MP⋆ 2011-2012

Une année de MAPLE en MP⋆

Guillaume CONNAN

SOMMAIRE

AU PROGRAMME

Voici quelques extraits du programme officiel :

L'informatique en classes préparatoires a pour principaux objectifs d'offrir dans le tronc commun :

- *Une familiarisation avec l'utilisation d'outils informatiques évolués (logiciel de calcul formel et numérique, logiciels d'acquisition et de traitement de données, logiciels de modélisation, logiciels de simulation...) en vue de permettre l'approfondissement des disciplines scientifiques et techniques ;*
- *Une introduction à l'informatique en tant que discipline, par une initiation élémentaire au traitement automatique de l'information, à l'algorithmique et à la programmation structurée (illustrée à l'aide du langage du logiciel de calcul formel retenu).*

[...]
On habituera les élèves à se servir de logiciels, qui fournissent un support au raisonnement par la confrontation rapide et commode des hypothèses et résultats, et permettent :

- *D'enrichir la compréhension des phénomènes mathématiques et des modèles physiques par la simulation de leurs comportements en fonction de divers paramètres ;*
- *De mieux cerner la notion de domaine de validité d'une hypothèse ou d'une méthode par l'étude de cas limites ;*
- *D'étudier certains problèmes par la mise en oeuvre de modèles dont la résolution numérique manuelle serait trop lourde ou trop complexe ;*
- *D'alléger la part de calcul systématique au profit de l'intuition mathématique ou du sens physique.*

[...]
La mise en œuvre de la programmation n'est pas séparée de l'utilisation du logiciel de calcul formel en tant qu'outil et s'effectue à l'occasion des séances de travaux pratiques, appliquées à la résolution de problèmes de mathématiques, de physique, de chimie, de mécanique et automatique.
[...]
L'outil informatique n'est pas une fin en soi mais un moyen efficace pour faire des mathématiques, des sciences physiques ou des sciences industrielles.

À CENTRALE

Voici extrait du rapport du jury 2010 concernant l'oral de mathématiques II en série MP :

Présentation de l'épreuve

L'épreuve de Maths II est une épreuve de mathématiques assistée par ordinateur, en l'occurrence par l'usage d'un logiciel de calcul formel. Elle comporte un seul exercice qui est préparé pendant 30 minutes avec accès libre à l'ordinateur ; puis le candidat vient présenter pendant 30 minutes ses résultats et poursuivre la résolution de l'exercice au tableau. Une fois au tableau, il est souhaitable d'indiquer succinctement les questions qui ont été élucidées pendant la préparation et ensuite de ne pas trop perdre de temps sur les questions élémentaires, pour arriver au coeur du sujet. De façon générale, le candidat doit être très attentif aux conseils et a fortiori aux indications données oralement, et surtout de ne pas s'enfermer dans une tentative de résolution si l'examinateur lui indique qu'elle risque de conduire à une impasse : on aboutit à un échec donc à une perte de temps qui empêchera la résolution d'autres questions. Il est essentiel que l'oral reste un dialogue, et une très bonne note peut être attribuée à un candidat qui, sans avoir résolu l'exercice lors de la préparation, aura montré une bonne réactivité aux indications proposées. Une des difficultés de cette épreuve est de savoir gérer l'aide que peut apporter le logiciel pour résoudre la question mathématique posée. Les exercices comportent pour la plupart au moins une question à résoudre avec l'outil informatique. Il est souvent attendu de pouvoir émettre une conjecture, qui sera démontrée dans la suite de l'exercice. Parfois, l'énoncé conseille d'utiliser le logiciel à bon escient au cours de l'exercice, sans qu'il soit imposé pour une question précise : il revient alors au candidat d'évaluer les questions où l'ordinateur lui apportera une aide précieuse. C'est par exemple le cas pour des calculs auxiliaires de développements limités ou d'intégrales élémentaires (coefficients de Fourier ...), ou des résolutions d'équations lors d'un exercice de géométrie : le logiciel permet d'alléger les calculs et d'éviter les erreurs liées au stress de l'épreuve. Il faut par contre savoir être circonspect sur certaines réponses du logiciel : que penser d'un candidat qui ne réagit pas face à l'affichage de valeurs propres « complexes » pour une matrice symétrique réelle, alors qu'une demande de valeurs approchées montre que leurs parties imaginaires sont des `10^(-6)*I` ? Il ne s'agit pas là de savoir comment fonctionne le logiciel mais d'avoir un minimum de recul ou de prudence sur l'affichage de certains résultats.
[...]

Commentaires et conseils concernant l'usage du logiciel de calcul formel

L'impression générale est ici un peu mitigée. Et il semble utile de redonner une liste de « savoir-faire » déjà publiée dans le rapport 2009 et qui figure en Annexe. Le jury a vu avec plaisir environ un quart des candidats très bien préparés, très à

l'aise avec cet aspect de l'épreuve. Une majorité a montré l'habitude de côtoyer le logiciel et la connaissance des commandes usuelles. Signalons qu'il n'est d'ailleurs attendu ni dextérité ni connaissance savante des options possibles d'une fonction prédéfinie ; et il est normal et raisonnable qu'un candidat s'assure du bon emploi d'une fonction du logiciel en ayant recours à l'aide en ligne (encore faut-il en connaître le nom... : alors pourquoi se priver très souvent de l'affichage à l'écran des fonctions résidant dans une librairie, par un `with(...):` au lieu d'un `with(...);` du logiciel Maple ?). Par contre, il faut constater et regretter qu'un trop grand nombre –même si c'est une minorité– continue à espérer découvrir les fonctions de base avec un usage fébrile de l'aide en ligne pendant la préparation : cela se traduit par une lourde perte de temps, et le bilan est en général catastrophique : aucun résultat ne sort d'une succession de lignes de code dont la plupart ont été rejetées par le logiciel.... À la moindre sollicitation de l'examinateur pour apporter les premières modifications permettant l'obtention d'au moins un résultat partiel (faute de temps, il ne peut être question de corriger les fautes accumulées), la réponse « je n'ai pas pratiqué le logiciel cette année » ne peut constituer une excuse. L'usage d'un logiciel de calcul formel figure pour tous, au programme des deux années de classes préparatoires, et les candidats savent qu'ils en auront besoin lors de l'épreuve de maths II du concours Centrale. Le jury sanctionnera plus nettement cette attitude désinvolte : on ne peut pas arriver à cette épreuve si on ne sait pas utiliser efficacement le logiciel. Le peu de programmation attendue ne dépasse pas l'écriture de boucles avec d'éventuelles instructions conditionnelles ! Il ne s'agit pas d'un exercice d'algorithmique. Relevons quelques pratiques maladroites. On regrette d'abord un recours trop systématique à l'écriture de nombreuses et inutiles procédures, quelle que soit la complexité de 32 Épreuves orales Rapport du jury 2010 - Filière MP la question posée : cette démarche peut sembler tout à fait honorable, mais conduit trop souvent hélas à un échec, et donc à l'absence de résultats effectifs ; or c'est le but attendu. Et puis c'est une mauvaise compréhension de l'intérêt d'un logiciel de calcul formel : les fonctions prédéfinies sont là pour gagner du temps, et écrire une procédure pour définir la fonction « factorielle » (exemple caricatural mais vu cette année) n'a aucun intérêt ici. En fait, il est rarement indispensable d'écrire une procédure lors des questions proposées, ce qui ne signifie pas qu'elles sont parfois bienvenues ; mais l'écriture directe d'une boucle est souvent suffisante. Il est nécessaire de savoir distinguer la manipulation des « fonctions » et des « expressions », et d'estimer quand l'usage des unes ou des autres est plus favorable. Signalons que l'usage des expressions est souvent plus souple lorsqu'il doit se doubler de la création d'un opérateur mathématique qui manipule ces expressions. Certains ne savent d'ailleurs pas créer une suite, ou une fonction ; l'usage des crochets ou des parenthèses est mal compris : u[n] ou u(n) ? Et plus gênant encore est de voir écrire des tentatives du type `u(n):=` ... ou `f(x):=` ... pour fabriquer une fonction, et des candidats qui sont surpris que le logiciel ne réponde pas à leur attente !! Le logiciel met à disposition des outils commodes pour créer des séquences de résultats. Combien de fois on a vu le recours à des copier-coller quand l'énoncé demandait une séquence d'une dizaine ou d'une vingtaine de résultats (nécessaires à l'ébauche d'une conjecture « fiable ») ? Il faut enfin savoir indiquer au logiciel qu'une variable est par exemple entière, réelle positive, etc... Et connaître quelques commandes qui simplifient ou convertissent ou transforment des expressions sous une forme souhaitée.
Mais voici quelques points encore trop mal maîtrisés :

- la construction de matrices de taille variable. Il faut savoir fabriquer une fonction « définissante » des coefficients. On a ainsi vu régulièrement des candidats contraints d'écrire une matrice 10*10 en tapant les 100 coefficients (dont beaucoup étaient nuls heureusement) !
- savoir obtenir des valeurs approchées des racines d'une équation, savoir que l'affichage d'un seul résultat numérique ne se traduit pas nécessairement par l'unicité d'une solution... ;
- pour le graphisme, il faut savoir superposer sur un même schéma divers types de graphes ;
- dans le cas particulier des équations différentielles, beaucoup ne savent pas visualiser le graphe d'une solution, lorsque le logiciel n'en donne pas une expression exacte.
- proscrire l'ouverture et l'usage simultanés des librairies Maple `linalg` ET `LinearAlgebra` ;
- connaître les inconvénients ou avantages respectifs des commandes « sum » et « add » ...

[...]

Liste de savoir-faire conseillés pour l'épreuve assistée par un logiciel de calcul formel

Calcul algébrique (entiers, polynômes, équations)

- savoir calculer le quotient, le reste dans une division euclidienne dans $\mathbf{Z}$, dans $\mathbf{Q}[X]$;
- savoir tester qu'un entier est premier, savoir travailler modulo n ;
- savoir factoriser (dans $\mathbf{Q}[X]$ et éventuellement dans une extension simple suggérée par l'énoncé), développer, ordonner un polynôme ;
- savoir obtenir tous les coefficients, ou des coefficients précis d'un polynôme ;
- savoir calculer le pgcd de deux entiers, de deux polynômes ;
- savoir obtenir un couple donnant la relation de Bézout ;
- savoir déterminer les racines d'une équation (algébrique ou non) de façon exacte, de façon approchée ; savoir déterminer une valeur approchée d'une racine localisée dans un intervalle ;
- savoir décomposer une fraction rationnelle en éléments simples dans $\mathbf{Q}(X)$ (éventuellement dans une extension simple de $\mathbf{Q}$ suggérée par l'énoncé).

Calcul matriciel

- savoir construire une matrice dont les coefficients sont donnés par une formule fonction du couple (i, j), et dont la taille peut être variable (il ne peut être question de se limiter à savoir entrer une matrice 3x3 par ses neuf coefficients) ;
- savoir calculer des produits matriciels, créer une matrice diagonale et a fortiori la matrice identité, former la transposée ;
- savoir calculer le rang, le noyau ou l'image (en obtenant une base de ces sous-espaces) ;
- savoir calculer le déterminant, éventuellement l'inverse, la comatrice (ou sa transposée) d'une matrice carrée ;
- savoir calculer le polynôme caractéristique d'une matrice carrée, ses valeurs propres, ses vecteurs propres ;
- savoir résoudre une équation d'inconnue matricielle (après l'avoir transformée en un ensemble d'équations scalaires d'inconnues les coefficients) ;
- savoir calculer le produit scalaire, le produit vectoriel de deux vecteurs de $\mathbf{R}^3$.

Fonctions d'une ou plusieurs variables réelles, calcul différentiel, calcul intégral

- savoir composer des fonctions (ou des opérateurs), calculer des dérivées d'ordre supérieur à un ;
- savoir calculer un développement limité, savoir extraire la partie régulière d'un tel développement ;
- savoir calculer une intégrale de façon exacte, de façon approchée, faire un changement de variable ou une intégration par parties ;
- comprendre pourquoi le logiciel n'affiche pas toujours une limite explicite, ou le résultat d'un calcul d'intégrale, par manque d'information sur la nature d'un paramètre introduit : savoir préciser à quelle partie de $\mathbf{R}$ il appartient (entier, réel positif...)

