R.A.I.R.O.
(7e année, juin 1973, B-2, p. 73-79)

MILADIE(*) UN MINI LANGAGE D'APPLICATION POUR LE TRAITEMENT NUMERIQUE D'EQUATIONS DIFFERENTIELLES ET DE RECURRENCE

par A. PAVÉ(1) et J. D. LEBRETON(1)

Résumé. — *La mise au point et l'utilisation de modèles de situations biologiques a amené les auteurs à traiter de façon purement numérique des équations différentielles et des équations de récurrence. A l'utilisation de procédures en langage évolué (ALGOL 60 ou FORTRAN IV), peu souples d'emploi, a été préférée la construction d'un programme conversationnel admettant en entrée des paramètres formels (expressions arithmétiques) représentant les parties droites des équations.*

INTRODUCTION

Parallèlement au développement de langages de programmation très évolués (PL/1 et bientôt ALGOL 68), se multiplient des *langages d'application* plus restreints que les Anglo-saxons désignent sous le nom de « Problem-oriented language » ou « Problem-defining language » destinés à la résolution d'une classe de problèmes bien délimitée [3] [4] [12].

La formalisation de phénomènes biologiques, débouche très souvent sur un des deux types de modèles suivants :

— Équations différentielles ou intégro-différentielles (modèles à compartiments, cinétiques chimiques et biochimiques, modèles intégro-différentiels de croissance d'organismes ou de populations...).

— Équations de récurrence (modèles de dynamique et de génétique des populations en « temps discret », processus de ramification, chaînes de Markov,...).

(*) MIni LAngage D'Intégration et d'Evaluation.

(1) Laboratoire de Biométrie, Université Claude Bernard, Lyon I.

Les équations obtenues sont rarement linéaires et n'admettent donc pas de solutions mathématiques, aussi est-on amené à les résoudre par voie numérique.

La largeur du champ d'application nous a amenés à réaliser, pour éviter la réécriture fastidieuse de programmes en langages évolués, à réaliser un programme conversationnel de résolution numérique de ces deux types d'équations. Cette démarche reflète des besoins réels en moyens de calculs qui devraient se traduire, à notre avis, par la multiplication de ce genre de travaux réalisés par des chercheurs non spécifiquement informaticiens.

De fait, nombreux sont les bio-mathématiciens qui se sont déjà penchés sur le problème de la résolution numérique de systèmes différentiels; malheureusement la communication avec l'ordinateur a été trop souvent particularisée ou sophistiquée à souhait, ce qui enlève une bonne part de leur intérêt à certains de ces langages :

— Restrictions aux équations de la cinétique chimique dans le langage de Garfinkel [5] [6].

— Passage par des opérateurs analogiques dans CSMP/1130, ce qui aux yeux du biologiste paraît un moyen de communication particulièrement artificiel [1] [2] [4].

— « Continuisation » inutile de processus discrets, dont la résolution (cachée à l'utilisateur) se fera par pas (King et Paulik utilisant Dynamo [8]).

Plutôt que de multiplier les intermédiaires, nous avons donc préféré nous en tenir au formalisme mathématique habituel, et nous avons adjoint, sans grande difficulté, la possibilité de traiter des systèmes discrets qui sont fort utiles en dynamique des populations par exemple.

Enfin, la nécessité d'une construction conversationnelle s'est d'autant plus imposée à nous que la programmation sur un petit ordinateur en était relativement commode : les méthodes d'estimation donnent le plus souvent des ordres de grandeur des paramètres plus que des valeurs, et il est primordial de pouvoir à tout instant recommencer le calcul après modification des valeurs de certaines variables ou des valeurs initiales.

I. STRUCTURE DU PROGRAMME. FONCTIONNEMENT

Le programme, baptisé MILADIE (MIni LAngage D'Intégration et d'Evaluation) permet de résoudre numériquement :

— des systèmes différentiels

$$\frac{dY}{dt} = F(t, Y)$$

où Y représente un vecteur de fonctions à évaluer, et t la variable indépendante ;

