitate sub forma de intervale de (sub) optimalitate pentru variabilele de stare x_i;

(2) Modulul de rezolvare a sistemului de ecuaţii cu timp-discret şi de calcul a unor funcţii neliniare care intervin în model;

(3) Modulul de grafică (în 2D), pentru afişarea rezultatelor obţinute prin simulare şi pentru compararea acestora cu datele rezultate din măsurători.

Programul poate semnala ieşirea din limitele de (sub) optimalitate a unor variabile de stare, ceea ce permite factorului de decizie să aprecieze dacă este sau nu cazul să consulte baza de cunoştinţe, prin intermediul shell-ului Inferno (ce va fi descris ulterior), astfel încît el poate fi utilizat şi ca Sistem Suport pentru Decizia Managerială bazată pe Simulare (SSD-MS).

Programul este foarte versatil, în sensul că utilizatorul (expertul) poate modifica condiţiile iniţiale, datele problemei, formule de calcul şi chiar modelul de simulare, în circumstanţe care cer acest lucru (de exemplu, atunci cînd apar neconcordanţe între rezultatele simulării şi măsurători).

Notă: Subliniem faptul că sistemul poate folosi şi alte limbaje sau produse-program de simulare, cum sunt Matlab, Simulink ş.a. dacă acestea oferă avantaje în simulare.

5.2 Programul de achizitionare şi prelucrare a cunoştinţelor: Inferno V1.0

Versiunea 1.0 al lui Inferno constă în integrarea acestuia cu produse deja existente în scopul extinderii structurilor de date şi a posibilităţilor de prelucrare.

Astfel, posibilitatea utilizării bazelor de date, matriceale Matlab,Mathcad, a bazelor de date Excel,a bazelor de date relaţionale sau a bazelor de date obiect a fost principala sarcină de realizat în versiunea actuală.

Inferno este un shell bazat pe reguli, cu aplicaţii în multe domenii de activitate. Principalele funcţii oferite de Inferno sunt:

- gestiunea bazei de cunoştinţe;
- înlănţuirea "forward" a regulilor;
- înlănţuirea "backward" a regulilor (la alegere);
- propagarea factorului de validitate (dacă acesta este dat);
- trasarea pas cu pas a procesului de înlănţuire;
- vizualizarea regulilor aplicate în timpul procesului de înlănţuire;
- asigurarea unui help corespunzător;
- gestiunea rezultatelor;
- interfata utilizator;
- lucrul cu fişiere;
- diverse statistici privind volumul bazei de cunoştinţe.

O regulă Inferno este un triplet de forma:

<Acțiune, Condiție, Factor-validitate>

Pentru o simplificare a formatului, dubletul <Acțiune, Factor-validitate> a fost implementat utilizînd structura:

- Cod acțiune;
- Atribut;
- Valoare;
- Factor-validitate.

<Condiție> este o expresie logică construită din formule elementare, cu ajutorul operatorilor AND, OR, NOT, o formulă elementară avînd structura:

- atribut;
- valoare;
- predicat(=, >, <, ≥, ≤, < >, predicate utilizator).

Tripletul <Atribut,Valoare,Predicat> ce apare în corpul unei condiții este elementul de bază al unei reguli. În timpul procesului de calcul Inferno caută valoarea(valorile) atributului al cărui nume este <Atribut> și compară aceste valori cu <Valoare> utilizînd predicatul <Predicat>. Valoarea (valorile) unui atribut dat se găsesc memorate în:

- baza de date Inferno
- baza de matrice Matlab
- baza de matrice Mathcad
- sheet Excel
- baza de date relațională
- baza de obiecte

Utilizarea mai multor tipuri de baze de date în Inferno este impusă de tipurile de aplicații la care se poate folosi acesta. Pentru problemele de simulare, matricele și vectorii sunt structuri de date de neînlocuit.

Dacă <Atribut> desemncază o matrice de forma cîmp [intreg1 intreg2] unde cîmp este un simbol, Inferno va căuta valoarea atributului respectiv în baza de date matriceală curentă. Dacă în baza matriceală curentă există matricea respectivă Inferno accesează elementul de tablou [intreg1 intreg2]. In cazul cînd matricea nu există în baza curentă sau indicii [intreg1 intreg2] sunt în afara limitelor matricei Inferno consideră că predicatul este fals. Baza matriceală curentă este comunicată lui Inferno prin selectarea cîmpului Dbase din meniul File. Dacă nu există nici o bază matriceală curentă Inferno consideră fals predicatul respectiv.

