Réalisation de modèles d'efficacité liés à des scénarii d'emplois opérationnels

Philippe Davignon

CONSERVATOIRE NATIONAL DES ARTS ET METIERS
Arcnam – Region Centre
Centre d'Enseignement Principal d'Orléans

MEMOIRE

présenté en vue d'obtenir

le DIPLOME d'INGENIEUR CNAM

Spécialité : Informatique

Option : Systèmes d'information

par : Philippe Davignon

SUJET

Réalisation de modèles d'efficacité liés à des scénarii d'emplois opérationnels

Soutenu le : 07 Juillet 2010

Jury	:		
membres Cnam			
président	:	Professeur	Badran
rapporteur	:	Monsieur	Martinez
examinateur	:	Madame	Kahlem
examinateur	:	Monsieur	Poussin
membre TDA			
tuteur	:	Monsieur	Laurent

résumé

Ce document regroupe les méthodes et outils nécessaires à la réalisation de modèle d'efficacité, les méthodes pour la réalisation d'un logiciel de calcul de probabilité selon la méthode de Monte-Carlo, les différentes étapes de développement, les principales varations de résultats obtenus durant la réalisation du projet.

abstract

In this document you can find some methods and tools for calculate a effectiveness model, the software developpement steps and some results obtained during this project.

Efficacité , Monte-Carlo, effectiveness ,

Remerciements

Je tiens a remercier Mr Doignon directeur technique de TDA pour avoir accepté de me recevoir au sein de la direction technique, Mr Laurent et son équipe du département architecture et modélisation pour m'avoir accueilli parmi eux pendant toute la période de mon stage.
Je tiens à remercier aussi Mr Fraissigné pour son expertise, Mr Laurent et Mr Hespel pour avoir eu la patience de me lire.
Je tiens à remercier tout particulièrement ma femme, mes enfants et mes amis pour m'avoir soutenus et supportés pendant toutes ces années.

Présentation TDA

Quelques mots sur l'histoire du site

De Brandt à Thales, un savoir-faire forgé sur le terrain. C'est en 1915, en pleine guerre mondiale, qu'Edgar Brandt, ferronnier d'art devenu soldat, met au point ses premiers mortiers, de nombreuses autres inventions suivront : roquettes, armes, légères, antichars...

Pour mettre en œuvre ses solutions novatrices, en 1938, Edgar Brandt fait l'acquisition de terrains à La Ferté Saint Aubin pour y implanter son entreprise et constitue la SIEM (Société Industrielle d'Explosifs et de Mécanique). Il y construit une usine baptisée aussitôt par les solognots « Usine de Chevau », du nom du lieu-dit où elle se situe.

Qui sommes-nous aujoud'hui

La société TDA est toujours installée à La Ferté Saint Aubin, sur une surface de 470 hectares qui abrite son centre industriel, toutes les compétences et tous les outils son réunis sur ce même site : de la conception à l'intégration système.

TDA Armements SAS, filiale à 100 % de Thales, conçoit, fabrique et commercialise des systèmes d'armes terrestre et Aéroportés,des systèmes de protection des forces, des munitions ainsi que des composants de missile. TDA possède des participations dans deux entreprises :

- Junghans Microtec, 45% détenu par TDA et 55% par le groupe Dhiel spécialisé dans le domaine des fusées et dispositifs de mise à feu des missiles,
- Forges de Zebbruge, détenue à 100% par TDA, acteur dans le domaine des armements aéroportés calibre 70 mm, implanté à Herstal en Belgique.

Notre activité économique

TDA Leader européen dans le domaine des systèmes de roquette air-sol, vend sur deux principaux marchés ; l'export et la France à part égale pour un chiffre d'affaire en 2009 de 81 M€ avec un effectif de 340 personnes, et a deux produits phares, le mortier de 120 rayé, En service dans 25 armées terrestres dont 5 nations de l'OTAN, apporte une puissance de feu précise jusqu'à 13 km et Commandé en 2004 par le Marine Corps des USA, des roquettes Pour hélicoptère ou pour avion. TDA avec sa filiale FZ est le seul industriel capable de fournir des systèmes complets, en calibre 68 mm, et en calibre américain 70 mm (2.75") pour les missions d'appui feu air-sol. Ces roquettes sont en service sur les hélicoptères et avions de combat des forces armées de 55 pays occidentaux.

Table des matières

Chapitre 1

Abréviations et glossaire

Abréviations

AMV

Anti Matériel et Véhicule

CMMI

Capability Maturity Model + Integration (modèle de maturité de capacité et d'intégration)

Cnam

Conservatoire national des arts et métiers

GED

Gestion Electronique des Documents

GPS

Global Positionning System

NTF

Nouvelle Triangulation de la France

RGF93

Réseau Géodésique Français

2R2M

véhicule équipé d'un mortier de 120mm rayé, d'un système de chargement automatique de munitions et d'une unité de navigation inertielle.

Glossaire

A

Aéronautique

Ensemble des sciences et des techniques ayant trait à la navigation aérienne et à la construction des aéronefs

B

Balistique

Relatif à l'art de lancer des projectiles : *La théorie balistique* est la science qui étudie le mouvement des projectiles[D52]

E

Ellipsoïde de révolution

modèle géométrique couramment utilisé pour approximer une géoïde. Un ellipsoïde ne définit pas à lui seul un système géodésique. Il doit être calé par rapport au géoïde.

G

Géodésie

science qui a pour but l'étude et la détermination de la taille et de la forme de la Terre. Elle permet, entre autres, de mesurer la position (latitude, longitude et altitude) et le mouvement de points à la surface de la Terre.

Géoïde

Surface équipotentielle du champ de pesanteur terrestre. Le géoïde exprime une réalité physique : si un point est situé au dessus du géoïde l'eau s'écoulera de ce point vers le géoïde et inversement.

GPS

Système de positionnement par triangulation composé de 24 satellites. Ils sont en orbite autour de la terre sur 6 plans différents. Le système a été développé pour les besoins de positionnement en temps réel de l'armée américaine.

R

référentiel Galiléens

Il éxiste des référentiels privilégiés dans lesquels un point matériel isolé a un mouvement rectiligne uniforme. Ces référentiels sont dits galiléens. Dans ces référentiels un point isolé a une vitesse constante.

S

Système géodésique

Un système géodésique peut être défini par un ellipsoïde de révolution conventionnel (choisi de manière à approcher le géoïde). Dans un système géodésique ainsi défini, un point est localisé par ses coordonnées géodésiques, exprimées en valeurs angulaires par la latitude L, la longitude G, et la hauteur géodésique h mesurée suivant la normale à l'ellipsoïde.

T

théâtre d'opérations

Zone où se déroule des opérations militaires[D52]

théâtre d'opérations extérieures

Zone d'opérations militaires située hors de France[D52]

triangulation

la triangulation est une méthode de mesure reposant sur la résolution des triangles. Connaissant précisément la longueur d'un côté d'un triangle et les deux angles adjacents, les longueurs des deux autres côtés s'obtiennent par l'intermédiaire des formules trigonométriques.

Chapitre 2

Références, définitions et conventions

2.1 L'étude statistique des effets

L'effet d'une munition tient à sa précision et à son effet terminal. Ne pouvant maitriser sa balistique il est utilisé les probabilités pour définir une probabilté d'atteinte et une probabilité de neutraliser la cible.

Pour le cas d'une munition à fléchettes on définit que la cible est neutralisée si elle est atteinte par au moins une fléchette ayant une énergie supérieure au seuil de neutralisation. La probabilité de neutralisation fait alors intervenir une probabilité conditionnelle qui est la probabilité de neutralisation sachant que la cible est touchée.

$$\begin{aligned} \mathbb{P}(kill/toucher) &= \frac{\mathbb{P}(kill \cap toucher)}{\mathbb{P}(toucher)} \\ \text{or } kill &\subseteq toucher \\ \text{alors } \mathbb{P}(kill/toucher) &= \frac{\mathbb{P}(kill)}{\mathbb{P}(toucher)} \\ \text{donc } \mathbb{P}(kill) &= \mathbb{P}(kill/toucher) \cdot \mathbb{P}(toucher) \end{aligned}$$

L'unité de surface composant la zone observée n'est pas égale à la surface de la cible et on observe que la surface de l'ombre A_{vul} est la surface équivalente au sol de la cible en fonction de l'angle de la fléchette à une altitude H. Si le point d'impact de la fléchette est dans l'ombre projetée de la cible sur le sol, on considère que la cible a été touchée par la fléchette. On pose comme hypothèse que la cible est perpendiculaire à l'axe de la trajectoire de la fléchette.

$$A_{vul} = L_{cib} \cdot H \cdot \tan \alpha$$

La surface unitaire A_{ij} n'est pas égale à la surface de l'ombre de la cible, il est donc nécessaire de calculer le nombre de fléchettes ayant atteint la cible nbf_{cib} sur chaque surface unitaire. On ne connait que le nombre de fléchettes tombées par unité de surface donc on calcul le produit de la probabilité que la fléchette tombe dans la surface unitaire avec la probabilité de présence de la cible dans cette surface.

$$\mathbb{P}_{vul} = \begin{cases} \frac{A_{vul}}{A_{ij}} & \text{si } 0 < A_{vul} \leq A_{ij} \\ 1 & sinon \end{cases}$$

$$\mathbb{P}_{touch} = \mathbb{P}_{vul} \times \mathbb{P} = \frac{nbf_{cib}}{ntf}$$

La probabilité qu'une fléchette munie d'une énergie suffisante touche la cible est le produit entre la probabilité de toucher et la probabilité de neutraliser la cible sachant qu'elle est touchée et la probabilité de neutraliser la cible sachant qu'elle a été touchée est le rapport du nombre de fléchettes qui ont touchées la cible avec une énergie suffisante nbf_{ok} sur le nombre de fléchettes tombées dans dans la zone de vulnérabilité.

$$\begin{aligned} \mathbb{P}_{kill/touch} &= \frac{nbf_{ok} \cdot nbf_{cib}}{d_{ij} \cdot nbf_{cib}} \\ \mathbb{P}_{kill/touch} &= \frac{nbf_{ok}}{d_{ij}} \\ \mathbb{P}_{kill} &= \mathbb{P}_{touch} \times \mathbb{P}_{kill/touch} \\ \mathbb{P}_{kill} &= \frac{nbf_{ok}}{d_{ij}} \cdot \frac{\mathbb{P}_{vul} \cdot d_{ij}}{ntf} \\ \mathbb{P}_{kill} &= \mathbb{P}_{vul} \cdot \frac{nbf_{ok}}{ntf} \end{aligned}$$

Nous avons ainsi pour chaque unité de surface de la zone observée la probabilité de neutraliser une cible dont on connaît les dimensions.

Afin de connaître le nombre de fléchettes tombées par unité de surface, il a fallu étudier la balistique de la roquette et des fléchettes en utilisant les règles de la mécanique du point.

2.2 la mécanique du point

Aristote [1] définissait deux sortes de mouvement, le mouvement par nature des corps qui vont vers leur lieu naturel, par exemple un corps tombe parce que sont lieu naturel est vers le bas et le mouvement par contrainte qui est dû à un *moteur*, correspondant à notre concept de force, qui communique une vitesse proportionnelle à la force.

Galilée [2] s'opposera à la mécanique d'Aristote, il est convaincu de la valeur du système de Copernic. Il énonce la loi de l'inertie peu avant Newton, esquisse la notion d'accélération, introduit la notion de moment et étudie les lois du pendule, de la chute des corps et de la statistique des fluides. Durant tout le XVII^{e} siècle, son œuvre est complétée par Descartes, Fermat, Roberval, Torricelli, Pascal.

Ainsi Evangelista Torricelli [3] élève puis secrétaire de Galilée avec lequel il améliore la loi de la chute d'un corps, traite complétement le problème pour une trajectoire dans le vide.

Isaac Newton [4] est le fondateur de la mécanique classique, souvent appelée mécanique newtonienne. Il donne une forme logique parfaite à la mécanique, qui repose sur trois lois :

- principe de l'inertie *tout corps persévère dans l'état de repos ou de mouvement uniforme dans lequel il se trouve, à moins que quelque force n'agisse sur lui et ne le contraigne à changer d'état.*
- principe fondamental *les changements qui arrivent dans la quantité de mouvement sont proportionnels à la force motrice et se font dans la ligne droite où cette force a été imprimée.*
- principe de l'action et de la réaction *l'action est toujours opposée et égale à la réaction.*

Il explique ainsi un très grand nombre de phénomènes terrestres(mouvements des projectiles, variation de la pesanteur avec l'altitude,...). Les travaux de Newton accomplirent une véritable révolution scientifique. Les successeurs de Newton développèrent de nouveaux calculs importants. Citons parmi les principaux Euler, Lagrange, Jean et Daniel Bernoulli, Fourier, Gauss, Poisson, Cauchy, Coriolis, Magnus,...

Leonard Euler [5] est l'un des hommes de sciences les plus éminents du XVIII^{e} siècle, il fut le fondateur de l'analyse mathématique. Ses travaux concernent aussi bien les sciences physiques, la navigation que l'artillerie, l'astronomie,...

1. philosophe et savant grec 384-322 av. J.-C.
2. Galileo Galilei mathématicien, physicien, astronome italien 1564-1642
3. mathématicien et physicien italien 1608-1647
4. mathématicien, physicien, astronome anglais 1642-1727
5. mathématicien suisse 1707-1783

Jacques Bernoulli [6] établit les bases du calcul des probabilités et perfectionne le calcul différentiel et intégral.

Johann Carl Friedrich Gauss [7] a apporté une trés grandes contributions aux mathématiques, à l'astronomie et à la physique. Il est notamment un pionnier du calcul des probabilités, il a laissé son nom à la courbe en cloche sur laquelle s'appuie la science de la statistique.

Gaspard Coriolis [8] effectua de nombreuses recherches sur la cinématique. On lui doit notamment le théorème fondamental qui porte son nom. Il a trait au mouvement relatif des mobiles qui se déplacent autour d'un objet lui même en mouvement et sont influencés par ce mouvement. Ce théorème s'applique fréquement en géophysique et en balistique. Il définit *la force de Coriolis* qui permet de mesurer les déviations de tous les mouvements à la surface de la terre.

Heinrich Gustav Magnus [9] étudia la trajectoire d'une balle en rotation qui se déplace dans un fluide et mis en évidence un effet qui porte son nom *l'effet Magnus*.

Et Gustave Eiffel [10] trés connu pour avoir réalisé la tour Eiffel monument historique de 320m de haut symbole de notre capitale, est aussi cité pour avoir posé les bases d'une science moderne : l'aérodynamique. Ces travaux ont porté sur l'étude des hélices, la voilure et les projectiles en soufflerie. Ces travaux déboucheront sur la conception d'un avion monoplan à la fin de la première guerre mondiale.

Tous ces travaux nous permettent aujourd'hui d'écrire que la position à un instant donné d'un point est défini par le vecteur position $\vec{p}$.

Quand le point est en mouvement dans le repère terrestre ses coordonnées sont fonction du temps et sont appelées équations horaires du mouvement. La trajectoire d'un point mobile est l'ensemble des positions occupées successivement par ce point.

Le rapport entre la différence de deux vecteurs positions et la différence des instants associés a ces points est un vecteur qui représente par définition le vecteur vitesse instantanée du point mobile. Ce vecteur a la direction de la tangente à la trajectoire, le sens du mouvement et pour norme la mesure de la vitesse instantanée.

$$\vec{v} = \frac{d\vec{p}}{dt} \cdot dt$$

Le vecteur vitesse se calcul en le décomposant en coordonnées. Les coordonnées du vecteur vitesse d'un point mobile pour un instant donné sont les dérivées par rapport au temps, des coordonnées du vecteur position au même instant.

6. mathématicien et physicien suisse 1654-1705
7. mathématicien, astronome et physicien allemand 1777-1855
8. Ingénieur en chef des Ponts et Chaussées, professeur de mécanique générale et d'analyse géométrique à l'Ecole Centrale des Arts et Manufactures 1792-1873
9. physicien allemand 1802-1870
10. ingénieur chimiste francais 1832-1923

Le vecteur vitesse caractérise la variation du vecteur position du point mobile au cours du temps. Le vecteur vitesse varie lui aussi. Par définition on appelle vecteur accélération d'un point mobile à un instant donné le vecteur dérivée, par rapport au temps, du vecteur vitesse à cet instant. Le vecteur accélération caractérise les variations du vecteur vitesse au cours du temps.

$$\vec{a} = \frac{d\vec{v}}{dt} \cdot dt$$

Les coordonnées du vecteur accélération d'un point mobile à un instant sont les dérivées, par rapport au temps, des coordonnées du vecteur vitesse au même instant.

La masse d'un corps est la quantité de matière contenu dans le corps. Elle représente l'inertie du corps, c'est-à-dire la difficulté à le mettre en mouvement. Le centre d'inertie d'un système est le barycentre des masses qui le constituent. Pour un système en mouvement la quantité de mouvement est un vecteur égal au produit de la masse par le vecteur vitesse en son centre d'inertie. Si un ensemble de forces appliqué à un solide provoque une variation de sa quantité de mouvement il existe une relation telle que la somme des forces est égale au rapport de la variation de la quantité de mouvement sur la variation du temps. C'est une relation fondamentale de la dynamique. La masse restant constante la somme des forces appliquées à un solide est égale au produit de sa masse par le vecteur accélération de son centre d'inertie.

$$\sum \vec{F} = m\vec{a}$$

Le vecteur accélération a la même orientation que la somme des forces appliquées . Cette relation permet de déterminer le mouvement du centre d'inertie d'un solide, mais non son mouvement propre. Une étude particulière du mouvement propre est nécessaire. Cette étude nécessite l'utilisation d'un autre théorème, qui fait intervenir le moment par rapport à un axe fixe des forces appliqués au solide.

Ce théorème relie deux quantités physiques : le moment angulaire qui est la grandeur physique qui joue un rôle analogue à la quantité de mouvement dans le cas des rotations et la somme des forces appliquées au solide. Parmi ces forces il existe la force d'attraction terrestre, elle peut varier localement mais dans un espace restreint de quelques kilomètres, le champ de pesanteur peut être considéré comme uniforme et donc pour des expériences locales, nous considérons le vecteur champ de pesanteur comme constant, orienté vers le centre de la Terre. Sa norme peut varier en fonction de l'altitude d'environ 3% pour 100km. Le mobile se déplaçant dans l'air, celui ci lui oppose une force de freinage aussi appelée force de trainée. Elle dépend de la vitesse du mobile, de la surface du mobile exposée au frottement, de la masse volumique du fluide et du coefficient de trainée C_x souvent déterminé par des essais en soufflerie.

Des forces sont aussi dues au mouvement du solide par rapport au référentiel. Cela revient à définir un référentiel centré sur le mobile et à exprimer les variations du repère du mobile par rapport au référentiel sous forme d'une distance entre leurs origines respectives et une différence angulaire entre les axes. Les angles ainsi définis sont appelés les angles d'Euler. Dans ce repère on définira les rotations autour des axes comme, précession, nutation et rotation propre.

Lors de l'étude du mouvement d'un solide dans le champs de pesanteur terrestre avec une vitesse initiale non nulle et selon un angle de tir par rapport au plan horizontal on choisit un référentiel, un système qui définit le solide pour lequel on établit le bilan des forces appliquées afin de calculer pour chaque instant les coordonnées spatiales. On considère pour un lancer de courte durée (un trajet de quelques kilomètres) le référentiel terrestre comme galiléen. Le théorème du centre d'inertie permet de déterminer l'accélération du solide en tenant compte de l'ensemble des forces exprimées sur le système, la pesanteur, la loi de freinage, l'effet de Magnus en cas de rotation de mobile. Il convient ensuite d'exprimer les équations horaires du mouvement par dérivées successives. Nous obtenons ainsi les équations cartésiennes du mouvement. Ces équations mettent en relations différentes informations, l'angle de tir, la vitesse initiale, afin de calculer la flèche (l'altitude maximum du mobile sur la trajectoire), le temps de vol, la vitesse au point d'impact, les coordonnées du point d'impact...

Pour étudier le mouvement d'un solide il faut définir un référentiel.

2.3 Le référentiel terrestre

Depuis longtemps les hommes cherchent à définir la forme et mesurer les dimensions de la Terre afin de réaliser des cartes, de localiser un point et se déplacer sur sa surface. La science qui regroupe ces recherches est la géodésie. Elle a commencé, au XVII^{e} siècle, lorsqu'un hollandais, Snellius, a trouvé un procédé permettant de mesurer la longueur d'un arc de méridien terrestre par une méthode qui est encore en usage de nos jours,la triangulation. La géodésie procède de la façon suivante : supposant que la Terre est un ellipsoïde parfait, on peut alors déterminer par le calcul, pour chaque point du globe, un certain nombre d'éléments, longitude, latitude, altitude, accélération de la pesanteur, distance à un autre point de référence.

Cette situation conduit à introduire un système d'axes cartésien ayant comme origine le centre de la Terre et tournant avec elle. Ce système d'axes est appelé système de référence terrestre. Sa construction est essentiellement implicite car aucun système de mesure ne permet directement de donner de positions exprimées dans le système terrestre. Les positions résultent toujours d'un calcul effectué à partir d'observations de triangulation. Le système de positionnement le plus connu, issu de ces méthodes de calculs, est le GPS.

Les systèmes de référence terrestres sont intimement liés aux mesures qui permettent de le construire. Le système est construit en posant des conditions qu'il doit vérifier. Les observations n'étant pas parfaites, ces conditions ne sont qu'approximativement vérifiées. C'est la raison pour laquelle il existe un grand nombre de systèmes de référence différents. Ainsi en France deux systèmes de référence coexistent : le système NTF, ancien système national réalisé à partir de mesures terrestres et le système RGF93 réalisé à partir de mesures satellitaires.

La représentation de la surface de la Terre passe par l'adoption d'un modèle de Terre aussi proche que possible de la réalité. En adoptant cette surface d'approximation, on cherche à cartographier la Terre en précisant ces irrégularités par rapport à cette surface(l'altitude) et en appliquant les détails horizontaux directement sur la surface. La surface de référence est aussi choisie de façon à être relativement simple. On choisit traditionnellement un ellipsoïde de révolution aplati aux pôles. Pour l'altitude on introduit une surface complémentaire, appelée géoïde, qui intervient dans l'expression des altitudes.

Un point sur la surface de la Terre est défini par des coordonnées géographiques, la latitude, la longitude qui sont des valeurs angulaires et par son altitude. Un point sur la surface de la Terre peut aussi être défini par des coordonnées cartésiennes sous réserve de définir un repère cartésien.

Les coordonnées de déplacement d'un objet à la surface de la Terre sont calculées à chaque instant dans le repère terrestre, cela implique des conversions entre le repère de l'objet et le repère terrestre.

Conventions d'écriture

Dans les algorithmes est utilisé l'opérateur ++ par définition , $valeur++$ signifie $valeur+=1$ ce qui signifie également $valeur = valeur + 1$.

Chapitre 3

Introduction

Dans ce document nous allons expliquer la démarche que nous avons suivi pour réaliser une partie d'un produit logiciel de calcul d'efficacité d'une munition.

Le logiciel quand il sera terminé devra permettre de simuler un scénario d'attaque d'une cible par un système d'armes. Un scénario est l'image informatique d'une situation réelle sur un théatre d'opération.

Nous imaginons un scénario : un ordre est donné à un hélicoptère tigre d'effectuer un appui aérien pour détruire une position ennemie qui a pris a parti une patrouille. L'hélicoptère est équipé de lance-roquettes chargés de roquettes AMV. La patrouille pris a parti informe de la présence d'une batterie de tir pouvant être une menace pour l'hélicoptère.

Quand le logiciel sera complet il devra répondre , avec un intervale de confiance, aux questions suivantes :

- Dans combien de temps l'hélicoptère arrivera sur zone ?
- Combien devra-t-il tirer de roquettes pour neutraliser la cible ?
- A qu'elle distance devra-t-il tirer pour atteindre la cible tout en se préservant de la menace annoncée ?

A partir de ce scenario en utilisant une méthode inspirée de [R95] nous allons concevoir les grandes étapes de réalisation de la simulation. Les ressources pour le développement d'un tel logiciel sont bien supérieures à 9mois/Homme. Le projet décrit dans ce document développera les étapes de réalisation de maquette nécessaires à l'élaboration des fonctions et structures qui seront intégrées dans le programme définitif.

Dans ce document nous montrons comment on utilise une approche probabiliste pour realiser une étude de l'efficacité d'une salve de roquette.

L'approche statistique utilise certains résultats obtenus soient par calcul avec une approche déterministe soient par réalisation d'essais. Elle utilise surtout les amplitudes des variations des résultats et la répétition d'un même événement un grand nombre de fois. Le matériel utilisé pour la majorité des calculs est un ordi-

nateur équipé de 1 Go de mémoire et d'un processeur.

Il n'existe pas de logiciel dans le commerce qui effectue ce type de calcul. Cela oblige le concepteur du modèle à utiliser un langage de programmation pour les réaliser.

La première approche calcule le résultat le plus précisément possible en s'appuyant sur des modèles restituant au mieux la réalité, la seconde estime le résultat en calculant la probabilité qu'ils se produisent.

Le chapitre 5 regroupe les méthodes et outils nécessaires à la réalisation de modèles d'efficacité selon des scénarii d'emplois opérationnels nécessaires à la réalisation de ce projet.

Utilisant des méthodes faisant appel à des générateurs de nombres aléatoires nous avons listé les différents algortihmes utilisés pour la génération de ces nombres et nous avons voulu savoir si le générateur de la bibliothèque standard du langage C suivait une loi uniforme. Le détail de ces récherches est en annexe.

Puis dans le chapitre 6 et 7 nous utiliserons ces méthodes pour la réalisation de maquettes de calcul Nous décrirons les différentes étapes de développement du projet : de la fiabilisation du besoin aux tests de validation.

Dans le chapitre 8 nous commenterons les principales variations de résultats obtenus durant la réalisation du projet et dans le chapitre 9 nous listerons le reste à faire et proposons quelques idées d'évolution du programme.

Chapitre 4

Présentation du projet

Ce chapitre décrit la solution logiciel envisagée pour le calcul de simulations d'efficacité des munitions associées à un système d'armes TDA. Après avoir définit un certain nombre de termes utilisés dans le document nous décrirons un cas réel que nous utiliserons pour généraliser la structure du logiciel de simulation. Nous terminerons avec une présentation de la conception générale du produit logiciel.

ce document de présentation générale doit servir de point de départ pour la rédaction des exigences du produit logiciel livrable. Il décrit les grandes étapes du développement et le planning de réalisation.

4.0.1 Notations

- géographie : représentation de la topographie du terrain, géodésie,
- météo : éléments naturels pris en compte pour les calculs,
- position : coordonnées des éléments définis dans le scénario,
- environnement : regroupe les informations géographiques, météorologiques, de positions,
- neutralisation : temps pendant lequel la cible est inopérante,
- cible : Ensemble d'informations propres à la cible, nécessaires aux calculs de neutralisation,
- objectif : regroupe les informations sur les cibles et leur temps respectif de neutralisation,
- système de détection : informe sur les déplacements de la ou des cibles,
- système de pointage : informations de réglage de l'arme ou du lanceur,
- munition : ensemble d'informations sur la nature de la munition,
- porteur : définit l'élément de transport de l'arme ou du lanceur,
- système d'arme : regroupe les informations sur le porteur, la munition, le système de pointage,
- efficacité : rapport entre la neutralisation obtenue et la neutralisation sou-

haité,
- scénario : regroupe les informations sur le système d'arme, l'objectif, l'environnement.

4.0.2 exemples de scénario

.scénario 1

On donne l'ordre à deux hélicoptères *tigre*, durant un vol d'observation, d'immobiliser définitivement un véhicule léger (A3F) équipé d'une mitrailleuse lourde de 12,7 mm avec 3 soldats à l'intérieur circulant sur une route dans une zone montagneuse. Les informations sur le mouvement du véhicule sont données par un observateur durant 5 minutes. Les hélicoptères sont armés de lance-roquettes contenant des roquettes de type AMV.

Questions

1. Lequel des deux hélicoptères arrivera le premier sur la zone ?
2. Combien devra-t-il tirer de roquettes pour neutraliser la cible ?
3. Qu'elles sont les conditions idéales de tir pour rester à l'abri d'un tir de riposte ?

.scénario 2

On donne l'ordre à un 2R2M de se mettre en batterie pour un tir sur un convoi de transport de munition par train. On veut arréter le train. La train se déplace vers le sud a une vitesse de 50 km/h.

Question

1. Combien d'obus devront-ils être tirés pour immobiliser le train ?

4.1 Vision logiciel

Si l'on regarde les scénarii sous un angle conceptuel nous pouvons extraire trois grands éléments (voir la figure 4.1) :
- un système d'arme composé d'un porteur, de munitions, d'un système de pointage
- un objectif composé de cibles et d'un niveau de neutralisation
- un environnement contenant la météo, les positions du système d'armes, des cibles et la géographie du théâtre d'opérations.

sous un angle fonctionnel nous allons décrire les différentes étapes de la réalisation d'un scénario, ainsi que son aspect dynamique.

Le scénario est constitué de 5 étapes :

1. Définition du scénario (initialisation)
   La première étape va être d'initialiser les éléments constituant le scénario ; le type de porteur, les munitions qu'il possède, le genre des cibles, leur niveau de neutralisation, la topographie du terrain, les éléments météo, leurs positions dans un repère géodésique global. L'ensemble de ces paramètres sont nécessaires aux différentes étapes de calcul ; calcul de la trajectoire des munitions, de l'effet terminal et de l'efficacité.
2. Trajectoire du porteur (calcul des conditions de tir initiales des munitions)
   La deuxième étape est le calcul des conditions de tir des salves de munitions. Elle contient les coordonnées des cibles, le calcul de la trajectoire du porteur, l'élaboration de la stratégie de tir, le séquencement des tirs, l'instant et les coordonnées des points d'origine des tirs de munition, un calcul d'optimisation de la stratégie de tir, du séquencement du tir.
3. Trajectoire Munition (calcul de la trajectoire)
   La troisième étape est le calcul des trajectoires des munitions. Les informations météo, géographiques et propre à la nature de la munition sont prisent en compte dans le calcul. Le résultat du calcul contient les instants et les coordonnées, de chaque point de la trajectoire et du point de mise à feu de l'effet terminal.
4. Effet Terminal (calcul la dynamique l'effet terminal de la munition)
   La quatrième étape est le calcul des effets dynamiques de la munition, la trajectoire des éclats, leur vitesse, leur énergie. Elle retourne le volume d'efficacité de l'effet terminal et la densité d'éclats par unité de surface.
5. Analyse scénario (calcul de l'efficacité)
   La cinquième et dernière étape est l'analyse, des effets des munitions sur les cibles en tenant compte de leurs caractéristiques propres. Durant cette étape il est calculé la surface d 'exposition de chaque cible en tenant compte de la dispersion de chaque munition, le niveau de neutralisation, l'efficacité globale du scénario, la création des représentations graphiques des résultats sous forme de planches ou d'animations 3D.

