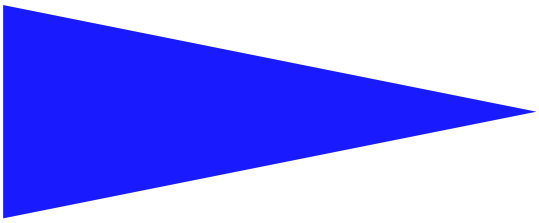
I R I S A
INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTÈMES ALÉATOIRES

P U B L I C A T I O N
I N T E R N E
N° 971



SIMULATION D'UN CIRCUIT ELECTRO-DOMESTIQUE EN SIGNAL

D. CHAUVEAU , M. BONS

ISSN 1166-8687

I R I S A
CAMPUS UNIVERSITAIRE DE BEAULIEU - 35042 RENNES CEDEX - FRANCE

IRISA
INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTÈMES ALÉATOIRES
Campus de Beaulieu – 35042 Rennes Cedex – France
Tél. : (33) 99 84 71 00 – Fax : (33) 99 84 71 71

Simulation d'un Circuit Electro-domestique en SIGNAL

D. Chauveau , M. Bons*

Programme 2 — Calcul symbolique, programmation et génie logiciel
Projet EP-ATR

Publication interne n° 971 — 28 novembre 1995 — 40 pages

Résumé : Nous décrivons dans ce document un simulateur de circuit électro-domestique. Une méthode de conception modulaire qui s'appuie sur les principes du langage SIGNAL, approche synchrone du temps et programmation par équations, est présentée.

(Abstract: pto)

*. Marc Bons effectue une thèse à EDF/DER/SER, 1 avenue du Général de Gaulle F-92141 CLAMART CEDEX, en Liaison avec Bernard Delyon - Projet AS, dans le cadre du contrat EDF/INRIA n° 93 C 285 00 31 301 011. La présente étude est en relation avec ce contrat.

Centre National de la Recherche Scientifique
(URA 227) Université de Rennes 1 – Insa de Rennes

INRIA
Institut National de Recherche en Informatique
et en Automatique – unité de recherche de Rennes

Simulation of a Domestic appliance circuit in SIGNAL

Abstract: In this document we describe a simulator of a domestic appliance circuit. We present a method of modular specification that uses the principles of the SIGNAL language namely a synchronous approach of time and programming using equations.

Table des matières

1 Reconnaissance non-intrusive des appareils électro-domestiques

1.1 Motivation du non-intrusif: la connaissance des usages

La connaissance des usages domestiques présente un grand intérêt pour les compagnies électriques. Elle permet de construire les prévisions de consommation à long terme, qui guident les choix d'investissements des moyens de production et aident à définir les politiques tarifaires et commerciales.

Le modèle prévisionnel sectoriel, actuellement utilisé à EDF, s'appuie sur une prévision des énergies annuelles par secteur d'activité ou par usage. Le transport ferroviaire, l'agro-alimentaire, l'eau chaude domestique à accumulation sont des exemples de ces secteurs et usages. Pour obtenir une prévision de la courbe de puissance totale (au pas horaire), l'énergie de chaque secteur ou usage est ventilée sur les 8760 points horaires de l'année. Pour cela des "formes" de secteurs ou usages ont été obtenues par des mesures, des enquêtes ou des modélisations.

Dans le domaine domestique la connaissance des usages n'est cependant pas suffisante. Pour obtenir des mesures tout en s'affranchissant de la lourdeur des instrumentations complètes appareil par appareil, il a été envisagé de recourir à la mesure non-intrusive. Si cette approche s'avère suffisamment efficace et bon marché, l'information sur les usages qu'elle procure pourrait également être fournie aux clients intéressés par un tel service. Un rapport leur serait alors envoyé avec les consommations et les prix estimés appareil par appareil, agrémentés de conseils sur une meilleure utilisation du produit électrique.

Un attrait partagé pour ce sujet a donné lieu à une collaboration avec Schlumberger Industries. Les objectifs et le principe de l'enregistrement ont été initialement déterminés à EDF [1], tandis que l'élaboration du prototype de l'enregistreur a été réalisée par Schlumberger [2]. Depuis, la collaboration a porté sur le développement du logiciel de post-traitement, qui vise à traiter les données recueillies pour effectuer la décomposition de la consommation [3][4][5].

1.2 Principe de la mesure non-intrusive

Par non-intrusif, il faut comprendre que seule l'installation dans son ensemble est mesurée et non pas les appareils séparément. La scrutation de la tension et du courant permet d'obtenir des grandeurs de puissance dont les variations reflètent le fonctionnement des appareils (pour l'essentiel, ces variations sont dues aux mises en marche et aux arrêts des appareils).

L'enregistreur non-intrusif, connecté au niveau de l'alimentation électrique de l'installation, détecte et enregistre les variations rapides de la puissance appelée, qui serviront de données de base pour les algorithmes de reconnaissance.

1.2.1 Enregistrement des données

Le comportement des charges peut être vu comme une succession de variations rapides de puissance que nous appelons *"événements"* et de périodes à puissance constante, ou variant faiblement, que nous appelons *"états stables"*. Le principe de l'enregistreur en découle : il détecte les événements et enregistre un certain nombre de grandeurs qui leur sont propres. Aucun enregistrement n'est effectué pendant les états stables.

Les grandeurs retenues pour caractériser ces "événements" sont les variations de puissance active et de puissance réactive. On rappelle que la puissance active notée P et la puissance réactive notée Q sont données par les expressions:

$$P = U_e I_e \cos\varphi$$
$$Q = U_e I_e \sin\varphi$$

Avec U_e la tension efficace, I_e le courant efficace et φ le déphasage entre la tension et le courant. Notons que le déphasage est nul dans le cas d'une charge purement résistive, positif dans le cas d'une charge avec une composante inductive, négatif pour une composante capacitive. La figure 1 illustre la détection d'un événement. Les échelles ne sont pas respectées et

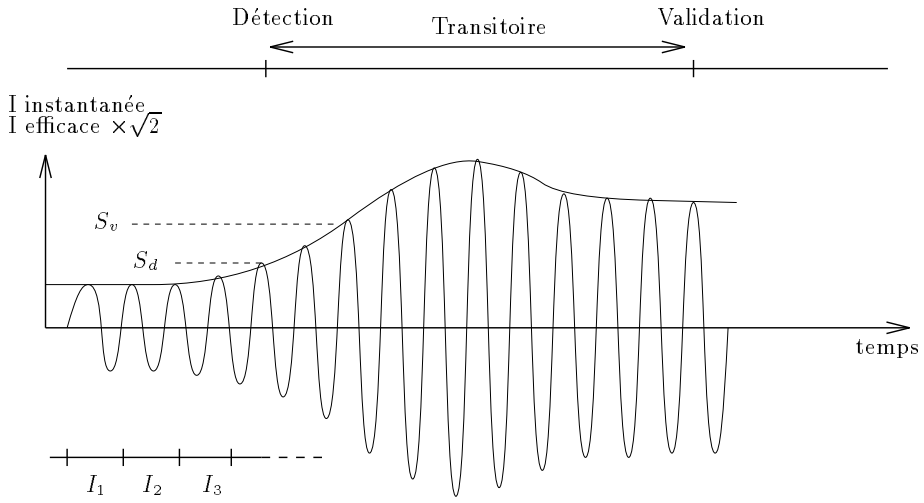


FIG. 1 - *Principe de la détection des événements : Les échantillons $I_1, I_2, \ldots, I_n$ du courant efficace sont comparés successivement. Dès qu'une variation survient, supérieure ou égale au seuil de détection S_d, un début de transition est détecté. Le transitoire est déterminé après un certain nombre d'échantillons stables*

la description du fonctionnement est volontairement grossière (voire fausse) : on s'intéresse

au principe de la détection. Le terme transitoire apparaît pour la première fois ; il désigne la période de transition entre deux états stationnaires, stables du signal, et peut être assimilé à un événement. Les grandeurs enregistrées pour chaque événement détecté et validé sont :

- la différence entre la puissance active avant et après l'événement : ΔP
- la différence entre la puissance réactive avant et après l'événement : ΔQ
- la durée du transitoire.
- la date de l'événement.
- la puissance active totale à la fin de l'événement.
- la puissance réactive totale à la fin de l'événement.

1.2.2 Traitement des données

Observons un exemple d'enregistrement dans le plan $(\Delta P, \Delta Q)$ (figure 2). On peut remarquer une certaine symétrie dans la répartition des points ainsi qu'une distinction entre charges purement résistives (sur l'axe $\Delta Q = 0$) et charges résistives + inductives ($\Delta Q > 0$) qui sont essentiellement des moteurs et des pompes. Nous pouvons également noter la présence de groupes de points d'étendues très variables. Chacun de ces groupes de points constitue une classe qui est censée représenter soit l'ensemble des mises en marche, soit l'ensemble des arrêts d'une même charge (un appareil ou un constituant d'un appareil).

L'aspect temporel des données peut être mis en valeur en représentant soit l'une des grandeurs précédentes en fonction du temps (figure 3), soit une reconstitution de la courbe de charge en fonction du temps pour l'une des deux puissances active ou réactive (figure 4).

Le travail des algorithmes de "post-traitement" consiste à regrouper les événements dus à un seul et même appareil et à reconstituer du mieux possible la courbe de charge correspondante (figure 5). De nombreuses difficultés surgissent, notamment lorsque plusieurs appareils ont des puissances quasiment identiques (ou en tout cas très proches) ou lorsque plusieurs événements se produisent en même temps (c'est la somme algébrique des puissances de ces événements qui est alors enregistrée). La théorie des HMMs permet de modéliser le fonctionnement des appareils (et leurs interactions) et de trouver le "meilleur décodage" susceptible d'expliquer la séquence des données, c.à.d la meilleure répartition des événements selon les appareils de l'installation. La première étape consiste à élaborer des modèles pour les différents types d'appareils existant. Le modèle le plus courant est très simple puisqu'il n'est constitué que de deux états ON et OFF. Une modélisation possible du lave-linge constitue un exemple plus complexe. Au préalable, une phase d'apprentissage doit cependant être entreprise afin d'initialiser le modèle global (les modèles des appareils pris tous ensemble), ce qui consiste à estimer les puissances des appareils.

Enfin les courbes de charge sont extraites en retenant, à l'aide de l'algorithme de Viterbi, la meilleure explication compte tenu du modèle choisi.

