Centre d'Etude Régional pour l'Amélioration d e l'Adaptation à la Sécheresse

IMERIR
Institut Méditerranéen d'Etude et de Recherche en Informatique, Intelligence Artificielle et Robotique

ANNEXES

AUTOPARAMETRISATION DE MODELES DE SIMULATION DU DEVELOPPEMENT DES CULTURES

Réalisé par : Kadidia SAKO.
Promotion Léonard De VINCI 1994 - 1997.
Maître de **stage** : David BOGGIO.

Du 01 avril au 01 octobre 1997.

SOMMAIRE

1. CAHIER DES CHARGES 3

1.1 INTRODUCTION.. 4
1.1.1 Le projet 4
1.1.2 Les motivations 4
1.1.3 Le marché 5
1.2 DOCUMENTS SUR LE SUJET 6
1.3 PROFIL DES UTILISATEURS FINAUX 6
1.4 OBJECTIFS SPECIFIQUES 6
15 CONTRAINTES DU SYSTEME 7
1.5.1 Contraintes matérielles 7
1.5.2 Contraintes logicielles 7
1. 5.3 Contraintes de fonctionnement 7
7.5.4 Autres facteurs de qualité. 7
1.6 ASPECTS CONTRACTUELS.. ..... .

2, PLAN DE DEVELOPPEMENT 8

2.1 INTRODUCTION 9
2.2 DOCUMENTS APPLICABLES 9
2.3 DOCUMENTATION PAR PHASE ..... .
2.3.1 Documents à produire.. 9
2.3.2 Documents de livraison.. 9
2.4 PLAN DE CHARGE.. 10
2.5 ORGANISATION 11
2.5.1 Choix méthodologique.. 11
2.5.2 Gestion des configurations.. 12
2.6 PLANNING PREVISIONNEL.. 12

3. SPECIFICATIONS FONCTIONNELLES 14

3.1 INTRODUCTION.. 15
3.2 L'AUTOPARAMETRISATION DANS LE SYSTEME DE MODELISATION AU CERAAS 15
3.3 CHOIX METHODOLOGIQUES 17
3.4 DOCUMENTS EN REFERENCE.. 17
3.5 GLOSSAIRE DES TERMES DU DOMAINE. 17
3.6 DESCRIPTION FONCTIONNELLE.. 18
3.6.7 Modifications apportées à la base de données des simulations.. 18
3.6.2 Description générale de l'application (Autoparamétrisation) 24
3.6.3 Description des différents traitements.. 26
3 7 SPECIFICATIONS TECHNIQUES D'INTERFACE 29

4, CONCEPTION 30

4.1 INTRODUCTION 31
4.2 DOCUMENTS EN REFERENCE.. 31
4.3 GLOSSAIRE.. 31
4.4 DESCRIPTION DES OUTILS DE DEVELOPPEMENT.. 32
4.4.1 Visual basic 4.0.. 32
4.4.2Access2.0.. 32
4.5 CONCEPTION DETAILLEE 33
4.5.7 L'application d'autoparamétrisation.. 33
4.5.2 Le système SIMPLEX.. 36
4.5.3 Le système GRADIENT 91
4.5.4 Le système SIMPLEX + GRADIENT.. 106
4.5.5 Le système PARAMETRES.. 410
4.6 STRATEGIES DE REALISATION ET REGLE DE CODAGE.. 117

5. MODELE DU POIDS SEC DES FEUILLES DE LA CULTURE DE MIL 118

1. CAHIER DES CHARGES

Date d'édition : 15/04/1997.

1.1 INTRODUCTION

1.1.1 Le projet

Autoparamétrisation des modèles de simulation du développement des cultures.

Ce projet consiste en la mise en place d'une application permettant de déterminer la valeur des paramètres des modèles de simulation du développement des cultures, afin d'ajuster les données résultats de simulation à celles observées aux champs.

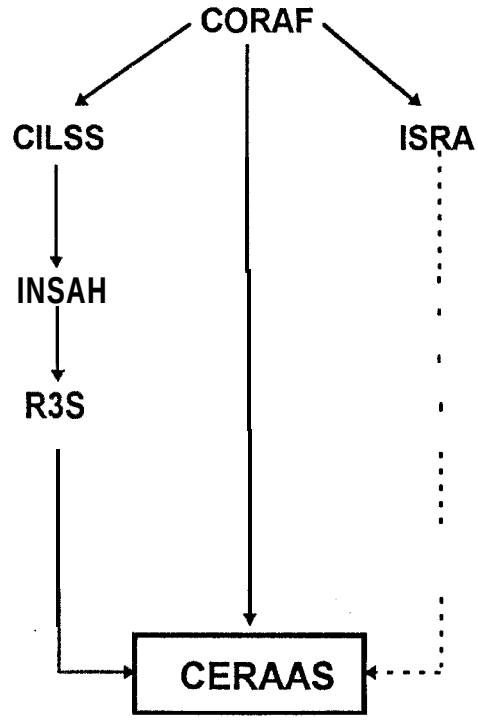
1 .1.2 Les motivations

1.1.2.1 Le *client*

Le client est le CERAAS (Centre d'Etude Régional pour l'Amélioration de l'Adaptation à la Sécheresse) dont le siège est à Bambey au SENEGAL.

Le CERAAS est abrité par le CNBA (Centre Nord Bassin Arachidier), l'un des centres de l'ISRA (Institut Sénégalais de Recherches Agricoles), membre de la CORAF (Conférence des Responsables de Recherche Agronomique en Afrique de l'Ouest et du Centre). Le CERAAS est sous la tutelle de la CORAF et du CILSS (Comité Inter-Etats de Lutte contre la Sécheresse au Sahel). Sa thématique « Amélioration de l'adaptation à la sécheresse et création variétale » est un des thèmes fédérateurs du R3S (Réseau de Recherche sur la Résistance à la Sécheresse), de l'INSAH (INstitut du SAHel).

Cette organisation est représentée par le schéma ci-dessous :



Les activités de recherches entreprises au CERAAS impliquent le développement des modèles de fonctionnement des cultures pour les principales espèces vivrières cultivées dans les zones sèches. Les résultats des modèles de simulation sont exploités dans des applications de suivi et de prévision agricole, par les différents intervenants des filières agricoles et alimentaires, les bailleurs de fonds et le gouvernement pour l'aide à la décision.

1.1.2.2 L'existant et le problème

La précision des modèles de culture dépend des valeurs des paramètres contrôlant leurs réponses aux conditions de l'environnement.

Actuellement, le modèle Ara.B.Hy. est utilisé au CERAAS ainsi que dans d'autres pays comme l'Argentine. D'autres modèles de simulation sont en cours de réalisation, notamment sur le soja et sur le mil. Le modèle Ara.B.Hy. simule le fonctionnement d'une culture d'arachide à partir d'informations climatiques et de données agronomiques et physiologiques recueillies sur le terrain. Ce modèle présente des résultats fiables avec un taux d'erreur de 10%.

Dans l'objectif d'améliorer et d'optimiser les résultats de modèles, le CERAAS examine la possibilité de modifier systématiquement les valeurs des paramètres choisis jusqu'à l'obtention de sorties simulées proches des valeurs observées aux champs.

Cependant, il n'existe aucun système adaptable à différents modèles permettant de caler (avec un taux d'erreur moindre) les données estimées par ces modèles avec celles observées sur le terrain.

1.1.3 Le marché

Sur le marché, il existe des modules de reparamétrisation des modèles de simulation du développement des cultures. Ces modules sont inclus dans le logiciel même et ne peuvent être interfacés avec d'autres modèles de simulation. De plus, les logiciels utilisant ces modèles coûtent cher. Par exemple, SAS (System an Applications System) Statistique, actuellement utilisé par le CERAAS, est un système qui permet entre autre, l'ajustement des paramètres d'un modèle à des données expérimentales. A partir d'un jeu de aonnées (entrée et sortie), d'une équation théorique et d'un jeu de paramètres initial, SAS calcule la valeur du jeu de paramètres qui optimise l'ajustement. On ne peut pas interfacer SAS avec une application extérieure.

1.2 DOCUMENTS SUR LE SUJET

- A simplex method for function minimisation, article de J. A. Nelder et R Mead dans Computer Journal, 308-313 (1965).
- METHODES D'ESTIMATION ET D'OPTIMISATION, DEA de Biomathématiques (S. HAZOUT)
- Dossier d'analyse provisoire (Y.C. SYLLA et D. BOGGIO)

f.3 PROFIL DES UTILISATEURS FINAUX

- **En utilisation** : l'application sera utilisée par les chercheurs du CERAAS. Ce sont des utilisateurs déjà initiés à l'informatique et connaissant certains outils statistiques.

- **En maintenance** : la maintenance sera réalisée par les informaticiens du CERAAS.

- **Environnement d'utilisation :** l'application sera installée sur les ordinateurs de type PC du CERAAS. Ce sont des Pentium.

***1.4** 05 JEC TIFS SPECIFIQUES*

Les objectifs spécifiques du stage sont :

- mettre en place une application « Autoparamétrisation » permettant d'ajuster des données estimées, contenues dans la base de données des simulations à celles observées aux champs. Le taux d'erreur engendré par cet ajustement doit être très faible. Les méthodes d'estimation et d'optimisation du simplex et du gradient (méthodes statistiques) seront utilisées.

  L'Autoparamétrisation doit être intetfaçable avec les modèles de simulation actuellement en cours de développement au CERAAS.

- modifier la base de données des simulations afin de l'adapter à l'autoparamétrisation.

1.5 CONTRAINTES DU SYSTEME

1.5.1 Contraintes matérielles

L'application utilisera les méthodes du simplex et du gradient. Ces dernières nécessitent beaucoup de calculs. Il faudra donc un ordinateur ayant une vitesse de fonctionnement élevée. Un ordinateur de type Pentium avec assez de place mémoire (pour la base de données des simulations) conviendra pour l'utilisation finale ainsi que pour le développement.

1.5.2 Contraintes logicielles

L'application sera développée avec Visual Basic 4.0 dans un environnement Windows 3.11. Elle sera donc réalisée dans le même environnement que les modèles de simulation en cours de développement actuellement au CERAAS. Les modifications de la base de données des simulations sera faite avec ACCESS 2.0 ou compatible.

1.5.3 Contraintes de fonctionnement

Pour le fonctionnement, il faut un personnel initié à l'informatique.

1.5.4 Autres facteurs de qualité

Le système doit être maintenable. Les I.H.M. (Interface Homme-Machine) doivent être conviviaux.

7.6 ASPECTS CONTRACTUELS

L'application ainsi que la documentation complète doivent être rendues à la fin du stage, c'est-à-dire le 01 octobre 1997. Elles restent une propriété du CERAAS.

2. PLAN DE DEVELOPPEMENT

Date d'édition : 17/04/1997.

2.7 INTRODUCTION

Le projet

Autoparamétrisation des modèles de simulation du développement des cultures.

Ce projet consiste en la mise en place d'une application permettant de déterminer la valeur des paramètres des modèles de simulation du développement des cultures, afin d'ajuster les données résultats de simulation à celles observées aux champs.

2.2 DOCUMENTS APPLICABLES

- Cahier des charges
- Plan qualité : les critères de qualité retenus sont :

1. La convivialité au niveau des IHM (Interface Homme-Machine)
2. La maintenabilité : la maintenance sera assurée par les informaticiens du CERAAS.
3. L'adaptabilité aux modèles de simulation du développement des cultures en cours de réalisation actuellement au CERAAS : l'application sera interfacée avec ces modèles.

2.3 DOCUMENTATION PAR PHASE

2.3.1 Documents à produire

Pour un bon déroulement du travail, les documents suivants seront réalisés :

- Le cahier des charges.
- Le plan de développement.
- Le dossier des spécifications fonctionnelles.
- Le dossier de conception.

2.3.2 Documents de livraison

A l'issue du stage, un rapport détaillé ainsi que les annexes seront remis en même temps que l'application.

2.4 PLAN DE CHARGE

Le tableau suivant montre la durée prévue pour chacune des tâches du projet. Il peut être sujet à des modifications tout au long de la réalisation.

Tâches	Durée prévue
Etude du besoin exprimé par le CERAAS	5j
Mise en place du plan de développement:	3j
Mise en place du dossier de spécifications fonctionnelles	3j
Mise en place du dossier de conception	18j
Codage des différentes fonctions	35j
Tests de validation	10j
Tests d'intégration	10j
Réalisation du rapport de stage	3j
Livraison de l'application	2j

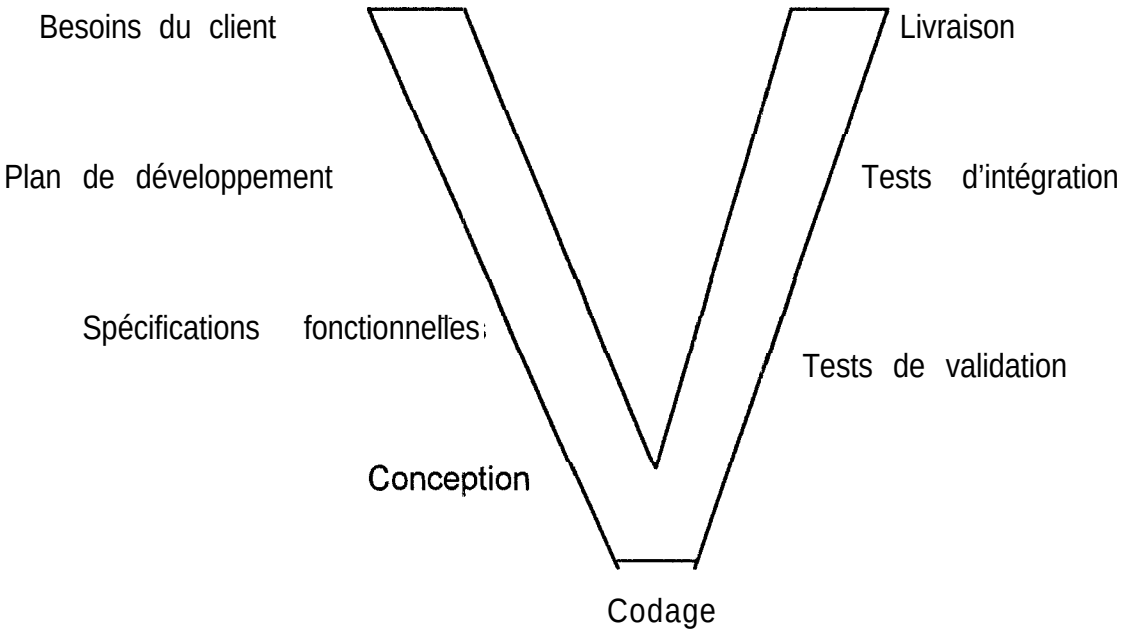
Cela fait au total 89 jours de travail, soit 74% du temps reparti sur les six mois (durée totale du stage). La marge obtenue est de 26%. Cette marge sera nécessaire tout au long de la réalisation du projet. Elle permet de parer aux imprévus.

2.5 ORGANISA TION

2.5.1 Choix méthodologique

La méthode d'analyse **MERISE** est utilisée pour la modification de la base de données des simulations. Cette méthode permet de bien analyser les différentes tables de données ainsi que les relations entre elles.

L.a méthodologie de développement est celle du cycle en V. Elle est bien adaptée à ce projet. C'est une méthodologie permettant de bien mener de bout en bout un projet informatique en passant par des étapes successives,, Chaque sous-module, réalisé, est testé, validé et puis intégré au reste de l'application. Le cycle en V se présente sous la forme suivante :



Au niveau de la validation du module final, après une évaluation du résultat obtenu par tes méthodes d'optimisation choisies au départ, la méthodologie à choisir sera celle de la spirale. Cette méthodologie permet de boucler sur les étapes de ta réalisation de l'application, c'est-à-dire de revenir à la première Etape du cycle en V. Cela permettra d'étudier d'autres méthodes d'optimisation"

2.5.2 Gestion des configurations

Le répertoire pour ce projet (situé sur le disque) est structuré de la façon suivante :

- Un sous-repertoire Document contenant tous les dossiers relatifs au projet (cahier des charges, Plan de développement, etc ...). Ce sous-repertoire est formé de deux parties :

  - Encours : pour les documents qui sont en cours de réalisation.
  - Final : pour les documents déjà validés.

- Un sous-repertoire codage renfermant le code de toutes les fonctions réalisées. Comme pour le premier, il est également formé de deux parties

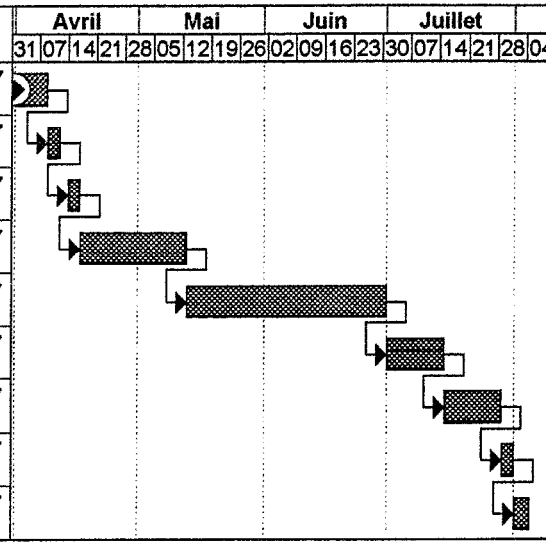
  - Encours : pour les modules qui sont en cours de réalisation.
  - Final : pour les modules déjà validés.

2.6 PLANNING PREVISIONNEL

Diagramme GANTT (Planning prévisionnel). Ce diagramme montre l'ordonnancement des tâches.

Diagramme GANTT

N°	Nom de la tâche	Durée	Début	Fin
1	Etude du besoin	5j	01/04/97	08/04/97
2	Mise en place du plan de développement	3j	09/04/97	11/04/97
3	Mise en place du dossier des spécifications fonctionnelles	3j	14/04/97	16/04/97
4	Mise en place du dossier de conception	18j	17/04/97	12/05/97
5	Codage des différentes fonctions de l'application	35j	13/05/97	30/06/97
6	Tests de validation	10j	01/07/97	14/07/97
7	Tests d'intégration	10j	15/07/97	28/07/97
8	Réalisation du rapport de stage	3j	29/07/97	31/07/97
9	Livraison de l'application	2j	01/08/97	04/08/97



Projet: Autoparamétrisation
Date de début :01/04/97
Date de fin : 0408197

Tâche
Progrès
Jalon
Récapitulative
Tâche reportée
Jalon reporté
Avancement reporté

3. SPECIFICATIONS FONCTIONNELLES

Date d'édition : 21/04/1997.

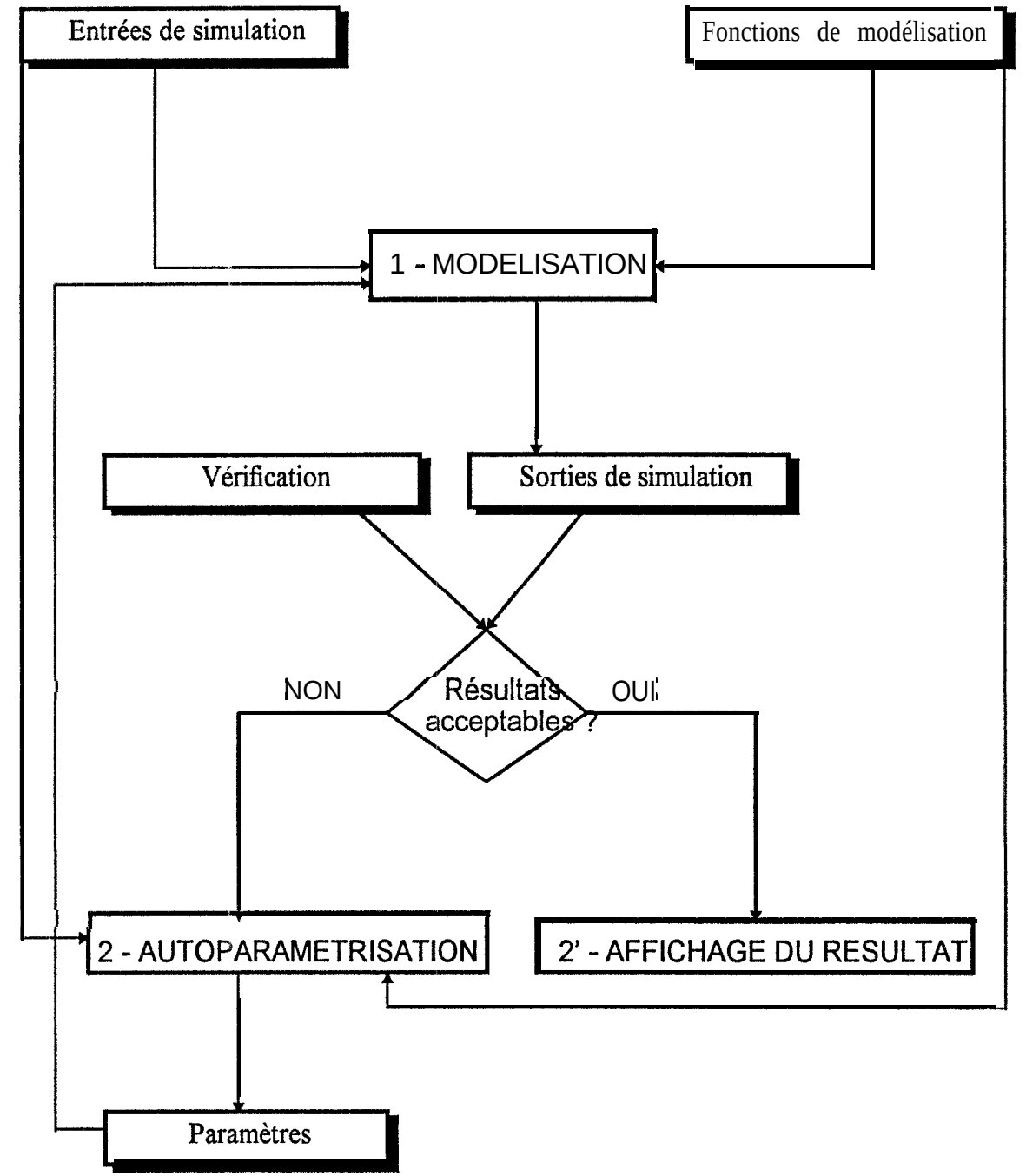
3.1 *INTRODUCTION*

Au toparamé trisa tion des modèles de simulation du développement des cultures.

Ce projet consiste en la mise en place d'une application permettant de déterminer la valeur des paramètres des modèles de simulation du développement des cultures, afin d'ajuster les données résultats de simulation à celles observees aux champs.
Le problème revient à minimiser une fonction à n variables. Cette fonction exprimant l'écart entre les données de sortie de simulation et celles observées aux champs.

3.2 L'autoparamétrisation dans le système de modélisation au CERAAS

Pour chaque simulation, les données d'entrées et les paramètres concernés sont appliqués à la fonction du modèle. Les données résultats sont comparées à celles de vérification recueillies sur le terrain. Si ces résultats sont satisfaisants par comparaison au taux d'erreur fixé au départ, ils sont affichés à l'écran. Dans le cas contraire et jusqu'à ce qu'ils soient satisfaisants, le traitement paramétrisation est à nouveau appelé. Les paramètres optimisés avec ce traitement sont stockés dans la table Paramètres. Le schéma récapitulatif est le suivant :



Tables de données
Traitements

Les différentes tables de données qui seront utilisées :

Entrées de simulation : table contenant les données fixes qui seront appliquées à la fonction de modélisation.

Fonctions de modélisation : renferme les fonctions permettant la modélisation des différentes cultures. Ces fonctions sont stockées sous forme de chaîne de caractères.

Vérification : contient des données réelles recueillies sur le terrain.

Paramètres : table contenant les variables du modèle qui sont modifiées pour l'auto-ajustement. Cette table est mise à jour par le module de paramétrisation.

Sorties de simulation : table remfermant les résultats de simulation,

3.3 CHOIX METHODOLOGIQUES

La programmation est du type événementielle. Les programmes sont commandés par des événements générés par les objets de l'interface utilisateur.

3.4 DOCUMENTS EN REFERENCE

- Le cahier des charges.
- Le plan de développement.

3.5 GLOSSAIRE DES TERMES DU DOMAINE

Evénement : un événement esl: provoqué par l'utilisateur (clic sur un bouton: frappe d'une touche) ou par un système (minuterie). II appelle la procédure qui lui est rattachée.

Objet : un objet peut-être une feuille, un contrôle, un écran, unapplication, etc.