4.1.1 Ebauche d'une solution technique

Etat de l'existant

Actuellement il existe plusieurs programmes pour réaliser des simulations. Certain sont écrits en Fortran et fonctionne sous VAX en réseau ou après adaptation

et compilation fonctionne sous Windows XP. Il existe aussi plusieurs logiciels qui remplissent certaines des étapes de calcul d'efficacité pour une munition ou un groupe de munitions, d'autres calculent les trajectoires...etc Le patrimoine logiciel est important et devrait être une source importante de fonctions qui après examen pourraient être réutilisables .

Choix logiciel

Nous avons imaginé une solution logiciel pour répondre à la demande. Elle est fermée, très complètes, d'utilisation aisée mais pour laquelle les évolutions obligeront la création de nouveaux modules à enficher sur la structure du logiciel à l'image d'un bus de périphériques d'extensions. Les fonctions existantes souvent écrites dans des langages différents devront être réécrites. Les différents programmes existant ne seront pas utilisables en l'état des passerelles de communication seront à construire si leur temps de réalisation n'est pas trop long.

Choix langage :

Nous proposons afin de facilité le développement l'utilisation du langage JAVA et son extension SWING pour la création d'interface.

Stockage des données :

Une base de données est un stock d'informations organisé et structuré de manière à pouvoir être facilement manipulé. Nous souhaitons que le logiciel expert dans le calcul d'efficacité, est un domaine d'application le plus vaste possible. Par conséquent, il nous semble pertinent que l'utilisateur bénéficie d'une base données stockant les informations sur les systèmes d'armes et les cibles. L'utilisateur aura la possibilité de compléter la base avec de nouveaux éléments. La base de données sera constituée des tables : cibles, porteurs, munitions dont les champs seront définis lors de l'étape de conception. Les tables seront interrogées par les différentes étapes du logiciel. Les résultats pourront être eux aussi stockés afin de disposer d'un historique de résultats d'étude d'efficacité déjà effectuées.

Nous proposons l'utilisation d'une base de donnée le modèle Express 11g d'oracle (gratuit).

Interface :

Nous souhaitons proposer à l'utilisateur une interface graphique conviviale, lui permettant d'initialiser ou modifier des variables à travers des boites de dialogue.

L'interface lui permettra de définir le scénario en sélectionnant des éléments présents dans la base de donnée, choisir les options du logiciel.

Les étapes du logiciel sont indépendantes, pour avoir la possibilité rejouer le scénario dans sa totalité ou partiellement. Cela implique qu'entre chaque étape du scénario, une interface contient les données initiales de l'étape suivante. Par

exemple, l'utilisateur pourra analyser la variation de l'efficacité en modifiant un ou plusieurs paramètres d'entrée d'une étape choisie sans avoir besoin des précédentes.

Visualisation :
Les cibles pourront être disposées dans un environnement en 3D. Pour facilité l'étude du scénario, l'utilisateur pourra visualiser une animation du scénario. Les résultats seront données sous forme de statistique et de graphique.

Grandes étapes du développement :
- création des interfaces,
- création du schéma de la base,
- création des traitements sur la base,
- algorithmes,
  - trajectoire porteur,
  - trajectoire munition,
  - effet terminal,
  - analyse scénario.

Estimation en nombre de lignes :

création interfaces	500 × 4	2000
création du schéma de la base	300 × 1	300
création des traitements sur la base	700 × 1	700
algorithmes		
trajectoire porteurs	1500 × 2	3000
trajectoire munitions	500 × 2	1000
effet terminal	500 × 5	2500
analyse scénario	500 × 1	500
total		10000

TABLE 4.1 – estimation du nombre de lignes pour le choix 1

Estimation de l'effort, du temps de développement, des ressources avec la méthode COCOMO.

Le programme est d'une taille inférieure à 50 klignes, l'estimation du nombre de lignes est dans le tableau 4.1. Le logiciel est dans un domaine familier, dans un environnement connu et stable, réalisé par une petite équipe dans un domaine connu du début jusqu'à la fin du projet. Les résultats de l'estimation sont dans le tableau 4.2.

Estimation nominale	
Effort	36 Mois x Homme
Temps de développement	10 mois
Ressource	3 hommes
Estimation pessimiste +75%	
Effort	62 Mois x Homme
Temps de développement	12 mois
Ressource	5 hommes
Estimation obtimiste -25%	
Effort	25 Mois x Homme
Temps de développement	9 mois
Ressource	2 hommes

TABLE 4.2 – calcul des estimations

Ce premier choix est à l'image d'un bus de communication sur lequel on raccorde des composants pour réaliser un scénario. Mais il est trop coûteux et trop complexe pour être réalisable dans l'immédiat. Il est donc proposé de réaliser depuis l'existant une étude pour la mise en bibliothèque des fonctions offrant une plus grande possibilité de réutilisation. Cette bibliothèque sera utilisable dans des programmes en C. Les sources des fonctions pourront aussi être utilisées pour réaliser des fonctions pour les programmes scilab et matlab. L'idée finale est de créer des blocs ranger dans une « toolbox » pour les programmes scicos et simulink afin de réaliser des scénarii en mode graphique.

Nous verrons dans les chapitres suivant deux réalisations une première au chapitre CHAP 6, qui est la réalisation de la maquette d'un sous-ensemble du module analyse scénario écrit en langage scilab qui pourrait à terme être appelée dans un bloc scicos, puis la seconde qui est une maquette d'un programme regroupant un ensemble de fonctions que l'on a voulu réutilisables pour essayer de définir au mieux les périmètres de construction des bibliothèques.

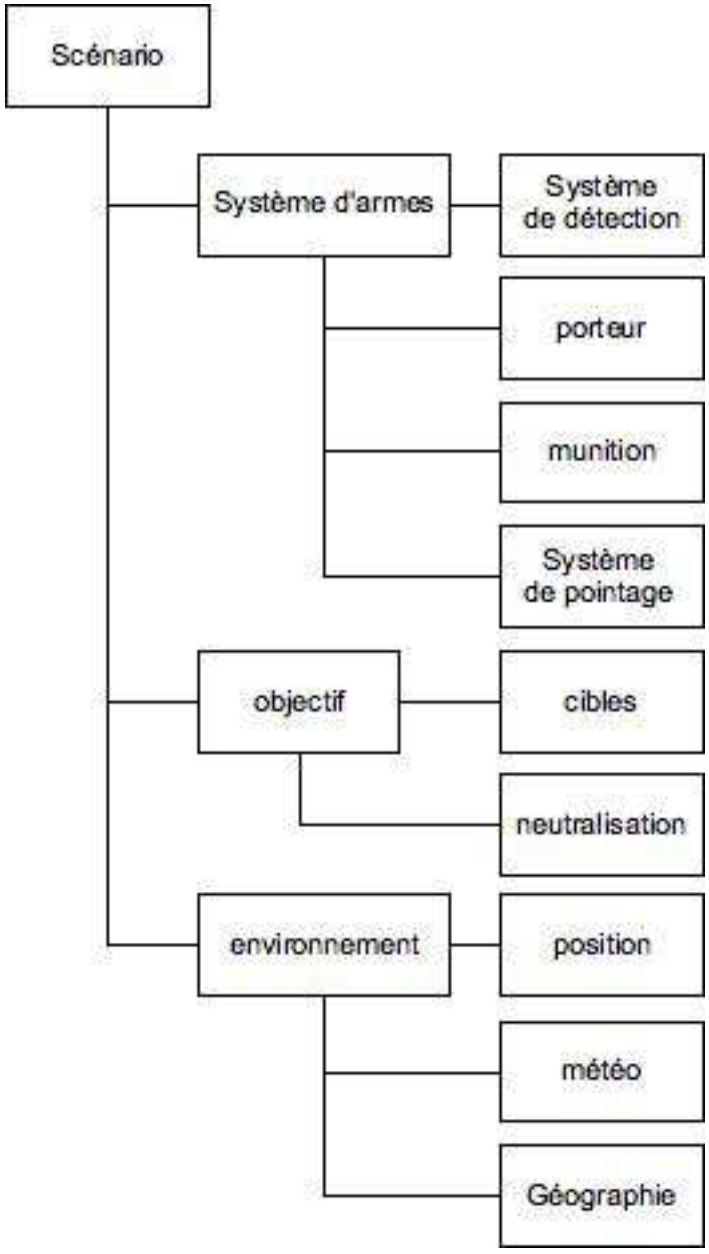

FIGURE 4.1 – organisation des principaux composants

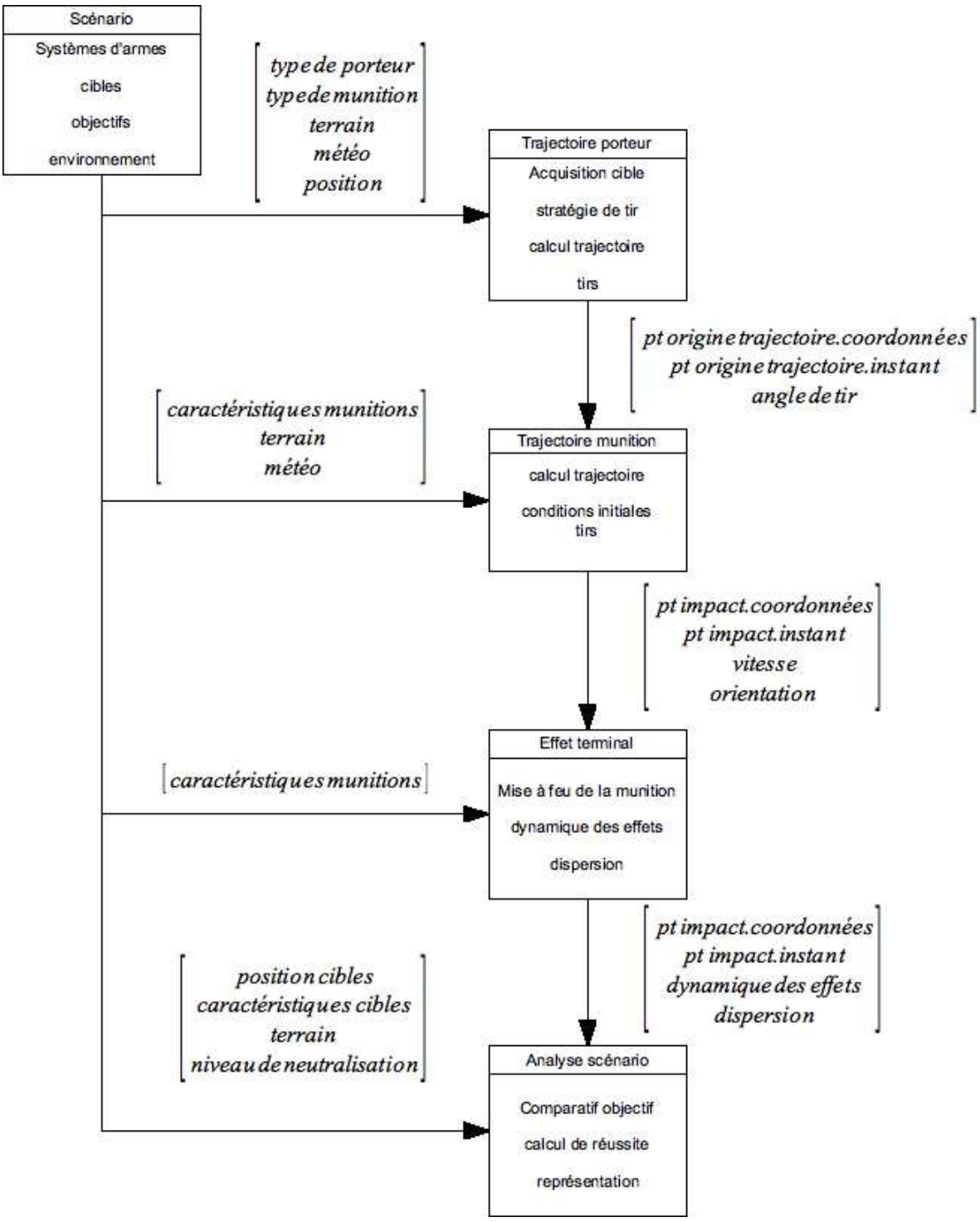

FIGURE 4.2 – passage des principaux arguments entre les composants

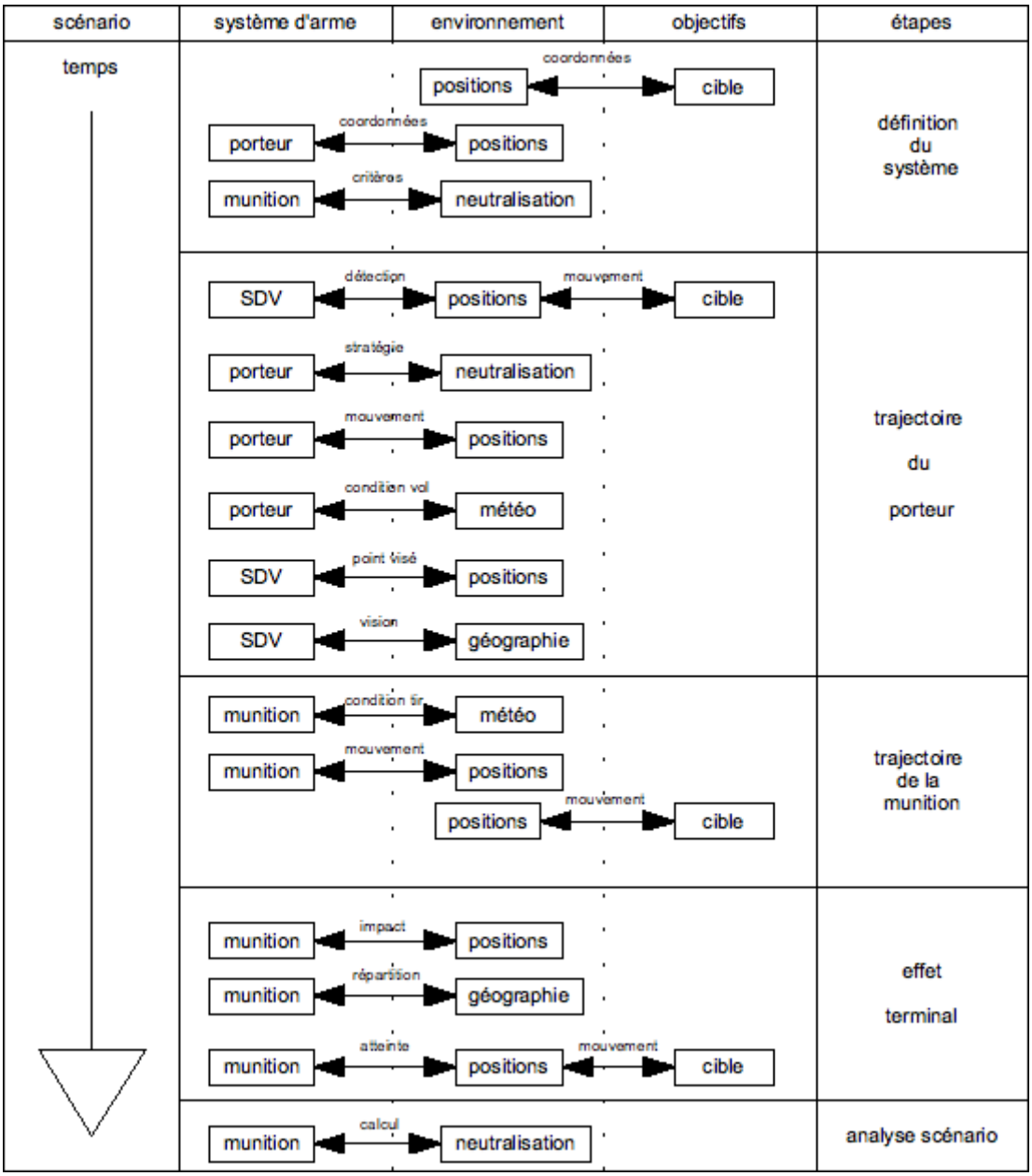

FIGURE 4.3 – schéma des principales communications durant un scénario

Chapitre 5

Méthodes et Outils

Dans ce chapitre nous allons parcourir les méthodes et outils qui ont été nécessaires pour la réalisation de cette étude. Nous expliquerons tout d'abord la méthode de conception logiciel que nous avons suivi, puis ayant remarqué, lors des premières recherches sur la méthode de monté-carlo, l'importance des variables aléatoires nous avons parcouru l'état de l'art dans ce domaine et recherché si le générateur de nombre pseudo-aléatoire *rand* de la bibliothèque standard du C suit une loi statistique, laquelle et convient-elle pour une utilisation dans un programme contenant une méthode de monté-carlo. Le détail de ces recherches sont en annexe. Le domaine métier de ce projet fait appel a des connaissances relevant de la science physique et de ce fait il nous a parut intéressant de présenter les différentes méthodes que nous avons utilisées durant l'écriture des maquettes.

5.1 Méthode de conception

A la lecture de l'énoncé du projet plusieurs questions se posaient. Le contenu du projet est principalement d'ordre technique. Il se compose pour l'essentiel de fonctions de calcul de comportements d'éléments réels dans un référentiel terrestre et de ce fait s'appuie sur la physique. D'autre part ne pouvant définir totalement ce comportement il est fait appel aux probabilités pour tenter d'avoir des calculs au plus proche de la réalité observée lors d'essais.

Les éléments étudiés sont réels il est donc facile de concevoir, avec leurs caractéristiques propres et leurs utilités, des classes les définissant pour une utilisation dans une méthode objet. Cependant L'approche objet n'est pas utilisée par les mathématiciens et physiciens lors de leurs raisonnements ni lors de l'élaboration de leurs calculs et ils n'en maîtrisent pas les concepts. La méthode la plus proche de leur façon de faire est la méthode structuré. De plus lors de leurs études les chercheurs voient l'informatique comme un outil, leur permettant de réaliser des

calculs complexes, de validation d'une démonstration et souvent quand le résultat est obtenu le programme est oublié ou détruit. Ils réalisent des programmes à usage unique et jetable. Il fallait donc trouver une méthode de conception logiciel orienté objet, acceptant pour certains sous-ensembles des modèles structurés, et dont la méthode de progression soit itérative pour y associer la réalisation de petites maquettes jetables. Ces maquettes apportent le bénéfice de convertir une solution souvent écrite en langage mathématique en un langage informatique, que ce soit dans un tableur ou dans un logiciel de calcul tel que Scilab. Cette programmation oblige le concepteur de la maquette à comprendre le raisonnement mathématique et découvre ainsi les limites inhérentes a son implémentation informatique. Ces limites serviront à la création des tests unitaires lors de sa programmation sous forme de méthode ou fonction dans le module définitif. Nous avons trouvé une méthode décrite dans [R95] que nous avons décidé d'adapter pour la réalisation de notre projet.

5.1.1 Présentation

La méthode décrite par J.-P. Rosen [1] se veut orientée objet, associée à un processus itératif et sur la réalisation de maquettes. Elle est plus spécialement destinée à un projet développé en ADA. Mais les lignes directrices sont suffisamments générales pour être applicables à tous projets. Elle est recommandée par son auteur pour des projets de 10 000 à 100 000 lignes de codes ce qui représente dans les descriptifs de la méthode COCOMO un projet de compléxité et de taille moyenne réalisé par une équipe composée de débutants, de personnes expérimentées n'ayant pas tous le même domaine d'expérience et où les domaines d'application du projet sont maîtrisés. Elle tente de promouvoir la réutilisation des composants logiciels existants.

Elle se décompose en trois phases , une phase d'initialisation dans laquelle on structure le projet, une phase de conception itérative de chaque bloc identifié dans la phase précédente , et une dernière de conclusion.

les principes de la méthode sont les suivants ; On décompose le problème par niveau sémantiques décroissant. Chaque étape est validée par des techniques de maquettage. La méthode est appliquée a chaque niveau sémantiques et s'arrête quand la description est complète.

Nous utiliserons cette méthode pour notre projet et nous discuterons des écarts que nous aurons consenti.

1. Ancien élève de l'ENST, professeur des universités, aujourd'hui directeur technique chez Adalog, société de formation et d'expertises dont il est le fondateur.

5.2 Variables aléatoires

Dans la plupart des phénomènes aléatoires, le résultat d'une épreuve peut se traduire par une grandeur mathématique, très souvent représentée par un nombre. La notion mathématique qui représente ce genre de situation est celle de variable aléatoire. Une variable aléatoire est caractérisée par l'ensemble des valeurs qu'elles peut prendre et par l'expression mathématique de la probabilité (loi ou distribution de probabilité) de ces valeurs.

Une variable aléatoire est associée à une loi de probabilité définie par une distribution de probabilité et une fonction de répartition qui correspond à la distribution des probabilités cumulées. Une loi de probabilité peut être caractérisée par certaine valeurs typiques correspondant aux notions de valeur centrale, de dispersion et de forme de distribution.

5.2.1 Générateurs de nombres pseudo-aléatoires

Les générateurs de nombres pseudo-aléatoires sont des algorithmes qui génèrent des séquences de nombres présentant certaines propriétés du hasard. Leurs utilisations sont multiples : sécurité informatique, simulation.

Les sorties d'un tel générateur ne sont pas entièrement aléatoires ; elles s'approchent seulement des propriétés idéales des sources complètement aléatoires.

L'utilisation de ces générateurs de nombres aléatoires est soumise à un biais inhérent aux méthodes employées : la périodicité. Pour une méthode tel que celle de Von Neuman (1946) nous relevons une périodicité très faible (inférieure à 50) ; par exemple avec le nombre 1234567892 sur 100 itérations on peut remarquer une périodicité [2] courte (comme le montre la figure FIG.5.1).

Il existe d'autres méthodes de génération de nombres aléatoires dont les algorithmes utilisent la congruence linéaire (Lehmer 1951), la suite de Fibonacci ou des registres à décalage. Cependant, après un certain nombre d'itérations, le générateur retrouvera le même état interne au moins deux fois et entrera obligatoirement dans un cycle. Avec les mêmes valeurs de départ un générateur produira toujours la même suite de nombre. Pour contourner cet obstacle théorique, le générateur peut commencer dans un état quelconque (la « graine »). Il produira toutefois la même suite si la graine reste identique. Pour améliorer la qualité des sorties, on peut introduire comme « graine » des composantes « aléatoires » provenant du système, comme par exemple le nombre de millisecondes depuis le lancement de l'ordinateur. La qualité de ces composantes est très importante surtout en sécurité informatique, on mesure leur robustesse en calculant leur entropie, ce qui revient à mesurer la capacité du générateur de graine à fournir des informations différentes.

2. intervalle entre les tirages d'une même valeur

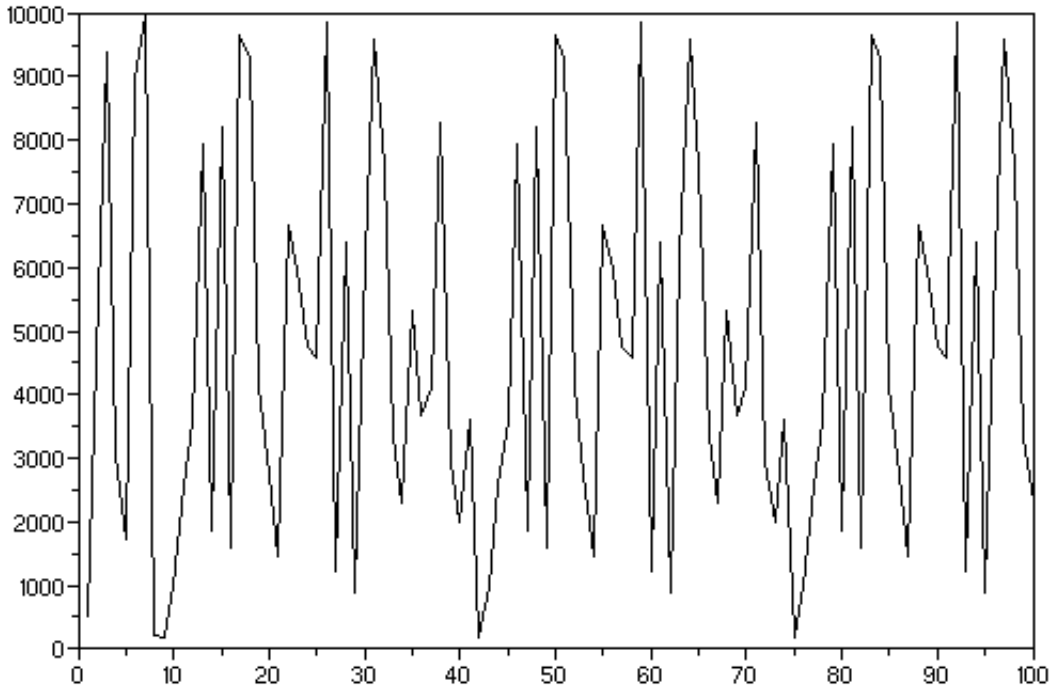

FIGURE 5.1 – représentation graphique de la génération de nombre aléatoire selon la méthode de Von Neuman

Aujourd'hui existent d'autres méthodes pour lesquelles la périodicité est bien plus grande : par exemple, le générateur Mersenne Twister (1997) dont l'algorithme est basé sur un type particulier de registre à décalage et tient son nom d'un nombre premier de Mersenne. Il existe au moins deux variantes majeures, la plus répandue étant MT 19937, utilisant le nombre premier de Mersenne 2^{19937} et dont la période est de $2^{19937} - 1$.

Il existe des tests statistiques pour évaluer les générateurs de nombres pseudo-aléatoires. Dans certains tests d'uniformité[C76] (DELTOUR 1967 et LINDER 1970) sont regardées l'homogénéité de la distribution, l'homogénéité de la distribution des chiffres dans les nombres, l'absence de corrélation entre les nombres et les coefficients de corrélation.

Il existe aussi les tests de Diehard[M93] mis au point par le professeur George Marsaglia [3] qui sont une série de 15 tests. Il s'agit de tests basés sur des manipula-

3. George Marsaglia est un mathématicien et informaticien. Il est connu pour le développe-

tions de bits ou de matrices obtenues avec l'échantillon. Ces tests retournent une valeur qui ne doit ni être 0 ou 1, ni trop proche de ces valeurs. Le test est considéré réussi si le résultat est compris entre 0,025 et 0,975. [4].

5.3 La loi normale

La loi normale ou encore appelée loi de Gauss, ou de Laplace-Gauss est le modèle probabiliste le plus utilisé pour décrire de très nombreux phénomènes observés, elle est aussi la loi de probabilité la plus utilisée en statistique.

Elle est caractérisée par les valeurs :

– μ : la moyenne arithmétique de la population Ω

– σ : écart-type associé à la moyenne

La fonction de répartition de la loi normale n'a pas de forme analytique et doit donc être recalculée pour chaque couple de valeurs (μ, σ). Afin d'éviter de refaire ces calculs il a été défini ce que l'on appelle « la loi normale centrée réduite », que l'on note : $N(0, 1)$. C'est une loi normale qui a pour paramètres $\mu = 0$ et $\sigma = 1$

5.4 Méthode de Monte-Carlo

La méthode de Monte-Carlo est une méthode numérique, qui utilise des tirages aléatoires pour réaliser le calcul d'une quantité déterministe. La méthode de simulation de Monte-Carlo permet aussi d'introduire une approche statistique. Elle consiste à isoler un certain nombre de variables-clefs du projet et à leur affecter une distribution de probabilités. Pour chacun de ces facteurs, on effectue un grand nombre de tirages aléatoires dans les distributions de probabilité déterminées précédemment, afin de calculer la probabilité d'occurrence de chacun des résultats.

L'utilisation de cette méthode implique un grand nombre de tirages. Les représentations graphiques des figures FIG.5.2 nous montrent le comportement du résultat d'un calcul par la méthode de Monte-Carlo en fonction du nombre de tirages effectués. Nous avons isolé une variable-clef et nous avons effectué le test .

Dans la figure FIG.5.2 nous pouvons voir 4 zooms de la courbe des résultats en fonction du nombre d'itérations pour une variable clef, d'un calcul selon la méthode de Monte-Carlo.

Nous pouvons remarquer que pour un petit nombre de tirage inférieur à 500 le résultat est très variable. Pour un résultat avec un niveau de précision de l'ordre

ment de certaines des méthodes les plus couramment utilisés pour générer des nombres aléatoires. Il est professeur émérite de mathématiques pures et appliquées.

4. Ces tests sont cités à titre d'exemple mais ne seront pas utilisés dans le cadre de ce projet.

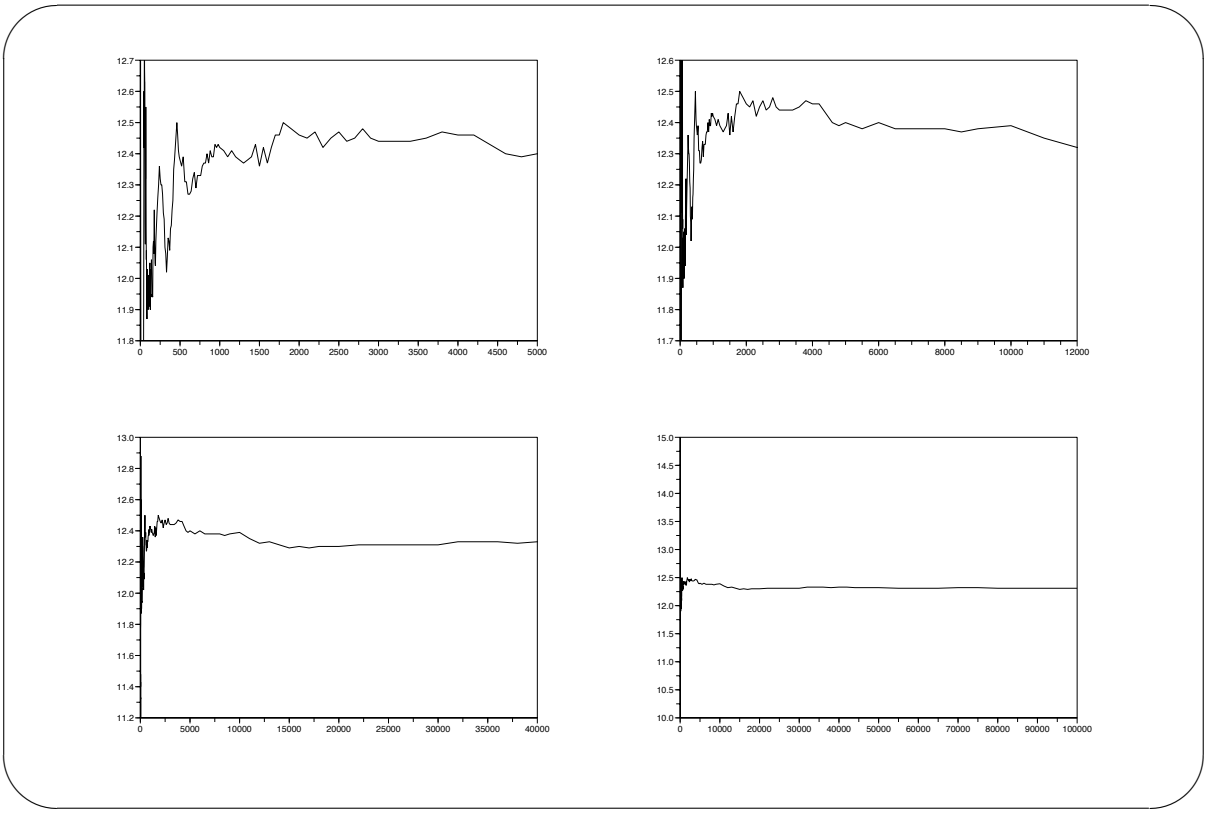

FIGURE 5.2 – comportement du résultat d'un calcul pour $\sigma = 0,0015$

de l'unité un nombre de tirages de l'ordre de 20000 itérations paraît suffisant. Nous retenons donc cette valeur pour nos calculs ultérieurs.