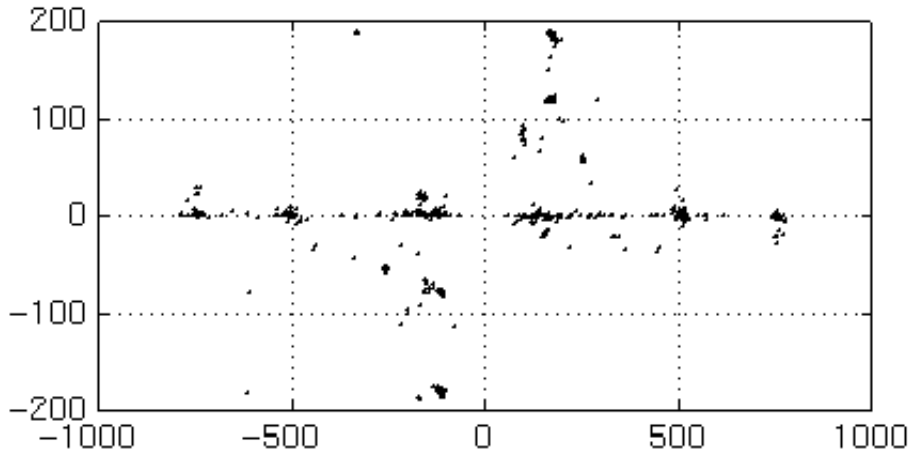


FIG. 2 - *Un exemple d'enregistrement dans le plan* $(\Delta P, \Delta Q)$

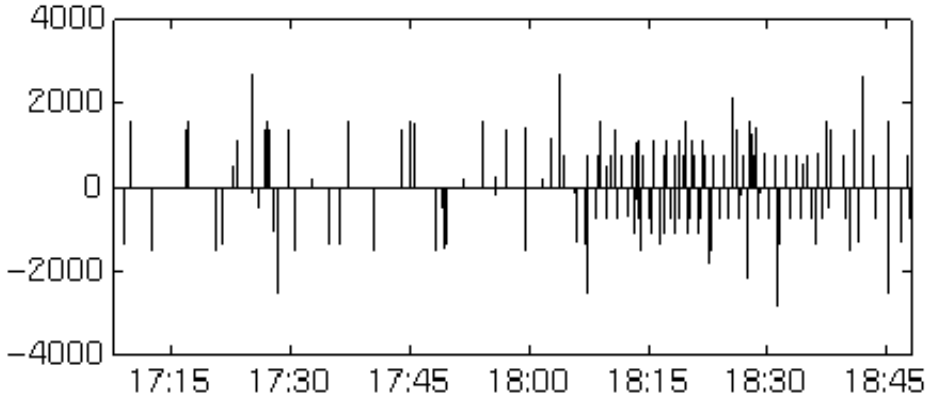


FIG. 3 - *L'aspect temporel des données : Chaque "aiguille" représente un événement à savoir, une variation rapide de puissance*

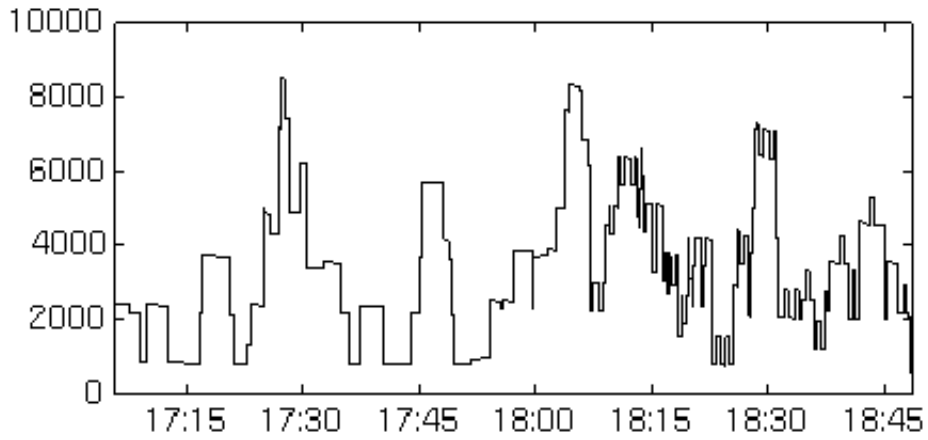


FIG. 4 - *L'aspect temporel des données : la courbe de puissance active en fonction du temps*

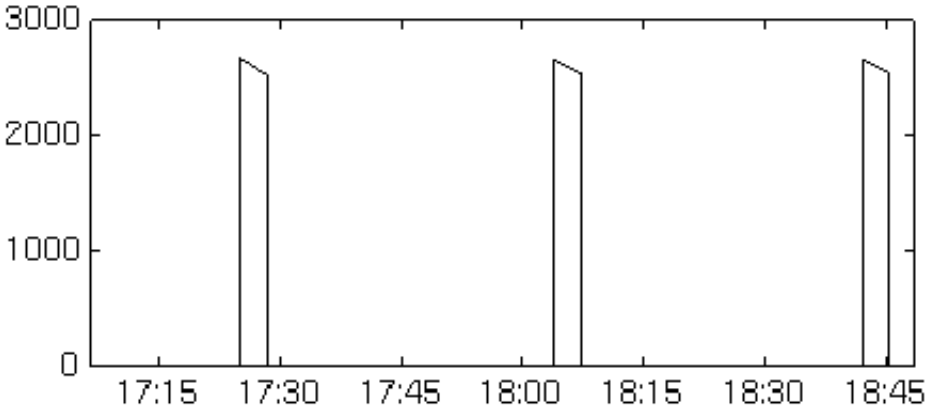


FIG. 5 - *La courbe de charge d'un appareil du circuit*

1.3 Tests et validation des algorithmes de post-traitement

Que ce soit pour comprendre précisément le fonctionnement des algorithmes de reconnaissance et les améliorer, ou pour valider les résultats en terme d'énergie reconstituée, l'utilité d'un simulateur d'installation domestique est incontestable.

Son utilisation s'inscrit dans le cadre de la constitution d'une base de test et de validation, qui fait aussi intervenir des mesures réelles. Les origines des données disponibles sont les suivantes :

1. des enregistrements non-intrusifs d'installations réelles avec, en parallèle, une instrumentation intrusive. L'information intrusive fonctionne cependant sur un principe différent puisque ce sont des énergies moyennes sur des périodes paramétrées à l'avance qui sont enregistrées pour chaque appareil (une période de quelques minutes sera retenue le plus souvent). Outre le dérangement occasionné chez le consommateur, un inconvénient supplémentaire est que la véritable explication (en termes d'événements) n'est pas connue. Cette approche est indispensable pour la validation, mais peut donc difficilement servir pour la mise au point des algorithmes.

2. des enregistrements en laboratoire permettent un meilleur contrôle et une grande liberté d'action. Il faut toutefois souligner la difficulté de créer et mettre en øeuvre des scénarios (sur plusieurs semaines) suffisamment variés, complets et réalistes. On s'attachera donc plutôt, d'une part à recréer des situations difficiles sur des durées plus courtes (comme le fonctionnement simultané de divers appareils) et , d'autre part, à acquérir des "signatures" d'appareils (en fait des enregistrements individuels d'appareils)

3. le simulateur

   - permet la mise au point des algorithmes:
     - en prodiguant une connaissance parfaite des événements générés. L'analyse du comportement des algorithmes peut ainsi se faire événement par événement.
     - en reproduisant certaines difficultés, et ce jusqu'au niveau "microscopique" (événements simultanés par exemple).
   - participe à la validation en générant une grande diversité de scénarios. En particulier, il permet de produire des données hybrides, en mixant enregistrements réels et appareils simulés. L'ajout de convecteurs à une installation réelle sans chauffage est un exemple intéressant de l'utilisation du simulateur.

2 Le langage SIGNAL

Le langage SIGNAL est développé dans le projet Environnement de Programmation pour des Applications temps Réel (EP-ATR) de l'IRISA. Il appartient comme les langages LUSTRE

[6] et ESTEREL [7] à la famille des *langages temps réels synchrones*. SIGNAL est un langage modulaire construit pour la description d'applications temps réel. Les constructeurs de base, ou processus élémentaires, décrivent des équations entre des suites non bornées de valeurs typées désignées par des noms (ou signaux).

La gestion du temps est rendue implicite en associant un index temporel (ou horloge) à tout signal. Cette particularité permet, d'une part d'accepter en entrée du système des signaux d'horloges différentes et, d'autre part, de construire à l'intérieur d'un système une donnée plus fréquente que les données d'entrée. Afin de répondre à la caractéristique des systèmes à temps réel, les opérateurs de SIGNAL ne font pas référence au futur, il n'y a donc pas de problème de causalité, et l'appel récursif de fonctions est prohibé.

2.1 Définition du langage

Les objets de base[1] du langage sont les *signaux* et les *processus*.

2.1.1 Les signaux

Les signaux sont des suites ordonnées potentiellement infinies de *valeurs typées*. À chaque signal est associé une *horloge* qui définit les instants où le signal est présent. La présence ou l'absence d'un signal est toujours déterminée relativement à la présence ou à l'absence d'autres signaux.

Exemple :

X :		2			5	3			1	. . . .
Y :				1	1			4	0	. . . .

Le signal **X** est présent aux instants 1, 3, 4, 6, ..., le signal **Y** aux instants, 2, 3, 5, 6, ...Ces instants sont relatifs à l'*environnement* considéré (i.e. à l'ensemble des signaux observés) ; il n'y a pas de référentiel de temps global.

Deux signaux qui sont présents aux mêmes instants pour tout environnement sont dits *synchrones* ; une classe d'équivalence de signaux synchrones est appelée l'horloge de ces signaux. On distingue deux types de signaux : les signaux d'entrée et les signaux de sortie.

2.1.2 Les Processus

Tout signal de nom X, hormis les entrées du système, est défini par une expression de la forme $X := E$ où E désigne une constante, une expression sur signaux ou un appel à processus externes. Une telle expression est appelée processus élémentaire.

Les expressions sur signaux sont construites à partir d'un ensemble d'opérateurs de base constituant le langage noyau et ayant pour opérandes un ensemble de signaux constituant les signaux d'entrée du processus élémentaire.

1. On trouvera une présentation complète des fondements théoriques du langage dans [8] et [9]

Les opérateurs sur signaux du langage noyau

- **Les expressions fonctionnelles** permettent d'étendre canoniquement les fonctions définies sur les types du langage aux signaux. Si f est une fonction n-aire, le processus élémentaire

$$Y := f(X1, X2, \ldots, Xn)$$

définit les valeurs du signal Y par $\forall t > 0,\ y_t = f(x1_t, x2_t, \ldots, xn_t)$. Les signaux $X1$, ..., Xn sont disponibles aux mêmes instants, la fonction ne consomme pas de temps, Y, $X1$, ..., Xn sont donc synchrones.

- **Le retard** est un opérateur donnant accès à la valeur passée d'un signal. Si X est un signal quelconque et $v0$ une contante de même type que X alors le processus

$$Y := X\ \$1\ \textbf{init}\ v0$$

définit un signal Y synchrone au signal X dont les valeurs à chaque instant sont données par : $Y_t = X_{t-1}\ \ t > 0, Y_0 = v0$.