Suites et séries numériques, suites et séries de fonctions

- savoir expliciter les premiers termes (de façon exacte ou approchée) d'une suite numérique ou d'une suite de fonctions, en particulier lorsqu'elle est définie par récurrence ;
- savoir obtenir un développement asymptotique d'une suite (fonction explicite de n) ;
- savoir calculer les coefficients de Fourier d'une fonction périodique ;
- savoir visualiser sur un même schéma les premiers termes d'une suite de fonctions.

Équations différentielles

- savoir résoudre une équation différentielle, un système d'équations différentielles, avec ou sans conditions initiales ;
- savoir récupérer une fonction solution et la tracer ;
- savoir tracer directement le graphe d'une solution obtenue par résolution numérique.

Graphisme

On a déjà évoqué le tracé de graphes de fonctions d'une variable réelle, de solutions d'une équation différentielle.

- savoir tracer une courbe du plan, définie par une équation cartésienne (de façon implicite), ou par un paramétrage, peut-être en coordonnées polaires, et gérer les discontinuités ;
- savoir tracer une courbe paramétrée de l'espace ;
- savoir tracer une surface définie par un paramétrage, ou par une équation cartésienne ;
- savoir visualiser un ensemble de points, sous forme d'une ligne polygonale ou non.

TD N°

1

Un peu d'arithmétique

1 Ave Cesar

Pour les problèmes de codage-décodage (cf X 2008), on a souvent besoin de manipuler des listes de caractères.
On peut utiliser la bibliothèque `StringTools` et en particulier :

- `Ord("car")` qui renvoie le code ASCII de `car`. Par exemple `Ord("a")` renvoie `97` ;
- `Char(code)` effectue l'opération inverse. Par exemple `Char(97)` renvoie `"a"` ;
- `Explode("abc")` renvoie `["a","b","c"]` ;
- `Implode(["a","b","c"])` renvoie `"abc"` ;

Créez une fonction `cesar:=proc(message,cle)` qui code un message rentré sous forme d'une chaîne selon la méthode bien connue de César, en se donnant la possibilité de choisir sa clé. On travaillera avec un alphabet de 95 lettres :

Caractère		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/	0	1	2
Code	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50
Caractère	3	4	5	6	7	8	9	:	;	<	=	>	?	@	A	B	C	D	E
Code	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69
Caractère	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Code	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88
Caractère	Y	Z	[	\	]	^	_	‘	a	b	c	d	e	f	g	h	i	j	k
Code	89	90	91	92	93	94	95	96	97	98	99	100	101	102	103	104	105	106	107
Caractère	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~
Code	108	109	110	111	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126

Table des caractères ASCII affichables

2 Éléments inversibles de Zn

Dans la suite, nous noterons $\mathbb{Z}_n$ l'ensemble $\mathbb{Z}/n\mathbb{Z}$.
Dressez la table de Pythagore de $(\mathbb{Z}_n, \times)$ pour $n \in \{9, 11, 12\}$.
Vous créerez une procédure `pyth:=proc(n)` qui dresse la table de $(\mathbb{Z}_n, \times)$ en commençant par exemple par :

```
pyth:=proc(n)
 local T,i,j;
 T:=array(1..n-1,1..n-1);
 ........................
   ......................
     T[i,j]:=...........
   ...
 ...
 print(T)
 end:
```

On observe par exemple que $4 \times 2 \equiv 4 \times 5[12]$ mais on n'a pas pour autant $2 \equiv 5[12]$.
Autre problème : on a $4 \times 3 \equiv 0[12]$, alors que ni 4 ni 3 n'est nul.

Donnez une condition nécessaire et suffisante pour qu'un élément de $(\mathbb{Z}_n, \times)$ soit inversible.
Créez une procédure `invm(a,n)` qui calcule, s'il existe, l'inverse de `a` dans $(\mathbb{Z}_n, \times)$.
On pourra utiliser la commande `igcdex(a,b,'u','v')` qui renvoie $a \wedge b$ et qui stocke des coefficients de Bézout dans les variables `u` et `v`.

```
> igcdex(7,9,'u','v');
                                  1
> u,v;
                                4, -3
```

Comparez `invm(a,n)` avec `1/a mod n`...

3 Chiffrement affine

On utilise encore notre alphabet de 95 caractères et on va coder les nombres associés par une fonction du style :

$$f : x \mapsto ax + b \ [95]$$

où a et b sont des entiers.
En utilisant certaines procédures utilisées précédemment avec le code de César, créez une procédure `code_affine:=proc(message,a,b)` qui code une message à l'aide de la fonction précédente.
Comment décoder un message ? Créez une procédure `decode_affine:=proc(crypte,a,b)` qui décode un message crypté à l'aide de la fonction $f : x \mapsto ax + b \ [95]$.

4 Chiffrement de Vigenère

Blaise de VIGENÈRE naquit en 1523 à Saint-Pourçain sur Sioule, célèbre pour ses vins d'Auvergne. Il fit paraître en 1586 le *Traicté des chiffres ou secrètes manières d'escrire*[a].
Nous allons étudier un cas particulier. Choisissons une clé qui peut être un mot, une phrase, un ensemble de phrase, un livre entier. Pour nous simplifier la vie, prenons par exemple la clé « VENTRILOQUIST »[b].
On veut chiffrer le message « LES SANGLOTS LONGS DES VIOLONS DE L'AUTOMNE »
On a besoin d'un tableau de chiffrement où on affiche les 26 lettres de notre alphabet actuel sur chaque ligne, en décalant les lettres d'un cran d'une ligne à l'autre.
On peut le faire à la main...ou demander à Maple de le faire grâce à la commande **`Char`** et en travaillant modulo 26.
Retirons les espaces de notre message et disposons les lettres ainsi :

```
LESSANGLOTSLONGSDESVIOLONSDELAUTOMNE
VENTRILOQUISTVENTRILOQUISTVENTRILOQU
```

Pour chiffrer le L, il suffit de regarder dans notre tableau la lettre se trouvant sur la ligne du L et la colonne du V : c'est G.
On fait pareil pour le E qui est chiffré en I, le S en F, le S suivant en L, etc.
Comme c'est assez répétitif, on va utiliser Maple pour faire le sale boulot. Il suffit de mathématiser un peu le travail et tout sera plus simple.
Vous devez obtenir par exemple :

a. Le manuscrit est disponible sur Gallica
b. En plus d'un mot anglais désignant une personne parlant sans ouvrir la bouche, ce terme était également le nom d'un réseau de résistance dirigé par Philippe de VOMÉCOURT en Sologne et qui sabota les voies de chemin de fer menant vers la Normandie à partir du 5 juin 44 suite à la réception d'un fameux message codé...

```
> code_vige('LESSANGLOTSLONGSDESVIOLONSDELAUTOMNE','VENTRILOQUIST')
            'GIFLRVRZENADHIKFWVAGWEFWFLYIYTLBZADY'
> decode_vige('GIFLRVRZENADHIKFWVAGWEFWFLYIYTLBZADY','VENTRILOQUIST'
    )
                'LESSANGLOTSLONGSDESVIOLONSDELAUTOMNE'
```

Plus la clé est longue, plus grande est la sécurité. D'ailleurs, le principe de la machine ENIGMA était semblable, avec une clé de longueur 26^n où n est le nombre de rotors utilisés.
Auparavant, les messages étaient cryptés à l'aide de clés représentés par des livres entiers.
Cependant, si la clé est trop courte, une étude statistique et arithmétique permet de casser le code comme le démontra Charles BABBAGE en 1846, près de trois siècles après la parution du traité de VIGENÈRE.

5 Fonction indicatrice d'Euler

5 1 Rappel

Il existe plusieurs définitions équivalentes de la fonction indicatrice d'EULER φ. Pour être en lien avec ce qui a été vu précédemment, nous dirons que $\varphi(n)$ est l'ordre du groupe des inversibles de $\mathbb{Z}_n$.
Ainsi, on obtient le théorème d'EULER : si a et n sont premiers entre eux, $a^{\varphi(n)} \equiv 1\ [n]$.

5 2 Théorème chinois

Notons $\dot{a}[n]$ la classe d'un entier a modulo n. Soit n et k deux entiers premiers entre eux.
Voici une version simplifiée mais suffisante pour la suite de nos aventures du fameux théorème :
L'application f définie par :

$$f : \begin{array}{ccc} \mathbb{Z}_{nk} & \to & \mathbb{Z}_n \times \mathbb{Z}_k \\ \dot{x}[nk] & \mapsto & (\dot{x}[n], \dot{x}[k]) \end{array}$$

est bien définie et c'est un isomorphisme d'anneaux.

5 3 Propriétés de la fonction indicatrice d'Euler

Pouvez-vous le démonter puis en déduire que pour tout couple (n,k) d'entiers naturels premiers entre eux :

$$\varphi(n \times k) = \varphi(n) \times \varphi(k)$$

Soit p un nombre premier et n un entier naturel. Prouvez que $\varphi\left(p^n\right) = p^n - p^{n-1}$ puis que, pour tout entier naturel non nul k, $\varphi(k) = k \prod_{p|k} \left(1 - \frac{1}{p}\right)$.

5 4 Calcul de la fonction indicatrice d'Euler

Déterminez une procédure `eul:=proc(n)` qui calcule $\varphi(n)$.
Vous pourrez utiliser la fonction Maple `ifactors(n)` qui renvoie, pour des entiers positifs non nuls, une liste contenant 1 et la liste des diviseurs premiers de n et de leur valuation.
Par exemple :

```
> ifactors(60);
                    [1, [[2, 2], [3, 1], [5, 1]]]
```

5 5 Une première application

Calculez $\varphi(100)$.
Trouvez, DE TÊTE, les trois derniers chiffres de :

$$789587546378637837837834545364653897897^{2085827145914585514551351351351474540982582580001}$$

Vérifiez directement avec Maple...
Qu'en concluez-vous sur le calcul de $x^m[n]$ lorsque x et n sont premiers entre eux ?

5 6 Résolution de x^m congru à a modulo n

Par exemple, on cherche un entier x tel que $x^{130355971} \equiv 3208718\ [9108163]$.

6 Oraux de Centrale

6 1 Exercice

Soit p un nombre premier. Pour tout entier $k \in \{1,...,p-1\}$, on pose :

$$p_k = \frac{(p-1)!}{k(p-k)}$$

1. Cette question est à traiter à l'aide d'un logiciel de calcul formel. Les instructions `ithprime` et `numer` de Maple pourront être utiles.

a. Étudier la divisibilité de l'entier $\sum_{k=1}^{p-1} p_k$ par p pour les 20 nombres premiers p.

b. On note $\frac{a_p}{b_p}$ la fraction irréductible représentant le rationnel $\sum_{k=1}^{p-1} \frac{1}{k}$. Étudiez la divisibilité de a_p par p^2 pour les 20 premiers nombres premiers.

2. On suppose que le nombre premier p est supérieur ou égal à 5 et on se place dans le corps $\mathbb{Z}_p$. On note $\overline{k}$ la classe d'un entier $k \in \mathbb{Z}$ dans $\mathbb{Z}_p$.

a. Montrer que $\overline{(p-1)!} = -1$.

b. Montrer que pour tout $k \in \{1,...,p-1\}$, on a : $\overline{p_k} = \left(\overline{k^2}\right)^{-1}$.

c. Montrer que $\sum_{k=1}^{p-1} \left(\overline{k^{-1}}\right)^2 = \sum_{k=1}^{p-1} \overline{k}^2$. Conclure quant à la divisibilité de $\sum_{k=1}^{p-1} p_k$ par p.