Vectorii coloană sunt referiți în Inferno prin nume [linie,1] iar vectorii linie sunt referiți prin nume [1,coloana]. Numerotarea liniilor și a coloanelor se face începînd cu 1.

O bază matriceală poate fi construită în mai multe feluri:

- utilizînd produsul Matlab (instrucțiunea Save);
- utilizînd produsul Mathcad (instrucțiunile Write, Writeprn);
- utilizînd un editor de texte.

În ultimile 2 cazuri, după ce s-a construit pentru fiecare matrice cîte un fișier ASCII, fișier în care liniile matricei sînt delimitate prin caracterul '\0' se utilizează programul 'ADD' pentru a memora matricele intr-o bază de date matriceală. Linia de comandă pentru apelul programului ADD este:

ADD <Matrice> <Baza matriceală>.

Dacă <Atribut> desemnează o celulă Excel de forma: (intreg1 intreg2) unde cîmp este un simbol, Inferno accesează în directorul curent sheetul cîmp WK1. Dacă acesta nu există sau dacă în celula (intreg1, intreg2) nu se găsește un număr real, Inferno consideră predicatul respectiv fals. Numerotarea celulelor se face începînd cu cifra 0. De remarcat că, în versiunea actuală se admit numai celule de tip real.

Dacă <Atribut> desemnează un slot dintr-un obiect: cîmp1.cîmp2.cîmp3 unde cîmp1, cîmp2, cîmp3 sunt formați dintr-un singur simbol atunci Inferno accesează o bază de obiecte în care cîmp1 este numele unui obiect,cîmp2 este o instanțiere a obiectului respectiv iar cimp3 este un slot. Pentru a lucra cu o bază de obiecte, utilizatorul trebuie să scrie in limbaj C funcția:

```
void obj(char *Obiect,char *Instantiere,char *slot,
         char *Predicat,char *Valoare,int Retur).
```

Funcția obj trebuie să găsească valoarea slotului respectiv, să o compare cu Valoare utilizînd predicatul <Predicat>; dacă comparația succede, se pune 1 în Retur iar în caz contrar se pune 0.

Dacă <Atribut> desemnează un atribut dintr-o bază de date relațională: cîmp1_cîmp2_cîmp3 unde cîmp1,cîmp2,cîmp3 sunt formate dintr-un singur simbol, atunci Inferno accesează o bază de date relațională unde cîmp1 este numele unei relații,cîmp2 este un index iar cîmp3 este un atribut al relației respective. Pentru a lucra cu o bază de date relațională utilizatorul trebuie să scrie în limbaj C funcția:

```
void rel(char *Relatie,char *Index,char *Atribut,
         char *Predicat,char *Valoare, int Retur).
```

Funcția 'rel' trebuie să găsească valoarea atributului respectiv, să o compare cu <Valoare> utilizînd predicatul <Predicat> și să puna în variabilă <Retur> 1 sau 0 după cum comparația succede sau nu.

Cele 2 funcții trebuie scrise cu opțiunea LARGE, iar legarea acestora cu Inferno se face utilizînd Link.bat.

Dacă <Atribut> nu are nici una din structurile de mai sus, Inferno consideră că valoarea atributului respectiv se găsește în baza sa internă. Pentru atributele care nu sunt memorate în baza de date Inferno, se consideră factorul de validitate 1.

Interfața utilizator

Inferno pune la dispoziția utilizatorului o interfață bazată pe meniuri, forme, ferestre, stări.

Meniul principal este un meniu linie ce include următoarele părți:

