LYCEE CARNOT

TIPE : Le TEMPS

Calcul de Fonctions usuelles

Théodon Léo – Abrieu Emilien – Malot Frédéric

2007

CLASSES PREPARATOIRES

Calcul de Fonction usuelles

I- Problématique de départ et position du problème

Exposition du problème, du sujet du TIPE, et définition du cadre de notre étude. Explication de la façon dont nous allons exposer notre travail, et nos éventuels résultats.

II- Remarques préliminaires

Remarques sur les limites que l'informatique impose à l'humain, et à la précision d'un calcul. Exemple du codage des flottants dans la mémoire informatique d'un ordinateur et erreur que commet ce dernier lors d'un calcul avec des flottants.

III- Etude de différents algorithmes

Exposition de deux types d'algorithmes. Pour chaque algorithme, on étudiera un ou deux fonctions différentes selon le temps d'exposé disponible, et la curiosité du jury. On parlera également d'un algorithme d'esprit différent, à savoir le CORDIC, que l'on utilisera ici pour le calcul de fonctions trigonométriques.

IV- Comparaison d'ordre temporel

Il s'agit ici de comparer le temps d'exécution des programmes codés en MAPLE. Mais on étudiera aussi la façon dont nos programmes fonctionnent. Par exemple, certains algorithmes utilisent des boucles, et il est intéressant de voir si l'utilisation de telles boucles peut nuire à la rapidité du programme, et si par conséquent, un autre algorithme, a priori moins rapide, ne peut pas donner lieu à une implémentation plus simple, tout en donnant de meilleurs résultats au niveau de la rapidité.

I-Problématique de départ et position du problème

Le sujet de cette année est « Le temps ». Par conséquent, il nous a semblé intéressant de nous pencher sur le problème du calcul de fonctions usuelles. En effet, avec les progrès de la technologie informatique, nous nous sommes accoutumés à cette situation où l'évaluation d'un sinus ou d'un cosinus nous est immédiatement fournie par notre calculatrice, ou notre ordinateur. Or, durant de nombreuses années, il était nécessaire de disposer d'imposantes tables de calcul pour connaitre de telles valeurs dans en temps raisonnable. Fort de ce constat, il nous est paru que s'intéresser à la façon dont on peut calculer la valeur d'une fonction usuelle, telle que le sinus, le cosinus, le logarithme népérien, l'exponentielle, en un point grâce à un calculateur informatique s'inscrivait parfaitement dans le cadre du thème de cette année. En effet, nous le verrons par la suite, il existe de nombreuses façons de calculer de tels nombres. Néanmoins, certaines s'avèrent naturellement bien plus rapides que d'autres. Il est donc légitime de comparer leur rapidité d'exécution, ainsi que leur fiabilité.

Nous verrons donc différents algorithmes permettant de calculer la valeur de différentes fonctions sur des intervalles de longueur variable. Néanmoins, nous nous restreindrons à l'étude de quelques fonctions bien précise, et ce pour plusieurs raisons. D'une part, parce que le temps n'est pas extensible, et qu'une étude approfondie de chaque fonction pourrait faire l'objet d'un TIPE à elle seule, d'autre part, parce que l'intérêt est relativement limité. En effet, l'étude de chaque fonction trigonométrique étant similaire dans chacun des cas, nous nous limiterons simplement à l'étude du sinus, qui à elle seule, se révèlera assez vaste. Nous nous pencherons dans le même temps sur deux autres fonctions, que sont l'exponentielle, et, dans une moindre mesure, le logarithme népérien.

II-Remarques préliminaires

En premier lieu, il est nécessaire de préciser que nous nous consacrerons au calcul numérique, et non à du calcul formel. Par conséquent, il n'est plus possible de parler de « calcul de fonctions usuelles », mais seulement d' «approximation ». En effet, il est totalement impossible de coder un nombre réel informatiquement parlant. La mémoire informatique étant finie, et la pluparts des réels ayant une écriture décimale infinie, nous commettons de cette manière une erreur à chaque fois que l'on entre un flottant en mémoire.