- **La condition :** un opérateur de SIGNAL permet de sous-échantillonner un signal par une condition booléenne : Si X est un signal quelconque et B un signal booléen alors le processus élémentaire

$$Y := X\ \textbf{when}\ B$$

définit un signal **Y** dont les occurrences sont les occurrences de **X** simultanées à une occurrence *vraie* du signal booléen **B**.

- **La fusion** est le dernier opérateur de base de SIGNAL. Il fusionne, de manière déterministe, deux signaux de même type :

$$Z := X\ \textbf{default}\ Y$$

La valeur de **Z** est égale à la valeur de **X** quand **X** est présent, à la valeur de **Y** quand **X** est absent et **Y** présent. **Z** est par conséquent plus fréquent que **X** et que **Y**.

Expressions fonctionnelles et *retard* expriment des équations entre signaux synchrones : ce sont des opérateurs monochrones. *Condition* et *fusion* décrivent des relations entre signaux d'horloges différentes et appartiennent à la famille des opérateurs polychrones.

Les opérateurs sur processus

- La composition de processus
  Si $P1$ et $P2$ sont deux processus, la *composition* de $P1$ et $P2$ définit un nouveau processus noté :
  $$P1 \mid P2$$
  où les noms de signaux communs à $P1$ et $P2$ référencent les mêmes signaux. $P1$ et $P2$ communiquent par l'intermédiaire de ces signaux communs. L'opérateur de composition ainsi défini est associatif et commutatif, il réalise l'union des deux systèmes d'équations décrits par $P1$ et $P2$.

- Le masquage
  On peut réduire les possibilités de communications d'un processus en masquant des signaux :
  $$P \ / \ a_1, \ldots, a_n$$
  définit un processus identique à P mais dans lequel les signaux $a_1, \ldots, a_n$ sont masqués. Ils restent locaux et ne peuvent pas servir de voies de communication.

- Le renommage
  Nous utiliserons aussi les renommages, qui permettent d'établir des connexions entre processus :
  $$P \ ? \ a : x \ ! \ b : y$$
  est le processus égal à P dans lequel l'entrée a est renommée en x, la sortie b en y. Ce nouveau processus peut ainsi communiquer avec un autre qui possède des signaux de nom x et y.

Les opérateurs dérivés

Autour du noyau de SIGNAL, est défini un ensemble d'opérateurs dérivés. Nous ne citerons ici que ceux qui seront utilisés dans la suite ; pour une description plus complète, se reporter à [10] ou à [11].

- **event** X
  est un signal booléen toujours vrai, de même horloge que $\mathbf{X}$, qui peut donc être assimilé à l'horloge de $\mathbf{X}$.

- X **cell** B
  est la mémorisation de $\mathbf{X}$. Quand $\mathbf{X}$ est présent, sa valeur est produite en sortie de la mémorisation , quand la valeur *vrai* est présente en $\mathbf{B}$ et que $\mathbf{X}$ est absent, la précédente valeur de $\mathbf{X}$ est transmise en sortie.

- **synchro** $X_1, X_2, \ldots, X_n$
  est une *synchronisation explicite*, elle spécifie que les signaux $\mathbf{X}_1$, ..., $\mathbf{X}_n$ ont la même horloge.

Les modèles de processus

Un ensemble de processus, récursivement composé de processus élémentaires ou de processus externes, définit un comportement qui peut être abstrait sous la forme d'un *modèle de processus*. Un programme SIGNAL est alors obtenu par la production d'un exemplaire selon un tel modèle. La déclaration d'un tel modèle associe à un nom :

- Une interface constituée d'un ensemble de paramètres statiques, d'un ensemble de signaux d'entrée et d'un ensemble de signaux de sortie.
- Un corps de processus qui définit son comportement, l'ensemble des signaux locaux et l'ensemble des modèles locaux de processus.

La syntaxe utilisée pour ls déclaration d'un modèle de processus est la suivante :

process *NOM* =
  (*paramètres*)
  {? *signaux d'entrées*
  ! *signaux de sorties* }
    (| *corps du processus*
    |)
where
  déclarations locales
end

Un exemplaire ou *instance* d'un modèle de processus est noté :

NOM(*définition des paramètres*),

et définit un processus identique au corps du modèle, qui ne peut communiquer que par les signaux cités dans l'interface.

- **Les processus locaux**
  Un processus peut être dénoté par un identificateur P dans une déclaration pouvant elle-même comporter des sous-déclarations. Une occurrence de l'identificateur P provoque la *duplication* et l'*insertion* du système qu'il dénote dans le contexte de signaux de cette occurrence.
- **Les processus externes**
  SIGNAL offre également la possibilité d'utiliser des processus externes qui doivent être déclarés. Cette déclaration se limite à l'interface du processus. On peut ainsi utiliser des opérateurs définis dans des bibliothèques scientifiques, programmer de façon impérative certaines parties des algorithmes, utiliser des fonctions d'affichage, ...

2.2 L'environnement de programmation

2.2.1 La représentation graphique des processus

Le langage SIGNAL se prête, par sa modularité, à une représentation graphique directe en blocs-diagrammes. Un éditeur [12] est disponible dans l'environnement SIGNAL qui permet la construction sous une forme mixte (mélangeant graphiques et textes) des programmes (figure 6).

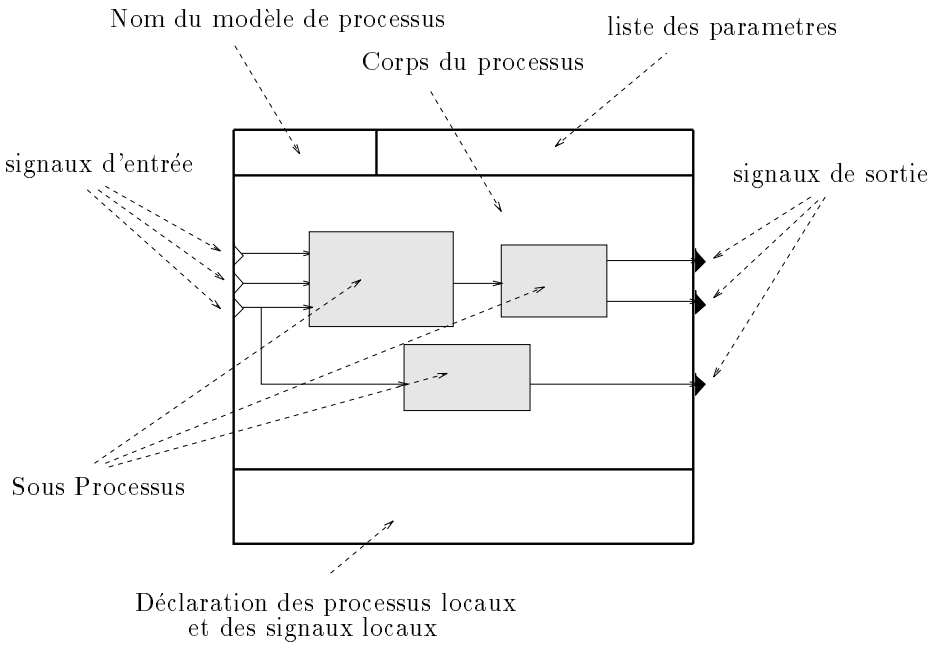


FIG. 6 - *processus signal sous sa forme graphique*

Les signaux d'entrée et de sortie du modèle sont symbolisés par les ports de connexions triangulaires situés sur les bords de la boîte corps de processus. Chaque boîte interne au corps représente une expression de processus qui peut être soit sous forme textuelle, soit construite récursivement à partir d'autres boîtes. Les signaux non masqués de chacune de ces sous expressions sont représentés par des ports de connexion. Les liens entre les différents ports indiquent les voies de communications entre les divers sous processus. Ces liens sont donc l'équivalent graphique des renommages : deux ports reliés par un même lien représentent le même signal.

Un décompilateur permet d'engendrer automatiquement un programme sous forme textuelle équivalent en vue de la compilation.

2.2.2 La compilation des programmes SIGNAL

La compilation des programmes SIGNAL s'effectue en plusieurs étapes [13]. Une première phase "obligatoire" consiste en

1. une analyse syntaxique qui fournit la représentation interne d'un programme source sous forme d'un arbre de syntaxe abstraite.
2. une "résolution", qui transforme l'arbre de syntaxe abstraite en une structure de donnée particulière appelée *graphe hiérarchisé aux dépendances conditionnées*. Cette transformation permet de rendre explicite les définitions et les ordres de calcul exprimés implicitement dans le programme de départ.

Différentes options de compilation sont ensuite disponibles. Suivant les objectifs de l'utilisateur elle permettent :

1. la production d'une représentation polynomiale dans $\mathcal{F}_3$ (le corps des entiers modulo 3) des synchronisation et des contraintes logiques du programme. Cette représentation devant permettre l'étude de propriétés (vivacité, invariance, stabilisabilité, ...) à l'aide d'un outil de preuve spécifique faisant parti de l'environnement SIGNAL [14].
2. la production d'une représentation externe du graphe en SIGNAL en vue d'un éventuel débogage ...
3. la production de code séquentiel afin de simuler l'exécution séquentielle du programme SIGNAL. Actuellement deux codes sont disponibles : fortran 77 et C.

3 Spécification d'un simulateur en SIGNAL

Nous avons décrit en SIGNAL un simulateur de circuits électro-domestiques. Pour modéliser les phénomènes aléatoires et réaliser une partie de l'environnement de simulation, nous avons, par le biais des processus externes, utilisé une bibliothèque de génération de nombres aléatoires que nous présentons brièvement en annexe A. Une interface graphique permet de contrôler le fonctionnement des appareils, de modifier la vitesse d'écoulement du temps et de visualiser à l'écran certains résultats. Cette interface est décrite en MOTIF à l'aide des langages C et MIL. Son contrôle est assuré par le programme SIGNAL par utilisation de processus externes. Le modèle de processus SIMULATEUR (figure 7) décrivant le simulateur dans son ensemble est composé de trois sous processus :

1. Le processus CIRCUIT qui modélise la partie circuit du simulateur avec les différents appareils qui le constituent.

2. Le processus CONTEXT directement relié à l'interface graphique par appel à des processus externes. Il permet de :
   (a) contrôler le temps
   (b) contrôler la mise en fonctionnement et l'arrêt des différents appareils du circuit.
3. Le troisième processus est composé de plusieurs sous processus concernant les résultats de la simulation. Il gère :
   (a) La construction du fichier contenant l'ensemble des événements de la simulation en prélevant sur le circuit l'information nécessaire.
   (b) L'affichage à l'écran du temps sous la forme [*jours,heures,minutes,secondes*]
   (c) L'affichage des courbes de puissance actives et réactives (courbes reconstruites à partir des résultats de mesures simulées) selon deux échelles de temps différentes.