- File
- ruLe
- Node
- Run
- Option
- Goal
- Help
- Explanation

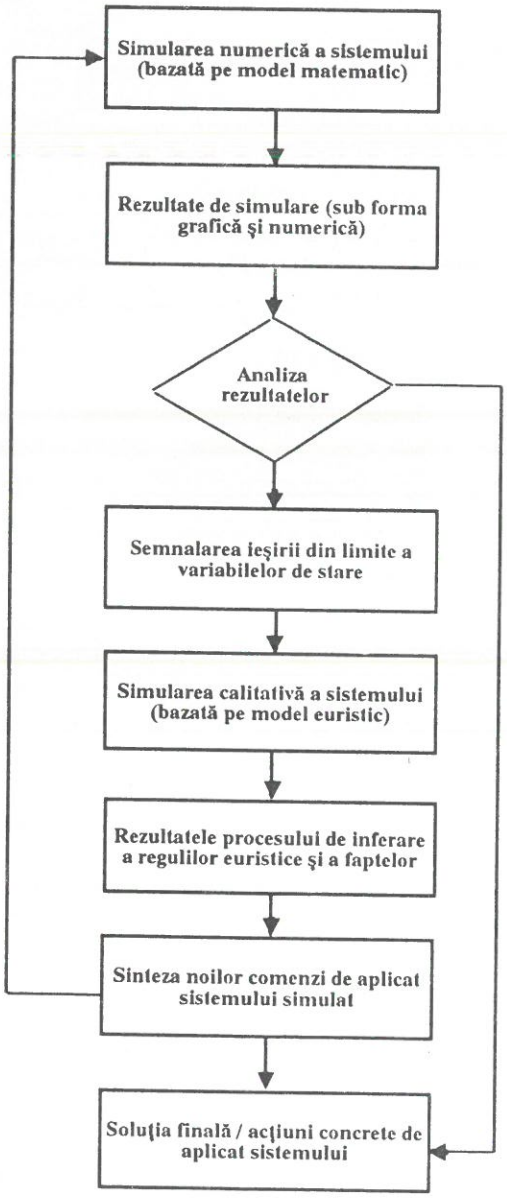


Figura 2. Schema experimentului prototip sistem pentru modelarea, simularea și controlul bazat pe cunoștințe

Resurse necesare

Resursele necesare produsului Inferno sunt următoarele:

- 640 Kb memorie internă;
- 400 Kb memorie externă pentru task-ul Inferno și fișierele atașate;
- memorie externă/internă pentru baza de cunoștințe,volumul acesteia depinzînd de numărul de reguli din baza de cunoștințe;
- memorie externă pentru fișierele de reguli .

De remarcat faptul că Inferno permite memorarea cunoștințelor într-o bază de cunoștințe situată pe disc sau în memoria internă.

Din testele făcute rezultă că memorarea bazei de cunoștințe în memoria internă duce la viteza de execuție mult mai ridicată față de cazul în care baza de cunoștințe este memorată pe disc(a se vedea rezultatele experimentelor și timpii măsurați, in [1]).

6. Experimente realizate și rezultate

6.1. Experimente realizate asupra prototipului

Modelul de simulare utilizat în experiment.

Testarea prototipului sistemului pentru modelarea, simularea și controlul bazat pe cunoștințe s-a făcut cu ajutorul unui model de simulare matematico-euristică, cu date reale (cazul unui sistem hidrologic din Delta Dunării compus din 6 lacuri interconectate prin canale și șenale). Cu acest prilej au fost testate:

- modelul de simulare matematico-euristică,
- algoritmul de simulare HYBRID,
- programul de simulare numerică (utilizînd limbajul de simulare Mathcad),
- produsul-program de tip shell, Inferno, pentru achiziționarea cunoștințelor, construcția bazei de cunoștințe pentru simulare și inferarea cunoștințelor.

Așadar, testarea s-a făcut cu ajutorul unui model de simulare hibridă (matematico-euristică), compus din (pentru detalii asupra modelului de simulare numerică a se vedea [1], [10]):

- un sistem de 6 ecuații de stare (ecuații diferențiale cu timp-discret, neliniare), cu condiții inițiale și date reale,
- 24 formule de calcul (funcții neliniare),
- 6 restricții nelineare (de tip dublă inegalitate),
- 510 reguli euristice (de comportare, de control și de decizie),
- 25 de fapte.

Precizăm că modelul de simulare include: 30 de variabile de stare, cîte 5 pentru fiecare lac: nivelul apei, adîncimea apei, volumul apei acumulate, factorul de împrospătare a apei și debitul de ieșire; 12 variabile de comandă, cîte două pentru fiecare lac: adîncimea canalului de intrare (a) și gradul de acoperire cu vegetație (f) (ambele influențează debitul de intrare al apei în lac).

Experimentul a decurs conform schemei din figura 2.

Baza de cunoștințe utilizate în experimentarea prototipului

Pentru experimentarea prototipului sistemului pentru modelare, simulare și controlul bazat pe cunoștințe-HYBRID, s-a utilizat o bază de cunoștințe compusă din reguli euristice (de comportare, de control și de decizie), scrise cu ajutorul limbajului Prolog, sub Inferno (un sistem expert shell, descris în alineatul 5.2). De subliniat faptul că, baza de cunoștințe include 510 reguli și că aceste reguli utilizează o logică temporală (dinamică, secvențială), pentru a descrie cunoștințele de tip expert la fiecare pas al simulării.