De plus, il est à noter que pour le calcul de fonctions trigonométriques, auquel nous nous intéresserons par le biais de la fonction sinus, il est parfois indispensable de connaitre la valeur du nombre π. Il apparait alors, que de nombreuses erreurs seront à prendre en compte. Premièrement parce que le nombre π ne peut être calculé de manière exacte, ensuite parce qu'il ne peut pas non plus être codé de manière exacte dans un ordinateur, et enfin parce que l'exécution de programmes utilisant une valeur approchée en lieu et place de ce qui devrait être une valeur exacte entraîne elle aussi des « erreurs » à répétition.

Ainsi, dans certains algorithmes, il faudra inévitablement tenir compte de l'erreur commise par la restriction de l' « intervalle de départ », et la façon que nous utilisons pour nous y ramener. En effet, il est parfois utile de n'évaluer une fonction que sur un intervalle particulier, et de s'y ramener d'une façon ou d'une autre, que ce soit par relation fonctionnelle, par exemple, ou encore par périodicité, comme avec les fonctions trigonométriques, ou même par parité, etc, …

Prenons par exemple :

$$PI = 3.14159$$

C'est-à-dire, le nombre π codé avec 5 chiffres après la virgule.

On a bien entendu en théorie : $\sin(20 * \pi) = 0$.

Or en effectuant le calcul sous Maple avec cette valeur de π, on obtient le résultat suivant :

$$\sin(10^{20} * PI) = 0{,}919276$$

(Voir annexe)

III-Etude de différents algorithmes

1. Algorithmes de types « Séries Entières »

Les fonctions que nous avons choisi d'étudier sont toutes développables en séries entières sur des intervalles plus ou moins grands. Par conséquent, il est naturel d'utiliser des sommes partielles dans le but d'obtenir une approximation de la valeur recherchée. De plus, l'utilisation de tels objets mathématiques nous permet d'obtenir plus d'informations sur les éventuelles erreurs mathématiques commises. La connaissance de majorants des restes des sommes partielles peuvent nous permettre notamment de réduire le temps de calcul, tout en maintenant un niveau théorique de précision très élevé.

Le Sinus :

On se restreint à l'intervalle $\left[0;\frac{\pi}{4}\right]$.

On note $R_n(x)$ le reste de la série entière du sinus. En effet :

$$\sin(x) = \sum_{n=0}^{+\infty} \frac{(-1)^n x^{2n+1}}{(2n+1)!}$$

On a donc : $R_n(x) = \sum_{k=n+1}^{+\infty} \frac{(-1)^k x^{2k+1}}{(2k+1)!}$

Et on peut appliquer le théorème spécial des séries alternées :

$$R_n(x) \leq \frac{\left(\frac{\pi}{4}\right)^{2n+2}}{(2n+2)!} \leq \frac{(0{,}8)^{2n+2}}{(2n+2)!} \ \forall x \in \left[0;\frac{\pi}{4}\right]$$

On cherche alors n tel que $|R_n(x)|$ soit inférieur à un certain epsilon.

(Voir annexe)

Ainsi, pour une précision donnée, on est capable de calculer un nombre d'itération suffisant. Reste alors à calculer une somme partielle de la série entière.

(Voir annexe)

L'exponentielle :

On note $R_n(x)$ le reste de la série entière de l'exponentielle. En effet :

$$\exp(x) = \sum_{n=0}^{+\infty} \frac{x^n}{n!}$$

On a alors :

$$R_n(x) = \sum_{k=n+1}^{+\infty} \frac{x^k}{k!} = \frac{x^{n+1}}{(n+1)!}\left(1 + \frac{x}{n+2} + \frac{x^2}{(n+2)(n+3)} + \cdots\right) \leq \frac{x^{n+1}}{n!}$$

(voir annexe)

Le logarithme népérien :

On note $R_n(x)$ le reste de la série entière du logarithme. En effet :

$$-\ln(1-x) = \sum_{n=1}^{+\infty} \frac{x^n}{n}\ , \forall x \in]-1; 1[$$

De plus, comme $\ln(x) = -\ln\left(\frac{1}{x}\right)$, on peut facilement se restreindre à une étude pour $x \geq 1$.
Ainsi, la série entière est alternée, et relève du théorème spécial.
On peut donc majorer le reste :

$$R_n(x) \leq \frac{x^n}{n}$$

Cependant, cela n'est exploitable que si $1 - x \in]-1; 1[$
Il faut donc se ramener dans cet intervalle.
Pour cela, on va calculer la constante $\ln(2)$ avec une très grande précision, grâce à la formule :

$$\ln(2) = \sum_{n=1}^{+\infty} \frac{1}{n} \cdot \left(\frac{1}{2}\right)^n$$

(On obtient une précision à 10^{-1000}près, pour une somme partielle de 3308 termes.)
Ainsi, comme l'on dispose de la relation : $\ln(2x) = \ln(x) + \ln(2)$, on peut aisément se ramener dans l'intervalle souhaité, et travailler sur des sommes partielles.

