REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

Université M^{ed} Khider, Biskra

Faculté des sciences exactes, des sciences de la nature et de la vie

Département des mathématiques

Intégration Numérique (1D) basée sur l'Intelligence en Essaim

Master en Science

M^{elle} Meriem Ben seghier

Année 2011/2012

Résumé

Ce mémoire propose une méthode de quadrature (FPSO) pour le calcul approximatif des intégrales définies en utilisant l'optimisation par Essaim de Particules (PSO). PSO est une technique basée sur la coopération entre les particules. L'échange d'informations entre ces particules permet de résoudre des problèmes difficiles.
Les formules de quadrature de Riemann (FR) et la formule de quadrature trapézoïdale (FT) seront examinées en premier, suivie par une comparaison avec la méthode qu'on a proposée.

Mots-clés : Intégration numérique, Optimisation par Essaim de Particules, Formule de Riemann, Formule des trapèzes.

Remerciement

Je tiens tout d'abord à remercier Dr KHELIL Naceur, maître de conférences à l'Université de Biskra, qui m'a donné ce sujet et m'a encadré pendant mon Master. Il a toujours été à mon écoute et son point de vue complémentaire est souvent été très utile.

Je remercie sincèrement le Dr. Rajeh Fouazia, de m'avoir fait l'honneur de présider le jury. Je suis très reconnaissant envers Monsieur Zeroug Abdelhamid, maître de conférences à l'Université de Biskra d'avoir manisfesté de l'intérêt pour mon travail en acceptant de participer à ce jury. Leurs remarques et commentaires constructifs m'ont permis d'en améliorer le manuscrit. Enfin, que toutes les personnes qui ont partcipé de prés ou de loin à la réalisation de ce travail trouvent içi l'expression de mes sincères remerciements. Mes derniers remerciements vont à mes parents et mes frères Amine, Akram, Chaouki et Mouad.

Table des matières

Liste des tableaux

Table des figures

Chapitre 1

Introduction générale

1.1 Motivation

Ce mémoire est basé essentiellement sur le chapitre 'book' (1) de N. Khelil, L. Djerou and M.Batouche "**Numerical Integration using swarm intelligence techniques**", pp 171-181 , DOI : 10.4018/978-1-4666-1830-5.ch010, http ://www.igi-global.com /book/ multidisciplinary-computational-intelligence-techniques/62627, dont l'objectif principal est l'évaluation numérique des intégrales

$$I = \int_a^b f(x)dx \tag{1.1}$$

qui sont impossibles ou difficiles à évaluer analytiquement. L'intégration numérique est également essentiel pour l'évaluation des intégrales des fonctions discrètes, ces fonctions qu'on rencontre souvent dans les solutions numériques des équations différentielles ou à partir des données expérimentales prises dans des intervalles discrets. Dans le calcul, une intégrale est une surface (signée) qui peut être approchée par la somme de petites surfaces, comme des rectangles par exemple.
On commence par le choix d'une partition de X qui subdivise $[a, b]$:

$$X = \{a \leq x_0 < x_1 < .... < x_{N-1} < x_N \leq b\}.$$

La valeur $(x_i - x_{i-1})$ obtenue du sous-intervalle $[x_{i-1}, x_i]$ détermine la largeur de chaque rectangle, en ce qui concerne la longueur, on peut prendre l'évaluation de n'importe quel point de cet intervalle, c-à-dire $f(x_i^*)$ où $x_i^* \in [x_{i-1}, x_i]$ (Figure 1.1).

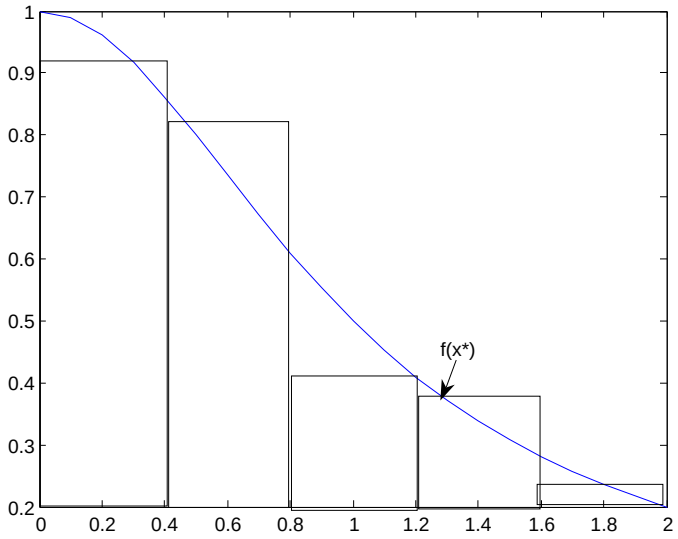

FIGURE 1.1 – Sommation

L' approximation qui en résulte est :

$$\int_a^b f(x)\,dx \approx \sum_{i=1}^{N} f(x_i^*)(x_i - x_{i-1}) \tag{1.2}$$

Dans ce mémoire, nous proposons un choix de $x_i^* \in [x_{i-1}, x_i]$ en utilisant l'optimisation par essaim de particules.

1.2 Organisation du mémoire

La partie principale de ce mémoire est composée de quatre chapitres. Le mémoire comporte également, un résumé, une introduction et une annexe.

Ce document est organisé comme suit :
Le chapitre 2 présente une approche d'intelligence collective permettant d'approximer une intégrale définie. L'approche retenue est l'optimisation par essaim de particules dont l'idée directrice est la simulation du comportement collectif des oiseaux à l'intérieur d'une nuée. La Section 2.1 rappelle l'origine de l'optimisation par essaim de particules et l'algorithme PSO que nous utiliserons tout au long de ce mémoire. La Section 2.4 passe en revue un exemple de calcul du maximum d'une fonction trés simple.
Le chapitre 3 rappelle les méthodes d'intégration les plus simples et les plus utilisées.
Le chapitre 4, s'occupe d'une nouvelle tentative que nous avons entreprise, concernant l'intégration numérique, et qui allait s'avérer comme fructueuse, cette tentative est basée sur l'application de l'optimisation par essaim de particules (PSO) dans l'estimation des approximations, c'est-à-dire on a utilisé les points PSO pour calculer une intégrale définie pour améliorer qualitativement les approximations. Cette nouvelle approche est examinée et évaluée avec un exemple illustratif.
La conclusion récapitule enfin nos contributions.

Chapitre 2

Optimisation par Essaim de Particules (PSO)

2.1 Origines

Plusieurs scientifiques ont créé des modèles interprétant le mouvement des vols d'oiseaux et des bancs de poissons. Plus particulièrement, Reynolds (2) et Heppner et al. (3) ont présenté des simulations sur un vol d'oiseaux. Reynolds était intriguépar l'aspect esthétique du déplacement des oiseaux en groupe et Hepnner, un zoologue, était intéressé à comprendre les règles permettant à un grand nombre d'oiseaux de voler en groupe : soit de voler sans se heurter, de changer soudainement de direction, de s'écarter et de se rapprocher de nouveau.
Cette étude a grandement inspiré le développement de l'algorithme PSO. En effet, lors de simulations des modèles mathématiques décrivant les vols d'oiseaux, Wilson (4) a suggéré que ces types de modèles pourraient très bien s'appliquer à la recherche de points caractéristiques dans un espace de recherche. Sa réflexion se base sur le fait que, lors de l'installation d'une mangeoire à oiseaux dans une cour, même si aucun oiseau ne l'a jamais visitée, après quelques heures de patience un grand nombre d'oiseaux viendront y manger. Lors des simulations de Wilson, la volée d'oiseaux cherchait une mangeoire dans un espace donné et finissait par dé couvrir son emplacement. En utilisant les algorithmes de modélisation de Heppner (3) et de Reynolds (2), et en modifiant le modèle mathématique de Wilson (4), Kennedy et Eberhart (5) ont transformé le tout en un vol d'oiseaux cherchant la « mangeoire » la plus grosse dans un lot de mangeoires contenues dans une région prédéterminée. L'algorithme d'optimisation PSO a ainsi vu le jour.