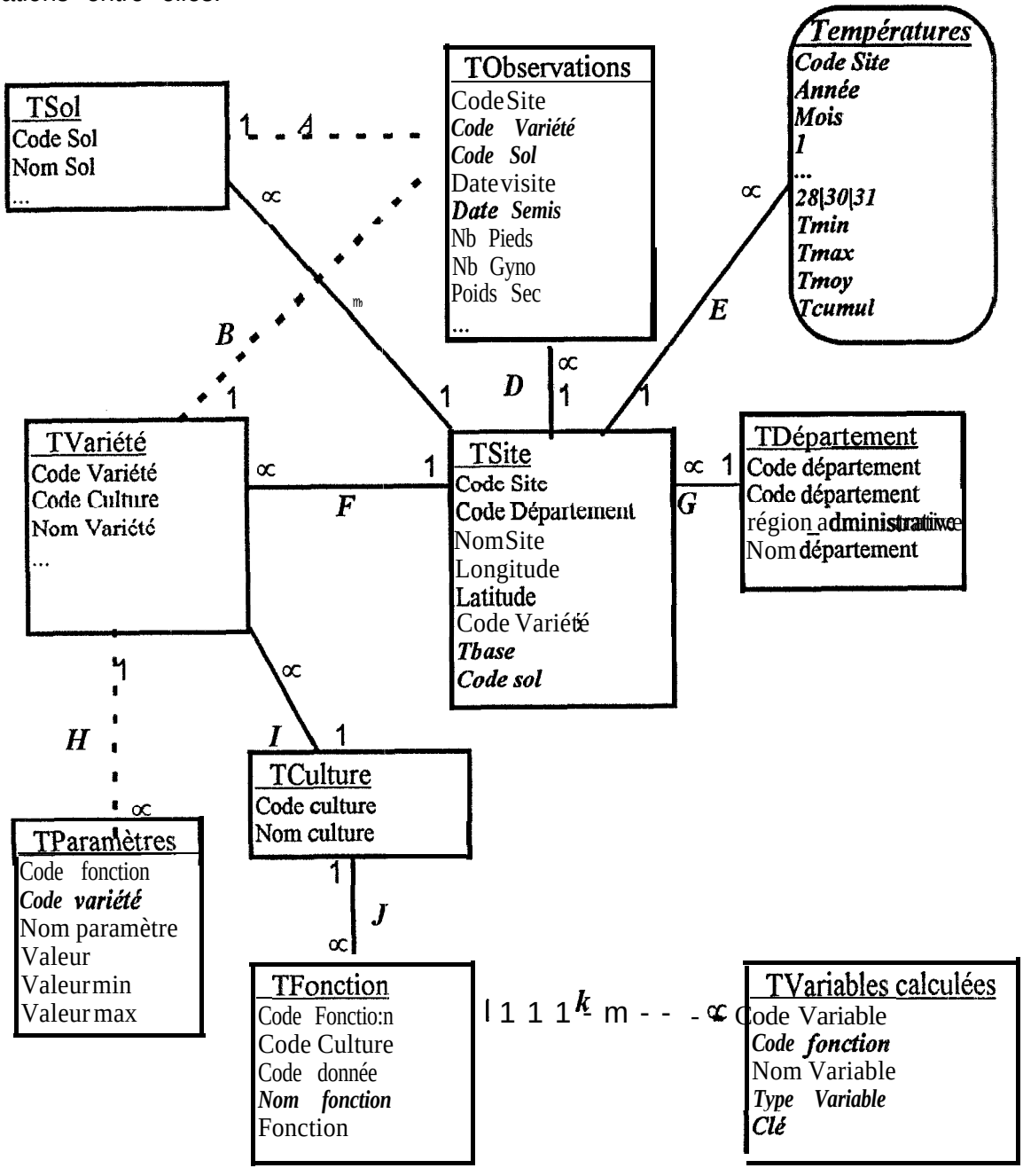
I.H.M. : interface Homme-Machine.

Fenêtre : I.H.M.

3.6 DESCRIPTION FONCTIONNELLE

3.6.1 Modifications apportées à la base de données des simulations

Le schéma suivant montre les modifications apportées à la base de données des simulations" II présente les différentes tables utilisées ainsi que les relations entre elles.



0 Nouvelle table à ajouter dans la base de données des simulations

▪ ▪ ▪ ▪ Nouvelles relations à prendre en compte.

——— Relations qui existaient déjà.

A , B, ... Code des relations.

En gras ***italique*** : Nouvelles données à ajouter dans la structure des tables.

3.6.1.1 Description des fables

TSol : décrit les caractéristiques de chaque type de sol où se développent des cultures

Nom du champ de donnée	Taille	Type	Description
Code sol	5	Texte	Code du sol
Nom sol	30	Texte	Nom du sol
Exposant	5	entier	Exposant
Multipliant	5	entier	Multipliant

TDépartement : découpage administratif du pays

Nom du champ de donnée	Taille	Type	Description
Code département	5	Texte	Code du département
Code région-adm	5	Texte	Code de la région administrative
Nom département	30	Texte	Nom du département

Températures : températures recensées sur un site donné

Nom du champ de donnée	Taille	Texte	Description
Code site	5	Texte	Code de la station
Année	4	Entier	Année à laquelle la température a été relevée
Mois	2	Entier	Mois à laquelle la température a été relevée
1 . . .28 | 30 | 31	2	Entier	Jour à laquelle la température a été relevée
Tmin	2	Réel	Température minimale relevée
Tmax	2	Réel	Température maximale relevée
Tmoy	2	Réel	Température moyenne relevée
Tcumul	2	Réell	Cumul des températures

TSite : table des différentes stations sur lesquelles les simulations ou les relevées sont effectuées

Nom du champ de donnée	Taille	Type	Description
Code site	5	Texte	Code du site
Code département	5	Texte	Code du département
Nom site	30	Texte	Nom du site
Longitude	4	Réel	Longittude
Latitude	4	Réel	Latitude
Code variété	5	Texte	Code de la variété de culture
Code sol	5	Texte	Code du type de sol
Tbase	2	Réel	Température de base

TObservations : les valeurs observées sur le terrain

Nom du champ de donnée	Taille	Type	Description
Code Obs	5	Texte	Code de l'observation
Code site	5	Texte	Code du site
Date visite	8	Date	Date de réalisation de l'observation
Code variété	5	Texte	Code de la variété de culture concernée
Code soll	5	Texte	Code du sol
Date semis	8	Date	Date de semis de la variété de culture
Poids sec feuille	5	Réel	Poids sec des feuilles de la variété de culture
. .			

TCulture : description des cultures

Nom du champ de donnée	Taille	Type	Description
Code culture	5	Texte	Code de la culture
Nom culture	30	Texte	Nom de la culture

TFonction : détail sur les fonctions de modélisation

Nom du champ de donnée	Taille	Type	Description
Code fonction	5	Texte	Code de la fonction de modélisation
Code culture	5	Texte	Code de la culture
Nom fonction	30	Texte	Nom de la fonction de modélisation
Fonction	variable-	Texte	Formulation de la fonction de modélisation

TParamètres : paramètres associés aux fonctions de modélisation

Nom du champ de donnée	Taille	Type	Description
Code paramètre	5	Texte	Code du paramètre
Code variété	5	Texte	Code de la variété de culture
Code fonction	5	Texte	Code de la fonction de modélisation
Nom paramètre	30	Texte	Nom du paramètre
Valeur	5	Réel	Valeur du paramètre
Valeur min	5	Réel	La plus petite valeur que peut prendre le paramètre
Valeur max	5	Réel	La plus grande valeur que peut prendre le paramètre

TVariété : les variétés de culture faisant l'objet d'une modélisation

Nom du champ de donnée	Taille	Type	Description
Code variété	5	Texte	Code de la variété de culture
Code culture	5	Texte	Code de la culture concernée
Nom variété	30	Texte	Nom de la variété de culture
...			

Tvariables calculées : les données dont les valeurs peuvent être obtenues paf un calcul

Nom du champ de donnée	Taille	Type	Description
Code variable	5	Texte	Code de la variable
Code fonction	5	Texte	Code de la fonction de modélisation
Nom variable	30	Texte	Nom de la variable
Type variable	30	Texte	Fixe ou calculée (Table dans laquelle se trouvent les résultats de calcul)
Clé	5	Texte	Clé d'accès à la table concernée par le calcul

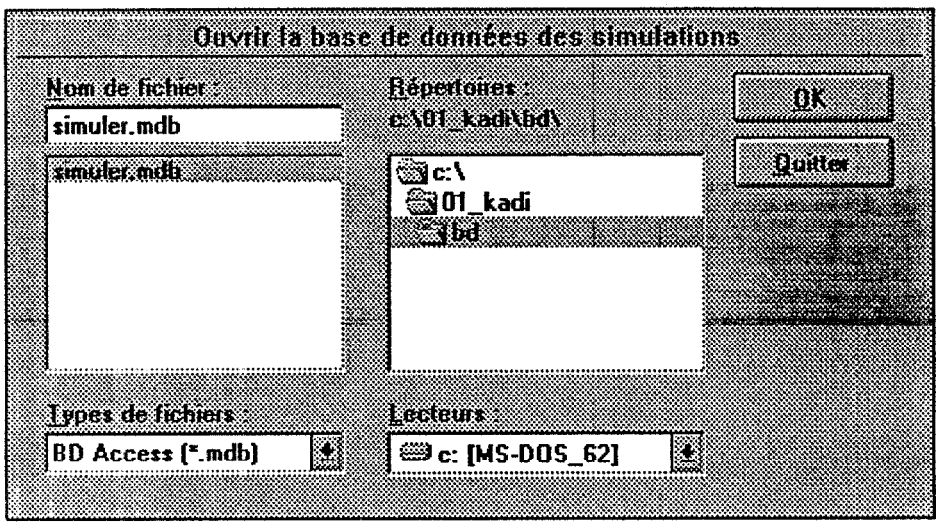
3.6.1.2 Description des relations

Code	Description
A	Un même type de sol peut faire l'objet de plusieurs observations.
B	Cette relation permet de connaître les differentes observation:; réalisées pour une variété de culture donnée.
C	Une site est constituée d'un ou plusieurs types de sol.
D	Les observations concernent une site bien précise.
E	Sur une même site sont relévées jounalièrement des températures.
F	Sur une site, sont cultivées une ou plusieurs variétés de cultures.
G	Un département est formé de plusieurs sites.
H	Une variété de culture admet des paramètres.
I	Une culture est formée de plusieurs variétés de cultures.
J	Une culture présente une ou plusieurs fonctins de modélisation.
K	Une fonction est formée entr autre de variables calculées.

3.6.2 Description générale de l'application (Autoparamétrisation)

Pour chaque fenêtre les boutons grisées ne sont pas accessibles à l'instant donné.

3.6.2.1 Fenêtre de recherche de la base de données des simulations

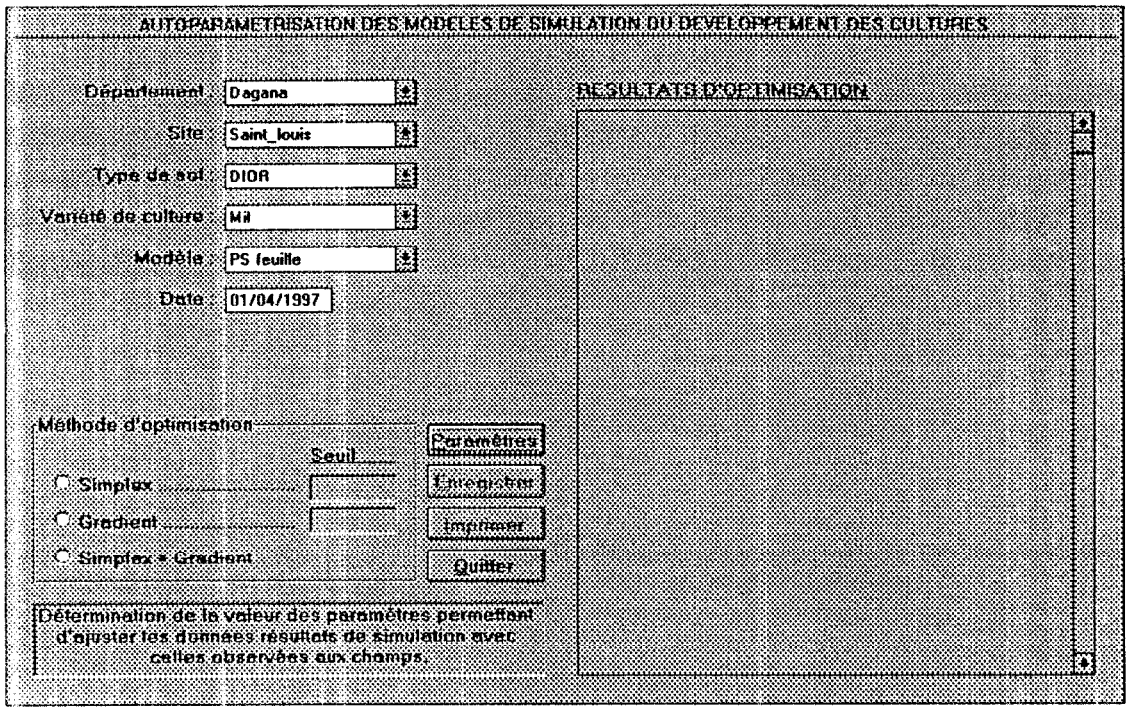


I.H.M. n°1

Au démarrage de l'application, cette boîte de dialogue s'affiche. Elle permet d'obtenir le chemin d'accès à la base de données des simulations où sont stockées toutes les données nécessaires à l'application d'autoparamétrisation. Elle doit être de type MS Access 2.0 ou compatible.

3.6.2.2 Fenêtre de saisie

Cette fenêtre s'affiche une fois la base de données des simulations trouvée.



I.H.M. n°2

Elle permet les traitements suivants :

1 - Le choix de la fonction de modélisation d'une variété de plante, cultivée sur un type de sol bien défini, d'un site, d'un département donné.

2 - La saisie de la date de paramétrisation.

3 - La modification de la limite inférieure et supérieure de chaque paramètre de la fonction de modélisation choisie : bouton **Paramètres.**

4 - Le choix de la méthode de détermination de la valeur des paramètres du modèle avec précision de la valeur d'arrêt **: choix de la méthode d'optimisation.**

5 - Le stockage des valeurs résultats d'optimisation des paramètres dans la table Tparamètres : bouton **Enregistrer.**

6 - L'affichage des valeurs résultats d'optimisation des paramètres et du coefficient de détermination sur l'écran **(RESULTATS D'OPTIMISATION) ou** sur une feuille papier (Bouton **Imprimer**).

7 - Quitter l'application d'autoparamétrisation : un clic de la souris sur le bouton **Quitter** ou une frappe simultanée des touches Alt et Q permettent de quitter l'application.

3.6.3 Description des différents traitements

3.6.3.f Le choix de la fonction de modélisation

La liste des fonctions de modélisation est obtenue après un choix :

- du nom du département où se situe la variété de culture
- d'un site de ce département
- d'un type de sol sur lequel se développe la variété de culture
- de la variété de culture concernée par le traitement

Toutes ces données sont définies dans la base de données des simulations. Elles sont présentées sur l'écran sous forme de liste déroulante.

Au départ la liste de tous les départements est accessible. Le choix d'un département enclenche l'accès à tous ses sites. Le choix d'un site permet a l'utilisateur de saisir un type de sol qui s'y trouve, une variété de culture ainsi qu'une fonction de modélisation.

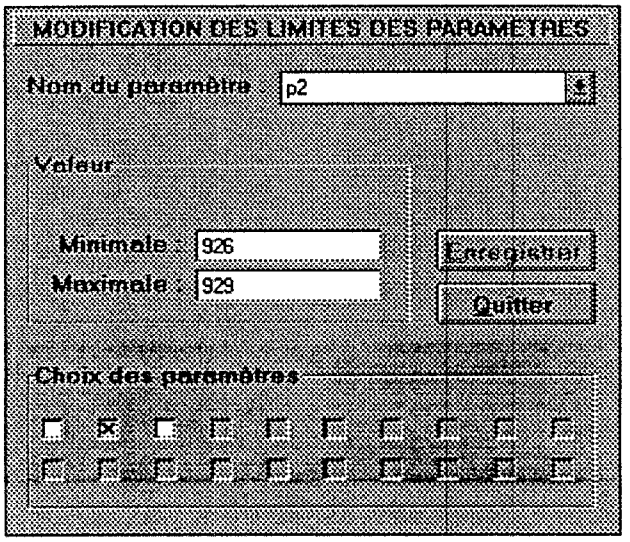
La recherche de la fonction de modélisation, dans la base de données des simulations, est effectuée à partir du nom du modèle choisi, de la variété de culture dont il faut déterminer la valeur des paramètres, du type de sol sur lequel elle est cultivé, du site ainsi que du département où elle se situe.

3.6.3.2 La saisie de la date de paramétrisation

Cette date permet de localiser dans la base de données des simulations, les données à prendre en compte dans la table Tobservations. Par défaut, elle correspond à la date du jour.

3.6.3.3 La modification de la limite inférieure et supérieure des paramètres

Un clic de la souris sur le bouton **Paramètres** ou une frappe simultanée des touches Alt et P permet l'affichage de la fenêtre suivante :



I.H.M. n°3

Le paramètre dont on veut modifier les bornes inférieures et supérieures est choisi soit à partir de la liste déroulante de **Nom du paramètre** soit en cochant une des cases non grisée (dont chacune définit un paramètre du modèle) de l'ensemble **Choix des paramètres.** Après, les valeurs minimale et maximale du paramètre choisi s'affichent en vue d'une modification.
Un clic de la souris sur le bouton **Enregistrer** ou une frappe simultanée des touches Alt et E permet de stocker les nouvelles limites du paramètres dans les champs Valeur min et Valeur max de la table TParamètres de la base de données des simulations.
Un clic de la souris sur le bouton **Quitter** ou une frappe simultanée des touches Alt et Q entraîne la fermeture de cette fenêtre et un retour à celle de la saisie.

3.6.3.4 La détermination de la valeur optimale des paramètres

Trois méthodes sont proposées pour déterminer la valeur des paramètres du modèle :

- **La méthode du simplex :** C'est une méthode très rapide mais pas très précise au niveau du minimum. Elle est convenable pour une première estimation. Elle permet ainsi d'obtenir le meilleur jeu de paramètres à la limite de la condition d'arrêt **(Seuil Simplex).** Elle converge très lentement au niveau de l'optimum. Cette méthode est à utiliser pour des problèmes complexes et dont l'espace d'exploration est très large.

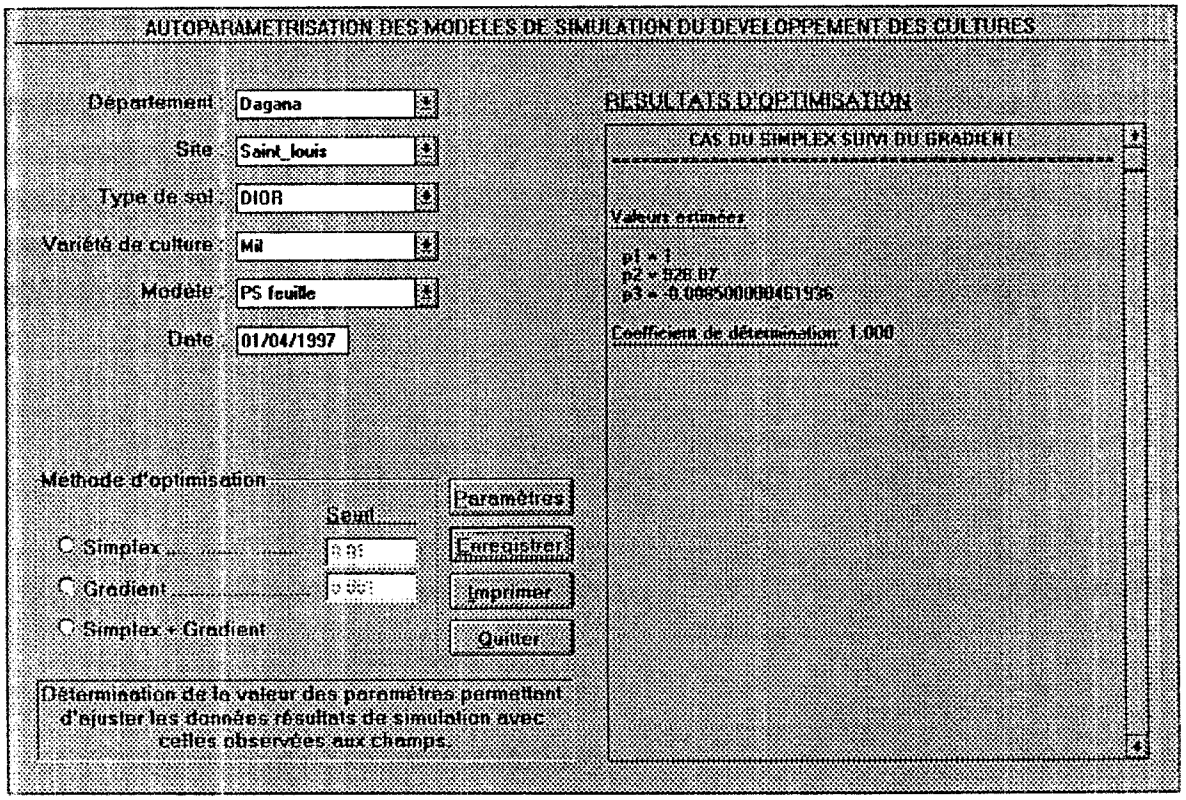
- **La méthode du gradient :** Cette méthode est lente au départ de l'optimisation et rapide à la fin. Elle donne, plus rapidement par rapport à la méthode simplex, aux alentours du minimum, un résultat satisfaisant. Cette méthode est à utiliser pour des problèmes moins complexes. La valeur exprimant la limite de la réalisation de cette méthode est **Seuil Gradient.**

- **La méthode du simplex suivie de celle du gradient :** Le jeu de paramètres obtenu avec la méthode du simplex est précisé par le gradient pour l'obtention rapide d'un résultat meilleur. Avec un bon paramétrage de **Seuil Simplex** et **Seuil Gradient,** cette manière d'optimiser donne un meilleur résultat.

*3.6.3.5 Le stockage des résultats d'optimisation dans la base table **Tparamètres***

Un clic de la souris sur le bouton **Enregistrer** ou une frappe simultanée des touches Alt et E permet de stocker les nouvelles valeurs des paramètres obtenues dans le champ Valeur de la table TParamètres de la base de données des simulations.

3.6.3.6 L'affichage du résultat de l'aufoparamétrisation



I.H.M. n°2'

Les valeurs des paramètres obtenues après optimisation sont affichées à l'écran. Le coefficient de détermination est une grandeur statistique intervenant dans la prise de décision au niveau des valeurs obtenues pour chaque paramètre. Cette prise de décision peut amener l'utilisateur à soit:

- choisir une autre méthode d'optimisation,
- changer les valeurs minimale et/ou maximale d'un ou plusieurs paramètres.

Un clic de la souris sur le bouton **imprimer** ou une frappe simultanée des touches Alt et I dirige les résultats de l'optimisation vers une imprimante.

3.7 SPECIFICA TIONS TECHNIQUES D'INTERFACE

L'application sera réalisée avec un ordinateur de type Pentium à 100 MHZ ayant 1,5 GIGA d'espace disque et 16 MO de mémoire vive.
Le développement se fera avec Visual Basic 4.0 sous Windows 3. Il. La base de données des simulations est réalisées avec ACCESS 2.0 (Système de gestion de base de données relationnelles) ou compatible.

4. CONCEPTION

Date d'édition : 08/05/1997.

4.7 INTRODUCTION

Autoparamétrisation des modèles de simulation du développement des cultures.

Ce projet consiste en la mise en place d'une application permettant de déterminer la valeur des paramètres des modèles de simulation du développement des cultures, afin d'ajuster les données résultats de simulation à celles observées aux champs.
Le problème revient à minimiser une fonction à n variables. Cette fonction exprimant l'écart entre les données de sortie de simulation et celles observées aux champs.

Normes et standards suivis

La méthodologre de développement est celle du cycle en V (Voir le dossier Plan de développement). Visual Basic 4.0 est utilisé pour la programmation
Les modifications sont apportées à la base de données des simulations sous ACCESS 2.0.

4.2 DOCUMENTS EN REFERENCE

- Cahier des charges
- Plan de développement
- Spécifications fonctionnelles

4.3 GLOSSAIRE

Paramètre : Quantité variable d'une fonction mathématique.
Jeu de paramètres : les différents paramètres d'une fonction mathématique.
Evénement : un événement est provoqué par l'utilisateur (clic sur un bouton, frappe d'une touche) ou par un système (minuterie). II appelle la procédure qui lui est rattachée.
Objet : un objet peut-être une feuille, un contrôle, l'écran, l'application, etc.
Point : Jeu de paramètres.

4.4 DESCRIPTION DES OUTILS DE DEVELOPPEMENT

4.4.1 Visual basic 4.0

C'est un systéme de développement en BASIC (Beginners All purpose Symbolic Instruction Code) regroupant l'éditeur de ressources, la boîte à outils, et l'environnement de programmation en une seule interface. La programmation est de type événementielle ; il n y a pas de programme principal, mais plutôt des objets, des événements.
Avec ce système, les I.H.M. (Interface Homme-Machine) sont vite réalisées. Le programme final est facilement transformé en fichier EXE (exécutable). Visual Basic 4.0 permet :

- l'appel aux DLL, bibliothèques de liens dynamiques,
- la programmation de graphismes grâce aux fonctions GDI (Graphics Device, interface),
- les échanges dynamiques de données DDE,
- la liaison et l'intégration d'objet (OLE),
- l'utilisation du presse-papiers (Clipboard),
- l'intégration des fichiers d'aide de Windows.