(Voir annexe)

2. Algorithmes utilisant des relations fonctionnelles

Les trois fonctions étudiées vérifient également des relations, dites fonctionnelles, qui vont nous permettre d'imaginer de nouveaux algorithmes, basés dessus, et d'esprit totalement différent de celui développé dans la partie précédente. Il s'agit en effet, de se ramener ici à l'approximation, sur un intervalle de longueur plus ou moins faible, selon la précision voulue, de notre fonction, par un polynôme, et ce, du fait qu'elle admet un développement limité en un point contenu dans l'intervalle considéré.

Le Sinus :

On utilise ici la relation fonctionnelle : $\sin(3x) = -4sin^3(x) + 3\sin(x)$
ainsi que le fait que : $\sin(x) = x + o(x)\ quand\ x \to 0$
Le programme est donc récursif, et il convient de majorer l'erreur commise lors de sa remontée.

On défini, pour un réel x donné, la suite $(x_n)_{n\in\mathbb{N}}$ par : $x_n = \frac{x}{3^n}$
On étudie alors la fonction $\varphi: x \to -4x^3 + 3x$
Cette fonction est de classe C^1 et par conséquent, d'après le théorème de représentation des fonctions de classe C^1, on a :

$$|\varphi(a) - \varphi(b)| \leq \int_a^b |\varphi'(x)|\, dx \leq 4{,}7.(a-b)\ \ (sur\ \left[-\frac{\pi}{4};\frac{\pi}{4}\right])$$

On défini ensuite la suite $(y_n)_{n\in\mathbb{N}}$ par : $y_n = \varphi^n(x_n)$

De plus, d'après le développement en série entière du sinus, on a :

$$|\sin(x_n) - x_n| \leq \frac{{x_n}^3}{3} \leq \frac{0{,}8^3}{3^{n+1}}$$

Ainsi, $|\sin(x_0) - y_n| = |\varphi(\sin(x_1)) - \varphi^n(x_n)| \leq 4{,}7.|\sin(x_1) - \varphi^{n-1}(x_n)|$

et donc, $|\sin(x_n) - y_n| \leq 4{,}7^n.|\sin(x_n) - x_n| \leq 4{,}7^n.\frac{0{,}8^3}{3^{n+1}}$
Ainsi, on peut majorer l'erreur de méthode, et connaitre le nombre maximal d'itérations qu'effectue notre programme.

(Voir annexe)

On peut aussi utiliser une autre relation fonctionnelle :
$\sin(2x) = 2\sin(x).\cos(x) = 2\sin(x).\sqrt{1 - sin^2(x)}$

On défini, pour un réel x donné, la suite $(x_n)_{n\in\mathbb{N}}$ par : $x_n = \frac{x}{2^n}$
On étudie alors la fonction $\varphi: x \to 2x.\sqrt{1-x^2}$
On a alors :

$$|\varphi(a) - \varphi(b)| \leq \int_a^b |\varphi'(x)|\, dx \leq 5{,}4.(a-b) \;\; (sur\ \left[-\frac{\pi}{4};\frac{\pi}{4}\right])$$

On défini ensuite la suite $(y_n)_{n\in\mathbb{N}}$ par : $y_n = \varphi^n(x_n)$

Et de la même manière que précédemment, on trouve cette fois :
$|\sin(x_n) - y_n| \leq 4{,}7^n.|\sin(x_n) - x_n| \leq 5{,}4^n \frac{0{,}8^3}{3^{n+1}}$

(Voir annexe)

Le logarithme népérien :
On utilise ici la relation fonctionnelle : $\ln(x) = 2.\ln(\sqrt{x})$
ainsi que le fait que : $\ln(1+x) = x + o(x)\ quand\ x \to 0$
Le programme est donc récursif, et il convient de majorer l'erreur commise lors de sa remontée.
Malheureusement, la même technique que la précédente ne nous permet pas d'expliciter une relation entre n et la précision souhaitée. On peut en revanche affirmer que l'erreur tend vers 0 quand n tend vers l'infini.