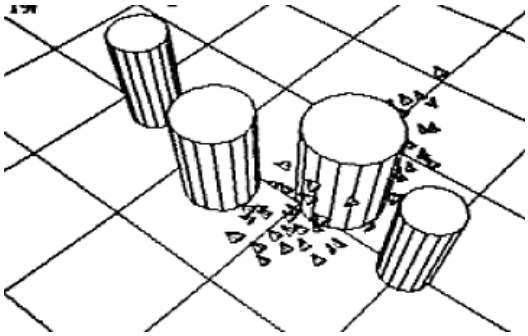

FIGURE 2.1 – Les Boids de Reynolds. Adapté de [Reynolds2001]

2.2 Formalisation et programmation

Le modèle PSO est un concept de la simulation de particules, et a été proposé par Eberhart et Kennedy (5). Basé sur une description mathématique du comportement social d'essaims, il a été montré que cet algorithme peut calculer efficacement de trés bonnes solutions à un certain nombre de situations compliquées tel que, par exemple, les problèmes de l'optimisation statique, l'optimisation topologique et autres (6)-(8)-(9)-(10). Dés lors, plusieurs variantes du PSO ont été développées (11)-(12)-(13)-(14)-(15)-(16)-(17).Il a été montré que la question de convergence de l'algorithme PSO est garantie implicitement si les paramètres sont sélectionnés adéquatement (18)-(19)-(8). Plusieurs variantes de problèmes de résolution commencent avec des simulations par ordinateur afin de trouver et analyser les solutions qui n'existent pas analytiquement ou spécifiquement ceux qui ont été prouvées intraitables théoriquement.
Le traitement de l'essaim de particule suppose une population d'individus conçue comme vecteurs de valeurs - particules, et quelque iterative classe de leur domaine d'adaptation doit être établi. Il est supposé que ces individus ont un comportement social qui implique que la capacité de conditions sociales, par exemple, l'interaction avec le voisinage, est un processus important pour trouver de bonnes solutions à un problème donné avec succès.
La version de base de l'algorithme PSO peut facilement être formalisé et programmée. L'algorithme maintient une population de particules, o ù chaque particule représente une solution potentielle d'un problème d'optimisation. Soit S la taille de l'essaim. L'espace de recherche est de dimension n. La position courante d'une particule i dans cet espace à l'itération k est donnée par un vecteur $x_i(k)$, à n composantes, donc. Sa vitesse courante est $v_i(k)$. La meilleure position trouvée jusqu'ici par cette particule est donnée par un vecteur $y_i(t)$. Enfin, la meilleure de celles trouvées par les informatrices de la particule est indiquée par un vecteur $\hat{y}(k)$.

$$\hat{y}(k) \in \{y_0(k), y_1(k), \ldots\ldots, y_s(k)\}$$

tel que

$$f(\hat{y}(k)) = min\{f(y_0(k)), f(y_1(k)), \ldots\ldots, f(y_s(k))\},$$

pour le modèle (global) gbest, ou

$$\hat{y}(k) \in \{y_{i-l}(k), y_{i-l+1}(k), \ldots\ldots, y_i(k), ..., y_{i+l}(k)\}$$

tel que

$$f(\hat{y}(k)) = min\{f(y_{i-l}(k)), f(y_{i-l+1}(k)), \ldots\ldots, f(y_i(k)), ..., f(y_{i+l}(k))\}$$

pour le modèle (local) lbest. où le symbole f désignera la fonction objective a minimiser. Donc, la différence entre les deux algorithmes est basée sur l'ensemble de particules (informatrices) avec lesquelles une particule donnée inter-réagira directement, où le symbole $\hat{y}$ est employé pour représenter cette interaction. L'équation de mise à jour pour la meilleure position personnelle est présentée dans l'équation suivante :

$$x_i(k+1) \quad = \begin{cases} y_i(k) & si \quad f(x_i(k+1)) \succeq f(y_i(k)) \\ y_i(k+1) & si \quad f(x_i(k+1)) < f(y_i(k)) \end{cases} \tag{2.1}$$

Avec ces notations, les équations de mouvement d'une particule i sont , pour chaque dimension j, à l'itération $(k+1)$:

$$v_{ij}(k+1) = w_j v_{ij} + c_1 r_{1j}[(y_{ij}(k) - x_{ij}(k)] + c_2 r_{2j}[(\hat{y}(k) - x_{ij}(k)] \tag{2.2}$$

$$x_{ij}(k+1) = v_{ij}(k+1) + x_{ij}(k) \tag{2.3}$$

où $x_{ij}(k+1), v_{ij}(k+1)$ sont respectivement $j^{\text{ème}}$ composante de la position et le vecteur vitesse de la particule i à l'itération $k+1$, $c_1$et c_2 sont les coefficients d'accélération pour chaque terme exclusivement placé dans l'intervalle $2-4$,w_j est le poids d'inertie avec sa valeur qui s'étend de 0.9 à 1.2, tandis que r_{1j}, r_{2j} sont des nombres aléatoires uniformes entre 0 et 1. Le rôle d'un choix approprié du poids d'inertie est important pour le succès du PSO. Dans le cas général, au début il peut être mis égal à sa valeur maximale et nous le diminuerons progressivement si la meilleure solution n'est pas atteinte.Trop souvent, dans la relation w_j, est remplacé par $\frac{w_j}{\sigma}$, où σdénote le facteur de constriction qui contrôle la vitesse des particules.
Pour réaliser son prochain mouvement,chaque particule combine trois tendances : suivre sa vitesse propre, revenir vers sa meilleure performance, aller vers la meilleure performance de ses informatrices.

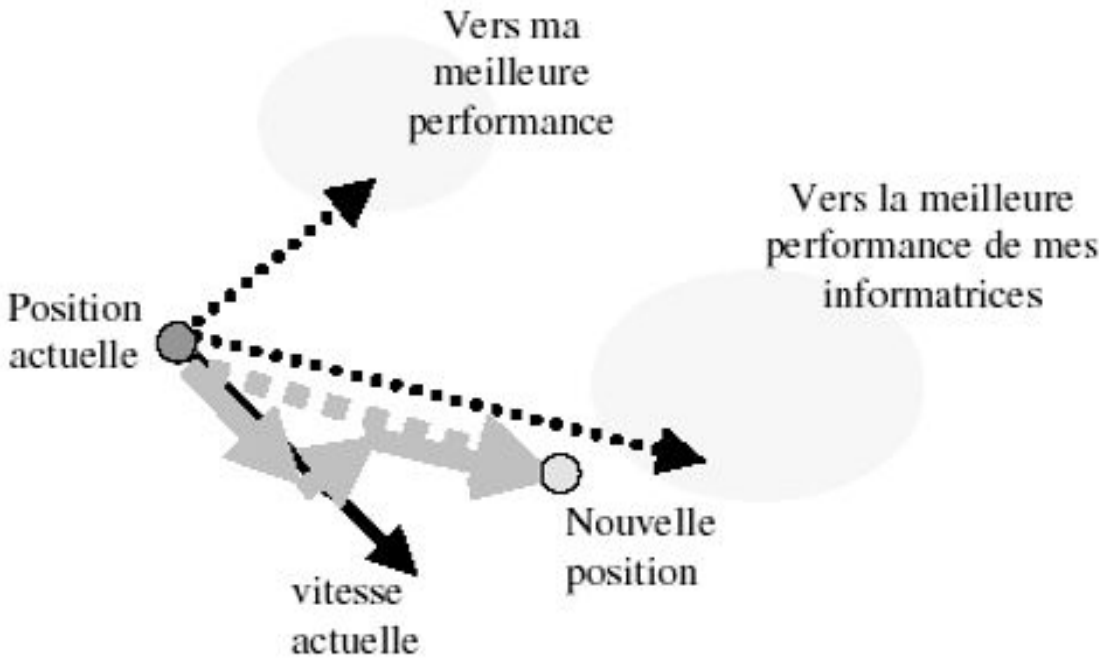

FIGURE 2.2 – Schéma de principe du déplacement d'une particule

2.3 Algorithme PSO

Le but de l'algorithme PSO est d'optimiser une fonction continue dans un espace donné. Dans la majorité des cas, l'algorithme d'optimisation recherche le maximum ou le minimum global de l'espace de recherche. Voici la description des étapes de l'algorithme PSO :

Dans cette section, l'algorithme sera présenté pas par pas, et le programme détaillé et appliqué à une fonction simple sera donné par la suite (voir 2.4)

2.3.1 Algorithme

Etape0 Initialiser les paramètres, les postions 'x_i et les vitesses 'v_i' des particules de manière aléatoire.