Așadar, în scrierea acestor reguli apare și pasul de simulare (momentul de timp discret, k=0,1,...,n; în cazul nostru n=10). Așa cum am arătat și în capitolul 3, baza de cunoștințe conține 3 specii de reguli euristice, pe care în continuare le vom particulariza în cazul unui sistem hidrologic real, compus din 6 lacuri interconectate prin canale și șenale (este un sistem hidrologic din Delta Dunării, care prezintă un interes special pentru hidrologi și ecologi - a se vedea și figura 3).

Baza de fapte experimentală

Pentru a testa prototipul sistemului pentru modelarea și simularea bazată pe cunoștințe s-au introdus în baza de fapte, un număr de 24 de fapte:

Cod faptă	Faptă
F1	a1 = stationary
F2	a2 = stationary
⋮	⋮
F10	a10 = stationary
F11	f1 = stationary
⋮	⋮
F14	f4 = stationary
F15	f5 = increase
⋮	⋮
F18	f8 = increase
F19	f9 = stationary
F20	f10 = stationary
F21	rough = increase
F22	h = decrease
F23	K = stationary
F24	h = stationary

Baza de date

Așa cum s-a specificat și în alineatul 5.2, programul Inferno-versiunca 1.0 este capabil să caute într-o bază de date matriceală/vectorială și să stabilească dacă un element al unei matrice (sau vector) este >, ≥ , =, < , ≤, decît o anumită valoare sau dacă este cuprins în intervalul [a, b]. În cazul de față este nevoie să se stabilească dacă elementele w [3 k] și q [3 k] ale matricelor w și q și elementele p [k 1], e [k 1] și h [k 1] ale vectorilor p, e și h satisfac anumite condiții, (la orice pas k). De aceea în baza de date a sistemului prototip s-au introdus matricele w și q, rezultate din experimentul de simulare și vectorii p, e și h din modelul de simulare, care pot fi consultate/utilizate de către shell-ul Inferno. De asemenea rezultatele sintezei noii variabile de control s-au introdus în baza de date și au fost utilizate de către programul de simulare.

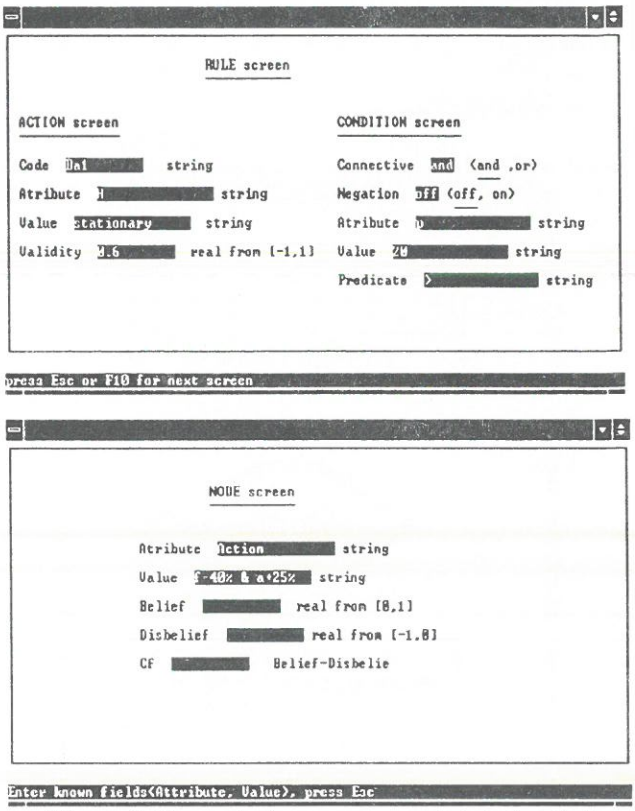
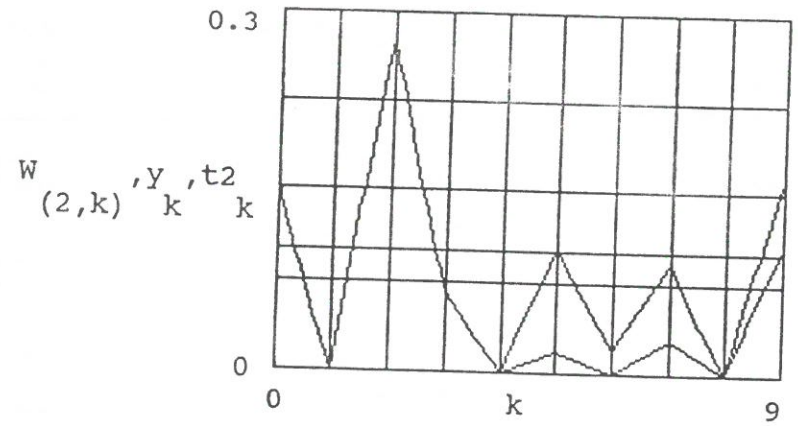