5.5 Méthode de calcul de la densité probable d'impacts par unité de surface .

Cette méthode a été écrite par Mr Fraissigné afin de calculer la densité probable de dards, portés par une ou plusieurs roquettes tirées depuis un avion, par unité de surface. Le calcul s'effectue à l'origine dans une page d'un tableur.

Je l'ai reécrite dans une fonction Scilab qui prend comme arguments :

- le nombre de roquettes N_r
- le nombre de dards par roquette
- le rayon de la gerbe de dards
- l'écart type de dispersion selon les axes x et z de la roquette
- la justesse du tir selon les axes x et z, mx, mz, représentant l'écart de distances entre le point visé et le point atteint.

– l'aire de la cible
– le nombre de dards nécessaires pour neutraliser la cible *nbredarkil*.

Elle retourne un vecteur contenant :
– la probabilité de succès de la salve
– la longueur du demi-côté de la gerbe carrée équivalente
– le nombre de dard pour neutraliser la cible
– la densité de dards dans la gerbe
– le nombre minimum de roquettes pour neutraliser la cible

5.5.1 Hypothèses de simplification

1. On suppose la cible fixe.
2. On suppose la répartition des dards dans la gerbe parfaitement homogène
3. On suppose que la densité de dards est constante à l'intérieur de la gerbe et nulle à l'extérieur.
4. On suppose que la gerbe est un carré dont la longueur d'un côté est égal à $2L$ tel que(pour une surface équivalente) :

$$2L = \sqrt{\pi} \cdot R$$

5. Les imprécisions verticales et horizontales sur x et z sont supposées gaussiennes. L'imprécision selon l'axe y est considérée comme négligeable.
6. Les valeurs de justesse sont données.

5.5.2 Description de la fonction

La probabilité qu'une roquette passe dans un carré centré sur la cible de côté $2L$ implique

$$-L \leq x \leq L$$
$$-L \leq z \leq L$$

5.5.3 Première estimation

On calcule le demi coté de la gerbe

$$L = \frac{\sqrt{\pi} \cdot rgerb}{2}$$

En fixant le nombre de dards pour neutraliser la cible ainsi que l'aire de la cible on calcule la densité de dards nécessaire par mètre carré pour neutraliser la cible.

$$ddkm = \frac{nbredarkil}{surfcib}$$

on calcule la densité de dards dans la gerbe par mètre carré.

$$ddm = \frac{nbredar}{4L^2}$$

En effectuant le rapport des densités de dards par mètre carré pour neutraliser une cible et de dards dans une gerbe on obtient le nombre minimum de roquettes nécessaire pour neutraliser la cible.

$$nbrm = \frac{ddkm}{ddm}$$

En tenant compte de la justesse du tir et de la dispersion de la roquette on obtient :

$$\frac{-L - mx}{\sigma} \leq \frac{x - mx}{\sigma} \leq \frac{L - mx}{\sigma}$$
$$\frac{-L - mz}{\sigma} \leq \frac{z - mz}{\sigma} \leq \frac{L - mz}{\sigma}$$

On a donc la probabilité qu'une roquette passe dans le carré de coté $2L$ est :

$$p = \left[F\left(\frac{L - mx}{\sigma}\right) - F\left(\frac{-L - mx}{\sigma}\right)\right] \times \left[F\left(\frac{L - mz}{\sigma}\right) - F\left(\frac{-L - mz}{\sigma}\right)\right]$$

avec F la fonction de densité de probabilité de la loi normale

Pour N_r roquettes tirées, la probabilité que k roquettes passent dans le carré de coté $2L$ centré sur la cible est

$$P_k = \frac{N_r!}{k! \cdot (N_r - k)!} \cdot P^k \cdot (1 - P)^{N_r - k}$$

La probabilité totale d'avoir une ou des roquettes touchant la cible est :

$$P_{total} = \sum_{k=1}^{k=N_r} P_k$$

5.6 Méthode de calcul de trajectoire d'un mobile non propulsé

Ce programme est extrait d'un calcul écrit par Mr Fraissigné dans une feuille de tableur. Il calcule pour chaque instant les coordonnées des points de la trajectoire d'une bille en tenant compte du coefficient de pénétration dans l'air de la bille et du coefficient de frottement de l'air.

hypothèses

- on utilise le trièdre aéronautique (figure **??**).
- la vitesse de la bille à l'origine de la trajectoire n'est pas nulle.

notations

- V_o vitesse de la bille à l'origine de la trajectoire en mètre par seconde.
- θ_o angle de la pente de la bille à l'origine de la trajectoire en radian.
- D diamètre de la bille en mètre
- S surface en contact avec l'air en m^2.
- c_x coefficient de pénétration dans l'air.
- ρ coefficient de résistance de l'air.
- M la masse de la bille en gramme
- δ_t pas de temps pour le calcul en seconde
- t instant pour chaque point de la trajectoire en seconde.
- $\frac{dV_x}{dt}$ projection de l'accélération de la bille sur l'axe x pour chaque instant de la trajectoire.
- $\frac{dV_z}{dt}$ projection de l'accélération de la bille sur l'axe z pour chaque instant de la trajectoire.
- $V_x(t)$ projection sur l'axe x du vecteur vitesse pour chaque instant de la trajectoire.
- $V_z(t)$ projection sur l'axe z du vecteur vitesse pour chaque instant de la trajectoire.
- $\vec{V}(t)$ vecteur vitesse à chaque instant de la trajectoire.
- $\theta(t)$ pente pour chaque instant de la trajectoire.
- $P_x(t)$ coordonnée de la bille en x à l'instant t.
- $P_z(t)$ coordonnée de la bille en z à l'instant t.

$\rho = 1.23$
$S = \pi \cdot \frac{D^2}{4}$

$$V_x(t) = \begin{cases} V_o \cdot \cos(\theta_o) & \text{si } t = 0 \\ V_x(t-1) + \frac{dV_x}{dt} \cdot \delta_t & \text{sinon} \end{cases}$$

$$V_z(t) = \begin{cases} V_o \cdot \sin(\theta_o) & \text{si } t = 0 \\ V_z(t-1) + \frac{dV_z}{dt} \cdot \delta_t & \text{sinon} \end{cases}$$

$$\theta(t) = \begin{cases} \theta_o & \text{si } t = 0 \\ \arctan(\frac{V_z}{V_x}) & \text{sinon} \end{cases}$$

$$V(t) = \begin{cases} V_o & \text{si } t = 0 \\ \sqrt{V_x^2 + V_z^2} & \text{sinon} \end{cases}$$

$$P_x(t) = \begin{cases} P_{xorig} & \text{si } t = 0 \\ P_x(t-1) + V_x(t) \cdot \delta_t & \text{sinon} \end{cases}$$

$$P_z(t) = \begin{cases} P_{zorig} & \text{si } t = 0 \\ P_z(t-1) + V_z(t) \cdot \delta_t & \text{sinon} \end{cases}$$

$$\frac{dV_x}{dt} = \begin{cases} \frac{1}{M} \cdot (-.5) \cdot \rho \cdot V_o^2 \cdot s \cdot Cx \cdot \cos(\theta_o) & \text{si } t = 0 \\ \frac{1}{M} \cdot (-.5) \cdot \rho \cdot V(t)^2 \cdot s \cdot Cx \cdot \cos(\theta(t)) & \text{sinon} \end{cases}$$

$$\frac{dV_z}{dt} = \begin{cases} \frac{1}{M} \cdot (-.5) \cdot \rho \cdot V_o^2 \cdot s \cdot Cx \cdot \sin(\theta_o) - M \cdot 9.81) & \text{si } t = 0 \\ \frac{1}{M} \cdot (-.5) \cdot \rho \cdot V(t)^2 \cdot s \cdot Cx \cdot \sin(\theta(t)) - M \cdot 9.81) & \text{sinon} \end{cases}$$

5.6.1 Spécialisation pour le calcul de trajectoire de dards

ce programme permet de calculer la trajectoire d'un dard en tenant compte de son cx pour chaque pas de calcul. Il se compose de deux parties ; le calcul du cx du dard en fonction de la vitesse et la fonction de calcul de la trajectoire du dard .

Pour le calcul de la trajectoire nous utilisons l'algorithme de calcul de la trajectoire d'une bille, dans lequel nous définissons la masse, le poids du dard comme constant, et nous calculons le cx du dard pour chaque pas de la trajectoire en fonction de sa vitesse.

calcul du cx

cette fonction est extraite du programme contenu dans le document [B18]

notation

v vitesse du dard a l'instant de l'éjection.
var indice calculé en fonction de la vitesse.
tcx tableau des valeurs discrètes du Cx du dard.

$$tcx = \{0.1019, 0.1017, 0.1027, 0.1560, 0.1982, 0.1875, 0.1719, 0.1592\}$$

$$var = floor(v/100)$$

$$Cx = \begin{cases} 0.1592 & \text{si } v > 800 \\ tcx(var-1) & \\ +(tcx(var) - tcx(var-1)) & \text{sinon} \\ *(v - var * 100)/100 & \end{cases}$$

5.7 Méthode de calcul de la répartition des dards sur une surface

Dans cette section est décrit la démarche d'étude pour la réalisation d'un modèle de calcul de la répartition des dards d'une salve de roquettes 68mm AMV sur une cible en mouvement (Vedette rapide). On souhaite que le temps de calcul soit court. Ne pouvant calculer les dispersions angulaires en sortie de lanceur des roquettes elles seront réalisées par tirage aléatoire. Pour la même raison les

dispersions sur le temps de dépotage seront-elles aussi réalisées par tirage aléatoire. L'avion se déplace selon un mouvement rectiligne uniforme.Le calcul de la trajectoire de référence pour la roquette sera effectuée avec un logiciel de calcul de trajectoire balistique pour une munition propulsée. Il comprend principalement un modèle à 6 degrés de liberté et un modèle aérodynamique de la munition. Nous obtiendrons les points de toutes les courbes des roquettes de la salve par translation et rotation des points de la courbe de référence. L'aspect dynamique de la cible nous amène à gérer le déplacement de la cible simultanément aux trajectoires des roquettes de la salve et des dards. Un dard touche la cible si son point d'impact à la surface de l'eau se situe dans l'ombre projetée du navire (projection du navire selon la direction azimut des dards). L'évaluation des conséquences des impacts des dards sur la cible n'est pas traitée par ce calcul. Un calcul élémentaire fournit le nombre de dards impactant la cible pour une salve de roquettes. Pour obtenir des résultats de calculs au plus proche de la réalité on utilisera la méthode de Monte-Carlo.

L'algorithme prend comme argument

1. les informations non modifiables durant l'utilisation :
   - le sigma de dispersion de la roquette (écart-type)
   - le nom du fichier contenant la trajectoire de référence des dards
   - le nom du fichier contenant la trajectoire de référence de la roquette
   - le nombre de salves calculées par le programme
2. les variables modifiables à chaque demande de calcul :
   - les valeurs de justesse du tir
   - la taille de la zone d'impact observée pour le calcul de densité
   - le nombre de paires de roquettes tirées dans chaque salve
   - la distance de dépotage
   - la longueur et la largeur du bateau
   - la vitesse du bateau
   - le nombre de dards nécessaires pour couler le bateau

L'algorithme affiche :

1. A la suite de la boucle de calcul du point d'impact de la roquette centrale de la salve :
   - L'instant d'impact de la roquette centrale dans l'eau
   - Le retard de dépotage
   - La distance avion cible selon l'axe Ox (en mètres)
   - L'angle de site de la cible sous axe avion (en degrés)
   - La vitesse des dards à l'impact (en m/s)
   - La pente du dard à l'impact (en degrés)
2. A la fin du calcul

- nombre moyen de dards qui ont touché le bateau par roquette
- nombre moyen de dards qui ont touché le bateau par salve
- Le seuil de performance ; nombre de dards nécessaires pour détruire la cible
- Proportion de salves satisfaisant la performance
- La proportion de dards tombés dans la zone définie pour le calcul de densité
- La proportion de dards tombés hors de la zone définie pour le calcul de densité.
- Le nombre total de dards tombés

Il enregistre dans un fichier le rapport du nombre de dards tombés par unité de surface sur le nombre de salves ; soit la moyenne des impacts de dards par unité de surface et par salve.

5.7.1 Calcul de la trajectoire d'une roquette

Cet algorithme sera utilisé pour toutes les roquettes de la salve. Il sera adapté selon le besoin pour répondre aux spécificités de calcul.

Etablir le point et l'instant de départ de la roquette en fonction de la position de l'avion.

Hypothèses :

- On suppose l'absence de changement d'attitude de l'avion pendant le tir de la salve. L'avion suit une trajectoire rectiligne uniforme.

Notations :

- nr nombre de roquettes dans la salve
- i indice de chaque roquette dans la salve ($1 \leq \mathrm{i} \leq \mathrm{nr}$)
- cd cadence de tir en roquette par seconde
- $P_o[i]$ position du point de tir de la roquette (coordonnées en mètres)

$$P_o[i] = \begin{bmatrix} x_o[i] \\ y_o[i] \\ z_o[i] \end{bmatrix}$$

- $T_o[i]$ instant de tir de la roquette (en seconde)
- θ_o pente de l'avion (en radians)
- $\vec{V_o}$ vecteur vitesse avion (coordonnées en m/s)

$$\vec{V_o} = \begin{bmatrix} v_o \cdot \cos\theta_o \\ 0 \\ v_o \cdot \sin\theta_o \end{bmatrix}$$

– P_{av} position de l'avion à t= 0 (coordonnées en mètres)

$$P_{av} = \begin{bmatrix} x_{av} \\ y_{av} \\ z_{av} \end{bmatrix}$$

$$\text{Pour } 1 \leq i \leq nr$$
$$T_o[i] = (i-1) \cdot \frac{1}{cd}$$
$$P_o[i] = P_{av} + T_o[i] \cdot \vec{V_o}$$

$$\text{d'où} \quad \begin{cases} x_o[i] = x_{av} + (i-1) \cdot \frac{1}{cd} \cdot v_o \cdot \cos\theta_o \\ y_o[i] = 0 \\ z_o[i] = y_{av} + (i-1) \cdot \frac{1}{cd} \cdot v_o \cdot \sin\theta_o \end{cases}$$

Tirage aléatoire de l'angle de la roquette en sortie de tube.

On utilise une fonction existante alnorGGT écrite par Claude Gorget pour obtenir la valeur de l'angle. Cet angle représente la dispersion des trajectoires balistiques compte tenu des perturbations de tir et des perturbations en vol.

Hypothèses :

– On suppose que la répartition des angles de la roquette en sortie de tube est gaussienne.

Notation :

– sganroq : écart-type de la dispersion (en radians)
– moanroq : moyenne de l'échantillon (en radians)
– dan_{site} : valeur aléatoire de dispersion de l'angle de la roquette en site
– dan_{azim} : valeur aléatoire de dispersion de l'angle de la roquette en azimut
– dan_{site} = alnorGGT(moanroq, sganroq)
– dan_{azim} = alnorGGT(moanroq, sganroq)

Tirage aléatoire de la dispersion du temps de dépotage des dards

Détermination de l'écart type de la dispersion de l'instant de dépotage

hypothèses :

- on considère la différence de tolérance pour un temps donné comme gaussienne et égale à 4 sigma.

notations :

- t_1 instant de dépotage en seconde
- t_{1max} instant de dépotage maximum en seconde
- t_{1min} instant de dépotage minimum en seconde
- tp temps programmé sur la roquette en seconde
- tol pourcentage de tolérance en fonction du temps programmé
- σ ecart type de la dispersion de l'instant de dépotage

$$
\begin{aligned}
tol(tp) &= \begin{bmatrix} 2.5\% \\ 1.75\% \\ 1.00\% \end{bmatrix} \text{ si } \begin{bmatrix} 1.28 \leq tp \leq 1.65 \\ 1.65 < tp \leq 2.00 \\ 2.00 < tp \leq 30.0 \end{bmatrix} \\
t_1 &= tp^{\pm tol} + 0.0023^{\pm 0.012} + 0.003^{\pm 0.002} \\
t_{1max} &= tp + tp \cdot tol + 0.040 \\
t_{1min} &= tp - tp \cdot tol + 0.012 \\
4 \cdot \sigma &= t_{1max} - t_{1min} \\
4 \cdot \sigma &= tp + tp \cdot tol + 0.040 - (tp - tp \cdot tol + 0.012) \\
4 \cdot \sigma &= 2 \cdot tp \cdot tol + 0.028 \\
\sigma &= \frac{tp \cdot tol}{2} + 0.007
\end{aligned}
$$

calcul de l'instant de dépotage

notations :

- $tp[i]$ temps programmé sur la roquette en seconde.
- dp dispersion du temps de dépotage par calcul avec la fonction alnorGGT
- td temps de dépotage de la roquette en seconde
- $instdp[i]$ instant de dépotage

$$dp = \text{alnorGGT}(0, \sigma)$$
$$td = tp[i] + dp$$
$$instdp[i] = To[i] + td$$

calcul des points et des instants de la trajectoire de la roquette

Hypothèses :

- On suppose que les trajectoires des différentes roquettes sont suffisamment proches de la trajectoire de référence pour pouvoir se déduire de cette trajectoire de référence par translation et rotation.
- On suppose l'absence de variation de vitesse entre chaque trajectoire translatée.
- On suppose la dispersion des angles de sortie de tube suffisamment faibles pour pouvoir considérer le cosinus de la dispersion $\approx$ 1 et le sinus de la dispersion $\approx$ de la dispersion en radian. Ce qui nous permet d'utiliser un microrotateur(voir le paragraphe 5.9.5).

Notations :

- *nbl* nombre de points de la trajectoire de référence.
- j indice de chaque point de la trajectoire de référence ($1 \leq$ j $\leq$ nbl)
- P_{ref}[j] point de la trajectoire de référence (coordonnées en mètres)

$$P_{ref}[\text{j}] = \begin{bmatrix} x_{ref}[\text{j}] \\ y_{ref}[\text{j}] \\ z_{ref}[\text{j}] \end{bmatrix}$$

  (Par convention $P_{ref}[1] = P_{av}$)
- T_{ref}[j] temps de vol en chaque pointP_{ref}[j] de la trajectoire de référence en seconde (par convention l'instant de tir = 0)
- P_{roq}[j] point de la trajectoire roquette (coordonnées en mètre)

$$P_{roq}[\text{j}] = \begin{bmatrix} x_{roq}[\text{j}] \\ y_{roq}[\text{j}] \\ z_{roq}[\text{j}] \end{bmatrix}$$

- T_{roq}[j]instant (en seconde) correspondant à la position P_{roq}[j] de la roquette.
- $\vec{\theta'}$ microrotateur représentant la dispersion de la roquette en sortie de tube(coordonnées en radians)

Translation et rotation des points de la trajectoire de référence vers la nouvelle trajectoire roquette.

$$\text{pour } 1 \leq j \leq nbl \text{ et pour la roquette i}$$
$$\overrightarrow{OP_{roq}[j]} = \overrightarrow{OP_o[i]} + \overrightarrow{P_{ref}[1]P_{ref}[j]} - \overrightarrow{P_{ref}[1]P_{ref}[j]} \wedge \vec{\Theta'}$$
$$\overrightarrow{T_{roq}[j]} = \overrightarrow{T_{ref}[j]} + T_o[i]$$

récupération de l'instant du point d'impact de la roquette

Notations :

- $T_{imp}[i]$ instant d'impact de la roquette au sol (en seconde)
- $Z_{roq}[t]$ coordonnée en z de la roquette en fonction du temps

$$Z_{roq}[t] = interpolationlineaire(Z_{roq}[j], Z_{roq}[j-1], T_{roq}[j], T_{roq}[j-1], t)$$
$$\text{quand } Z_{roq}[t] = 0 \text{ on recherche } t = t_{impact}$$
$$t_{impact} = interpolationlineaire(T_{roq}[j], T_{roq}[j-1], Z_{roq}[j], Z_{roq}[j-1], 0)$$
$$T_{imp}[i] = t_{impact}$$

calcul des coordonnées du point de dépotage

Hypothèses :

- on suppose que l'instant correspondant à l'instant de dépotage moins le temps d'origine de la trajectoire roquette est supérieur au temps de dépotage.

Notations :

- pt_{impact} point d'impact de la roquette avec l'eau
- $pt_{trajroquette}$ point de la trajectoire de la roquette
- $pt_(avant)$ point de la trajectoire avant le point de dépotage
- $pt_(suivant)$ point de la trajectoire suivant le point de dépotage
- $inst_{avant}$ instant avant le temps de dépotage
- $inst_{suivant}$ instant suivant le temps de dépotage
- $distance_{courant}$ distance parcourue par la roquette sur sa trajectoire
- $distance$ distance restant à parcourir avant le point d'impact
- $distance_{depotage}$ distance de dépotage des dards

– Pt_{dp} point de dépotage des dards (coordonnées en mètres)

$$Pt_{dp} = \begin{bmatrix} x_{dp} \\ y_{dp} \\ z_{dp} \end{bmatrix}$$

tantque $distance \geq distance_{depotage}$ faire

$$pt_{avant} = pt_{courant}$$
$$pt_{trajroquette} + +$$
$$pt_{courant} = pt_{trajroquette}$$
$$distance_{avant} = distance$$
$$distance = pt_{impact} - distance_{courant}$$

fin tantque

$$x_{dp} = interpollineaire(px_{avant}, inst_{avant}, px_{courant}, inst_{courant}, td)$$
$$y_{dp} = interpollineaire(py_{avant}, inst_{avant}, py_{courant}, inst_{courant}, td)$$
$$z_{dp} = interpollineaire(pz_{avant}, inst_{avant}, pz_{courant}, inst_{courant}, td)$$

5.7.2 Calcul de la trajectoire des dards

A chaque point de la trajectoire de référence des dards on applique une rotation et une translation à l'aide d'un microrotateur pour obtenir les points de la trajectoire du dard courant. On applique le même algorithme que pour la trajectoire de la roquette.

5.7.3 Calcul de l'instant et des coordonnées du point d'impact des dards

Hypothèses :

– On suppose que le dard k impacte l'eau et que $z_{dard}[i] \leq 0$ et $z_{dard}[i-1] > 0$.

Notations :

– $timp_{dard}[k]$ instant de l'impact du dard dans l'eau.
– $pimp_{dard}[k]$ point d'impact du dard dans l'eau

$$pimp_{dard}[k] = \begin{bmatrix} x_{impdard} \\ y_{impdard} \\ z_{impdard} \end{bmatrix}$$

– $p_{dard}[k]$ point de la trajectoire du dard

$$p_{dard}[k] = \begin{bmatrix} x_{dard} \\ y_{dard} \\ z_{dard} \end{bmatrix}$$

– $pavant_{dard}[k]$ point de la trajectoire du dard avant l'impact dans l'eau

$$pavant_{dard}[k] = \begin{bmatrix} x_{avdar} \\ y_{avdar} \\ z_{avdard} \end{bmatrix}$$

– $papres_{dard}[k]$ point de la trajectoire du dard après l'impact dans l'eau

$$papres_{dard}[k] = \begin{bmatrix} x_{apdar} \\ y_{apdar} \\ z_{apdard} \end{bmatrix}$$

$$\begin{aligned} timp_{dard}[k] &= interpollineaire(t_{dard}[k-1], z_{avdard}, t_{dard}[k+1], z_{apdard}, 0) \\ x_{dard} &= interpollineaire(x_{avdard}, t_{dard}[k-1], x_{apdard}, t_{dard}[k+1], timp_{dard}[k]) \\ y_{dard} &= interpollineaire(y_{avdard}, t_{dard}[k-1], y_{apdard}, t_{dard}[k+1], timp_{dard}[k]) \\ z_{dard} &= interpollineaire(z_{avdard}, t_{dard}[k-1], z_{apdard}, t_{dard}[k+1], timp_{dard}[k]) \end{aligned}$$

5.7.4 Calcul de la position du bateau

Hypothèses :

– on suppose que la hauteur du bateau n'influence pas de façon importante la surface de l'ombre du bateau et peut donc être négligée dans le calcul.

Notations

– $\begin{bmatrix} a \begin{bmatrix} x \\ y \end{bmatrix} \\ b \begin{bmatrix} x \\ y \end{bmatrix} \\ c \begin{bmatrix} x \\ y \end{bmatrix} \\ d \begin{bmatrix} x \\ y \end{bmatrix} \end{bmatrix}$ les quatre points du rectangle représentant l'ombre du bateau.

- $t_{central}$ instant de l'impact dans l'eau du dard central de la roquette centrale de la gerbe.
- $x_{central}$ position en x du point d'impact du dard de la roquette centrale de la gerbe.
- $y_{central}$ position en y du point d'impact du dard de la roquette centrale de la gerbe.
- v_{bateau} vitesse du bateau.
- $long_{bateau}$ longueur du bateau
- $larg_{bateau}$ largeur du bateau
- $offset_x$ erreur de justesse en x
- $offset_y$ erreur de justesse en y

$$
\begin{aligned}
a\begin{bmatrix}x\\y\end{bmatrix} &= \begin{bmatrix}x_{central} + offset_x - \frac{long_{bateau}}{2} + v_{bateau}\cdot(timp_{dard}[k] - t_{central})\\ y_{central} + offset_y + \frac{larg_{bateau}}{2}\end{bmatrix}\\
b\begin{bmatrix}x\\y\end{bmatrix} &= \begin{bmatrix}x_{central} + offset_x + \frac{long_{bateau}}{2} + v_{bateau}\cdot(timp_{dard}[k] - t_{central})\\ y_{central} + offset_y + \frac{larg_{bateau}}{2}\end{bmatrix}\\
c\begin{bmatrix}x\\y\end{bmatrix} &= \begin{bmatrix}x_{central} + offset_x + \frac{long_{bateau}}{2} + v_{bateau}\cdot(timp_{dard}[k] - t_{central})\\ y_{central} + offset_y - \frac{larg_{bateau}}{2}\end{bmatrix}\\
d\begin{bmatrix}x\\y\end{bmatrix} &= \begin{bmatrix}x_{central} + offset_x - \frac{long_{bateau}}{2} + v_{bateau}\cdot(timp_{dard}[k] - t_{central})\\ y_{central} + offset_y - \frac{larg_{bateau}}{2}\end{bmatrix}
\end{aligned}
$$

5.7.5 Comptage des impacts sur le bateau

Notations :

- *toucher* nombre de dards qui ont impacté la zone d'ombre du bateau.

$$
toucher = toucher + 1 \text{ si } \begin{cases} & x_{impdard} > a_x\\ \wedge & x_{impdard} < b_x\\ \wedge & y_{impdard} < a_y\\ \wedge & y_{impdard} > d_y\end{cases}
$$

5.7.6 Comptage des impacts par unité de surface

notations :

- $long_{zone}$ longueur en mètre de la zone observée
- $larg_{zone}$ largeur en mètre de la zone observée
- $zone_impact[long_{zone}, larg_{zone}]$matrice de la zone d'impact
- $indice_x$ indice en longueur de la zone observée

– $indice_y$ indice en largeur de la zone observée
– $\begin{bmatrix} debut_{zone\_impactx} \\ fin_{zone\_impactx} \\ debut_{zone\_impacty} \\ fin_{zone\_impacty} \end{bmatrix}$ points de définition de la zone observée.
– $nbreimpactdard_{danszone}$ compteur d'impacts dans la zone
– $nbreimpactdard_{horszone}$ compteur d'impacts hors de la zone
– $nbreimpactdard$ compteur d'impacts dans et hors de la zone.

Pour chaque impact de dard faire

$$\text{si} \begin{cases} & x_{impdard} \geq debut_{zone\_impactx} \\ \wedge & x_{impdard} \leq fin_{zone\_impactx} \\ \wedge & y_{impdard} \geq debut_{zone\_impacty} \\ \wedge & y_{impdard} \leq fin_{zone\_impacty} \end{cases}$$

$$\text{alors} \begin{cases} nbreimpactdard_{danszone} + + \\ indice_x = x_{impdard} - debut_{zone\_impactx} \\ indice_y = y_{impdard} - debut_{zone\_impacty} \\ zone\_impact(indice_x, indice_y) + + \end{cases}$$

sinon

$$nbreimpactdard_{horszone} + +$$

finsi

$$nbreimpactdard + +$$

fin

5.7.7 Calcul du nombre de salves satisfaisant la performance définie

Notations :

– it nombre de salves dans la boucle du Monte-Carlo
– $nbre\_Imp\_surbateau\_parsalve[it]$ compteur du nombre de dards tombés sur l'ombre du bateau par salve
– $seuil\_de\_performance$ nombre de dards minimums pour détruire le bateau
– $nbre\_salve\_satis\_perfo$ nombre de salve satisfaisant la performance.

si $nbre\_Imp\_surbateau\_parsalve[it] \geq seuil\_de\_performance$
alors
$nbre\_salve\_satis\_perfo++$
finsi

5.8 Méthode de création d'un mot de passe

La structure du mot de passe destiné à ce programme devra suivre la recommandation :

> *Un mot de passe fort est complexe (associe des lettres en majuscules et en minuscules, des nombres et des caractères spéciaux) et n'est pas un mot.* [MS08-09]

FIGURE 5.3 – définition d'un mot de passe "fort"

avec le mot de passe *ABC1234/-* nous obtenons à l'issue d'essais avec des testeurs de mot de passe sur internet ;

- [AL08-09] a noté ce mot de passe comme "fort",
- [MSC08-09] comme "élevé",
- [CS08-09] lui donne sur une grille de notation de la complexité allant de 2 à 20 une note de 13 sur 20, sachant que 20 défini un mot de passe d'une très grande complexité et 2 un mot de passe composé d'une seule lettre. A titre d'exemple les mots de passe
  - *m0nMoT2p@s5E* obtient 16/20 et
  - *E=1/2m*V**2ouh.nuOubienm.C**2* obtient la note de 19 sur 20.
- [TC08-09] nous donne le temps de résistance à une attaque "brute-force" d'un mot de passe en fonction de sa taille et de sa composition. Pour ce faire Il définit plusieurs niveaux d'attaques :

  1. standard : Attaque de type brute-force exécuté sur un PC familial récent utilisant des outils de cassage de mot de passe disponibles gratuitement sur internet. Puissance estimée : 1 000 000 combinaisons testées par seconde.
  2. distribué : Mise en parallèle d'un réseau de 2500 ordinateurs zombies au sein d'un même botnet (estimation) pour se répartir la tâche de casser un mot de passe. Puissance estimée : 2 500 000 000 combinaisons testées par seconde.