Etape1 Calculez la matrice des meilleurs positions et les valeurs de la fonction correspondantes.

Etape2 Si le critère d'arrêt est vérifié, alors l'algorithme se termine. S'il ne l'est pas, une nouvelle itération commence. Le critère d'arrêt correspond généralement à un nombre d'itérations prédéfinies, mais on peut également spécifier un critère d'arrêt en fonction de la meilleure valeur de qualité $f(x_i)$obtenue pour l'ensemble des particules. Pour toutes les particules de la population, exécuter les Étapes 1 à 5.

Étape 3 Mettre à jour la vitesse de déplacement $v_i(k+1)$ de la particule i. Cette mise à jour tient compte de la vitesse précédente de la particule $v_i(k)$, de sa position présente $x_i(k)$, de la position de la meilleure qualité $y_i(k$)obtenue par cette particule ainsi que de la position de la meilleure qualité globale $\widehat{y}$ obtenue par la population. De plus, deux paramètres, $r_1$et r_2 , sont utilisés pour ajuster l'importance des termes $(y_i(k) - x_i(k))$ et $(\widehat{y}(k) - x_i(k))$de l'équation de mise à jour de la vitesse (équation 2.2).

Etape4 Mettre à jour la position $x_i(k)$de la particule i (équation 2.3). Cette mise à jour tient compte de la position précédente de la particule $x(k)$ ainsi que de la nouvelle vitesse $v_i(k+1)$ calculée à l'étape 3.

Etape5 Revenir à l'etape 2.

Remarque Contrairement à bien d'autres heuristiques qui restent purement expérimentales, il existe une analyse mathématique précisant les conditions de convergence et le choix des paramètres (19) et annexe(A).

2.4 Fonction test utilisée

Nous achevons ce chapitre par un exemple simple qui nous permet de mieux comprendre le fonctionnement de l'aalgorithme ; supposons qu'on veut maximiser la fonction

$$F_{obj} = 15x - x^2$$

sur l'intervale

$$0 \leq x \leq 15$$

Le résultat est $x = 7.5$

clc ;
clear all ;
commentaire ; objectif function
$f = inline('15 * x - x^{2\prime}, ' x')$;
$liminf = 0$;
$limsup = 15$;
commentaire ; Initialization Parameters
$iteration = 50$;
$swarmsize = 20$;
$c1 = 2.0; c2 = 2.0$;
$w = 0.5$;
commentaire ; Initial swarm position
for $k = 1 : swarmsize$
$swarm(k, 1) = rand * (limsup - liminf)$;
$Pbest(k, 1) = swarm(k, 1)$;
$Evaluation(k) = f(Pbest(k, 1))$;
$v(k, 1) = 0$;
end
for i=1 :iteration
$[fbest, best] = max(Evaluation)$;
$gbest = swarm(best, 1)$;
for k=1 :swarmsize
v(k,1)=0.5*v(k,1)+c1*rand*(Pbest(k,1)-swarm(k,1))+c2*rand*(gbest-swarm(k,1));
newpos = swarm(k,1)+v(k,1);
if newpos > liminf δ $newpos < limsup$
$swarm(k, 1) = newpos$;
if $f(newpos) > f(Pbest(k, 1))$
$Pbest(k, 1) = newpos$;
$Evaluation(k) = f(Pbest(k, 1))$;
end
end
end
end
end
gbest

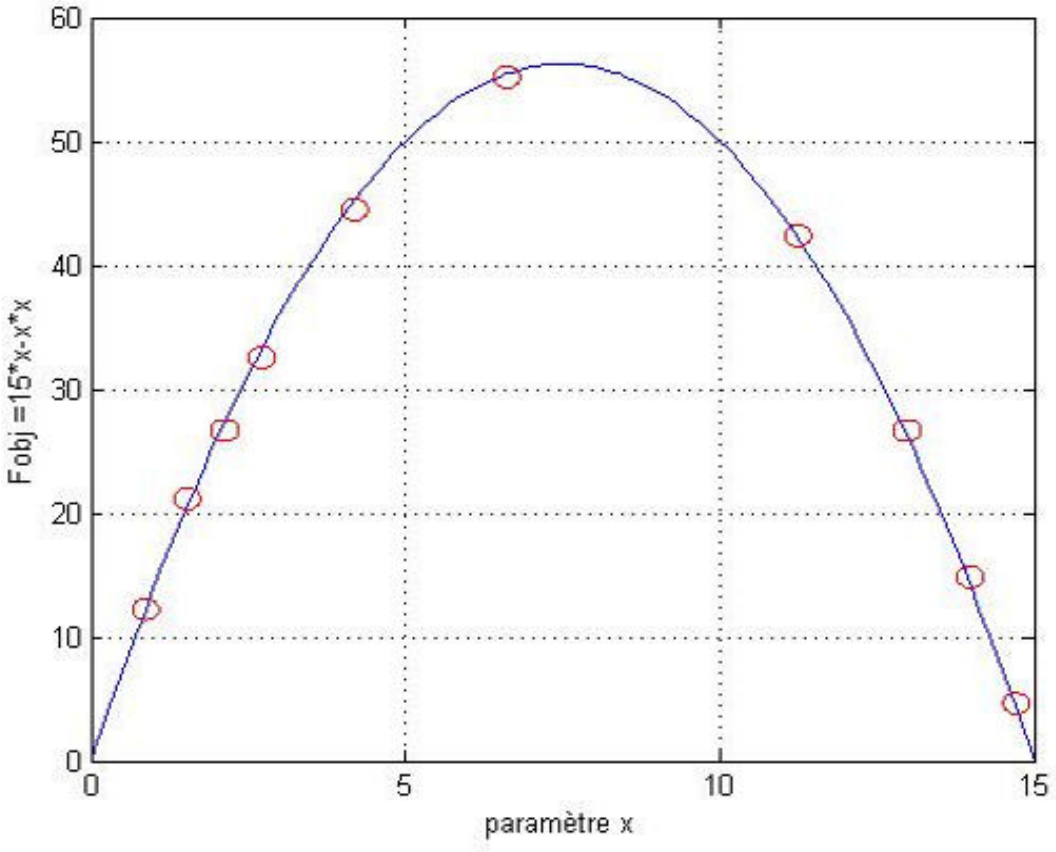

FIGURE 2.3 – Application du PSO sur un exemple simple

Chapitre 3

Quelques méthodes d'intégration numérique

Rappelons que l'objectif de ce mémoire est l'évaluation numérique des intégrales

$$I = \int_a^b f(x)dx$$

qui sont impossibles ou difficiles à évaluer analytiquement. Dans le calcul, une intégrale est une surface (signée) qui peut être approchée par la somme de petites surfaces, comme des rectangles par exemple.
On commence par le choix d'une partition de X qui subdivise $[a, b]$:

$$X = \{a \leq x_0 < x_1 < .... < x_{N-1} < x_N \leq b\}.$$

La valeur $(x_i - x_{i-1})$ obtenue du sous-intervalle $[x_{i-1}, x_i]$ détermine la largeur de chaque rectangle, en ce qui concerne la longueur, on peut prendre l'évaluation de n'importe quel point de cet intervalle, c-à-dire $f(x_i^*)$ où $x_i^* \in [x_{i-1}, x_i]$.
L' approximation qui en résulte est :

$$\int_a^b f(x)\,dx \approx \sum_{i=1}^{N} f(x_i^*)(x_i - x_{i-1})$$

Et pour utilser cette dernière formule afin d'approcher une intégrale, on devera spécifier un nombre x_i^* de l'intervalle $[x_{i-1}\ x_i\ ]$, plusieurs choix sont possibles :