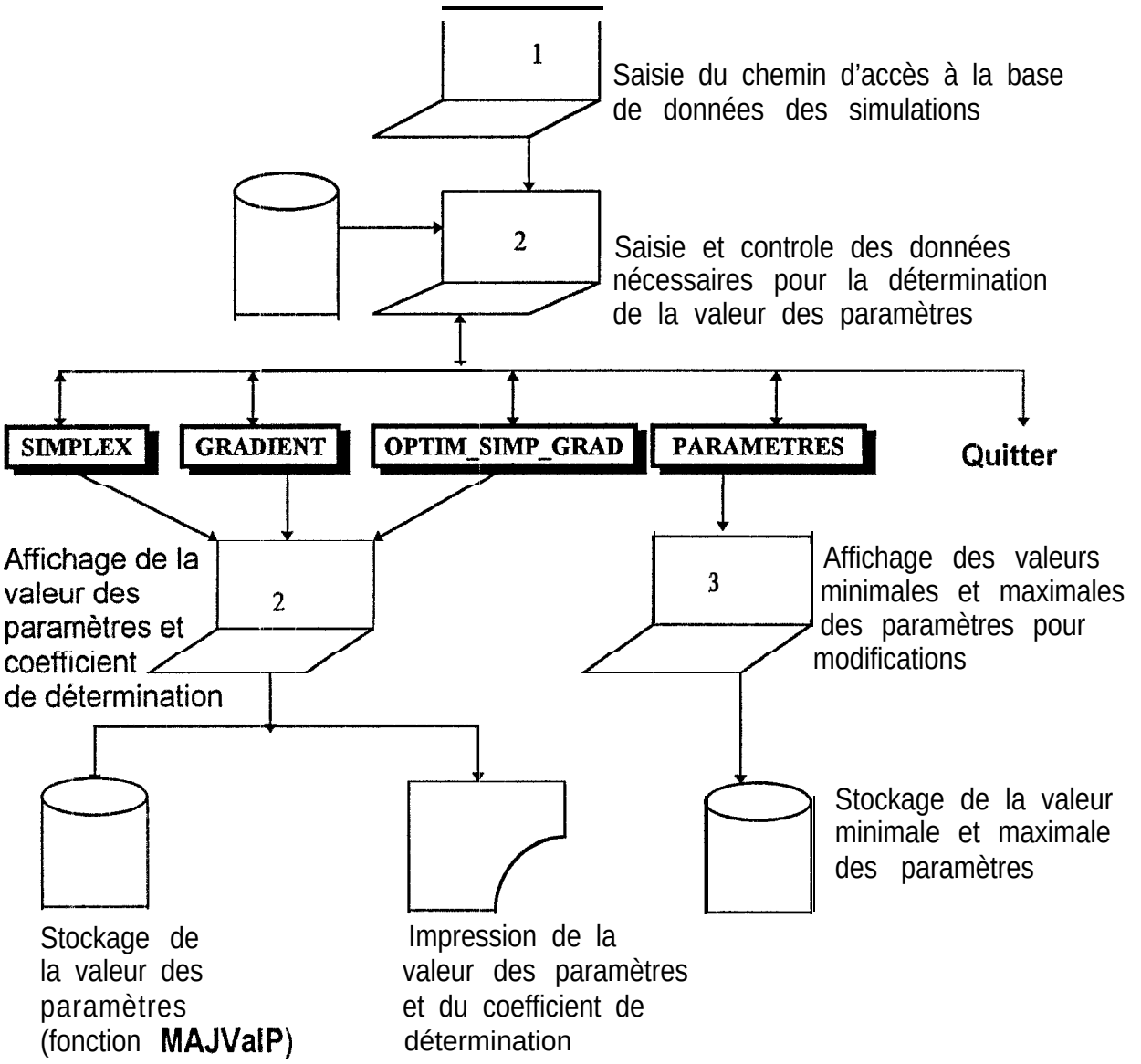
4.4.2 Access 2.0

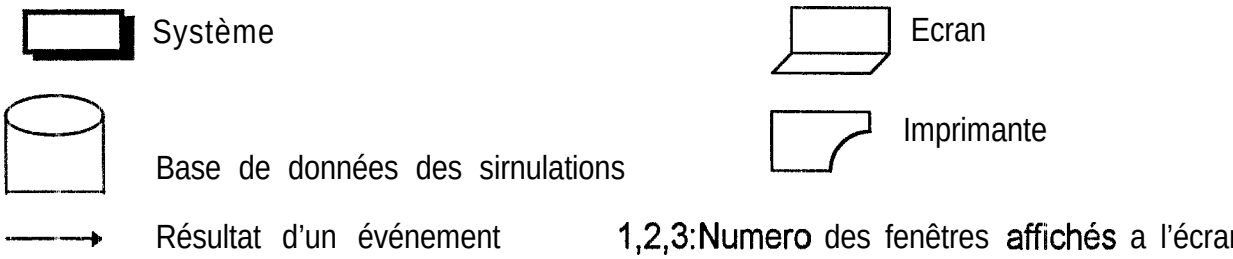
C'est un système de gestion de bases de données relationnelles. II fait partie de Microsoft Office, famille de produits conçus pour être ensemble. II bénéficie donc de procédures d'échanges de données avec de nombreuses applications comme Visual Basic. ACCESS dispose de nombreux outils de manipulation des données de la base très pratiques et très simples permettant d'effectuer facilement certaines opérations. Le langage de programmation est le Basic.
C'est le gestionnaire de bases de données utilisé au CERAAS.

4.5 CONCEPTION DETAILLEE

4.51 L'application d'autoparamétrisation

4.5.1.1 Architecture générale





4.5.1.2 Description

Problématique

Obtenir les valeurs des paramètres, d'une fonction de modélisation donnée: permettant de minimiser l'écart entre les données résultats de simulation et celles mesurées aux champs.

Analyse du problème

Les valeurs optimisées des paramètres d'une fonction de modélisation donnée sont déterminées soit par :

- la méthode du gradient,
- !a méthode du simplex,
- la méthode du simplex suivie de celle du gradient.

Par la suite elles sont affichées à l'écran. Suivant le choix de l'utilisateur, elles peuvent être stockées dans la table Tparamètres et/ou imprimées.

Algorithme général de l'application

DEBUT

Saisie à partir de la fenêtre n°1, de la base de données des simulations.
Ouverture de la base de données des simulations.
Récupération de tous fes noms de départements, de sites, de types de sol,, de variétés de cultures et de fonction de modélisation.
Saisie à partir de la fenêtre n°2 du nom du département, du site, du type de sol, de la variété de culture et de la fonction du modélisation (choix de l'utilisateur), la date
Fermeture de la base de données des simulations.
Appel de CONTFORMDAT
SI valide *ALORS*
Attendre le choix de l'utilisateur
TANT QUE le choix est différent de **Quitter** *FAIRE*
Cas du choix d'une méthode d'optimisation
SI choix de la méthode d'optimisation **= Simplex** *ALORS*
Saisie de **Seuil Simplex**
Appel du système SIMPLEX : les résultats sont affichés à l'écran n°2
FIN SI
SI choix de la méthode d'optimisation **= Gradient** *ALORS*
Saisie de **Seuil Gradient**
Appel du système GRADIENT : les résultats sont affichés à l'écran n°2
*FIN **SI***
SI choix de la méthode d'optimisation **= Simplex+Gradient** *ALORS*
Saisie de **Seuil Simplex** et de **Seuil Gradient**
Appel du système OPTIM_SIMP_GRAD : **les** résultats sont affichés à l'écran n°2
FIN SI

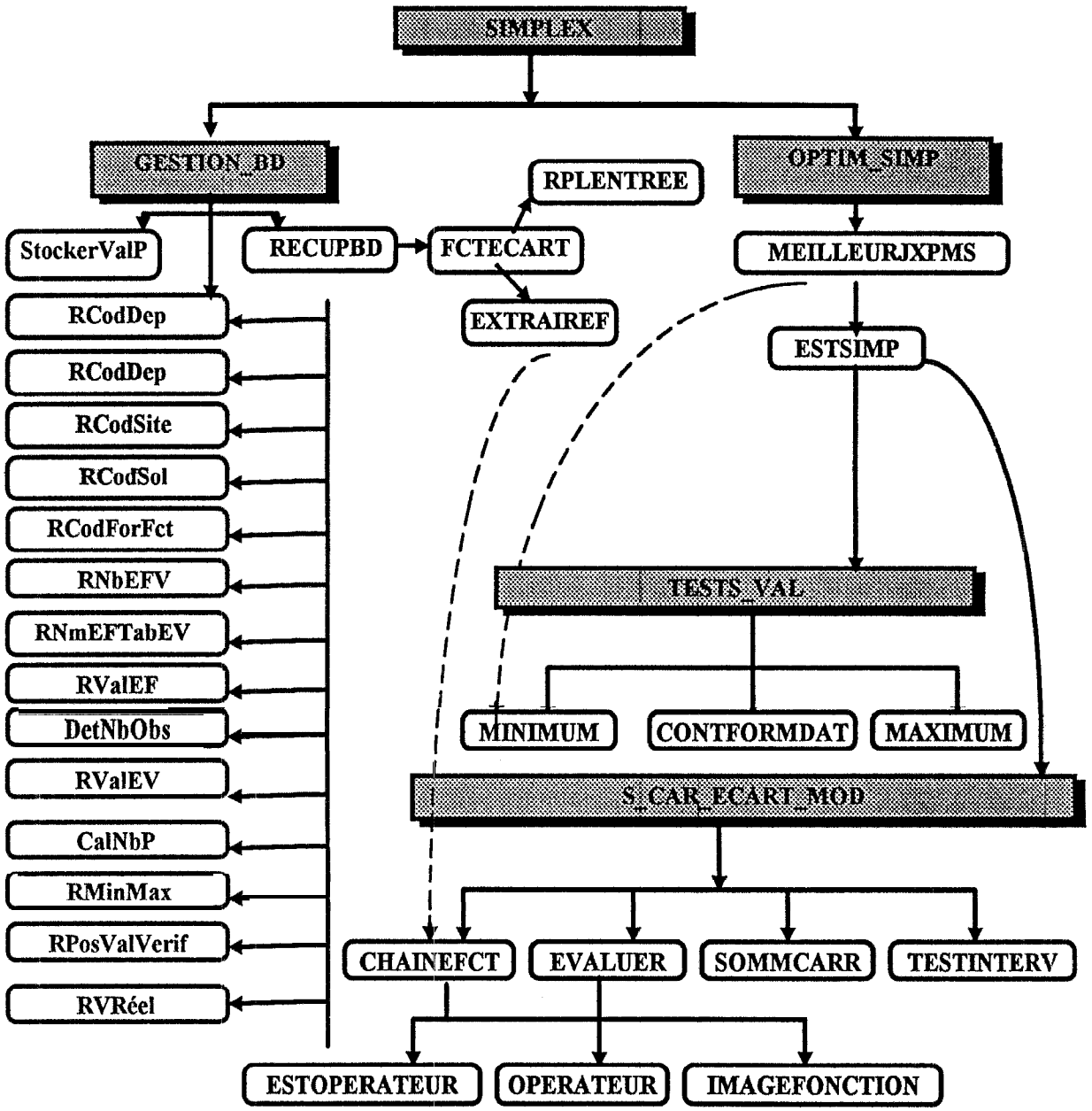
FIN Cas
SI choix = **Paramètres** *ALORS*
Appel du système PARAMETRES : les résultats sont affichés à l'écran n°3
FIN SI
SI choix = **Enregistrer** *ALORS* Appel de MAJValP *FIN* ***SI***
SI choix = **Imprimer** *ALORS*
Diriger les résultats vers une imprimante
FIN SI
FIN ***TANT*** *QUE*
Fermer l'application d'autoparamétrisation
SINON
Afficher un message d'erreur
FIN SI
FIN

Fonctions ou systèmes appelés :
SIMPLEX
GRADIENT
OPTIM_SIMP_GRAD
MAJValP

Fonctions ou modules appelants: Néant.

4.5.2 Le système SIMPLEX

4.5.2.1 Architecture générale



Module

Fonction / Procédures

Appel à **certaines fonctions d'un autre module**

Appel à une fonction d'un autre module

4.5.2.2 Description des modules

4.5.2.2.1 SIMPLEX

Problématique

Déterminer des paramètres perrnettant de minimiser l'écart entre les données résultats de simulation et celles mesurées aux champs avec la méthode du simplex.

Analyse du problème

La précision des modèles de diiveloppement de cultures dépend des valeurs des paramètres contrôlant leurs réponses aux conditions de l'environnement, Ces modèles sont représentés sous forme de fonction mathématique.

La méthode des moindres carres est utilisée pour le calcul des paramétres. Elle consiste à mettre en place et à minimiser une fonction exprimant l'écart entre les valeurs observées sur le terrain et celles obtenues avec les modèles de simulation.

L'optimisation revient à la recherche de paramètres permettant de minirniser la fonction des moindres carrés.

La méthode du simplex permet d'obtenir rapidement un premier jeu de paramètres.

Algorithme sous forme de texte

DEBUT

Appel de RECUPBD.
Appel de MEILLEURJXPMS.
Appel de StockerValP.
Affichage de la valeur optimisée de chaque paramètre et du coefficient de détermination dans **RESULTATS D'OPTIMISATION** à l'écran.

FIN

Fonctions ou procédures appelées :

RECUPBD du module GESTION-BD
MEILLEURJXPMS du module OPTIM_SIMP
StockerValP du module GESTION-BD

Fonctions ou procédures appelantes: Néant.

Paramètres en entrée

NCulture : Nom de la variété de culture dont on veut calculer les paramètres.
NDep : Nom du départernent.
NSite : Nom du site.
NSol : Type de sol.
NFonction : Nom de la fonction de modélisation.
BaseParam : Base de données des simulations.
SeuilErr : Valeur indiquant la condition d'arrêt de l'algorithme du simplex.

Paramètres en sortie

CD : Coefficient de détermination.
XMeilS(NbParam) : Tableau à une dimension contenant le meilleur jeu de paramètres obtenu.

Paramétres intermédiaires

NbParam : Nombre total des paramètres du modèle.
NParam() : Tableau à une dimension contenant le nom de chaque paramètre du modèle.
Fct : Fonction des moindres carrés sous forme de chaîne de caractères.
St : Somme totale des valeurs de vérification.
CFct : code de la fonction du modèle.
NbreObS : Nombre total des observations.
SMeilS : Valeur de la fonction des moindres carrés avec le meilleur jeu de paramètres obtenu.
MinMax(NbParam, 2) : Tableau à deux dimensions contenant la valeur minimale et maximale de chaque paramètre du modèle.

Algorithme détaillé

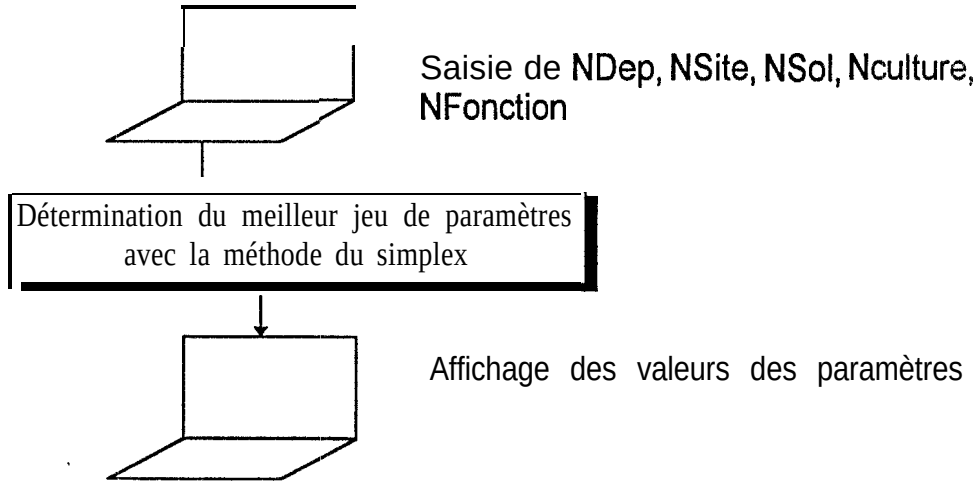
DEBUT

RECUPBD (NDep, NSite, NSol, NCulture, NFonction, NbParam, MinMax(), NParam(), Fct, St, NbreObS, CFct, CV, BaseParam)
MEILLEURJXPMS (SeuilErr, NbParam, MinMax(), XMeilS(), Fct, SMeilS)
CD = 1 - (SMeilS / St)
StockerValP (NbParam, XMeilS(), BaseParam)
Afficher les résultats à l'écran : SMeilS(), CD.

FIN

Jeu d'essai

PS (Poids sec) est une fonction de modélisation du mil, une culture du site de Saint-Louis dans le département de Diagana. Le mil est cultivé sur un sol de type DIOR.
Le meilleur jeu de paramètres obtenu est affiché à l'écran



4.5.2.2.2 GESTION BD

Ensemble des traitements liés à la base de données des simulations.

4.5.2.2.2.1 Procédure RECUPBD

Problématique

- Récupérer à partir de la base de données des simulations toutes les données nécessaires à la mise en place dle la fonction des moindres carrés.
- Mettre en place la fonction des moindres carrés.

Analyse du problème

Pour le calcul des valeurs des paramètres optimales, certaines valeurs de la base de données des simulations sont nécessaires. Ce sont : les valeurs minimale et maximale de chaque paramètre du modèle, la fonction de modélisation sous forme de chaîne de caractères, les valeurs observées sur le terrain, les variables d'entrée de la fonction de modélisation.
La recherche de ces données est effectuée à partir du nom de la variété de culture dont il faut déterminer la valeur des paramètres, du type de sol sui lequel elle est cultivé, du site ainsi que du département où elle se situe. La fonction des moindres carrés est ensuite mise en place.

Algorithme sous forme de texte

DEBUT

Ouverture de la base de données des simulations.
Appel de RCodDep.
Appel de RCodSite.
Appel de RCodSol.
Appel de RCodForFct.
Appel de RNbEFV.
Appel de RNmEFTabEV.
Appel de RRValEF.
Appel de DetNbObs.
Appel de RValEV.
Appel de CalNbP.
Appel de RMinMax.
Appel de RPosValVerif
Appel de RVRéel.
Fermeture de la base de données des simulations.
Appel de FCTECART.

FIN

Fonctions ou procédures appelées :

RCodDep
RCodSite
RCodSol
RCodForFct
RNbEFV
RNmEFTabEV
RValEF
DetNbObs
RValEV
CalNbP
RMinMax
RPosValVeri
RVRéel
FCTECART

Fonctions ou procédures appelantes : Néant.

Paramètres en entrée

DP : Nom du département sélectionné par l'utilisateur.
St : Nom du site sélectionné par l'utilisateur.
SI : Type de sol sélectionné par l'utilisateur.
Ct : Nom de la variété de culture sélectionnée par l'utilisateur.
NomFct : Nom de la fonction du modèle par l'utilisateur.
BaseParam : Base de données des simulations.

Paramètres en sortie

NbParam : Nombre de paramètres du modèle.
FONCTION : Expression de la fonction de modélisation sous forme de chaîne de caractères.
SVTot : Valeur totale des observations.
NbObserv : Nombre total des observations sur le terrain.
CodCulture : Code de la variété de culture choisi par l'utilisateur.
CodFct : Code de la fonction de modélisation.
NomParam() : Tableau contenant le nom de chaque paramètre du ma dèle.
MinMax(NbParam, 2) :Tableau à deux dimensions contenant la valeur minimale et maximale de chaque paramètre du modèle.

Paramètres intermédiaires

VariationsP : Base de données pour l'utilisation des différentes tables.
TravEspaceP : Espace de travail.
TParamètres : Table Tparamètres.
TFonct : Table TFonctions.
TVérif : Table TObservations.
TEntréeF : Table TVariété.
TEntréeV : Table TVariables Calculées.
TDépart : Table TDépartement.
TSites : Table TSite.
TSols : Table TSol.
TabCalcul : Table contenant la valeur des entrées variables.

FModèle : Fonction de modélisation.
CodDep : Code du département choisi par l'utilisateur.
CodSol : Code du type de sol choisi par l'utilisateur.
CodSite : Code du site choisi par l'utilisateur.
NbFEntrées : Nombre de valeurs d'entrées fixes.
NbVEntrées : Nombre de valeurs d'entrées variables.
PosNomFct : Position du champ d'observation dans la table TVérif.
Réelles(NbObserv) : Tableau des valeurs réelles recueillies sur le terrain.
PosCalcul : Position du champ contenant la valeur des entrées variables dans TabCalcul.
NEntréeesF(NbFEntrées) : Tableau du nom des entrées fixes du modèle.
VEntréeesF(NbFEntrées) : Tableau de la valeur des entrées fixes du modèle.
NEntréeesV(NbVEntrées) : Tableau du nom des entrées variables du modèle.
NomTabCalcul(NbVEntrées) : Nom de la table où se trouve les valeurs de l'entrée variable.
CléTabCalcul(NbVEntrées) : clé d'accès à la table des valeurs de l'entrée variable.
VEntréeesV(N bVEntrées, NbObserv) : Tableau de la valeur des entrées variables du modèle.

Algorithme détaillé

DEBUT

Mise en place TravEspaceP
Ouverture de VariationsP
Ouverture de TParamètres, TFonct, TVérif, TEntréeF, TEntréeV, TDépart, TSites, Tsols.
'~~ Récupération du code du département choisi
RCodDep (TDépart, Dp, CodDep)
'~~ Récupération du code du site choisi
RCodSite (TSites, St, CodSite)
'~~ Récupération du code du type de sol choisi
RCodSol (TSols, Sl, CodSol)
'-- Récupération du code et de la formulation la fonction de modélisation et du code de la variété de culture
RCodForFct (TEntréeF, Ct, CodCulture, TFonct, NomFct, FModèle, CodFct)
'~~ Récupération du nombre des entrées fixes et variables du modèle
RNbEFV (NbFEntrées, NbVEntrées, TEntréeV, CodFct)
'~~ Récupération du nom des entrées fixes et variables
'--- et du nom de la table où sont stockéees les entrées variables ainsi que la clé d'accès
RNmEFTabEV (TEntréeV, CodFct, NEntréesF(), NEntréesV() NomTabCalcul(), CléTabCalcul())
'~~ Récupération de la valeur dle chaque entrée fixe
RValEF (TEntréeF, CodCulture, NEntréesF(), VEntréesF(), NbFEntrée.s)
'~~ Détermination de NbObserv
DetNbObs (NbObserv, TVérif, CodSite, CodCulture, CodSol)
'~~ Récupération de la valeur dle chaque entrée variable
RValEV (NbVEntrées, VariationsP, NomTabCalcul(), NEntréesV(), VEntréesV(), CléTabCalcul())
'~~ Calcul de NbParam

CalNbP (TParamètres, CodCulture, CodFct, NbParam)
'~~ Récupération de la valeur rninimale et maximale de chaque paramètre du modéle
RMinMax (TParamètres, CodCulture, CodFct, MinMax(), NomParam())
'~~ Détermination de la position de la valeur à vérifier
RPosValVerif (PosNomFct, TVérif, NomFct)
'~~ Récupération des valeurs réelles recueillies aux champs
RVRéel (Réelles(), TVérif, CodSite, CodCulture, CodSol, PosNomFct)
Fermeture de TFonct, TEntréeF, TEntréeV, TDépart, TSites, TSols, TVérif, TParamètres, VariationsP, TravEspaceP
'~~ Appel à la procédure de mise en place de FONCTION
FCTECART NbObserv, Réelles(), NbFEntrées, NEntréesF(), VEntréesF(), NbVEntrées, NEntréesV(), VEntréesV(), FModèle, FONCTION
FIN

4.5.2.2.2.2 Procédure RCodDep

Problématique

Récupérer à partir de la table TDépartement de la base de données des simulations, le code correspondant au nom du département choisi par l'utilisateur.

Fonctions ou procédures appelées : Néant

Fonctions ou procédures appelantes : RECUPBD

Paramètres en entrée

TD : Table TDépartement.
ND : Nom du département dont on veut avoir le code.

Paramètres en sortie

CD : Code du département.

Algorithme

DEBUT
Se positionner sur le premier enregistrement de TD
TANT QUE pas fin du fichier TD ***FAIRE***
SI « Nom département » = ND ***ALORS***
CD = « Code département »
Sortir de la boucle
FIN SI
Passer à l'enregistrement suivant de TD
FIN TANT QUE
FIN

4.5.2.2.2.3 Procédure RCodSite

Problématique

Récupérer à partir de la table TSite de la base de données des simulations, le code correspondant au nom du site choisi par l'utilisateur.

Fonctions ou procédures appelées : Néant

Fonctions ou procédures appelantes : RECUPBD

Paramètres en entrée

TS : Table TSite.
NS : Nom du site dont on veut avoir le code.