3. avec l'ordinateur le plus puissant de la planète : Puissance de calcul de l'ordinateur le plus puissant de la planète, un ordinateur IBM BlueGene/L - eServer Blue Gene Solution du Lawrence Livermore National Laboratory (Californie - USA) utilisé en recherche. Puissance estimée : 478 200 GFlops soit 478 200 000 000 000 combinaisons testées par seconde.
4. Attaque utilisant les 500 plus puissants ordinateurs de la planète : Puissance de calcul obtenue en combinant le travail des 500 plus puissants ordinateurs de la planète (scénario peu concevable dans les faits). Puissance estimée : 6 965 082 GFlops soit 6 965 082 000 000 000 combinaisons testées par seconde. source : [T508-09]

Les résultats sont regroupés dans le tableau TAB.5.1 :

niveaux d'attaques	jours	années	siècles
1	137515.51	376.5	3.8
2	55.01	0.2	-
3	-	instantané	-
4	-	instantané	-

TABLE 5.1 – Evaluation de la résistance du mot de passe à différentes attaques brute-force

– [IR08-09] comme ayant un indice de compléxité de 280.

Il existe un grand nombre de méthodes pour évaluer la complexité d'un mot de passe cependant on retrouve souvent comme élément de mesure de la complexité les éléménts énoncés dans la recommandation FIG.5.3. Les valeurs données par les testeurs de mot de passe ne sont valables qu'en comparaison avec d'autres valeurs. A ce titre [IR08-09] a donné un indice de complexité de 275 au mot de passe *m0nMoT2p@s5E* qu'il donne en exemple sur leur page.

En conclusion nous pouvons estimer au regard de ces résultats que la structure du mot de passe utilisée en exemple peut être retenue pour notre programme.

5.9 Choix de quelques outils pour la réalisation

5.9.1 Fonction cdfnor() de Scilab

Scilab propose des fonctions cdf*, à partir desquelles on retrouve la fonction de répartition, la densité et la fonction quantile des lois les plus courantes.

Toutes ces fonctions sont structurées de façon analogue : si on leur donne en entrée toutes les quantités sauf une, ainsi que l'option appropriée, la fonction retournera la quantité manquante.

Dans l'exemple de la loi normale (fonction cdfnor), l'option "P Q" permet de calculer la fonction de répartition. Pour la fonction quantile, il faut utiliser l'option "X". Les options "Mean" et "Std" permettent de calculer la moyenne ou l'écart-type d'une loi normale dont on connaît l'autre paramètre et une valeur de la fonction de répartition.

P,Q =cdfnor("PQ",X,Mean,Std)

X =cdfnor("X",Mean,Std,P,Q)

Mean =cdfnor("Mean",Std,P,Q,X)

Std =cdfnor("Std",P,Q,X,Mean)

5.9.2 Fonction alnorGGT

La fonction alnorGGT prend comme argument la moyenne et l'écart-type associé d'une loi de probabilité. Elle tire au hasard un chiffre (a) compris entre 0 et 1 qui sera vue comme une probabilité, projette l'image de ce point sur la courbe de la fonction de répartition de loi normale centrée réduite$N(0,1)$ et projette le point (x) de la courbe sur l'axe des abscisses (voir la figure 5.9.2) et retourne

$$\text{moyenne} + \text{ecart-type} \times x$$

5.9.3 Fonction rand() du C

La fonction rand() est dans la bibliothèque standard du langage C. Elle retourne un entier pseudo-aléatoire. En l'absence d'initialisation d'une graine la fonction rand() utilise 1 comme valeur de départ.

Une étude sur les nombres générés par la fonction rand() de la bibliothèque standard du C a été effectuée. Elle montre que l'algorithme utilisé génère une suite de nombres pseudo-aléatoires qui suit une loi uniforme. Les résultats des tests sont en annexe(voir en annexe CHAP.11).

5.9.4 Fonctions rand() et grand() de Scilab

Fonction rand

Elle retourne une matrice ou un vecteur composé de réel. Le générateur utilisé pour cette fonction est basé sur la théorie suggérée par D.E. Knuth ((1969),vol 2, State). Il est rapide mais un peu dépassé. La documentation de Scilab recommande de ne pas l'utiliser pour des simulations sérieuses.

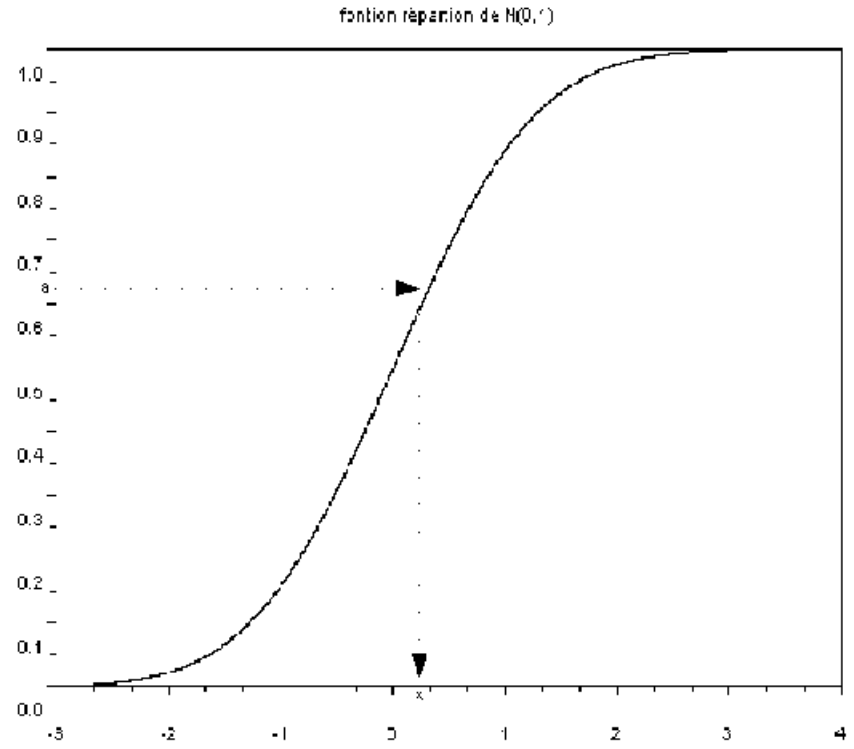

FIGURE 5.4 – fonction alnorGGT

Fonction grand

Elle retourne comme la fonction rand une matrice ou un vecteur composé de réels. A l'inverse de la fonction rand il est possible de choisir le générateur de nombre pseudo-aléatoire parmi une liste. La graine d'initialisation ne peut être choisie par l'utilisateur elle est définie par le générateur que l'on choisit.

Il est possible de choisir parmi les générateurs suivant :

- Mersenne-Twister de M. Matsumoto et T. Nishimura avec une période de 2^{19937} et une graine donnée par un tableau de 624 entiers.
- kiss de G.Marsaglia avec une période de 2^{123}et une graine donnée par un tableau de 4 entiers.
- clcg2 une combinaison de 2 générateurs de P L'Ecuyer avec une période de 2^{61}et une graine donnée par un tableau de 2 entiers.
- clcg4 une combinaison de 4 générateurs de P L'Ecuyer avec une période de 2^{121}et une graine donnée par un tableau de 4 entiers.
- urand le générateur de la fonction rand()et une graine donnée par un entier.

– fsultra un générateur "Subtract-with-Borrow" [5] mélangé avec le générateur de Arif Zaman et George Marsagalia avec une période de 10^{356} et une graine donnée par un tableau de 37 entiers.

Le générateur utilisé par défaut est le générateur Mersenne-Twister de M. Matsumoto and T. Nishimura .

5.9.5 Microrotateur

le microrotateur permet la rotation des points d'une courbe de référence autour de son point d'origine sans appel aux fonctions de calcul de cosinus et sinus de l'angle de rotation.

hypothèses

– l'angle de rotation est de quelques milliradians.

notations

– $OP_{roq}[j]$: point courant de la trajectoire après rotation.
– $OP_o[j]$: point origine de la trajectoire.
– $P_{ref}[1]$: premier point de la trajectoire de référence.
– $P_{ref}[j]$: point courant de la trajectoire.
– $\vec{\Theta}$: angle de déviation de la trajectoire.
– $danx$,$dany$,$danz$: valeurs de déviations des angles.
– xo,yo,zo : coordonnées de l'origine de la trajectoire translatée.
– xav,yav,zav : premier point de la trajectoire. de référence.
– $xref$,$yref$,$zref$: point courant de la trajectoire de référence.
– $xroq$,$yroq$,$zroq$: coordonnées du point courant de la trajectoire après rotation.

$$\overrightarrow{OP_{roq}[j]} = \overrightarrow{OP_o[i]} + \overrightarrow{P_{ref}[1]P_{ref}[j]} - \overrightarrow{P_{ref}[1]P_{ref}[j]} \wedge \vec{\Theta}$$

$$\overrightarrow{OP_o[i]} = \begin{bmatrix} x_o[i] - 0 \\ y_o[i] - 0 \\ z_o[i] - 0 \end{bmatrix} \quad \vec{\Theta} = \begin{bmatrix} danx \\ dany \\ danz \end{bmatrix} \quad \overrightarrow{P_{ref}[1]P_{ref}[j]} = \begin{bmatrix} x_{ref}[j] - x_{av} \\ y_{ref}[j] - y_{av} \\ z_{ref}[j] - z_{av} \end{bmatrix}$$

$$\overrightarrow{P_{ref}[1]P_{ref}[j]} \wedge \vec{\Theta} = \begin{bmatrix} (y_{ref}[j] - y_{av} \cdot danz) + (z_{ref}[j] - z_{av} \cdot dany) \\ (x_{ref}[j] - x_{av} \cdot danz) - (z_{ref}[j] - z_{av} \cdot danx) \\ (x_{ref}[j] - x_{av} \cdot dany) + (y_{ref}[j] - y_{av} \cdot danx) \end{bmatrix}$$

5. soustraire avec emprunt

$$\overrightarrow{OP_{roq}[j]} = \begin{bmatrix} x_o[i] \\ y_o[i] \\ z_o[i] \end{bmatrix} + \begin{bmatrix} x_{ref}[j] - x_{av} \\ y_{ref}[j] - y_{av} \\ z_{ref}[j] - z_{av} \end{bmatrix} - \begin{bmatrix} (y_{ref}[j] - y_{av} \cdot danz) + (z_{ref}[j] - z_{av} \cdot dany) \\ (x_{ref}[j] - x_{av} \cdot danz) - (z_{ref}[j] - z_{av} \cdot danx) \\ (x_{ref}[j] - x_{av} \cdot dany) + (y_{ref}[j] - y_{av} \cdot danx) \end{bmatrix}$$

5.9.6 Interpolation linéaire

Cette fonction calcule l'abscisse d'un point d'une droite connaissant son ordonnée et deux points appartenant à la droite .

notation

y :coordonnée du point sur l'axe des ordonnées
x :coordonnée du point sur l'axe des abscisses.
a :coefficient directeur de la droite
b :ordonnée à l'origine.
x_1 ,y_1,x_2,y_2 : deux points appartenant à la droite.

$$a = \frac{y_2 - y_1}{x_2 - x_1}$$
$$b = y_2 - a \cdot x_2$$
$$x = \frac{y - b}{a}$$

si $x1 = x2$ alors $x = x1$
si $y1 = y2$ alors $x = \infty$

5.9.7 Choix des outils de rédaction

La rédaction de note technique contenant des formules mathématiques avec un éditeur standard est très consommatrice de temps. Le choix du composeur de texte LaTeX s'est imposé tout naturellement après la difficulté rencontrée pour rédiger une note technique sous Word. le choix de LaTeX apporte son lot d'avantages.

Afin de ne pas être obligé de rédiger un document technique lors de la création de programme en C nous recherchions un outil qui au stade du codage du projet nous permet d'extraire les commentaires écris dans du code avec l'idée de ne rédiger qu'une seule documentation en langage LaTeX associée au programme et l'extraire pour réaliser une note technique. Notre choix s'est porté vers FunnelWeb un préprocesseur de programmation lettrée de Ross Williams, indépendant du langage de programmation, fonctionnant sur plusieurs plate-formes dont Linux, MAC OSX , Windows. Rapide, il supporte la génération de document aux formats TeX, LaTeX, HTML.

Pour écrire l'ensemble de ces documents il nous fallait choisir un éditeur de texte. Le choix d'un éditeur s'est fait selon plusieurs critères. Il devait permettre la rédaction de code en C, en FunnelWeb, en LaTeX, avoir un langage de création de macro , le lancement de commande depuis une console, en fait être multi-langages, personnalisable, multi-plateformes, ayant une mini console intégrée et sauvegardant les fichiers au format texte. L'éditeur Xemacs a été retenu.

Nous développons les raisons de ces choix en annexe CHAP.12.

5.10 Suivi documentaire

Il associe les documents du référentiel THALES aux étapes de stratégie de développement logiciel, en phase avec les processus définie dans le CMMI au niveau de maturité 2. Le CMMI est un référentiel composé d'un ensemble structuré de bonnes pratiques, destiné à appréhender, évaluer et améliorer les activités des entreprises d'ingénierie. Il est un complément de la norme ISO9001 dans la cadre du développement logiciel.

Nous présentons le CMMI et réalisons la corrélation entre le référentiel CMMI le référentiel ISO et la structure de la documentation TDA dans l'annexe CHAP.12.

Chapitre 6

Réalisation d'une maquette de calcul d'efficacité

Nota : Les valeurs de dispersions prises en compte dans cette note ne sont pas les valeurs réelles. Elles ne sont retenues que pour illustration de la méthode de calcul.

6.1 L'étape de fiabilisation du besoin

6.1.1 Le Besoin

Le besoin est de déterminer le nombre de dards qui touchent un bateau. Les dards sont expulsés après un temps programmable entre 2 et 30 secondes d'une tête de roquette tirée depuis un avion. L'avion vole à une vitesse de 450 nœuds, à une altitude de 10000 ft. Le bateau mesure 10 m de long, 3 m de large et se déplace à la vitesse de 55 nœuds. Le pilote souhaite tirer à une distance de 6 km avec un angle de site de 20° devant la cible. La cadence de tir est de 30 coups / seconde.

6.1.2 Une première évaluation

Dans un premier temps nous avons effectué une première évaluation en réalisant un calcul de probabilité d'efficacité d'un tir de roquette de 36 dards sur une cible de 1 m^2. Le programme est réalisé avec un tableur. Pour une réutilisation plus facile dans un module j'ai reécrit le programme sous Scilab. Ce module sera depuis le bloc *effet terminal* , les résultats seront utilisables dans le module suivant *analyse scénario.*

Dans le tableau ci-dessous sont stockés les résultats des premières estimations.

Nous regardons la variation de probabilité qu'un dard tombe sur une surface en fonction de la dispersion de la roquette et du nombre de roquettes tirées.

hypothèses

1. on néglige l'effet de rotation propre de la roquette.
2. on considère que la dispersion du temps de dépotage est nulle.
3. on considère que la conduite de tir est parfaite.

conditions initiales

1. la cible est centrée en (0,0) (pas d'erreur de justesse de tir)
2. l'aire de la cible est de $1m^2$.
3. le nombre de dards nécessaire pour neutraliser la cible est égal à 1
4. le rayon de la gerbe est variable : il dépend de la temporisation du dépotage et de la dispersion de l'angle d'ouverture d'éjection des dards qui est de quelques milliradians par $\frac{1}{2}$ angle. de façon arbitraire nous posons le rayon de la gerbe à 13 mètres
5. le nombre de roquette est variable.
6. la distance de tir est de 6000m
7. la dispersion de la roquette est de quelques milliradians en site et azimut.

résultats

dispersion en mètre	nombre de roquettes											
	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0	0	0	0	0
30	0	0	0	0	0	0	0	0	0	0	0	0

TABLE 6.1 – tableau des probabilités d'un dard par m^2 fonction de la dispersion et du nombre de roquettes

A l'examen des résultats nous allons chercher le rayon de gerbe en fonction du nombre de roquettes qui nous donnera un meilleur résultat.

hypothèses

1. on néglige l'effet de rotation propre de la roquette.
2. on considère que la dispersion du temps de dépotage est nulle.
3. on considère que la conduite de tir est parfaite.

conditions initiales

1. la cible est centrée en (0,0) (pas d'erreur de justesse de tir)
2. l'aire de la cible est de $1m^2$.
3. le nombre de dard nécessaire pour neutraliser la cible est égal à 1
4. le rayon de la gerbe est variable.
5. le nombre de roquette est variable.
6. les écarts de dispersion de la roquette sont posés à 0.

résultats

rayon en mètre	nombre de roquettes											
	1	2	3	4	5	6	7	8	9	10	11	12
1	0,39	0,62	0,77	0,86	0,91	0,94	0,96	0,98	0,99	0,99	0,99	0,99
2	0,85	0,97	0,99	0,99	0,99	0,99	0,99	0,99	1,0	1,0	1,0	1,0
4	0	0,99	,99	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0
6	0	0	0	0,99	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0
8	0	0	0	0	0	1,0	1,0	1,0	1,0	1,0	1,0	1,0
10	0	0	0	0	0	0	0	0	1,0	1,0	1,0	1,0
12	0	0	0	0	0	0	0	0	0	0	0	0

TABLE 6.2 – tableau des probabilités d'un dard par m^2 fonction du rayon de la gerbe et du nombre de roquettes

Ce tableau nous montre que pour un nombre de roquettes égal à 12 et un rayon de gerbe inférieur à 12 mètres il y a une forte probabilité de réussite de mettre un dard par mètre carré. Cependant ce calcul ne représente pas la réalité.

Nous réalisons donc un nouveau calcul avec une surface plus grande et un nombre de dards différent.

hypothèses

1. on néglige l'effet de rotation propre de la roquette.

2. on considère que la variation du temps de dépotage est nulle.
3. on considère que la conduite de tir est parfaite.

conditions initiales

1. la cible est centrée en (0,0) (pas d'erreur de justesse de tir)
2. l'aire de la cible est de $30m^2$.(choix arbitraire)
3. le nombre de dards nécessaire pour neutraliser la cible est égal à 5,
4. le rayon de la gerbe est variable.
5. le nombre de roquettes est variable.
6. les écarts de la roquette sont posés à 0.

résultats

rayon en mètre	nombre de roquettes											
	1	2	3	4	5	6	7	8	9	10	11	12
1	0,39	0,62	0,77	0,86	0,91	0,94	0,96	0,98	0,99	0,99	0,99	0,99
2	0,85	0,97	0,99	0,99	0,99	0,99	0,99	0,99	1,0	1,0	1,0	1,0
4	0,99	0,99	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0
6	0,99	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0
8	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0
10	0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0
15	0	0	0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0
20	0	0	0	0	0	1,0	1,0	1,0	1,0	1,0	1,0	1,0
25	0	0	0	0	0	0	0	0	0	1,0	1,0	1,0

TABLE 6.3 – tableau des probabilités d'un dard pour une cible de $30m^2$ fonction du rayon de la gerbe et du nombre de roquettes

Nous constatons que le rayon de la gerbe satisfaisant les contraintes a augmenté ; de 10 mètres à 25 mètres pour un nombre de roquettes égal à 12, mais cela est sans compter avec les écarts de dispersions des roquettes.

Nous allons donc faire varier les écarts de dispersion de la roquette pour simuler des points d'impacts à coté de la cible. Nous définissons un quart de zone d'observation de 50 mètres de long par 30 mètres de large.

hypothèses

1. on néglige l'effet de rotation propre de la roquette.

2. on considère que la variation du temps de dépotage est nulle.
3. on considère que la conduite de tir est parfaite.
4. on suppose une répartition symétrique des dards sur la surface de la zone observée.

conditions initiales

1. la cible est centrée en (0,0) (pas d'erreur de justesse de tir)
2. l'aire de la cible est de $30m^2$.
3. le nombre de dard nécessaire pour neutraliser la cible est égal à 5,
4. le rayon de la gerbe est de 15 m.
5. le nombre de roquette est de 12.
6. les écarts de dispersion de la roquette sont variables.

résultats

dispersion de la roquette à l'impact					
en y	en x				
	0	**17**	**25**	**40**	**50**
0	1	0,97	0,78	0,38	0,23
7	1	0,95	0,73	0,33	0,19
15	0,99	0,66	0,78	0,11	0,23
22	0,87	0,81	0,16	0,04	0,02
30	0,63	0,18	0,07	0,02	0,00

TABLE 6.4 – tableau des probabilités qu'au moins 5 dards touchent le bateau en fonction de la dispersion de la roquette

Ces derniers résultats nous montrent que, sous réserve que la justesse de tir soit parfaite, il est possible de toucher de 5 dards un bateau avec une salve de 12 roquettes tirée depuis un avion. En observant le tableau TAB.6.4. On constate qu'il existe une probabilité de plus de 90% de satisfaire la performance dans un rectangle de longueur inférieure à 17 m de long et de largeur inférieure à 15 m avec le bateau en son centre .

Nous allons poursuivre, en utilisant le même cas, avec la réalisation d'une seconde maquette afin de valider un méthode de communication entre les modules

Chapitre 7

Réalisation d'une maquette de communication

Ce programme va être écrit en langage C il servira de test pour définir une méthode de construction de fonctions que l'on va vouloir réutilisable dans des programmes future en C.On utilisera la méthode de Monte-Carlo pour les calculs statistiques. Ce programme devra permettre de réaliser des graphiques de répartition de dards dans une zone observée, il affichera sur la console le pourcentage de salves de n roquettes satisfaisant la contrainte de performance pour détruire le bateau(elle se traduit par un nombre minimum de dards ayant touché le bateau), la vitesse et la pente du dard à l'impact pour permettre des calculs de perforation qui ne font pas partie de ce programme.

Pour des raisons de temps de calcul, il sera utilisé une trajectoire de référence pour la roquette ainsi que pour les dards. A ces trajectoires nous ajouterons un terme probabiliste représentant la dispersion . Ces dispersions seront simulées par des variables aléatoires gaussiennes caractérisées par un écart-type. Ces variables influencent la justesse de tir, les écarts de temps de dépotage, l'angle d'éjection des dards. Les trajectoires de référence devront être modifiables. Un logiciel de calcul est prévu pour pouvoir réaliser des trajectoires de référence utilisables avec ce logiciel. La création de ce logiciel ne fait pas partie de ce projet.

Les arguments pour les calculs seront :

- l'altitude de l'avion
- la pente de l'avion
- le nombre de salves
- le nombre de roquettes dans une salve
- le nombre minimum de dards nécessaire pour détruire le bateau
- la valeur de la dispersion de la justesse en sortie de lanceur
- la dimension de la zone d'impact observée
- la dimension du bateau

Le programme nous affiche comme résultat :
- l'instant d'impact de la roquette centrale dans l'eau
- le retard de dépotage
- la distance entre l'avion et la cible sous axe avion(en mètres)
- la vitesse des dards à l'impact(en ms^{-1})
- la pente du dard à l'impact (en degrés)
- le nombre moyen de dards qui touche le bateau pour chaque roquette
- le nombre moyen de dards qui touche le bateau pour une salve.
- le nombre de salves dont au moins le nombre de dards nécessaire ont touché le bateau
- la proportion de dards tombés dans la zone observée
- la proportion de dards tombés hors de la zone observée
- le nombre total de dard tombés.

Il écrit aussi dans un fichier les valeurs de densité de dards par mètre carré dans la zone observée.

7.0.3 L'interface graphique

Pour ce programme l'interface graphique sera sobre, la priorité est placée sur la précision du calcul ainsi que sa vitesse. Il ne sera donc pas réalisé d'interface avec des fenêtres et menu mais seulement une suite de questions posées dans une console. L'avantage est que le programme est peu encombrant et ne demande aucune bibliothèque extérieure. De plus l'absence d'interface élaborée permet de gagner du temps.

7.0.4 Les étapes de la conception

Nous décidons de créer un logigramme pour segmenter le programme en actions simples.

Ensuite nous écrivons une note technique décrivant en détail les opérations qui constitueront le programme à l'exception des fonctions de lecture d'argument sur la console, de lecture de fichiers contenant les trajectoires de référence, d'écriture de fichier résultat et d'affichage des résultats. Cette note technique permet aussi d'expliquer la démarche employée pour réaliser nos calculs et de rappeler nos hypothèses.

7.0.5 Le codage

Le codage d'une première version est réalisé. il reprend les opérations du logigramme de la figure FIG.7.1 dans un programme monolithique , sans création de

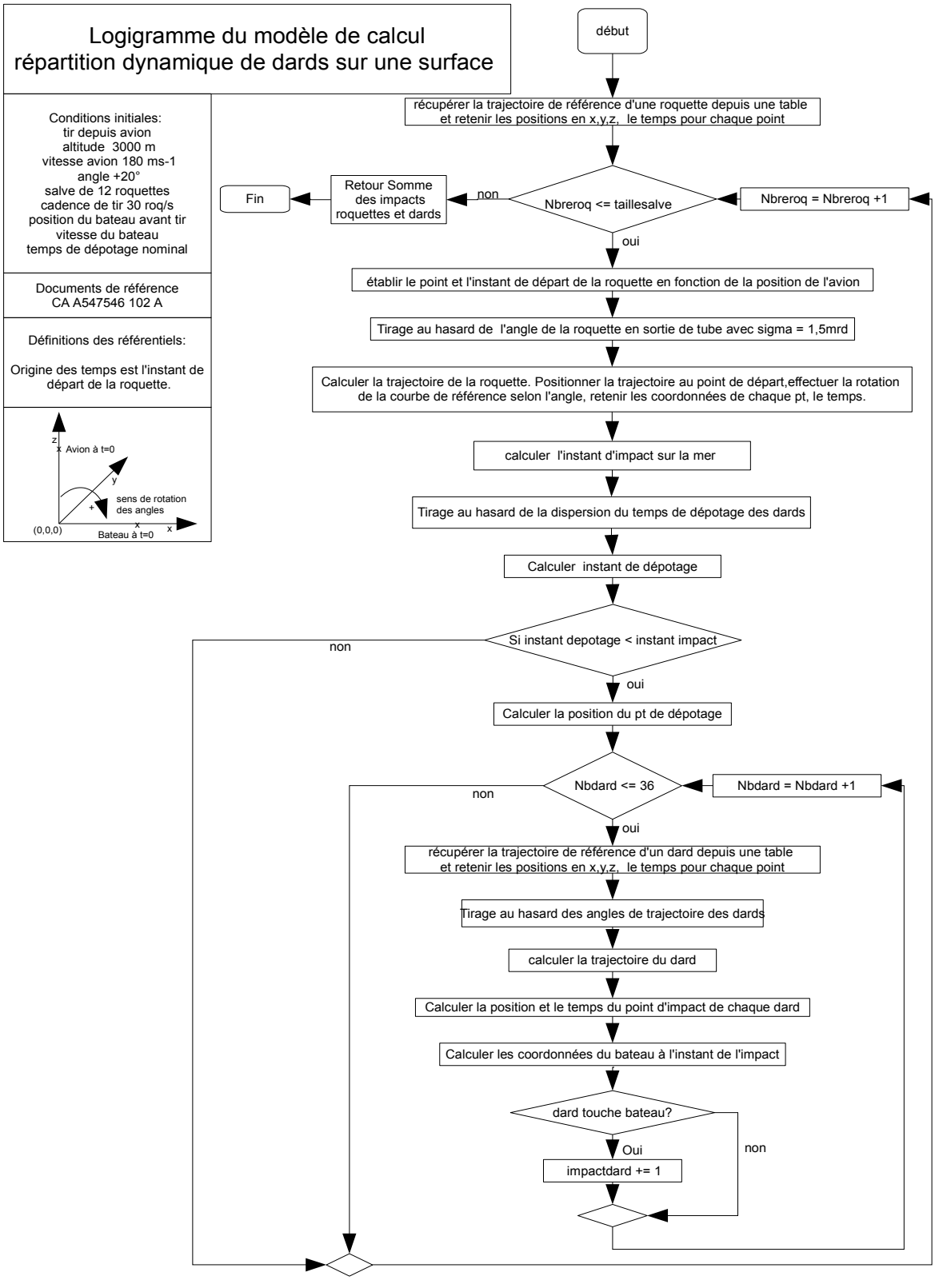

FIGURE 7.1 – logigramme du programme de calcul de densité de dards

fonctions. Il nous permet d'obtenir les premiers résultats, de mettre au point les calculs .

Cette première étape réalisée nous définissons les fonctions nécessaires à l'évolution du programme. Elles doivent remplir au plus près les besoins définis dans le logigramme.

Le programme doit évoluer afin de pouvoir changer de trajectoire, réaliser des graphiques de probabilité d'atteinte du bateau selon sa position par rapport au point d'impact de la roquette centrale, connaître au mieux la vitesse, les coordonnées et la pente des dards à leur point d'impact.

Ces nouveaux besoins nous amènent à apporter quelques modifications ; le calcul d'une trajectoire de référence des dards à chaque début de calcul, l'écriture de deux nouveaux programmes : l'un pour la création de fichiers de trajectoire de référence pour les roquettes, l'autre pour le calcul des probabilités de réussite.

Pour le calcul de la trajectoire de référence des dards nous nous sommes inspirés d'un programme existant fonctionnant dans une page de tableur qui calcule la trajectoire d'une bille quand on connaît sa position de départ, sa vitesse initiale, son coefficient de traînée. Il est réécrit pour fonctionner sous Scilab et en C et nous spécialisé pour le calcul de trajectoires de dards.

Le calcul des trajectoires de référence pour les roquettes est réalisé par un logiciel spécifique écrit en Fortran et compilé pour fonctionner sous Windows. Lors de son lancement il lit un fichier de référence contenant les valeurs d'initialisation du calcul. C'est un fichier texte sans commentaire dans lequel nous devons modifier 4 lignes parmi une centaine. Le risque de modifier un paramètre par erreur est grand et un programme est créé pour automatiser au mieux la création de trajectoires de référence pour les roquettes. Ce programme ce décompose en quatre étapes :

- Création du fichier de paramètres pour le calcul de la trajectoire
- Appel du programme de calcul de la trajectoire
- Post-traitement des fichiers résultats
- création du fichier de trajectoire .

Ce fichier de trajectoire est un élément d'entrée pour le bloc *trajectoire munition*. A terme son stockage dans une base de données est a examiner.

Le programme de calcul de probabilité est réalisé en utilisant les modules déjà écrits pour le programme de calcul de répartition. On utilise le programme de calcul de répartition que l'on répète dans une double boucle dont les arguments sont les positions possibles du bateau en rapport du point d'impact de la roquette centrale de la salve, les dimensions de la surface observée et le nombre d'itération pour la méthode de Monte-Carlo.