3.1 Formule de Riemann (FR)

Il y a deux possibilités, on prend x_i^* comme étant la borne inférieure de $[x_{i-1}\ x_i\ ]$ (Figure 3.1), alors

$$\int_a^b f\,(x)\,dx \approx \sum_{i=1}^{N} f(x_{i-1})(x_i - x_{i-1})$$

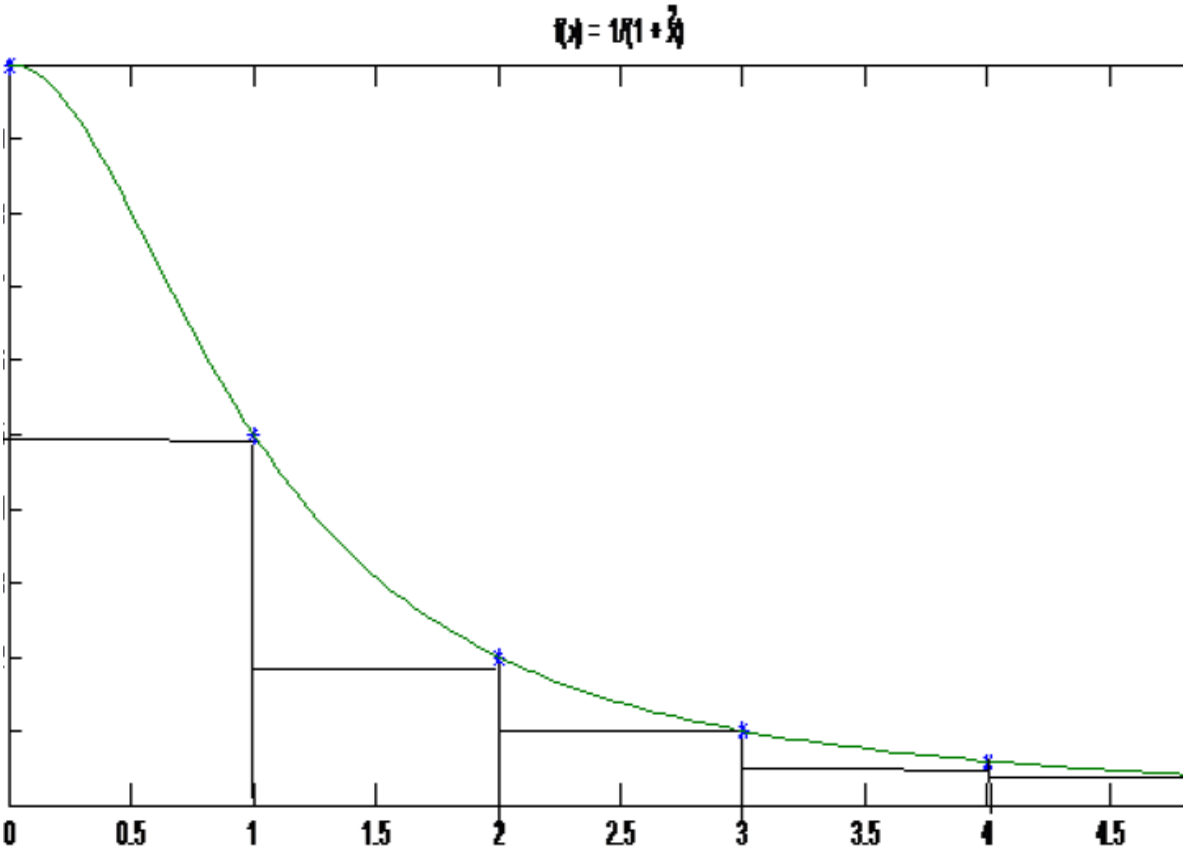

FIGURE 3.1 – Formule de Riemann Gauche, FRG

ou bien x_i^* comme étant la borne supérieure de $[x_{i-1}\ x_i\ ]$ (Figure 3.2), alors

$$\int_a^b f\,(x)\,dx \approx \sum_{i=1}^{N} f(x_i)(x_i - x_{i-1})$$

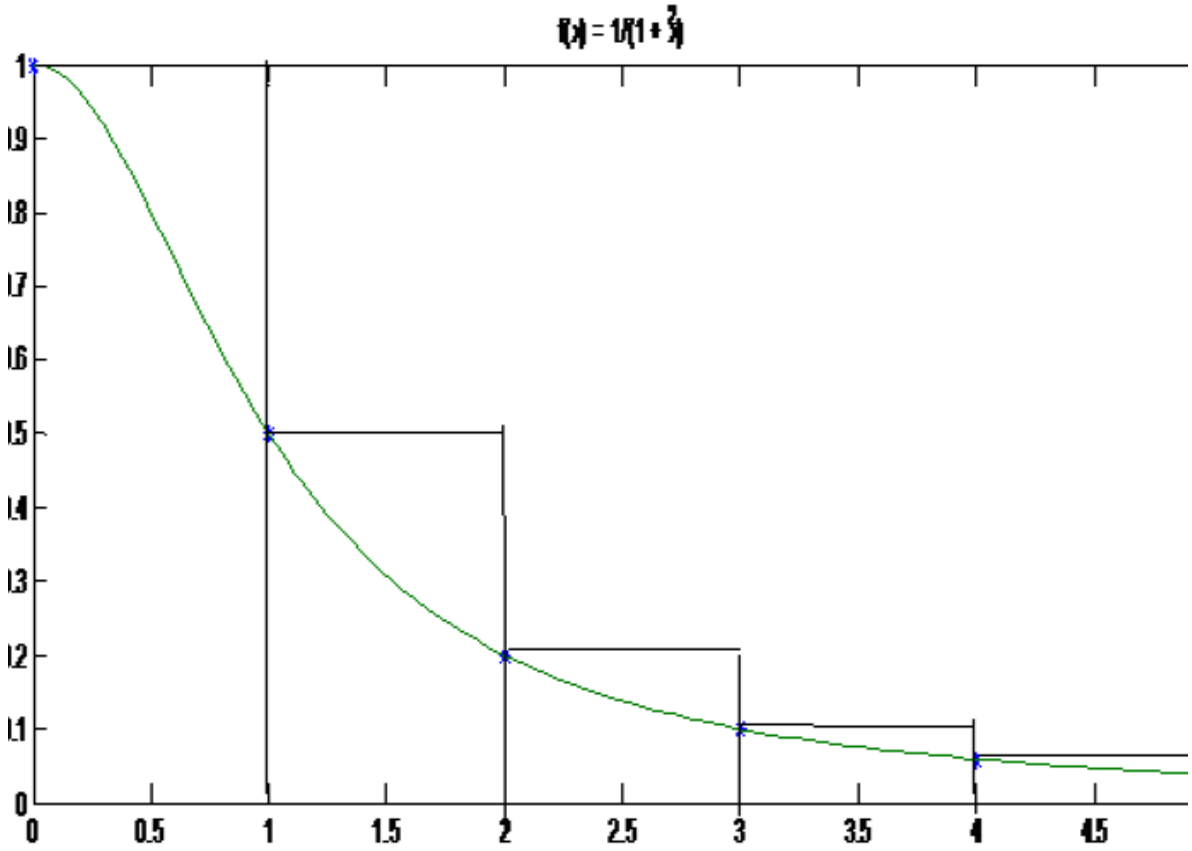

FIGURE 3.2 – Formule de Riemann Droite, FRD

3.2 Formule des Trapèzes (FT)

Si on prend x_i^* comme étant la moyenne des borne supérieure et inférieure de $[x_{i-1}\ x_i\ ]$ (Figure 3.3), alors

$$\int_a^b f\,(x)\,dx \approx \sum_{i=1}^{N} f(\tfrac{x_{i-1}+x_i}{2})(x_i - x_{i-1})$$