3. Exprimer $pb_p \sum_{k=1}^{p-1} p_k$ en fonction de a_p. Qu'en conclut-on ?

6 2 Exercice sur la fonction de Möbius

La fonction $\mu : \mathbb{N}^\star \to \{-1,0,1\}$ de MÖBIUS est définie suivant $\mu(1) = 1$ et, pour tout $n \geqslant 2$, $\mu(n) = (-1)^r$ si n est un produit de r nombres premiers deux à deux distincts, $\mu(n) = 0$ si n est divisible par le carré d'un nombre premier.

1. Montrer que μ est une fonction *multiplicative*, c'est-à-dire : $\mu(1) = 1$ et, si a et b sont premiers entre eux, $\mu(ab) = \mu(a)\mu(b)$.

2. On note $d \mid n$ pour signifier que l'entier naturel $d \in \mathbb{N}^\star$ divise $n \in \mathbb{N}^\star$. Montrer que :

$$\forall n \geqslant 2, \quad \sum_{d|n} \mu(d) = 0$$

3. Calculer les $\mu(n)$ pour $n \in \{1,2,...,20\}$ à l'aide de Maple *sans utiliser* de commande Maple détectant les nombres premiers (de type `isprime` par exemple)

ÉCOLE POLYTECHNIQUE

ÉCOLE SUPÉRIEURE DE PHYSIQUE ET CHIMIE INDUSTRIELLES

CONCOURS D'ADMISSION 2008

FILIÈRE MP - OPTION PHYSIQUE ET SCIENCES DE L'INGÉNIEUR **FILIÈRE PC**

COMPOSITION D'INFORMATIQUE

(Durée : 2 heures)

L'utilisation des calculatrices **n'est pas autorisée** pour cette épreuve.
Le langage de programmation choisi par le candidat doit être spécifié en tête de la copie.

⋆ ⋆ ⋆

Ave Cesar (zud bdrzq)

On cherche à crypter un texte t de longueur n composé de caractères en minuscules (soit 26 lettres différentes) représentés par des entiers compris entre 0 et 25 ($0 \leftrightarrow$ `a`, $1 \leftrightarrow$ `b`, $\ldots 25 \leftrightarrow$ `z`). Nous ne tenons pas compte des éventuels espaces.

Ainsi, le texte `ecolepolytechnique` est représenté par le tableau suivant où la première ligne représente le texte, la seconde les entiers correspondants, et la troisième les indices dans le tableau t.

e	c	o	l	e	p	o	l	y	t	e	c	h	n	i	q	u	e
4	2	14	11	4	15	14	11	24	19	4	2	7	13	8	16	20	4
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17

Codage de César

Ce codage est le plus rudimentaire que l'on puisse imaginer. Il a été utilisé par Jules César (et même auparavant) pour certaines de ses correspondances. Le principe est de décaler les lettres de l'alphabet vers la gauche de 1 ou plusieurs positions. Par exemple, en décalant les lettres de 1 position, le caractère `a` se transforme en `z`, le `b` en `a`, ... le `z` en `y`. Le texte `avecesar` devient donc `zudbdrzq`.

Question 1 Que donne le codage du texte `maitrecorbeau` en utilisant un décalage de 5 ?

Question 2 Écrire la fonction codageCesar(t, n, d) qui prend en arguments le tableau t, sa longueur n et un entier d ; et qui retourne un tableau de même taille que t contenant le texte t décalé de d positions.

Question 3 Écrire de même la fonction decodageCesar(t, n, d) prenant les mêmes arguments mais qui réalise le décalage dans l'autre sens.

Pour réaliser ce décodage, il faut connaître la valeur du décalage. Une manière de la déterminer automatiquement est d'essayer de deviner cette valeur. L'approche la plus couramment employée est de regarder la fréquence d'apparition de chaque lettre dans le texte crypté. En effet, la lettre la plus fréquente dans un texte suffisamment long en français est la lettre `e`.

Question 4 Écrire la fonction frequences(t', n) qui prend en argument un tableau t' de taille n représentant le texte crypté ; et qui retourne un tableau de taille 26 dont la case d'indice i contient le nombre d'apparitions du nombre i dans t ($0 \leq i < 26$).

Question 5 Écrire la fonction decodageAuto(t', n) qui prend en argument le tableau t' de taille n représentant le texte crypté ; et qui retourne le texte t d'origine (en calculant la clé pour que la lettre `e` soit la plus fréquente dans le texte décrypté).

Codage de Vigenère

Au XVIème siècle, Blaise de Vigenère a modernisé le codage de César très peu résistant de la manière suivante. Au lieu de décaler toutes les lettres du texte de la même manière, on utilise un texte clé qui donne une suite de décalages.

Prenons par exemple la clé `concours`. Pour crypter un texte, on code la première lettre en utilisant le décalage qui envoie le `a` sur le `c` (la première lettre de la clé). Pour la deuxième lettre, on prend le décalage qui envoie le `a` sur le `o` (la seconde lettre de la clé) et ainsi de suite. Pour la huitième lettre, on utilise le décalage `a` vers `s`, puis, pour la neuvième, on reprend la clé à partir de sa première lettre. Sur l'exemple `ecolepolytechnique` avec la clé `concours`, on obtient : (la première ligne donne le texte, la seconde le texte crypté et la troisième la lettre de la clé utilisée pour le décalage)

e	c	o	l	e	p	o	l	y	t	e	c	h	n	i	q	u	e
g	q	b	n	s	j	f	d	a	h	r	e	v	h	z	i	w	s
c	o	n	c	o	u	r	s	c	o	n	c	o	u	r	s	c	o

Question 6 Donner le codage du texte `becunfromage` en utilisant la clé de codage `jean`.

Question 7 Écrire la fonction codageVigenere(t, n, c, k) qui prend comme arguments un tableau t de taille n représentant le texte à crypter, et un tableau d'entiers c de longueur k donnant la clé servant au codage ; et qui retourne un tableau de taille n contenant le texte crypté t'.

Maintenant, on suppose disposer d'un texte t' assez long crypté par la méthode de Vigenère, et on veut retrouver le texte t d'origine. Pour cela, on doit trouver la clé c ayant servi au codage. On procède en deux temps : 1) détermination de la longueur k de la clé c, 2) détermination des lettres composant c.

La première étape est la plus difficile. On remarque que deux lettres identiques dans t espacées de $\ell \times k$ caractères (où ℓ est un entier et k la taille de la clé) sont codées par la même lettre dans t'. Mais cette condition n'est pas suffisante pour déterminer la longueur k de la clé c puisque des répétitions peuvent apparaître dans t' sans qu'elles existent dans t. Par exemple, les lettres `t` et `n` sont toutes deux codées par la lettre `h` dans le texte crypté à partir de `ecolepolytechnique` avec `concours` comme clé. Pour éviter ce problème, on recherche les répétitions non pas d'une

lettre mais de séquences de lettres dans t' puisque deux séquences de lettres répétées dans t, dont les premières lettres sont espacées par $\ell \times k$ caractères, sont aussi cryptées par deux mêmes séquences dans t'.

Dans la suite de l'énoncé, on ne considère que des séquences de taille 3 en supposant que toute répétition d'une séquence de 3 lettres dans t' provient *exclusivement* d'une séquence de 3 lettres répétée dans t. Ainsi, la distance séparant ces répétitions donne des multiples de k.

La valeur de k est obtenue en prenant le PGCD de tous ces multiples. Si le nombre de répétitions est suffisant, on a de bonnes chances d'obtenir la valeur de k. On suppose donc que cette assertion est vraie.

Question 8 Écrire la fonction pgcd(a, b) qui calcule le PGCD des deux entiers strictement positifs a et b par soustractions successives de ses arguments.

Question 9 Écrire la fonction pgcdDesDistancesEntreRepetitions(t', n, i) qui prend en argument le texte crypté t' de longueur n et un entier i $(0 \leq i < n - 2)$ qui est l'indice d'une lettre dans t' ; et qui retourne le pgcd de toutes les distances entre les répétitions de la séquence de 3 lettres $\langle t[i], t[i+1], t[i+2] \rangle$ dans la suite du texte $\langle t[i+3], t[i+4], \ldots t[n-1] \rangle$. Cette fonction retourne 0 s'il n'y a pas de répétition.

Question 10 Écrire la fonction longueurDeLaCle(t', n) qui prend en argument le texte crypté t' de longueur n ; et qui retourne la longueur k de la clé de codage.

Question 11 Donner le nombre d'opérations réalisées par la fonction `longueurDeLaCle` en fonction de la longueur n ? (On ne comptera que le nombre d'appels à la fonction PGCD).

Question 12 Une fois la longueur de la clé connue, donner une idée d'algorithme permettant de retrouver chacune des lettres de la clé. (Il s'agit de décrire assez précisément l'algorithme plutôt que d'écrire le programme).

Question 13 Écrire la fonction decodageVigenereAuto(t', n) qui prend en argument le tableau t' de taille n représentant le texte crypté ; et qui retourne le texte t d'origine. (On n'hésitera pas à recopier des parties de texte dans des tableaux intermédiaires).

* *
*

TD N°

2

Permutations, tris et polynômes

Nous travaillerons aujourd'hui sur les deux dernières épreuves de 2 heures de l'X/ENS Cachan. Comme il s'agit d'une épreuve écrite, on s'attachera à rédiger des solutions sur papier, présentées de manière claire, après avoir travaillé sur brouillon-papier et brouillon-écran.

1 Sujet 2011 - Permutations

On ne s'occupera aujourd'hui que des premières questions destinées à se familiariser avec ce genre d'épreuve et dégager quelques réflexes qui aideront à traiter le second sujet.

Si on rédige en Maple, on peut utiliser pour représenter les permutations (puis les polynômes), des listes ou des tableau (type `array`).

Les plus commodes à utiliser sont les listes. Nous introduisons ici les primitives `allouer` et `taille` souvent introduites dans les sujets même si la première est inutile avec les listes de Maple. Elles faciliteront cependant la tâche du correcteur.

```
> allouer := proc(m)
    RETURN([0 $ m])
  end:
> taille := proc(t)
    RETURN(nops(t))
  end:
```

1 1 Ordre d'une permutation

1. Diverses solutions sont possibles : réfléchir à ce qui empêche une *application* d'être une permutation. En Maple, les deux éléments de $\mathcal{B}_2$ sont `true` et `false`. Mais avant tout, comment représenter une permutation à l'aide d'une liste ?
2. Normalement, il ne faut pas oublier de tester si la composition est possible avant de l'effectuer mais l'énoncé nous dispense de le faire...
3. Si `u` est votre inverse, que vaut `u[t[i]]` ?
4. No comment...
5. On pourra introduire la liste correspondant à l'identité.

2 Sujet 2010 - Échangeurs de polynômes

Nous allons travailler sur des listes plutôt qu'avec des tableaux. On a souvent besoin de considérer la *tête* d'une liste qui est son premier élément et la *queue* d'une liste qui est la liste privée de son premier élément.

Nous pouvons donc introduire deux primitives :

```
> Tete := proc(L)
   RETURN(L[1])
  end:
> Queue := proc(L)
   RETURN(L[2..-1])
  end:
```

Pour la beauté du geste, vous essaierez d'éviter les boucles *pour* ou *tant que* pour répondre aux 5 premières questions au moins.

1. Il est plus léger ici d'utiliser l'algorithme de HÖRNER pour évaluer un polynôme. Prenons l'exemple de $P(x) = 3x^5 - 2x^4 + 7x^3 + 2x^2 + 5x - 3$. Le calcul classique nécessite 5 additions et 15 multiplications.

   On peut faire pas mal d'économies de calcul en suivant le schéma suivant :

$$\begin{aligned}
P(x) &= \underbrace{a_n x^n + \cdots + a_2 x^2 + a_1 x}_{\text{on met } x \text{ en facteur}} + a_0 \\
&= \left(\underbrace{a_n x^{n-1} + \cdots + a_2 x}_{\text{on met } x \text{ en facteur}} + a_1 \right) x + a_0 \\
&= \cdots \\
&= (\cdots(((a_n x + a_{n-1})x + a_{n-2})x + a_{n-3})x + \cdots)\, x + a_0
\end{aligned}$$

Ici cela donne $P(x) = (((((3x)-2)x+7)x+2)x+5)x-3$ c'est-à-dire 5 multiplications et 5 additions. En fait il y a au maximum 2×degré de P opérations (voire moins avec les zéros).