Pour finir nous avons ajouté une fonction de demande et de gestion d'un mot de passe à l'ouverture des deux programmes de calcul.

7.0.6 Les tests

Pour ce premier programme il n'a pas été réalisé de revue de code. Seul ont été effectués des tests de fonctionnement et de non régression pour les deux programmes principaux de calcul ainsi que pour les fonctions rajoutées pour le calcul de trajectoire de dards ainsi que la fonction de demande de mot de passe.

Les différents tests ont permis de remarquer la répartition équiprobable des impacts sur le bateau entre les roquettes composants une salve.

7.1 Réalisation de la maquette

A ce stade du développement du projet le plan de développement a été mis à jour. Ce document est le document de référence pour la prévision et le suivi du développement d'un produit logiciel.

Il peut, pour des projets dont la charge est inférieure à 1 homme/an, se constituer , de la description du processus logiciel, la fiche de suivi , le planning de l'affaire.

Pour le projet en cours une évaluation de l'effort, du temps de développement et des ressource selon la méthode COCOMO a été éffectuée.

A la suite de la réalisation du programme monolithique nous pouvons évaluer le nombre de kligne à 1.3. Le programme est de taille inférieure à 50 klignes, dans un domaine parfaitement connu et réalisé par une petite équipe soit pour la méthode COCOMO ; un programme de type 'S'.

Les résultats du calcul d'une première estimation avec un intervalle de confiance de -25% à +75% sont dans le tableau TAB.7.1.

	Optimiste	Probable	Pessimiste
effort(MH)	3.16	4.21	7.36
TDEV (Mois)	3.87	4.32	5.34
Productivité	0.41	0.31	0.18
Ressource (Homme)	0.82	0.98	1.38

TABLE 7.1 – tableau des évaluations en mode basic

7.1.1 Réalisation des spécifications de besoin

A ce stade nous allons reprendre l'ensemble des besoins exprimés lors de la création de la maquette pour les classer. Ce classement sera guidé par la méthode expliquée dans [THA1] [THA2] [THA3] .

Ces spécifications permettront d'établir les tests au tout au long du développement, elles sont notées dans un document technique appelé « Spécification des

exigences pour le programme XXXX ». Cette note sera classée dans la GED avec un numéro (souvent le numéro du projet) et un code-type de document 306.

Son contenu rappelle le domaine d'application du logiciel suivi d'une description générale du produit logiciel et des exigences. Le but est de rédiger des exigences claires, non équivoques, quantifiables et vérifiables, et applicables au logiciel.

La description générale reprend les grandes fonctions que doit remplir le logiciel ; dans notre cas elles sont au nombre de 4 :

1. Gérer la saisie du mot de passe.
2. Convertir des unités de vitesse, de distance, d'angle.
3. Créer des fichiers de trajectoire de référence pour les roquettes.
4. Calculer le nombre d'impact de dard sur une cible et sur une zone observée

Les domaines des exigences sont les suivants :
- Exigences relatives aux capacités du logiciel ;
- Exigences relatives aux interfaces externes ;
- Exigences relatives à la sécurité-confidentialité et à la propriété
- Exigences relatives aux moyens informatiques ;
- Exigences relatives à la formation ;

La description des exigences peut être :
- soit pour une exigence spécifique au logiciel clairement définie comme par exemple pour la gestion du mot de passe :

  - *Le mot de passe est composé de 3 lettres 4 chiffres et deux caractères non alpha numériques différents. La fréquence de changement du mot de passe est journalière et automatique.*

- ou faire référence à un document pour des produits existant comme :

  Pour le programme de calcul de trajectoire de référence de roquette voir le document A000002-306

  Elles définissent les arguments entrant et sortant du programme, leurs limites, leur unité, leur disposition sur les écrans...etc.
  En plus des exigences le document décrit les caractéristiques qualité ; la capacité fonctionnelle, fiabilité,maintenabilité..., les contraintes de conception et d'implémentation et le classement des exigences par ordre de priorité.

7.1.2 Réalisation des étapes de conception

Il existe dans les guides de conception[THA10] [THA9] deux possibilités de concevoir le logiciel ; en mode structuré ou mode objet.La décision a été prise de concevoir le logiciel en mode structuré.

Les étapes de conception sont décrites dans un document dont le code type est 549. Il se compose de deux parties :

- la conception générale (ou préliminaire) du produit logiciel, son architecture, ses besoins en ressources, sa décomposition en sous programmes et leur conception. On définira aussi à cette étape les règles de codage globales tel que le choix de valeurs de sortie de programme...
- la décomposition des sous-ensembles et leur conception (ou conception détaillée).

dont voici un extrait

Décisions de conception générale

Le programme calmoddards réalise le calcul de répartition de dards sur une surface, de la somme des impacts sur une cible de type bateau.

Pour réaliser ces calculs il est utilisé la méthode de Monté-Carlo. Le choix a été fait d'utiliser une trajectoire de référence pour les roquettes et les dards, de réaliser des tirages aléatoires d'indices de variation et d'appliquer les modifications aux trajectoires de référence pour chaque roquette et chaque dards. Les indices de variation influencent autant les instants que les coordonnées. Les variables influencées sont : l'angle de sortie de lanceur de la roquette, l'angle d'éjection des dards et l'instant de dépotage des dards.

Les indices de variation sont considérés comme gaussien. Il est donc utilisé une fonction de tirage aléatoire de valeur reposant sur la fonction de répartition de la loi Normale centrée réduite.

Il a été choisi de réaliser les modifications sur les coordonnées composant les trajectoires avec un microrotateur prenant en charge les translations et rotations de chaque point de la trajectoire.

Le choix du numéro 9 comme chiffre pour sortir des différents écrans du logiciel est arbitraire.

Le nom (densitedards.txt) du fichier de sauvegarde des valeurs de proportion d'impact par m^2 sur la zone observée est fixe, et arbitraire.

L'ordre des choix affichés dans l'écran principal de calmoddards est motivé par les raisons suivantes :

- Il peut être nécessaire de convertir des valeurs de vitesses ou de distances avant de réaliser le calcul d'une trajectoire de référence ou le calcul d'impacts.
- Le calcul d'une trajectoire de référence pour les roquettes est indispensable lors de la première utilisation de calmoddards.
- Il doit être possible d'utiliser le calcul d'impact avec des trajectoires de référence existantes sans être obligé de les recalculer.

Conception de l'architecture du produit logiciel

composants du produit logiciel

Le produit calmoddards est composé d'un module principal de calcul d'impacts sur une surface ou une cible auquel est associé deux modules complémentaires : un convertisseur et un créateur de fichier de trajectoire de référence pour les roquettes.

Chacun de ces modules est composé de fonctions.

Les données nécessaires aux calculs sont stockées dans des structures.

aspects dynamiques

Les éléments nécessaires aux calculs sont saisis dans différents écrans affichés dans une console. Selon le choix de l'utilisateur il peut accéder à un des trois modules comme décrit dans la figure 7.2. Les choix des utilisateurs sont retenus dans le programme sous forme de chiffre. Les fonctions calculimpact et trajectoire roquette sont composés d'appel à des fonctions. Le choix a été fait d'utiliser une structure de fonctions représentée dans la figure 7.6. L'ordonnancement des fonctions est décrite dans les figures 7.3, 7.4,7.5 .

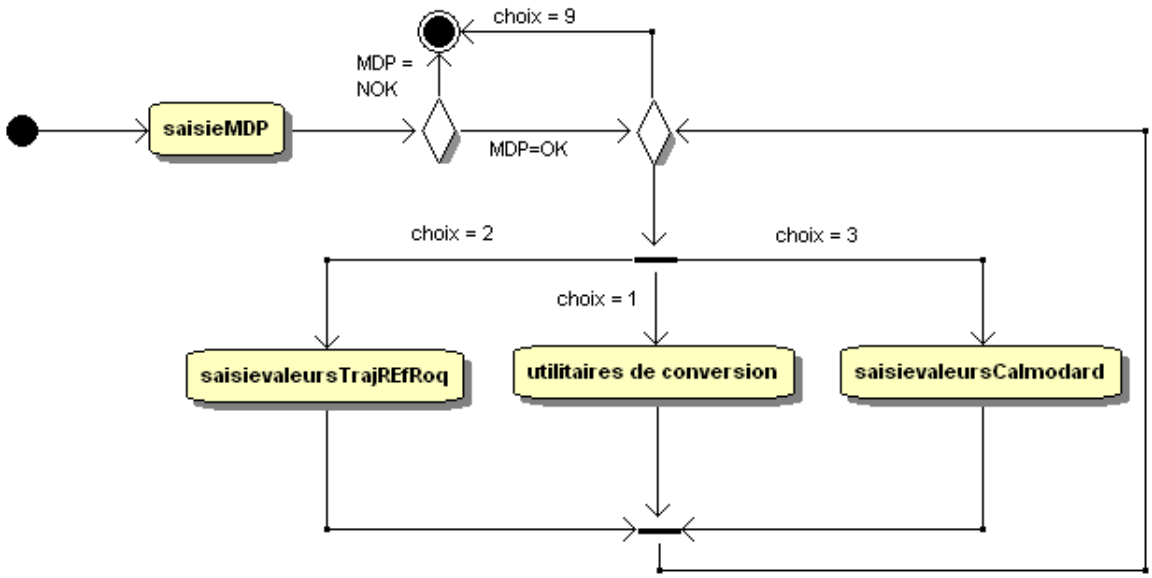

FIGURE 7.2 – diagramme de transition d'état de l'écran principal

7.1.3 conception des interfaces

Dans cette section sont décrits les différents écrans qui composent l'interface de calmoddards.

Après la saisie du mot de passe s'affiche un écran principal depuis lequel on peut accéder aux 3 fonctions principales de calmoddards soit : le convertisseur, le créateur de fichiers de trajectoire de référence pour les roquettes, le calcul d'impact sur une surface et une cible comme décrit dans la figure 7.2. La figure 7.7 est une copie de l'écran visible après la frappe du mot de passe.

Lors du lancement de la fonction de calcul des impacts, l'utilisateur renseigne les informations nécessaires aux calculs (voir la figure 7.8). Durant le calcul un écran affichant les informations globales est visible. Depuis cet écran l'utilisateur peut vérifier l'avancement du calcul grâce à la présence d'une barre de progression comme sur la figure 7.9. Le calcul est fini quand les informations statistiques sont affichées ainsi que le temps de calcul(voir la figure 7.10). Les densités d'impact sont écrites à la fin du calcul dans deux fichiers se nommant « densitedards.txt » pour une utilisation avec un tableur et « densitedards.sci » pour une utilisation directe dans Scilab.

Les différents états de transition de la fonction calculimpact sont décrits dans les figures 7.3, 7.4, 7.5 . A la fin de l'utilisation d'un de ces 3 composants le logiciel réaffiche l'écran principal.

On note dans le document les décisions de conception générale ou globale comme le choix d'un nom de fichier de résultat ou le choix d'un code de sortie des menus de choix.

On notera l'architecture du logiciel, les composants existants intégrés au programme, les composants propres. Une description des interfaces homme-machine sera effectuée. Les aspects statiques et dynamiques des interfaces (homme-machine, système) sont aussi décrits.

Conception détaillée

Pour le paragraphe concernant la structure détaillée sont inscrits les définitions des structures à utiliser ainsi que les signatures des fonctions.

liste des structures de données

```
typedef struct {
int x;
int y;
}I_point2d;
```

```
typedef struct {
float x;
float y;
float z;
} F_point3d ;

typedef struct{
int lon;
int lar;
int vitesse;
float xa;
float xb;
float xc;
float xd;
float ya;
float yb;
float yc;
float yd;
}  bateau;

typedef struct  {
int longueur;
int largeur;
int debutZoneEnX;
int finZoneEnX;
int debutZoneEnY;
int finZoneEnY;
} surface;

typedef struct {
F_point3d coord;
float instant;
float theta;
float vitesse;
} pt_trajectoire;

typedef struct{
float nbreImpactDard;
float nbreImpactDansZone;
float nbreImpactHorsZone;
float nbreImpactSurBateau;
```

```
 float nbreImpactSurBateauParSalve;
 float nbreSalveSatisfaisantLaPerf;
 float nbreRoquetteQuidepote;
 } gestionImpact;
```

liste des fonctions

fonctions de niveau 1

```
 /* fonction motDePasse()  Son fonctionnement est decrit
  * dans la note NT/PhD/014.
  */
int motDePasse();

 /* La fonction saisieprincipale(int* choix) affiche un menu de
  * choix et retourne la valeur du choix retenu par
  * l'utilisateur.
  */

int saisieprincipale(int* choix);

 /* la fonction convertisseur() contient les fonctions de calcul de
  * conversion.
  */

int convertisseur();

 /* la fonction trajrefroquette() affiche un menu de choix pour
  * permettre a l'utilisateur de réaliser plusieurs trajectoire
  * de référence sans rappeler le programme a chaque fois.
  */

int trajrefroquette();

 /* fonction de calcul de repartition des impacts sur une zone
  * et comptabilise le nombre d'impact sur une cible de type
  * bateau
  */

int calculimpact();
```

fonctions utilisées pour créer une trajectoire de référence

```
/* fonction d'ecriture du fichier F3avion.txt d'initialisation
 * du programme simul1a.exe de calcul de trajectoire pour
 * roquette
 *
 * elle prend comme arguments sur le porteur :
 *      l'altitude en metre
 * la pente en degres
 * la vitesse en m/s
 */

int creatF3avion(float altitude, float pente, float vitesse);

/* fonction d'ecriture du fichier de trajectoire de reference
 * pour roquette. Elle utilise l'altitude, la pente, la vitesse
 * pour creer le nom du fichier.
 *
 * le nom du fichier est de la forme : Traj_AAAA_PPP_VVVV.txt
 *
 * avec AAAA : l'altitude,
 *      PPP  : La valeur de la pente et
 *      VVVV : la valeur de la vitesse en m/s.
 */
```

fonctions utilisées pour le calcul d'impact de fléchettes

```
/* Fonction d'initialisation des variables necessaires aux
 * calculs.
 * elle affiche une suite de demande et retourne:
 * le chemin d'acces au fichier de la trajectoire de
 * reference pour les roquettes,
 * la valeur de la dispersion de la roquette,
 * la quantite de paires de roquettes par salve,
 * la quantite de flechettes necessaires pour
 * neutraliser la cible,
 * la distance d'ejection des flechettes sur la
 * trajectoire,
 * les coordonnees souhaitees pour deplacer le point
 * vise, pour simuler une erreur de vise,
```

```
 * les dimensions et la vitesse de la cible,
 * les dimensions de la surface observee,
 * le nombre d'iterations pour la boucle de Monte-Carlo.
 *
 */
int saisievaleur(char* nomfichier, float* disproq, int* qteroqsalve,
        int* perfo, int* distdepot, I_point2d* just, bateau* cibl,
        surface* zone, int* nombresalve);


/* Fonction de comptage du nombre de ligne contenu dans le fichier
 * de trajectoire de reference pour les roquettes pour dimensionner
 * la taille du tableau contenant les instants et les coordonnees
 * de la trajectoire de reference.
 *
 * Elle prend comme argument en entree un pointeur sur le premier
 * element du tableau contenant le chemin d'acces au fichiers de
 * trajectoire de reference des roquettes.
 * Elle retourne un entier contenant le nombre de lignes dans le
 * fichier.
 */
int cptligne(char* pnom);


 /* Fonction qui copie les informations contenues dans le fichier
  * de trajectoire de reference pour les roquettes dans un tableau
  * de structure pt_trajectoire.
  *
  * Elle prend comme arguments :
  * un pointeur sur le premier element du tableau de caracteres
  * contenant le nom du fichier de la trajectoire de reference,
  * un pointeur sur la premiere adresse du tableau de structure
  * pt_trajectoire.
  */
int initTrajRefRoq(char* nomfichiertrajref,
          pt_trajectoire* tableauTrajRefRoq);


/* Fonction d'initialisation de la position de l'avion . L'altitude,
 * la pente, le vitesse sont recuperees depuis le nom du fichier
 * contenant la trajectoire.
```

```
 *
 * Elle prend comme arguments :
 * un pointeur sur la premiere adresse du tableau
 * de structure pt_trajectoire contenant la trajectoire
 * de reference pour les roquettes,
 * un pointeur sur le premier element du tableau de caracteres
 * contenant le nom du chemin d'acces au fichier contenant les
 * valeurs de la trajectoire de reference,
 * un pointeur sur l'adresse de la structure pt_trajectoire
 * pour initialiser la position initiale du porteur au debut
 * du tir de la salve.
 */
int initPosiAvion(pt_trajectoire* TrajRefroq,char* nomfichiertrajRef,
          pt_trajectoire* posiAvion);


 /* Fonction qui retourne la position de la roquette centrale dans
  * la salve
  *
  * Elle prend comme arguments :
  * un entier dont la valeur est egale au nombre de roquettes
  * tirees par salve,
  * un pointeur sur l'adresse d'un entier dans laquelle elle
  * retourne la valeur de la position centrale.
  *
  * Regle de calcul avec un nombre pair. La fonction retourne
  * nombre/2 + 1. L'operation s'effectue sur des entiers.
  *
  */
int positioncentrale(int quteRoqparsalve, int* positionRoqCentral);


 /* Fonction qui calcul la trajectoire de la roquette en fonction
  * de sa position dans la salve et de l'indice de variation
  * aleatoire en sortie de tube.
  *
  * Elle prend comme arguments :
  * un entier contenant le rang de la roquette dans la salve,
  * un entier contenant le nombre de roquette tiree par seconde,
  * un pointeur sur l'adresse de la structure contenant la
  * position initiale de l'avion,
```

```
  * un reel contenant la valeur de dispersion de la roquette en
  * radian,
  * un entier contenant le nombre de ligne du tableau contenant
  * les valeurs de la trajectoire de reference pour les roquettes,
  * un pointeur sur l'adresse de la premiere structure du tableau
  * contenant la trajectoire de reference pour les roquettes,
  * un pointeur sur l'adresse de la premiere structure du tableau
  * recevant les valeurs de la trajectoire calculee,
  * un  pointeur sur un reel qui recoit la valeur de l'angle
  * en site apres calcul en fonction de la valeur de la
  * dispersion,
  * un pointeur sur un reel qui recoit la valeur de l'angle
  * en azimut apres calcul en fonction de la valeur de la
  * dispersion.
  */
int trajroq(int rangRoq, int cadenceTir, pt_trajectoire* posiavion,
        float sigma,int nbligne, pt_trajectoire* tabTrajRefRoq,
        pt_trajectoire* tabtrajroq,float* anglesite,
        float* angleazim);


 /* Fonction qui retourne dans une structure pt_trajectoire
  * les coordonnees et instant du point d'impact dans l'eau
  * de la roquette centrale ou d'un dard.
  *
  * Elle prend comme arguments :
  * un pointeur sur la premiere adresse du tableau de structure
  * pt_trajectoire contenant la trajectoire d'une roquette,
  * un entier contenant le nombre de ligne du tableau de structure
  * pt_trajectoire contenant la trajectoire d'une roquette,
  * un pointeur sur l'adresse d'une structure pt_trajectoire pour
  * retourner les valeurs du point d'impact pour une altitude egale
  * a zero.
  */
int impactDansEau(pt_trajectoire* trajroquette, int nbligne,
          pt_trajectoire* pointImpactEau);

 /* Fonction qui parcours la trajectoire de la roquette centrale
  * pour, en fonction de la distance de depotage, retourner
  * l'instant de depotage.
  *
```

```
  * Elle prend comme arguments :
  * un entier contenant la position de la roquette centrale,
  * un entier contenant le nombre de roquette tiree par seconde,
  * un entier contenant le nombre de ligne du tableau de structure
  * pt_trajectoire contenant la trajectoire d'une roquette,
  * un pointeur sur l'adresse d'une structure pt_trajectoire
  * contenant les valeurs du point d'impact pour une altitude
  * egale a zero.
  * un entier contenant la distance de depotage des flechettes sur
  * la trajectoire.
  * un pointeur sur la premiere adresse du tableau de structure
  * pt_trajectoire contenant la trajectoire d'une roquette,
  * un pointeur sur l'adresse d'une structure pt_trajectoire
  * pour retourner les valeurs du point de depotage.
  */
int retardDepotage(int positionRoqCentrale, int  cadenceTir,
          int nbreligne, pt_trajectoire* pointImpactEau,
          int distdepotage,pt_trajectoire* trajroquette,
          pt_trajectoire* ptDepotage);


 /* Fonction qui affiche les valeurs globales calculer lors des
  * operations precedentes.
  *
  * La fonction affiche :
  * l'instant d'impact de la roquette centrale dans l'eau
  * en seconde,
  * l'instant de depotage des flechettes en seconde,
  * la distance avion cible selon l'axe Ox en metre,
  * la pente de la flechette a l'impact en degres,
  * la vitesse de la flechette a l'impact en m/s.
  *
  *     Elle prend comme arguments :
  * un pointeur sur l'adresse d'une structure pt_trajectoire
  * contenant les valeurs du point de depotage,
  * un pointeur sur l'adresse d'une structure pt_trajectoire
  * pour retourner les valeurs du point d'impact de la roquette
  * centrale,
  * un pointeur sur l'adresse d'une structure pt_trajectoire
  * pour retourner les valeurs du point de d'impact de
  * la flechette centrale.
```

```
  *
  */
int affichageValeursGlobales(pt_trajectoire* pointImpact,
        pt_trajectoire* ptDepot,pt_trajectoire* pointimpdard);

  /* Fonction  de calcul de la trajectoire de reference des dards
   *
   * Elle prend comme arguments :
   *  un pointeur sur l'adresse d'une structure pt_trajectoire
   * contenant les valeurs du point de depotage,
   *  un entier contenant le pas de calcul de la trajectoire
   * de reference des flechettes,
   *  un pointeur sur l'adresse de la premiere structure
   * pt_trajectoire du tableau recevant les valeurs de
   * la trajectoire calculee.
   */
int initrajrefdard(pt_trajectoire* pointDepotage,
        int nbpastrajrefdard, pt_trajectoire* trajrefdards);


  /* Fonction de calcul de trajectoire des dards par modification
   * des coordonnees de la trajectoire de reference.
   *
   * Elle prend comme arguments :
   *  un pointeur sur l'adresse d'une structure
   * pt_trajectoire
   * contenant les valeurs du point de depotage,
   *  un pointeur sur l'adresse d'une structure
   * pt_trajectoire
   * contenant les valeurs du point de depotage,
   *  un reel contenant la valeur de la dispersion des
   * flechettes,
   *  un entier contenant le nombre de pas de calculs
   * de la trajectoire
   * de reference des flechettes,
   *  un reel contenant la dispersion de la roquette en
   * site,
   *  un reel contenant la dispersion de la roquette en
   * azimut,
   *  un entier contenant le rang de la roquette dans la
   * salve,
```

```
   *  un entier contenant le nombre de roquette tiree par
   * seconde,
   *  un pointeur sur un tableau de structures pt_trajectoire
   * pour recevoir les valeurs de chaque point de la nouvelle
   * trajectoire.
   */
int trajdard(pt_trajectoire* pointDepotage,
        pt_trajectoire* trajrefdards,float sigma,
        int nbpastrajrefdard,float dansiteroq,float danazimroq,
        int rangRoq,int cadenceTir, pt_trajectoire* trajdards);


 /* fonction qui initialise la structure et le tableau de comptage
  * des impacts de dard
  *
  * Elle prend comme arguments :
  * un pointeur sur une structure surface qui definie
  * la zone observee,
  * un pointeur sur une structure pt_trajectoire contenant
  * les valeurs du point d'impact theorique de la flechette,
  * un pointeur sur un tableau d'entier initialise a 0.
  *
  */
int initZoneObs(surface* zoneobs, pt_trajectoire* pointimptdard,
           int* tabzone);


 /* fonction qui initialise les attributs de la structure
  * gestionImpact
  *
  * Elle prend comme arguments :
  * un pointeur sur une structure gestionImpact a initialisee,
  * un pointeur sur la premiere adresse du tableau
  * d'entiers qui les impacts sur le bateau par salve,
  * un entier qui contient le nombre de salve.
  */
int initcptimpats(gestionImpact* cptimpact,int* tabimpbat,
           int nbsalve);


 /* fonction qui initialise l'instant de depotage des dards
```

```
   * en fonction de l'indice de variation aleatoire
   *
   * Elle prend comme arguments :
   * un entier dont la valeur est la position de la roquette
   * dans la salve,
   * un entier dont la valeur est le nombre de roquettes tiree
   * par seconde,
   * un pointeur sur une structure pt_trajectoire qui contient
   * les valeurs du points de depotage.
   */
int tirageSigmaRetard(int idroq,int cadenceTir,
              pt_trajectoire* pointDepotage);

 /* fonction d'affichage des valeurs statistiques en fin de
   * calcul.
   *
   * Elle prend comme arguments :
   * un pointeur sur une structure gestionImpact,
   * un entier qui contient le nombre de salve,
   * un pointeur sur l'adresse du premier element d'un
   * tableau d'entier contenant le nombre d'impact par
   * roquette,
   * un entier qui contient le nombre de roquette par salve.
   */
int affichageValeursStat(gestionImpact* cptimpact, int nbsalve,
              int* tabimpParroq, int nbroq);

 /* fonction d'initialisation de la tructure bateau
   *
   * Elle prend comme arguments :
   * un pointeur sur une structure bateau a initialisee,
   * un pointeur sur une structure pt_trajectoire qui
   * contient les valeurs du point d'impact dans l'eau
   * de la roquette centrale de la salve,
   * un pointeur sur une structure pt_trajectoire qui
   * contient les valeurs du point d'impact du dard central
   * theorique,
   * un pointeur sur une structure I_point2d qui contient
   * les valeurs de choix de correction de tir.
   */
int initposiBateau(bateau* cible,pt_trajectoire* pointImptEau,
```

```
            pt_trajectoire* pointimptdard,I_point2d* justess);

 /* fonction de mise de la structure gestionImpact
  *
  * Elle prend comme arguments :
  * un pointeur sur une structure pt_trajectoire qui
  * contient les valeurs du point d'impact du dard central
  * theorique,
  * un pointeur sur une structure bateau,
  * un pointeur sur une structure surface qui definie
  * la zone observee,
  * un pointeur sur la premiere adresse d'un tableau
  * d'entier contenant les impacts par unite de surface,
  * tableau d'entier contenant le nombre d'impact par
  * roquette,
  * un pointeur sur une structure gestionImpact.
  */
int comptageImpact(pt_trajectoire* pointimpactdard, bateau* cible,
            surface* zoneobs ,int* tabzone,int* tabimpParRoq,
            gestionImpact* cptimpact);


 /* fonction de gestion de comptage des salves satisfaisant
  * la performance
  *
  * Elle prend comme arguments :
  * un pointeur sur une structure gestionImpact,
  * un entier contenant le nombre de flechettes necessaires
  * pour neutraliser la cible.
  */
int comptageSalveOkPerf(gestionImpact* cptimpact,int perfo);


 /* fonction de conversion d'un nombre de seconde en jours, heures,
  * minutes, secondes.
  *
  * Elle prend comme arguments :
  * un reel contenant le nombre de seconde ecoulees pendant
  *     le calcul,
  * un pointeur sur un entier qui contiendra le nombre de jours,
  * un pointeur sur un entier qui contiendra le nombre d'heures,
```

```
  * un pointeur sur un entier qui contiendra le nombre de minutes,
  * un pointeur sur un entier qui contiendra le nombre de secondes,
  */

int conversionHoraire(double secEntree, int* jours ,int* heures,
            int* minutes, int* secondes);

 /* fonction qui ecrit les fichiers densiteDards.txt et
  * densiteDards.sci
  *
  * Elle prend comme arguments :
  * un pointeur sur la premiere adresse d'un tableau contenant les
  * impacts par unite de surface,
  * un pointeur sur une strucutre surface,
  * un pointeur sur une structure gestionImpact,
  * un pointeur sur une structure pt_trajectoire contenant les
  * valeurs du point d'impact de la roquette centrale,
  * un pointeur sur une structure bateau contenant les valeurs
  * definissant la cible,
  * un entier contenant le nombre de salves.
```

Exemple d'utilisation des fonctions définies

Le but de cet exemple est de vérifier les choix de découpage en fonction pour la réutilisation de celle-ci lors d'évolutions ou de nouveaux programmes.

Nous allons à l'aide de ces fonctions réaliser une évolution de notre programme en ajoutant la possibilité de calculer la répartition des impacts en connaissant les valeurs du point de dépotage. Les informations connues sont les coordonnées du point de dépotage, les valeurs des projections du vecteur vitesse. Nous ajoutons afin de pouvoir réaliser le calcul la justesse de tir et la dimension de la zone observée. Le résultat retourné par le programme est un fichier formaté pour être directement traité par scilab pour obtenir une représentation graphique de la répartition.

Nous allons ordonner dans une liste les actions à réaliser pour effectuer le calcul.