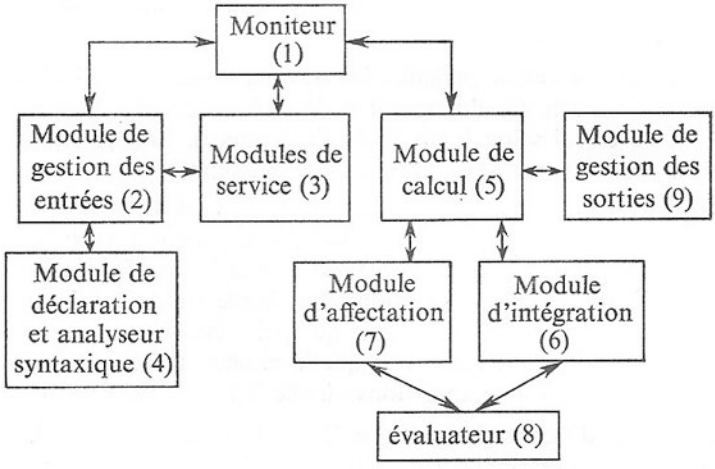
— des équations de récurrence

$$Y_t = f(t, Y_{t-1}, Y_{t-2}, \ldots Y_{t-p})$$

parmi lesquelles se rangent notamment des évaluations par pas de solutions d'équations différentielles.

Pour faciliter la réalisation et pour permettre l'écriture de versions plus performantes sur d'autres ordinateurs (en particulier au Centre de Calcul Inter-Universitaire de Lyon-Saint-Étienne), le programme a été conçu de façon modulaire. Les différents modules sont appelés successivement par un moniteur qui assure la communication avec l'utilisateur selon les ordres fournis par ce dernier.

Le schéma ci-après détaille son architecture



1) Le moniteur traite les ordres fournis par l'utilisateur, c'est par son unique intermédiaire que l'accès dans les différents modules est assuré. En outre, il prend en compte une interruption externe (« intervention opérateur ») qui permet de stopper un calcul en cours, par pression sur une touche du pupitre prévue à cet effet (elle joue le rôle de la touche « ATTENTION » en APL, par exemple).

2) Le module de gestion des entrées utilise un analyseur lexicographique par l'intermédiaire duquel passe toute communication provenant du clavier.

3) Les modules de services forment un ensemble de sous-programmes accessibles, chacun par un ordre moniteur ; ils permettent l'initialisation des fonctions, le choix d'un intervalle de calcul, l'affectation de valeurs numériques aux paramètres, le choix des sorties ainsi que leur mise en forme, l'impression

de messages d'erreurs détectées au niveau de l'analyse syntaxique ou de l'exécution... (10).

4) Déclaration et analyse syntaxique

Pour augmenter la rapidité de l'analyse et faciliter la gestion des identificateurs, les fonctions (partie gauche des expressions arithmétiques) ainsi que la variable indépendante doivent être déclarées, les identificateurs correspondants sont rangés en tête de la table des identificateurs, ils peuvent être employés en partie droite.

L'analyseur syntaxique transforme en chaînes post-fixées les expressions arithmétiques transmises par l'utilisateur, les identificateurs non déclarés constituent les paramètres des équations, leur première occurence constitue leur déclaration implicite (avec initialisation à zéro). La chaîne post-fixée obtenue est une séquence d'adresses d'opérandes et d'adresses d'opérateurs (chaque opération est programmée) l'adresse d'opérateur est affectéee d'un drapeau pour la distinguer des opérandes (on utilise le bit de plus fort poids du mot correspondant).

5) Le module de calcul enchaîne les travaux concernant l'intégration ou l'évaluation. Il teste la fin du travail et les fréquences des sorties (tous les 1, 2... n pas de calcul selon le choix de l'utilisateur). Lorsqu'une sortie doit avoir lieu il appelle le sous-programme correspondant.

Quelle que soit l'option choisie, les évaluations des parties droites sont réalisées vectoriellement : après évaluation des expressions arithmétiques, les résultats sont rangés dans un tableau transitoire et le transfert dans le tableau des valeurs — dans le cas d'une évaluation simple — ou l'utilisation par la procédure d'intégration, n'est réalisé qu'après évaluation de la dernière expression arithmétique, de façon à ce que le résultat de l'une ne puisse modifier à ce pas les valeurs des expressions où elle figure en partie droite.

6) Le module d'intégration est appelé par le module de calcul. Actuellement, nous avons programmé une méthode de Runge-Kutta d'ordre 4 et de rang 2. Nous prévoyons, dans une version ultérieure, de mettre à la disposition de l'utilisateur une gamme de modules d'intégration dans laquelle il pourra choisir la méthode qui lui paraîtra la plus appropriée.