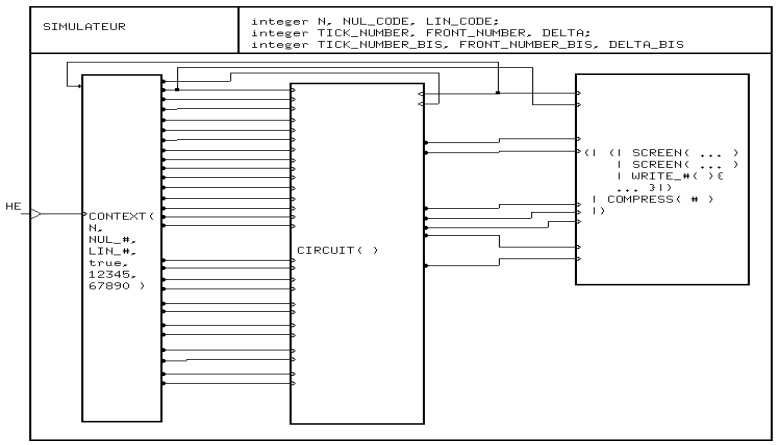


FIG. 7 - *Le modèle de processus SIMULATEUR*

Il est important de préciser ici que nous ne cherchons pas à simuler le comportement réel des différents appareils, du circuit et de l'appareil de mesure. En effet seuls nous intéressent le résultats des mesures et la trace laissée par chaque appareil sur les mesures du circuit total. En conséquence, nous nous contentons de construire

- Un fichier contenant une simulation d'un enregistrement total obtenu à partir d'un appareil de mesure placé au compteur électrique d'une installation sur une période donnée.

– Pour chaque appareil, un fichier qui contient l'enregistrement obtenu à partir d'un appareil de mesure placé sur l'appareil et sur la même période que précédemment.

La description que nous donnons du simulateur est composée de deux parties bien distinctes qui sont :

1. La modélisation effective du circuit électrique avec les différents appareils qui le constituent.
2. L'environnement de simulation qui vise à faire fonctionner le circuit de façon contrôlée et à permettre un traitement sous différentes formes des résultats de la simulation.

3.1 Modélisation du circuit électrique

Le modèle de processus CIRCUIT que l'on peut voir sur la figure 8 est une modélisation du circuit total. Ce processus contient deux types de sous-processus :

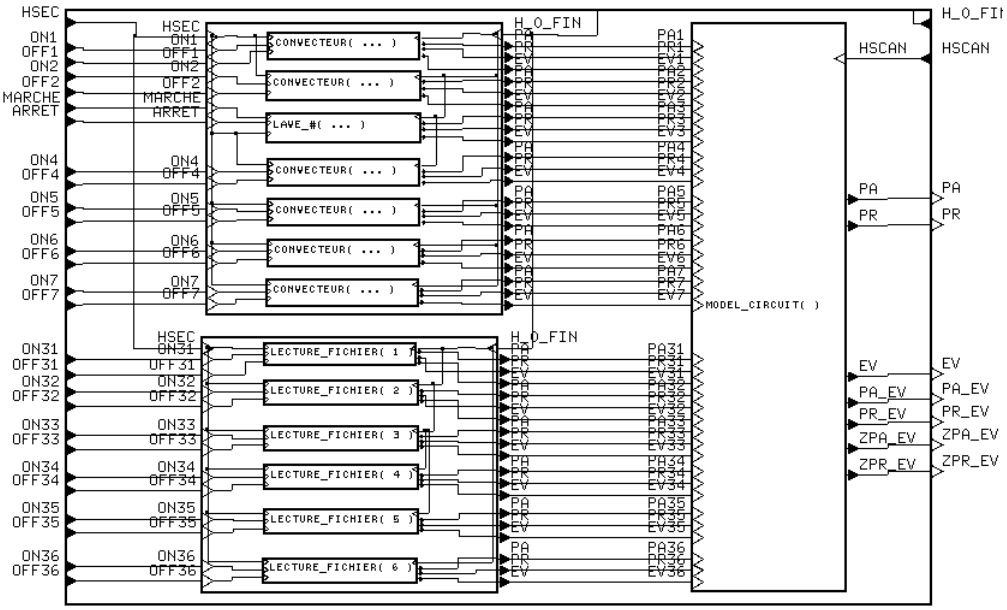


FIG. 8 - *Corps du modèle de processus* CIRCUIT

1. Le processus MODEL-CIRCUIT qui permet de décrire l'ensemble des connexions entre les différents appareils, il se réduit ici à une simple somme sur l'ensemble des appareils des puissances actives et réactives consommées.

2. Un ensemble de processus modélisant des appareils. Ces processus sont ici regroupés en deux classes :

   (a) Les appareils qui sont des instances de modèles entièrement simulés.

   (b) Les appareils dont le fonctionnement est décrit par un fichier.

Les signaux d'entrée sont destinés à contrôler la mise en fonctionnement et l'arrêt des différents appareils. On retrouve également le signal booléen H-0-FIN permettant d'initialiser le simulateur (initialisation de l'interface graphique, ouverture et fermeture des fichiers résultats), le signal HSEC donnant l'horloge associée à la seconde et enfin l'horloge HSCAN qui est l'horloge la plus rapide à laquelle puisse être rendue disponible à l'extérieur du circuit la puissance consommée.

En signaux de sortie on trouve la puissance active et réactive totale (PR-EV et ZPR-EV) consommée par le circuit à une sous-horloge de HSCAN, destinés à l'affichage des courbes de puissance. L'horloge exacte de ces deux signaux est déterminée non pas par le circuit lui-même mais à la demande, par le processus d'affichage. Les autres signaux (EV, PA-EV, ZPA-EV, PR-EV et ZPR-EV) contiennent toute l'information nécessaire à la construction du fichier résultat.

3.1.1 Modélisation des appareils

Les appareils que nous étudions peuvent être modélisés par des machines d'états finis. Ceux pouvant prendre un nombre infini d'états (lampes halogènes, ...) ne seront pas pris en considération, ni ceux dont la puissance varie continûment (régulation de certains moteurs). Nous nous intéressons principalement aux changements d'états (ou transitions) qui sont marqués par une rapide variation de puissance et que nous appelons événement (voir 1.2.1). Nous ne prendrons pas cependant en compte les transitoires que nous supposerons ici de durée inférieure à la seconde. Les caractéristiques prises en compte pour chaque événement seront donc :

- Les puissances active et réactive consommées à la date de cet événement (après passage de la transition).

- Les variations des puissances active et réactive observées lors du changement d'état (variation de puissance entre avant et après le passage de la transition).

Nous imposons que chaque appareil intervenant dans le circuit soit construit sur un modèle bien précis (figure 9). Il devront en conséquence avoir quatre signaux d'entrée qui en permettent le contrôle :

1. un signal H-0-FIN pour l'ouverture et la fermeture de fichiers en lecture pour des appareils décrits par des fichiers ou en écriture pour des appareils entièrement simulés[2].

2. Au cours de la simulation, un fichier contenant l'ensemble des événements propres à l'appareil est construit sur le même principe que le fichier contenant le résultat de la simulation du circuit complet décrit en 3.2.4

2. un signal HSEC pour l'horloge des secondes
3. un signal ON pour la mise en fonctionnement de l'appareil
4. un signal OFF pour l'arrêt de l'appareil

et trois signaux de sortie :

1. un signal EV dont les instants de présence correspondent aux instants de présence d'un événement sur la courbe de charge de l'appareil.
2. un signal PA qui donne la puissance active à toute horloge inférieure ou égale à l'horloge HSEC.
3. un signal PA qui donne la puissance réactive à toute horloge inférieure ou égale à l'horloge HSEC.

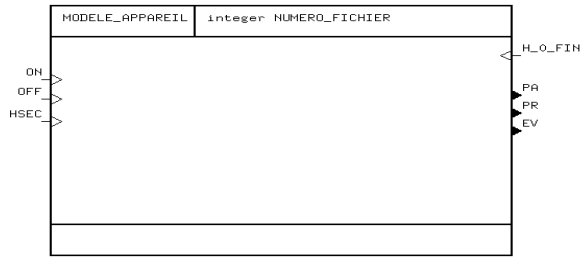


FIG. 9 - *Interface d'un modèle de processus de type appareil*

Les horloges des signaux de puissance active (PA) et de puissance réactive (PR) ne doivent pas être fixés par le processus décrivant l'appareil.

Nous donnons deux exemples de modélisation d'appareils. Le premier est un convecteur qui est un appareil de type (ON, OFF) à deux états. Le second est un exemple un peu plus complexe, il s'agit d'un lave-linge.

Un appareil à deux états : exemple du modèle convecteur

Un convecteur est modélisé par une machine à deux états (figure 10) : un état où il consomme de l'énergie (état ON) et un état où il n'en consomme pas (état OFF). À chaque changement d'état correspond une variation de puissance qui en tout état de cause doit nous ramener à une puissance consommée nulle dans le cas d'une transition ON→OFF.

Des études menées par EDF ont permis de montrer que la distribution des valeurs prises par $(\Delta P_{ON \to OFF}, \Delta Q_{ON \to OFF})$ et par $(\Delta P_{OFF \to ON}, \Delta Q_{OFF \to ON})$ peut être modélisée par

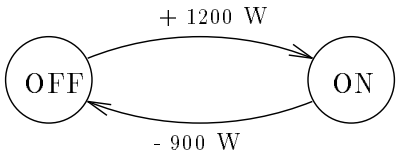


FIG. 10 - *Exemple de machine à états modélisant un convecteur. La variation de puissance associée à une transition* OFF→ON *est de +1200 Watts, celle associée à une transition* ON→OFF *est de -900 Watts*

deux gaussiennes bi-dimensionnelles. Les durées des ON et des OFF sont déterminées au choix de l'utilisateur par une Weybull, une rayleigh ou une inverse gaussienne (voir A.2.2).

La simulation d'un convecteur nécessite l'utilisation de 6 variables aléatoires.

- 1 pour la durée d'un ON
- 1 pour la durée d'un OFF
- 2 pour le couple $(\Delta P_{ON}, \Delta Q_{ON})$ (loi jointe)
- 2 pour le couple $(\Delta P_{OFF}, \Delta Q_{OFF})$ (loi jointe)

Quatre générateurs de nombres aléatoires indépendants sont utilisés pour la simulation de ces 6 variables aléatoires.