Figura 3. Exemple de ecrane completate (sus) și acțiuni rezultate din inferarea cunoștințelor (jos)

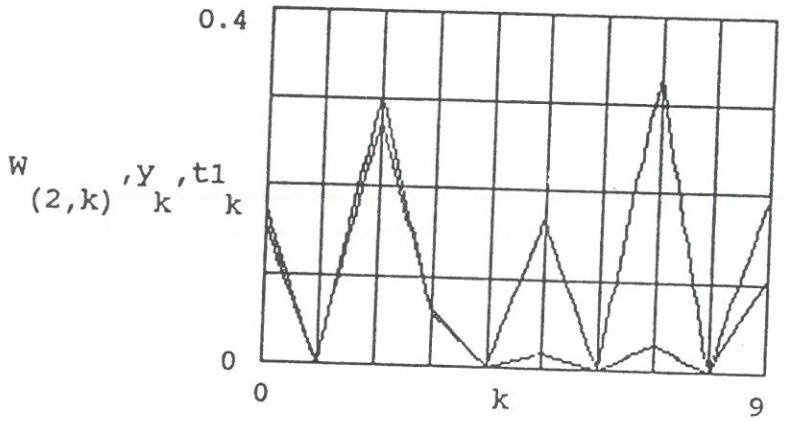
6.2. Rezultate obținute

Experimentele efectuate asupra prototipului sistemului de modelare, simulare și control HYBRID, au confirmat, așa cum vom arăta în continuare, justețea soluției alese. Experimentele care au avut la bază modelul expus în aliniatul 6.1 au decurs în mai multe etape și anume:

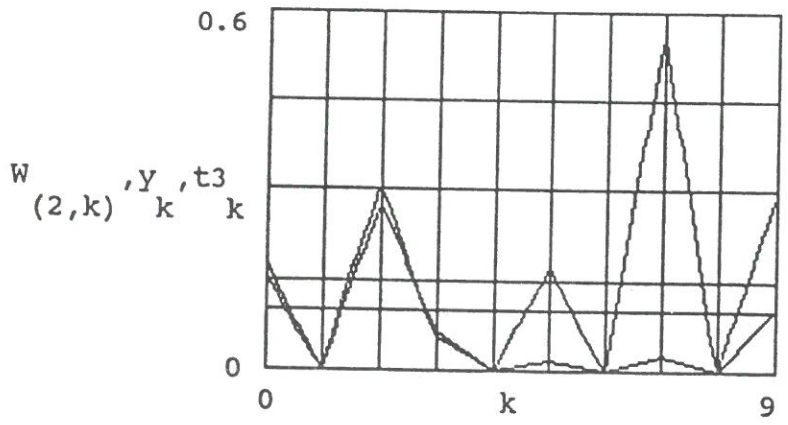
Etapa 1: s-a activat fișierul Deltas 3, care conține modelul de simulare numerică, cu date reale. Programul de calcul a fost rulat, iar rezultatele de simulare au fost prezentate, atît grafic (în 2D), cît și numeric, acestea din urmă fiind stocate în fișierul Baza (sub forma matriceală sau vectorială). Experimentul s-a efectuat pe durata de 1 an, cu pasul de simulare 1 lună (k = 1 lună), datorită faptului că datele reale, obținute de la hidrologi (ecologi), reprezintă medii lunare. Aceasta explică și faptul că, pe graficele obținute, curbele nu sunt suficient de netede; acest lucru însă nu afectează experimentul, curbele respective putînd fi îmbunătățite cu ajutorul unor date, cu o frecvență de culegere mai mare (ex. zilnic).



a) Efectul diminuării factorului f cu 40% (k=5)



b) Efectul diminuării factorului f cu 50% (k=5) și creșterea lui a cu 25%



c) Efectul diminuării factorului f cu 40% (k=5, 6, 7) și creșterea lui a cu 25%

Figura 4. Grafice prezentînd rezultate ale experimentului de simulare folosind prototipul sistemului HYBRID: evoluția naturală a mărimii de stare W(2,k) (curbele inferioare) și evoluția modificată a W(2,k), urmare a modificării unor mărimi de comandă, evaluate cu ajutorul prelucrării cunoștințelor din baza de cunoștințe (curbele superioare)

Mai importantă ni se pare precizarea faptului că expertul este interesat, în acest experiment, de evoluția factorului W(i,k) (factorul de împrospătare a apei lacurilor, i = 0, 1, 2, ..., 5; k=0,1,2,...,n), fapt pentru care acesta a constituit și obiectivul principal al testului.