7) Le module d'affectation effectue le transfert des valeurs calculées du tableau transitoire au tableau des valeurs après évaluation de la dernière expression arithmétique (*cf.* module de calcul).

8) Évaluateur : il s'agit d'un évaluateur récursif à partir d'une chaîne post-fixée, qui utilise une pile de valeurs pour le rangement des paramètres et des résultats. En fin de chaîne, le résultat final se trouve en bas de pile. A chaque opération un test de débordement arithmétique est effectué.

9) Le module de gestion des sorties organise les transferts vers l'imprimante suivant une description donnée lors de l'exécution d'un module de service

précédant le lancement du calcul (module de préparation des sorties). Les ordres sont inspirés de FORTRAN [10], l'utilisateur dispose notamment d'une instruction FORMAT, associée à un ordre PRINT, permettant la mise en page et la présentation des résultats. Une procédure de tracé graphique permet de visualiser immédiatement les résultats.

L'utilisateur dispose des fonctions standards habituelles (sinus, cosinus, arctangente, logarithme népérien, exponentielle, valeur absolue, changement de signe), ainsi que d'une fonction logique *signe* : SGN (⟨ Expression arithmétique ⟩), qui joue le rôle d'un comparateur en calcul analogique, en réalisant une modification automatique des systèmes en cours d'exécution. Cette fonction permet d'écrire une expression arithmétique généralisée au sens d'ALGOL 60 :

$$X := \text{'SI'}\ A\ \text{'ALORS'}\ B\ \text{'SINON'}\ C\ ;$$

Elle est à rapprocher aussi de la conversion automatique des variables booléennes en variables arithmétiques en PL/1.

Ce programme a été d'abord réalisé sur le CAE 510 du laboratoire, il occupe 7,5 K mots de 18 bits, cet encombrement s'explique par la présence de toutes les fonctions standards microprogrammées. Une version de ce système existe sur IRIS 50, une autre version sera programmée sur IRIS 80.

II. APPLICATIONS

Par quelques explorations numériques, nous avons pu dresser une liste non limitative des applications de MILADIE en biologie.

1) Système d'intégration

— De nombreux modèles biologiques utilisent des systèmes d'équations différentielles :

. croissance exponentielle, logistique de populations (9),

. relations entre plusieurs espèces animales (9),

. équations de la cinétique chimique, et, plus généralement, modèles à compartiments (5).

— Certains modèles intégro-différentiels peuvent se ramener au schéma précédent, comme celui de la croissance embryonnaire (Kostitzin [9]).

— La résolution par approximation discrète sur une des variables, d'équations aux dérivés partielles (équation de la chaleur...) s'est révélée beaucoup plus rapide et commode qu'en employant un calculateur analogique sur lequel doit être réalisé un câblage fastidieux.

2) Système d'évaluation vectorielle

La résolution de systèmes différentiels par la méthode de la tangente s'écrit très facilement : par ce biais, il est possible d'introduire des retards d'un ou plusieurs pas dans les équations de croissance précédemment mentionnées; on sait que ces extensions se refusent à toute solution analytique. La programmation du retard est explicitement laissée à l'utilisateur en utilisant des inconnues auxiliaires, ce qui évite la difficile réalisation d'une *fonction retard* pour emmagasiner implicitement des valeurs antérieures.

Parmi les plus classiques de la dynamique des populations, les modèles de croissance matricielle (Leslie [11]) ont pu être commodément utilisés. Diverses techniques itératives de calcul matriciel (recherche de valeurs et vecteurs propres) ont été mises en œuvre à cette occasion.

La souplesse et la rapidité du système ont aussi été mises à profit pour la tabulation de lois de probabilités nécessaires à des ajustements.

Enfin, diverses équations de récurrence ont été testées, dont le calcul de la somme de séries.

III. CONCLUSION

Le manque de souplesse des langages de calcul évolués courants, rarement conversationnels (FORTRAN IV, ALGOL 60), est très flagrant : par exemple, une expression arithmétique ne peut être une donnée pour un programme FORTRAN qu'au prix de l'écriture d'un évaluateur lent et lourd...