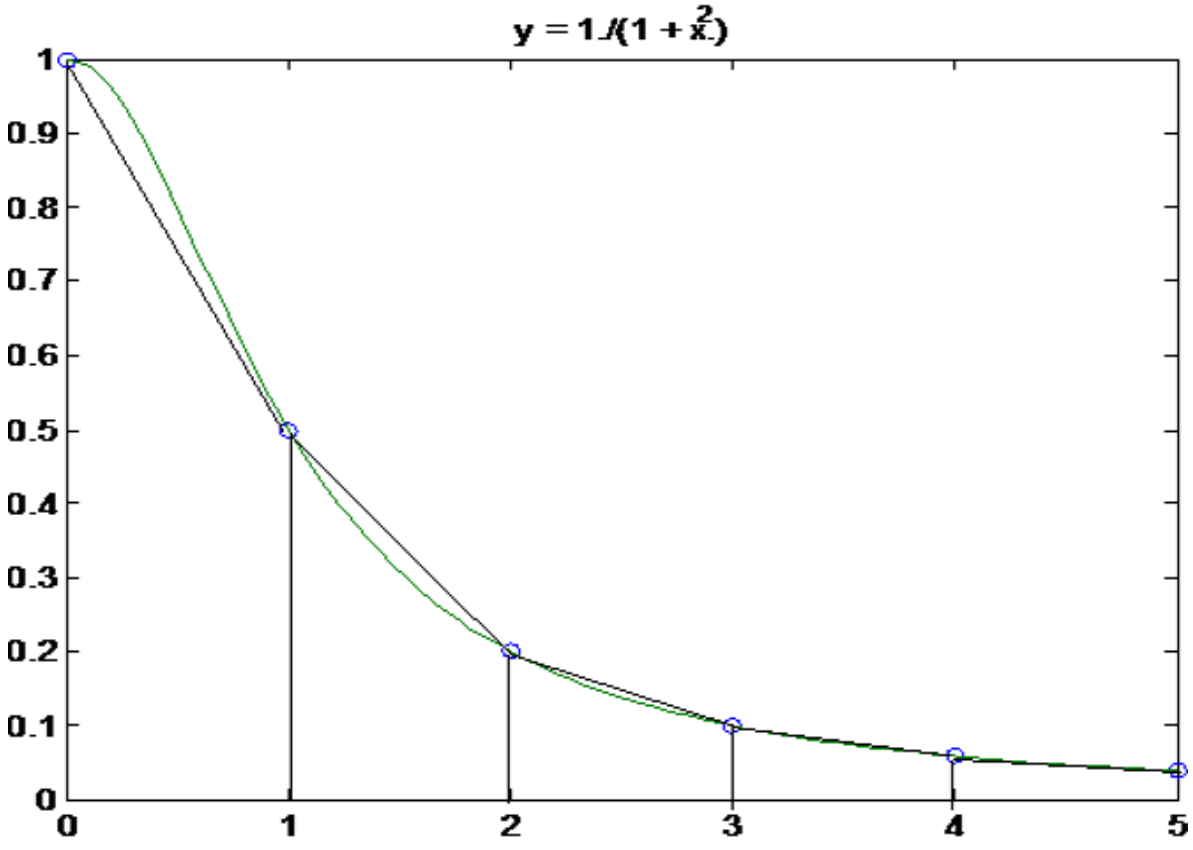

FIGURE 3.3 – Formule des Trapèzes, FT

Chapitre 4

Méthode proposée

4.1 PSO et le calcul d'intégrale

Ce chapitre étend la discussion sur l'intégration, mais la considère comme étant un perspective de l'optimisation par essaim de particule. Calculer une intégrale est répété comme un problème d'optimisation et devient par défaut un problème d'optimisation traditionnel.

Comme on le sait, que pour une fonction f bornée sur l'intervalle$[a, b]$ est intégrable, si et seulement si :

$$\forall \varepsilon \succ 0, \exists\ X\ (partition) \text{ de } [a, b] \text{ tel que } \sum_{i=1}^{N}(L_i - l_i)(x_{i-1} - x_i) \leq \varepsilon$$

avec $L_i = sup_{[x_{i-1}\ x_i]}f(x)$ et $l_i = inf_{[x_{i-1}\ x_i]}f(x)$. Donc, le problème qui se pose est de trouver $x_i^* \in [x_{i-1}\ x_i]$ pour lesquels la quantité $\sum_{i=1}^{N}(L_i - l_i)(x_{i-1} - x_i)$ sera la plus petite possible (Figure 4.1). Par conséquent,

$$I(f) = \int_a^b f(x)dx \approx \sum_{i=1}^{N}(f(x_i^*)(x_i - x_{i-1}) \tag{4.1}$$

La qualité du résultat dépend du choix de x_i^*. Comment choisir ces points ? Faire ceci, conceptuellement, nous aimerions sélectionner ces points en ayant recours à l'optimisation par essaim de particules (PSO)(2.3).

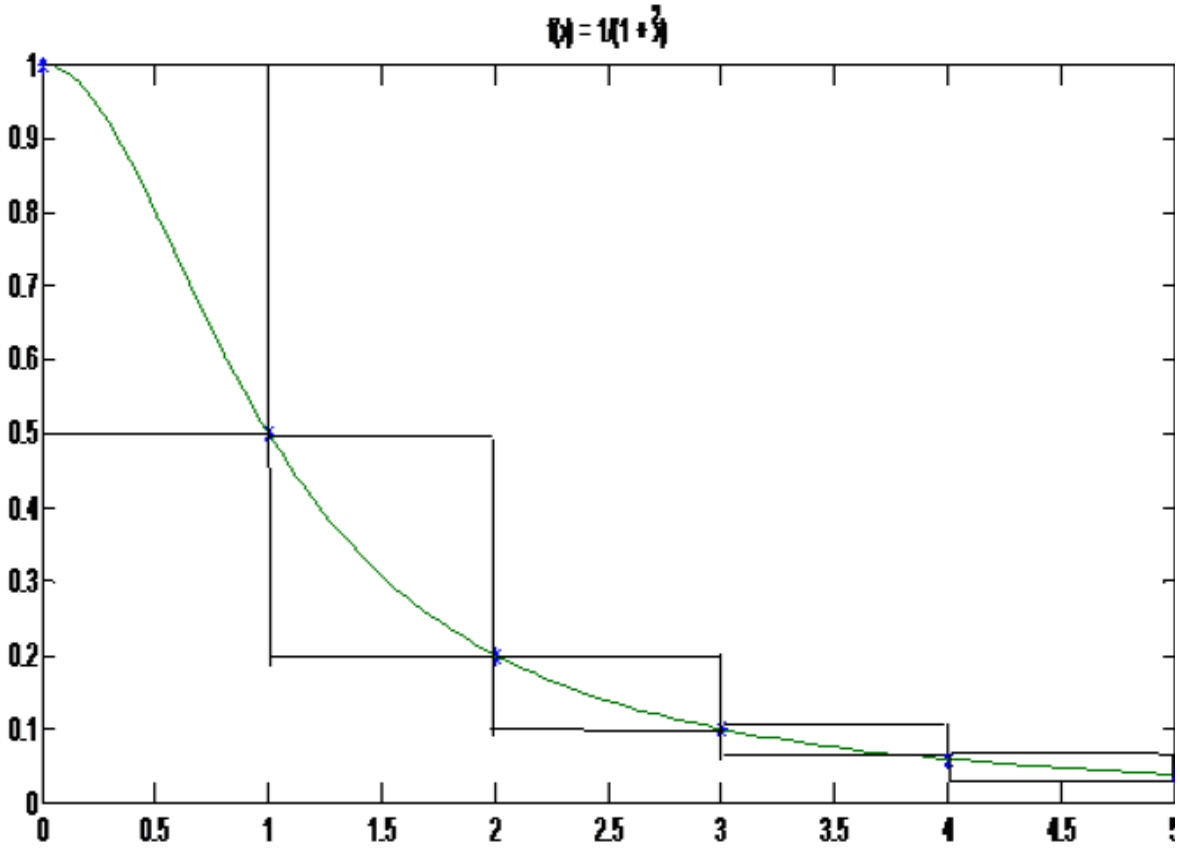

FIGURE 4.1 – Erreur

4.2 Algorithme proposé

Cet algorithme est accompli avec succés avec les étapes suivantes (20)-(21)-(22) :

1. Poser la dimension de l'éspace égale à N et la taille de l'essaim à s :
2. Initialiser le nombre d'itération k (en général est égale à zéro).
3. Placer chaque agent entre a and b; arrange les selon une suite croissante respectivement. Alors il y a $N+2$ noeuds et $N+1$ segments, puis évaluer la fonction objectif en $N+1$ segments. La fonction objectif est définie par :

$$f(i) = \sum_{j=1}^{N+1} (L_{ij} - l_{ij})(x_{ij} - x_{ij-1}) \ . \tag{4.2}$$

   Si la condition de terminaison est atteint, alors stop (la condition de terminaison est définie par : choisir un ε infiniment petit, si le minimum de la fonction objectif est inférieur ε alors stop), choisir la solution optimum $X_*(a = x_{*0} < x_{*1} < \cdots < x_{*N} < x_{*N+1} = b)$ et alors

$$\int_a^b f(x)dx \simeq \sum_{j=1}^{N+1} f(x_j^*)(x_j - x_{j-1}) \ . \tag{4.3}$$

   Sinon, continue.
4. Chaque agent doit etre mis à jour en appliquant son vecteur vitesse et sa position antérieure en utilisant équation (2.2).
5. Répéter les étapes précedentes (3,4 and 5) jusqu'au momennt où le critère de convergence est atteint.