Ces données sont suffisantes dans le cas de notre étude à la simulation d'un convecteur. Cependant, dans la simulation du circuit total il est nécessaire de disposer de la puissance consommée à chaque instant. Nous avons choisi de considérer que la variation en puissance entre un ON et un OFF était linéaire en fonction du temps tout en sachant que dans la réalité, il s'agit plutôt d'une décroissance exponentielle. L'approximation reste cependant admissible. Nous obtenons, $\forall i \geq 0$:

$$\forall t \in]t_{ON_i}, t_{OFF_i}[, \quad P(t) = \frac{P(t_{OFF_i}) - P(t_{ON_i})}{t_{OFF_i} - t_{ON_i}}(t - t_{ON_i}) + P(t_{ON_i}) \tag{1}$$

$$Q(t) = \frac{Q(t_{OFF_i}) - Q(t_{ON_i})}{t_{OFF_i} - t_{ON_i}}(t - t_{ON_i}) + Q(t_{ON_i}) \tag{2}$$

$$\forall t \in]t_{OFF_i}, t_{ON_{i+1}}[, \quad P(t) = 0.0 \tag{3}$$

$$Q(t) = 0.0 \tag{4}$$

Il est à noter que nous aurons par ailleurs pour les appareils de type convecteur :

$$P(t_{ON_i}) = +\Delta P_{ON_i} \tag{5}$$

$$Q(t_{ON_i}) = +\Delta Q_{ON_i} \tag{6}$$

$$P(t_{OFF_i}) = -\Delta P_{OFF_i} \tag{7}$$

$$Q(t_{OFF_i}) = -\Delta Q_{OFF_i} \tag{8}$$

L'algorithme qui est mis en application pour modéliser en signal un appareil de type convec-

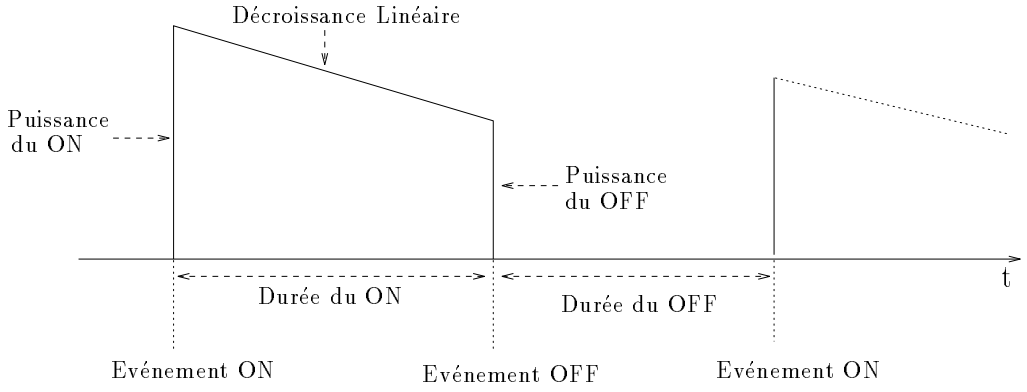


FIG. 11 - *Courbe de puissance du modèle convecteur à deux états*

teur est le suivant :

1. À chaque événement ON générer
   (a) la durée du ON
   (b) le couple $(\Delta P_{ON}, \Delta Q_{ON})$
   (c) le couple $(\Delta P_{OFF}, \Delta Q_{OFF})$
2. Entre les événements ON et OFF la puissance consommée est donnée par (1) et (2)
3. À chaque événement OFF générer la durée du OFF
4. Entre un événement OFF et un événement ON la puissance consommée est nulle.

Un appareil à plus de deux états : exemple du lave-linge
Ne disposant pas de loi de fonctionnement d'un modèle universel de lave-linge, nous avons simplement cherché à simuler des mesures de puissances se rapprochant le plus possible de mesures réelles observées, sans respecter ici de modèle aléatoire précis. Le lave-linge est principalement constitué de trois composants consommateurs d'énergie électrique :

- Un tambour
- Une résistance
- Une pompe

Ces trois éléments que l'on voie apparaître sur la représentation graphique du modèle de processus LAVE-LINGE (figure 12) sont des appareils à deux états de type convecteur et peuvent donc être modélisés sur le même principe. Il sont "pilotés" par un programmateur supposé ne pas consommer de puissance électrique. La puissance consommée par l'électro-vanne est également supposée négligeable.

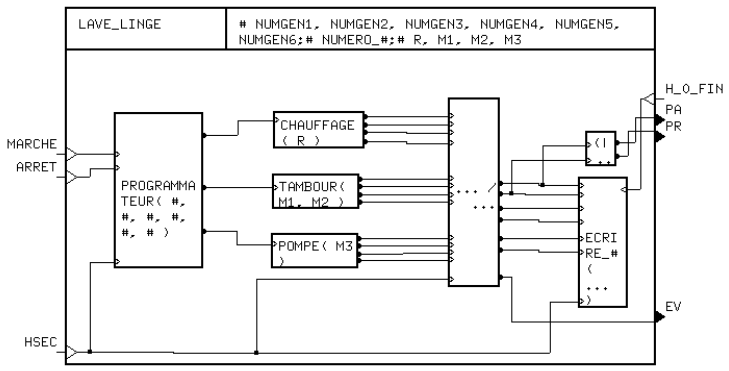


FIG. 12 - *Le modèle de processus LAVE-LINGE*

3.1.2 Description d'appareils par des fichiers

Nous donnons la possibilité d'utiliser dans le circuit des appareils décrits par des fichiers. Nous donnons pour cela un modèle d'appareil particulier, le processus LECTURE-FICHIER dont l'interface contient les mêmes signaux d'entrée/sortie qu'un appareil classique et qui prend en paramètre un fichier de lecture. Trois utilisations peuvent en être faites :

1. EDF a mis en service une structure permettant d'effectuer en laboratoire des mesures précises sur des appareils. Les fichiers ainsi obtenus peuvent être directement utilisés comme modèles d'appareils.

2. Il est également possible de décrire à la main des modèles précis d'appareils avec en particulier des dates précises d'événements ou de modifier des descriptions fichiers existantes.

3. Il est enfin possible d'utiliser les résultats de simulations antérieures comme modèles d'appareils (ce qui était un circuit, devient un appareil dans la nouvelle simulation). On se donne ainsi la possibilité de rajouter incrémentalement de nouveaux appareils.

4. À une installation réelle (par exemple sans chauffage), on peut rajouter des convecteurs simulés.

On peut ainsi aisément mélanger mesures réelles et mesures entièrement simulées, rajouter de nouveaux appareils sur des circuits déjà étudiés, entrer des modèles précis d'appareils avec en particulier des dates précises d'événements,...

Structure des fichiers
Chaque ligne du fichier correspond à un événement et se caractérise par une date sous la forme (heure,minute,seconde), une variation de puissance active ΔP et la valeur de la puissance active après la variation ainsi que la variation de la puissance réactive ΔQ et la valeur de la puissance réactive après la variation. Les événements sont classés chronologiquement.

$$\begin{array}{ccccccc} h_0 & min_0 & sec_0 & \Delta P_0 & \Delta Q_0 & P_0 & Q_0 \\ h_1 & min_1 & sec_1 & \Delta P_1 & \Delta Q_1 & P_1 & Q_1 \\ & & & \vdots & & & \\ h_n & min_n & sec_n & \Delta P_n & \Delta Q_n & P_n & Q_n \end{array}$$

Deux contraintes doivent impérativement être respectées lorsque ces fichiers sont construits manuellement :

- Les événements doivent être classés dans leur ordre chronologique
- Deux événements ne doivent en aucun cas avoir la même date.

Ouverture et fermeture d'un fichier
Les fichiers sont tous ouverts en lecture lors de la phase d'initialisation qui a lieu à la première (et unique) occurrence vraie du signal H-0-FIN. Ils sont fermés lorsque l'utilisateur quitte le simulateur (Valeur faux du signal H-0-FIN).

Parcours du fichier

1. À l'ouverture du fichier, la première ligne correspondant au premier événement du fichier est lue
2. Lorsque l'utilisateur décide de mettre en fonctionnement l'appareil correspondant à ce fichier, un chronomètre associé à l'appareil et initialisé à $t = 0$ est déclenché.
3. Tant que le fichier n'est pas vide :
   (a) chaque seconde, la date lue dans le fichier est comparée à la valeur courante du chronomètre.
   (b) En cas d'égalité entre la valeur lue et la valeur courante du chronomètre
      - l'événement est déclenché
      - la ligne suivante correspondant à l'événement suivant est lue

Reconstruction de la courbe de puissance
La puissance est rendue disponible à chaque instant entre deux événement par interpolation linéaire.

$$\forall i \geq 0,\ \forall t \in [t_i, t_{i+1}[, \quad P(t) = P_i + \frac{P_{i+1} - \Delta P_{i+1} - P_i}{t_{i+1} - t_i}(t - t_i)$$
$$Q(t) = Q_i + \frac{Q_{i+1} - \Delta Q_{i+1} - Q_i}{t_{i+1} - t_i}(t - t_i)$$

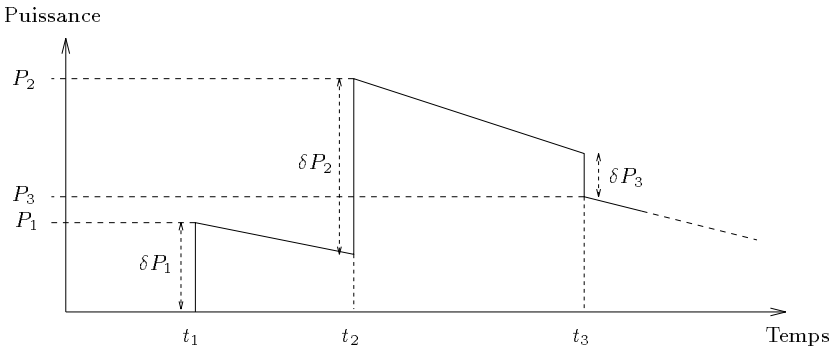


FIG. 13 - *Reconstruction d'une courbe de puissance à partir de la description fichier d'un appareil*

3.1.3 Calcul des événements du circuit total

Considérons que les différents appareils constituant le circuit sont numérotés de 1 à P. Pour tout $i \in \{1, .., P\}$, soit n_i le nombre d'états possibles pour l'appareil numéro i. Le circuit peut lui-même être modélisé par une machine à états dont le nombre d'états possibles est au plus

$$\prod_{i=1}^{P} n_i$$

Chaque transition d'un état à un autre sera considéré comme un événement auquel sont associées une date, une puissance et une variation de puissance.

- Les événements: Soit E_i l'ensemble des événements associés à l'appareil i. Alors l'ensemble E des événements associés au circuit total est donné par

$$E = \bigcup_{i=1}^{P} E_i$$

Ce qui se traduit en SIGNAL par la fusion de l'ensemble des signaux représentant les événements des différents appareils.

- Les puissances

$$\forall e \in E,\ P(t_e) = \sum_{i=1}^{P} P_i(t_e)$$

- Les variations de puissance

$$\forall e \in E,\ \Delta P(t_e) = \sum_{e \in E_i} \Delta P_i(t_e)$$

Si n événements ont lieu à la même date, dans le circuit total il sont regroupés en un seul événement, de même date et de variation de puissance égale à la somme des variations de puissance des différents appareils.

3.2 L'environnement de simulation

Autour de la modélisation du circuit est construit un environnement de simulation qui permet à l'utilisateur de contrôler l'ensemble du simulateur à l'aide d'un tableau de contrôle, d'afficher à l'écran certains résultats, de construire le fichier contenant l'ensemble de la simulation ainsi que des fichiers associés aux différents appareils.