Au contraire, les langages futurs permettront probablement beaucoup plus facilement l'écriture de programmes pour l'instant réservés aux spécialistes du SOFTWARE. Il sera donc possible d'écrire des mini-compilateurs de langages d'application, pour lesquels peut donc être envisagé, dans un proche avenir, un important développement. L'utilisateur courant disposera alors non plus d'un programme par problème, mais d'un système de calcul par classe de problèmes qu'il programmera à son gré par des ordres très simples.

Actuellement, la construction de tels outils exige encore l'emploi de langages d'assemblage, mais structures et algorithmes préfigurent déjà les analyses futures dont la programmation en « super-langages » sera facilitée. Aussi pensons-nous qu'il est d'ores et déjà souhaitable de promouvoir ces méthodes de résolution pour les problèmes auxquels sont confrontés expérimentateurs et théoriciens, particulièrement en Biologie.

De la première phase de délimitation de la classe de problèmes à résoudre, se déduit l'interface qui permettra à l'utilisateur de communiquer avec la machine : dans notre cas, il s'agit d'un mini-langage conversationnel qui bénéficiera des propriétés syntaxiques simples des langages hors-contexte.

La traduction met en œuvre des algorithmes classiques de compilation (10), dont la programmation est plus ou moins commode : s'il y a évidemment

beaucoup à attendre des « super-langages » en ce domaine, nous devons nous déclarer particulièrement satisfaits du MACRO-ASSEMBLEUR sur CAE-510, qui s'est révélé remarquablement dense et concis.

Par l'emploi de techniques informatiques relativement spécialisées, il est donc possible par un gain de souplesse de mise en œuvre d'élargir le champ des applications immédiates de l'ordinateur à la Biologie.

La mise au point de techniques de SOTFWARE affirmées prend dans cette optique un intérêt tout particulier même pour un laboratoire de Biologie; la programmation des modules de calcul en langage évolué sera un pas important sur une machine disposant d'un éditeur de liens; et l'un des prolongements les plus directs du système MILADIE est la réalisation de programmes de simulation probabilistes pour certains processus couramment utilisés.

Nous insistons aussi sur l'intérêt pour le biologiste des modèles d'évolution discrète, souvent plus commodes d'emploi que leurs homologues continus, mais dont l'extension est récente et parallèle à l'avènement des calculateurs.

BIBLIOGRAPHIE

[1] Binet (P.), Beaugas (C.), Lesieur, Pages (J. C.) et Perrin (P.), *Simulation des systèmes biologiques sur ordinateur par le langage CSMP/1130. Application à la splénoportographie isotopique.* Service développement scientifique, 1968, I.B.M. France. 92-Neuilly.

[2] Brennan (R. D.), Dewit (C. T.), Williams (W. A.) et Quattrin (E. V.), *The utility of a digital simulation language for Ecological Modeling.* Œcologia, 1970, 4, 2, 114-132.

[3] Caracciolo di Forino (A.), *Programming languages.* Adv. in inform. Syst. Sciences, 1969, 1, 59-116.

[4] Chanle (B.), Higgin (J. J.) et Garfinkel (D.), *Analogue and digital computer representations of biochemical processes.* Fed. Procced., 1962, 21, 1, 75-86.

[5] Garfinkel (D.), Rutledge (J. P.) et Higgins (J. J.), *Simulation and analysis of biochemical systems.* I. *Representation of chemical kinetics.* Comm. Assoc. Comput. Machin., 1961, 4, 559-562.

[6] Garfinkel (D.), *Computer simulation in Biochemistry and ecology in theoretical and Mathematical Biology.* Edité par Waterman (T. H.) et Morowitz (H. J.). Blaisdell Publ. Comp., 1965.

[7] Hopgood (F.R.A.), *Techniques de compilation*, Dunod, 1970.

[8] King (L. E.) et Paulik (G. J.), *Dynamic models and the simulation of ecological systems.* J. of Theor. Biol., 1967, 16, 2, 251-267.

[9] Kostitzin (V. A.), Biologie mathématique. Armand Colin, Paris, 1937.

[10] Lebreton (J. D.) et Pavé (A.), *MILADIE.* Manuel d'utilisation (publication interne du laboratoire), 1972.

[11] Leslie (P. H.), *On the use of matrices in population mathematics.* Biometrika, 1945, 33, 183-212.

[12] Pring (M.), *The simulation and analysis by digital computer of biochemical systems in terms of kinetics models.* J. Theor. Biol., USA, 1967, 17, 3, 421-440.