Proposez ici une version récursive utilisant la liste des coefficients et nos primitives `Tete` et `Queue`.

2. Encore une fois, une version récursive est rapide à écrire. Vous aurez peut-être besoin de distinguer trois conditions. Utilisez alors la structure conditionnelle :

```
if ... then ...
elif ... then ...
else ...
fi
```

3. Une version récursive est attendue. Vous aurez sûrement besoin de concaténer deux listes ou d'ajouter un élément à une liste.

Pour ajouter un 0 en bout de liste par exemple, on peut faire :

```
liste := [op(liste),0]
```

4. Une structure conditionnelle `if...elif...else...fi` traduit l'énoncé. On pourra utiliser la commande `signum(n)` qui renvoie 1 si $n > 0$, -1 si $n < 0$ et 0 sinon.

5. Voici une petite question qui cache un chapitre de l'option informatique consacrée aux tris !... Le sujet est vaste... Pour votre culture, nous aborderons tout de même deux tris possibles et leur complexité : le tri par insertion et le tri fusion.

Tri par insertion C'est le tri qui correspond à un joueur qui classe ses cartes.

Nous privilégierons une version récursive bien sûr.

Commencez par créer une procédure récursive qui insère un élément dans une liste. Comme vous avez lu tout le sujet, vous savez qu'à la question suivante, il sera utile de trier une liste avec deux relations d'ordre différentes.

Vous allez donc créer une procédure `insere := proc(Ordre,P,Liste)` qui insère le polynôme `P` dans la liste de polynômes `Liste` selon l'ordre croissant correspondant à `Ordre` (qui sera par exemple `Compare_neg` dans cette question).

Pour obtenir la liste triée, il suffit de trier un à un les éléments de la liste avec une procédure récursive `tri := proc(Ordre,Liste)` qui utilise `insere`.

Par exemple, vous devriez obtenir :

```
> tri(Compare_neg,[[1],[0,-1],[0,1]]);

                        [[0, 1], [0, -1], [1]]
```

qui correspond à la sixième figure proposée dans le sujet.

Combien d'opérations élémentaires la procédure va-t-elle effectuer dans le pire des cas ?

Tri fusion On utilise la méthode diviser pour régner : on « coupe » le tableau à trier en deux, on trie chaque tableau et on fusionne les deux tableaux obtenus, de manière récursive.

Il faut donc commencer par créer une procédure

```
divise := proc(Ordre,L)
```

qui divise la liste L en deux listes de longueurs égales ou presque, selon la parité de la taille.

Il faudra ensuite créer une procédure récursive

```
fusionne := proc(Ordre,L1,L2)
```

qui prend deux listes triées et en fait une seule liste triée.

On utilise ensuite ces deux procédures pour créer une procédure récursive

```
tri_f := proc(Ordre,Liste)
```

qui trie la liste L.

Pour la *complexité*, on peut supposer que la division s'effectue à temps constant et que la fusion de deux listes triées est de l'ordre de la taille n du tableau.

Soit K_n la complexité du tri d'une liste de taille n.

On obtient donc que K_n est de l'ordre de $2K_{\lfloor n/2 \rfloor} + n$.

Il s'agit donc d'étudier la suite de terme général $u_n = 2u_{\lfloor n/2 \rfloor} + n$ avec $u_0 = u_1 = 0$.

Montrez que u_n est croissante.

En introduisant la suite définie par $x_k = u_{2^k}$, montrez que :

$$n\lfloor \log_2(n) \rfloor \leqslant u_n \leqslant 2n\left(\lfloor \log_2(n) \rfloor + 1\right)$$

6. Tout est expliqué... Après ce gros effort de récursion, vous pouvez utiliser une boucle *pour*. Utilisez à bon escient `RETURN` qui permet de sortir de la procédure, ce qui est adapté à ces procédure de vérification pour économiser du temps : on en sort dès qu'on tombe sur un os.

```
> verifier_permute([2,3,1],[[0,1],[0,-1],[1]]);

                                true
```

7. On peut utiliser ici une triple boucle *pour* indexée sur les coefficients a, b et c.

Attention ! < est une relation d'ordre binaire en Maple ce qui interdit l'utilisation directe de `a<b<c<d`. On passera par :

```
(x<y)and (y<z)
```

Pour traduire la double inégalité $x < y < z$.

```
> est_echangeur_aux([2,4,1,3],2);

                                true

> est_echangeur_aux([2,4,1,3],1);

                                false
```

8. Pas de difficulté particulière :

```
> est_echangeur([2,4,1,3]);

                                false
```

9. L'utilisation de `sum` dans une récursion est peu fructueuse en Maple. On préférera une double boucle.

```
> nombre_echangeurs(4);

                                22
```

10. On traduit simplement l'énoncé.

11. C'est la question compliquée...

```
> enumerer_echangeurs(4);
[
[1,2,3,4],[1,2,4,3],[1,3,2,4],[1,3,4,2],[1,4,2,3],[1,4,3,2],[2,1,3,4],

[2,1,4,3],[2,3,1,4],[2,3,4,1],[2,4,3,1],[3,1,2,4],[3,2,1,4],[3,2,4,1],

[3,4,1,2],[3,4,2,1],[4,1,2,3],[4,1,3,2],[4,2,1,3],[4,2,3,1],[4,3,1,2],

[4,3,2,1]
]
```

3 Un petit oral de Centrale pour la route...

Exercice 2 - 1 Centrale, PC 2010

1. Soit $n \in [\![0, 10]\!]$. Montrer en utilisant Maple qu'il existe $P_n \in \mathbb{R}[X]$ tel que

$$\forall t \in \mathbb{R}, \quad \sin(nt) = P_n(\cos t)\sin t$$

2. Soit $n \in \mathbb{N}$. Montrer qu'il existe un unique $P_n \in \mathbb{R}[X]$ tel que

$$\forall t \in \mathbb{R}, \quad \sin(nt) = P_n(\cos t)\sin t$$

3. Montrer que $\forall n \in \mathbb{N}, \quad P_{n+2} = 2XP_{n+1} - P_n$

4. Écrire une procédure permettant de calculer P_n.

— Optionnel —

5. Montrer que l'application $(P, Q) \mapsto \displaystyle\int_{-1}^{1} P(t)Q(t)\sqrt{1-t^2}\,dt$ définit un produit scalaire sur $\mathbb{R}[X]$.

6. Montrer que la famille $(P_n)_{n\in\mathbb{N}}$ est une famille orthogonale de $\mathbb{R}[X]$.

Je vous donne la partie Maple. Vous noterez l'utilisation de `subs` pour les changements de variable, de l'option `remember` pour les tableaux récursifs, et `evalb(test)` qui évalue si un test est vrai ou faux.

```
# a)
for n from 0 to 10 do
    Q := expand(sin(n * t) / sin(t)):
    P[n] := subs(cos(t)=x, Q);
    od;

# c)
tchebychev := proc(n)
    option remember;
    if n=0 then
        RETURN(0)
    elif n=1 then
        RETURN(1)
    else
        RETURN(expand(2*x*tchebychev(n-1) - tchebychev(n-2)));
    fi;
```

```
    end:

# ça marche ?

for n from 0 to 10 do
    evalb(tchebychev(n) = P[n]);
    od;

tchebychev(35);
```

ÉCOLE POLYTECHNIQUE – ÉCOLE NORMALE SUPÉRIEURE DE CACHAN

ÉCOLE SUPÉRIEURE DE PHYSIQUE ET DE CHIMIE INDUSTRIELLES

CONCOURS D'ADMISSION 2011 — FILIÈRE MP HORS SPÉCIALITÉ INFO
FILIÈRE PC

COMPOSITION D'INFORMATIQUE – B – (XEC)

(Durée : 2 heures)

L'utilisation des calculatrices n'est pas autorisée pour cette épreuve.
Le langage de programmation choisi par le candidat doit être spécifié en tête de la copie.

⋆ ⋆ ⋆

Sur les permutations

La notion mathématique de permutation formalise la notion intuitive de réarrangement d'objets discernables. La permutation est une des notions fondamentales de la combinatoire, l'étude des dénombrements et des probabilités discrètes. Elle sert par exemple à étudier sudoku, Rubik's cube, etc. Plus généralement, on retrouve la notion de permutation au cœur de certaines théories des mathématiques, comme celle des groupes, des déterminants, de la symétrie, etc.

Une *permutation* est une bijection d'un ensemble E dans lui-même. On ordonne un ensemble E fini de taille n en numérotant ses éléments à partir de 1 : $x_1, x_2, \ldots x_n$. En pratique, puisque seuls les réarrangements des éléments de E nous intéressent, on considère l'ensemble E_n des *indices* qui sont les entiers de 1 à n, bornes comprises. On représente alors simplement une application f sur E_n par un tableau t de taille n dont les éléments sont des indices. Autrement dit, $f(k)$ est $t[k]$, où $t[k]$ désigne le contenu de la case d'indice k du tableau t, et $t[k]$ est lui même un indice. On notera que f est une permutation, si et seulement si les contenus des cases de t sont exactement les entiers de E_n.

Dans tout le problème, les tableaux sont indicés à partir de 1. L'accès à la $i^{\text{ème}}$ case d'un tableau t de taille n est noté $t[i]$ dans l'énoncé, pour i entier compris entre 1 et n au sens large. Quel que soit le langage utilisé, on suppose que les tableaux peuvent être passés comme arguments des fonctions et renvoyés comme résultat. En outre, il existe une primitive allouer(n) pour créer un tableau de taille n (le contenu des cases du nouveau tableau ont des valeurs inconnues), et une primitive taille(t) qui renvoie la taille du tableau t. Enfin, les booléens vrai et faux sont utilisés dans certaines questions de ce problème. Bien évidemment, le candidat reste libre d'utiliser les notations propres au langage dans lequel il compose.

Partie I. Ordre d'une permutation

Question 1 Écrire une fonction estPermutation(t) qui prend une application (représentée par un tableau d'entiers t) en argument et vérifie que t représente bien une permutation. Autrement dit, la fonction renvoie vrai si t représente une permutation et faux sinon.

On suppose désormais, sans avoir à le vérifier, que les tableaux d'entiers (de taille n) donnés en arguments aux fonctions à écrire représentent bien des permutations (sur E_n). Dans cet esprit, on confond par la suite les permutations et les tableaux d'entiers qui les représentent en machine. Plus généralement, si l'énoncé contraint les arguments des fonctions à écrire, le code de ces fonction sera écrit en supposant que ces contraintes sont satisfaites.

Question 2 Écrire une fonction composer(t,u) qui prend deux permutations sur E_n en arguments et renvoie la composée $t \circ u$ de t et de u. On rappelle que la composée $f \circ g$ de deux applications est définie comme associant $f(g(x))$ à x.

Question 3 Écrire une fonction inverser(t) qui prend une permutation t en argument et renvoie la permutation inverse t^{-1}. On rappelle que l'application inverse f^{-1} d'une bijection est définie comme associant x à $f(x)$.

La notation t^k désigne t composée k fois, — la définition est correcte en raison de l'associativité de la composition. On définit l'*ordre* d'une permutation t comme le plus petit entier k non nul tel que t^k est l'identité.

Question 4 Donner un exemple de permutation d'ordre 1 et un exemple de permutation d'ordre n.

Question 5 En utilisant la fonction composer, écrire une fonction ordre(t) qui renvoie l'ordre de la permutation t.

Partie II. Manipuler les permutations

La *période* d'un indice i pour la permutation t est définie comme le plus petit entier k non nul tel que $t^k(i) = i$.

Question 6 Écrire une fonction periode(t,i) qui prend en arguments une permutation t et un indice i et qui renvoie la période de i pour t.

L'orbite de i pour la permutation t est l'ensemble des indices j tels qu'il existe k avec $t^k(i) = j$.