1. Une première partie concerne le contrôle du simulateur. Elle se situe en "amont" du circuit, est décrite par le processus CONTEXT et permet de contrôler :
   (a) le temps à l'aide d'un PACE-MAKER. On cherche bien entendu à pouvoir réaliser plusieurs jours de simulations en un minimum de temps mais on veut aussi pouvoir facilement mettre en marche ou arrêter les appareils à des instants précis. Il faut pour cela être en mesure de ralentir la vitesse de simulation, voire l'arrêter complètement et ce de façon temporaire.
   (b) la mise en fonctionnement et l'arrêt des appareils, qu'il s'agisse des appareils modélisés en SIGNAL ou des appareils décrits par des fichiers. Ce contrôle est réalisé manuellement et il est également possible de faire intervenir des variables aléatoires.
2. Une deuxième partie concernant plus particulièrement le traitement des résultats de la simulation est décrite par un processus situé en "aval" du processus CIRCUIT. Ce processus est composé de plusieurs sous-processus permettant
   (a) d'afficher les courbes de puissance active et réactive à deux échelles de temps différentes sur quatre fenêtres graphiques.
   (b) d'afficher le temps écoulé depuis le début de la simulation.
   (c) de construire le fichier contenant le résultat de la simulation du circuit total.

3. Une troisième partie se trouve dissimulée à l'intérieur du processus circuit. Il s'agit de processus servant à l'enregistrement de mesures faites individuellement sur chacun des appareils du circuit.

3.2.1 Le contrôle du temps

Nous cherchons à réaliser des simulations sur des durées pouvant aller de plusieurs jours à plusieurs semaines voire plusieurs mois en un temps minimal (quelques minutes ou quelques heures) tout en étant capable de contrôler parfaitement les instants de mise en marche et d'arrêt des appareils. Il nous faut alors un moyen pour contrôler la vitesse de fonctionnement du simulateur, ce qui revient à faire varier le rapport

$$\frac{\textit{Durée de la simulation}}{\textit{Durée simulée}}$$

vers des valeurs très petites lorsque l'on ne cherche pas à contrôler les appareils, et à l'inverse vers des valeurs plus grandes lorsque l'on cherche à mettre en fonctionnement ou à arrêter des appareils à des instants précis. Nous pouvons également parfois désirer suspendre temporairement la simulation (i.e. arrêter le temps) dans l'objectif de mettre en fonctionnement et d'arrêter simultanément plusieurs appareils. Ceci est rendu possible par l'utilisation d'un pace-maker.

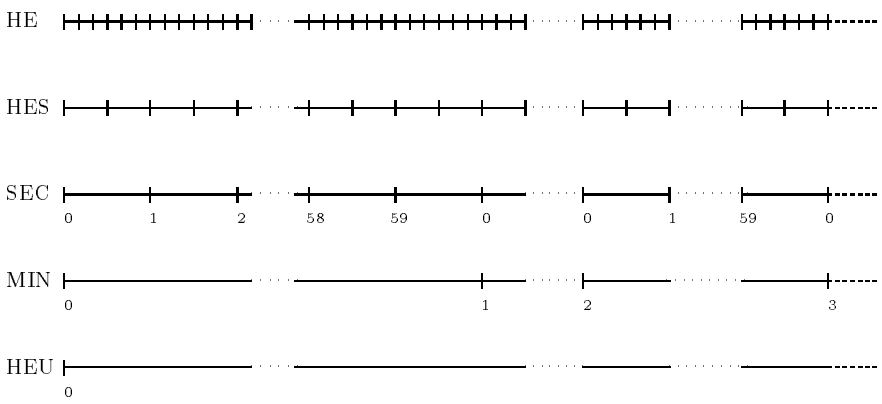


FIG. 14 - *La gestion du temps*

Le pace-maker
Le processus SIMULATEUR (figure 7) possède en entrée un seul signal, le signal HE qui détermine l'horloge la plus rapide du simulateur. À partir de cette horloge, on construit par

échantillonnage une deuxième horloge : l'horloge HES. C'est cette nouvelle horloge qui est utilisée comme horloge de références (toutes les autres horloges du simulateur en découlent directement ou indirectement). Le pacemaker consiste simplement à permettre à l'utilisateur de faire varier la fréquence d'échantillonnage à l'aide d'un tableau de contrôle (figure 15).

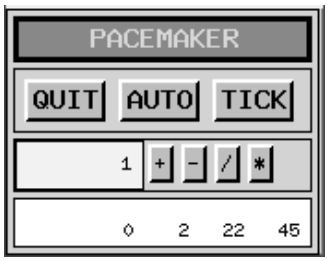


FIG. 15 - *Tableau de contrôle du pace-maker*

Calcul des heures, minutes, secondes
La plus petite unité de temps que nous manipulons dans ce simulateur est la seconde. Nous construisons donc à partir de l'horloge HES l'horloge des secondes. Par des techniques de compteurs modulo on en déduit le calcul des heures, minutes et secondes (figure 14).

3.2.2 Le contrôle des appareils

Nous cherchons à modéliser l'utilisation qui est faite de l'appareil. Il s'agit ici plutôt d'appareils type lave-linge qui ne sont pas mis en fonctionnement pour de longues périodes mais qui ont un cycle de fonctionnement de faible durée, et dont la fréquence d'utilisation dépend du contexte et non de l'appareil lui-même.

Contrôle manuel
Un tableau de contrôle contenant des boutons poussoirs permet à l'utilisateur de mettre en marche ou d'arrêter manuellement chacun des appareils (figure 16). Le processus SCAN() récupère l'état des boutons du tableau de contrôle des appareils. On appelle pour cela des processus externes à l'horloge HSCAN. À chaque fois que l'état d'un bouton est modifié, un signal de mise en fonctionnement ou d'arrêt de l'appareil correspondant est envoyé. Lors de la phase d'initialisation, tous les appareils sont arrêtés.

Scénarios aléatoires
Certains appareils sont mis en service pour une durée très longue. Il n'est alors pas nécessaire de chercher à modéliser autre chose que le comportement de l'appareil lui même. C'est le

FIG. 16 - *Tableau pour le contrôle des appareils*

cas par exemple du convecteur dont la succession entre les états "ON" et les états "OFF" est due à un thermostat qui est un composant à part entière de l'appareil.

Par contre, pour d'autres types d'appareils, il peut être intéressant de donner une modélisation statistique d'une partie de l'environnement. C'est le cas du lave-linge dont le cycle de fonctionnement dure entre une et deux heures et pour lequel on aimerait simuler aléatoirement la fréquence d'utilisation (qui dépend du ménage qui l'utilise et non d'un comportement propre à l'appareil). Nous utilisons pour cela un processus qui génère aléatoirement des ON et des OFF. Ce processus est contrôlé par l'utilisateur à partir du tableau de commande et c'est ce dernier qui contrôle directement l'appareil.

3.2.3 L'affichage des courbes de puissances

Les courbes de puissance active et réactive sont visualisées à deux échelles de temps différentes. Une nouvelle fois, il ne s'agit pas ici de courbes de puissances réelles mais de courbes de puissances reconstruites à partir des simulations de mesures. Ces courbes sont essentiellement visualisées à l'écran pour permettre à l'utilisateur de mieux suivre le déroulement de la simulation. Cet affichage est réalisé par le processus SCREEN. Les paramètres de

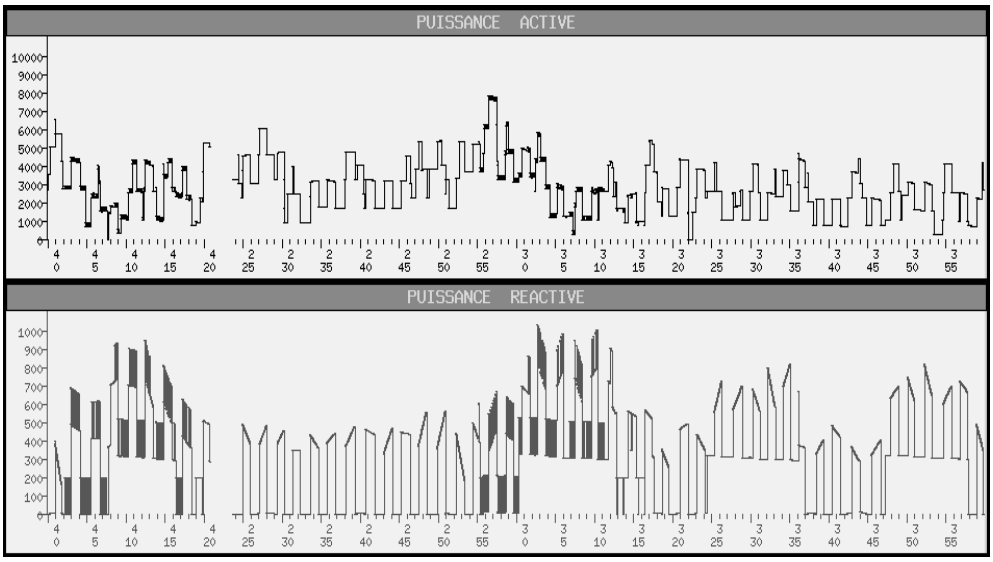


FIG. 17 - *Affichage des courbes de puissances actives et réactives, à l'une des deux échelles*

ce processus permettent essentiellement de choisir pour chacune des courbes de puissance :

- L'échelle de l'axe temporel

– L'échelle de l'axe des puissances

Quatre fenêtres graphiques permettent ainsi à l'utilisateur de visualiser chacune des courbes de puissance actives et réactives sur deux fenêtres à des échelles de temps différentes.

3.2.4 La génération de fichiers résultats

Comme nous l'avons mentionné plus haut, deux types de fichiers résultats sont générés. Celui que nous décrivons ici concerne le circuit complet mais les fichiers associés à chaque appareil sont construits de la même façon.

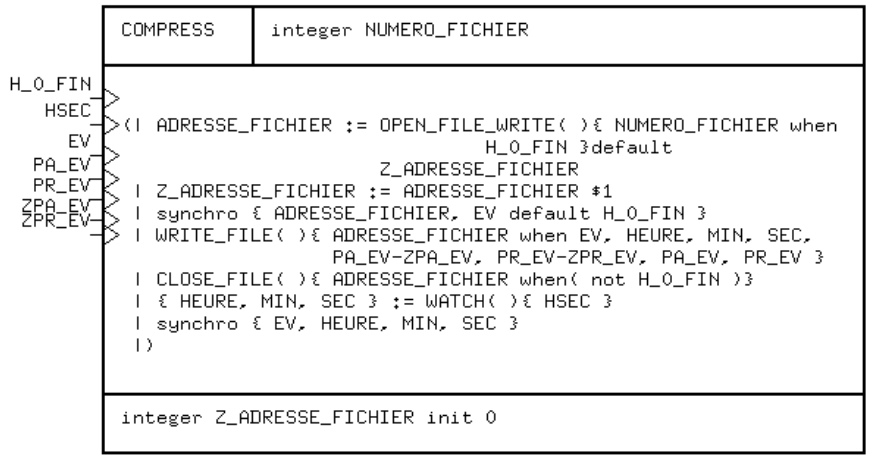


FIG. 18 - *Processus de construction du fichier résultat*

Dans les signaux de sortie du processus CIRCUIT un signal EV représente l'ensemble des instants où un événement apparaît sur l'un des appareils du circuit. Quatres autre signaux synchrones au signal EV donnent les puissances active et réactive consommées avant la transition (ZPA-EV et ZPREV) et après la transition (PAEV et PREV). À chaque instant de présence d'un événement (signal EV présent) la ligne

```
HEURE  MIN  SEC    (PA_EV - ZPA_EV)    (PR_EV - ZPR_EV)    PA     PR
```

est écrite sur le fichier résultat. Les processus OPEN-FILE-WRITE, WRITE-FILE et CLOSE-FILE sont des processus externes. Le processus WATCH calcule les heures minutes et secondes à partir de l'horloge des secondes HSEC.