L'algorithme PSO est appliqué, avec les paramètres du Tableau (Table 4.1).

TABLE 4.1 – Parameters Setting pour généré l'algorithme PSO pour cette étude

	Exemple
Taille de la population	21
Nombre d'itérations	500
Coefficients d'acceleration : c_1 and c_2	0.5
poids d'inertie	1.2 to 0.4
Erreur désirée	10^{-5}

Afin de valider la feasibilité de cet algorithme pour l'intégration numérique, voici un exemple illustratif.

4.3 Illustration

Cet exemple peut être vu comme un cas typique qui fournit une bonne illustration. Notons que, la précision des résultats dépend manifestement de la réussite des particules dans l'essaim pour trouver les meilleurs points. Pour faciliter l'inteprétation, les résultats numériques évaluées par l'algorithme PSO, et ceux obtenus par la formule de Riemann et des trapèzes ont été comparés Tableau Table (4.2).
Calculons l'intégrale $\int_0^1 400x(1-x)e^{-2x}dx$ dont la valeur exacte est 27.0671

4.3.1 Approximation avec la formule de Riemann (FR)

Approximons l'intégrale avec la formule de Riemann avec $N = 5$.
Si on divise [0 1] en cinq intervalles égaux et choisissons le point comme étant simplement la borne supérieure de chaque intervalle $[x_{i-1}\ x_i]$ (Figure 4.2).
Alors

$$x_1^* = 0.2,\ x_2^* = 0.4,\ x_3^* = 0.6,\ x_4^* = 0.8,\ x_5^* = 1.0.$$

Et par conséquent :

$$\begin{aligned}\int_0^1 400x(1-x)e^{-2x}dx &= \sum_1^5 (x_i - x_{i-1})f(x_i^*) \\ &= (0.2)[f(0.2) + f(0.4) + f(0.6) + f(0.8) + f(1.0)] \\ &= 25.5744 \end{aligned} \tag{4.4}$$

avec une erreur 1.4927

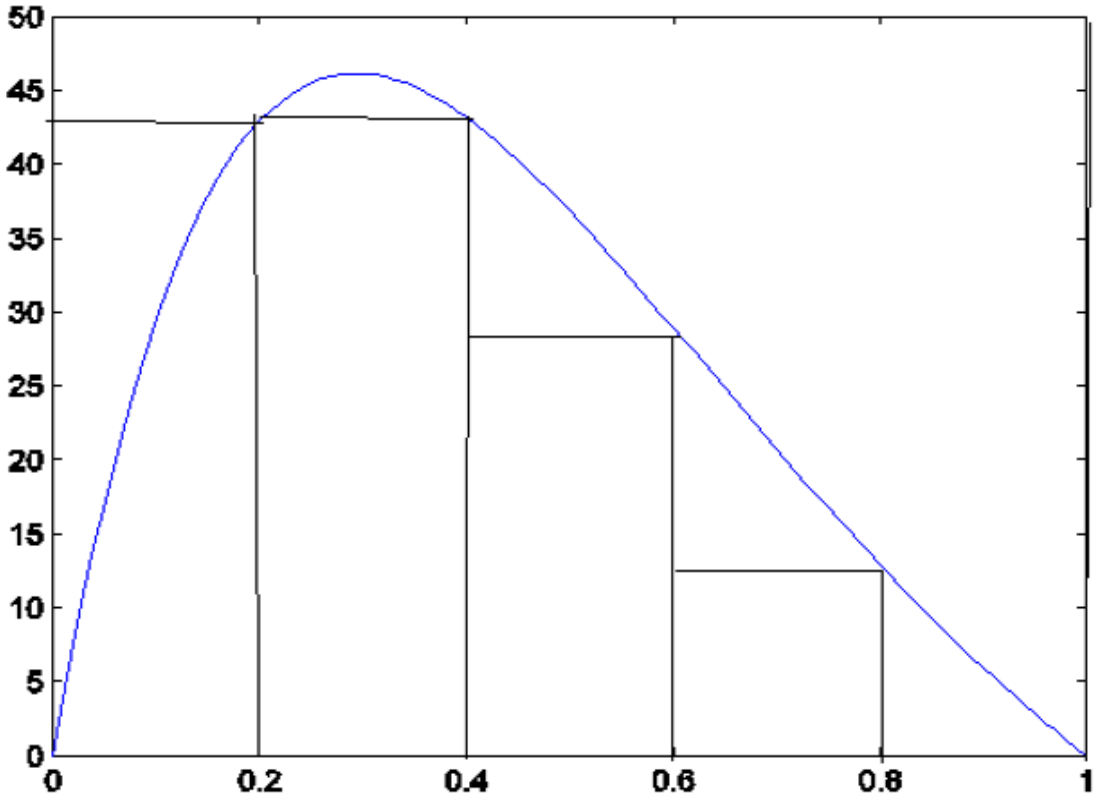

FIGURE 4.2 – Intégration avec la FRD

4.3.2 Approximation avec la formule des Trapèzes (FT)

Approximons l'intégrale avec la formule des Trapèzes avec $N = 5$.
Si on divise [0 1] en cinq intervalles égaux. Alors (Figure 4.3).

$$x_1^* = 0.2,\ x_2^* = 0.4,\ x_3^* = 0.6,\ x_4^* = 0.8,\ x_5^* = 1.0$$

Et par conséquent :

$$\begin{aligned}\int_0^1 400x(1-x)e^{-2x}dx &= \sum_1^5 (x_i - x_{i-1})f(x_i^*) \\ &= (0.2/2)[f(0.2) + 2 * f(0.4) + 2 * f(0.6) + 2 * f(0.8) + f(1.0)] \\ &= 21.2844 \end{aligned} \tag{4.5}$$

avec une erreur 5.7827

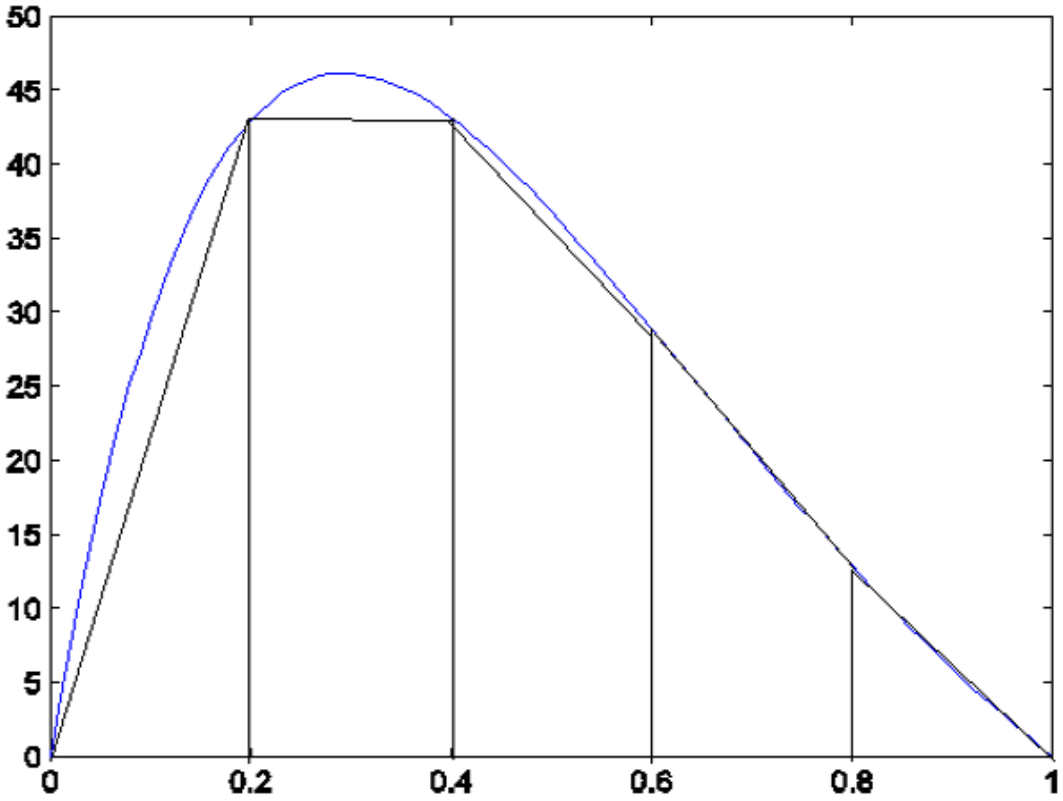

FIGURE 4.3 – Intégration avec la FT

4.3.3 Approximation avec la formule PSO (FPSO)

Approximons l'intégrale avec la formule PSO, avec $N = 5$.
Si on divise [0 1] en cinq intervalles égaux et choisissons le point $(x_i^*)_i$ ceux qui minimisent la quantité $f(i) = \sum_{j=1}^{5}(L_{ij} - l_{ij})(x_{ij} - x_{ij-1})$ avec PSO (Figure 4.4). Alors