Etapa a 2-a: din examinarea graficului W2k (reprezentînd factorul de împrospătare a apei în lacul nr.2), se constată că acesta se află pe perioada k = 5 - 8 (corespunzătoare lunilor mai-august) sub limita inferioară W1 = 0.1! (a se vedea și figura 4).

Concluzia firească este că, în acest caz, factorul de decizie (managerul, expertul) trebuie să ia o hotărîre și să acționeze în concordanță cu aceasta. Acțiunea respectivă urmează să fie fundamentată cu ajutorul sistemului HYBRID astfel:

Etapa a 3-a: pentru a determina acțiunea ce trebuie efectuată și a evalua modificările ce trebuie aduse unor variabile (factori) de comandă, în sensul readucerii mărimii de stare W2k (k = 5-6) în intervalul de suboptimalitate [0.1, 1], s-a apelat shell-ul Inferno, precum și următoarele fișiere:

- fișierele care conțin baza de reguli: Prot 1, Prot 2,..., Prot 10 (în total: 51x10 = 510 reguli euristice);
- fișierul Fapt (care conține 24 de fapte);
- fișierul Baza (care conține rezultatele de simulare ale mărimii de stare Wik, i = 0,1,...5; k = 0,1,...,11, sub forma matriceală, precum și alte date vectoriale).

După încărcarea fișierelor amintite, s-a dat comanda Run (timpii de compilare și execuție ai programului au fost măsurați). După rularea programului Inferno și prelucrarea cunoștințelor de tip expert, sub forma regulilor euristice (în mod forward), sunt afișate regulile care au fost activate și acțiunile alese, pentru a continua experimentul de simulare. Aceste acțiuni au fost următoarele:

Acțiunea nr.1: a5 + 25%; f5 - 50% (0.75) (această acțiune se traduce astfel: mărește factorul a cu 25% și diminuează pe f5 cu 50% la pasul 5);

Acțiunea nr.2: a7+25%; f7-50% (0.75)(cu aceeași semnificație, dar la pasul 7);

Acțiunea nr.3: a = a3, f= f3 (0.70) (parametrii a și f rămîn aceeași, cu cei de la pasul 3).

Așadar, sistemul propune factorului de decizie două acțiuni efective - 1 și 2 - ambele cu același grad de încredere (0.75), dar la pași diferiți (k=5 și respectiv k=7).

Programul Inferno, în funcție de regulile euristice incluse în baza de cunoștințe, de faptele din baza de fapte și de rezultatele de simulare anterior obținute, din fișierul Baza, selectează una, două (uneori chiar 3) acțiuni de urmat, factorul de decizie avînd a alege pe cea sau cele, care conform expertului are cele mai multe șanse să realizeze dezideratul acestuia. În experimentul de față am presupus că expertul va aplica acțiunea nr.2, pentru considerentul că ea rezolvă o situație mai gravă (W=0).

Etapa a 4-a: pentru a verifica dacă acțiunile propuse de programul Inferno sunt corecte, s-a creat un nou fișier (sub Mathcad), rezultat din vechiul fișier Deltas 3, în care s-au introdus noile comenzi (acțiunile specificate la etapa 3-a). Fișierul rezultat a fost numit Deltas 4. Cu ajutorul lui s-a resimulat sistemul experimentat, noile rezultate fiind trasate pe aceleași grafice, cu cele de la etapa 1-a.

Se poate observa o îmbunătățire netă a mărimii de stare W2k (k=5- 8), aceasta ajungînd pînă la valoarea 0.3 și chiar mai mare (ceea ce presupune o valoare a factorului de împrospătare a apei de 30-40% ceea ce este, ecologic vorbind, satisfăcător - a se vedea și figura 4).

Așadar, concluzia este că, în urma experimentelor efectuate, **prototipul sistemului HIBRID, pentru modelarea, simularea și controlul bazat pe cunoștințe, a dat deplină satisfacție**, mai ales dacă ținem seama și de următoarele aspecte:

- modelul de simulare numerică a fost destul de complex și puternic nelinear;
- numărul de reguli a fost mare (510 de reguli euristice);
- timpii de execuție ai programului de căutare în baza de cunoștințe și de inferență au fost rezonabili (maximum 25").

Dacă mai adăugăm și faptul că, prototipul a fost testat pe un model cu date reale, deci care a permis compararea cu date rezultate din măsurători (simulare numerică) și cu previziunile expertului (simulare euristică) care au confirmat experimentul, avem toate motivele să declarăm că prototipul sistemului HYBRID a corespuns acestor teste. Ar mai fi poate de efectuat doar teste, care să ateste dimensiunile maxime ale problemei de modelare - simulare bazată pe cunoștințe, problemă care ar putea fi abordată cu ajutorul sistemului HYBRID.