Question 7 Écrire une fonction estDansOrbite(t,i,j) qui prend en arguments une permutation t et deux indices, et qui renvoie vrai si j est dans l'orbite de i et faux sinon.

Une transposition est une permutation qui échange deux éléments *distincts* et laisse les autres inchangés.

Question 8 Écrire une fonction estTransposition(t) qui prend une permutation t en argument et renvoie vrai si t est une transposition et faux sinon.

Un cycle (simple) est une permutation dont exactement une des orbites est de taille strictement supérieure à un. Toutes les autres orbites, s'il y en a, sont réduites à des singletons.

Question 9 Écrire une fonction estCycle(t) qui prend une permutation t en argument et renvoie vrai si t est un cycle et faux sinon.

Partie III. Opérations efficaces sur les permutations

On commence par écrire une fonction qui calcule les périodes de tous les éléments de E_n et qui soit la plus efficace possible.

Question 10 Écrire une fonction periodes(t) qui renvoie le tableau p des périodes. C'est-à-dire que $p[i]$ est la période de l'indice i pour la permutation t. On impose un coût linéaire, c'est-à-dire que la fonction periodes effectue au plus $C \cdot n$ opérations avec C constant et n taille de t.

On envisage ensuite le calcul efficace de l'itérée t^k. On remarque en effet que $t^k(i)$ est égal à $t^r(i)$, où r est le reste de la division euclidienne de k par la période de i.

Question 11 Écrire une fonction itererEfficace(t,k) ($k \geq 0$) qui calcule l'itérée t^k en utilisant le tableau des périodes. On rappelle que t^0 est l'identité. Si besoin est, les candidats pourront utliser la primitive reste(a,b) qui renvoie le reste de la division euclidienne de a par b ($a \geq 0$, $b > 0$).

La fonction ordre de la question 5 n'est pas très efficace. En effet, elle procède à de l'ordre de o compositions de permutations, où o est l'ordre de la permutation passée en argument. Or, o est de l'ordre de $n^{\sqrt{n}}$ dans le cas le pire. On peut améliorer considérablement le calcul de l'ordre en constatant que l'ordre d'une permutation est le plus petit commun multiple des périodes des éléments.

Question 12 Donner un exemple de permutation dont l'ordre excède strictement la taille.

Question 13 Écrire une fonction pgcd(a,b) qui prend en arguments deux entiers strictement positifs a et b, et renvoie le plus grand diviseur commun de a et b. On impose un calcul efficace selon l'algorithme d'Euclide qui repose sur l'identité pgcd(a,b) = pgcd(b,r), avec r reste de la division euclidienne de a par b.

Question 14 Écrire une fonction ppcm(a,b) qui prend en arguments deux entiers strictement positifs a et b, et renvoie le plus petit commun multiple de a et b. On utilisera l'identité ppcm(a,b) = $(a \times b)$/pgdc(a,b).

ÉCOLE POLYTECHNIQUE

ÉCOLE SUPÉRIEURE DE PHYSIQUE ET CHIMIE INDUSTRIELLES

CONCOURS D'ADMISSION 2010

FILIÈRE MP - OPTION PHYSIQUE ET SCIENCES DE L'INGÉNIEUR **FILIÈRE PC**

COMPOSITION D'INFORMATIQUE

(Durée : 2 heures)

L'utilisation des calculatrices **n'est pas autorisée** pour cette épreuve.
Le langage de programmation choisi par le candidat doit être spécifié en tête de la copie.

⋆ ⋆ ⋆

Échangeurs de Polynômes

Dans ce problème on s'intéresse à des polynômes à coefficients réels qui s'annulent en 0. Un tel polynôme P s'écrit donc $P(x) = a_1x + a_2x^2 + \cdots + a_mx^m$. Le but de ce problème est d'étudier la position relative autour de l'origine de plusieurs polynômes de ce type.

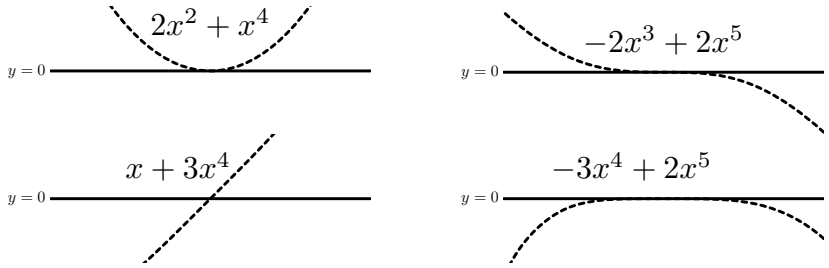
Dans tout le problème, les polynômes sont représentés par des tableaux de nombres flottants de la forme `[`a_1`;`a_2`;`$\cdots$`;`a_m`]`. Le nombre a_m peut être nul, par conséquent un polynôme donné admet plusieurs représentations sous forme de tableau. Ces tableaux sont indexés à partir de 1 et les éléments d'un tableau de taille m sont donc indexés de 1 à m. On suppose qu'il existe également une primitive `allouer(`m`)` pour créer un tableau de m cases. La taille m d'un tableau t est renvoyée par la primitive `taille(`t`)`. L'accès à la $i^{\text{ème}}$ case d'un tableau t est noté $t[i]$. Par ailleurs, on suppose que les tableaux peuvent être passés en argument ou renvoyés comme résultat de fonction, quel que soit le langage utilisé par le candidat pour composer. Enfin, les booléens `vrai` et `faux` sont utilisés dans certaines questions de ce problème. Le candidat est libre d'utiliser les notations propres à ces booléens dans le langage dans lequel il compose.

Le problème est découpé en deux parties qui peuvent être traitées de manière indépendante. Cependant, la **partie II** utilise les notions et notations introduites dans la **partie I**.

I. Permutation de n polynômes

Question 1 Afin de se familiariser avec cette représentation, écrire une fonction `evaluation` qui prend en arguments un polynôme P, représenté par un tableau, et un nombre flottant v, et qui renvoie la valeur de $P(v)$.

Nous commençons notre étude par quelques observations. Voici des exemples de graphes de polynômes autour de l'axe des abscisses.

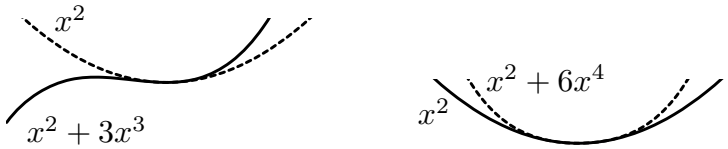


On remarque que le comportement au voisinage de l'origine est décrit par le premier monôme $a_k x^k$ dont le coefficient a_k est non nul (les coefficients a_1, ..., a_{k-1} étant donc tous nuls). En effet, quand x est petit, le terme $a_{k+1}x^{k+1} + \ldots + a_m x^m$ est négligeable devant le terme $a_k x^k$. Cet entier k est la *valuation* du polynôme à l'origine. Par exemple, la valuation du polynôme $-2x^3 - 3x^5 + 4x^7$ est 3. On remarque alors les deux règles suivantes au voisinage de l'origine :

- Si la valuation k est *paire*, le graphe du polynôme reste du *même* côté de l'axe des abscisses.
- Si la valuation k est *impaire*, le graphe du polynôme *traverse* l'axe des abscisses.

Question 2 Écrire une fonction `valuation` qui prend en argument un polynôme P et renvoie sa valuation. Par définition, cette fonction renverra 0 si P est le polynôme nul.

On s'intéresse maintenant aux positions relatives autour de l'origine des graphes de deux polynômes P_1 et P_2. La figure suivante montre les graphes de polynômes autour de l'origine.



On remarque que le comportement de ces graphes dépend de la parité de la valuation de la différence $P_1 - P_2$:

- Si la valuation de $P_1 - P_2$ est *paire*, les deux graphes se touchent mais ne se traversent pas à l'origine.
- Si la valuation de $P_1 - P_2$ est *impaire*, les deux graphes se traversent à l'origine.

Question 3 Écrire une fonction `difference` qui prend en arguments deux polynômes P_1 et P_2 (dont les tailles peuvent être différentes) et qui renvoie la différence des polynômes $P_1 - P_2$.

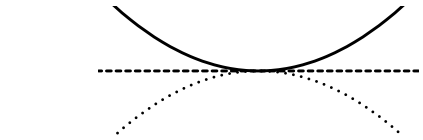
Question 4 Écrire une fonction `compare_neg` qui prend en arguments deux polynômes P_1 et P_2 et qui renvoie :

- un entier strictement négatif si $P_1(x)$ est plus petit que $P_2(x)$, pour x *négatif* assez petit
- 0 si les deux polynômes P_1 et P_2 sont égaux
- un entier strictement positif si $P_1(x)$ est plus grand que $P_2(x)$, pour x *négatif* assez petit.

On admettra sans démonstration que la fonction `compare_neg` définit une relation d'ordre.

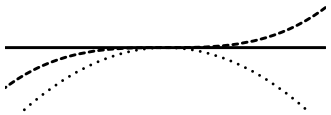
Enfin, passons à l'étude des graphes de trois polynômes. Les figures ci-après montrent les positions relatives de trois polynômes P_1, P_2 et P_3 autour de l'origine, avec la légende suivante :

$P_1(x)$ —— $P_2(x)$ ------- $P_3(x)$ ·····

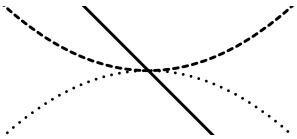


$P_1(x) = x^2 \quad P_2(x) = 0 \quad P_3(x) = -x^2$

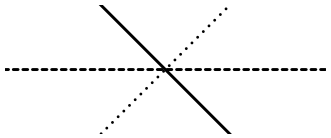
$P_1(x) = x^2 \quad P_2(x) = -x^3 \quad P_3(x) = 0$



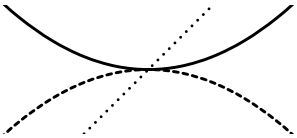
$P_1(x) = 0 \quad P_2(x) = x^3 \quad P_3(x) = -x^2$



$P_1(x) = -x \quad P_2(x) = x^2 \quad P_3(x) = -x^2$



$P_1(x) = -x \quad P_2(x) = 0 \quad P_3(x) = x$



$P_1(x) = x^2 \quad P_2(x) = -x^2 \quad P_3(x) = x$

Le choix de ces polynômes est fait pour qu'à chaque fois les inégalités $P_1(x) > P_2(x) > P_3(x)$ soient vérifiées pour x légèrement négatif. Maintenant, observons les positions relatives de ces graphes pour x légèrement positif. On remarque que l'ordre des courbes est *permuté* : on passe de l'ordre $P_1(x) > P_2(x) > P_3(x)$ à un autre ordre. La donnée des trois polynômes P_1, P_2 et P_3 définit donc une unique *permutation* π de $\{1, 2, 3\}$ telle que $P_{\pi(1)}(x) > P_{\pi(2)}(x) > P_{\pi(3)}(x)$, pour x positif et assez petit. On note que les *six* permutations de $\{1, 2, 3\}$ sont possibles, comme le montrent les six exemples ci-dessus.

De manière générale, on dit qu'une permutation π de $\{1, 2, \ldots, n\}$ *permute* les polynômes P_1, P_2, …, P_n si et seulement si :

$$\begin{aligned} & P_1(x) > P_2(x) > \ldots > P_n(x) && \text{pour } x \textit{ négatif} \text{ assez petit} \\ \text{et} \quad & P_{\pi(1)}(x) > P_{\pi(2)}(x) > \ldots > P_{\pi(n)}(x) && \text{pour } x \textit{ positif} \text{ assez petit} \end{aligned}$$

Ce qui était vrai pour trois polynômes ne l'est plus à partir de quatre polynômes : il existe des permutations qui ne permutent aucun ensemble de polynômes P_1, P_2, …, P_n.

Dans la suite, les permutations de $\{1, 2, \ldots, n\}$ seront représentées par des tableaux d'entiers de taille n, indexés à partir de 1 et contenant tous les entiers entre 1 et n.