Paramètres en sortie

CS : Code du site.

Algorithme

```
DEBUT
    Se positionner sur le premier enregistrement de TS
    TANT QUE pas fin du fichier TS FAIRE
        SI « Nom site » = NS ALORS
            CS = « Code site »
            Sortir de la boucle
        FIN SI
        Passer à l'enregistrement suivant de TS
    FIN TANT QUE
FIN
```

4.5.2.2.2.4 Procédure RCodSol

Problématique

Récupérer à partir de la table TSol de la base de données des simulations, le code correspondant au nom du type de sol choisi par l'utilisateur.

Fonctions ou procédures appelées : Néant

Fonctions ou procédures appelantes : RECUPBD

Paramètres en entrée

TSI : Table TSol.
NSI : Nom du type de sol dont on veut avoir le **code.**

Paramètres en sortie

CSI : **Code** du type de sol.

Algorithme

DEBUT
Se positionner sur le premier enregistrement de TSI
TANT QUE pas fin du fichier TSI *FAIRE*
SI « Nom sol » = NSI *ALORS*
CSI = « Code sol »
Sortir de la boucle
FIN SI
Passer à l'enregistrement suivant de TSI
FIN TANT QUE
FIN

4.5.2.2.2.5 Procédure RCodF orFct

Problématique

- Récupérer le code de la variété de culture dont le nom est choisi par l'utilisateur.
- Récupérer à partir de la table TFonction de la base de données des simulations, le code de la fonction de modélisation choisi par l'utilisateur ainsi que sa formulation. Cette récupération se fait à partir du code de la culture ainsi que de celui de la variété de culture.

Fonctions ou procédures appelées : Néant

Fonctions ou procédures appelantes : RECUPBD

Paramètres en entrée

TEF : Table TVariété.
NV : Nom de la variété de culture choisie par l'utilisateur.
TF : Table TFonction
NF : Nom de la fonction de modélisation choisie par l'utilisateur

Paramètres en sortie

CC : Code de la variété de culture choisi par l'utilisateur.
Fm : Formulation de la fonction de modélisation.
CF : Code de la fonction de modélisation.

Algorithme

```
DEBUT
    Se positionner sur le premier enregistrement de TEF
    TANT QUE pas fin du fichier TEF FAIRE
        SI « Nom variété » = NV ALORS
            CC = « Code variété » de TEF
            Se positionner sur le premier enregistrement de TF
            TANT QUE pas fin du fichier TF FAIRE
                SI « Code culture » de TF = « Code culture » de
                  TEF et « Nom fonction » de TF = NF ALORS
                    Fm = « Fonction » de TF
                    CF = « Code fonction » de TF
                    Sortir de la boucle
                FIN SI
                Passer à l'enregistrement suivant de TF
            FIN TANT QUE
            Sortir de la boucle
        FIN SI
        Passer à l'enregistrement suivant de TEF
    FIN TANT QUE
FIN
```

4.5.2.2.2.6 Procédure RNb EFV

Problématique

Récupérer le nombre total des entrées fixes et variables du modèle à partir du code de la fonction de modélisation.

Fonctions ou procédures appelées : Néant

Fonctions ou procédures appelantes : RECUPBD

Paramètres en entrée

TEV : Table Tvariables calculées.
CF : Code de la fonction de modélisation.

Paramètres en sortie

NbFE : Nombre total des entrées variables du modèle.
NbVE : Nombre des entrées variables du modèle.

Algorithme

DEBUT
NbFE = 0
NbVE = 0
Se positionner sur le premier enregistrement de TEV
TANT QUE pas fin du fichier TEV ***FAIRE***
SI « Code fonction » = CF ***ALORS***
SI « Type variable » = F ALORS
NbFE = NbFE + 1
SINON
NbVE = NbVE + 1
FIN SI
FIN SI
Passer à l'enregistrement suivant de TEV
FIN TANT QUE
FIN

4.5.2.2.2.7 Procédure RNmEFTabEV

Problématique

Récupérer le nom de chaque entrée fixe et variable ainsi que celui des tables où se trouvent les valeurs des entrées variables du modèle.

Fonctions ou procédures appelées : Néant

Fonctions ou procédures appelantes : RECUPBD

Paramètres en entrée

TEV : Table Tvariables calculées.
CF : Code de la fonction de modélisation.

Paramètres en sortie

NEF() : Tableau contenant le nom des entrées fixes du modèle..
NEV() : Tableau contenant le nom des entrées variables du modèles
NTabC() : Tableau contenant le nom des tables où se trouvent les valeurs des entrées variables.
CléTabC() : Tableau contenant la clé d'accès aux tables des entrées variables.

Paramètres intermédiaires

I : Indice de parcours de NEF().
J : Indice de parcours de NEV().

Algorithme

```
DEBUT
        I = 0
        J= 0
        Se positionner sur le premier enregistrement de TEV
        TANT QUE pas fin du fichier TEV FAIRE
                SI « Code fonction » = CF ALORS
                        SI « Type variable » = F ALORS
                                NEF(I) = « Nom variable »
                                I = I + 1
                        SINON
                                NEV(J) = « Nom Variable »
                                NTabC(J )= « Type variable »
                                CléTabC (J) = « Clé »
                                J = J + 1
                        FIN SI
                FIN SI
                Passer à l'enregistrement suivant de TEV
        FIN TANT QUE
FIN
```

4.5.2.2.2.8 Procédure RValEF

Problématique

Récupérer la valeur de chaque entrée fixe du modèle à partir du code de la variété de culture.

Fonctions ou procédures appelées : Néant

Fonctions ou procédures appelantes : RECUPBD

Paramètres en entrée

TEF	: Table Tvariables calculées.
CC	: Code de la variété de culture.
NEF()	: Tableau contenant le nom des entrées fixes du modèle.
NbFE	: Nombre des entrées fixes du modèle.

Paramètres en sortie

VEF()	: Tableau contenant la valeur de chaque entrée fixe du modèle

Paramètres intermédiaires

I, J	: indices de parcours de boucle.

Algorithme

```
DEBUT
    Se Positionner sur le premier enregistrement de TEF
    TANT QUE Pas fin du fichier TEF FAIRE
        SI « Code variété » = CC ALORS
            POUR I = 0 à NbFE - 1
                J = 0
                TANT QUE le nom du champ J <> NEF(I) FAIRE
                    J = J + 1
                FIN TANT QUE
                VEF(I) = valeur contenue dans le champ
            FIN POUR
            Sortir de la boucle
        FIN SI
        Passer à l'enregistrement suivant de TEF
    FIN TANT QUE
FIN
```

4.5.2.2.2.9 Procédure DetNbObs

Problématique
Déterminer le nombre total des observations réalisées pour un modèle donné à partir du code du site, de la variété de culture et du type de sol.

Fonctions ou procédures appelées : Néant

Fonctions ou procédures appelantes : RECUPBD

Paramètres en entrée
TV : Table TObservations.
CS : Code du site concerné.
c c : Code de la variété de culture.
CSI : Code du type de sol.

Paramètres en sortie
NbObs : Nombre total des observations réalisées.

Algorithme
DEBUT
NbObs = 0
Se positionner sur le premier enregistrement de TV
TANT QUE pas fin du fichier TV ***FAIRE***
SI (« Code site » = CS) et (« Code variété » = CC)
et (« Code sol » = CSI) ALORS
NbObs = NbObs + 1
FIN SI
Passer à l'enregistrement suivant de TV
FIN TANT QUE
FIN

4.5.2.2.2.10 Procédure RValEV

Problématique
Récupérer la valeur des entrées variables du modèle à partir de leur nom, les tables où elles se trouvent et de la clé d'accès à ces tables.

Fonctions ou procédures appelées : Néant

Fonctions ou procédures appelantes : RECUPBD

Paramètres en entrée
NbVE : Nombre total des entrées variables du modèle.
VP : Base de données des simulations.
NEV() : Tableau contenant le nom des entrées variables.
NTabC() : Tableau contenant le nom des tables où se trouvent les valeurs des entrées variables.
CléTabC() : Tableau contenant la clé d'accès aux tables des entrées variables.

Paramètres en sortie
VEV() : Tableau contenant la valeur des entrées variables.

Paramètres intermédiaires
I, J, K : Indices de parcours de tableau et de boucle.
TabC : Table contenant les valeurs d'une entrée variable.

Algorithme

```
DEBUT
    POU?? I=0 à NbVE - 1
        TabC = NTabC(I)
        Ouverture de TabC
        Se positionner sur le premier champ de données de TabC, J = 0
        TANT QUE on n'est pas arrivé au dernier champ de TabC FAIRE
            SI le nom dlu champ = NEV(I) ALORS
                Sortir de la boucle
            FIN SI
            J = J + 1
        FIN TANT QUE
        Se positionner sur le premier enregistrement de TabC
        K=O
        TANT QUE pas fin du fichier TabC FAIRE
            SI « Clé » = CléTabC (I) ALORS
                VEV(I, K) = la valeur du champ J, K = K + 1
            FIN SI
            Passer à l'enregistrement suivant de TabC
        FIN TANT QUE
        Fermeture de TabC
    FIN POUR
FIN
```

4. 5.2.2.2.7 1 Procédure CalNb P

Problématique

Calculer la nombre de paramètres du modèle par l'intermédiaire du code de la variété de culture et de celui de la fonction de modélisation.

Fonctions ou procédures appelées : Néant

Fonctions ou procédures appelantes : RECUPBD

Paramètres en entrée

TP : Table TParamètres.
CC : Code de la variété de culture.
CF : Code de la fonction du modèle.

Paramètres en sortie

NbP : Nombre de paramètres du modèle.

Algorithme

DEBUT

NbParam = 0
Se positionner sur le premier enregistrement de TP
TANT QUE pas fin du fichier TP ***FAIRE***
SI (« Code variété » = CC) et (« Code fonction » = CF) ***ALORS***
NbP = NbP + 1
FIN Si
Passer à l'enregistrement suivant de TP
FIN TANT QUE

FIN

4.5.2.2.2.12 Procédure RMinMax

Problématique

Récupérer le nom ainsi que la valeur minimale et maximale de chaque paramètre du modèle dans la table Tparamètres de la base de données des simulations par l'intermédiaire du code de la variété de culture et de celui de la fonction de modélisation.

Fonctions ou procédures appelées : Néant

Fonctions ou procédures appelantes : RECUPBD

Paramètres en entrée

TP : Table TParamètres.
CC : Code de la variété de culture.
CF : Code de la fonction du modèle.

Paramètres en sortie

NomP() : Tableau contenant le nom de tous les paramètres du modèle.
MnMx() : Tableau contenant la valeur minimale et maximale de chaque paramètre du modèle.

Paramètres intermédiaires

I : Indice de parcours de tableau.

Algorithme

```
DEBUT
        I=0
        Se positionner sur le premier enregistrement deTP
        TANT QUE pas fin du fichier TP FAIRE
                SI (« Code variété! » = CC) et (« Code fonction » = CF) ALORS
                        MnMx(I,0) = « Valeur Min »
                        MnMx( I, 1) = « Valeur Max »
                        I = I + 1
                FIN SI
                Passer à l'enregistrement suivant de TP
        FIN TANT QUE
FIN
```

4.5.2.2.2.13 Procédure RPosValVerif

Problématique

Récupérer dans la table TObservations, la position du champ contenant les valeurs des observations effectuées aux champs en fonction du nom de la fonction de modélisation.

Fonctions ou procédures appelées : Néant

Fonctions ou procédures appelantes : RECUPBD

Paramètres en entrée

TV : Table TObservations.
NF : Nom de la fonction de modélisation.

Paramètres en sortie

PNFct : Position du champ contenant les valeurs des observations pour le modèle.

Paramètres intermédiaires

I : Indice de parcours de boucle.

Algorithme

```
DEBUT
        I = 0
        Se positionner sur le premier champ de TV
        TANT QUE on n'est pas arrivé au dernier champ de TV FAIRE
                SE le nom de la variable = NF ALORS
                        PNFct = I
                        Sortir de la boucle
                FIN SI
                I = I + 1
        FIN TANT QUE
FIN
```

4.5.2.2.2.14 Procédure RVRéel

Problématique

Récupérer les valeurs réelles recueillies sur le terrain pour le modèle par l'intermédiaire du code du site, du type de sol, de la variété de culture ainsi que de la date de visite.

Fonctions ou procédures appelées : Néant

Fonctions ou procédures appelantes : RECUPBD

Paramètres en entrée

TV : Table TObservations.
CS : Code du site.
CC : Code de la variété de culture.
CSI : Code du type de sol.
PNFct : Position du champ contenant les valeurs d'observations pour le modèle.

Paramètres en sortie

Réel() : Tableau contenant les valeurs réelles recueillies sur le terrain.

Paramètres intermédiaires

I : Indice de parcours de tableau.

Algorithme

```
DEBUT
    I=0
    Se positionner sur le premier enregistrement de TV
    TANT QUE pas fin du fichier TVérif FAIRE
        SI (« Code site » = CS) et (« Code variété » = CC)
          et (« Code sol » = CSI) et (« Date visite » <= Date) ALORS
              Réel(I) = la valeur du champ à la position PosNomVerif
              I = I + 1
        FIN SI
        Passer à l'enregistrement suivant de TV
    FIN TANT QUE
FIN
```

4.5.2.2.2.15 Procédure FCTECART

Problématique

Mettre en place la fonction des moindres carrés.

Analyse du problème

La fonction des moindres carrés exprime l'écart entre les valeurs observées sur le terrain « 0 » et celles obtenues avec la simulation. Pour un modèle de simulation f(X, P) où X est la matrice des variables d'entrées du modèle et P la matrice des paramètres, la fonction des moindres carrés S(P) est exprimée ainsi :

$$S(P) = \sum_{i=1}^{n} (Oi - f(Xi, P))^2$$

i = numéro de l'observation

Algorithme sous forme de texte

```
DEBUT
        Appel de RPLENTREE
        Ecart i = 0
        POUR i = 0 à nombre total d'observations - 1
                Ecart i-1 = Ecart i
                Appel de RPLENTREE
                Ecart i = Ecart i-1 + (Oi - f(Xi, P)^2)
        FIN POUR
        S(P) = Ecart i
FIN
```

Fonctions ou procédures appelées : RPLENTREE()

Fonctions ou procédures appelantes : RECUPBD()

Paramètres en entrée

Taille	: Nombre d'observations réalisées pour ce modèle.
Obs()	: Tableau des valeurs observées sur le terrain.
TentF	: Nombre des entrées fixes.
NEF()	: Tableau contenant le nom des entrées fixes.
VEF()	: Tableau contenant la valeur des entrées fixes.
TentV	: Nombre des entrées variables.
NEV()	: Tableau contenant le nom des entrées variables.
VEV()	: Tableau contenant la valeur des entrées variables.
F	: Fonction de modélisation.

Paramètres en sortie

FE	: Fonction des moindres carrés.

Paramètres intermédiaires

I	: Indice de parcours de boucle et de tableau.
J	: Indice de parcours de boucle et de tableau.
ChObs	: Valeur d'une observation.
FEntl	: F après remplacement de chaque entrée par sa valeur.
FEnt2	: Fonction réelle de modélisation.
Ecart(Taille)	: Tableau renfermant la formulation de l'écart pour chaque observation.

Algorithme

```
DEBUT
    FEntl = F
    POUR I = 0 à TentF - 1
        RPLENTREE (NEF(I)), VEF(I), FEntl)
    FIN POUR
    POUR I = 0 à Taille - 1
        FEnt2 = FEntl
        Mettre ChObs sous forme de chaîne de caractères
        Enlever les espaces de gauche et de droite de ChObs
        POUR J = 0 à TentV - 1
            RPLENTREE (NEV( 1)), VEV( 1), FEnt2)
        FIN POUR
        FEnt2 = EXTRAIREF (FEnt2)
        Ecart(I) = "("&"("& ChObs &"-"&"("& Fent2&")"& ")" & "^" & "2" & ")"
        Enlever les espaces de gauche et de droite de Ecart(I)
    FIN POUR
    FE = "(" & Ecart(O)
    Enlever les espaces de gauche et de droite de FE
    POUR I = 1 à Taille - 1
        FE = FE & "+" & Ecart(I)
        Enlever les espaces de gauche et de droite de FE
    FIN POUR
    FE = FE & ")"
    Enlever les espaces de gauche et de droite de FE
FIN
```

Jeu d'essai

La fonction de modélisation de PS (Poids sec) feuille du mil est :
PS feuille = Psmax/(a+b*exp(c*SomTemp))
Psmax est une entrée fixe dont la valeur est 296
SomTemp est une entrée variable prenant les valeurs : 24, 48, 72
Après remplacement de Psmax: par sa valeur,PS = 296/(a+b*exp(c*SomTemp))
Ecart (0) = PS après remplacement de SomTemp par sa première valeur (24) élevé au carrée = ((296/(a+b*exp(c*24)))^2)
Ecart(I) = PS après remplacement de Som Temp par sa deuxième valeur (48) élevé au carrée = ((296/(a+b*exp(c*48)))^2)
Ecart(2) = PS après remplacement de Som Temp par sa troisième valeur (72) élevé au carrée = ((296/(a+b*exp(c*72)))^2)
FE = (((296/(a+b*exp(c*24)))^2)-((296/(a+b*exp(c*48)))^2)-((296/(a+b*exp(c*72)))^2))

4.5.2.2.2.16 Procédure RPLENTREE

Problématique
Remplacement d'une entrée par sa valeur dans la fonction de modélisation.

Fonctions ou procédures appelées : Néant

Fonctions ou procédures appelantes : FCTECART()

Paramètres en entrée
Nm : Nom de l'entrée à remplacer dans le modèle.
Val : Valeur de l'entrée

Paramètres en sortie
Fm : Fonction de modélisation après remplacement de Nm par sa valeur.

Paramètres intermédiaires
LgFm : Longueur de la chaîne Fm.
PosEnt : Position de Nm dans Fm.
P1 : Partie de Fm en avant de Nm.
P2 : Partie de Fm jusqu'à Nm.
P3 : Partie de Fm après Nm.

Algorithme détaillé

```
DEBUT
    Initialiser PI, P2, P3 à vide
    PosEnt = 1
    TANT QUE PosEnt <> 0 FAIRE
        Calculer LgFm
        SI PosEnt = 0 ALORS
            Sortir de la boucle
        F-IN SI
        PI = Partie gauche de Fm
        P2 = Partie gauche de Fm & Val
        P3 = Partie droite de Fm
        Fm = P2 & P3
    FIN TANT QUE
FIN
```

Jeu d'essai
La fonction de modélisation de PS (Poids sec) du mil est :
PS feuille = Psmax/(a+b*exp(c*SomTemp))
Nm = Psmax
Val = 296
Apres exécution de la procédure, Psmax est remplacé par 296 et
Fm = 296/(a+b*exp(c*Som Temp))

*4.5.2.2.2.17 **Fonction** EXTRAIREF*

Problématique

Extraire la formulation de la fonction de modélisation de la chaîne des conditions multiples.

Analyse du problème

Les fonctions de modélisation peuvent être exprimées sous une forme conditionnelle dans la base de données des simulations : (SI Condition ; fonction1 ; fonction2), cela se traduit par, si Condition est vraie alors la fonction de modélisation est fonction1 sinon c'est fonction2.

Algorithme sous forme de texte

DEBUT
SI Condition **= « *SI* » *ALORS***
Déterminer et évaluer la partie condition
Extraire la fonction
SINON
Prendre la chaîne entière
FIN SI
FIN

Fonctions ou procédures appelées :

CHAINEFCT du module S_CAR_ECART_MOD

Fonctions ou procédures appelantes : FCTECART

Paramètres en entrée

Ch : Chaîne de caractères dans laquelle se fera l'extraction

Paramètres en sortie

Le paramètre en sortie est la valeur retournée par la fonction. Elle est de type variable. C'est la fonction de modélisation.

Paramètres intermédiaires:

RECond : Résultat de l'évaluation de la condition.
CCond : La partie condition de ch.
CExt : Partie de la chaîne à extraire.
Cond : Condition de la formule c'est à dire « SI ».
I, J, K : Indice de parcours de tableau et de boucle.
E : Code de l'erreur..

Algorithme

DEBUT
Déterminer Cond
SI Cond = « SI » ***ALORS***
Déterminer CCond
RECond = CHAINIEFCT (CCond , E)

```
            I = CCond + 6
            J - O
            SI RECond = « VRAIE » ALORS
                  EXTRAIREF = EXTRAIREF(La partie « vraie » de ch)
            SINON
                  EXTRAIREF = EXTRAIREF(La partie « fausse » de ch)
            FIN SI
      SINON
            EXTRAIREF = ch
      FIN SI
FIN
```

Jeu d'essai

Exemple de modélisation sur le mil (Poids sec feuille) :
(SI 24<=1200;296/(a+b*exp(c*24)); 296/(a+b*exp(c*1200))-(24-1200)*0.075)
CCond = 24<=1200
Après une évaluation de CCond,
RECond = VRAIE
ch = 296/(a+b*exp(c*24))

4.5.2.2.2.18 Procédure StockerValP

Problématique
Mettre la valeur optimisée de chaque paramètre dans la table MAJPARAM.

Analyse du problème
Après optimisation, les valeurs des paramètres obtenues sont stockées dans MAJPARAM qui sera rattachée à la base de données des simulations. A la demande de t'utilisateur, le contenu de cette table sera vidée dans la table Tparamètres auquel cas MAJPARAM sera supprimée.

Algorithme générale
DEBUT
Créer MAJPARAM, ouvrir MAJPARAM
Y stocker la valeur optimisée de chacun des paramètres.
Fermer MAJPARAM
FIN

Fonctions ou procédures appelées: Néant.

Fonctions ou modules appelants:
SIMPLEX, GRADIENT, OPTIM_SIMP_GRAD.

Paramètres en entrée
NB : Nombre total des paramètres du modèle.
CodV : Code de la variété de culture.
XMS() : Tableau contenant la valeur optimisée de chaque paramètre.
BaseParam: Base de données des simulations.

Paramètres en sortie: Néant
Paramètres intermédiaires:
VariationsP : Base de données pour l'utilisation MAJPARAM
TP : Table MAJPARAM.
PARAM : Unique champ de MAJPARAM
I : Indice de parcours de boucle
TravEspaceP : Espace de travail.

Algorithme détaillé
DEBUT
Mise en place de TravEspaceP, ouverture de BaseParam
Création et rattachement de MAJPARAM à BaseParam
I=0
TANT QUE I < NB **FAIRE**
« PARAM » = XMS (1), I = I + 1
Ajouter un enregistrement à MAJPARAM
FIN TANT QUE
Fermeture de MAJPARAM, BaseParam, TravEspaceP
FIN

4.5.2.2.2.19 Procédure MAJValP

Problématique
Mettre la valeur optimisée des paramètres dans la table Tparamètres

Analyse du problème
Les valeurs des paramètres optimisés peuvent être stockées dans la base de données en vue de leur utilisation future pour réaliser des simulations. En effet, Ces paramètres appliqués au modèle de simulation, permettent d'avoir des résultats plus proches de ceux observés aux champs.

Algorithme générale
DEBUT
Ouvrir la table de données des paramètres.
Se positionner sur l'enregistrement correspondant aux paramètres du modèle
Y stocker la valeur optimisée de chacun des paramètres.
Fermer la table des paramètres.
FIN

Fonctions ou procédures appelées: Néant.
Fonctions ou procédures appelantes: la procédure principale.

Paramètres en entrée
CF : Code de la fonction du modèle.
CodV : Code de la variété de culture.
BaseParam : Base de données des simulations.

Paramètres en sortie: Néant
Paramètres intermédiaires
TravEspaceP : Espace de travail.
TParamètres : Table Tparamètres.
VariationsP : Base de données pour l'utilisation de la table Tparamètres.
NP : Table intermédiaire contenant la valeur optimisée de chaque paramètre.

Algorithme détaillé
DEBUT
Mise en place de TravEspaceP,
Ouverture de BaseParam, TParamètres, NP
Se positionner sur le premier enregistrement de TParamètres et de NP
TANT QUE pas fin de fichier TParamètres ***FAIRE***
SI (« Code de la variété » de TParamètres = CV) et
(« Code de la fonction » de TParamètres = CF) ***ALORS***
« Valeur » de TParamètres = « PARAM »
Passer à l'enregistrement suivant de NP
FIN SI
Passer à l'enregistrement suivant de TParamètres
FIN TANT QUE
Fermeture deTParamètres, NP, BaseParam, TravEspaceP
F I N

4.5.2.2.3 OPTIM SIMP

Ce module perrnet d'obtenir un jeu de paramètres avec la méthode du simplex

4.5.2.2.3.7 Procédure MEILLEURJXPMS

Problématique
Obtenir le meilleur jeu de paramètres.

Analyse du problème
La méthode du simplex démarre avec plusieurs jeux de paramètres au départ. Ces jeux sont obtenus de manière aléatoire. La méthode du simplex est testée un certain nombre de fois (Nbllter = 6)avec des jeux de paramètres différents dans chaque cas. Le meilleur résultat est accepté.