3. Algorithme du CORDIC dans le cadre des fonctions trigonométriques

Cet algorithme est un peu particulier, et n'entre dans notre étude qu'à titre d'information.
En effet, son principe est de ne faire subir au calculateur que des opérations qui lui sont naturelles, à savoir des additions et des décalages de bits (multiplications et divisions par 2). Il a donc été très utile, notamment lors de la conquête de l'espace, dans les années 60, quand les calculateurs n'étaient pas assez puissants pour effectuer un trop grand nombre de divisions par des trop grands nombre en un temps raisonnable.

L'idée consiste, pour calculer la tangente d'un angle θ, à faire subir à un vecteur $(x; y)$ des rotations d'angles de plus en plus proches de 0, dont la somme est égale à θ.
On a alors : $\frac{y}{x} = \tan(\theta)$

Soit $\theta \in \left[0; \frac{\pi}{4}\right]$ et une suite $(\theta_i)_{i \leq n}$ avec $\theta_n \leq \theta_{n-1} \leq \cdots \leq \theta_1 \leq \theta_0$.
Si on note : $M_O = \begin{pmatrix} x_0 \\ y_0 \end{pmatrix}$, alors on a : $M_1 = \begin{pmatrix} \cos(\theta_0) & -\sin(\theta_0) \\ \sin(\theta_0) & \cos(\theta_0) \end{pmatrix} . \begin{pmatrix} x_0 \\ y_0 \end{pmatrix}$

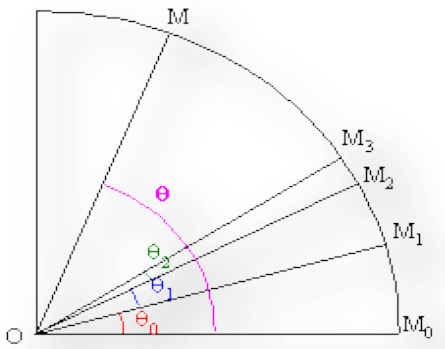
Ainsi, on a de manière générale : $M_{i+1} = \begin{pmatrix} \cos(\theta_i) & -\sin(\theta_i) \\ \sin(\theta_i) & \cos(\theta_i) \end{pmatrix} . \begin{pmatrix} x_i \\ y_i \end{pmatrix}$

On décide alors arbitrairement de prendre $M_O = \begin{pmatrix} k \\ 0 \end{pmatrix}$ avec par exemple $k = 1$.
On obtient alors, si on note $M = \begin{pmatrix} x \\ y \end{pmatrix}$, $x = k.\cos(\theta)$ et $y = k.\sin(\theta)$.
(On voit bien que pour calculer la tangente, la valeur de k importe peu.)

Une fois la valeur de $\tan(\theta)$ obtenue, on peut obtenir les valeurs du sinus et du cosinus grâce aux formules : $\cos(\theta) = \frac{1}{\sqrt{1+\tan^2(\theta)}}$ et $\sin(\theta) = \frac{\tan(\theta)}{\sqrt{1+\tan^2(\theta)}}$.
Cet algorithme nécessite donc une table de valeur, puisque l'on a besoin de connaitre des valeurs de $\arctan(10^{-i})$. On peut toutefois, pour i assez grand, supposer

$$\arctan(10^{-i}) \approx 10^{-i}$$

Ce qui permet de ne pas stocker un trop grand nombre de valeurs.



(Voir annexe)

4. Comparaison d'ordre temporel

Nous nous intéresserons ici au cas du calcul du sinus d'un angle.