En fin de simulation, nous obtenons un fichier contenant sur chaque ligne

– soit un événement simple ayant eu lieu sur l'un des appareils,

– soit un événement multiple correspondant à plusieurs événements ayant pu se produire sur plusieurs appareils.

Chacun de ces événements est décrit par sa date, les puissances active et réactive après franchissement de la transition et les variations de puissance correspondantes. Ils sont bien entendu classés par ordre chronologique croissant.

$$\begin{array}{cccccccc} h_0 & min_0 & sec_0 & P_0 & Q_0 & P_0 & Q_0 \\ h_1 & min_1 & sec_1 & \Delta P_1 & \Delta Q_1 & P_1 & Q_1 \\ & & & \vdots & & & \\ h_{i+1} & min_{i+1} & sec_{i+1} & \Delta P_i & \Delta Q_i & P_i & Q_i \\ & & & \vdots & & & \end{array}$$

A Modélisation des phénomènes aléatoires

Le langage SIGNAL ne permet pas actuellement de traiter les phénomènes aléatoires. Il est cependant possible d'avoir recours à des processus externes, ce qui va ici nous permettre de faire appel à une bibliothèque de génération de nombres aléatoires décrite en langage C.

Nous décrivons dans un premier temps les phénomènes étudiés nécessitant l'utilisation de variables aléatoires. Nous présentons ensuite brièvement le fonctionnement du générateur de nombres aléatoires pour nous intéresser aux trois lois de probabilité non-standards dont nous avons besoin pour modéliser les durées.

A.1 La Génération de nombres aléatoires

Les premiers générateurs de nombres aléatoires [15] étaient basés sur des phénomènes physiques (bruits blancs, émission radioactive de particules, etc ...) dont les propriétés stochastiques avaient été démontrées. Une telle méthode présentait de nombreux inconvénients : elle nécessitait la connexion au calculateur d'une unité spéciale produisant le phénomène utilisé pour la génération des nombres et les séquences de nombres générés n'étaient pas reproductibles. Il était donc très difficile de tester ou de mettre au point une simulation.

L'étape suivante fut l'utilisation des tables de nombres aléatoires construites à partir de phénomènes physiques qui permettaient toute la reproductibilité désirée. L'utilisation de ces tables n'étaient pas très aisée. Aussi chercha-t-on très tôt à trouver des algorithmes permettant de générer des suites de nombres apparemment au hasard.

Le principe général des algorithmes générateurs de nombres aléatoires est de définir un "état de machine" par un ou plusieurs nombres et de changer cet "état" grâce à un algorithme déterministe quelconque, chaque fois qu'un nouveau nombre est demandé. Du fait que le nombre d'états est nécessairement fini, une propriété primordiale des générateurs aléatoires est de produire des suites périodiques de nombres. Il est donc important d'essayer d'obtenir la période la plus longue possible. Ceci va dépendre de la manière dont un état est défini, mais aussi de l'algorithme qui définit les changements d'états.

Depuis la première génération d'ordinateurs, de nombreux algorithmes ont été proposés pour générer des suites de nombres aléatoires uniformément distribués. Les tout premiers algorithmes avaient non seulement le désavantage de ne pas donner de très bon résultats du point de vue de la qualité statistique, mais aussi celui de ne permettre aucune étude théorique des suites générées (période, auto-corrélation,...)

Le générateur que nous allons utiliser repose sur une méthode proposée en 1951 par Lehmer. Il s'agit de la *méthode congruentielle multiplicative*. Nous nous contentons ici de présenter le minimum d'éléments nécessaires à une bonne utilisation de la bibliothèque de génération de nombres aléatoires.

A.1.1 Méthode congruentielle multiplicative

Comme beaucoup d'autres, cette technique est basée sur une relation de congruence. Le k-ième nombre aléatoire est donné par la récurrence :

$$s_k = as_{k-1} \text{ MOD } m \ , k = 1, 2, 3, \ldots$$

Nous remarquons alors que pour déterminer le générateur aléatoire, il nous faut fixer le s_0 (la *racine* du générateur), m (le *terme* de la congruence) et a (le *multiplicateur*).

Ces différents paramètres sont choisis de façon à satisfaire deux critères de qualité qui sont :

1. la période du générateur est maximale et égale à $m - 1$
2. les nombres générés sont indépendants et uniformément distribués dans l'intervalle $]0, m[$.

En général, le fait qu'un générateur aléatoire congruentiel a une période maximale est une propriété algébrique provenant d'une structure de groupe cyclique.

On peut ainsi établir que m est choisi tel que

$$m = 2^q - 1 (\text{nombre de Mersenne ou Fermat})$$

avec la contrainte supplémentaire que q soit premier. Ainsi pour une machine à mots de 32 bits, il convient de choisir

$$m = 2^{31} - 1$$

Une fois m choisi, il nous reste à prendre a parmi les racines primitives de m pour obtenir une période maximale. (si $m = 2^{31} - 1$ il en existe $\varphi(m - 1) = 534\ 000\ 000$).

Il faut ensuite choisir parmi les racines primitives de m celles qui assurent de bonnes propriétés statistiques aux suites générées. Notons que lorsque a est une racine primitive de m, pour que la période soit maximale, il faut que s_0 soit premier avec m.

Les nombres ainsi obtenus sont compris entre 1 et m-1, une division par m permet d'avoir des nombres compris dans $]0, 1[$ comme désiré.

Pour augmenter la période et obtenir une meilleur robustesse statistique, il est également possible de faire fonctionner en parallèle plusieurs générateurs congruentiels multiplicatifs et de combiner leurs états.

Pour l générateurs :

$$\begin{aligned} s_{0,i} &= a_0 s_{0,i-1} \text{ MOD } m_0 \\ s_{1,i} &= a_1 s_{1,i-1} \text{ MOD } m_1 \\ &\vdots \\ s_{l,i} &= a_0 s_{l,i-1} \text{ MOD } m_l \end{aligned}$$

L'Écuyer propose la combinaison

$$Z_i = \left(\sum_{j=1}^{l} (-1)^{j-1} s_{j,i} \right) \text{ MOD } [m_1 - 1]$$

qui fourni une approximation de la v.a. uniforme discrète sur $\{0,\ldots,m_1 - 2\}$.

A.1.2 Générateurs indépendants

Une autre de nos préoccupations est de pouvoir disposer d'un nombre suffisant de générateurs indépendants pour simuler l'ensemble des variables aléatoires de façon indépendante. Nous détaillons ici une méthode qui consiste à construire plusieurs générateurs indépendants à partir d'un seul générateur congruentiel multiplicatif.

Considérons un générateur de base de période p. Soient G, V et W trois entiers positifs tels que $G2^V 2^W \leq p$. Soit $S = \{Z_0, Z_1, \ldots\}$ l'ensembles des termes et T la fonction de transition du générateur

$$\begin{aligned} T : S &\longrightarrow S \\ Z_i &\longrightarrow Z_{i+1} = T(Z_i) \end{aligned}$$

Nous partitionnons ainsi la suite $(Z_n)_{n \geq 0}$ en G sous-suites disjointes finies dont le terme initial est donné par :

$$\forall g \in \{1..G\}, \quad I_g = Z_{(g-1)VW} = T^{VW}(I_{g-1})$$

Chacune de ces G sous-suites est elle même composée de V sous-sous-suites de longueurs W. On obtient ainsi G générateurs virtuellement indépendants.

À chaque instant durant l'exécution du programme, chacun des générateurs correspondant à une sous-suite est dans un état C_g (état courant du générateur G), et dans la k-ième sous-sous suite de ce générateur.

$$\forall g \in \{1..G\}, \quad \exists k, 0 \leq k < V, \quad \exists n, 0 \leq n < W, \quad C_g = T^{(k-1)W+n}(I_g)$$

Posons

$$\begin{aligned} L_g &= T^{(k-1)W}(I_g) \\ N_g &= T^{kW}(I_g) \\ &= T^{W}(L_g) \end{aligned}$$

L_g est le premier terme de la sous-sous-suite dans lequel se trouve l'état courant C_g du générateur g et N_g est le premier terme de la sous-sous-suite suivante du générateur g.

A.2 Une bibliothèque de génération de nombres aléatoires

La bibliothèque de génération de nombres aléatoires que nous utilisons a été écrite par Barry W. Brown et James Lovato[3]. Les algorithmes ont été décrits par Pierre l'Écuyer et Serge Côté [16]. Nous utilisons une version écrite en C. Nous avons également rajouté quelques fonctions supplémentaires pour la génération de nombres aléatoires suivant des lois non standards que nous décrivons dans A.2.2. Nous conservons dans toute cette partie les notations de la section A.1.

Cette bibliothèque utilise deux générateurs congruentiels multiplicatifs en parallèle, la période est d'environ 2.3×10^{18}. Pour les deux générateurs congruentiels de base, les paramètres de la relation de congruence sont :

$$\begin{aligned} m_1 &= 2147483563 \\ m_2 &= 2147483399 \\ a_1 &= 40014 \\ a_2 &= 40692 \end{aligned}$$

Nous disposons de 32 générateurs virtuellement indépendants :

$$\begin{aligned} G &= 32 \\ V &= 20 \\ W &= 30 \end{aligned}$$

A.2.1 Fonctions de contrôle et d'accès aux générateurs

Lors de l'initialisation des générateurs g:

$$\begin{aligned} C_g &= I_g \\ L_g &= I_g \end{aligned}$$

Au cours de l'exécution du programme, pour chaque générateur, le triplet (I_g, L_g, C_g) est mémorisé. Il est alors possible de ré-initialiser individuellement chacun des générateurs g à l'une des trois valeurs I_g, L_g ou N_g. La valeur de L_g est alors réactualisée si nécessaire.