Algorithme général

```
DEBUT
    Nblter = 0
    TANT QUE Nblter <= 5 FAIRE
        Mettre en place les jeux de paramètres de départ de manière aléatoire
        Appel de ESTSIMP
    FIN TANT QUE
    Comparer les résultats obtenus et accepter le meilleur
FIN
```

Fonctions ou procédures appelées
ESTSIMP
MINIMUM du module TESTS-VAL

Fonctions ou procédures appelantes : Néant.

Paramètres en entrée
SErr : Valeur indiquant la condition d'arrêt de l'algorithme du simplex.
NbP : Nombre de paramètres du modèle.
Mm() :Tableau à deux dimensions contenant la valeur minimale et maximale des paramètres.
Fct : Fonction des moindres carrés sous forme de chaîne de caractères.

Paramètres en sortie
MP() : Tableau à une dimension contenant le meilleur jeu de paramètres obtenu
VMeil : Valeur de la fonction des moindres carrés pour le meilleur jeu de paramètres obtenu.

Paramètres intermédiaires:
I : Indice de parcours de boucle (de 0 à Nblter) et de tableau (XMeilS())
J : Indice de parcours des lignes du tableau MP() et des colonnes du tableau XMeilS().

K : Indice de parcours des lignes du tableau Mm() et des colonnes du tableau MP().
PMin : Position de la plus petite valeur contenue dans le tableau SMeilS().
Min La plus petite valeur contenue dans le tableau SMeilS().
Nblter : Nombre de fois où la méthode du simplex est appelée avec des jeux de paramètres différents.
Erreurs : Code erreur.
SMeil : Valeur de la fonction des moindres carrés pour le meilleur jeu de paramètres.
Limite : Constante permettant de déterminer la marge d'évolution des paramètres dans la rnise en place des jeux de paramètres.
SMeilS(Nblter) : Tableau des meilleures valeurs obtenues..
JxPm(NbP+1, NbP) : Tableau des différents points.
XMeilS(Nblter, NbP) : Tableau des jeux de paramètres dont les valeurs minirnisent la fonction des moindres carrés.

Algorithme

```
DEBUT
    Initialiser Limite
    Nblter = 5
    FAIRE
        POUR I = 0 à Nblter
         '~~ Mise en place des jeux de paramètres
         POUR J=O à NbP
            POURK=OàNbP-1
               JxPm(J, K)= Mm(K,O) <= Valeur aléatoire <= Mm(K,1)
            FIN POUR
         FIN POUR
         '~~ Appel à la fonction la méthode du simplex.
         ESTSIMP (SErr, NbP, JxPm(), MP(), Mm(), SMeil, Fct, Erreurs)
         SI Erreurs = -1 ALORS
               SMeilS(I) = Limite
         SINON
               SMeilS(I) = SMeil
               POUR I = 0 à NbP - 1
                     XMeilS(I, J) = MP(J)
               FIN POUR
         FIN SI
        FIN POUR
        '~~ Recherche de la valeur minimale parmi les résultats obtenus
        MINIMUM (Nblter, SMeilS(), PMin, Min)
        '~~ Mise en place de MP
        POUR I = 0 à NbP - 1
               MP(I) = XMeilS(PMin, I)
        FIN POUR
        VMeil = SMeilS(PMin)
        Effacer les tableaux JxPm(), XMeilS() et SMeilS()
    TANT QUE VMeil <> Limite
FIN
```

4.5.2.2.3.2 Procédure ESTSIMP

Problématique

Mettre en place la méthode du simplex.

Analyse du problème

La méthode du simplex est réalisée à partie de plusieurs jeux de paramètres (N + 1 dans la cas où on a N paramètres dans le modèle).

Algorithme général

DEBUT

Fixer alpha, bêta, gamma, NbIterSimplex = 1.
Générer les n+1 jeux de paramètres (chaque jeu de paramètres constitue un point P).
Calculer la valeur de la fonction des moindres carrés(E) pour chaque point
TANT QUE pas satisfaisant ***FAIRE***
Bary = (1/n)∑ Pi (i : 0,1, . .., n).
Déterminer PI (Point pour lequel E est la plus basse)
Calculer Spl (E pour PI)
Déterminer Ph (Point pour lequel E est la plus grande)
Calculer SPh (E pour Ph)
Calculer Pr (Point réfléchi)
Calculer SPr (E pour Pr)
SI SPr <SPh ***ALORS*** Ph = Pr ***FIN SI***
SI SPr < SPI ***ALORS***
Pe = (1 + gamma) * Pr - gamma * Bary. (Point d'extension]
Calculer SPe (E pour Pe)
SI SPe < SPI ***ALORS***
Ph = Pe
SINON
Ph ▫ : Pr
FIN SI
SINON
SI SPr > SP ***ALORS***
SI SPr < SPh ***ALORS*** Ph = Pr ***FIN SI***
Pc=(bêta * Ph) + (1 - bêta) * Bary (Point de contraction)
Calculer SPc (E pour Pc)
Si SPc > SPh ***ALORS***
P = (P + PI) / 2 ***SINON***
Ph = Pc
FIN SI
SINON
Ph = Pr.
FIN SI
FIN Si
Déterminer PI, calculer SPI
FIN TANT QUE
Retourner le nouveau jeu de paramètres.
FIN

Fonctions ou procédures appelées

SOMMCARR du module S_CAR_ECART_MOD
MINIMUM du module TESTS-VAL
MAXIMUM du module TESTS-VAL
EVALUER du module S_CAR_ECART_MOD
TESTINTERV du module S_CAR_ECART_MOD

Fonctions ou procédures appelantes : MEILLEURJXPMS()

Paramètres en entrée

SS : Valeur indiquant la limite de réalisation du simplex.
N : Nombre de paramètres du modèle.
X() : Tableau des différents jeux de paramètres,
MnMx() : Tableau contenant la valeur minimale et maximale des paramètres
Fct : La fonction des moindres carrés.

Paramètres en sortie

SM : Valeur de la fonction des moindres carrés pour le meilleur jeu de paramètre.
E : Code de l'erreur
XM() : Tableau contenant le meilleur jeu de paramètres obtenu.

Paramètres intermédiaires

Pr() : Le point réfléchi.
Alpha : Constante utilisée pour le calcul du point Pr().
Pc() : Le point de contraction.
Bêta : Constante utilisée pour le calcul de Pc().
Pe() : Le point d'extension
Gamma : Constante utilisée pour le calcul de Pe().
FS : Fonction des moindres carrés après remplacement des paramètres par leur valeur.
XG(N+1 ,N) : Tableau des jeux de paramètres.
XGI(N, N) : Tableau de tous les points sauf le point Ph().
XP(N) : Tableau d'un jeu de paramètres.
SG(N + 1) : Tableau des résultats de la fonction pour chaque point.
SGI(N + 1) : Tableau des résultats de la fonction pour chaque point sauf Ph
PMaxil : Position du point dont le SG est la 2ième plus élevée.
SPI : Deuxième valeur de la fonction la plus élevée.
SP : Valeur de la fonction pour un point donné.
Pmini : Position du point dont la valeur de la fonction est la plus petite.
Pmaxi : Position du point dont la valeur de la fonction est maximum.
SPh : Valeur de la fonction des moindres carrés pour le point Ph().
SPl : Valeur de la fonction des moindres carrés pour le point Pl().
SPr : Valeur de la fonction des moindres carrés pour le point Pr().
SPe : Valeur de la fonction des moindres carrés pour le point Pe().
SPco : Valeur de la fonction des moindres carrés pour le point Pc().
SPIB : Valeur avant l'itération encours de SPI.
Pl() : Point pour qui la valeur de la fonction des moindres carrés est la plus basse.

Ph() : Point pour qui la valeur de la fonction des moindres carrés est la plus élevée.
Bary(N) : Tableau contenant la valeur du barycentre des points.
NbIter : Entier indiquant le nombre d'itérations dans la boucle principale.
I : Indice de parcours de boucle et **de** tableau.
J : Indice de parcours de boucle de tableau.
K : Indice de parcours de tableau.

NDepass : Nombre de fois où une erreur intervient dans l'évaluation de la fonction.
Limite : La plus grande valeur acceptée par le logiciel de développement
SomTotP(N) : Tableau contenant le total des différents points.
SommePm(N + 1) : Tableau contenant le total de chaque paramètre.

Algorithme

```
DEBUT
    Initialiser FS à vide
    E=O
    '~~ Buff érisation des premières valeurs des jeux de paramètres.
    POUR I=0 à N
        POUR J=O à N
            XG(I, J)=X(I, J)
        FIN POUR
    FIN POUR
    '~~ La valeur de la fonction des moindres carrés pour chaque point.
    NDepass = 0
    POUR I = 0 à N
        K=O
        POURJ=OàN-1
            XP(K) = XG(I, J)
            K = K + 1
        FIN POUR
        SOMMCARR (Ft, N, XP(), FS)
        SP = EVALUER (FS, E)
        SI E = -1 ALORS
            NDepass = NDepass + 1
            SG(I) = Limite
        SINON
            SG(I) = SP
        FIN SI
    FIN POUR
    SI NDepass = N + 1 ALORS
        Sortir de la procédure
    SINON
        E = 0
    FIN SI
    '~~ Détermination de PI().
    MINIMUM (N + 1, SG(), PMini, SPI)
    POURE=OàN-1
        PI(I) = XG(PMini, I)
```

```
FIN FOUR
SPIB = 0
Nblter = 0
FAIRE
    SI Nblter <> 0 ALORS
        SPIB = SPI 'Bufferisation de la valeur de SPI.
    FIN SI
    '--Détermination de Ph().
    MAXIMUM (N + 1, SG(), PMaxi, SPh)
    POUR I = 0 à N - 1
        Ph(I) = XG(PMaxi, I)
    FIN POUR
    '~~ Mise en place du tableau XGI pour le calcul du barycentre
    K=O
    POUR I = 0 à N
        SI I <> PMaxi ALORS
            POUR J = 0 à N - 1
                XGI(K, J) = XG(I, J)
            FIN POUR
            K = K + I
        FIN SI
    FIN POUR
    '-- Mise en place du tableau SGI
    POUR I = 0 à N - 1
        K=O
        POURJ=OàN-1
            XP(K) = XGI(I, J)
            K = K + I
        FIN POUR
        SOMMCARR (Ft, N, XP(), FS)
        SP = EVALUER (FS, E)
        SI E = -1 ALORS Sortir de la procédure FIN SI
        SGI(I) = SP
    FIN POUR
    '-- Détermination du deuxième point le plus élevé.
    MAXIMUM (N, SGI(), PMaxiI, SPI)
    '~~ Calcul du barycentre des points
    J=O
    FAIRE
        I = 0
        '~~ Calcul de la somme totale de chaque paramètre.
        TANT QUE (I <= N - 1) FAIRE
            SI I = 0 ALORS
                SommePm(I) = XGI(I, J)
            SINON
                SommePm(I) = SommePm(I - 1) + XGI(I, J)
            FIN SI
            I = I + 1
        FIN TANT QUE
        SomTotP(J) = SommePm(N - 1)
```

```
        Bary(J) = SomTotP(J) / N     'Détermination du barycentre.
        J = J + 1
TANT QUE J <> N
'~~ Calcul de Pr().
POUR I = 0 à N - 1
            Pr(I) = (1 + Alpha) * Bary(I) - (Alpha * Ph(I))
FIN POUR
'~~ Test de la valeur des paramètres de Pr().
TESTINTERV (N, MiniMaxi(), Pr())
'~~ Calcul de SPr.
SOMMCARR (Ft, N, Pr(), FS)
SPr = EVALUER(FS, E)
SI E = -1 ALORS Sortir de la procédure FIN SI
SI SPr < SPI ALORS
        POUR I = 0 à N - 1     'Détermination de Pe().
            Pe(I) = (1 + Gamma) * Pr(I) - Gamma * Bary(I)
        FIN POUR
        '~~ Test de la valeur des paramètres de Pe().
        TESTINTERV (N, MiniMaxi(), Pe())
        '~~ Calcul de SPe
        SOMMCAR!R (Ft, N, Pe(), FS)
        SPe = EVALUER(FS, E)
        SI E = -1 ALORS Sortir de la procédure FIN SI
        SI SPe < SPI ALORS
            POUR I = 0 à N - 1
                Ph(I) = Pe(I)
                XG(PMaxi, I) = Ph(I)
            FIN POUR
            SG(PMaxi) = SPe
        SINON
            POUR I = 0 à N - 1
                Ph(I) = Pr(I)
                XG(PMaxi, I) = Ph(I)
            FIN POUR
            SG(PMaxi) = SPr
        FIN SI
SINON
        SI SPr > SPI ALORS
            SI SPr <= SPh ALORS
                POUR I=O à N-1
                    Ph(I) = Pr(I)
                    XG(PMaxi, I) = Ph(I)
                FIN POUR
                SG(PMaxi) = SPr
            FIN SI
            '~~ Calcul de Pc().
            POUR I = 0 à N - 1
                Pc(I) = (Bêta * Ph(I)) + (1 - Bêta) * Bary(I)
            FIN POUR
            '~~ Test de la valeur des paramètres de Pr().
```

```
                    TESTINTERV (N, MiniMaxi(), Pc())
                    '~~ Calcul du SPco.
                    SOIMMCARR (Ft, N, Pc(), FS)
                    SPco = EVALUER(FS, E)
                    SI E = -1 ALORS Sortir de la procédure FIN SI
                    SI SPco > SPh ALORS
                        POUR I = 0 à N
                            K=O
                            '~~De nouvelles valeurs pour les points
                            POURJ=OàN-1
                                XG(I, J) = (XG(I, J) + PI(J)) / 2
                                XP(K) = XG(I, J)
                                K = K + I
                            FIN POUR
                            TESTINTERV (N, MiniMaxi(), Pc())
                            SOMMCARR (Ft, N, XP(), FS)
                            SP = EVALUER(FS, E)
                            SI E = -1 ALORS
                                Sortir de la procédure
                            FIN SI
                            SG(I) = SP
                        FIN POUR
                    SINON
                        POURI=OàN-1
                            Ph(I) = Pc(I)
                            XG(PMaxi, 1) = Ph(I)
                        FIN POUR
                        SG(PMaxi) = SPco
                    FIN SI
                SINON
                    POUR I = 0 à N - 1
                        Ph(I) = Pr(I)
                        XG(PMaxi, I) = Ph(I)
                    FIN POUR
                    SG(PMaxi) = SPr
                FIN SI
            FIN SI
            '~~ Détermination du nouveau PI().
            MINIMUM (N + 1, SG(), PMini, SPI)
            POUR I = 0 à N - 1
                PI(I) = XG(PMini, I)
            FIN POUR
            '~~ Modification de la condition d'arrêt dans le cas où SPI = 0
            SI SPI = 0 ALORS Sortir de cette boucle (TANT QUE) FIN SI
            Nblter = Nblter + 1
        TANT QUE Valeur absolue (SPI - SPIB) / SPI)) > SS
        '~~ Le meilleur jeu de paramètres obtenu est mis dans le tableau XM()
        POUR I = 0 à N
            XM(I) = XG(PMini, I)
        FIN POUR
```

SM = SPI
Effacer les tableaux XG, XGI, XP, SG, SGI, PI, Ph, Pr, Pe, Pc, SomTotP, SommePm, Bary
FIN

4.5.2.2.4 TESTS_VAL

Ce module regroupe les procédures permettant de connaître :
- ia plus petite valeur contenue dans un tableau de N valeurs.
- la plus grande valeur contenue dans un tableau de N valeurs.
- la validité d'une date

4.5.2.2.4.1 Procédure MINIMUM

Problématique
Fournir la plus petite valeur contenue dans un tableau.

Fonctions ou procédures appelées: Néant

Fonctions ou procédures appelantes:
ESTSIMP du module OPTIM-SIMP
MEILLEURJXPMS du module OPTIM-SIMP

Paramètres en entrée
TailleN : Nombre de valeurs dans le tableau ValFN().
ValFN() : Tableau des valeurs parmi lesquelles on veut connaître la plus petite.

Paramètres en sortie
PosMini : Position de la plus petite valeur contenue dans le tableau ValFN().
Minr . La plus petite valeur contenue dans le tableau ValFN().

Paramètres intermbdiaires
I : indice de parcours du tableau ValB().
J : Indice de parcours du tableau ValFN().
ValB(TailleN) : Tableau contenant les mêmes valeurs que celles de ValFN()

Algorithme détaillé

```
DEBUT
        '~~ Mise en place de ValB()
        POUR I = 0 à TailleN N - 1
                ValB(I) = ValFN(I)
        FIN POUR
        I = 0, J = 1
        '--Recherche de la valeur minimale dans le tableau
        TANT QUE J < TailleN FAIRE
                SI ValB(I) > ValFN(J) ALORS I = J FIN SI
                J = J + 1, Mini = ValB(I)
                PosMini = I
        FIN TANT QUE
FIN
```

4.5.2.2.4.2 Procédure MAXIMUM

Problématique
Fournir la plus grande valeur contenue dans un tableau.

Fonctions ou procédures appelées: Néant

Fonctions ou procédures appelantes :
ESTSIMP du module OPTIM_SIMP

Paramètres en entrée
TailleX : Nombre de valeurs contenues dans le tableau ValFX().
ValFX() : Tableau des valeurs parmi lesquelles on veut connaître la plus grande.

Paramètres en sortie
PosMaxi : Position de la plus grande valeur contenue dans le tableau ValFX().
Maxi : La plus grande valeur contenue dans le tableau ValFX().

Paramétres intermédiaires
ValB(TailleX) : Tableau contenant les mêmes valeurs que celles de ValFX()
I : indice de parcours du tableau ValB().
J : Indice de parcours du tableau ValFX().

Algorithme

```
DEBUT
        '~~ Mise en place de ValB()
        POUR I = 0 à TailleX - 1
                ValB(I) = ValFX(I)
        FIN POUR
        I = 0
        J = I
        '~~ Recherche de la valeur maximale dans le tableau
        TANT QUE J < TailleX FAIRE
                SI ValB(I) < ValFX(J) ALORS
                        I = J
                FIN SI
                J = J + 1
                Maxi = ValB(I)
                PosMaxi = I
        FIN TANT QUE
FIN
```

4.5.2.2.4.3 Procédure CONTFORMDAT

Problématiaue
Déterminer si une date saisie est valide ou pas.

Fonctions ou procédures appelées: Néant

Fonctions ou procédures appellantes:

Paramètres en entrée
D : Date saisie à l'écran.

Paramètres en sortie

Paramètres intermédiaires
Delim : Séparateur du jour du mois et de l'année.

Algorithme détaillé

```
DEBUT
    SI Delim=« . » ou « / » ou « : » ou « - » ou « » ALORS
        SI D est de la forme 00 Delim 00 Delim 0000 ALORS
            Accepter D
        SINON
            Afficher uin message d'erreur
        FIN SI
    SINON
        Afficher un message d'erreur
    FIN SI
FIN
```

4.5.2.2.5 S CAR ECART MOD

Ce module est formé de procédures qui traitent la fonction des moindres carrés

4.5.2.2.5.1 Procédure TESTINTERV

Problématique

Tester si une donnée se trouve clans un intervalle fixé.

Analyse du problème

La valeur d'un paramètre doit toujours être dans l'intervalle fixé au départ dans la table TParamètres de la base de données des simulations.

Algorithme sous forme de texte

DEBUT

SI la valeur du paramètre > la valeur de la borne supérieure fixée ALORS
Lui attribuer la valeur de la borne supérieure
SINON
SI la valeur du paramètre < la valeur de la borne inférieure fixée ALORS
Lui attribuer la valeur de la borne inférieure
FIN Si, FIN SI

FIN

Fonctions ou procédures appelées : Néant

Fonctions ou procédures appelantes

ESTSIMP du module OPTIM_SIMP
NUMGRAD du module OPTIM-GRAD
ESTGRAD du module OPTIM-GRAD

Paramètres en entrée

NP : Nombre de paramètres du modèle.
Bornes() : Tableau des bornes inférieures et supérieures des paramètres.

Paramètres en sortie

P() : Tableau contenant la valeur des paramètres à vérifier

Paramètres intermédiaires :

I : Indice de parcours du tableau Bornes().

Algorithme détaillé

DEBUT

POUR I = 0 ***à*** NP - 1
SI P(I) < Bornes(I, 0) ***ALORS*** P(I) = Bornes(I, 0)
SINON
SI P(I) > Bornes(I, 1) ALORS P(I) = Bornes(I, 1) ***FIN SI***
FIN SI, FIN POUR, FIN

4.5.2.2.5.2 Procédure SOMMCA RR

Problématique

Remplacer chaque paramètre par une valeur dans la fonction des moindres carrés.

Analyse du problème

Dans la fonction de modélisation, le premier paramètre est codé X(O), le deuxième X(1) et ainsi de suite. Chacun de ces paramètres doit être remplacé par une valeur. Cela permet d'évaluer plus tard la fonction des moindres carrés.

Algorithme sous forme de texte

DEBUT

Remplacer chaque paramètre par sa valeur dans la fonction des moindres carrés

Bien organiser la chaîne de caractères obtenue

FIN

Fonctions ou procédures appelées :

ESTSIMP du module OPTIM_SIMP

Fonctions ou procédures appellantes:

ESTSIMP du module OPTIM_SIMP
NUMGRAD du module OPTIM-GRAD
ESTGRAD du module OPTIM-GRAD

Paramètres en entrée

Fn : Fonction des moindres carrés
Nb : Nombre de paramètres du modèle.
X() : Point pour qui on veut connaître la valeur de la fonction des moindres carrés.

Paramètres en sortie

FnBuf : Fonction des moindres carrés après remplacement de chaque paramètre par sa valeur.

Paramètres intermédiaires:

Partiel : Partie de la fonction se trouvant juste avant le code ou paramètre à remplacer.
Partie2 : Partie se trouvant juste avant le code du paramètre à remplacer + valeur du paramètre
Partie3 : Partie de la fonction se trouvant juste après le code du paramètre à remplacer.
I : Indice de parcours du tableau X().
ExpRech : Expression que l'on cherche dans Fn
SgRpl : Signe de remplacement
DebRech : Indique la position du début de la recherche dans Fn.

LgFnBuf : Taille de Fn après avoir remplacer chaque paramètre par sa valeur.
LgPartie1 : Taille de Partiel.
PosParam : Position d'un paramètre donné dans Fn.
PosSigne : Position des signes --,-+,+-, ++ dans Fn après la prise en compte de la valeur de chaque paramètre

Algorithme détaillé

```
DEBUT
    DebRech = 1
    FnBuf = Fn
    Supprimer les espaces de gauche et de droite de FnBuf
    F=O
    '~~ Remplacement de chaque paramètre de FnBuf par sa valeur
    POUR I = 0 à Nb - 1
        PosParam = 1
        ExpRech = "X( I)"
        TANT QUE PosParam <> 0 FAIRE
            LgFnBuf = Longueur (FnBuf)
            Remplacer le I de ExpRech par la valeur du I
            PosParam = Position de ExpRech dans Fn
            SI PosParam = 0 ALORS Sortir de la boucle FIN SI
            Partiel =Partie gauche de FnBuf jusqu'à PosParam-1
            Lg Partiel = Longueur (Partiel )
            SI X(I) < 0 ALORS
                Partie2 = Partiel & "(" & X(I) & ")"
            SINON
                Partie2 = Partiel & X(I)
            FIN SI
            Partie3 = Partie droite (FnBuf) à partir de la position
                        (LgFnBuf - LgPartie1 - 4)
            FnBuf = Partie2 & Partie3
        FIN TANT QUE
    FIN POUR
'~~ Structuration de la fonction avec prise en compte des signes:--,-+.+-,++
    POUR I = 0 à 3
        SI I=0 ALORS
            ExpRech = "--"
            Sg Rpl = "+"
        FIN SI
        SI I=I ALORS
            ExpRech = "-+" A LORS
            SgRpl = "-"
        FIN SI
            SI I = 2 ALORS
            ExpRech = "+-"
            SgRpl = "-"
        FIN Si
            SI I = 3 ALORS
            ExpRech = "++"
```

```
                SgRpl = "+"
            FIN SI
            PosSigne = 1
            TANT QUE PosSigne <> 0 FAIRE
                LgFnBuf = Longueur (FnBuf)
                PosSigne = Position de ExpRech dans Fn
                SI PosSigne = 0 ALORS Sortir du POUR FIN SI
                Partiel = Partie gauche (FnBuf) jusqu'à la position
                    (PosSigne - 1) en commençant paf 1
                LgPartie1 = Longueur (Partiel)
                Partie2 = Partiel & SgRpl
                Partie3 = Partie droite de FnBuf à partir de la position
                    (LgFnBuf - LgPartie1 - 2)) jusqu'à la fin de la chaîne
                FnBuf = Partie2 & Partie3
            FIN TANT QUE
        FIN POUR
FIN
```

Jeu d'essai

Soit Fn = (((296/(a+b*exp(c*24)))^2)-((296/(a+b*exp(c*48)))^2)-((296/(a+b*exp(c*72)))^2))

Nb = 3

X(0) = 1, valeur du paramètre a

X(1) = 928.63, valeur du paramètre b

X(2) = -0.0082, valeur du paramètre c

Exemple pour le paramètre a

Première itération

Partiel = (((296/(

Partie2 = (((296/(1

Partie3 = +b*exp(c*24)))^2)-((296/(a+b*exp(c*48)))^2)-((296/(a+b*exp(c*72)))^2))

Fn=(((296/(1+b*exp(c*24)))^2)-((296/(a+b*exp(c*48)))^2)-((296/(a+b*exp(c*72)))^2))

Deuxième itération

Partiel = (((296/(1+b*exp(c*24)))^2)-((296/(

Partie2 = (((296/(1+b*exp(c*24)))^2)-((296 / (1

Partie3 = +b*exp(c*48)))^2)-((296 / (a+b*exp(c*72)))^2))

Fn = (((296/(1+b*exp(c*24)))^2)-((296/(1+b*exp(c*48)))^2)-((296/(a+b*exp(c*72)))^2))

Troisième itération

Partiel = (((296/(1 +b*exp(c*24)))^2)-((296/(1 +b*exp(c*48)))^2)-((296 / (

Partie2 = (((296/(1+b*exp(c*24)))^2)-((296 / (1 +b*exp(c*48)))^2)-((296/(1

Partie3 = +b*exp(c*72)))^2))

Fn =(((296/(1 +b*exp(c*24)))^2)-((296/(1+b*exp(c*48)))^2)-((296/(1+b*exp(c*72)))^2))

4.5.2.2.5.3 Fonction CHAINEFCT

Problématique

Evaluer une condition exprimée sous forme de chaîne de caractères

Analyse du problème

La partie condition de la formulation de la fonction de modélisation est de la forme : A opérateur B, opérateur peut être <, >, etc. A et B peuvent avoir une forme imbriquée, c'est à dire exprimées sous forme de sous-conditions. Le résultat de cette condition joue sur le choix de la partie à extraire dans la formulation de la fonction de modélisation.
Cette fonction est réalisée de manière récursive.