1. saisir les valeurs,
2. initialiser la trajectoire de référence des dards
3. calculer la trajectoire d'un dard de référence pour centrer la cible et la zone d'observation,
4. calculer le point d'impact,
5. afficher les valeurs globales,
6. initialiser la zone d'observation

7. pour chaqued'itération
   - pour chaque dard
     - calculer leur trajectoire,
     - calculer le point d'impact,
     - enregistrer les impacts,
8. sauvegarder les données
9. afficher les valeurs statistique

Pour réaliser cette suite d'action nous allons réutiliser plusieurs des fonctions définies, nous avons seulement à écrire deux fonctions particulières ; la fonction de saisie des valeurs et la fonction d'affichage des valeurs globales qui sont toutes les deux spécifiques au besoin. Les autres fonctions seront utilisées sans changement.

liste des fonctions utilisées

```
int saisievaleursPtDep(pt_trajectoire* ptdp, bateau* cib,
                       surface* s, int* it, I_point2d* justess);

int initrajrefdard(pt_trajectoire* pointDepotage,
                   int nbpastrajrefdard,
                   pt_trajectoire* trajrefdards);

int trajdard(pt_trajectoire* pointDepotage,
                         pt_trajectoire* trajrefdards,float sigma,
                         int nbpastrajrefdard, float dansiteroq,
                         float danazimroq, int rangRoq,
             int cadenceTir, pt_trajectoire* trajdards);

int impactDansEau(pt_trajectoire* trajroquette,int nbligne,
                  pt_trajectoire* pointImpactEau);

int affichageValeursGlobalesDards(pt_trajectoire* ptDepot,
                                  pt_trajectoire* pointimpdard);

int initZoneObs(surface* zoneobs, pt_trajectoire* pointimptdard,
                                  int* tabzone);

int initcptimpacts(gestionImpact* cptimpact,int* tabimpbat,
                                        int nbsalve);
```

```
int comptageImpact(pt_trajectoire* pointimpactdard, bateau* cible,
                   surface* zoneobs ,int* tabzone,int* tabimpParRoq,
                   gestionImpact* cptimpact);

int sauvegardeDonnees(int* zone, surface* zoneobs,
                      gestionImpact* cptimpact,
                      pt_trajectoire* ptimpactroq,
                      bateau* cible, int nbsalve);

int barreProgression(int idsalve, int nbresalve,int* resold);

int initposiBateau(bateau* cible,pt_trajectoire* pointImptEau,
                   pt_trajectoire* pointimptdard,I_point2d* justess);
```

Cet exemple est une première étape. On remarque que certain choix de nom pour les fonctions devront être revus avant de les intégrer dans une bibliothèque, que l'utilisation de ces fonctions réduit la taille du code, pour exemple le code de cette fonction est de 156 lignes, est plus clair et concis.

```
int calculimpactPtDep() {

    /*  variable pour le calcul du temps d'execution */

 time_t depart;
 time_t fin;


    /* saisie des valeurs */

 pt_trajectoire ptDepotage;
 surface surfObservee;
 bateau cible;
 int nbreIteration;
 I_point2d justesse;

   saisievaleursPtDep(&ptDepotage, &cible,  &surfObservee,
                      &nbreIteration, &justesse);

    time(&depart);
 /* calcul trajectoire ref dard */
```

```
int nbrepas;
nbrepas = 500;
pt_trajectoire trajrefdards[nbrepas];
float sigmadard = 0.022;
pt_trajectoire trajdards[nbrepas];
float anglesiteroq = 0.0;
float angleazimroq = 0.0;
int idroq = 1;
int cadenceTir = 30;

 initrajrefdard(&ptDepotage, nbrepas, &trajrefdards[0]);

 /* calcul trajectoire dard */

 pt_trajectoire pointimpactdard;

 trajdard(&ptDepotage, &trajrefdards[0], 0.0,
                      nbrepas, anglesiteroq, angleazimroq, idroq,
          cadenceTir, &trajdards[0]);

 impactDansEau(&trajdards[0], nbrepas, &pointimpactdard);

 pt_trajectoire* pointImpactEauC = &pointimpactdard;

 affichageValeursGlobalesDards(&ptDepotage, pointImpactEauC);

 /* init zone observee */

 int zone[surfObservee.longueur][surfObservee.largeur];


 initZoneObs(&surfObservee, &pointimpactdard, &zone[0][0]);
 initposiBateau(&cible, pointImpactEauC, &pointimpactdard,
               &justesse);


  /*    initialisation compteurs impacts          */

  int quteRoqparSalve =1;
  gestionImpact cptimpact;
  int nbreImpactSurBatParRoq[quteRoqparSalve];
```

```
 initcptimpacts(&cptimpact, &nbreImpactSurBatParRoq[0],
                qutеRoqparSalve);


/* boucle de comptage */
int index;
int dardcourant;
int res;
for(index = 0; index < nbreIteration; index ++){

barreProgression(index, nbreIteration, &res);

  for(dardcourant = 0; dardcourant < 36; dardcourant++) {

/* calcul trajectoire */

  trajdard(&ptDepotage, &trajrefdards[0], sigmadard ,
           nbrepas, anglesiteroq, angleazimroq, idroq,
           cadenceTir, &trajdards[0]);

  impactDansEau(&trajdards[0], nbrepas, &pointimpactdard);

/* enregistrement des impacts */

 comptageImpact(&pointimpactdard, &cible,
                                  &surfObservee , &zone[0][0],
                &nbreImpactSurBatParRoq[idroq], &cptimpact);

 }/* fin boucle dards */
} /* fin boucle iteration*/

/* post traitement du fichier impacts */

   sauvegardeDonnees(&zone[0][0], &surfObservee, &cptimpact,
                pointImpactEauC, &cible, nbreIteration);
  affichageValeursStat(&cptimpact, nbreIteration,
              &nbreImpactSurBatParRoq[0], qutеRoqparSalve);

time(&fin);
```

```
            /* affichage de temps de calcul */

  double delai = difftime(fin,depart);
/*   printf("temps de calcul : %f \n",delai); */
  int j = 0;
  int h = 0;
  int m = 0;
  int s = 0;
  conversionHoraire(delai, &j , &h,  &m, &s);
  printf("\ntemps de calcul : %d j %d h %d m  %d s \n\n",j,h,m,s);

  system("PAUSE");
  return 0;
  }
```

7.1.4 Réalisation du codage

A partir des éléments contenus dans le document de conception détaillé, le développeur réalise le codage du logiciel en respectant les contraintes et exigences définies. Il est conseillé de suivre le guide de bonnes pratiques [THA17] lors de la réalisation du codage. L'utilisation de ce guide n'est pas obligatoire cependant son utilisation garantit le respect de règles qui faciliteront dans l'avenir la maintenance du code. Les règles sont simples et précises et couvrent la structure du fichier source, ainsi que la totalité des aspects du langage.

Lors de cette étape le développeur doit aussi faire la mise au point de ses sous-ensembles en prévision des tests unitaires.

Dans ce projet a été utilisé un pré compilateur de programmation lettrée (voir annexe CHAP 12). Cette méthode permet de concevoir la rédaction de la documentation dans le même document que le code. Il est possible de faire apparaître ou non le code dans la documentation. Pour illustrer ce propos voici un extrait du script de création de la fonction d'interpolation linéaire. Dans ce projet cette méthode a été surtout utilisée pour réaliser la documentation et le codage des sous-ensembles, la construction du programme a été réalisée dans un environnement intégré comme le propose le logiciel DEV-C/C++.

entête du script

```
@!page type pour la cr\'eation d'un programme avec funnelweb

@! date: 27 02 2009
@! fichier : interpolationlineaire.fw
```

```
@p typesetter = tex
```

⋮

une partie documentation

```
\input{TDA}
\usepackage{amsmath}

\begin{document}

\section{r\'esum\'e}

Cette fonction calcul la position en x d'un point d'une droite
connaissant deux points de la droite et la valeur de y correspondante.

\section{acronymes}
\section{pr\'esentation des algorithmes}
\subsection{hypoth\`eses}
\subsection{notation}
%
\begin{itemize}
\item [y] :coordonn\'ee du point sur l'axe des ordonn\'ees
\item [x] :coordonn\'ee du point sur l'axe des absices.
\item [a] :pente de la droite
\item [b] :ordonn\'ee \`a l'origine.
\item [$x_1$],$y_1$,$x_2$,$y_2$ : deux points appartenant \`a la droite.
\end{itemize}
%
\subsection{algortihmes}
%
\begin{equation}
y = a \cdot x + b
\end{equation}
```

⋮

une partie écritrure des sources

```
\end{document}
@!%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
@!% section de cr\'eation des programmes
@!%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

@o@<interpolation.sci@>==@{@-
function x = interpolation(x1,y1,x2,y2,y)
map = 0
if x2 == x1 then
  x = x1;
  elseif y2 == y1 then
   x = %inf;
   else
    a = (y2-y1)/(x2-x1)
    b = y2 - a*x2
    x = (y - b) / a
    if map == 1 then
     printf("y = %fx + %f\n",a,b)
     printf("avec x = %f\n",x)
     printf("y = %f\n",a*x+b)
    end
end;
endfunction
@}

@o@<interpollineaire.c@>==@{@-

int interpollineaire(float x1, float y1, float x2, float y2, float y, float* x)
{
 float a;
 float b;
```

⋮

Nous pouvons remarquer que dans le dernier extrait de ce script la présence de la ligne

```
 \end{document}
```

indique que les codes sources des deux fonctions écrites l'une pour Scilab , l'autre en C ne sont pas visible dans la documentation.

Il faut noter que ces quelques extraits sont loin de traiter de façon exhaustive toutes les possibilités de la programmation lettrée. Que cette méthode de rédaction

vient compléter l'offre dans ce domaine et y apporte une solution différente par sa possibilité de produire une documentation de qualité imprimable à l'inverse des solutions habituellement utilisées(Oxygen, JAVADOC) dont la finalité est de produire plutôt une documentation en ligne au format HTML.

7.1.5 Les tests

La liste des tests est regroupée dans un document appelé « Description des tests de qualification du logiciel (DTL) ». Il est possible lors de projet important de réaliser un "DTL" pour le produit logiciel principal et des "DTL" pour certains sous-ensembles.

Dans un "DTL" nous retrouverons pour un produit logiciel les listes de tests :
- pour les interfaces homme-machine
- pour chaque composant existant intégré dans le produit
- pour le programme principal.

Différents types de tests seront définis ; unitaire, aux limites, d'ergonomie, d'intégration... L'ensemble de ces tests devra répondre aux exigences énoncées dans le document de définition des exigences.

Les résultats des ces tests sont notés dans un rapport de tests. En cas d'échec lors de la batterie de tests un arbitrage est organisé afin de décider des actions correctives à apporter au composant concerné.

A la suite de la réussite des tests le produit logiciel peut être mis en production.

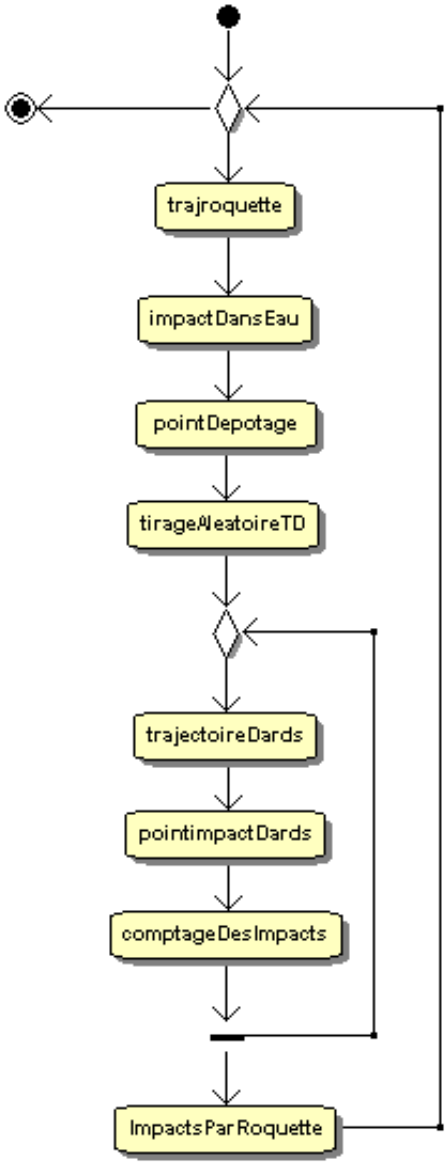

FIGURE 7.3 – diagramme de transition d'état du sous-ensemble calcul impact

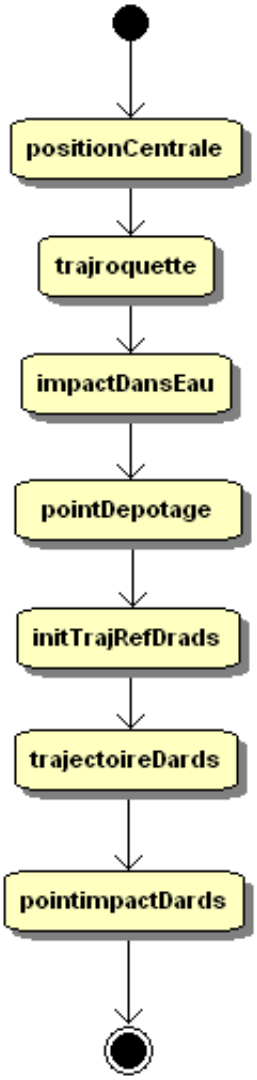

FIGURE 7.4 – diagramme de transition d'état du sous-ensemble roquette centrale

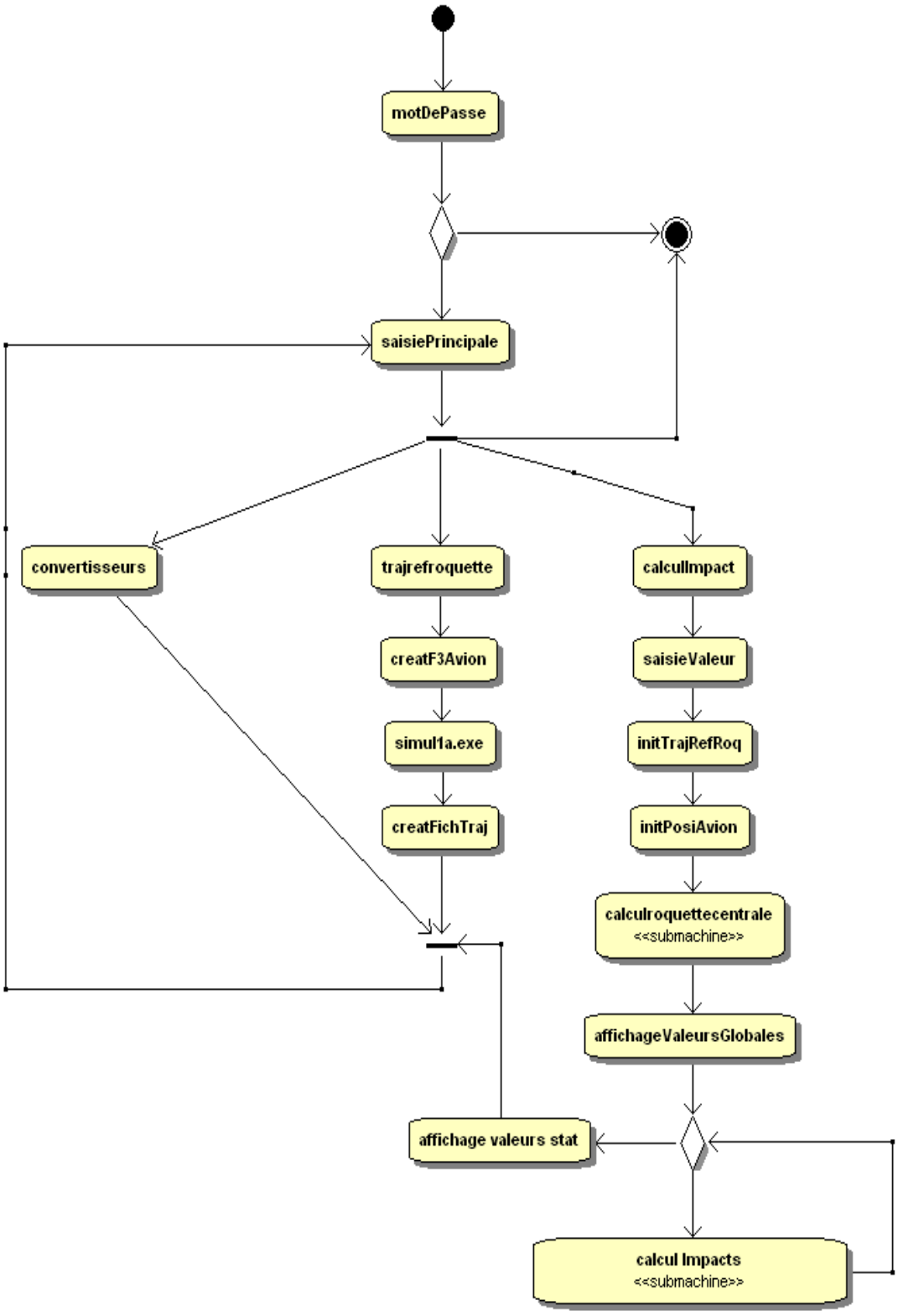

FIGURE 7.5 – diagramme de transition d'état de Calmoddards

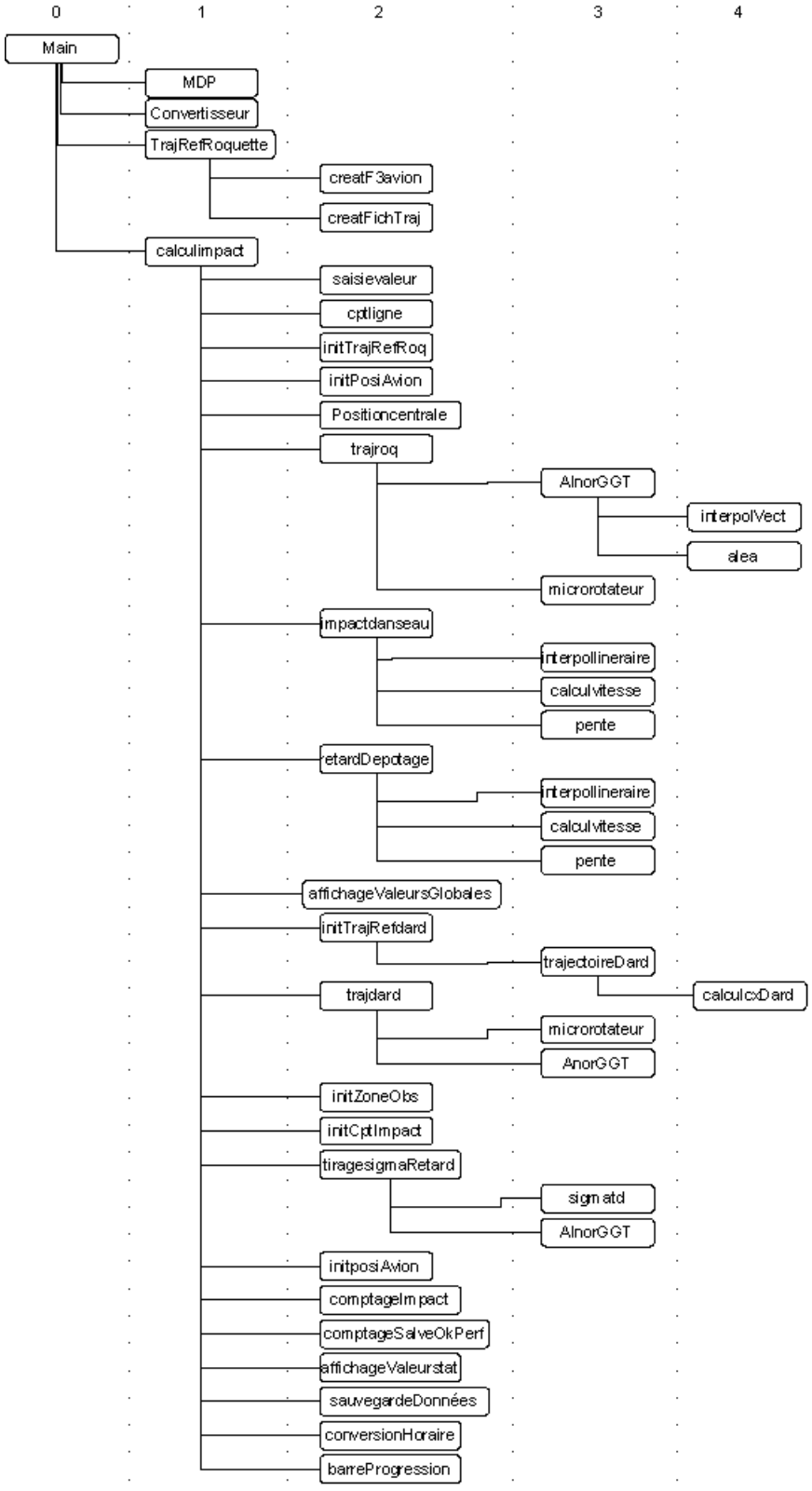

FIGURE 7.6 – structure arborescente des appels de fonctions

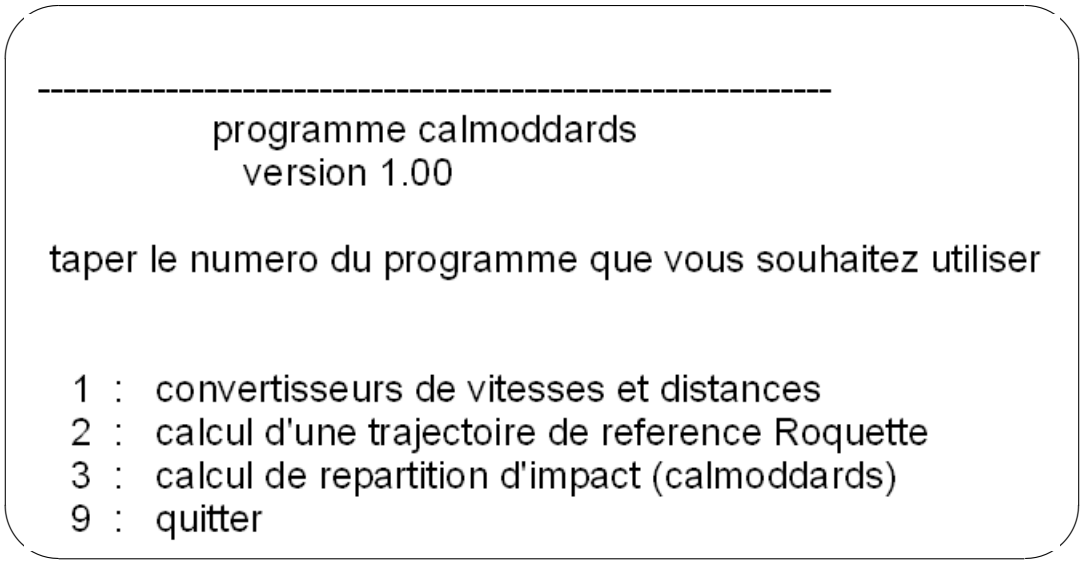

FIGURE 7.7 – aspect de l'écran principal

```
-------------------------------------------------

        saisie des valeurs pour le calcul
            des impacts de dards



  nom du fichier de trajectoire de reference roquette
I:\SimulateurDeModeleEfficacite\tiravion3000\codes\nouvCalmod
dards\Traj_4523_-30_250.txt

 nombre de pas d'iteration pour le monte-carlo(nombre entier)
30000

 dispersion de la roquette en sortie de lanceur(en radian)
.004

 quantite de paire de roquettes par salve
12

 nombre de dards pour detruire le bateau
5

 distance de depotage
450

 justesse de tir
 format de saisie : x y
10 10

 dimension et vitesse du bateau
 format de saisie : x y v
10 3 30

 dimension de la zone observee
 format de saisie : Lon lar
400 200
```

FIGURE 7.8 – aspect de l'écran principal de saisie de la fonction calculimpact

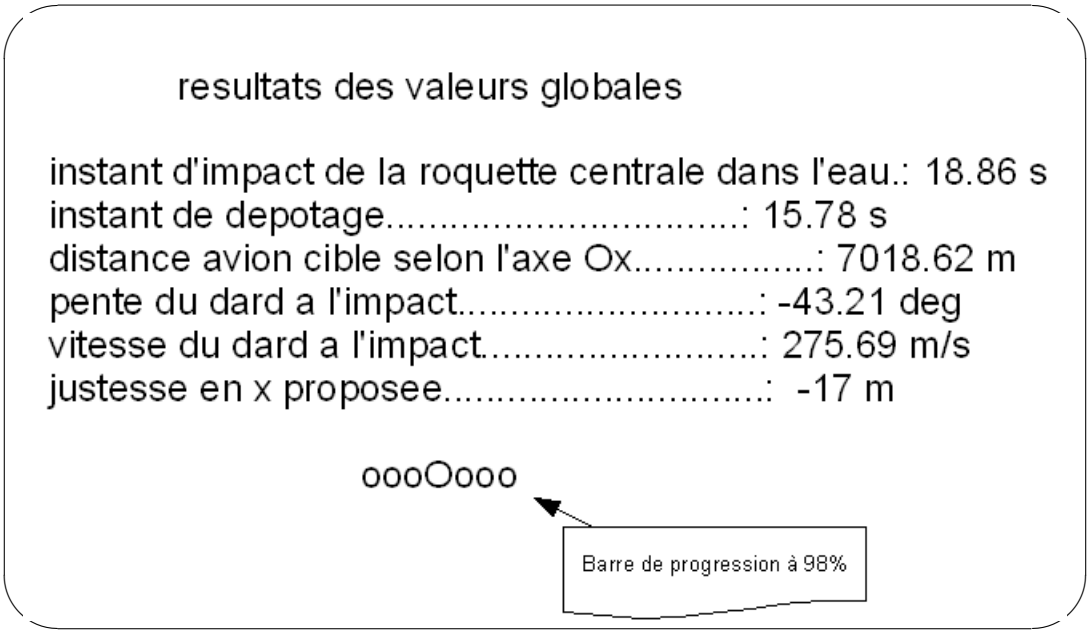

FIGURE 7.9 – aspect de l'écran de résultats intermédiaires de la fonction calculimpact

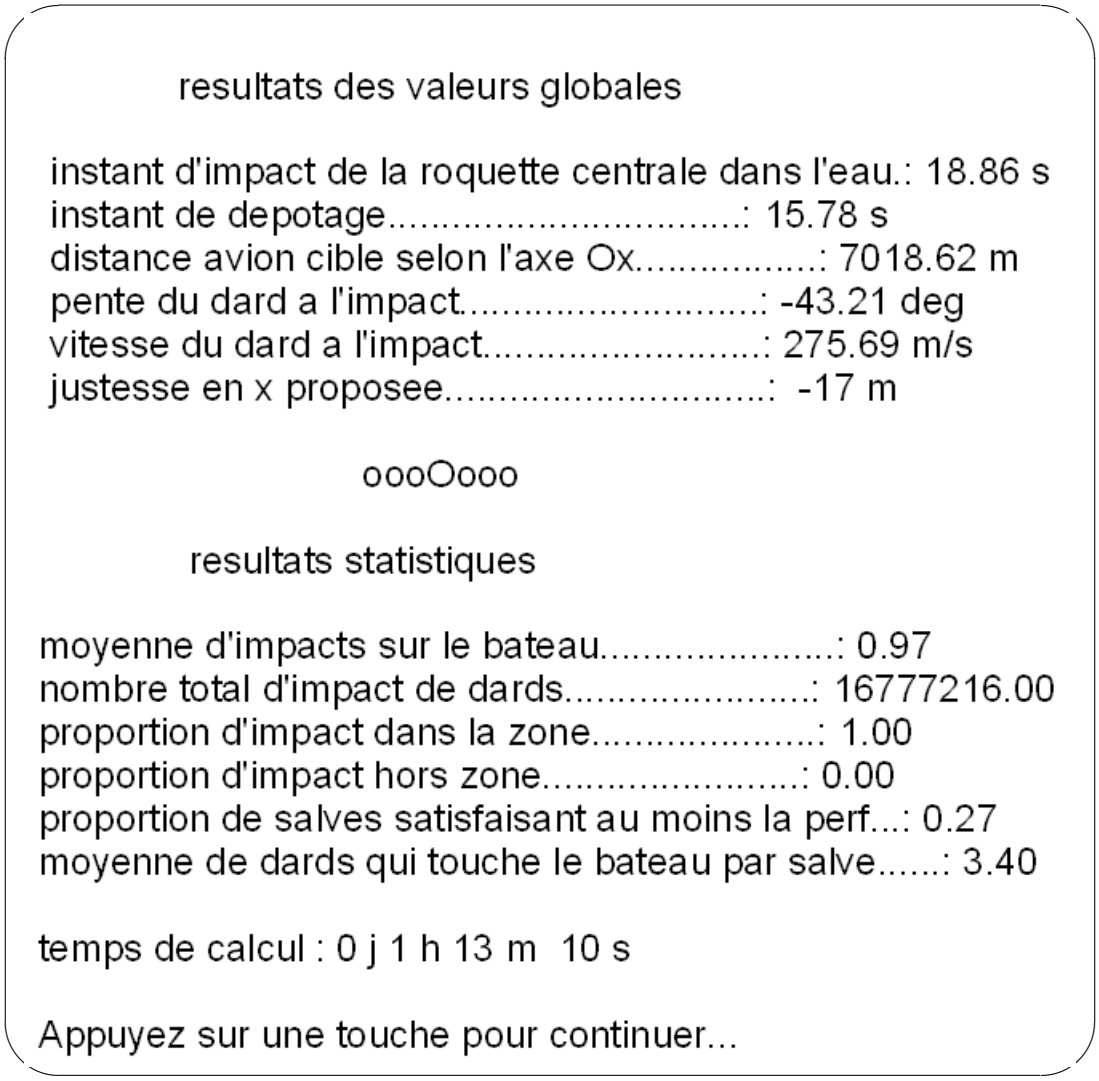

FIGURE 7.10 – aspect de l'écran de résultats finaux de la fonction calculimpact

Chapitre 8

Résultats et Discussion

8.1 Etudes des variations entre les programmes retournant des probabilités

Nous avons vu dans le chapitre 2 qu'il est possible de mesurer la probabilité d'obtenir un nombre d'impact sur une unité de surface de façon statistique. Lors de cette première estimation il n'est absolument pas fait référence aux trajectoires de mobiles propulsés ou non.

Des écarts sur la définition des paramètres d'entrée du calcul par rapport aux définitions des paramètres que nous utiliserons lors de la réalisation du programme écrit en C sont à noter.

- pour ce calcul la dispersion des roquettes sur un axe est directement traduite en une distance en mètre.
- le rayon de la gerbe de dards est connu, par la suite cette grandeur ne sera plus exprimée.

Comment faire le lien entre cette probabilité et les valeurs données lors de nos calculs ?

Définition de l'algorithme de calcul du rayon de la gerbe de dards

Pour réaliser ce calcul nous possédons déjà un certain nombre d'informations, mais une n'apparaît pas dans les résultats. Il manque la pente de la roquette à l'instant de l'éjection des dards. La valeur qui est retenue est celle calculée sur la trajectoire de la roquette centrale, elle est utilisée pour calculer le point d'impact, l'angle et la vitesse du dard.

Il va donc falloir modifier le programme de calcul de simulation pour qu'il affiche cette information. Puis à l'aide de l'algorithme ci-dessous calculer le rayon correspondant.

notation

- angd angle de la roquette à l'instant de l'éjection exprimé en degrés
- α angle de la roquette à l'instant de l'éjection exprimé en radians
- σ dispersion des dards exprimée en radians
- $\phi = \alpha - \sigma$
- $\beta = \alpha + \sigma$
- $avcix$ distance avion cible selon l'axe Ox
- $disde$ distance de dépotage
- $disdz$ projection du point de dépotage sur l'axe z
- $rayon$ rayon de la gerbe

algorithme

$$\begin{aligned} \alpha &= angd \cdot \frac{\pi}{180} \\ disdz &= disde \cdot \cos\alpha \\ \phi &= \alpha - \sigma \\ \beta &= \alpha + \sigma \\ rayon &= \frac{disdz \cdot (\tan\beta - \tan\phi)}{2} \end{aligned}$$

De plus on regarde le résultat donné par le programme "modcalprob" qui calcule la probabilité de réussite d'une salve par unité de surface selon la méthode de Monte-Carlo. Le calcul étant relativement long après quelques essais on constate une symétrie horizontale qui va permettre de ne calculer que la moitié de la surface observée.