3. Barry W. Brown and James Lovato are with Department of Biomathematics, Box 237, The University of Texas, M.D. Anderson Cancer Center, 1515 Holcombe Boulevard,Houston, TX 77030

La valeur de $Z_0 = I_1$ est théoriquement choisie automatiquement mais il est possible d'en choisir une différente. Pour chacun des autres générateurs ($g > 1$), la valeur de I_g est recalculée en fonction de la valeur de I_1 et chaque C_g et L_g est également réactualisé.

Les cinq fonctions suivantes sont les fonctions utilisées pour gérer l'utilisation des 32 générateurs.

void setall(long iseed1, long iseed2)
Met les germes initiaux du générateur 1 à $0 \leq iseed1 \leq 2147483562$ et $0 \leq iseed2 \leq 2147483398$. Les germes initiaux des autres générateurs sont recalculés à partir de ces deux valeurs. Cette procédure est appelée automatiquement au début de l'exécution du programme avec par défaut les paramètres $iseed1 = 1234567890$ et $iseed2 = 123456789$. Les états courants des générateurs sont positionnés suivant ces valeurs initiales.

void initgn(long isdtyp)
ré-initialise l'état du générateur courant : si isdtyp=-1, ré-initialise les germes à leur valeur initiale; si isdtyp=0, ré-initialise les germes aux premières valeurs du bloc courant; et si isdtyp=1, ré-initialise les germes aux premières valeurs du bloc suivant.

void advnst(long k)
avance l'état du générateur courant de 2^k valeurs et remet les germes initiaux a cette valeur. Ainsi chaque générateur peut produire plusieurs suites disjointes. (si V et W sont suffisamment grand) et chaque suite est indépendante de la longueur des suites précédentes.

void gscgn(long getset, long *g)
si getset = 0, retourne la valeur du générateur courant et si getset = 1, le générateur courant devient le g.

void getsd(long *iseed1, long *iseed2)
retourne la valeur des deux germes entiers du générateur courant.

A.2.2 Génération de variables aléatoires

Les méthodes de génération de nombres aléatoires que nous avons présentés permettent de simuler une loi uniforme sur $\{1, ..., M\}$ à partir de laquelle on obtient par division une loi uniforme sur $[0, 1]$. Ceci donne lieu aux deux fonctions **ignlgi()** et **ranf()** :

long ignlgi()
fonction retournant un entier suivant une loi de distribution uniforme sur $(1, .., 2147483562)$, en utilisant le générateur courant.

float ranf()
fonction retournant un réel en suivant une loi de distribution uniforme sur $[0, 1]$

C'est à partir de ces deux fonctions de base que sont obtenues toutes les autres lois.

La plupart des lois standards sont disponibles dans toutes les bibliothèques de génération de nombres aléatoires. Nous n'entrerons pas dans les détails des différentes méthodes utilisées, mais nous proposons d'étudier ici le cas de trois lois moins courantes dont nous avons néanmoins besoin dans le cadre de notre application. L'objectif est une nouvelle fois de se faire une idée des méthodes utilisées en génération de nombres aléatoires. Il n'est entre autre pas fait mention des méthodes d'optimisation et d'analyse numérique utilisées dans cette bibliothèque.

Nous présentons trois lois permettant de simuler des durées : la *weybull*, la *rayleigh* et l'*inverse gaussienne*. Les deux premières sont générées par une méthode classique d'inversion de la fonction de répartition, la troisième fait appel à un algorithme spécifique que nous détaillerons.

Première méthode: Inversion de la fonction de répartition

Pour utiliser cette méthode, il faut connaître une forme explicite de la fonction inverse de la fonction de répartition, c'est-à-dire une forme explicite de $F^{-1}(x)$. La méthode procède en deux étapes :

1. génération d'un nombre U avec un générateur uniformément distribué dans $]0,1[$.
2. génération d'un nombre X tel que $X = F^{-1}(U)$.

Montrons que X a bien la propriété désirée, c'est-à-dire que la fonction de répartition de X est $F(x)$

$$\begin{aligned} P[X \leq x] &= P[F^{-1}(U) \leq x] \quad \text{par définition de X} \\ &= P[F(F^{-1}(U)) \leq F(x)] \quad \text{car } F \text{ est croissante} \\ &= P[U \leq F(x)] \\ &= F(x) \quad \text{puisque } U \text{ est uniformément distribuée dans }]0,1[. \end{aligned}$$

Nous donnons ici deux exemples de lois pour lesquelles nous utilisons cette méthode de génération.

Rayleigh

Les caractéristiques techniques sont données par le tableau 1. On en déduit l'algorithme de simulation suivant :

1. générer une loi uniforme U sur $[0,1]$
2. retourner $Y = \sqrt{(a^2 - 2 * \log(U))}$

Weybull

Les caractéristiques techniques sont données par le tableau 2. L'algorithme qui en ressort utilise la génération d'une loi exponentielle de paramètre $\lambda = 1.0$. Il consiste à :

1. générer une loi normale exponentielle E de paramètre 1.0

2. Retourner $Y = E^{1/a}$

Nous noterons que la loi exponentielle est elle calculée à partir d'une loi uniforme sur $[0, 1]$.

Deuxième méthode: exemple de l'inverse Gausienne

La fonction de répartition de l'inverse gaussienne n'étant pas inversible (voir tableau 3) nous avons ici recours à une méthode directement liée à des propriétés de la loi elle même.

Lemme A.1 *(Shuster, 1968)*
Si X suit une loi inverse gaussienne $I(\lambda, \mu)$ alors la variable aléatoire

$$\frac{\lambda(X-\mu)^2}{\mu^2 X}$$

a la distribution du carré une variable aléatoire suivant une loi normale, i.e. elle suit un χ^2 à un degré de liberté.

Générateur de Michael, Schucany et Haas pour une inverse gaussienne

1. générer une loi normale N
2. poser $Y = N^2$
3. poser $X_1 = \mu + \frac{\mu^2 Y}{2\lambda} - \frac{\mu}{2\lambda}\sqrt{4\mu Y + \mu^2 Y^2}$
4. générer une uniforme U sur $[0, 1]$
5. si $U \leq \frac{\mu^2}{X_1}$
   (a) alors retourner $X = X_1$
   (b) sinon retourner $X = \frac{\mu^2}{X_1}$

Densité	$f(x) = \frac{x}{\sigma^2} \exp -\frac{x^2}{2\sigma^2}, \quad x \geq 0$
fonction de répartition	$F(x) = \int_0^x \frac{t}{\sigma} \exp -\frac{t^2}{2\sigma^2} dt = \exp -\frac{x^2}{2\sigma^2} - 1$
inverse de la fonction de répartition	$F^{-1}(y) = \sqrt{-\frac{1}{\lambda} \log y}$
espérance et variance	$E[X] = \sigma\sqrt{\frac{\pi}{2}} \quad \text{et} \quad Var[X] = \sigma^2(2 - \frac{\pi}{2})$

TAB. 1 - *Caractéristiques d'une Rayleigh*

Densité	$f(x) = ax^{a-1} \exp -x^a, \qquad x \geq 0,\ a > 0$
fonction de répartition	$F(x) = \int_0^x at^{a-1} \exp t^a dt = \exp x^a - 1$
inverse de la fonction de répartition	$F^{-1}(y) = (-\frac{1}{\lambda} \log y)^{1/b}$
espérance et variance	$E[X] = \Gamma(1 + \frac{1}{a}) \quad \text{et} \quad Var[X] = \Gamma(1 + \frac{2}{a}) - \Gamma(1 + \frac{1}{a})^2$

TAB. 2 - *Caractéristiques d'une Weybull*

Densité	$f(x,\lambda,\mu) = \begin{cases} \sqrt{\frac{\lambda}{2\pi x^3}} \exp -\frac{\lambda(x-\mu)^2}{2\mu^2 x} & \text{si } x > 0 \\ 0 & \text{sinon} \end{cases}$
fonction caractéristique	$\phi(t) = \exp \frac{\lambda}{\mu}[1-(1-2it\mu^2/\lambda)^{\frac{1}{2}}]$
espérance et variance	$E[X] = \mu \quad \text{et} \quad Var[X] = \frac{\mu^2}{\lambda}$

TAB. 3 - *Caractéristiques d'une Inverse Gaussienne*

Références

[1] Sultanem F. *Repérage des charges usage domestique par l'analyse des signaux au niveau du tableau de comptage.* note HR 20-2072, EDF, 1989.

[2] Chazottes B. *Reconnaissance des charges : présentation du prototype du dispositif d'acquisition des signatures électriques.* note HR 22-2841, EDF, 1992.

[3] Bouliou O. et B. Chazottes. *Logiciel d'identification des signatures des appareils électro-domestiques - bilan des travaux.* note HR 22-2953, EDF, 1993.

[4] Grahovac G. *Appliance identification in the NIALMS.* repport 8246, Schlumberger Montrouge Research, 1993.

[5] M. Bons et Y. Deville. *Reconnaissance non-intrusive des appareils électro-domestique - État d'avancement d'une thèse.* Note HR 21/94/11, EDF, 1994.

[6] Nicolas Halbwachs, Paul Caspi, Paul Raymond, and Daniel Pilaud. The synchronous data flow programming language LUSTRE. *Proceedings of the IEEE*, 79(9):1305–1321, September 1991.

[7] G. Berry and G. Gonthier. *The* ESTEREL *Synchronous Programming Language: Design, Semantics, Implementation.* Research Report 842, INRIA, May 1988.

[8] Paul Le Guernic and Albert Benveniste. *Real-time synchronous, data-flow programming: The language* SIGNAL *and its mathematical semantics.* Research report 620, INRIA, Rocquencourt, France, June 1986.

[9] Albert Benveniste, Bernard Le Goff, and Paul Le Guernic. *Hybrid Dynamical Systems theory and the language* SIGNAL. Research Report 838, INRIA, Rocquencourt, April 1988.

[10] Paul Le Guernic, Albert Benveniste, Patricia Bournai, and Thierry Gautier. SIGNAL*: a data flow oriented language for signal processing.* Research report 378, INRIA France, Rocquencourt, March 1985.

[11] Thierry Gautier. *Conception d'un langage flot de données pour le temps réel.* PhD thesis, Université de Rennes 1, France, 1984. in french.

[12] Patricia Bournai, Vivianne Kerscaven, and Paul Le Guernic. Un environnement graphique pour la conception d'applications temps réel. In *Colloque sur l'ingénierie des interfaces homme-machine*, pages 181–190, Sophia-Antipolis, 1989.

[13] Loïc Besnard. *Compilation de* SIGNAL*: horloges, dépendances, environnement.* PhD thesis, Université de Rennes 1, France, September 1992. in french.

[14] Bruno Dutertre. *Spécification et preuve de systèmes dynamiques.* PhD thesis, Université de Rennes 1, France, December 1992. in french.

[15] Jacques Leroudier. *La simulation événements discrets.* Monographie d'informatique de l'AFCET, 1980.

[16] Pierre L'ECUYER and Serge CÔTÉ . Implementing a random number package with splitting facilities. *ACM Transaction on Mathematical Software*, 17(1):98–111, March 1991.