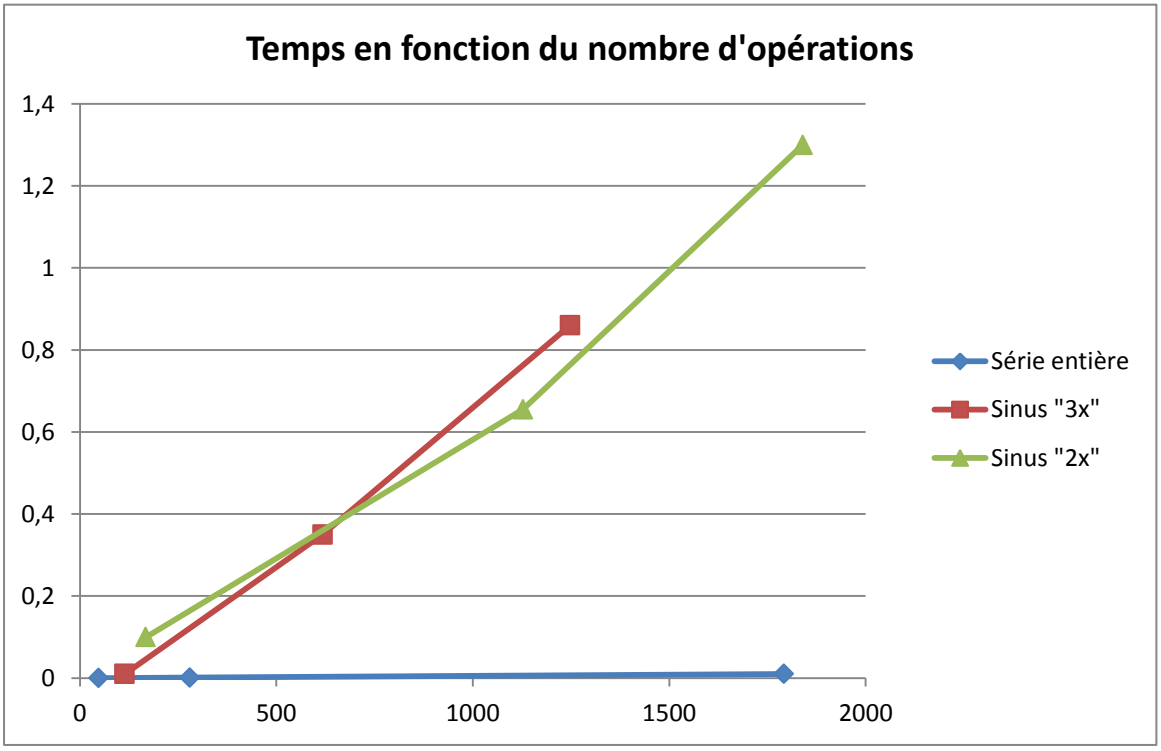
	Série entière			Sinus « 3x »			Sinus « 2x »		
Précision	10^{-10}	10^{-100}	10^{-1000}	10^{-10}	10^{-100}	10^{-1000}	10^{-10}	10^{-100}	10^{-1000}
Nb. It.	6	35	224	19	208	2094	21	230	2316
Nb. Op.	48	280	1792	114	1248	12564	168	1840	18528
Temps	$10^{-4}s$	$10^{-3}s$	$10^{-2}s$	$10^{-2}s$	$10^{-1}s$	8,4s	$10^{-1}s$	$1{,}3s$	$16{,}3s$

Informations complémentaires :

-On travaille avec un nombre de digits fixés, suffisant pour que la précision n'en soit pas affectée.

-On prend à chaque fois $x \in \left[0; \frac{\pi}{4}\right]$.