Le tableau nous donne les probabilités respectives des deux programmes.

	offsetx	offsety	prog stat(valeurs)	modcalprob(plage de valeurs)
1	0	0	1.0	0.8->1
2	12	0	0.8	0.8->1
3	12	10	0.32	0.4->.6
4	12	12	0.15	0.4->0.6
5	12	15	0.02	0.2->0.4
6	0	12	0.8	0.6->0.8
7	15	12	0.02	0.4->0.6

TABLE 8.1 – valeurs de probabilités de toucher une cible

Les résultats nous montrent que plus la précision du tir de roquette est grande plus il est important de tirer juste ; un tir décalé de 12 mètres en longueur et 12 mètres de coté fait chuter la prévision de 85%, alors que le même calcul avec une dispersion deux fois plus grande ne fait chuter la prévision que de moitié.

Remarque : L'image de la répartition des dards au sol est ovoïdece qui a pour conséquence que l'erreur de tir latérale influence plus le résultat que l'erreur longitudinale. Ce phénomène explique les différences de prévision que l'on retrouve en lignes 3 et 4 du tableau. Le fait est que le programme statistique considère la zone d'impact comme rectangulaire et non ovoïde.

Cependant le programme utilisant les statistiques est très rapide au regard du temps de calcul avec la méthode de Monte-Carlo ce qui en fait un très bon outil de première estimation.

Le temps de calcul avec la méthode de Monte-Carlo étant trés long, nous avons regardé la possibilité de remplacer ce calcul par une estimation faite par la loi de Poisson. Les essais montrent des variations entre les résultats de l'ordre de 10% ce qui a amené la décision d'abandonner cette solution.

8.2 variation entre les différentes versions du programme de calcul d'impacts

Lors de la réalisation de la maquette 8 versions différentes on été écrites. Pour chaque version une ou plusieurs modifications ont été apportées ; parmi les plus significatives on notera :

- Le remplacement du temps de dépotage par la distance de dépotage. Ce remplacement a modifié la perception de cette grandeur qu'est le retard ce essentiellement lors des contrôles des versions du logiciel . Les valeurs utilisées avant le changement étaient des valeurs entières par exemple 14 secondes ce qui limitait l'exploration de la plage des valeurs possibles. Après le changement ce constat était valable pour les distances. Une attention particulière a été portée lors des contrôles suivant sur le choix des valeurs employées. Ce changement a permis aussi de mettre en relation la distance et le temps. Au début les temps utilisés correspondaient à des distances peu vraisemblables (très près de la cible) le fait d'utiliser une distance a corrigé ce défaut.
- l'ajout du calcul de la trajectoire de référence des dards en fonction de la trajectoire roquette. Ce dernier changement a été motivé par la recherche d'informations plus précises sur les dards ; la vitesse et la pente à l'impact.
- l'évolution de l'affichage des informations tel que : la suppression de la répartition des impacts par roquette. Cette information nous a permis de constater l'équiprobabilité des répartitions des impacts sur la cible par roquette.

N'étant plus utile par la suite la décision a été prise de ne plus afficher cette information.

Après chacun des contrôles de non-régression lors des changements de version on remarque de petites différences dans les résultats.

- Parmi Les plus significatifs il y a la vitesse du dard et sa pente à l'impact selon le type de trajectoire de référence roquette utilisée. Dans les premières versions ces changements étaient moins visibles. Ces informations ont bougé lors de la mise en place du calcul de trajectoire de référence des dards.
- La distance avion cible a augmenté.
- La quantité moyenne d'impacts sur la cible a baissé

Chapitre 9

Idées d'évolution

Il reste encore beaucoup de cas a étudier, nous n'avons pas encore abordé l'étude du calcul d'efficacité. Pour ce faire il est nécessaire d'étudié la possibilité d'ajouter un programme de calcul d'air d'efficacité pour une munition à éclats avec dans un premier cycle une fusée déclenchant la mise à feu au contact avec le sol puis avec une fusée à mise à feu programmable ; Dans le premier cycle on ajoute un module de calcul de projection d'éclat puis dans le deuxième on utilise les méthodes de calcul de trajectoire et de retard pour placer cet effet dans l'espace et en étudier le résultat et de définir les fonctions qui devront être aux modules de calcul d'effets terminaux et d'analyse du scénario.

Sous un aspect plus technique il serait intéressant, de regarder la génération de nombres aléatoires en ajoutant une fonction de création de graines différentes pour chaque appel de la fonction rand(). Cette évolution oblige à une maîtrise de la génération de graine afin de conserver la reproductibilité des résultats.

Chapitre 10

Conclusion

Pendant la réalisation de ce projet nous avons pu constater la difficulté à estimer la qualité d'un algorithme de génération de code aléatoire par des tests élaborés pour leur utilisation en cryptographie. Cependant quelques observations nous ont permis de mettre en évidence la capacité du générateur de la fonction rand(), contenue dans la bibliothèque standard du C, à générer des nombres pseudo-aléatoires suivant une loi uniforme.

Nous avons pu analyser et programmer un algorithme mettant en oeuvre la loi binomiale, constater sa rapidité d'exécution, au regard du temps nécessaire pour obtenir un résultat équivalent, en l'occurrence plus précis, avec une méthode de Monte-Carlo.

Nous avons aussi pu constater que la robustesse d'un mot de passe est toute relative et qu'elle dépend pour l'essentiel de la variété et du nombre des caractères qui le compose.

Nous avons constaté aussi que donner un résultat n'est pas suffisant il faut l'accompagner d'un minimum d'explication pour permettre son interprétation et donner son intervalle de validité.

Chapitre 11

Annexe A

Résumé

Dans cette étude il va être regardé sur un échantillon de 1 million de nombres pseudo-aléatoires générés par la fonction rand , les valeurs minimum et maximum retournées, la présence de répétitions, la fréquence et le cycle d'apparition de la plus grande et de la plus petite des répétitions, le classement des nombres par nombre de répétitions et l'on regardera si le générateur suit une loi uniforme.

La fonction rand() étudiée provient de la bibliothèque standard qui accompagne le compilateur C/C++ GCC 3.2 (mingw special 20020817-1). La bibliothèque est contenue dans le package MingW32 revision : 1.15. L'ensemble des résultats de cette étude sont valables uniquement pour la bibliothèque citée. Aucune vérification n'a été faite pour regarder la cohérence des résultats avec une autre bibliothèque standard distribuée avec un autre compilateur ou même avec une version différente de ce compilateur.

Présentation

Pendant l'éxécution d'un calcul selon la méthode de Monte-Carlo l'algorithme appelle 880.000 fois la fonction rand(). Le souhait est d'avoir une suite de nombres suivant une loi uniforme.

La documentaton de la fonction rand() indique qu'elle retourne un entier dans le domaine [0...32767]. Cela laisse à supposer que lors d'un nombre de tirages plus grand que 32767 il va se créer des répétitions.

Ne sachant pas comment la fonction distribue ses valeurs. Il a été réalisé un fichier contenant 1 million de valeurs générées avec la graine par défaut égale à 1.

Dans ce fichier vont être recherchés ;

– la génération de répétition en fonction du nombre de tirages.

– la valeur minimum générée
– la valeur maximum générée
– le nombre maximum de répétitions
– le nombre minimum de répétitions
– Le classement des nombres par le nombre de répétitions.

Résultats

La génération de répétitions en fonction du nombre de tirages

le but est de savoir à partir de combien d'itérations il y a génération de doublons.

Les valeurs retenues pour le nombre d'itérations sont des valeurs particulières prises arbitrairement.

nombre iterations	nbre total de répétitions
296	0
888	10
1776	43
4440	290
8880	1088
13320	2340
17760	4006
22200	6088
26640	8422
35520	13800
44400	20055
66600	38153
88800	58285
133200	100996
222000	189277

TABLE 11.1 – nombres de répétitions en fonction du nombre d'itérations

la valeur minimum générée

la valeur minimum est 0. Elle apparait 33 fois parmi un million de tirages.
les positions dans le fichier sont :

25199.	74833.	132068.	157073.
157427.	185619.	197379.	232558.
239165.	310666.	314608.	333659.
415662.	469058.	490225.	500845.
501895.	528141.	584506.	587510.
610674.	647046.	657051.	662863.
720140.	738050.	812883.	861009.
884091.	916572.	941522.	943440.
971489.			

les intervalles entre les valeurs sont :

49634.	57235.	25005.	354.
28192.	11760.	35179.	6607.
71501.	3942.	19051.	82003.
53396.	21167.	10620.	1050.
26246.	56365.	3004.	23164.
36372.	10005.	5812.	57277.
17910.	74833.	48126.	23082.
32481.	24950.	1918.	28049.

la valeur maximum générée

La valeur maximum générée est 32767. Elle apparait 35 fois parmi un million de tirages.

les positions dans le fichier sont :

46064.	88058.	88061.	108140.	175438.
196075.	209976.	225674.	239344.	251389.
276291.	279509.	305073.	315196.	352978.
354961.	401383.	406274.	419510.	444156.
487094.	537250.	547156.	573367.	604192.
650428.	677871.	759051.	773765.	794904.
822885.	867418.	910655.	925073.	950590.

Les intervalles entre les valeurs sont :

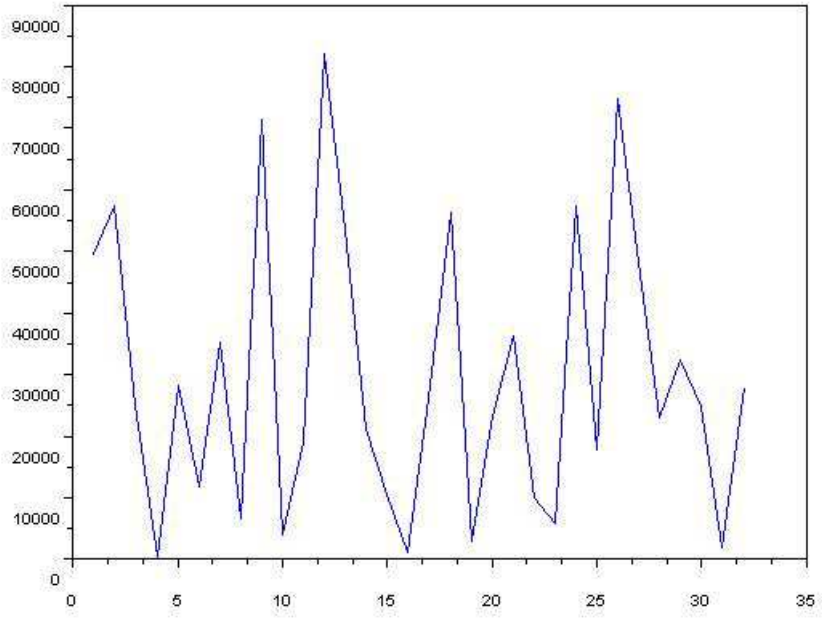

FIGURE 11.1 – intervalles entre les apparitions de la valeur 0

41994.	3.	20079.	67298.
20637.	13901.	15698.	13670.
12045.	24902.	3218.	25564.
10123.	37782.	1983.	46422.
4891.	13236.	24646.	42938.
50156.	9906.	26211.	30825.
46236.	27443.	81180.	14714.
21139.	27981.	44533.	43237.
14418.	25517.		

le nombre maximum de répétitions

Le nombre qui apparait le plus grand nombre de fois est 20335 . Il apparait 53 fois.

les positions dans le fichier sont :

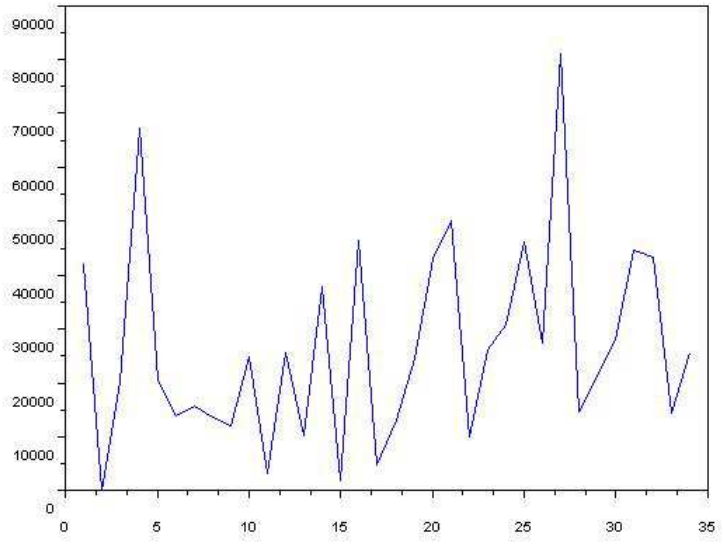

FIGURE 11.2 – intervalles entre les apparitions de la valeur 32767

5439.	61364.	69144.	98427.	125605.	142034.
165516.	169174.	269099.	274402.	278442.	281915.
286793.	332379.	354148.	377698.	426244.	429616.
433991.	442456.	459096.	486990.	519663.	548851.
560568.	564903.	583736.	593371.	595992.	598995.
605925.	613268.	613516.	642593.	650337.	657181.
658694.	680543.	683371.	688612.	695007.	701134.
723182.	786568.	801122.	827812.	832842.	878516.
910945.	955318.	961667.	986875.		

Les intervalles entre les valeurs sont :

55925.	7780.	29283.	27178.	16429.	23482.
3658.	99925.	5303.	4040.	3473.	4878.
45586.	21769.	23550.	48546.	3372.	4375.
8465.	16640.	27894.	32673.	29188.	11717.
4335.	18833.	9635.	2621.	3003.	6930.
7343.	248.	29077.	7744.	6844.	1513.
21849.	2828.	5241.	6395.	6127.	22048.
63386.	14554.	26690.	5030.	45674.	32429.
44373.	6349.	1680.	23528.		

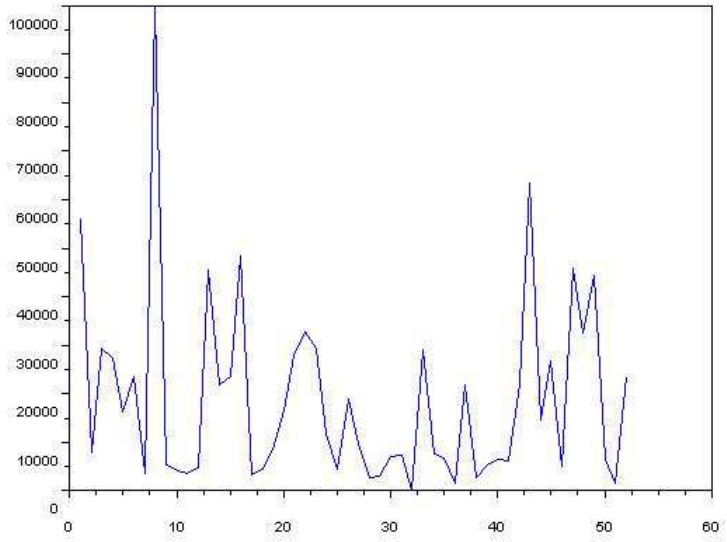

FIGURE 11.3 – intervalles entre les apparitions de la valeur 20335

le nombre minimum de répétitions

Le nombre qui apparait le moins grand nombre de fois est 28856 . Il apparait 10 fois.

les positions dans le fichier sont :

3975.	57041.	128820.	375112.	549757.
605696.	605888.	810751.	832570.	955787.

Les intervalles entre les valeurs sont :

53066.	71779.	246292.	174645.	55939.
192.	204863.	21819.	123217	

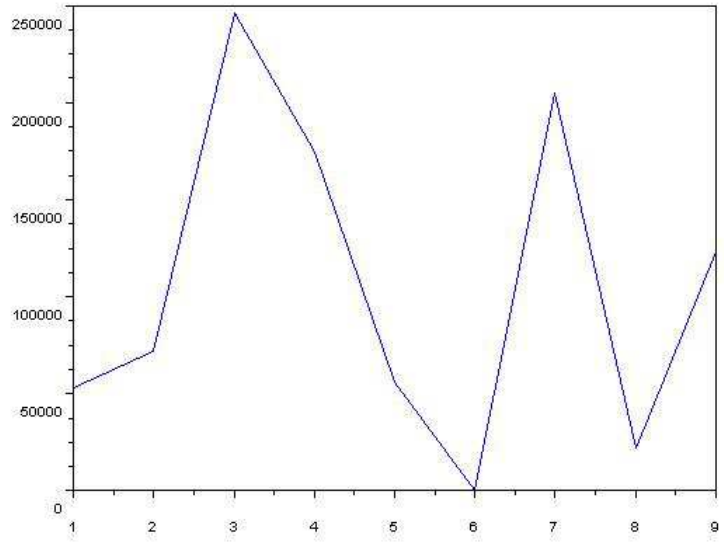

FIGURE 11.4 – intervalles entre les apparitions de la valeur 28856

Le classement des nombres par le nombre de répétitions

Dans cette partie nous allons compter les nombres qui apparaissent le même nombre de fois.

0.	0.	0.	0.	0.	0.
0.	0.	0.	1.	2.	1.
6.	9.	31.	57.	80.	138.
241.	390.	496.	774.	988.	1233.
1519.	1796.	2048.	2124.	2362.	2432.
2352.	2252.	2090.	1858.	1591.	1337.
1104.	885.	729.	531.	407.	289.
211.	128.	98.	72.	34.	26.
17.	11.	7.	10.	1.	

le générateur suit-il une loi uniforme ?

Nous allons regarder si la probabilité (moyenne) d'obtenir une valeur est proche de la probabilité calculée avec la loi des grands nombres.

hypothèse

– le générateur suit une loi uniforme

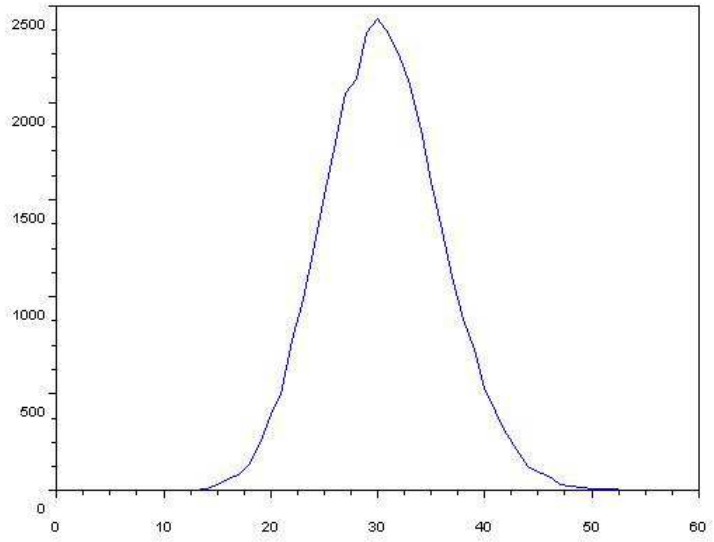

FIGURE 11.5 – courbe de répartition des effectifs

notation

- n effectif de notre échantillon
- p probabilité d'obtenir le nombre souhaité
- $\overline{y}_k$ probabilité moyenne d'obtenir le nombre souhaité

développement

Nous pouvons donc écrire que :

Soient $x_1 \cdots x_n$ des variables aléatoires suivant une loi de Bernoulli, indépendantes, et identiquement distribuées. Nous avons pour chacune de ces variables la probabilité $p(x) = \frac{1}{32767} = 30.5 \cdot 10^{-6}$.

D'autre part nous posons $y_k = \sum_{i=1}^{n} x_i$ est une variable aléatoire suivant la loi binomiale $B(n, p)$ d'espérence

$$E(y_k) = n \cdot p = 1 \cdot 10^6 \times \frac{1}{32767} = 30.52$$

et lorsque $n \to +\infty$

$$\frac{1}{n} \cdot y_k \to E(x_i) = p$$

avec $\overline{y}_k = 30.52$ nous obtenons

$$\frac{1}{n} \cdot \overline{y}_k = \frac{30.52}{1 \cdot 10^6} = 30.5 \cdot 10^{-6}$$

or

$$p = \frac{1}{32767} = 30.5 \cdot 10^{-6}$$

conclusion

Ces résultats nous montrent
- que très rapidement le générateur fournit des doublons,
- que la plage couverte par le générateur est [0 ;32767],
- que les intervalles entre chaque apparition pour les nombres observés ne montrent pas de cycle,
- que les apparitions des nombres observés sont réparties sur une grande partie de la plage comme le montre le tableauTAB.11.2,

nombre	'1^{re} apparition	dernière apparition	intervalle
0	25199	971489	946290
32767	46064	950590	904526
20335	5439	986875	981436
28856	3975	955787	951812

TABLE 11.2 – apparition et intervalle des nombres observés

- que même avec un nombre important de tirages le nombre maximum de répétitions d'un même nombre est très faible ($\frac{53}{1 \cdot 10^6}$) ,
- la médiane de répétition est peu différente de 30,
- que si le nombre de tirages est important la génération de nombres aléatoires en moyenne tend vers une loi uniforme.

Chapitre 12

Annexe B

Choix du composeur de texte LaTeX

Aucun outil n'est imposé. Il a donc été choisi un compilateur C/C++, une suite bureautique pour la réalisation des différents documents.

Cependant rapidement un constat s'est imposé ; la rédaction de note technique contenant des formules mathématiques avec un éditeur standard est trés consommatrice de temps.

Le choix de LaTeX s'est fait tout naturellement après la difficulté rencontrée pour rédiger une note technique sous Word. La méthode de création des fonds de page du document de référence TDA rendait impossible l'utilisation de OpenOffice dans lequel l'éditeur d'équations est plus facile à utiliser et dans lequel l'insertion de formule ne perturbe pas la mise en page.

Ce choix n'est pas sans conséquence car il impose la création d'une classe LaTeX spécifique pour reproduire la mise en page ainsi que les fonds de page TDA appartenant au référentiel ISO de l'entreprise, une production de document au format PDF [1] afin d'en permettre la lecture et la modification éventuelle à défaut d'une personne connaissant le langage de composition LaTeX.

Cependant le choix de LaTeX apporte son lot d'avantages, la qualité des documents produits, la facilité de créer des formules mathématiques, leurs alignements... La possibilité de créer de nouvelles commandes, son langage de composition, le format de stockage des fichiers sources, la robustesse du compilateur, et grace à la distribution en liveCD « TeXLive » par l'association GUTtenberg [2] l'installation est grandement simplifiée.

1. et non DVI le format par défaut
2. Groupe francophone des utilisateurs de TeX `http://www.gutenberg.eu.org`

Choix de l'éditeur XEmacs

Le choix d'un éditeur s'est fait selon plusieurs critères. Il devait permettre la rédaction de code en C, en FunnelWeb, en LaTeX, avoir un langage de création de macro , le lancement de commande depuis une console, en fait être multi-langages, personnalisable, multi-plateforme, ayant une mini console intégrée et sauvegardant les fichiers au format texte.

Les éditeurs comme le bloc-note sont trop basiques et ne permettent pas la création de macro pour automatiser certaines commandes fréquemment utilisées. Les suites bureautiques Office ou OpenOffice ne sont pas conçues pour cette utilisation. Il restait la solution d'utiliser un éditeur différent pour chaque langage, souvent intégré dans un environnement de développement spécifique à un système d'exploitation.

A part Xemacs ou un des nombreux produits dérivés(JOVE, Freemacs, MG, MicroEmacs...) dont certains ne sont pas portés sous Windows , nous n'avons pas trouvé d'éditeur répondant à nos besoins.

Enfin l'avantage de l'utilisation d'un seul éditeur est la possibilité de retenir les raccourcis claviers appelant les fonctions les plus utilisées de l'éditeur, un seul langage d'écriture de macro à apprendre, et ceci est vraiment pertinent quand l'éditeur est multiplateforme, ce qui est le cas de Xemacs.

Choix du précompilateur FunnelWeb

Au départ nous recherchions un outil qui au stade du codage du projet nous permettre d'extraire les commentaires écrits dans du code avec l'idée de ne rédiger qu'une seule documentation associée au programme et l'extraire pour réaliser une note technique.

Le problème était de trouver un outil permettant d'extraire d'un seul fichier le code source directement compilable et le code source de la documentation, en langage LaTeX, mise en page avec des formules mathématiques, l'insertion de graphique, l'écriture d'algorithme.

Très rapidement cette idée est devenue irréalisable ; Les éditeurs habituellement utilisés pour l'écriture de code ne permettent pas de telles mises en page.

La solution était de trouver un logiciel différent pour extraire les commentaires, de repenser la rédaction du code source pour placer la documentation comme élément central lors du développement. Après quelques recherches nous avons constaté que cette réflexion avait déjà été menée et avait abouti à la programmation lettrée.

La programmation lettrée est une approche de la programmation qui donne une grande importance à la documentation, placant celle-ci au centre du développement. Elle a été introduite par Donald Knuth [3] en 1984.

3. Donald Knuth est professeur émérite en informatique à l'université de Standford et créateur

« I believe that the time is ripe for significantly better documentation of programs, and that we can best achieve this by considering programs to be works of literature. Hence, my title : "Literate Programming." » D.Knuth

« Je crois que le temps est venu pour une amélioration significative de la documentation des programmes, et que le meilleur moyen d'y arriver est de considérer les programmes comme des œuvres littéraires. D'où mon titre "programmation lettrée".

FIGURE 12.1 – propos de Donald Knuth au sujet de la programmation lettrée

Le principe de cette façon de programmer est de placer le code source associé au commentaire au milieu de celui-ci ; le code est une illustration du commentaire. L'idée de Donald Knuth va même plus loin ; pour lui la rédaction d'un programme lettré doit être bien rédigé, soigné et avoir pour but de faciliter la compréhension des concepts présentés. En programmation lettrée on écrit un ouvrage littéraire contenant du code informatique.

Il appela le précompilateur « WEB » et l'utilisa pour commenter les codes sources de son programme TEX puis développa CWEB avec Silvio Levy pour gérer la documentation des codes sources écris en C,C++, ANSI C, JAVA et utilise TEX pour la création de la documentation.

Aujourd'hui on utilise LATEX [4] plutôt que TEX pour réaliser un document. Ce composeur de texte est largement répandu dans le milieu scientifique et la communauté d'utilisateurs a produit beaucoup d'extensions rendant son utilisation plus facile et de ce fait place TEX en retrait.

« Imaginez LATEX comme une maison dont la charpente et les clous seraient fournis par TEX. Vous n'en avez pas besoin pour vivre dans la maison, mais ils sont pratiques pour y ajouter une nouvelle pièce. »

FIGURE 12.2 – propos de Leslie LAMPORT au sujet de LATEX

Il existe aujourd'hui un certain nombre de programmes inspirés de WEB dont FunnelWeb un pré processeur de programation lettrée de Ross Williams, indépendant du langage de programmation, fonctionnant sur plusieurs plate-formes dont

du système de composition de documents TEX

4. créé par Leslie LAMPORT professeur en informatique spécialiste de l'algorithmique répartie. Il travaille actuellement pour les centres de recherche de Microsoft. Ses dernières recherches portent sur le langage algorithmique PlusCal

Linux, MAC OSX , Windows. Rapide, il supporte la génération de document aux formats TEX, LATEX, HTML.

FunnelWeb se comporte toujours de la même manière. Il sépare le programme du texte de la documentation et les sauvegarde dans des fichiers séparés. Pour cela dans le texte du fichier source il faut utiliser des balises propres à FunnelWeb.

exemple de script

– le commentaire de début de fichier avec la définition des paramètres destinés à funnelWeb (précéder de @p).

```
@! programme de test de l'utilisation de FunnelWeb
@! avec l'addition des commandes de Latex pour
@! r\'ealiser la pr\'esentation de la documentation
@!
@! date  19 mars 2008

@p typesetter = tex
```

– début de la documentation écrite en LATEX

```
\documentclass[a4paper, 10pt]{article}
\usepackage[frenchb]{babel}
\usepackage [T1]{fontenc}
\usepackage[latin1]{inputenc}
\usepackage{url}

\begin{document}

\title{test de funnelWeb\\ avec \LaTeX}
\author{philippe Davignon}
\date{\today}
\maketitle

\section*{introduction}
 FunnelWeb est un compilateur de programme litt\'eraire.
...
```

Un problème subsite ; FunnelWeb ne supporte pas les caractères accentués. Ce qui rend difficile la production de documents dans une autre langue que l'anglais.

Pour réussir la modification il est alors envisagé de réaliser une fonction exécutable sous Emacs. Cette fonction sera écrite en ELisp le langage d'extension de Emacs qui permet de programmer des fonctions et macro pour Emacs.

– extrait de la déclaration de fonctions au milieu de la documentation

```
@$@<script variable@>@Z==@{@-
; definit le choix du dictionnaire par defaut
```

```
(setq ispell-dictionary "francais")

;definition de la longueur d'une ligne
;pour repondre a la contrainte de funnelWeb
;la longueur max de la ligne est 80 caracteres

(setq fundamental-mode-hook 'turn-on-auto-fill)
(setq fill-column 72)
@}

et la fonction pour la gestion des caract\'eres accentu\'es

@$@<fonction change-accented-character@>@Z==@{@-

; define a function for change a syntax of accented character
(defun change-accented-character ()

"search and replace a accented character like define bollow.
It's a location for a french language"
@}
...
```

– indépendant du langage utilisé dans les fonctions FunnelWeb autorise l'utilisation de plusieurs langages de programmation dans un même script.

```
un extrait de code pour le logiciel Scilab
Cette fonction additionne les deux nombres passes en arguments.

@$@<fonction add@>==@{@-
function [z] = add(x,y)
    z = 0;
    z = x + y;
endfunction @}
@!
exemple d'utilisation dans un script FunnelWeb:
@O@<test.sce@>@{@-
@<fonction add@>
A = 12;
B = 15;
c = add(A,B);
@}
...
```

L'ensemble composé de la fonction add et du programme principal est copié

dans le fichier test.sce et peut être directement chargé dans Scilab.

Toutes ces fonctions ont été extraites du même fichier contenant un script Interprétable par FunnelWeb.

Le CMMI

Le but est de réaliser la corrélation entre le référentiel CMMI, le référentiel ISO 9000 et la structure de la documentation TDA.

Il reprend les grandes étapes du référentiel CMMI, sa représentation étagée, son organisation des domaines de processus par niveau de maturité et catégorie, la correspondance avec le référentiel ISO9000 :2000.

Il associe les documents du référentiel THALES aux étapes de stratégie de développement logiciel, en phase avec les processus définie dans le CMMI au niveau de maturité 2.

Présentation du CMMI

Le CMMI est un référentiel composé d'un ensemble structuré de bonnes pratiques, destiné à appréhender, évaluer et améliorer les activités des entreprises d'ingénierie.

Le CMMI-DEV est un sous-ensemble du CMMI. C'est un modèle de processus pour la réalisation de tout type de produit. Il s'adapte bien au développement de logiciel. Il couvre la totalité de la vie du logiciel.

Il est un complément de la norme ISO9001 dans le cadre du développement logiciel.

Le modèle se décompose en 5 niveaux de maturité qui regroupent des domaines de processus regroupés en catégorie.

Description des niveaux de maturité : [5]

initial Au niveau de maturité 1, les processus sont habituellement circonstanciels et chaotiques. En général, l'organisation ne fournit pas l'environnement stable qui permettrait de les pérénniser. Le succès dépend de la compétence et de la bonne volonté de ses membres et non de l'application de process éprouvés. Malgrés ce chaos, les organisations de ce niveau de maturité produisent souvent des produits qui fonctionnent. Toutefois, il est fréquent qu'elles dépassent leurs budgets et ne respectent pas leurs délais.