7. Concluzii

S-a elaborat un **prototip** al sistemului pentru modelarea, simularea și controlul bazat pe prelucrarea cunoștințelor a sistemelor complexe, prototip sistem denumit HYBRID. Denumirea sistemului derivă din faptul că metoda, ca și algoritmul de simulare utilizat, se bazează pe un model de simulare hibrid (matematico-euristic), compus dintr-un model de simulare numerică și un model euristic de simulare.

Justificarea soluției are la bază faptul că sistemele

reale, la care sistemul HYBRID se aplică, sunt sisteme complexe, în care au loc procese nelineare, în simularea unor astfel de sisteme trebuind să se țină seama de incertitudinea structurală și/sau de incertitudinea datorată incompletitudinii cunoștințelor, informațiilor și datelor referitoare la sistemul simulat.

Construirea modelului de simulare numerică, în forma standard (un set de ecuații cu timp-discret, nelineare și formule de calcul) se face utilizînd facilitățile limbajului de simulare Mathcad. Modelul mai conține și restricții de tipul dublă inegalitate, care reflectă dorința expertului ca mărimile de stare să aparțină unor intervale de (sub)optimalitate.

Modelul euristic de simulare este constituit dintr-un set de reguli euristice, derivate din cunoașterea de tip expert (din experiență). Achiziționarea cunoștințelor de la expert, construcția bazei de cunoștințe, ca și prelucrarea cunoștințelor se fac cu ajutorul unui shell denumit **Inferno** (aflat la versiunea V.1).

Întreaga concepție de realizare a prototipului sistemului de modelare și simulare bazat pe cunoștințe, poate fi rezumată astfel: simulare numerică a sistemului analizat — stările sistemului — verificarea apartenenței stărilor la intervalele de (sub)optimalitate — dacă stările (eventual unele) nu aparțin intervalelor date de către expert, atunci se trece la evaluarea, cu ajutorul bazei de cunoștințe, a unei noi comenzi — resimularea sistemului cu noile comenzi — stop!

Algoritmul de simulare hibridă, bazat pe modelul matematico-euristic, are și un suport teoretic, care constă dintr-o **teoremă de compatibilitate**, care arată în ce condiții pot funcționa împreună un model de simulare numerică și un model euristic.

Prototipul sistemului HYBRID este util, nu numai în aplicații de modelare și simulare, ci și în **control**, el fiind capabil de a efectua sinteza comenzii capabilă să readucă stările sistemului în intervalele dorite de către expert.

O importantă parte din munca de cercetare pentru elaborarea prototipului sistem a fost consacrată pentru **experimentarea prototipului** cu date reale și cunoștințe de la expert. Experimentele s-au referit la un sistem (complex) hidrologic, compus din 6 lacuri mari interconectate prin canale și șenale, din sistemul Delta Dunării. Mărimile de stare (nivelul apei, adîncimea apei, volumul apei în lacuri, ca și factorul de împrospătare a apei) au fost modelate cu ajutorul unor ecuații cu timp-discret și formule de calcul din mecanica fluidelor și hidrologie, în total 30 de ecuații și/sau formule de calcul. Modelul euristic a constat din 510 de reguli euristice (de comportare, de control și de decizie). Experimentul de simulare s-a efectuat cu pasul de simulare = 1 lună, pe un interval de timp de 10 luni (și aceasta deoarece datele avute la dispoziție reprezintă medii lunare).

La sugestia unui expert, experimentul decisiv a constat în simularea numerică a celor 5 mărimi de stare pentru toate cele 6 lacuri și la sinteza, cu ajutorul shell-ului Inferno, a noii comenzi capabilă să crească factorul de împrospătare a apei în lacul nr. 2, de la valoarea $W << 0.1$, la valoarea $W > 0.3$ (la pasul $k=5 - 8$, adică pentru lunile mai - august).

Experimentul realizat cu ajutorul prototipului sistem a dat deplină satisfacție, așa cum rezultă din figura 4.

Față de sistemul experimental, prototipul sistem prezintă următoarele îmbunătățiri substanțiale:

- **o versiune îmbunătățită** a shell-ului Inferno (versiunea V1);
- sistemul de prelucrare a cunoștințelor poate lucra acum și asupra **rezultatelor de simulare numerică**, deci poate lucra, nu numai cu reguli calitative, ci și cu elemente de matrice și/sau vectori.