Question 5 Écrire une fonction `tri` qui prend en argument un tableau t contenant n polynômes et qui le trie en utilisant la fonction `compare_neg`, de telle sorte que l'on ait $t[1](x) > t[2](x) > \cdots > t[n](x)$ pour x négatif et assez petit. Le candidat ne pourra pas utiliser pour cette question de fonction de tri prédéfinie dans la bibliothèque du langage qu'il utilise pour composer.

Question 6 Écrire une fonction `verifier_permute` qui prend en argument une permutation π de $\{1, 2, \ldots, n\}$ et un tableau t de même taille supposé trié par la fonction `tri`, et renvoie `vrai` si π permute les n polynômes $t[1]$, $t[2]$, …, $t[n]$ contenus dans t, et `faux` sinon. On pourra s'aider d'une fonction `compare_pos`, similaire à la fonction `compare_neg`, pour comparer deux polynômes pour x *positif* assez petit.

II. Échangeurs de n polynômes

Dans la suite, nous dirons qu'une permutation π de $\{1, 2, \ldots, n\}$ est un *échangeur* s'il existe n polynômes P_1, P_2, ..., P_n tels que π permute ces polynômes. Nous allons maintenant écrire des fonctions qui répondent aux questions suivantes : Une permutation π est-elle un échangeur ? Peut-on dénombrer les échangeurs ? Peut-on énumérer les échangeurs ?

Une condition nécessaire et suffisante pour qu'une permutation soit un échangeur est la suivante : une permutation π de $\{1, 2, \ldots, n\}$ est un échangeur si et seulement si il n'existe aucun entiers a, b, c, d tels que $n \geq a > b > c > d \geq 1$ et

$$\pi(b) > \pi(d) > \pi(a) > \pi(c) \quad \text{ou} \quad \pi(c) > \pi(a) > \pi(d) > \pi(b) \tag{1}$$

Question 7 Écrire une fonction `est_echangeur_aux` qui prend en argument une permutation π de $\{1, 2, \ldots n\}$ et un entier d tel que $1 \leq d \leq n$ et qui renvoie `vrai` s'il n'existe aucun entier a, b et c tels que $n \geq a > b > c > d$ et vérifiant (1), et `faux` sinon.

Question 8 En utilisant la fonction `est_echangeur_aux`, écrire une fonction `est_echangeur` qui prend en argument une permutation π de $\{1, 2, \ldots, n\}$ et renvoie `vrai` si π est un échangeur, et `faux` sinon.

On admet sans démonstration que la relation de récurrence suivante permet de compter le nombre $a(n)$ de permutations de $\{1, 2, \ldots, n\}$ qui sont des échangeurs :

$$a(1) = 1, \qquad a(n) = a(n-1) + \sum_{i=1}^{n-1} a(i) \times a(n-i)$$

Question 9 Écrire une fonction `nombre_echangeurs` qui prend un entier n en argument et renvoie le nombre d'échangeurs $a(n)$. Enfin, les deux questions suivantes ont pour but d'énumérer tous les échangeurs de $\{1, 2, \ldots, n\}$.

Question 10 Écrire une fonction `decaler` qui prend en arguments un tableau t de taille n et un entier v, et renvoie un nouveau tableau u de taille $n + 1$ tel que

$$\begin{cases} u[1] = v \\ u[i] = t[i-1] & \text{si } t[i-1] < v \text{ et } 2 \leq i \leq n+1 \\ u[i] = 1 + t[i-1] & \text{si } t[i-1] \geq v \text{ et } 2 \leq i \leq n+1 \end{cases}$$

L'algorithme que nous allons utiliser pour énumérer les échangeurs de $\{1, 2, \ldots, n\}$ consiste à énumérer successivement les échangeurs de $\{1, 2, \ldots, k\}$, pour tout k de 1 à n, dans un tableau t de taille $a(n)$. Si on suppose qu'un tableau t contient les m échangeurs de $\{1, \ldots, k\}$ entre les cases $t[1]$ et $t[m]$, on peut en déduire les échangeurs de $\{1, \ldots, k+1\}$ de la manière suivante : pour tout entier v entre 1 et $k+1$ et tout entier i entre 1 et m, on décale (à l'aide de la fonction `decaler`) l'échangeur $t[i]$ avec v puis on teste si le résultat est un échangeur (avec la fonction `est_echangeur_aux`).

Question 11 Écrire une fonction `enumerer_echangeurs` qui prend un entier n en argument et renvoie un tableau contenant les $a(n)$ échangeurs de $\{1, 2, \ldots, \mathtt{n}\}$. On pourra utiliser un second tableau pour stocker temporairement les nouveaux échangeurs.

* *
*

TD N°

3

Autour du calcul matriciel

1 Étude de quelques algorithmes

1.1 Fabriquons nos outils

1.1.1 Les primitives

Les matrices seront crées comme une liste de listes représentant les lignes.

Par exemple, $\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$ sera creée avec :

```
> m := [[1,2],[3,4],[5,6]]:
```

On obtiendra alors $m_{i,j}$ en entrant `m[i][j]`.
Pour avoir un affichage convivial, on crée une procédure convertissant la liste en tableau (type `array`) :

```
> affiche:=proc(T)
    print(convert(T,array))
  end:
```

On obtient l'affichage de le matrice en entrant `affiche(m)`.
On aura également besoin de deux primitives `Lignes` et `Cols` qui « compterons » le nombre de lignes et de colonnes de la matrice :

```
> Lignes := proc(M)
    RETURN(nops(M))
  end:

> Cols := proc(M)
    RETURN(nops(M[1]))
  end:
```

On aura souvent besoin de créer une matrice nulle d'une certaine taille :

```
> nulle := proc(lignes,colonnes)
    RETURN([[0 $ colonnes] $ lignes])
  end:
```

Notez bien l'ordre d'appel de `colonnes` et `lignes`.

1.1.2 Transposée d'une matrice

Déterminez une procédure `transpose := proc(M)` qui renvoie la transposée d'une matrice M.

1.1.3 Unité

Déterminer une procédure `Id := proc(n)` qui renvoie la matrice unité d'ordre n.

1.2 Pivot de Gauss

1.2.1 Opérations élémentaires

Créer tout d'abord une procédure `GJ(M) := proc(M)` qui fabrique une matrice où se juxtaposent la matrice initiale et la matrice identité.

Par exemple, $M = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix}$ est complétée en :

$$T = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

On montre ensuite qu'en effectuant des combinaisons sur les lignes ou des permutations de lignes, si on arrive à obtenir la matrice unité dans la partie gauche du tableau alors M est inversible et son inverse est la partie droite du tableau.
On commence par créer une procédure `mult_ligne := proc(k,M,j)` qui effectue la transformation $L_j \leftarrow k \times L_j$.

```
> mult_ligne := proc(k,M,i)
  local A,j;
  A := M;
  for j from 1 to Cols(M) do
    A[i][j] := k*M[i][j]
  od:
  RETURN(A)
  end:
```

Selon le même modèle, créer une procédure `comb_lignes := proc(ki,kj,M,i,j)` qui effectue la transformation $L_j \leftarrow k_i \times L_i + k_j \times L_j$.
Créer aussi une procédure `swap := proc(M,i,j)` qui effectue l'échange de lignes $L_i \leftrightarrow L_j$.
On peut alors effectuer une succession d'opérations élémentaires pour vérifier si M et inversible et obtenir alors son inverse :

```
> T := GJ(M): affiche(T);
```

$$\begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

```
> T := comb_lignes(-1,1,T,1,3): affiche(T);
```

$$\begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & -1 & -1 & 0 & 1 \end{bmatrix}$$

On continue ainsi jusqu'à obtenir :

$$\begin{bmatrix} 1 & 0 & 0 & 1/2 & -1/2 & 1/2 \\ 0 & 1 & 0 & -1/2 & 1/2 & 1/2 \\ 0 & 0 & 1 & 1/2 & 1/2 & -1/2 \end{bmatrix}$$

Il reste à extraire la partie gauche du tableau en créant une procédure `JG := proc(T)`. On se souviendra que `L[k..m]` renvoie la sous-liste de L contenant les éléments de `L[k]` jusque `L[m]`.

1 2 2 Calcul direct de l'inverse par la méthode de Gauss-Jordan

lLidée est de balayer le tableau par colonne.
Étant donné une colonne, on cherche un élément non nul. S'il n'y en a pas, la matrice n'est pas inversible et le système n'admet pas une unique solution ; sinon, on permute éventuellement deux lignes pour placer l'élément non nul de la colonne k sur la ligne k et on divise tous les éléments de la ligne par le nouveau a_{kk} pour obtenir 1.

Il reste ensuite à remplacer chaque ligne (autre que L_k) dont l'élément de la colonne k est non nul par $L_i - a_{ik} \times L_k$.
Analyser le programme suivant :

```
> inv_gauss := proc(M)
    local S,NoLigne,NoCol,E,k,pivot;
    if Cols(M) <>  Lignes(M) then
       RETURN('La matrice doit être carrée')
    fi;
    S := GJ(M);
    for NoCol from 1 to Lignes(M) do
       NoLigne := NoCol;
       while (S[NoLigne][NoCol] = 0) do
         if (NoLigne = Lignes(M)) then
            RETURN('Matrice non inversible')
         fi;
         NoLigne := NoLigne + 1;
       od;
       S := swap(S,NoLigne,NoCol);
       pivot := S[NoCol][NoCol];
       S := mult_ligne(1/pivot,S,NoCol);
       E := {seq(k,k=1..Lignes(T))} minus {NoCol};
       for k in E do
          S := comb_lignes(-S[k][NoCol],1,S,NoCol,k);
       od;
     od;
     RETURN(JG(S))
  end:
```

13 Généralisation

Afin de résoudre des systèmes linéaires, il ne faut pas se limiter aux matrices inversibles ni même aux matrices carrées.
On va essayer d'écrire la matrice originale sous forme « triangulaire supérieure » en généralisant cette configuration aux matrices « rectangulaires ».
On s'inspire de la fonction précédente, mais au lieu de travailler sur toutes les lignes, on ne transforme que les lignes en-dessous du pivot.
Il faudra donc faire attention maintenant à utiliser le minimum entre le nombre de lignes et le nombre de colonnes de la matrice.

```
> A:=[[4,1,-2,8,1],[5,-13,7,6,1],[-1,9,-11,9,1],[2,-8,12,-3,1]]:
> affiche(A);
```

$$\begin{bmatrix} 4 & 1 & -2 & 8 & 1 \\ 5 & -13 & 7 & 6 & 1 \\ -1 & 9 & -11 & 9 & 1 \\ 2 & -8 & 12 & -3 & 1 \end{bmatrix}$$

```
> affiche(tri_gauss(A));
```

$$\begin{bmatrix} 4 & 1 & -2 & 8 & 1 \\ 0 & -57 & 38 & -16 & -1 \\ 0 & 0 & 1216 & -1916 & -248 \\ 0 & 0 & 0 & -1924320 & -594624 \end{bmatrix}$$

1 3 1 Base de l'image - Rang d'une matrice

On écrit les vecteurs engendrant l'image de l'endomorphisme en ligne : bref, on transpose la matrice. En effet, ce sera plus simple d'étudier la matrice par lignes car il s'agit des sous-listes de la liste représentant la matrice. Suite à la réduction de `tri_gauss` on ne garde que les lignes non nulles : la famille obtenue est une base de l'image.
Déterminer une fonction `image := proc(M)` qui retourne une base de l'image puis une fonction `rang := proc(M)` qui renvoie le rang de la matrice M.

1 3 2 Calcul du déterminant

On utilise `tri_gauss` sur une matrice carrée : on obtient ainsi une matrice triangulaire. Le déterminant est égal au produit des éléments diagonaux à un détail près : il ne faut pas oublier de tenir compte de la parité du nombre d'échanges de lignes ainsi que des multiplications des lignes modifiées par combinaisons.