Discipliné Au niveau de maturité 2, les projets de l'organisation respectent des processus planifiés et exécutés selon les règles. Ils emploient du personnel compétent qui dispose des ressources adéquates pour produire des sorties contrôlées. Ils impliquent les parties prenantes concernées. Ils sont surveillés,

5. description issues de CMMI pour le développement V1.2

contrôlés, revus et évalués au regard de leur respect des descriptions de processus. La discipline reflétée au niveau 2 de maturité permet d'assurer que les pratiques existantes sont maintenues pendant les périodes de stress. Quand ces pratiques sont en place, les projets sont réalisés et contrôlés selon les plans documentés. Au niveau de maturité 2, le statut des produits d'activité et des prestations de services est constaté lors d'échéances défi nies (par exemple aux jalons importants et lors de l'accomplissement des tâches importantes). Des engagements sont pris par les parties prenantes concernées et mis à jour si nécessaire. Les produits d'activité sont convenablement contrôlés. Les produits d'activité et les services respectent leurs descriptions de processus spécifi ques, normes et procédures.

Ajusté Au niveau de maturité 3, les processus sont précisés et compris. Ils sont décrits dans les normes, les procédures, les outils et les méthodes. L'ensemble des processus standards de l'organisation, qui sert de référentiel à ce niveau de maturité, est établi et amélioré avec le temps. Ces processus standards sont employés pour établir l'uniformité/l'homogénéité au sein de l'organisation. Les projets élaborent leurs processus ajustés en adaptant l'ensemble des processus standards de l'organisation aux directives d'ajustement. Une distinction essentielle entre les niveaux de maturité 2 et 3 a trait à la portée des normes, des descriptions de processus et des procédures. Au niveau de maturité 2, celles-ci peuvent être très différentes dans chaque instance spécifique du processus (par exemple pour un projet particulier). Au niveau 3, les normes, les descriptions de processus et les procédures qui s'appliquent à un projet particulier ou à une unité organisationnelle sont ajustées à partir de l'ensemble des processus standards de l'organisation et sont donc plus uniformes, à l'exception des différences autorisées par les directives d'ajustement. Le niveau 3 présente une autre différence capitale : les processus y sont décrits plus rigoureusement qu'au niveau 2. Une description de processus ajusté énonce clairement l'intention, les entrées, les critères d'entrée, les activités, les rôles, les mesures, les étapes de vérification, les sorties et les critères de sortie. Au niveau de maturité 3, les processus sont gérés de manière plus proactive, grâce à la compréhension des interrelations entre les activités du processus et aux mesures détaillées du processus, de ses produits d'activité et de ses services. Au niveau 3, l'organisation doit approfondir la maturité des domaines de processus issue du niveau 2. Les pratiques génériques associées aux objectifs génériques qui n'ont pas été traitées au niveau 2 sont appliquées pour atteindre le niveau 3.

Géré quantitativement Au niveau de maturité 4, l'organisation et les projets fixent des objectifs quantitatifs de qualité et de performance et les emploient comme critères pour gérer les processus. Ces objectifs quantitatifs sont fon-

dés sur les besoins du client, des utilisateurs, de l'organisation et de ceux qui appliquent les processus. La qualité et la performance des processus sont traitées en termes statistiques et gérées durant toute leur durée de vie [SEI 2001]. Pour les sous-processus choisis, des mesures de performance détaillées sont recueillies et analysées statistiquement. Les mesures de qualité et de performance sont intégrées au référentiel de mesures de l'organisation pour permettre aux prises de décision de s'appuyer sur des faits [McGarry 2000]. Les causes spéciales de variation des processus sont identifiées et, le cas échéant, leurs sources sont corrigées pour empêcher la répétition de ces variations. Une distinction essentielle entre les niveaux de maturité 3 et 4 porte sur la prévisibilité de l'exécution des processus. Au niveau 4, l'exécution des processus est contrôlée grâce à des statistiques et à d'autres techniques quantitatives. Elle est quantitativement prévisible. Au niveau 3, les processus ne sont généralement prévisibles qu'au seul plan qualitatif.

En optimisation Au niveau de maturité 5, une organisation améliore continuellement ses processus en s'appuyant sur une compréhension quantitative des causes communes de variation inhérentes aux processus. Le niveau 5 se concentre sur l'amélioration continue de la performance des processus en s'appuyant sur les améliorations incrémentales Des objectifs quantitatifs d'amélioration des processus organisationnels sont établis, continuellement mis à jour pour s'adapter aux évolutions des objectifs stratégiques et employés comme critères de contrôle. Les effets des améliorations déployées sont mesurés et évalués par rapport aux objectifs quantitatifs défi nis. Les processus ajustés et l'ensemble des processus standards de l'organisation constituent les cibles des activités d'amélioration mesurables. Une distinction essentielle entre les niveaux de maturité 4 et 5 est relative au type de variation de processus concerné. Au niveau 4, l'organisation s'occupe de traiter les causes spéciales de variation des processus et d'assurer la prévisibilité statistique des résultats. Les processus peuvent produire des résultats prévisibles, mais ceux-ci ne sont pas nécessairement suffisants pour réaliser les objectifs fixés. Au niveau 5, l'organisation s'intéresse aux causes communes de variation des processus et à la modification de processus (pour déplacer la moyenne de performance des processus ou réduire la variation inhérente constatée) afin d'en améliorer l'exécution et d'atteindre les objectifs quantitatifs d'amélioration fixés.

Domaine de processus	Catégorie	Niveau de maturité
Gestion des éxigences	Ingénerie	2
Planification de projet	Gestion de projet	2
Surveillance et contrôle de projet	Gestion de projet	2
Gestion des accords avec les fournisseurs	Gestion de projet	2
Gestion de configuration	Support	2
Mesure et analyse	Support	2
Assurance-qualité processus et produits	Support	2
Développement des éxigences	Ingénerie	3
Solution technique	Ingénerie	3
Intégration du produit	Ingénerie	3
Validation	Ingénerie	3
Vérification	Ingénerie	3
Gestion des risques	Gestion de projet	3
Gestion de projet intégrée + IPPD	Gestion de projet	3
Focalisation sur le processus organisationnel	Gestion des processus	3
Définition des processus organisationnel	Gestion des processus	3
Formation organisationnelle	Gestion des processus	3
Analyse et prise de décision	Support	3
Gestion de projet quantitatif	Gestion de projet	4
Performance de processus organisationnel	Gestion des processus	4
Innovation et déploiement organisationnels	Gestion des processus	5
Analyse causale et résolution	Support	5

TABLE 12.1 – Domaines de processus par niveau de maturité et catégorie

Détails des Domaines de processus du CMMI applicables aux niveau de maturité 2 dit « discipliné »

Les domaines de processus se composent de deux grandes parties ; les objectifs génériques et les pratiques spécifiques ou géneriques pour atteindre ces objectifs.

Gestion des éxigences

Objectifs génériques

SG1 Gérer les éxigences : identifier les différences entre les éxigences et le plan du projet, les incohérences entre les éxigences.

Pratiques spécifiques

SP1.1 Obtenir une compréhension des éxigences : développer une compréhension commune des éxigences et de leur signification avec ceux qui les ont fournies.

SP1.2 Obtenir l'engagement sur les éxigences : obtenir des participants au projet leur engagement sur les exigences.

SP1.3 Gérer les modifications aux exigences : Gérer les modifications aux exigences au fur et à mesure de leur évolution en cours de projet.

SP1.4 Maintenir la traçabilité bidirectionnelle des exigences : Etablir la traçabilité entre les exigences sources et les exigences de plus bas niveau. S'assurer que toutes les exigences sont correctement traitées.

SP1.5 Identifier les incohérences entre les produits du projet et les exigences :

Planification de projet

Objectifs génériques

SG1 Etablir les estimations : Les estimations des paramètres de planification de projet sont établies et maintenues.

SG2 Développer un plan de projet : Un plan de projet est établi et maintenu pour servir de base à la gestion du projet.

SG3 obtenir l'engagement sur le plan : Les engagements sur le plan du projet sont établis et maintenus.

Pratiques Spécifiques

SP1.1 Faire l'estimation de la portée du projet : Etablir un découpage de haut niveau (WBS) pour faire l'estimation de la portée du projet.

SP1.2 Etablir les estimations des attributs des produits d'activité et des tâches : Etablir et maintenir les estimations des attributs des produits d'activité et des tâches.

SP1.3 Définir le cycle de vie du projet : Définir les phases du cycle de vie du projet sur lesquelles la planification est faite.

SP1.4 Déterminer les estimations de charge et de coût : faire l'estimation de charge et de coût du projet pour les produits d'activité et les tâches en se basant sur une approche raisonnée.

SP2.1 Etablir le budget et le calendrier : Etablir et maintenir le budget et le calendrier du projet.

SP2.2 Identifier les risques du projet : Identifier et analyser les risques du projet.

SP2.3 Prévoir la gestion des données : Les représentent les différentes formes

de documentation requises pour prendre en charge les domaines d'un programme.

SP2.4 Prévoir les ressources du projet : Définir les ressources du projet (main-d'œuvre, machine, matériaux, méthodes...)

SP2.5 Prévoir les connaissances et compétences nécessaires : prévoir la formation du personnel et l'acquisition de connaissances provenant de l'extérieur.

SP2.6 Prévoir l'implication des parties prenantes : Désigner les interlocuteurs et les fonctions qui doivent être représentés dans le projet.

SP2.7 Etablir le plan du projet

SP3.1 Passer en revue les plans qui ont des répercussions sur le projet : Passer en revue tous les plans qui ont des répercussions sur le projet afin de comprendre les engagements sur le projet.

SP3.2 Concilier le niveau de charge et de ressources : Mettre en le plan de projet en cohérence avec les ressources disponibles par rapport aux ressources estimées.

SP3.3 Obtenir l'engagement du plan : Obtenir l'engagement des parties prenantes concernées qui sont responsables de réaliser le plan ou d'en soutenir la réalisation.

Surveillance et contrôle de projet

Objectifs génériques

SG1 Surveiller le projet par rapport au plan : La performance et l'avancement réels du projet sont surveillés par rapport au plan de projet.

SG2 Gérer l'action corrective jusqu'a clôture : Prendre des actions correctives pour les écarts identifiés.

Pratiques Spécifiques

SP1.1 Surveiller les paramètres de planification de projet : Surveiller les valeurs réelles des paramètres de planification de projet par rapport au plan de projet.

SP1.2 Surveiller les engagements : Surveiller les engagements par rapport à ceux identifiés dans le plan de projet.

SP1.3 Surveiller les risques du projet : Surveiller les risques par rapport à ceux identifiés dans le plan de projet.

SP1.4 Surveiller la gestion des données : Surveiller la gestion des données du projet par rapport au plan de projet.

SP1.5 Surveiller l'implication des parties prenantes.

SP1.6 Mener des revues d'avancement.

SP1.7 Mener des revues sur jalons.

SP2.1 Analyser les écarts.
SP2.2 Appliquer une action corrective.
SP2.3 Gérer un action corrective.

Gestion de configuration

Objectifs génériques

SG1 Etablir les référentiels des produits d'activité.
SG2 Suivre et contrôler les modifications.
SG3 Etablir l'intégrité

Pratiques Spécifiques

SP1.1 Identifier les éléments de configuration qui seront gérés en configuration
SP1.2 Etablir un système de gestion de configuration.
SP1.3 Créer ou figer des référentiels.
SP2.1 Suivre les demandes de modification
SP2.2 Contrôler les éléments de configuration
SP3.1 Etablir des enregistrements de gestion de configuration
SP3.3 Mener des audits de configuration

Gestion des accords avec les fournisseurs

Objectifs génériques

SG1 Etablir les accords avec les fournisseurs
SG2 Se conformer aux accords avec les fournisseurs

Pratiques Spécifiques

SP1.1 Déterminer le type d'acquisition pour chaque produit ou composant de produit à acquérir
SP1.2 Choisir des fournisseurs
SP1.3 Etablir des accords formels avec les fournisseurs
SP2.1 Exécuter les activités avec le fournisseurs telles qu'elles sont spécifiées dans l'accord.
SP2.2 Sélectionner, surveiller et analyser des processus utilisés par le fournisseur.
SP2.3 Dans le cas d'un fournisseur de produits sur commande, sélectionner et évaluer des produits d'activité de celui-ci.
SP2.4 S'assurer que l'accord avec le fournisseur est satisfait avant d'accepter le produit acquis.

SP2.5 Transférer au projet les produits acquis du fournisseur.

Mesure et analyse

Objectifs génériques

SG1 Aligner le activités de mesure et d'analyse.
SG2 Fournir des résultats de mesures

Pratiques Spécifiques

SP1.1 Etablir des objectifs de mesure
SP1.2 Spécifier des mesures
SP1.3 Spécifier des procédures de collecte et de stockage des données.
SP1.4 Spécifier comment les données de mesure seront analysées et communiquées.
SP2.1 Obtenir les données de mesure spécifiées.
SP2.2 Analyser et interpréter les données de mesure.
SP2.3 Gérer et stocker les données de mesure, les spécifications de mesure et les analyses de résultats.
SP2.4 Communiquer les résultats des activités de mesure et d'analyse à toutes les parties prenantes concernées.

Assurance-qualité processus et produit

Objectifs génériques

SG1 Evaluer de manière objective des processus et des produits d'activité
SG2 Fournir une image objective.

Pratiques Spécifiques

SP1.1 Evaluer de manière objective des processus
SP1.2 Evaluer de manière objective des produits d'activité et des services
SP2.1 Communiquer et assurer la résolution des non-conformités.
SP2.2 Etablir des enregistrements.

Pratiques Géneriques

Les pratiques génériques sont communes pour tous les domaines de processus.
GG2 Institutionaliser un processus discipliné.

GP2.1 Etablir une directive organisationnelle : Etablir et maintenir une directive organisationnelle traitant de la planification et de la mise en œuvre du processus.

GP2.2 Planifier le processus : Etablir et maintenir le plan pour mettre en œuvre le processus .

GP2.3 Fournir les ressources : Fournir les ressources adéquates pour mettre en œuvre le processus et fournir les services couverts par le processus.

GP2.4 Assigner la responsabilité : Assigner la responsabilité et l'autorité pour mettre en œuvre le processus, développer les produits d'activité et fournir les services couverts par le processus.

GP2.5 Former les personnes

GP2.6 Gérer en configuration

GP2.7 Identifier et impliquer les parties prenantes concernées : Identifier et impliquer les parties prenantes concernées par le processus comme prévu dans le plan.

GP2.8 Surveiller et contrôler le processus : Surveiller et contrôler le processus vis-a-vis de son plan de mise en œuvre et prendre les actions correctives appropriées.

GP2.9 Evaluer la conformité de manière objective : Evaluer de manière objective le respect par le processus , des normes et des procédures qui devraient être respectées , et traiter les non-conformités détectées.

GP2.10 Passer le statut en revue avec la hiérarchie : Passer en revue avec la hierarchie, les activités, le statut et les résultats du processus et résoudre les problèmes.

Tableau de correspondance entre les processus du CMMI maturité 2 et ceux de l'ISO 9000

Objectif génériques	pratiques spécifiques	description	ISO 9001 : 2000
		Gestion des éxigences	
SG1		Gérer les exigences	
	SP1.1	Obtenir une compréhension des exigences	7.2.1,7.2.2, 8.2.4
	SP1.2	Obtenir l'engagement sur les éxigences	7.2.3
	SP1.3	Gérer les modifications aux exigences	7.2.2
	SP1.4	Maintenir la traçabilité bidirectionnelle des exigences	7.5.3
	SP1.5	Identifier les incohérences entre les produits du projet et les exigences	7.2.2
		Planification de projet	
SG1		Etablir les estimations	
	SP1.1	Faire l'estimation de la portée du projet	7.1,7.3.1
	SP1.2	Etablir les estimations des attributs des produits d'activité et des tâches	7.1,7.3.1
	SP1.3	Définir le cycle de vie du projet	7.1,7.3.1
	SP1.4	Déterminer les estimations de charge et de coût	7.3.1
SG2		Développer un plan de projet	
	SP2.1	Etablir le budget et le calendrier	7.1,7.3.1
	SP2.2	Identifier les risques du projet	7.1,7.3.1
	SP2.3	Prévoir la gestion des données	7.1,7.3.1
	SP2.4	Prévoir les ressources du projet	6.4,7.1,7.3.1
	SP2.5	Prévoir les connaissances et compétences nécessaires	7.1,7.3.1
	SP2.6	Prévoir l'implication des parties prenantes	7.1,7.3.1
	SP2.7	Etablir le plan du projet	7.1,7.3.1
SG3		Obtenir l'engagement sur le plan	
	SP3.1	Passer en revue les plans qui ont des répercussions sur le projet	7.3.1
	SP3.2	Concilier le niveau de charge et de ressources	7.3.1
	SP3.3	Obtenir l'engagement du plan	7.3.1

suite sur la page suivante

Objectif génériques	pratiques spécifiques	description	ISO 9001 : 2000
		Surveillance et contrôle de projet	
SG1		Surveiller le projet par rapport au plan	
	SP1.1	Surveiller les paramètres de planification de projet	
	SP1.2	Surveiller les engagements	
	SP1.3	Surveiller les risques du projet	
	SP1.4	Surveiller la gestion des données	
	SP1.5	Surveiller l'implication des parties prenantes.	8.2.1
	SP1.6	Mener des revues d'avancement.	5.6.2,5.6.3, 7.3.4
	SP1.7	Mener des revues sur jalons.	5.6.2,5.6.3, 7.3.4
SG2		Gérer l'action corrective jusqu'a clôture	
	SP2.1	Analyser les écarts.	5.6.2,5.6.3, 8.2.3,8.3, 8.5.2
	SP2.2	Appliquer une action corrective.	5.6.2,5.6.3, 8.2.3,8.3, 8.5.2
	SP2.3	Gérer une action corrective	5.6.2,5.6.3, 8.2.3,8.3, 8.5.2
		Gestion de configuration	
SG1		Etablir les référentiels des produits d'activité.	
	SP1.1	Identifier les éléments de configuration qui seront gérés en configuration	4.2.3, 7.3.7,7.5.3, 8.3
	SP1.2	Etablir un système de gestion de configuration.	4.2.3, 7.3.7, 8.3
	SP1.3	Créer ou figer des référentiels.	4.2.3, 7.3.7,7.5.3, 8.3

suite sur la page suivante

Objectif génériques	pratiques spécifiques	description	ISO 9001 : 2000
SG2		Suivre et contrôler les modifications.	
	SP2.1	Suivre les demandes de modification	4.2.3, 7.3.7,7.5.3, 8.3
	SP2.2	Contrôler les éléments de configuration	4.2.3, 7.3.7,7.5.3, 8.3
SG3		Etablir l'intégrité	
	SP3.1	Etablir des enregistrements de gestion de configuration	4.2.3, 7.3.7,7.5.3, 8.3
	SP3.3	Mener des audits de configuration	4.2.3, 7.3.7, 8.2.4,8.3
		Gestion des accords avec les fournisseurs	
SG1		Etablir les accords avec les fournisseurs	
	SP1.1	Déterminer le type d'acquisition pour chaque produit ou composant de produit à acquérir	7.4.1,7.4.2
	SP1.2	Choisir des fournisseurs	7.4.1
	SP1.3	Etablir des accords formels avec les fournisseurs	4.1, 7.4.2, 8.2.4
SG2		Se conformer aux accords avec les fournisseurs	
	SP2.1	Exécuter les activités avec le fournisseurs telles qu'elles sont spécifiées dans l'accord.	7.4.2,7.4.3
	SP2.2	Sélectionner, surveiller et analyser des processus utilisés par le fournisseur.	
	SP2.3	Dans le cas d'un fournisseur de produits sur commande, sélectionner et évaluer des produits d'activité de celui-ci.	4.1, 7.4.3, 8.4
	SP2.4	S'assurer que l'accord avec le fournisseur est satisfait avant d'accepter le produit acquis.	7.4.3
	SP2.5	Transférer au projet les produits acquis du fournisseur.	

suite sur la page suivante

Objectif génériques	pratiques spécifiques	description	ISO 9001 : 2000
		Mesure et analyse	
SG1		Aligner le activités de mesure et d'analyse.	
	SP1.1	Etablir des objectifs de mesure	8.1, 8.2.1, 8.5.1
	SP1.2	Spécifier des mesures	8.1,8.2.1, 8.2.3, 8.5.1
	SP1.3	Spécifier des procédures de collecte et de stockage des données.	8.1, 8.2.3
	SP1.4	Spécifier comment les données de mesure seront analysées et communiquées.	8.1, 8.5.1
SG2		Fournir des résultats de mesures	
	SP2.1	Obtenir les données de mesure spécifiées.	8.2.1, 8.4, 8.5.1
	SP2.2	Analyser et interpréter les données de mesure.	8.4, 8.5.1
	SP2.3	Gérer et stocker les données de mesure, les spécifications de mesure et les analyses de résultats.	8.4
	SP2.4	Communiquer les résultats des activités de mesure et d'analyse à toutes les parties prenantes concernées.	7.2.3, 8.2.2
		Assurance-qualité processus et produit	
SG1		Evaluer de manière objective des processus et des produits d'activité	8.2.2
	SP1.1	Evaluer de manière objective des processus	8.2.2
	SP1.2	Evaluer de manière objective des produits d'activité et des services	8.2.2, 8.2.4
SG2		Fournir une image objective.	8.2.2
	SP2.1	Communiquer et assurer la résolution des non-conformités.	8.2.2
	SP2.2	Etablir des enregistrements.	8.2.2

Tableau des documents THALES/TDA en regard des processus du CMMI

TABLE 12.2 – correspondance entre les processus de maturité 2 et le référentiel ISO9000

	CMMI	
documents TDA	processus	objectif
Plan de projet (SDP) code type 311	surveillance et contrôle de projet	Surveiller le projet par rapport au plan
		Gérer l'action corrective jusqu'à clotûre
	Planification de projet	Etablir les estimations
		Développer un plan de projet
spécification des exigences (SRS) code type 306	Gestion des exigences	Gérer les exigences
spécification des interfaces (IRS) code type 206	Gestion des exigences	Gérer les exigences
Revue des éxigences (RE)	Assurance-qualité processus et produit	Evaluer de manière objective Des processus et des produits d'activité
documents de conception logiciel (SDD) code type 549		
Description des tests de qualification (STD)	Assurance-qualité processus et produit	Evaluer de manière objective Des processus et des produits d'activité
	mesure et analyse	Aligner les activités de mesure et d'analyse
documents de conception des interfaces (IDD)		

suite du tableau à la page suivante

documents TDA	processus	objectif
revue des documents de conception (CDR)	Assurance-qualité processus et produit	Evaluer de manière objective Des processus et des produits d'activité
Ebauche de description des tests (STR)code type 279	Assurance-qualité processus et produit	Evaluer de manière objective Des processus et des produits d'activité
	mesure et analyse	Aligner les activités de mesure et d'analyse
revue de préparation aux tests (TRR)	Assurance-qualité processus et produit	Evaluer de manière objective Des processus et des produits d'activité
	mesure et analyse	Aligner les activités de mesure et d'analyse
Spécification de produit logiciel (SPS) code type 502	mesure et analyse	Gérer et stocker les données de mesure, les spécifications de mesure et les analyses de résultat.
support matériel code type 300	mesure et analyse	Gérer et stocker les données de mesure, les spécifications de mesure et les analyses de résultat.
		fin du tableau de correspondance

Présence des documents dans le référentiel TDA et dans la GED

Le réferentiel TDA possède une banque de formulaire vierge pour la rédaction des différents documents. Les documents sont gérer dans la GED avec leurs code type. Le tableau recense l'éxistance d'un formulaire et du code type dans la GED.

documents TDA	réferentiel TDA	GED
Plan de projet	inconnu	311
spécification des exigences	inconnu	306
spécification des interfaces	inconnu	206
revue des exigences	inconnu	inconnu
documents de conception logiciel	inconnu	549
suite du tableau sur la page suivante		

documents TDA	processus	objectif
description des test de qualification	inconnu	inconnu
documents de conception des interfaces	inconnu	inconnu
revue des documents de conception	inconnu	inconnu
ébauche des descriptions de tests	inconnu	279
revue de préparation aux tests	inconnu	inconnu
spécification de produit logiciel	inconnu	502
support matériel	sans objet	300

Bibliographie

les ouvrages littéraires

[C08] Chrissis Mary Beth, Konrad Mike, Shrum Sandy(2008) *"CMMI® guide des bonnes pratiques pour l'amélioration des processus"* Pearson Education FRANCE

[C72] Cazin Michel (1972) *"Cours de mécanique générale et industrielle Tome 1"* Paris gauthier-villars

[C74] Cazin Michel (1974) *"Cours de mécanique générale et industrielle Tome 2"* Paris gauthier-villars

[D52] Guillon E, Hollier-Larousse J.P., Ibos -Auge J., Moreau C., Moreau J.L. (1952). *" Dictionnaire encyclopédique, Nouveau petit larousse illustré"* Paris, Librairie Larousse.

[L02] Stroustrup Barjne(2002) *"Le Langage C++"* Paris Pearson Education

[L05] Mittelbach Franck, Goossens Michel(2005) *"LaTeX Companion"* paris Pearson Education

[L94] Leroy Bernard(1994) *"Langage C programmation"* Sybex

[M72] Muller Jacques(1972) *"Formulaire technique de Mécanique Générale"* Abbeville Imprimerie F. Paillart

[M73] Cazin Michel, Metge Jeanine(1973) *"Cours de mécanique industrielle élémentaire"* Paris gauthier-villars

[P07] Bogeart Patrick(2007) *"Probabilités pour scientifiques et ingénieurs"* Bruxelles éditions De Boeck Université

[P97] Loukides Mike , Oram Andy(1997) *" Programmer avec les outils GNU" Traduction de Manuel et Nat Makarévitch* Paris, éditions O'Reilly

[R95] Rosen Jean-Pierre(1995) *"Méthode de génie logiciel avec Ada95"* Paris InterEditions

[S00] Welsh Matt, Dalheimer Kalle et Kaufman Lar(2000) *"Le système Linux" traduction de René Cougnenc, Manjuel Makarévitch, Jean-Michel Vansteene* Paris, éditions O'Reilly

[S07] Chancelier Jean-Philippe, Goursat Maurice, Delebecque Francois, Nikoukhah Ramine, Gomez Claude, Steer Serge(2007) *"introduction à SCILAB"* Springer

[T96] Seroul Raymond(1996) *"Le petit livre de TEX"* Paris, éditions MASSON

[U04] Fowler Martin(2004) *"UML 2.0"* Paris Pearson Education

articles et revues

[B96] Bru Bernard(1996) *Problème de l'efficacité du tir à l'école d'artillerie de Metz. Aspects théoriques et expérimentaux* NUMDAM

[C76] Claustriaux J.J.(1976) *Génération et contrôle de validité de nombres pseudo-aléatoires sur ordinateur à mots de 16 bits Revue de statistique appliquée, tome 24, N° 2, p75-88* NUMDAM

[M93] Marsaglia George(1993) *Monkey Tests for Random Number Generators extracted from an article in Computers & Mathematics with Applications 9, 1–10, 1993*

[S06] ministère de la Défense(2006) *SAPEUR revue du génie militaire français N° 6*

[T09] Jacques André(2009) *Petites leçons de typographie Irisa*

guides et notes techniques THALES/TDA

[B18] BUCS.Aéroportées/NT/N° 018/02-A *simulateur tigre sous système roquette 68 mm* mise en œuvre descriptif du logiciel du modèle d'efficacité de la roquette AMV

[B19] BUCS.Aéroportées/NT/N° 019/02-A *simulateur tigre sous système roquette 68 mm* mise en œuvre du logiciel du modèle d'efficacité de la roquette AMV

[TDA1] Note Technique/NT/N° 46810 (2006) *Méthodologie de développement logiciel*

[THA1] Guide pour l'allocation des exigences (2004) enoncé des éxigences logiciel

[THA2] Analyse et spécification des exigences du logiciel (2004)

[THA3] Guide de rédaction des spécifications sur le logiciel (1997)

[THA4] Guide de rédaction des spécifications sur le logiciel (2004) *Développement orienté objet création de documents techniques tome 4*

[THA5] Guide de rédaction pour la description du test du logiciel (1997)

[THA6] Guide de rédaction pour la spécification des exigences sur les interfaces (1997)

[THA7] Démarche de tests IHM (2005)

[THA8] Elaboration des documents logiciel (2002)

[THA9] Guide de rédaction pour le Document de conception du logiciel (1997)

[THA10] Guide pour la conception détaillée (2004)

[THA11] Principes de test de composants logiciels (2005)

[THA12] La conception structurée (2004)

[THA13] Guide de rédaction pour le document de conception d'interface (1997)

[THA14] Guide de rédaction pour le rapport de test du logiciel (1997)

[THA15] Cycle de vie du logiciel (2004)

[THA16] Elaboration de plan de développement logiciel (2005)

[THA17] Guide de programmation en C ANSI(2005)

liens et sites internet

[AL08-09] le site de alaide.com(2009)
`http://www.alaide.com/outils_testmotdepasse.php`

[AT12-08] Site officiel de l'armée de terre Française(2008).
`http://www.defense.gouv.fr/terre`

[CS08-09] `http://cs76.free.fr/test-mot-passe.php`

[DK05-09] Site du Professeur Donald Knuth(2009)
`http://www-cs-faculty.stanford.edu/~knuth/`

[Em12-08] Site wiki des nouveaux utilisateurs de Emacs(2008)
`http://www.emacswiki.org/emacs-fr/NouvelUtilisateurEmacs#toc21`

[FW12-08] Site officiel de FunnelWeb(2008)
`http://www.ross.net/funnelweb/index.shtml`

[GM08-09] Site du Professeur Georges Marsaglia(2009)
`http://www.stat.fsu.edu/pub/diehard/`

[IN12-08] Site de l'institut national de la propriété industrielle (2008).
`http://www.inpi.fr/fr/services-et-prestations/enveloppe-soleau/la-vie-de-l-enveloppe-soleau.html`

[IR08-09] site de inforisque(2009) `http://www.inforisque.info/fiches-pratiques/verification-mot-de-passe.php`

[JN12-08] Site du journal du net(2008).
http://www.journaldunet.com/solutions/0312/031217_juridique.shtml

[LL05-09] Site du Professeur Leslie Lamport(2009)
http://research.microsoft.com/en-us/um/people/lamport/

[MS08-09] site MSDN de microsoft(2009)
http://msdn.microsoft.com/fr-fr/library/b05h65z0.aspx

[MSC08-09] site de Microsoft au canada
http://www.microsoft.com/canada/fr/athome/security/privacy/password_checker.mspx

[T508-09] http://www.top500.org/lists/2007/11/performance_development

[TC08-09] http://tools.cases.lu/pwdtest/index.php

Table des figures

Liste des tableaux