Algorithme sous forme de texte

```
DEBUT
    SI Résultat d'évaluation = VRAIE OU FAUX ALORS
        fin du traitement
    SINON
        TANT QUE Résultat d'évaluation <> VRAIE OU FAUX FAIRE
            Continuer à évaluer la sous-condition
        FIN TANT QUE
    FIN SI
FIN
```

Fonctions ou procédures appelées

ESTOPERATEUR
OPERATEUR
IMAGEFONCTION

Fonctions ou procédures appelantes : EXTRAIREF du module GESTION-BD

Paramètres en entrée

Chn	: Chaîne de caractères à évaluer.
Erreur	: Code erreur.

Paramètres en sortie

C'est la valeur de retour de la fonction. Elle est de type booléen (Vraie ou Faux). Vraie : si la condition est Vraie et Faux si la condition est fausse,

Paramètres intermédiaires:

LongChn	: Longueur de Chn.
Nbl	: Partie gauche d'une opération.
Nb2	: Partie droite d'une opération.
Condtion	: Booléen indiquant si un caractère est de type numérique ou pas.
I	: Indice de parcours de Chn.
J	: Entier servant à la reconnaissance des parenthèses.
K	: Indice de parcours de Chn.
IDFonction	: Indentificateur d'une fonction prédéfinie (Ex: In, exp).
Err	: Code de l'erreur engendrée par les calculs.

Algorithme

```
DEBUT
    Erreur = Err
    LongChn = Longueur de Chn
    SI Err = -1 ALORS
       Sortir de la fonction
    FIN SI
    '~~ Cas d'une simple valeur numérique
    SI Chn est un nombre ALORS
       CHAINEFCT = Val (Chn)
       Sortir de la fonction
    FIN SI
    Extraire le dernier caractère de Chn
    SI le dernier caractère de Chn est un nombre ALORS
       Nb1 = ""
       I = LongChn
       SI le I ème caractère de Chn est un nombre ou est égal à « » ALORS
             Gondtion = Vrai
       SINON
             Gondtion = Faux
       FIN SI
       '~~ Mise en place de la première opérande
       TANT QUE Condtion = VRAI FAIRE
             Nbl = Le I ème caractère de Chn & Nb1
             I - I - 1
             SI le I ème caractère de Chn est un nombre ou = « . » ALORS
                   Condtion = Vrai
             SINON
                   Condtion = Faux
             FIN SI
       FIN TANT QUE
       SI le I - 1 ème caractère = < ou > ou e ou E ou o ou 0 ALORS
             Nb2 = Partie gauche de Chn jusqu'à la position I - 2
             CHAINEFCT = OPERATEUR(les deux caractères à la position I-1
                   de Chn,CHAINEFCT(Nb2, Err), CHAINEFCT(Nb1, Err), Err)
       SINON
             Nb2 = Partie gauche de Chn jusqu'à la position I - 1
             CHAINEFCT = OPERATEUR(le caractère à la position I de Chn,
                         CHAINEFCT(Nb2, Err), CHAINEFCT(Nb1, Err), Err)
       FIN SI
       SI Err = - 1 ALORS Sortir de la fonction FIN SI
    FIN SI
    SI le caractère à la position LongChn de Chn = ")" ALORS
       J = I
       I = LongChn - 1
       '--Recherche de parenthcises pour connaître le bloc interne à évaluer
       TANT QUE J <> 0 FAIRE
             SI le I ème caractère de Chn = ")" ALORS J = J + 1 FIN SI
             SI le I ème caractère de Chn = "(" ALORS J = J - 1 FIN SI
             I = I - I
```

```
        FIN TANT QUE
        '~~ Evaluation du bloc de données se trouvant entre parenthèses
        SI I = 0 ALORS
              CHAINEFCT = CHAINEFCT(LongChn - 2 caractères de Chn à
                          partir de la position 2, Err)
              Sortir de la fonction
        FIN SI
        '~~ En présence d'un opérateur, prise en compte des parties gauches et
            droites
        SI ESTOPERATEUR(I ème caractère de Chn) ALORS
           CHAINEFCT = OPERATEUR(le I ème caractère de Chn,
           CHAINEFCT(Partie gauche de Chn jusqu'à la position I-1, Err),
           CHAINEFCT(LongChn caractères à partir de la position I + 1 ,Err), Err)
           Sortir de la fonction
        FIN SI
        K = I
        '~~ Mise en place de l'identificateur de la fonction à appliquer
        IDFonction = ""
        TANT QUE (K <> 1) et (ESTOPERATEUR(le caractère à la position K
                          de Chn) = Faux) FAIRE
              IDFonction = le caractère à la position K de Chn & IDFonction
              K = K - 1
        FIN TANT QUE
        SI K = 1 ALORS
              IDFonction = le caractère à la position K de Chn & IDFonction
              CHAINEFCT = IMAGEFONCTION( IDFonction,
                    CHAINEFCK(LongChn-I-2 caractères à partir de la position
                    I+2 de Chn, Err), Err)
              SI Err = -1 ALORS Sortir de la fonction FIN SI
        SINON
              CHAINEFCT = OPERATEUR(le caractère à la position K de Chn,
              CHAINEFCT(LongChn-K caractères à partir de la position K+I de
              Chn, Err), CHAINEFCT(Partie gauche de Chn jusqu'à la position
              K-I , Err))
        FIN SI
    FIN SI
FIN
```

Jeu d'essai

Au départ, Chn = 24<=1200
Après évaluation, Chn = VRAIE

4.5.2.2.5.4 Fonction EVALUER

Problématique

Evaluer une fonction exprimée sous forme de chaîne de caractères.

Analyse du problème

La fonction de modélisation est de la forme : A opérateur B, opérateur peut être +,- etc. A et B peuvent avoir un forme imbriquée, c'est à dire être aussi exprimées sous forme de sous-fonction.
Cette fonction est réalisée de manière récursive.

Algorithme sous forme de texte

```
DEBUT
    SI Résultat d'évaluation = un nombre ALORS
        fin du traitement
    SINON
        TANT QUE Résultat d'évaluation <> un nombre FAIRE
            Continuer à évaluer la sous-fonction
        FIN TANT QUE
    FIN SI
FIN
```

Fonctions ou procédures appelées

ESTOPERATEUR
OPERATEUR
IMAGEFONCTION

Fonctions ou procédures appelantes:

ESTSIMP du module OPTIM_SIMP
NUMGRAD du module OPTIM_GRAD
ESTGRAD du module OPTIM_GRAD

Paramètres en entrée

Chn : Chaîne de caractères à évaluer.
Erreur : Code de l'erreur.

Paramètres en sortie

C'est la valeur de retour de la fonction. Elle est de type flottant.

Paramètres intermédiaires

LongChn : Longueur de Chn.
Nb1 : Partie gauche d'une opération.
Nb2 : Partie droite d'une opération.
Condtion : Booléen indiquant si un caractère est de type numérique ou pas.
I : Indice de parcours de Chn.
J : Entier servant à la reconnaissance des parenthèses.
K : Indice de parcours de Chn.
IDFonction : Indentificateur d'une fonction prédéfinie (Ex: In, exp).
Err : Code de l'erreur engendrée par les calculs.

Algorithme détaillé

```
DEBUT
    LongChn = Longueur de Chn
    Erreur = Err
    SI Err = -1 ALORS Sortir de la fonction FIN SI
    '~~ Cas d'une simple valeur numérique
    SI Chn est un nombre ALORS
       EVALUER = Nombre
       Sortir de la fonction
    FIN SI
    Extraire le dernier caractère de Chn
    SI le dernier caractère de Chn est un nombre ALORS
       Nb1 = ""
       I = LongChn
       SI le I ème caractère de Chn est un nombre ou est égal à « . » ALORS
             Condtion = Vrai
       SINON
             Condtion = Faux
       FIN SI
       '~~ Mise en place de la première opérande
       TANT QUE Condtion = VRAI FAIRE
          Nbl = Le I ème caractère de Chn & Nbl
          I = I - 1
          SI le I ème caractère de Chn est un nombre ou est égal à « . » ALORS
             Condtion = Vrai
          SINON
             Condtion = Faux
          FIN SI
       FIN TANT QUE
       '~~ Mise en place de la deuxième opérande
       Nb2 = Partie gauche de Chn jusqu'à la position I - 1
             EVALUER = OPERATEUR(le caractère à la position I de Chn,
                   EVALUER(Nb2, Err), EVALUER(Nb1, Err), Err)
       SI Err = -1 ALORS Sortir de la fonction FIN SI
    FIN SI
    SI le caractère à la position LongChn de Chn = ")" ALORS
       J = I
       I = LongChn - 1
       '--Recherche de parenthèses pour connaître le bloc interne à évaluer
       TANT QUE J <> 0 FAIRE
             SI le I ème caractère de Chn = ")" ALORS J = J + 1 FIN SI
             SI le I ème caractère de Chn = "(" ALORS J = J - 1 FIN SI
             I = I - 1
       FIN TANT QUE
       '~~ Evaluation du bloc de données se trouvant entre parenthèses
       SI I = 0 ALORS
             EVALUER = EVALUER(LongChn - 2 caractères de Chn à partir
                   de la position 2, Err)
             Sortir de la fonction
       FIN SI
```

```
'~~ En présence d'un opérateur, prise en compte des parties gauches et
droites
SI ESTOPERATEUR(I ème caractère de Chn) ALORS
    EVALUER = OPERATEUR(le I ème caractère de Chn,
     EVALUER(Partie gauche de Chn jusqu'à la position I - 1, Err),
     EVALUER(LongChn caractères à partir de la position I + 1, Err), Err)
     Sortir de la fonction
FIN SI
K = I
'~~ Mise en place de l'identificateur de la fonction à appliquer
IDFonction = ""
TANT QUE (K <> 1 ) et (ESTOPERATEUR(le caractère à la position K
                                    de Chn) = Faux) FAIRE
     IDFonction = le caractère à la position K de Chn & IDFonction
     K = K - 1
FIN TANT QUE
SI K = 1 ALORS
     IDFonction = le caractère à la position K de Chn & IDFonction
     EVALUER = IMAGEFONCTION(IDFonction, EVALUER(LongChn-
          I-2 caractères à partir de la position I+2 de Chn), Err)
     Si Err = -1 ALORS Sortir de la fonction FIN SI
     SINON
          EVALUER = OPERATEUR(le caractère à la position K de
          Chn, EVALUER(LongChn-K caractères à partir de la
          position K+1 de Chn, Err), EVALUER(Partie gauche de
          Chn jusqu'à la position K-I, Err), Err)
          SI Err = -1 ALORS Sortir de la fonction FIN SI
     FIN SI
FIN SI
FIN
```

Jeu d'essai

Au départ, Chn = 24-1200
Après évaluation, Chn = 11 76

4.5.2.2.5.5 Fonction ESTOPERATEUR

Problématique
Voir si un identifiant est un opérateur ou non.

Analyse du problème
Il faut recenser tous les caractères reconnus comme opérateur. Cela permet de voir si un opérateur y est répertorié ou pas.

Algorithme sous forme de texte
DEBUT
SI l'identifiant est dans la liste des opérateurs du système d'autoparamétrisation ***ALORS***
Retourner VRAI
SINON
Retourner FAUX
FIN SI
FIN

Fonctions ou procédures appelées : Néant

Fonctions ou procédures appelantes
EVALUER
CHAINEFCT

Paramètres en entrée
Caractère : Chaîne de caractère(s) à tester

Valeur retour
Elle est de type variable. C'est l'identifiant de opérateur.

Paramètres intermédiaires: Néant

Algorithme
DEBUT
SI Caractère appartient à la liste {+, -, *, /, ^, \, >, <, <>, <=, >=, =, et, Et, eT, ET, Ou, oU, ou, OU) ***ALORS***
ESTOPERATEUR = VRAIE
SINON
ESTOPERATEUR = FAUX
FIN ***SI***
FIN

Jeu d'essai
Cas 1 : Caractère = +
OPERATEUR = VRAI
Cas 2 : la valeur de Caractère n'existe pas dans la liste
Caractère = m
OPERATEUR = FAUX

4.5.2.2.5.6 Fonction *OPERATEUR*

Problématique
Evaluer un opérateur.

Algorithme sous forme de texte
DEBUT Lire l'opérateur et l'appliquer aux opérandes ***FIN***

Fonctions ou procédures appelées : Néant

Fonctions ou procédures appelantes
EVALUER
CHAINEFCT

Paramètres en entrée
Signe : l'opérateur à appliquer.
Nbl : Opérandel.
Nb2 : Opérande2.
EO : Code de l'erreur.

Paramètres en sortie
C'est la valeur de retour de la fonction. Elle est de type variable

Paramètres intermédiaires: Néant

Algorithme détaillé

```
DEBUT
    SI Signe = « + » ALORS OPERATEUR = Nbl + Nb2 FIN SI
    SI Signe = « - » ALORS OPERATEUR = Nbl - Nb2 FIN SI
    SI Signe = « * » ALORS OPERATEUR = Nbl * Nb2 FIN SI
    SI Signe = « / » ALORS OPERATEUR = Nbl / Nb2 FIN SI
    SI Signe = « ^ » ALORS OPERATEUR = Nbl ^ Nb2 FIN SI
    SI Signe = « \ » ALORS OIPERATEUR = Nbl \ Nb2 FIN SI
    SI Signe = « > » ALORS
        SI Nbl > Nb2 ALORS
            OPERATEUF! = VRAIE
        SINON
            OPERATEUR = FAUX
        FIN SI
    FIN SI
    SI Signe = « < » ALORS
        SI Nbl < Nb2 ALORS
            OPERATEUR =VRAIE

        SINON
            OPERATEUR = FAUX
        FIN SI
    FIN SI
```

```
        SI Signe = « >= » ALORS
                SI Nbl >= Nb2 ALORS
                        OPERATEUR = VRAIE
                SINON
                        OPERATEUR = FAUX
                FIN SI
        FIN SI
        SI Signe = « <= » ALOR!S
                SI Nbl <= Nb2 ALORS
                        OPERATE'UR = VRAIE
                SINON
                        OPERATEUR = FAUX
                FIN SI
        FIN SI
        SI Signe = « <> » ALORS
                SI Nb1< > Nb2 ALORS
                        OPERATEUR = VRAIE
                SINON
                        OPERATEUR = FAUX
                FIN SI
        FIN SI
        SI Signe en majuscule = « ET » ALORS
                SI (Nbl = VRAIE) et (Nb2 = VRAIE) ALORS
                        OPERATEIJR = VRAIE
                SINON
                        OPERATEIJR = FAUX
                FIN SI
        FIN SI
        SI Signe en majuscule = « OU » ALORS
                SI (Nbl = FAUX) et (Nb2 = FAUX) ALORS
                        OPERATEIJR = FAUX
                SINON
                        OPERATEUR =VRAIE
                FIN SI
        FIN SI
        SI EO = -1 ALORS
                Sortir de la fonction
        FIN SI
FIN
```

Jeu d'essai

Signe = <=
Nbl = 24
Nb2 = 1200
OPERATEUR =VRAIE

4.5.2.2.5.7 Fonction IMAGEFONCTION

Problématique
Calculer l'image d'une fonction. Cette fonction permet de reconnaître au moins toutes les fonctions mathématiques de Visual Basic 4.0.

Fonctions ou procédures appelées : Néant

Fonctions ou procédures appelantes : EVALUER

Paramètres en entrée
IDFonction : Identificateur de la fonction
Nb : Nombre dont on veut calculer l'image.

Valeur retour
Elle est de type flottant. Résultat du calcul.

Alaorithme

```
DEBUT
    '-- Valeur absolue
    SI IDFonction = "ABS" ALORS
        IMAGEFONCTION = Valeur absolue(Nb)
    FIN SI
    '-- Renvoie la partie entière d'un nombre
    SI IDFonction = "FIX" ALORS IMAGEFONCTION = Fix(Nb) FIN SI
    '--Convertit une expression en donnée de type entier
    SI IDFonction = "CINT" ALORS IMAGEFONCTION = CInt(Nb) FIN SI
    '--Renvoie un nombre aléatoire
    SI IDFonction = "RND" ALORS IMAGEFONCTION = Rnd(Nb) FIN SI
    '~~ Sinus
    SI IDFonction = "SIN" ALORS IMAGEFONCTION = Sin(Nb) FIN SI
    '~~ Cosinus
    SI IDFonction = "COS" ALORS IMAGEFONCTION = Cos(Nb) FIN SI
    '~~ Tangente
    SI IDFonction = "TAN" ALORS IMAGEFONCTION = Tan(Nb) FIN SI
    '~~ Arctangente
    SI IDFonction = "ATN" ALORS IMAGEFONCTION = Atn(Nb) FIN SI
    '~~ Logarithme népérien
    Si IDFonction = "LN" ALORS IMAGEFONCTION = Log(Nb) FIN SI
    '~~ Exponentielle
    SI IDFonction = "EXP" ALORS IMAGEFONCTION = Exp(Nb) FIN SI
    '~~ Racine carrée
    SI IDFonction = "SQR" ALORS IMAGEFONCTION = Sqr(Nb) FIN SI
    '~~ Sécante
    SI IDFonction = "SEC" ALORS IMAGEFONCTION = 1 / Cos(Nb) FIN SI
    '~~ Cosécante
    SI IDFonction="COSEC" ALORS IMAGEFONCTION = 1 / Sin(Nb) FIN SI
    '~~ Cotangente
    SI IDFonction="COTAN" ALORS IMAGEFONCTION=1/ Tan(Nb) FIN SI
```

```
SI IDFonction = "ARCSIN" ALORS          '~~ Arc sinus
        IMAGEFONCTION = Atn(Nb / Sqr(-Nb * Nb + 1))
FIN SI
SI IDFonction = "ARCCOS" ALORS          '~~ Arc cosinus
        IMAGEFONCTION = Atn(-Nb / Sqr(-Nb * Nb + 1)) + 2 * Atn(1)
FIN SI
SI IDFonction = "ARCSEC" ALORS          '~~ Arc sécante
   IMAGEFONCTION = Atn(Nb/Sqr(Nb*Nb-1)) + Sgn((Nb)-1) * (2*Atn(1))
FIN SI
SI IDFonction = "ARCCOSEC" ALORS        '~~ Arc cosécante
   IMAGEFONCTION = A.tn(Nb/Sqr(Nb*Nb-1)) + (Sgn(Nb)-1) * (2*Atn(1))
FIN SI
SI IDFonction = "ARCCOTAN" ALORS        '~~ Arc cotangente
        IMAGEFONCTION = Atn(Nb) + 2 * Atn(1)
FIN SI
SI IDFonction = "HSIN" ALORS            '~~ Sinus hyperbolique
        IMAGEFONCTION = (Exp(Nb) - Exp(-Nb)) / 2
FIN SI
SI IDFonction = "HCOS" ALORS            '~~ Cosinus hyperbolique
        IMAGEFONCTION = (Exp(Nb) + Exp(-Nb)) / 2
FIN SI
SI IDFonction = "HTAN" ALORS            '~~ Tangente hyperbolique
        IMAGEFONCTION = (Exp(Nb) - Exp(-Nb)) / (Exp(Nb) + Exp(-Nb))
FIN SI
SI IDFonction = "HSEC" ALORS            '~~ Sécante hyperbolique
        IMAGEFONCTION = 2 / (Exp(Nb) + Exp(-Nb))
FIN SI
SI IDFonction = "HCOSEC" ALORS          '~~ Cosécante hyperbolique
        IMAGEFONCTION = 2 / (Exp(Nb) - Exp(-Nb))
FIN SI
SI IDFonction = "HCOTAN" ALORS          '~~ Cotangente hyperbolique
        IMAGEFONCTION = (Exp(Nb) + Exp(-Nb)) / (Exp(Nb) - Exp(-Nb))
FIN SI
SI IDFonction = "HARCSIN" ALORS         '~~ Sinus hyperbolique inverse
        IMAGEFONCTION = Log(Nb + Sqr(Nb * Nb + 1))
FIN SI
SI IDFonction = "HARCCOS" ALORS         '~~ Cosinus hyperbolique inverse
        IMAGEFONCTION = Log(Nb + Sqr(Nb * Nb - 1))
FIN SI
SI IDFonction = "HARCTAN" ALORS         '~~ Tangente hyperbolique inverse
        IMAGEFONCTIONI = Log((1 + Nb) / (1 - Nb)) / 2
FIN SI
'~~ Sécante hyperbolique inverse
SI IDFonction = "HARCSEC" ALORS
        IMAGEFONCTION = Log((Sqr(-Nb * Nb + 1) + 1) / Nb)
FIN SI
'~~ Cosécante hyperbolique inverse
SI IDFonction = "HARCCOSEC" ALORS
        IMAGEFONCTION = Log((Sgn(Nb) * Sqr(Nb * Nb + 1) + 1) / Nb)
```

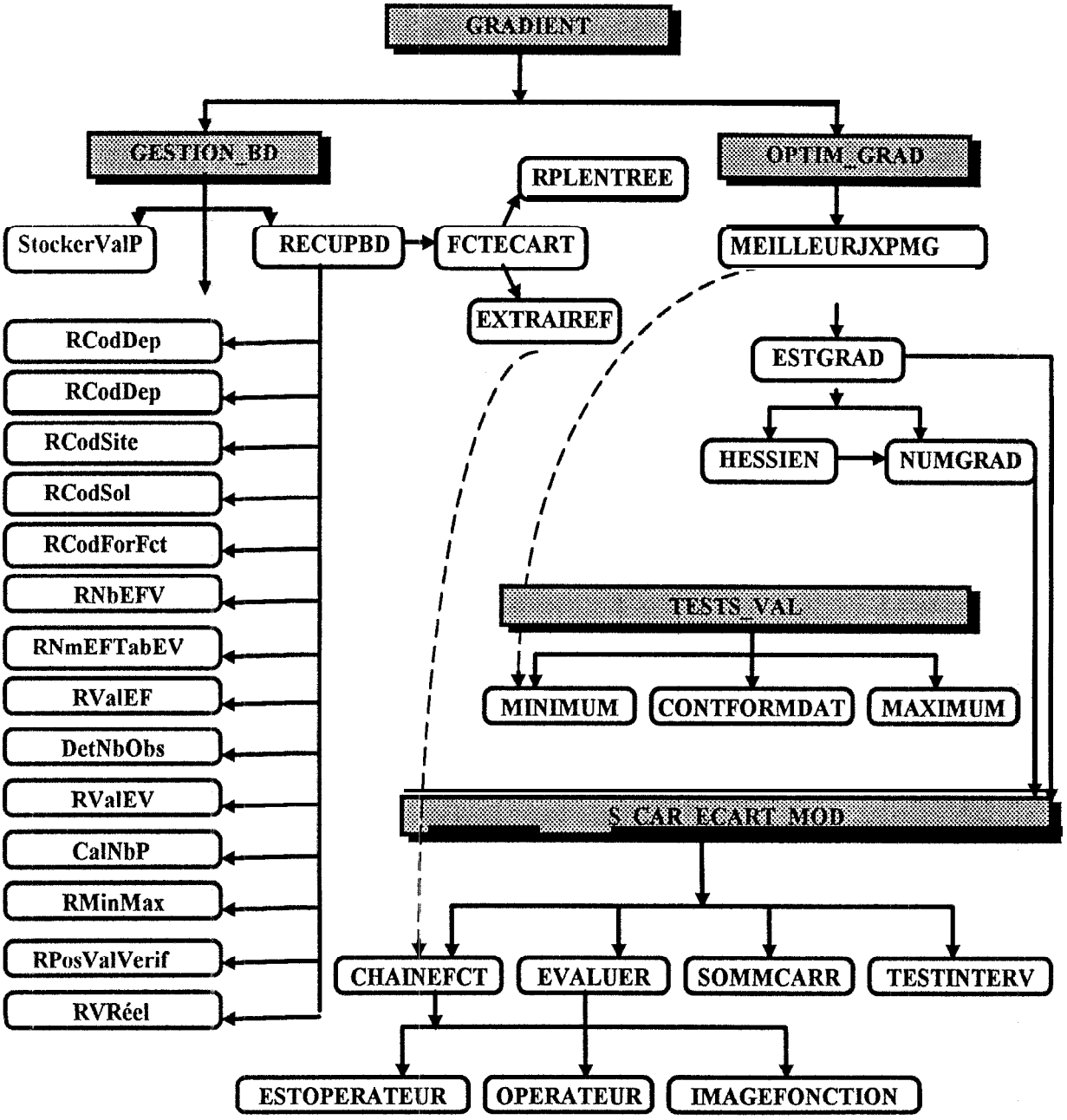
```
        FIN SI

        '~~ Cotangente hyperbolique inverse
        SI IDFonction = "HARCCOTAN" ALORS
                IMAGEFONCTION = Log((Nb + 1) / (Nb - 1)) / 2
        FIN SI
FIN
```

Jeu d'essai

IDFonction = sin
Nb = 2
IMAGEFONCTION = 0.03489

4.5.3 Le système GRADIENT

453.1 Architecture générale



Module
Fonction / Procédures
→ Appel à certaines fonctions d'un autre module
- 3 Appel à une fonction d'un autre module

4.5.3.2 Description des modules

4.5.3.2.1 GRADIENT

Problématique:
Déterminer des paramètres permettant de minimiser l'écart entre les données résultats de simulation et celles mesurées sur le terrain avec la méthode du gradient.