1 3 3 Polynôme caractéristique et valeurs propres

Déduisez-en une procédure `polycar := proc(M)` qui renvoie le polynôme caractéristique de M.
On pourra introduire `idx := proc(n,x)`, une variante de `id(n)`, qui renvoie $x\mathrm{I}_n$, puis une procédure `Ex := proc(M,x)` qui renvoie la matrice $\mathrm{M} - x\mathrm{I}_n$.

```
> B := [[1,1,0],[-1,-1,0],[1,-1,0]]:
> affiche(Ex(B,x))
```

$$\begin{bmatrix} -x+1 & 1 & 0 \\ -1 & -x-1 & 0 \\ 1 & -1 & -x \end{bmatrix}$$

```
> polycar(B);
```

$$-x^3$$

Comment déterminer à l'aide de Maple si cette matrice est diagonalisable ?
Déterminer un test `est_diagonalisable := proc(M)` qui renvoie vrai ou faux.

2 Maple à Centrale

Exercice 3 - 1 Centrale, 2010

Si $n \geq 2$, on note $\mathcal{B} = (\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n)$ la base canonique de $\mathbb{R}^n$.
Soit $f_n \in \mathcal{L}(\mathbb{R}^n)$ tel que $f_n(\boldsymbol{e}_i) = \boldsymbol{e}_{i+1}$ si $i \in [\![1, n-1]\!]$ et $f_n(\boldsymbol{e}_n) = \boldsymbol{e}_1$.

1. Écrire une procédure permettant d'obtenir la matrice A_n de f_n dans la base $\boldsymbol{e}$.
2. Dans cette question, $n = 6$.
   a. Calculer le polynôme caractéristique de A_6. Factoriser ce polynôme sur $\mathbb{R}[\mathrm{X}]$.
   On écrit $\chi = \prod_{1 \leq k \leq p} \mathrm{R}_k$ où les R_k sont irréductibles dans $\mathbb{R}[\mathrm{X}]$.
   b. Déterminer $\ker \mathrm{R}_k(f_6)$ pour $k \in [\![1, p]\!]$. Montrer que $\mathbb{R}^6 = \ker \mathrm{R}_1(f_6) \oplus \cdots \oplus \ker \mathrm{R}_p(f_6)$.
   c. Donner la matrice de f_6 dans une base adaptée à cette décomposition.
3. Si $n \geqslant 2$, calculer le polynôme caractéristique χ de A_n. Le factoriser sur $\mathbb{R}[\mathrm{X}]$.

Voyons quelques commandes utiles issues de la bibliothèque `LinearAlgebra` :

```
with(LinearAlgebra);
# a)
A := proc(n);
    M := Matrix(n);
    for i to n-1 do
        M[i+1, i] := 1;
        end do;
    M[1, n] := 1;
    return M
    end proc;

M := A(6);
I6 := IdentityMatrix(6);
latex(M);

# b)
#    i)
chi := CharacteristicPolynomial(M, x);
chi := factor(chi);
R := op(chi);

#    ii)
# Détermination des sev Ker(Rk(f))
F1 := NullSpace(M - I6);
F2 := NullSpace(M + I6);
F3 := NullSpace(M^2 + M + I6);
F4 := NullSpace(M^2 - M + I6);

P := <F1[1]|F2[1]|F3[1]|F3[2]|F4[1]|F4[2]>;
# Trois manières de vérifier que P est inversible :
Determinant(P);
NullSpace(P);
ColumnSpace(P);

#    iii)
# Matrice de f_6 dans un base adaptée :
P^(-1).M.P;

# c)
for n from 2 to 10 do
    CharacteristicPolynomial(A(n), x),\
    factor(CharacteristicPolynomial(A(n), x));
    end do;
```

Exercice 3 - 2 Centrale, 2010

1. Démontrer que, si deux endomorphismes u et v d'un espace vectoriel E commutent, alors, les sous-espaces propres de u et l'image de u sont stables par v.
Dans les deux cas suivants :

$$A = \begin{pmatrix} 20 & 12 & -4 & 12 \\ -4 & -3 & 9 & -5 \\ -4 & 1 & 5 & -5 \\ -8 & -10 & 6 & -2 \end{pmatrix} \quad \text{et} \quad A = \begin{pmatrix} -12 & -16 & -8 & -4 \\ 4 & 13 & 1 & -1 \\ 4 & 5 & 9 & -1 \\ 8 & 10 & 2 & 6 \end{pmatrix}$$

2. Préciser les matrices qui commutent avec A (structure, dimension, base éventuelle).

3. Etudier dans $\mathcal{M}_4(\mathbb{R})$, puis dans $\mathcal{M}_4(\mathbb{C})$, l'équation

$$X^2 = A$$

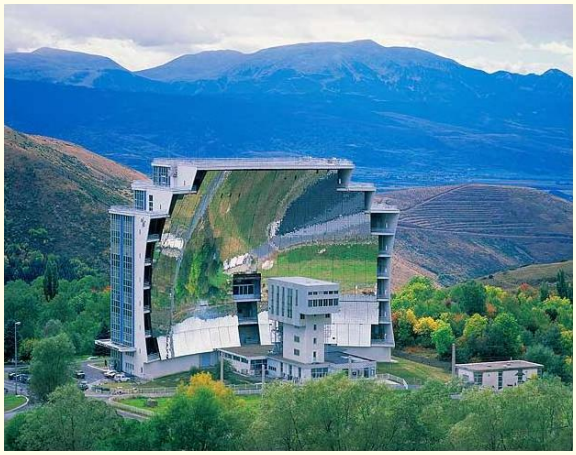
(nombre de solutions, un exemple de solution quand il y en a, somme et produit des solutions quand elles sont en nombre fini).

```
with(LinearAlgebra):
# a)
A := <<20|12|-4|12>, <-4|-3|9|-5>, <-4|1|5|-5>, <-8|-10|6|-2>>;
(vpA, P) := Eigenvectors(A);
P^(-1).A.P;

B := <<-12|-16|-8|-4>, <4|13|1|-1>, <4|5|9|-1>, <8|10|2|6>>;
(vpB, Q) := Eigenvectors(B);
Q^(-1).B.Q;

# c)
Determinant(A);
Determinant(B);
```

TD N°

4

Autour des formes quadratiques



1 Réduction d'une conique

On utilisera la bibliothèque **linalg** ou (exclusif !) la bibliothèque **LinearAlgebra**.
Je vous propose des solutions avec la première citée.
Testez **matrix** :

```
matrix([[1,2],[3,4]])
```

On entrera une forma quadratique $q(x) = ax^2 + bxy + cy^2 + dx + ey + f$ sous la forme d'une liste **[a,b,c,d,e,f]**.
Créez une fonction **q2a:=proc(Q)** qui renvoie la matrice de la partie quadratique de q.
La fonction **eigenvectors(A)** renvoie une liste contenant des listes du type $[\lambda, \dim(E_\lambda), \{\vec{v}_\lambda\}]$.
La fonction **normalize(liste)** renvoie $\frac{\texttt{liste}}{\|\texttt{liste}\|}$.
Créez une fonction **EV:=proc(Q)** qui renvoie une matrice orthogonale de vecteurs propres et la matrice diagonale associée.
Par exemple, pour Q = [1,6,1,4,0,1] on obtient :

$$\begin{bmatrix} \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix}, \begin{bmatrix} 4 & 0 \\ 0 & -2 \end{bmatrix}$$

ou dans l'ordre inverse.
La fonction **transpose(M)** renvoie la transposée de M et **evaml(A&*B)** renvoie la matrice produit de A par B.
En écrivant sous forme matricielle la forme quadratique exprimée dans la nouvelle base, affichez la forme quadratique réduite.
Dans l'exemple précédent, on doit par exemple obtenir :

$$4X^2 - 2Y^2 - 2\sqrt{2}X + 2\sqrt{2}Y + 1 +$$

On peut « récupérer » les coefficients à l'aide de la fonction **coeff**.
Par exemple, si on appelle QR le polynôme précédent, **coeff(QR,X^2)** renvoie 4.
Créez une fonction qui renvoie le type de conique (parabole, ellipse, hyperbole) ou la nature de la dégénérescence (réunion de deux droites, droites sécantes, vide, etc.), une équation réduite si possible du type $\alpha x^2 + \beta y^2 + \gamma = 0$, les éléments caractéristiques, un tracé de la conique dans l'ancien et le nouveau repère (on pourra utiliser **implicitplot**).

2 Réduction d'une quadrique

Si vous êtes en forme, adaptez ce qui vient d'être fait à l'étude générale d'une quadrique...

3 À Centrale

Exercice 4 - 1 Centrale, 2010

Soit $(\mathcal{S})$ la surface d'équation $\dfrac{x^2}{9} + y^2 - \dfrac{z^2}{4} = 1$

1. Nature de $(\mathcal{S})$? Donner un paramétrage de $(\mathcal{S})$. Tracer $(\mathcal{S})$ avec Maple.
2. Déterminer l'intersection de $(\mathcal{S})$ avec le plan d'équation $z = \alpha$.
3. Calculer le volume du solide défini par $\left\{M(x,y,z) \mid \dfrac{x^2}{9} + y^2 - \dfrac{z^2}{4} = 1 \text{ et } a \leq z \leq b\right\}$
4. La surface $(\mathcal{S})$ admet-elle des points singuliers ?

5. Donner l'équation du plan tangent à $(\mathcal{S})$ au point $M_t = (3\cos t, \sin t, 0)$.

Exercice 4 - 2 Centrale, 2010

Soit $(\mathcal{E})$ la surface d'équation $x^2 + y^2 + 4z^2 = 1$.

1. Représenter $(\mathcal{E})$.

2. Soit R_t la rotation d'axe dirigé et orienté par $\boldsymbol{u} = \begin{pmatrix} 4 \\ 3 \\ 0 \end{pmatrix}$ et d'angle t.
Soit (x', y', z') l'image de (x, y, z) par R_t. Exprimer (x', y', z') en fonction de (x, y, z).

3. Déterminer une équation de $(\mathcal{E}_t)$ l'image de $(\mathcal{E})$ par R_t.

Pour le c), un petit Joker...

```
# Avec LinearAlgebra

M := <x, y, z>;
Mt := Transpose(Rt).M;
eq := x^2 + y^2 + 4*z^2 - 1;
eqt := simplify(subs({x=Mt[1], y=Mt[2], z=Mt[3]}, eq));

# Avec linalg

M := matrix(3,1,[x, y, z]);
Mt := evalm(transpose(Rt) &* M);
eq := x^2 + y^2 + 4*z^2 - 1;
eqt := simplify(subs({x=Mt[1], y=Mt[2], z=Mt[3]}, eq));
```

TD N° 5

Courbes elliptiques

En 1995, le mathématicien anglais Andrew Wiles fit sensation en démontrant enfin le grand théorème de Fermat, trois siècles après sa formulation, à l'aide des courbes elliptiques ce qui surprit les spécialistes de la théorie des nombres. Elles jouent actuellement un rôle essentiel dans la sécurité informatique mais on les retrouve en géométrie, en mécanique. Elles vont nous permettre aujourd'hui de mettre un pied dans un domaine mathématique encore tout chaud...

1 Un peu de lecture...

Nous ne pouvons, à notre niveau, que parcourir rapidement ce riche domaine des mathématiques en admettant la plupart des résultats que vous démontrerez peut-être un jour. Pour les plus curieux, vous pourrez lire le très bel ouvrage de Marc HINDRY *Arithmétique : Primalité et codes, Théorie analytique des nombres, Equations diophantiennes, Courbes elliptiques* paru en 2008 chez Calvage & Mounet ainsi que l'*Introduction élémentaire à la théorie des courbes elliptiques*[a] de Marc JOYE de l'Université de Louvain[b], un des hauts lieux de la cryptographie, où vous retrouverez de nombreuses démonstrations.
Les ouvrages suivants proposent des activités pratiques :

— *Initiation à la cryptographie* de Gilles DUBERNET (Vuibert 2002) ;

— *Algèbre, arithmétique et maple* de Bernadette PERRIN-RIOU (Cassini 2000) ;

— *Mathématiques & informatique* de François MORAIN et Jean-Louis NICOLAS (Vuibert 1995).