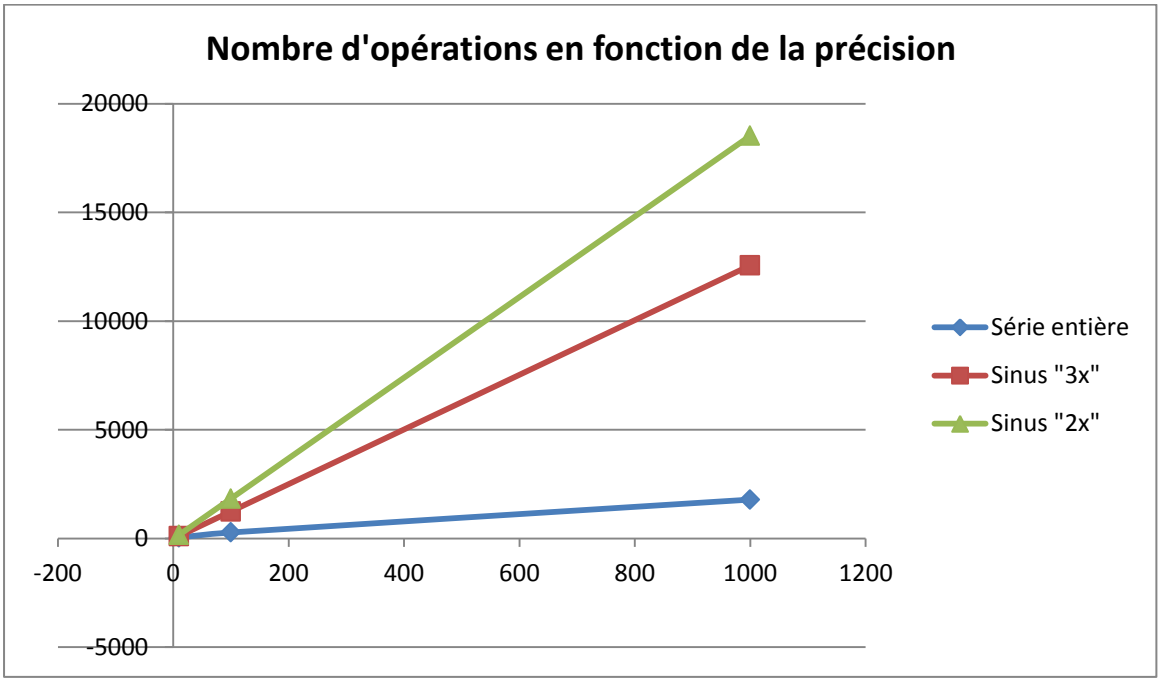
-Les multiplications comptent autant que les additions, et le calcul d'une racine compte comme trois multiplications.

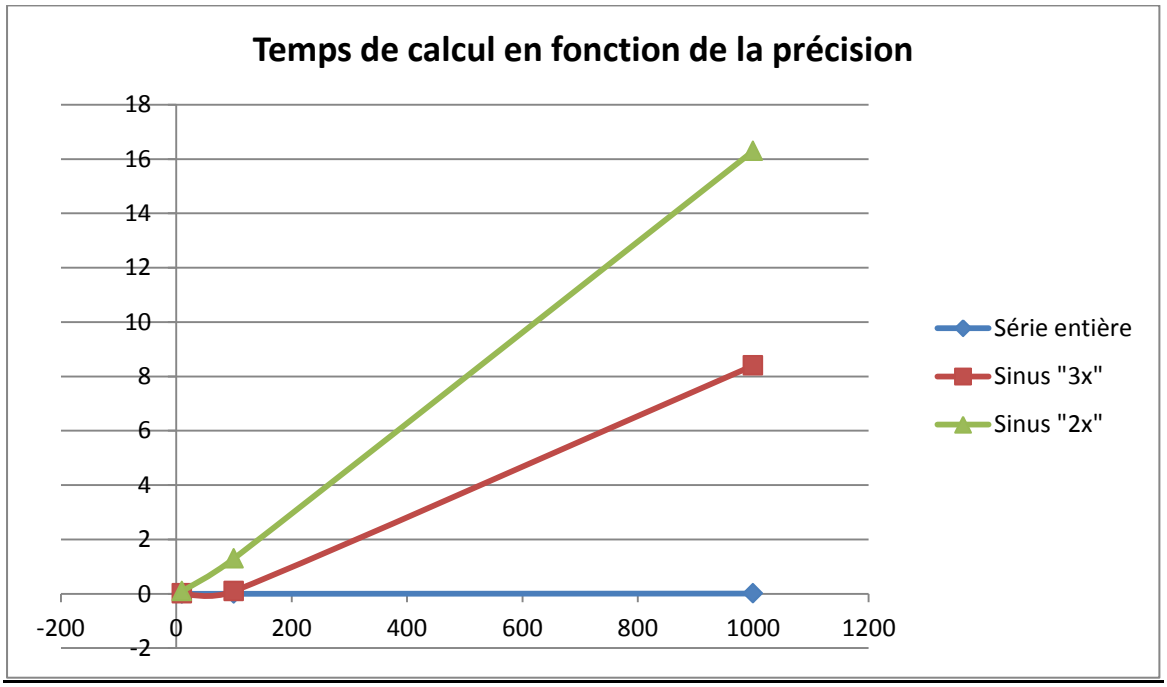


On s'aperçoit alors que l'algorithme de la série entière prend un net avantage sur les deux autres, et ne souffre pas du d'un grand nombre de calculs, contrairement notamment à celui du sinus « 2x ». Cela peut s'expliquer en partie du fait qu'il ne s'agit pas d'un algorithme récursif.