Analyse du problème
La précision des modèles de simulation du développement **des** cultures dépend des valeurs des paramètres, contrôlant leurs réponses aux conditions de l'environnement. Ces modèles sont représentés sous forme de fonction mathématique.
La méthode des moindres carres est utilisée pour le calcul des paramètres. Elle consiste en la mise en place d'une fonction exprimant l'écart entre les valeurs observées sur le terrain et celles obtenues avec les modéles de simulation.
L'optimisation revient à la recherche de paramètres permettant de minimiser la fonction des moindres carrés.
La méthode du gradient permet d'obtenir un meilleur jeu de paramètres, mais il est très lent au niveau de l'optimum.

Algorithme sous forme de texte

DEBUT
Appel de RECUPBD.
Appel de MEILLEURJXPMG.
Appel de StockerValP.
Affichage de la valeur de chaque paramètre et du coefficient de détermination dans **RESULTATS D'OPTIMISATION** sur l'écran.
FIN

Fonctions ou procédures appelées :
RECUPBD du module GESTION_BD du système simplex
MEILLEURJXPMG
StockerValP du module GESTION-BD du système simplex

Fonctions ou procédures appelantes: Néant.

Paramètres en entrée
NCulture : Nom de la variété de culture dont on veut calculer les paramètres.
NDep : Nom du dépatternent.
NSite : Nom du site.
NSol : Type de sol.
NFonction : Nom de la fonction de modélisation.
BaseParam : Base de données des simulations.

Paramètres en sortie

CD : Coefficient de détermination.
XMeilG(NbParam): Tableau à une dimension contenant le meilleur jeu de paramètres obtenu.

Paramètres intermédiaires

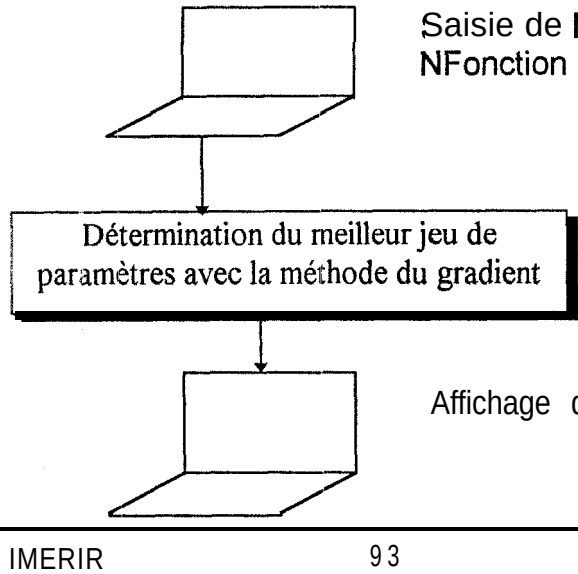
NbParam : Nombre total des paramétres du modèle.
NParam() : Tableau à une dimension contenant le nom des paramètres du modèle.
Fct : Fonction des moindres carrés sous forme de chaîne de caractères.
SeuilErr : Valeur indiquant la condition d'arrêt de l'algorithme du gradient.
St : Somme totale des valeurs de vérification.
CFct : code de la fonction du modèle.
NbreObS : Nombre total des observations.
SMeilS : Valeur de la fonction des moindres carrés avec le meilleur jeu de paramètre obtenu.
MinMax(NbParam,2) : Tableau à deux dimensions contenant la valeur minimale et maximale de chaque paramètre du modèle

Algorithme détaillé

DEBUT
RECUPBD (NDep, NSite, NSol, NCulture, NFonction, NbParam, MinMax() NParam(), Fct, St, NbreObS, CFct, CV, BaseParam)
MEILLEURJXPMG (SeuilErr, NbParam, MinMax(), XMeilG(), Fct, SMeilG)
CD = 1 - (SMeilG / St)
StockerValP (NbParam, XMeilG(), BaseParam)
Afficher les résultats à l'écran : SMeilG(), CD.
FIN

Jeu d'essai

PS (Poids sec) est une fonction de modélisation du mil, une culture du site de Saint-Louis. Le mil est cultivé sur un sol de type DIOR. Le meilleur jeu de paramètres obtenu est affiché à l'écran



4.5.3.2.2 OPTIM GRAD

Ce module permet d'obtenir un meilleur jeu de paramètres avec la méthode du gradient.

4.5.3.2.2.1 Procédure MEILLEURJXPMG

Problématique,
Obtenir le meilleur jeu de paramètres avec la méthode du gradient.

Analyse du problème
La méthode du gradient démarre avec un jeu de paramètres obtenu de manière aléatoire. Cette méthode permet d'obtenir un meilleur jeu de paramètres par un calcul de dérivées partielles.

Aigorithme général
DEBUT
TANT QUE le nombre d'itérations fixé n'est pas atteint ***FAIRE***
Mettre en place le jeu de paramètres de départ de manière aléatoire
Appeler la fonction de mise en place du gradient
Stocker la valeur de la fonction des moindres carrés pour le point résultat de la méthode du gradient dans un tableau T
FIN TANT QUE
Comparer les valeurs stockées dans T et prendre la meilleure
FIN

Fonctions ou procédures appelées
ESTGRAD
MINIMUM du module TESTS-VAL du système SIMPLEX.

Fonctions ou procédures appelantes : Néant.

Paramètres en entrée
SErr : Valeur indiquant la condition d'arrêt de l'algorithme du gradient.
NbP : Nombre de paramètres du modèle.
Mm() :Tableau à deux dimensions contenant la valeur minimum et maximum de chaque paramètre du modèle.
Fct : Fonction des moindres carrés sous forme de chaîne de caractères.

Paramètres en sortie
MP() : Tableau à une dimension contenant le meilleur jeu de paramètres obtenu.
VMeil : Valeur de la fonction des moindres carrés pour le meilleur jeu de paramètres obtenu.

Paramètres intermédiaires
I : Indice de parcours de la boucle principale (elle va de 0 à Nblter) et des lignes du tableau XMeilS().

J : Indice de parcours des lignes du tableau MP() et des colonnes du tableau XMeilS().
K : Indice de parcours des lignes du tableau Mm() et des colonnes du tableau MP().
PMin : Positian de la plus petite valeur du tableau SMeilS().
Min : La plus petite valeur du tableau SMeilS().
Nblter : Nombre de fois où la méthode du gradient est appelée avec un jeu de paramètre de départ différent à chaque fois.
SMeil : Valeur de la fonction (somme des carrés des écarts) pour le meilleur jeu de paramètres.
Limite : Constante correspondant au chiffre le plus élevé acceptable par le logiciel de développement.
Erreurs : Code erreur.
JxPm(NbP) : Tableau du jeu de paramètre aléatoire.
SMeilG(Nblter) : Tableau des meilleurs valeurs obtenues.
XMeilG(Nblter, NbP) : Tableau des jeux de paramètres dont les valeurs Minimisent la fonction (somme des carrés des écarts).

Algorithme

```
DEBUT
    Initialiser Limite, Nblter = 5
    FAIRE
        POUR I = 0 à Nblter
            '~~ Mise en place des jeux de paramètres
            POURK=OàNbP-1
                JxPm(J)= Mm(J,0) <= Valeur <= Mm(J, 1)
            FIN POUR
            Erreurs = 0
            FIN POUR
            '~~ Appel de la méthode du gradient
            ESTGRAD (SErr, NbP, JxPm(), MP(), Mm(), SMeil, Fct, Erreurs)
            SI Erreurs = -1 ALORS
                SMeilG(I) = Limite
            SINON
                SMeilG(I) = SMeil
                POUR I = 0 à NbP - 1
                    XMeilG(I, J) = MP(J)
                FIN POUR
            FIN SI
        FIN POUR
        '~~ Recherche de la valeur minimum parmi les résultats obtenus
        MINIMUM (Nblter, SMeilG(), PMin, Min)
        '~~ Mise en place de MP
        POURI=OàNbP-1
            MP( I) = XMeilG(PMin, I)
        FIN POUR
        VMeil = SMeilG(PMin)
        Effacer les tableaux JxPm(), XMeilG() et SMeilG()
    TANT QUE VMeil <> Limite
FIN
```

4.5.3.2.2.2 Procédure ESTGRAD

Problématique

Fournir à partir d'un jeu de paramètres (donc un point), à chacun des paramètres du modèle, une valeur permettant de minimiser la fonction des moindres carrés avec la méthode du gradient.

Analyse du problème

La méthode du gradient est réalisée à partir d'un jeu de paramètre. Le gradient exprime les dérivées partielles d'ordre 1 de la fonction des moindres carrés par rapport aux paramètres. Chaque paramètre évolue jusqu'à une valeur perrnettant de minimiser la fonction des moindrs carrés. Les dérivées partielles d'ordre 2 (le hessien) permettent de déterminer ce pas d'évolution.

Algorithme sous forme de texte

DEBUT

TANT QUE on n'est pas arrivé au seuil fixé au départ du programme ***FAIRE***

Déterminer un jeu de paramètre au départ
Evaluer le gradient
Evaluer le hessien
Déterminer le pas d'évolution des paramètres
Mettre en place les nouveaux paramètres

FIN TANT QUE

FIN

Fonctions ou procédures appelées

HESSIEN
NUMGRAD
TESTINTERV du module S-CAR-ECART-MOD du système SIMPLEX.
EVALUER du module S-CAR-ECART-MOD du système SIMPLEX.
SOMMCARR du module S-CAR-ECART-MOD du système SIMPLEX.

Fonctions ou procédures appelantes : Néant

Paramètres en entrée

SE : Valeur indiquant la limite de réalisation du gradient
NE : Nombre de paramètres du modèle.
XE() : Tableau des différents jeux de paramètres.
B() : Tableau contenant la valeur minimum et maximum de chaque paramètre.
SC : Fonction des moindres carrés sous forme de chaîne de caractères.

Paramètres en sortie

MG(): Tableau contenant le meilleur jeu de paramètres obtenu.
VF : Valeur de la fonction des moindres carrés pour le meilleur jeu de paramètre obtenu.
Ers : Code de l'erreur.

Paramètres intermédiaires

U(NE)	: Vecteur indiquant la direction à suivre.
Norme	: Norme du vecteur gradient.
Lam bda	: Valeur permettant de minimiser la fonction des moindres carrés.
XB(NE+1,NE)	: Valeurs antérieures des paramètres.
GE(NE)	: Valeur du gradient.
G(NE, 1)	: Gradient sous forme matricielle.
GT(1, NE)	: Gradient transposé sous forme matricielle.
LDenom1(1, NE)	: Résultat de la multiplication de GT et H.
LDenom2	: Résultat de la multiplication de LDenom1 et G.
TotLD	: Total utilisé dans le calcul des multiplications de matrice.
Nm(NE)	: Intervient dans le calcul de la norme du vecteur gradient.
Point(NE)	: Tableau contenant les coordonnées du point à optimiser.
PointB(NE)	: Tableau contenant les coordonnées du point antérieur obtenu.
Delta(NE)	: Indique le pas de diminution de chaque paramètre.
I	: indice de parcours de tableau.
J	: Indice de parcours de tableau.
K	: Indice de parcours de tableau.
L	: Indice de parcours du tableau Point().
M	: Indice de parcours des lignes du tableau XB(().
Math(NE, NE)	: Matrice Hessienne.
Cpt	: Compteur pour comparaison de paramètres.
FCT1	: Valeur antérieure de la fonction des moindres carrés.
FCT2	: Valeur de la fonction des moindre carrés.
FB	: SC après remplacement de chaque paramètre par sa valeur.

Algorithme détaillé

```
DEBUT
    POUR I = 0 à NE - 1
        Point(I) = XE(I)
    FIN POUR
    FCT2 = 1
    POUR I = 0 à NE - 1
        Delta(I) = 0
        U(I) = 0
    FIN POUR
    Lambda = Cl
    FAIRE
        FCT1 = FCT2
        '~~ Bufferisation de l'ancien point
        POUR I = 0 à NE - 1
                PointB(I) = Point(I)
        FIN POUR
        '~~ Mise en place du nouveau point
        POUR I = 0 à NE - 1
                Point(I) = Point(I) + Delta(I)
        FIN POUR
```

```
'~~ On sort si le point n'évolue plus
SI NbIterG <> 0 ALORS
        Gpt=O
        TANT QUE Point(Gpt) = PointB(Gpt) et (Gpt < NE) FAIRE
                Gpt -- Gpt + 1
        FIN TANT QUE
        SI Gpt = NE ALORS
                Sortir du tant que
        FIN SI
FIN SI
'~~ Waleur de la fonction pour ce point
TESTINTERW (NE, B(), Point())
SOMMGARR (SC, NE, Point(), FB)
FGT2 = EVALUER. (FB, Ers)
SI Ers = -1 ALORS
        Sortir de la procédure
FIN SI
SI FGT2 < 0 ALORS
        '~~ Mise en place de MG
        POUR I = 0 à NE - 1
                MG(I) = PointB(I)
        FIN POUR
        '-- Mise en place de VF
        VF = FGT2
        Sortir de la procédure
FIN SI
'~~ Calcul du gradient numérique
NUMGRAD (NE, Point(), GE(), B(), SC)
'~~ Mise en place de U
Nm(0) = GE(0) ^ 2
POUR I = 1 à NE - 1
        Nm(I) = Nm(I - 1) + (GE(I) ^ 2)
FIN POUR
Norme = racine carrée (Nm( NE - 1))
SI Norme = 0 ALORS
        POUR I = 0 à NE - 1
                U(I) = 0
        FIN POUR
SINON
        POUR I = 0 à NE - 1
                U( I) = (- GE( I)) / Norme
        FIN POUR
FIN SI
'~~ Mise en place de G et GT
POUR I = 0 à NE - 1
        G(I, 0) = GE(I)
        GT(0, I) = GE(I)
FIN POUR
'~~ Détermination de la matrice hessienne
HESSIEN (Point(), B(), NE, SC, Math())
```

```
            '~~ Mise en place de LDenom1
            POUR I = 0 à NE - 1
                  TotLD = 0
                  POUR J = 0 à NE - 1
                        TotLD = TotLD + (GT(0, J) * Math(J, I))
                  FIN POUR
                  LDenoml (0, I) = TotLD
            FIN POUR
            '~~ Mise en place de LDenom2
            LDenom2 = 0
            POUR I = 0 à NE - 1
                  LDenom2 = LDenom2 + (LDenom1(0, I) * G(I, 0))
            FIN POUR
            '~~ Mise en place de Lambda
            SI LDenom2 = 0 ALORS
                  Lambda = Norme ^ 3
            SINON
                  Lambda = (Norme ^ 3) / LDenom2
            FIN SI
            '~~ Mise en place de Delta
            POURI=OàNE-1
                  Delta( I) = Lambda * U( I)
            FIN POUR
            SI (FCT2 = 0) ALORS
                  'Mise en place de MG
                  POUR I = 0 à NE - 1
                        G( I) = Point(I)
                  FIN POUR
                  '~~ Mise en place de VF
                  VF = FCT2
                  Sortir du TANT QUE
            FIN SI
      TANT QUE Valeur absolue ((FCT2 - FCT1) / FCT2) <= SE
      '~~ Mise en place de GE()
      POUR I = 0 à NE - 1
            MG(I) = Point(I)
      FIN POUR
      '-- Mise en place de VF
      VF = FCT2
FIN
```

4.5.3.2.2.3 Procédure NUMGRAD

Problématique

Fournir la valeur du gradient numérique (dérivées partielles d'ordre 1) pour un jeu de parametres donné.

Analyse du problème

Le gradient numérique f' d'une fonction f par rapport à une variable x est donne par la formule :

$$f'(x) = \lim_{x \to h} (f(x+h)-f(x-h))/2*h$$

$$\text{gradient} = \begin{vmatrix} \frac{\partial f}{\partial p1} \\ \frac{\partial f}{\partial p2} \\ \cdots \\ \frac{\partial f}{\partial pn} \end{vmatrix}$$

Voir Dossier d'analyse provisoire.

Algorithme sous forme de texte

DEBUT

Fixer h

POUR chaque paramètre p

exprimer f(P)

FIN POUF?

FIN

Fonctions ou procédures appelées :

EVALUER du module S_CAR_ECART_MOD du système SIMPLEX.
SOMMCARR du module S_CAR_ECART_MOD du système SIMPLEX.
TESTINTERV du module S_CAR_ECART_MOD du système SIMPLEX.

Fonctions ou procédures appelantes

ESTGRAD
HESSIEN

Paramètres en entrée

NpG: Nombre de paramètres du modèle.
PG(): Tableau des paramètres.
BnG():Tableau des bornes inférieures et supérieures de chaque paramètre.
FM: Formule de la fonction de modélisation sous forme de chaîne de caractères.

Paramètres en sortie

GPG() : Valeur du gradient.

Paramètres intermédiaires

TG(lO, 10) : Tableau de travail pour le calcul de GPG().
PGS(NpG) : Buffer des valeurs de PG().
I : Indice de parcours de boucle et de tableau
J : Indice de parcours de boucle et de tableau
EIG : Sert au calcul de la dérivée
EIG2 : EIG au carrée.
IG : Indice de boucle et de parcours de tableau
Hn : Constante de dérivation
PERRG : Valeur de l 'erreur.
FACG : Sert au calcul de la dérivée
ERRG : Valeur de l 'erreur.
TEMPG1 : Variable temporaire
TEMPG2 : Variable temporaire
FRG : Fonction après un remplacement de chaque paramètre par sa valeur.
FONCTGI : Valeur de ta fonction pour X + h.
FONCTG2 : Valeur de la fonction pour X - h.
E : Code de l'erreur.

Algorithme

```
DEBUT
    EIG = 2.4
    EIG2 = EIG * EIG
    POUR I = 0 à NpG - 1 'Sauvegarde des paramètres initiaux
        PGS( 1) = PG( I)
    FIN POUR
    POURIG=OàNpG-1
        POURI=OàNpG-1
                PG(I) = PGS(I)
        FIN POUR
        '~~ Initialisation du tableau pour le calcul du gradient GPG()
        POUR I = 0 à 9
            POUR J = 0 à 9
                TG(I, J) = 0
            FIN POUR
        FIN POUR
        Hn = 5
        PG( IG) = PGS( IG) + Hn
        '~~ Evaluation de la fonction
        TESTINTERV NpG, BnG(), PG()
        SOMMCARR Fm, NpG, PG(), FRG
        FONCTGI = EVALUER(FRG, E)
        PG(IG) = PGS(IG) - Hn
        '~~ Evaluation de la fonction
        TESTINTERV NpG, BnG(), PG()
        SOMMCARR Fm, NpG, PG(), FRG
        FONCTG2 = EVALUER(FRG, E)
        '~~ Fonction de dérivation
        TG(0, 0) = (FONCTGI - FONCTG2) / (2# * Hn)
```

```
            PERRG = 1E+30
            POUR I = 1 à 9
                Hn = Hn / EIG     'Diminution de la constante de dérivation
                PG(IG) = PGS(IG) + Hn
                '~~ Evaluation de la fonction
                TESTINTERV NpG, BnG(), PG()
                SOMMCARF! Fm, NpG, PG(), FRG
                FONCTG1 = EVALUER(FRG, E)
                PG(IG) = PGS(IG) - Hn
                '~~ Evaluation de la fonction
                TESTINTERV NpG, BnG(), PG()
                SOMMCARF! Fm, NpG, PG(), FRG
                FONCTG2 = EVALUER(FRG, E)
                TG(0, I) = (FONCTG1 - FONCTG2) / (2# * Hn)
                FACG = EIG.2
                POUR J = 1 à I
                    G(J, I) = (TG(J-1, I)* FACG-TG(J-1, I-1))/(FACG-1#)
                    FACG = EIG2 * FACG
                    TEMPGI = Valeur absolue(TG(J, I) - TG(J - 1, I))
                    TEMPG2 = Valeur absolue(TG(J,I) - TG(J - 1, I - 1))
                    ERRG = TEMPGI
                    SI (TEMPG2 > ERRG) ALORS
                        ERRG = TEMPG2
                    FSI
                    SI (EF!RG <= PERRG) ALORS
                        PERRG = ERRG
                        GPG(IG) = TG(J, I)
                    FIN SI
                FIN POUR
                SI (Valeur absolue(TG(I,I) - TG(I - 1, I - 1)) >= 2# *
                                                        PERRG) ALORS
                    Sortir du POUR
                FIN SI
            FIN POUR
        FIN POUR
        '~~ Remise en place du tableau des paramètres
        POUR I = 0 à NpG - 1
            PG(I) = PGS(I)
        FIN POUR
FIN
```

4.5.3.2.2.4 Procédure HESSIEN

Problématique

Fournir la matrice hessienne.

Analyse du problème

La matrice hessienne exprime les dérivées partielles d'ordre 2 de la fonction des moindres carres par rapport à chaque paramètre.

$$\text{Hessien} = \begin{vmatrix} \frac{\partial^2 f}{\partial p1 \partial p2} & \cdots & \frac{\partial^2 f}{\partial p1 \partial pn} \\ \cdots & \frac{\partial^2 f}{\partial pi \partial pj} & \cdots \\ \frac{\partial^2 f}{\partial pn \partial p1} & \cdots & \frac{\partial^2 f}{dpndpn} \end{vmatrix}$$

Algorithme sous forme de texte

DEBUT

POUR chaque paramètre

Déterminer sous forme de chaîne de caractère l'expression de la dérivée partielle d'ordre1

Déterminer les dérivées partielles d'ordre 2

FIN POUR

FIN

Fonctions ou procédures appelées

NUMGRAD

Fonctions ou procédures appelantes

ESTGRAD

Paramètres en entrée

PtG : Tableau contenant les coordonnées du point.

BordG : Tableau des valeurs minimum et maximum de chaque paramètre.

NpmG : Nombre de paramètres du modèle.

SCarG : Fonction des moindres carrés.

Paramètres en sortie

MhG : Matrice hessienne.