În plus, experimentele și testările efectuate asupra prototipului au fost efectuate cu ajutorul unei baze de cunoștințe conținînd 510 reguli euristice (și nu 50, ca în cazul sistemului experimental).

Cercetările vor continua pe două planuri:

- Cercetarea de perspectivă (fundamentală): dezvoltarea unui **sistem multinivel pentru modelarea, simularea și controlul sistemelor mari, complexe, bazate pe cunoștințe.**

Această direcție este dictată de cerințele sistemelor reale, care sunt sisteme mari, complexe și deci deciziile (ca și controlul) se fundamentează la 3 niveluri: managerial, expert, cercetare. Ca atare, cunoștințele utilizate, ca și procesul de prelucrare a cunoștințelor, trebuie să fie multinivel [11] orientate obiect [2] și să utilizeze tehnici de modelare și simulare hibridă [14].

- Cercetarea aplicativă: testarea și experimentarea, în continuare, a prototipului sistemului HYBRID, în dublul scop: (a) pe de o parte, pentru a trage concluzii de îmbunătățire a prototipului (în vederea finalizării unui produs-program) (b) pe de altă parte valorificarea în noi aplicații (ex. sisteme tehnice).

În plus se mai are în vedere și crearea unui sistem de simulare și control în timp real, în care datele să fie culese cu ajutorul unor senzori, implementați în sistemul simulat și controlat, fie el sistem ecologic sau sistem tehnic (aflat la standul de probe).

Autorii sunt interesați în orice propunere de colaborare, pentru implementarea de aplicații utilizînd sistemul de simulare și control bazat pe cunoștințe, dar și în primirea de semnale privind modul cum cititorii au receptat această lucrare.

Bibliografie

1. **STĂNCIULESCU, F., POPA, V.:** Sistem pentru modelarea și simularea bazată pe cunoștințe. Raport de cercetare, ICI, București, iunie/decembrie 1993.
2. **de SWAAN ARONS, H.:** Object Orientation in Simulation. In: Proceedings of the 1993 European Simulation Symposium, Delft (The Netherlands), pp. 694-699.
3. **CUENA, J.:** Building Expert System Based on Simulation Models: An Essay in Methodology. In: Expert System Applications (L. Bolc and N.J. Coomles, eds.). Springer-Verlag, Berlin, 1988.
4. **ZOBEL, R., N.:** Generic Models for Rapid Product Simulation. In: Proceedings of ESM'94, Barcelona (Spain).
5. **REDDY, Y., V., FOX, M., S.:** KBS: An Artificial Intelligence Approach to Flexible Simulation. Research Report. The Robotics Institute, Carnegie-Mellon University, 1982.
6. **GOTTLOB, G., LACKINGER, F.:** Qualitative Simulation: Deriving Behaviour from Knowledge about Structure and Function. In: EUROSIM-Simulation News Europe, July, 1991, pp.4-6.
7. **CELLIER, F.:** Mixed Quantitative and Qualitative Modelling and Simulation (Advanced Tutorial). European Simulation Symposium, Delft, 1993.
8. **LEIMANN, A.:** Knowledge-Based Systems in Simulation. Trends and Applications. In: Computational Systems Analysis, Elsevier, Amsterdam, pp.39-42.
9. **STĂNCIULESCU, F.:** Knowledge-based Simulation and Control of Large Scale Systems. Methodology and Results. In: Handbook Computational Systems Analysis. Topics and Trends (A. Sydow, ed.), Elsevier, Amsterdam, 1992, pp. 323-342.
10. **STĂNCIULESCU, F.:** Systems Analysis Mathematical-Heuristic Modelling, Simulation and Control. Research Report, ICI, Bucharest, 1990.
11. **OHSUGA, S.:** How Can Knowledge-based Systems Solve Large Scale Problems?: Model-based Decomposition and Problem Solving. In: Knowledge-based Systems, vol.6, nr.1, 1993, pp.38-62.
12. **TRENTELMAN, H., L., WILLEMS, J., C.,** (eds.): Essays on Control: Perspectives in the Theory and its Applications. Birkhäuser Verlag, Boston /Basel/Berlin, 1993.
13. **Mc. FARLANE, A.G.J.:** Information, Knowledge and Control. In: Essays on Control: Perspectives in the Theory and its Applications, Basel, 1993, pp.1-28.
14. **BROCKETT, R., W.:** Hybrid Models for Motion Control Systems. Ibidem, pp.29-54.
15. **DEKKER, L.:** Simulation of Systems. In: Simulation Practice and Theory, nr. 1, 1993, pp.1-3.