$$x_1^* = 0.2,\ x_2^* = 0.4,\ x_3^* = 0.6,\ x_4^* = 0.8,\ x_5^* = 1.0$$

Et par conséquent ;

$$\int_0^1 400x(1-x)e^{-2x}dx = \sum_1^5 (x_i - x_{i-1})f(x_i^*)$$
$$= (0.2)[f(0.2) + 2 * f(0.4) + 2 * f(0.6) + 2 * f(0.8) + f(0.877)]$$
$$= 27.0680 \quad (4.6)$$

avec une erreur $9.4405e^{-004}$.

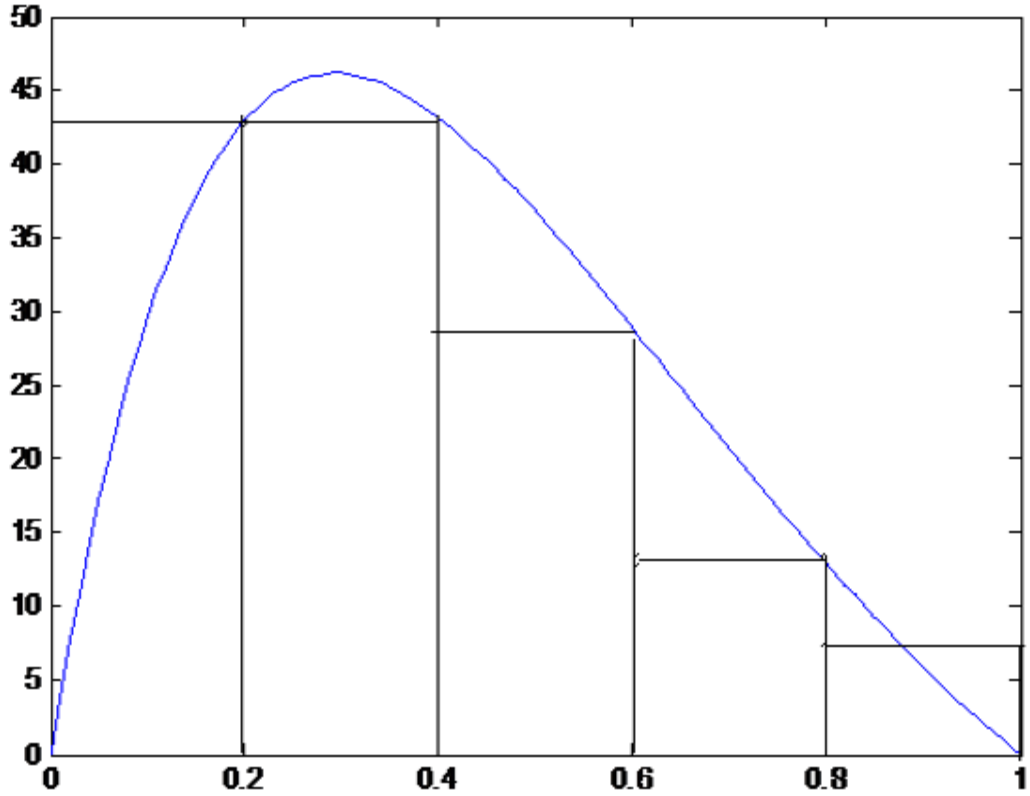

FIGURE 4.4 – Intégration avec la FPSO

4.4 Diagnostique de l'erreur

La complexité computationelle (CC) est basée sur le nombre de segments, dans l'exemple, pour un nombre de segments égale à 5 l'algorithme FPSO est netement supérieur aux méthodes conventionnelles Tableau (Table 4.2).

TABLE 4.2 – Diagnostique de l'erreur

N.de segments	la méthode	h	x^*	Valeur	Erreur
5	FR	h=0.2	[0.2 0.4 0.6 0.8 1.0]	25.5744	1.4927
5	FT	h/2=0.1	[0.2 0.4 0.6 0.8 1.0]	21.2844	5.7827
5	FPSO	h =0.2	[0.2 0.4 0.6 0.8 0.877]	27.0680	0.0009

Conclusion

Les méthodes d'intégration numérique, qui fournissent facilement une solution, présentent l'inconvénient dans certains cas de donner une qualité médiocre : la différence entre la valeur exacte et son approximation (par une formule d'intégration) peut être grande. Dans ce mémoire nous avons montré, qu'il est possible de remédier à cela, en utilisant l'optimisation par Essaim de particules ; en transformant le problème d'intégration en un problème d'optimisation. A titre d'application, on a calculé les points d'intégration avec PSO, à la place des points utiliser par les méthodes de Riemann ou la méthode des trapèzes pour améliorer qualitativement, les solutions. Cette nouvelle approche est maniée avec soin et a été testé et comparée avec les méthodes ci-citées sur un exemple.

Bibliographie

[1] Khelil N., Djerou L. and Batouche M. : Numerical Integration using swarm intelligence techniques,pp 171-181, DOI :10.4018/978-1-4666-1830-5.ch010, http ://www.igi-global.com/book/multidisciplinary-computational-intelligence-techniques/62627 (2012)

[2] Reynolds, C.W. : Flocks, herbs and schools : a distributed behavioral model. Computer Graphics, 21(4), 25-34.(1987)

[3] Heppner F. and Grenander U. : stochastic nonlinear model for coordinated bird flocks. In S. Krasner, Ed., The Ubiquity of Chaos. AAAS Publications, Washington, DC, 233-238.(1990)

[4] Wilson, E.O. : Sociobiology : The new synthesis. Cambridge, MA : Belknap Press.(1975)

[5] Eberhart R.C. and Kennedy J. : A new optimizer using particles swarm theory. Sixth International Symposium on Micro Machine and Human Science, pp.39–43, Nagoya, Japan, (1995)

[6] Parsopoulos K. E. and Vrahatis M. N. : Modification of the Particle Swarm Optimizer for Locating all the Global Minima. Kurkova V. et al., eds., Artificial Neural Networks and Genetic Algorithms, Springer, New York, pp. 324-327,(2001)

[8] Parsopoulos K. E. et al. : Stretching technique for obtaining global minimizers through particle swarm optimization. Proc. of the PSO Workshop, Indianapolis, USA, pp. 22-29, (2001)

[8] Parsopoulos K. E. et al. : Objective function stretching to alleviate convergence to local minima. Nonlinear Analysis TMA 47, 3419-3424, (2001)

[9] Fourie P.C., and Groenwold A.A. : Particle swarms in size and shape optimization. Proceedings of the International Workshop on Multi-disciplinary Design Optimization, Pretoria, South Africa, August 7â10, pp.97â106, (2000)

[10] Fourie P.C. and Groenwold A.A. : Particle swarms in topology optimization. Extended Abstracts of the Fourth World Congress of Structural and Multidisciplinary Optimization, Dalian, China, June 4-8, pp.52-53, (2001)

[11] Eberhart R.C., et al. : Computational Intelligence PC Tools. Academic Press Professional, Boston, (1996)

[12] Kennedy J. : The behaviour of particles. Evol. Progr. VII, 581-587, (1998)

[13] Kennedy J. and Eberhart R. C. : Swarm Intelligence. Morgan Kaufmann Publishers, San Francisco, (2001)