En revanche, il est à noté que les deux algorithmes récursifs doivent calculer à chaque fois ne nombre d'itération à effectuer, pour savoir quand s'arrêter en fonction de la précision souhaitée. Le désavantage de l'algorithme de la série entière est donc qu'il faut au préalable calculer le nombre d'itération nécessaire. Toutefois, si l'on se restreint à un intervalle prédéfini comme on l'a

fait, on peut voir que choisir un nombre d'itération maximal ne gène en rien la rapidité d'exécution des programmes.





Fiche Synoptique :

De nos jours, l'usage de la calculatrice ou d'un calculateur informatique est devenu banal, à tel point que l'on ne se pose plus de questions quant à la façon dont la machine évalue la valeur d'un sinus, d'une racine ou d'un logarithme...
Lorsqu'il s'agit de le faire manuellement, nous sommes bien vite limités.
L'objet de ce TIPE est l'étude de diverses méthodes de calcul par des logiciels tels que Maple, ainsi que de leur comparaison du point de vue temporel, afin de relier notre travail au thème de cette année.
De telles avancées ont eu pour effet de nous dispenser de l'usage de règles à calcul, et de ce fait de pouvoir accéder **rapidement** à l'interprétation de phénomènes physiques sans être ralenti par des obstacles de nature purement calculatoire.

Remarque préliminaire

On se consacre au calcul numérique, et donc, non formel (dès que l'on entre un flottant dans une machine, on commet une erreur). Ainsi, il convient de signaler l'inexactitude des résultats renvoyés, non seulement d'origine théorique, mais provenant d'une limite physique, inhérente au caractère fini de la mémoire informatique.
C'est notamment le cas lors de l'approximation de la valeur du sinus d'un réel, étant donné que la connaissance de la valeur de Pi est nécessaire.

Etude des différents algorithmes

Pour toutes les fonctions usuelles considérées et que nous avons choisi d'étudier, c'est à dire le sinus, l'exponentielle, le logarithme népérien, nous avons utilisé deux types d'algorithmes.
- Un algorithme profitant du fait que ces fonctions sont développables en séries entières sur un intervalle plus ou moins grand. Il consiste tout simplement à sommer les termes jusqu'à un rang donné, ce rang étant obtenu grâce à une majoration du reste.
-Un algorithme basé sur une ou des relations fonctionnelles propres à chaque fonctions, dès lors que celles-ci existent et sont exploitables informatiquement. Le principe de cet algorithme est de ce ramener à évaluer les fonctions auxquelles nous limitons notre étude à un intervalle sur lequel un développement limité nous en offre une bonne approximation. On peut alors jouer sur la longueur de cet intervalle pour réguler la précision du résultat. Cette longueur est obtenue grâce à une étude théorique du problème, utilisant principalement le théorème de représentation intégrale des fonctions continûment dérivables, visant à majorer l'erreur informatique, et l'erreur mathématique commise.
-Pour l'étude du sinus, nous parlerons également de l'algorithme du Cordic, reposant sur le calcul de composantes de vecteurs auxquels on applique une rotation bien choisie. Il prend historiquement tout son intérêt lors du calcul de fonctions trigonométriques, en renvoyant relativement rapidement un résultat, avec une précision donnée.
En revanche, cela requiert la donnée d'une table de valeurs prédéterminées, ce qui entraîne une limitation drastique de la précision.

Comparaison d'ordre temporel

Deux modes d'évaluation de la durée du calcul ont été envisagés: le chronométrage "classique", en situation, et le dénombrement de l'ensemble des opérations effectuées. Le temps d'évaluation est généralement très court, à moins d'imposer des contraintes fortes (nombre important de chiffres significatifs).
Néanmoins, un écueil se dresse devant nous: peut-on négliger le temps d'interprétation des programmes

face au temps du calcul des nombres qui nous intéressent en effectivement? En effet, en plus du calcul destiné à estimer la valeur d'une fonction en un point, il peut se faire qu'un algorithme nécessite d'effectuer des tests, demandant des calculs complémentaires, dont le temps d'exécution peut être conséquent, dans le cadre de condition de fin de boucles par exemple. Il est toutefois à noter que l'on ne travaillera pas en multiprécision, et que par conséquent, la durée de chaque opération élémentaire sera constante.