Paramètres intermédiaires:

Grad(NpmG) : Résultats obtenus avec le gradient
LgSCarGBuf : Longueur de la fonction dont on a remplacé les paramètres par leur valeur
LPartie1G : Longueur de la Partiel G
PosPG : Position de chaque paramètre
I : Indice de boucle
ERechP : Code du paramètre dans la fonction **ou** le signe **à remplacer**
RemplDG : Variable de remplacement dans la fonction
Partiel G : Partie de la fonction se trouvant juste avant le code du paramètre
Partie2G : Partie se trouvant juste avant le code du paramètre + valeur du paramètre
Partie3G : Partie de la fonction se trouvant juste après le code du paramètre
PartG(2) : SCarG après remplacement de PtG(I) par PtG(I) + h et PtG(I) - h
SCarGBuf : Fonction dont on a remplacé les paramètres par leur valeur
h : Constante de dérivation
DenomG : Dénominateur de la fonction pour la dérivation
J Indice de la boucle de mise en place du tableau PartG
AjoutG : + h ou - h
DRehG Indique la position du début de la recherche
DerivPartG(NpmG) : Tableau contenant le formulation de chaque dérivée partielle

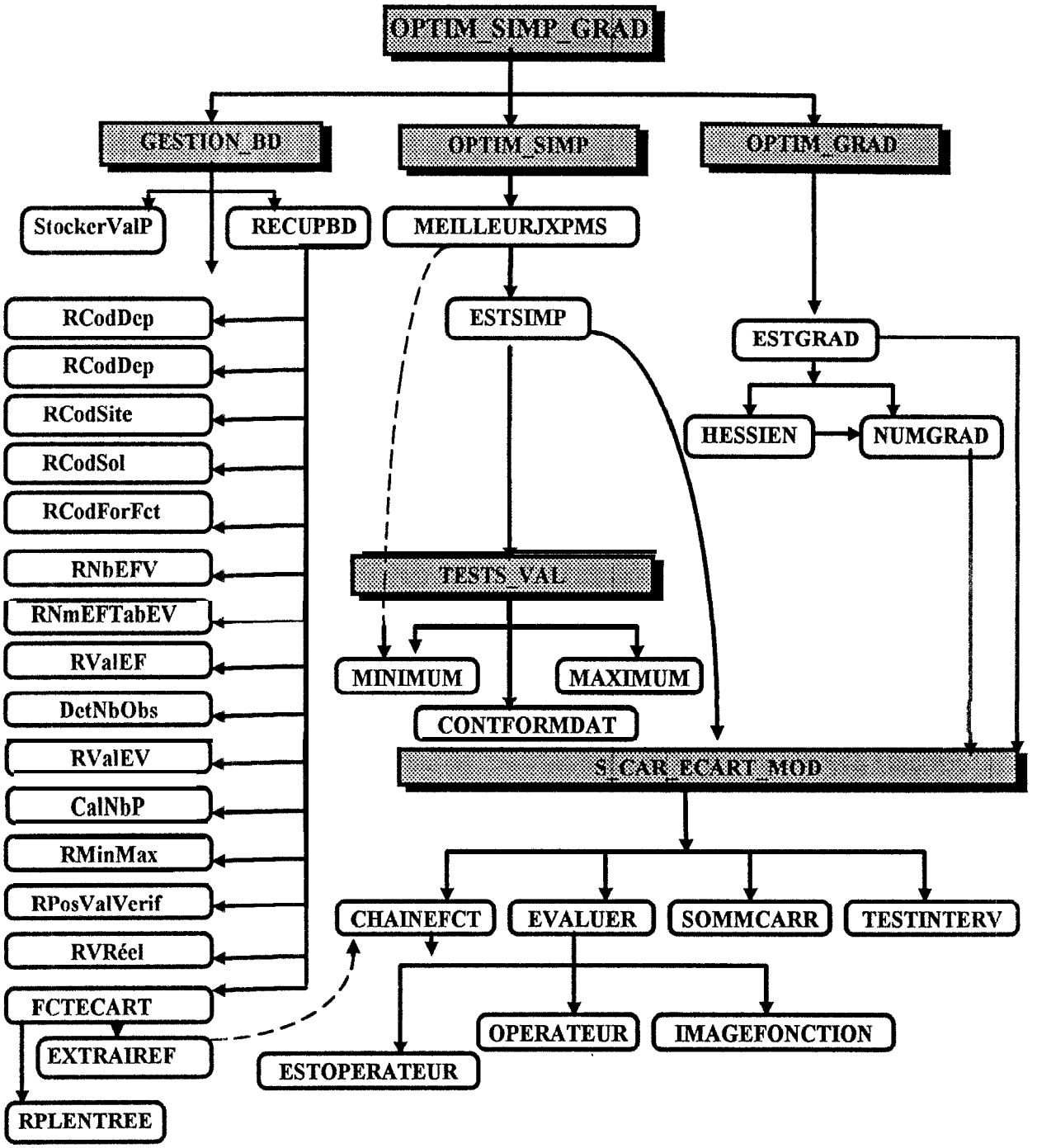
Algorithme

```
DEBUT
    h = 5
    DenomG = 2 * h
    F = O
    ERechP = ""
    RemplDG = ""
    '~~ Mise en place de la formulation des dérivées partielles
    POUR I = 0 à NpmG - 1
        ERechP = "X(1)"
        AjoutG = "+      "
        Ajouter la valeur de h à AjoutG
        Enlever les espaces au début et à la fin de la chaîne AjoutG
        POUR J = 0 à 1
            DRehG = 1
            SCarGBuf = SCarG
            Enlever les espaces au début et à la fin de SCarGBuf
            PosPG = 1
                TANT QUE PosPG = 0 FAIRE
                Calcul de LgSCarGBuf
                Remplacer le I de ERech par sa valeur
                RemplDG = ERechP & AjoutG
                Détermination de PosPG
                SI PosPG = 0 ALORS
```

```
                    Sortir du tant que
                FIN SI
                Détermination de Partiel G
                LPartie1G = Longueur(Partie1G)
                Partie2G = Partie1G & "(" & RemplDG & ")"
                Détermination de Partie3G
                SCarGBuf = Partie2G & Partie3G
                DRehG = LPartie1G + Longueur(RemplDG) + 2
            FIN TANT QUE
            PartG(J) = SCarGBuf
            AjoutG = "- "
            Ajouter la valeur de h à AjoutG
            Enlever les espaces au début et à la fin de AjoutG
        FI N POUR
        Mettre DenomG sous forme de chaîne de caractères
        Enlever les espaces au début et à la fin de la chaîne DenomG
        DerivPartG(I) = "(" & "(" & "(" & PartG(0) & ")" & "-" & "(" &
        PartG( 1) & ")" & "/" & DenomG & ")"
        Enlever les espaces au début et à la fin de DerivPartG(I)
    FIN POUR
    '~~ Mise en place de la matrice hessienne
    POUR I = 0 à NpmG - 1
        NUMGRAD (NpmG, PtG(), Grad(), BordG(), DerivPartG(I))
        POUR J = 0 à NpmG - 1
            MhG(I, J) = Grad(J)
        FIN POUR
    FIN POUR
FIN
```

4.5.4 Le système SIMPLEX + GRADIENT

4.5.4.1 Architecture générale



Module
Fonction / Procédures
Appel à certaines fonctions d'un autre module
Appel à une fonction d'un autre module

4.5.4.2 Description des modules

4.5.4.2.1 OPTIM_SIMP_GRAD

Problématique:
Déterminer des paramètres permettant de minimiser l'écart entre les données résultats de simulation et celles mesurés sur le terrain.

Analyse du problème
La précision des modèles de simulation du développement des cultures dépend des valeurs des paramètres contrôlant leurs réponses aux conditions de l'environnement. Ces modèles sont représentés sous forme de fonction mathématique.
La méthode des moindres carrés est utilisée pour le calcul des paramètres. Elle consiste en la mise en place d'une fonction exprimant l'écart entre les valeurs observées sur le terrain et celles obtenues avec les modèles de simulation.
L'optimisation revient à la recherche de paramètres permettant de minimiser la fonction des moindres carrés. Deux outils statistiques sont successivement utilisés pour cela :
La méthode du simplex d'abord : permet d'obtenir rapidement un premier jeu de paramètres.
La méthode du gradient ensuite : lent, permet d'obtenir un meilleur jeu de paramètres.

Algorithme sous forme de texte

DEBUT
Appel de RECUPBD.
Appel de MEILLEURJXPMS
Appel de ESTGRAD
Appel de StockerValP.
Affichage de la valeur de chaque paramètre et du coefficient de détermination dans *RESULTATS D'OPTIMISATION* sur l'écran.
FIN

Fonctions ou procédures appelées :
RECUPBD du module GESTION-BD du système SIMPLEX
MEILLEURJXPMS du module OPTIM_SIMP du système SIMPLEX
ESTGRAD du module OPTIM_GRAD du système GRADIENT
StockerValP du module GESTION-BD du système SIMPLEX

Fonctions ou procédures appelantes: Néant.

Paramètres en entrée
NCulture : Nom de la variété de culture dont on veut calculer les paramètres.
NDep : Nom du département.
NSite : Nom du site.
NSol : Type de sol.
NFonction : Nom de la fonction de modélisation.
BaseParam : Chemin d'accès à la base de données des simulations

Paramètres en sortie

CD : coefficient de détermination.
XMeilG(NbParam) : Tableau à une dimension contenant le meilleur jeu de paramètres obtenu.

Paramètres intermédiaires

Es : Code erreur.
NbParam : Nombre total des paramètres du modèle.
NParam() : Tableau à une dimension contenant le nom des paramètres du modèle.
Fct Fonction des moindres carrés sous forme de chaîne de caractères.
SeuilErr Valeur indiquant la condition d'arrêt de l'algorithme du simplex.
St : Somme totale des valeurs de vérification.
CFct : code de la fonction du modèle.
NbreObS : Nombre total des observations.
SMeilG . Valeur de la fonction des moindres carrés avec le meilleur jeu de paramètre obtenu avec la méthode du gradient.
SMeilS : Valeur de la fonction des moindres carrés avec le meilleur jeu de paramètre obtenu.
MinMax(NbParam, 2) : Tableau à deux dimensions contenant la valeur minimale et maximale de chaque paramètre du modèle.
XMeilS(NbParam): Tableau à une dimension contenant le meilleur jeu de paramètres obtenu avec la méthode du simplex.

Algorithme détaillé

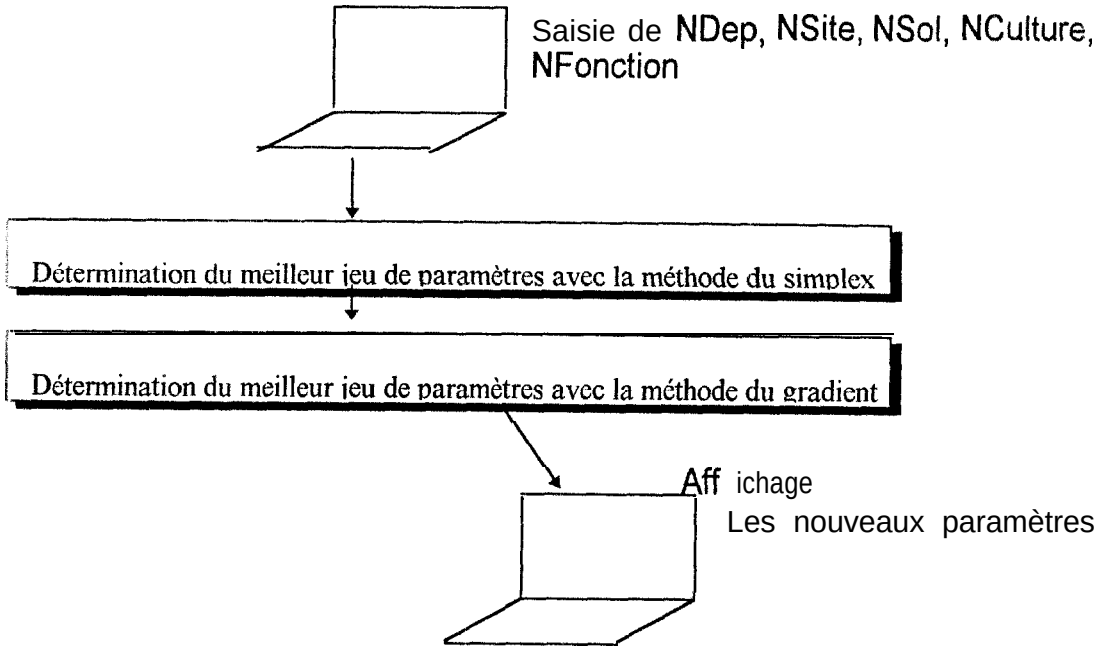
DEBUT

RECUPBD (NDep, NSite, NSol, NCulture, NFonction, NbParam, MinMax(), NParam(), Fct, St, NbreObS, CFct, CV, BaseParam)
MEILLEURJXPMS (SeuilErr, NbParam, MinMax(), XMeilS(), Fct, SMeilS)
ESTGRAD (SeuilErr, NbParam, XMeilS(), XMeilG(), MinMax(), SMeilG, Fct, Es)
CD = 1 - (SMeilG / St)
StockerValP (NbParam, XMeilG(), BaseParam)
Afficher les résultats à l'écran : SMeilG(), CD.

FIN

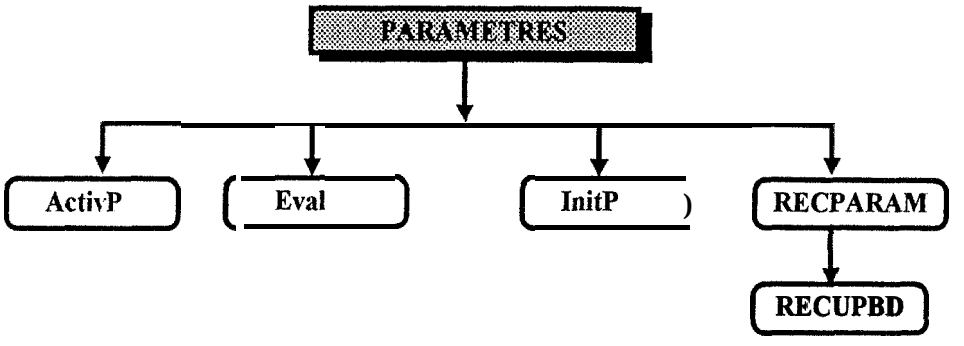
Jeu d'essai

PS (Poids sec) est une fonction de modélisation du mil, une culture du site de Saint-Louis. Le mil est cultivé sur un sol de type DIOR.
Le meilleur jeu de paramètres obtenu pour PS est stocké dans la base de donnée des simulations et est affiché sur l'écran


Saisie de NDep, NSite, NSol, NCulture, NFonction
Détermination du meilleur jeu de paramètres avec la méthode du simplex
Détermination du meilleur jeu de paramètres avec la méthode du gradient
Affichage
Les nouveaux paramètres

4.5.5 Le système PARAMETRES

4.5.5.1 *Architecture* générale



Module

Fonction / Procédures

Appel à une fonction

4.5.5.2 *Description des modules*

Problématique

Permettre à l'utilisateur de modifier la valeur minimale et maximale de chaque paramètre du modèle.

Analyse du problème

La table Tparamètres de la base de données des simulations renferme ies valeurs minimale et maximale initiales de chaque paramètre des modèles de simulation du développement des cultures. Si les valeurs des paramètres obtenues après optimisation ne sont pas satisfaisantes, l'utilisateur a la possibilité de modifier la valeur minimale et maximale de chacun d'eux. Ces nouvelles valeurs seront stockées dans la table TParamètres.

Fonctions ou procédures appelées :

RECPARAM
Eval
InitBtP
ActivP

Fonctions ou procédures appelantes: Néant.

Paramètres en entrée

NCulture : Nom de la variété de culture dont on veut calculer les paramètres.
NDep : Nom du département.
NSite : Nom du site.
NSol : Type de sol.
NFonction : Nom de la fonction de modélisation.
BaseParam : Base de données des simulations.

Paramètres en sortie : Néant

Paramètres intermédiaires

NbParam : Nombre total des paramètres du modèle.
NParam() : Tableau à une dimension contenant le nom des parametres du modèle.
Fct : Fonction des moindres carrés sous forme de chaîne de caractères.
CFct : code de la fonction du modèle.
CV : Code de la variété de culture
E : Code Erreur
MinMax(NbParam,2) : Tableau à deux dimensions contenant la valeur minimale et maximale de chaque paramètre du modèle.

Algorithme

DEBUT
Affichage de la fenêtre n°3
RECPARAM (NDep, NSite, NSol, NCulture, NFontion, NbParam, NParam(),MinMax(), Fct, CV, BaseParam)
Appel de InitBtP(NbParam)
Nom paramètre = NParam()
Attendre choix de l'utilisateur
TANT **QUE** choix **<> Quitter FAIRE**
Si choix **= Nom paramètre ou** choix **= Choix paramètre** *ALORS*
Afficher « Valeur min » et « Valeur max » de **Nom paramètre** choisi
ActivP (N° de **Choix paramètre)**
FIN SI
SI modification de **Valeur minimale ALORS**
Appel de Eval (Valeur min, E)
*FIN **SI***
SI modification de **Valeur maximale ALORS**
Appel de Eval (Valeur max, E)
SI E = -1 **ALORS** Afficher un message d'erreur **FIN SI**
***FIN** SI*
SI choix **= Enregistrer ALORS**
SI **Valeur minimale >= Valeur maximale** *ALORS*
Afficher un message d'erreur
SINON
Enregistrer la nouvelle valeur minimale et maximale

du paramètre

FIN SI

FIN SI

FIN TANT QUE

Fermer la fenêtre n°3 et retourner à la fenêtre n°2

FIN

4.5.5.2.1 Procédure RECPARAM

Problématique

Récupérer à partir de la base de données des simulations, les données nécessaires pour la modification des paramètres.

Fonctions ou procédures appelées :

RECUPBD

Fonctions ou procédures appelantes: Néant.

Paramétres en entrée

Paramètres en sortie : Néant

NbParam : Nombre total des paramètres du modèle.
Nparam() : Tableau à une dimension contenant le nom des paramètres du modèle.
CFct : Code de la fonction du modèle.
CV : Code de la variété de culture
MinMax(NbParam, 2) : Tableau à deux dimensions contenant la valeur minimale et maximale de chaque paramètre du modèle.

Paramètres intermédiaires :

NCulture : Nom de la variété de culture dont on veut calculer les paramètres.
NDep : Nom du département.
NSite : Nom du site.
NSol : Type de sol.
NFonction : Nom de la fonction de modélisation.
BaseParam : Base de données des simulations.

Algorithme

DEBUT

Initialiser NCulture, NDep, NSite, NSol, NFonction, BaseParam
RECUPBD (NDep, NSite, NSol, NCulture, NFonction, NbParam, MinMax(), NParam(), Fct, St, NbreObS, CFct, V, BaseParam)

FIN

4.5.5.2.2 Procédure InitBtP

Problématique
Déterminer les cases à cocher. Chaque case à cocher correspond à un paramètre du modèle. L'application d'autoparamétrisation permet l'utilrsation au maximum vingt paramètres.

Fonctions ou procédures appelées : Néant

Fonctions ou procédures appelantes: Néant.

Paramètres en entrée
N : Nombre total de paramètres du modèle.

Paramètres en sortie : Néant

Paramètres intermédiaires :
I : Indice de parcours boucle

Algorithme

```
DEBUT
    POUR I = 1 à N
        Rendre visible la case à cocher I
    FIN POUR
FIN
```

4.5.5.2.3 Procédure ActivP

Problématique

Activer la case à cocher choisie par l'utilisateur

Fonctions ou procédures appelées : Néant

Fonctions ou procédures appelantes: Néant.

Paramètres en entrée

Ind : Numéro de la case à cocher choisi par l'utilisateur.

Paramètres en sortie : Néant

Paramètres intermédiaires :

I : Indice de parcours boucle

Algorithme

DEBUT

Rechercher la case à cocher n°Ind

Activer la case à cocher n°Ind

FIN

4.5.5.2.4 Fonction EVAL

Problématique
Déterminer la validité d'une valeur tapée par l'utilisateur.

Fonctions ou procédures appelées : Néant

Fonctions ou procédures appelantes: Néant.

Paramètres en entrée
E : Valeur à tester.
ErrCode : Code de l'erreur.

Paramètres en sortie : Néant

Paramètres intermédiaires :
Tbl() : Tableau contenant les valeurs reconnaissables par l'application d'autoparamétrisation.

Algorithme
```
DEBUT
        Tbl() = -+01 23456789
        SI E appartient à Tbl () ALORS
                ErrCode = 0
        SINON
                ErrCode = -1
        FIN SI
FIN
```

4.6 STRATEGIES DE REALISATION ET REGLE DE CODAGE

L'espace disque est suffisant. Le répertoire de travail est divisé en sous-repertoires pour chaque version de l'application. Des sauvegardes régulières sont effectuées sur le SERVEUR du CERAAS.
Les procédures sont courtes et claires. Le code sera lisible donc bien commenté, cela facilitera la maintenance.

5. Modèle du poids sec des feuilles de la culture de Mil

Date d'édition : 19/09/1997.

Exemple de modélisation sur le mil (D. Annerose) à Montpellier le 21/04/1995
Modifié le 23/04/1997 par J.P.

Fonction de croissance : SI **Temp cum <= Temp sénes**
Alors **Ps=Psmax/(a+b*exp(c*Temp cum))**
Sinon **Ps=Psmax/(a+b*exp(c*Temp sénes)) - (Temp cum - Temp sénes)*s**

Fonction de croissance : Ps=Psmax/(a+b*exp(c*SomTemp))	
Valeurs initiales	
a	1
b	928.63
c	-0.0082
Psmax	296
Fonction de sénescence : DeltaPS=s*SomTemp (à partir de 1200°c)	
Valeurs initiales	
s	0.075
Temp. sénes.	1200
Ps à TSénes	**282.05**

Temp. (°c)	Temp cum. (°c)	Poids sec (g.pied)	Sénescence	Poids sec +sénescence	Modele
24	24	0.39	0	0.39	0.39
24	48	0.47	0	0.47	0.47
24	72	0.57	0	0.57	0.57
24	96	0.70	0	0.70	0.70
24	120	0.85	0	0.85	0.85
24	144	1.03	0	1.03	1.03
24	168	1.26	0	1.26	1.26
24	192	1.53	0	1.53	1.53
24	216	1.86	0	1.86	1.86
24	240	2.26	0	2.26	2.26
24	264	2.75	0	2.75	2.75
24	288	3.34	0	3.34	3.34
24	312	4.06	0	4.06	4.06
24	336	4.93	0	4.93	4.93
24	360	5.98	0	5.98	5.98
24	384	7.25	0	7.25	7.25
24	408	8.78	0	8.78	8.78
24	432	10.62	0	10.62	10.62
24	456	12.83	0	12.83	12.83
24	480	15.47	0	15.47	15.47
24	504	18.62	0	18.62	18.62
24	528	22.37	0	22.37	22.37
24	552	26.79	0	26.79	26.79
24	576	31.99	0	31.99	31.99
24	600	38.06	0	38.06	38.06
24	624	45.07	0	45.07	45.07
24	648	53.12	0	53.12	53.12
24	672	62.24	0	62.24	62.24
24	696	72.46	0	72.46	72.46
24	720	83.76	0	83.76	83.76
24	744	96.07	0	96.07	96.07
24	768	109.25	0	109.25	109.25
24	792	123.13	0	123.13	123.13
24	816	137.47	0	137.47	137.47
24	840	152.02	0	152.02	152.02
24	864	166.48	0	166.48	166.48
24	888	180.60	0	180.60	180.60
24	912	194.12	0	194.12	194.12
24	936	206.84	0	206.84	206.84
24	960	218.60	0	218.60	218.60
24	984	229.31	0	229.31	229.31
24	1008	238.93	0	238.93	238.93
24	1032	247.45	0	247.45	247.45

24	1104	266.98	0	266.98	266.98
24	1128	271.74	0	271.74	271.74
24	1152	275.78	0	275.78	275.78
24	1176	279.18	0	279.18	279.18
24	1200	282.05	0	282.05	282.05
24	1224	284.44	1.8	280.25	280.25
24	1248	286.44	3.6	278.45	278.45
24	1272	288.10	5.4	276.65	276.65
24	1296	289.48	7.2	274.85	274.85
24	1320	290.63	9	273.05	273.05
24	1344	291.57	10.8	271.25	271.25
24	1368	292.35	12.6	269.45	269.45
24	1392	293.00	14.4	267.65	267.65
24	1416	293.53	16.2	265.85	265.85
24	1440	293.97	18	264.05	264.05
24	1464	294.33	19.8	262.25	262.25
24	1488	294.63	21.6	260.45	260.45
24	1512	294.87	23.4	258.65	258.65
24	1536	295.07	25.2	256.85	256.85
24	1560	295.24	27	255.05	255.05
24	1584	295.37	28.8	253.25	253.25
24	1608	295.48	30.6	251.45	251.45
24	1632	295.58	32.4	249.65	249.65
24	1656	295.65	34.2	246.05	247.85
					246.05
24	1704	295.77	37.8	244.25	244.25
24	1728	295.81	39.6	242.45	242.45
24	1752	295.84	41.4	240.65	240.65
24	1776	295.87	43.2	238.85	238.85
24	1800	295.89	45	237.05	237.05
24	1824	295.91	46.8	235.25	235.25
24	1848	295.93	48.6	233.45	233.45
24	1872	295.94	50.4	231.65	231.65
24	1896	295.95	52.2	229.85	229.85
24	1920	295.96	54	228.05	228.05
24	1944	295.97	55.8	226.25	226.25
24	1968	295.97	57.6	224.45	224.45
24	1992	295.98	59.4	222.65	222.65
24	2016	295.98	61.2	220.85	220.85
24	2040	295.99	63	219.05	219.05
24	2064	295.99	64.8	217.25	217.25
24	2088	295.99	66.6	215.45	215.45
24	2112	295.99	68.4	213.65	213.65
24	2136	295.99	70.2	211.85	211.85
24	2160	295.99	73.8	210.05	210.05
24	2184	296.00		208.25	208.25