[14] Shi Y. H. and Eberhart R. C. : Fuzzy adaptive particle swarm optimization. IEEE Int. Conf. on Evolutionary Computation, pp. 101-106, (2001)

[15] Shi Y. H. and Eberhart R. C. : A modified particle swarm optimizer. Proc. of the 1998 IEEE International Conference on Evolutionary Computation, Anchorage, Alaska, May 4-9, (1998)

[16] Shi Y. H. Eberhart and R. C. : Parameter selection in particle swarm optimization. Evolutionary Programming VII, Lecture Notes in Computer Science, pp. 591-600, (1998)

[17] Clerc M. : The swarm and the queen : towards a deterministic and adaptive particle swarm optimization. Proceedings of the 1999 IEEE Congress on Evolutionary Computation, Washington DC pp.1951-1957, (1999)

[18] Eberhart R.C., and Shi, Y. : Parameter selection in particle swarm optimization. in Porto, V.W., (1998)

[19] Cristian T. I. : The particle swarm optimization algorithm : convergence analysis and parameter selection. Information Processing Letters, Vol. 85, No. 6, pp.317-325, (2003)

[20] Zerarka A. and Khelil N. : A generalized integral quadratic method : improvement of the solution for one dimensional Volterra integral equation using particle swarm optimization. Int. J. Simulation and Process Modelling 2(1-2), 152-163, (2006)

[21] Zerarka A. , Soukeur A. , Khelil N. : The particle swarm optimization against the Runge's phenomenon : Application to the generalized integral quadrature method. International Journal of Mathematical and Statistical Sciences pp-171-176, 1 :3 (2009)

[22] Khelil N. et al. : Improvement of Gregory's formula using Particle Swarm Optimization. In the proceeding of International Conference on Computer and Applied Mathematics, pp.940-942, volume 58, (2009)

[23] van den Bergh F. : An Analysis of Particle Swarm Optimizers. PhD thesis, Department of Computer Science, University of Pretoria (2002).

Annexe A

PSO, Dérivation des Equations Explicites

Les équations de la mise à jour de l'algorithme PSO sont :

$$v_{ij}(k+1) = w_j v_{ij} + c_1 r_{1j}[(y_{ij}(k) - x_{ij}(k)] + c_2 r_{2j}[(\hat{y}(k) - x_{ij}(k)] \tag{A.1}$$

$$x_{ij}(k+1) = v_{ij}(k+1) + x_{ij}(k) \tag{A.2}$$

Pour simplifier la notation, la dérivation sera exécutée seulement avec une dimension, employant une particule simple. Cela nous permet de laisser tomber tous les deux indices i et j, désignons par x_k la valeur de x a l'itération k, plutôt que la valeur x associé à la particule k. Les composantes stochastiques seront aussi enlevées temporairement avec les substitutions $\phi_1 = c_1 r_{1j}$ et $\phi_2 = c_2 r_{2j}$.

La position de la particule sera considérée dans le temps discret seulement, aboutissant aux équations suivantes :

$$v_{k+1} = w v_k + \phi_1(y_k - x_k) + \phi_2(y_k - x_k) \tag{A.3}$$

$$x_{k+1} = v_{k+1} + x_k \tag{A.4}$$

La substitution de (A.3) dans (A.4), donne

$$x_{k+1} = x_k + wv_k + \phi_1(y_k - x_k) + \phi_2(\hat{y}_k - x_k) \tag{A.5}$$

Le regroupement des termes semblables donne

$$x_{k+1} = (1 - \phi_1 - \phi_2)x_k + \phi_1 y_k + \phi_2 \hat{y}_k + wv_k \tag{A.6}$$

Mais, de l'équation (A.4) on a : $x_k = v_k + x_{k-1}$, ce qui donne

$$v_k = x_k - x_{k-1} \tag{A.7}$$

remplaçant (A.7) dans (A.6) et regroupons les termes

$$x_{k+1} = (1 + w + \phi_1 - \phi_2)x_k + \phi_1 y_k + \phi_2 \hat{y}_k \tag{A.8}$$

Et ceci est une relation de récurrence non-homogène, qui peut être résolue en employant des techniques standard [125]. Cette relation de récurrence peut être écrite sous forme d'un produit matriciel

$$\begin{pmatrix} x_{k+1} \\ x_k \\ 1 \end{pmatrix} = \begin{pmatrix} 1 + w + \phi_1 - \phi_2 & -w & \phi_1 y_k + \phi_2 \hat{y}_k \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_{k+1} \\ x_k \\ 1 \end{pmatrix} \tag{A.9}$$

Le polynôme caractéristique de cette matrice est

$$x_{k+1} = (1 - \lambda)(w - \lambda(1 + w - \phi_1 - \phi_2) + \lambda^2) \tag{A.10}$$

qui possède une racine triviale $\lambda = 1.0$, et deux autres racines :

$$\alpha = \frac{1 + w - \phi_1 - \phi_2 + \gamma}{2} \tag{A.11}$$

$$\beta = \frac{1 + w - \phi_1 - \phi_2 - \gamma}{2} \tag{A.12}$$

avec

$$\gamma = \sqrt{(1 + w - \phi_1 - \phi_2)^2 - 4w} \tag{A.13}$$

Notons que α et β sont les deux valeurs propres de la matrice donnée dans l'équation (A.9). La forme explicite de la relation de récurrence (A.8)est donnée par :

$$x_k = c_1 + c_2\alpha^k + c_3\beta^k \tag{A.14}$$

Où c_1, c_2 et c_3 sont des constantes déterminées par les conditions initiales du système. Puisqu'il y a trois inconnues, un système de trois équations doit être construit pour trouver leurs valeurs. Les conditions initiales du PSO fournissent deux de ces conditions, x_0 et x_1, correspondant à la position de la particule aux pas de temps 0 et 1(c'est l'équivalent de la spécification de la position initiale x_0 et la vitesse initiale v_0. La troisième contrainte du système peut être calculée en employant la relation de récurrence Donc x_2 sera tiré de (A.8), $x_2 = (1 + w - \phi_1 - \phi_2)x_1 - wx_0 + \phi_1 y_k + \phi_2 \hat{y}_k$

De ces trois conditions initiales, le système

peut être résolu en employant l'élimination de Gauss, ce qui donne :

$$c_1 = \frac{\alpha\beta x_0 - x_1(\alpha+\beta) + x_2}{(\alpha-1)(\beta-1)}$$

$$c_2 = \frac{\beta(x_0 - x_1) - x_1 + x_2}{(\alpha-1)(\beta-1)}$$

$$c_3 = \frac{\alpha(x_1 - x_0) + x_1 - x_2}{(\alpha-1)(\beta-1)}$$

En employant la propriété $\alpha - \beta = \gamma$, ces équations peuvent être plus loin simplifiées pour donner

$$c_1 = \frac{\phi_1 y_k + \phi_2 \hat{y}_k}{\phi_1 + \phi_2} \tag{A.15}$$

$$c_2 = \frac{\beta(x_0 - x_1) - x_1 + x_2}{\gamma(\alpha - 1)} \tag{A.16}$$

$$c_3 = \frac{\alpha(x_1 - x_0) + x_1 + x_2}{\gamma(\beta - 1)} \tag{A.17}$$

Notez qu'et y_k et $\hat{y}_k$ dépendent du pas de temps k. Ces valeurs peuvent changer avec le pas de temps, ou ils peuvent rester constants pour des durées, selon la fonction objective et la position des autres particules. Chaque fois que l'un ou l'autre y_k et $\hat{y}_k$ change les valeurs de c_1,c_2 et c_3 doit être recalculé. Il est possible d'extrapoler la trajectoire d'une particule en maintenant, y_k et $\hat{y}_k$ constantes. Cela implique que les positions des autres particules restent fixées et que la particule ne découvre pas de meilleures solutions lui-même. Bien que ces équations aient été tirées sous la supposition de temps discret, aucune telle restriction n'est nécessaire. L'équation (A.14) peut être exprimée dans le temps continu aussi, aboutissant

$$x(t) = c_1 + c_2\alpha^k + c_3\beta^k \tag{A.18}$$

En employant les mêmes valeurs pour c_1,c_2 et c_3 comme tiré ci-dessus pour le cas discret. Pour plus d'informations voir (23).