2 Quelques résultats préliminaires

La méthode de la tangente et de la sécante dont nous allons parler n'est pas nouvelle. FERMAT, voire DIOPHANTE, ont tourné autour puis NEWTON, LAGRANGE, CAUCHY, SYLVESTER, POINCARÉ. Il faut pourtant attendre les années 1920 et André WEIL pour qu'elle soit parfaitement maîtrisée même si un article publié en allemand en 1896 par le Danois JUEL donne une description complète de la loi de groupe sur une cubique mais semble ne pas avoir été lu par ses contemporains[c]...
Il a cependant fallu attendre la fin du XX^{e} siècle pour l'utiliser en arithmétique avec le succès que l'on sait car ces courbes permettent d'obtenir efficacement des factorisations d'entiers comme le montra le néerlandais Hendrik LENSTRA.
Nous aurons besoin de connaître quelques résultats sur les groupes et les corps finis que vous avez découvert avec M. SAUVAGEOT[d].
Nous aurons également besoin de parler de manière intuitive du plan projectif $\mathbb{P}_2(\mathbf{K})$ que vous avez découvert lors du chapitre sur les quadriques et les polynômes homogènes sur $\mathbf{K}[X,Y,Z]$.
Disons qu'en gros, on définit sur $\mathbf{K}^3 \setminus \{(0,0,0)\}$ une relation d'équivalence :

$$(X,Y,Z) \equiv (X',Y',Z') \Leftrightarrow \exists t \in \mathbf{K}^*,\ (X',Y',Z') = t(X,Y,Z)$$

et que $\mathbb{P}_2(\mathbf{K})$ est l'ensemble des classes d'équivalence.
On établit une « correspondance » entre l'espace affine $\mathbb{A}_2(\mathbf{K})$ et $\mathbb{P}_2(\mathbf{K})$: au point de coordonnées (x,y) du plan affine, on fait correspondre le point de coordonnées $(X:Y:Z)$ dans le plan projectif en posant $x = \frac{X}{Z}$ et $y = \frac{Y}{Z}$. Se pose alors le problème du cas $Z = 0$. Ce cas correspond au « point à l'infini » où deux droites parallèles du plan projectif se croisent. Nous en distinguerons par la suite un particulier de coordonnées $(0:1:0)$ que nous noterons $\mathcal{O}$.

3 Loi de groupe sur une cubique

On peut montrer que dans le plan projectif, toute droite coupe une cubique d'équation $Y^2Z = X^3 + aXZ^2 + bZ^3$ en exactement trois points, en tenant compte de l'ordre de multiplicité,

a. http://sciences.ows.ch/mathematiques/CourbesElliptiques.pdf
b. http://www.uclouvain.be/crypto/publications/latest
c. Voir l'article de Norbert SCHAPPACHER www-irma.u-strasbg.fr/ schappa/NSch/Publications_files/DPP.pdf
d. Qui, comme vous le savez, est le traducteur d'une des « bibles » de la théorie des nombres : *Introduction à la théorie des nombres* paru en 2006 chez Vuibert...

lorsque a et b remplissent certaines conditions qui rendent la courbe « lisse » et sans racine double.
En fait, quand on repasse « en affine », on obtient $y^2 = x^3 + ax + b$. Vous avez évoqué le discriminant de $x^3 + ax + b$ lors de l'étude de la méthode de CARDAN pour résoudre les équations de degré 3. En fait, il faut que ce discriminant soit non nul : nous supposerons donc par la suite que $4a^3 + 27b^2 \neq 0$.
Il faut aussi que la caractéristique du corps soit différente de 2 ou 3 mais cela ne nous inquiète pas car nous allons travailler dans des corps finis de très grande caractéristique (factoriser par des petits nombres ne nécessite pas de sortir l'artillerie lourde...).
Bon, regardons les points à l'infini : $Z = 0 \Longrightarrow X = 0$. Il n'y a donc qu'un seul qui est dans la direction $x = 0$ et qui a pour coordonnées dans $\mathbb{P}_2$ $\mathcal{O}(0:1:0)$.

Par la suite, nous considérerons donc une cubique $\mathcal{E}$ d'équation $y^2 = x^3 + ax + b$ de discriminant non nul.
On notera $\mathcal{E}(\mathbf{K})$ les solutions (x, y) de l'équation $y^2 = x^3 + ax + b$ sur $\mathbf{K}^2$ auxquelles on ajoute $\mathcal{O}$.
Par exemple, on peut visualiser ce que cela donne sur $\mathbb{R}$:

```
> anim_elliptic:=proc()
   local P,a,b;
   P:=NULL;
   for a from -10 to 5 by 2 do
     for b from -10 to 5 by 2  do
       if (4*a^3+27*b^2)<>0 then
        P:=P,plots[implicitplot](y^2=x^3+a*x+b,x=-10..10,y=-10..10);
   fi;od;od;
   plots[display]([P],insequence=true,view=[-10..10,-10..10]);
  end:
```

Soit P un point quelconque. Une droite (P$\mathcal{O}$) est donc une droite passant par P et parallèle à l'axe $x = 0$.
On a admis que toute droite coupait $\mathcal{E}$ en un troisième point en tenant compte de la multiplicité. On peut donc définir une loi sur $\mathcal{E}(\mathbf{K})$:

— si P et Q sont deux points distincts de $\mathcal{E}(\mathbf{K})$, alors la droite (PQ) recoupe la courbe en un unique troisième point que nous noterons P $\star$ Q. La droite joignant P $\star$ Q et $\mathcal{O}$ recoupe la courbe en un troisième point que nous noterons P + Q.

— si P = Q, le point P $\star$ P sera l'intersection de la tangente avec la courbe. On admettra (ce sera vu plus tard dans l'année avec M. Sauvageot) qu'une courbe lisse admettant une équation implicite de la forme $f(x, y) = 0$ admet une tangente au point (x_0, y_0) d'équation :

$$(x - x_0)\frac{\partial f}{\partial x}(x_0, y_0) + (y - y_0)\frac{\partial f}{\partial y}(x_0, y_0) = 0$$

On peut vérifier que cette loi est commutative, que $\mathcal{O}$ en est l'élément neutre. et que le symétrique de P par rapport à l'axe $(0x)$ est le symétrique de P pour la loi + et qu'il appartient à la courbe.
Il resterait à vérifier l'associativité mais cela nécessiterait pas mal de travail et nous n'en avons pas le temps : vous pourrez étudier la littérature pour vous en convaincre.

1. Soit $P_1(x_1, y_1)$ et $P_2(x_2, y_2)$. Déterminez les coordonnées (x_3, y_3) de la somme $P_1 + P_2$ en distinguant les différents cas. On prendra $\mathbf{K} = \mathbf{R}$

Déduisez-en une procédure `somme := proc(P1,P2,a,b)` qui calcule les coordonnées de la somme $P_1 + P_2$ pour une courbe elliptique donnée dépendant des paramètres a et b. On pourra créer une procédure intermédiaire qui vérifie qu'un point appartient à la courbe elliptique de paramètres a et b. On symbolisera le point $\mathcal{O}$ par `[0]` (à ne pas confondre avec 0 ...).

Vous pourrez ensuite visualiser la construction :

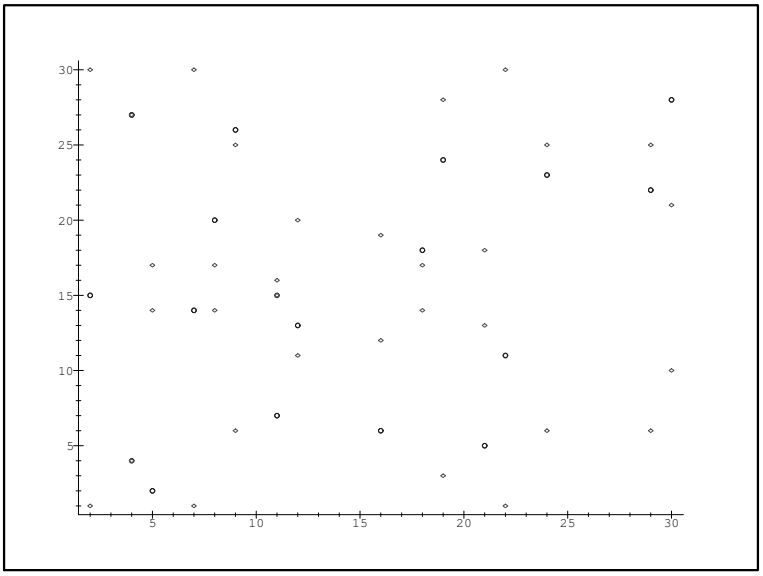
```
> dessin_elliptic:=proc(P1,P2,a,b)
    local P3,P4,y2,E,L1;
    P3 := somme(P1,P2,a,b);
    if P1 = [0] or P2 = [0] or P3 = [0]
       then ERROR('Point à l infini');fi;
    P4 := [P3[1],-P3[2]];
    E := plots[implicitplot](y^2 = x^3 + a*x+b,x=-10..10,y
       =-10..10 ,color=green);
    L1 := plot([P1,P4,P3],color=blue,thickness=3);
    plots[display]([E,L1]);
  end:

> point_courbe:=proc(x,a,b)
    RETURN([(x),(sqrt(x^3+a*x+b))]);
  end:

> dessin_elliptic(point_courbe(-2,-5,3),point_courbe(-1,-5,3)
   ,-5,3);
```

2. On veut maintenant tracer la courbe en travaillant dans $\mathbb{F}_p$ avec bien sûr p un nombre premier. Qu'est-ce qui va changer dans nos procédures ?

La recherche d'une racine carrée dans $\mathbf{Z}/n\mathbf{Z}$ n'est pas tout à fait évidente. Pendant vos temps libres, vous pourrez vérifier que si p est un nombre premier congru à 3 modulo 4, alors on arrive à s'en sortir mais si p est congru à 1, alors on passe par l'algorithme probabiliste de SHANK. Nous n'avons pas besoin d'être efficace ici car il s'agit seulement de dessiner la courbe avec une valeur de p petite donc une recherche exhaustive suffira.

Pour ce qui est de la recherche de l'inverse, on a déjà traité ce cas dans les TD précédents et Maple sait d'ailleurs se débrouiller : `1/16 mod 29` renvoie bien 20 car $16\times 20 \equiv 1[29]$ mais `1/15 mod 27` renvoie une erreur.

Voici par exemple la courbe elliptique dans $\mathbf{Z}/31\mathbf{Z}$ d'équation $y^2 = x^3 + 26x + 3$ où les points sont représentés par des losanges ainsi qu'une droite passant par deux points de la courbe dont les points ont été représentés par des cercles...



Voilà qui perturbe un peu nos habitudes !...

3. Que se passe-t-il dans $\mathbf{Z}/n\mathbf{Z}$ lorsque n est composé ? Nous allons voir plus loin que ce n'est pas un problème et qu'il s'agit plutôt de la clé du succès des courbes elliptiques.

4. Déterminez une procédure donnant les différents multiples d'un point dans le groupe associé à une courbe elliptique donnée.

```
> multiples(point_courbe_p(1,26,3,31),33,26,3,31);

[0], [2, 1], [16, 12], [22, 1], [7, 30], [11, 15], [12, 11],
[18, 14],[5, 14], [29, 6], [19, 28], [24, 6], [21, 13], [9, 6],
[30, 10],[8, 17], [4, 4], [4, 27], [8, 14], [30, 21], [9, 25],
[21, 18],[24, 25], [19, 3], [29, 25], [5, 17], [18, 17],
[12, 20],[11, 16], [7, 1], [22, 30], [16, 19], [2, 30], [0]
```

5. Déterminez une procédure `k_multiple_p:=proc(P,k,a,b,p)` qui calcule le k^e multiple du point P. Vous pourrez améliorer votre procédure en utilisant un algorithme d'exponentiation rapide en pensant à la décomposition en base 2 de k.

4 Factorisation d'entiers : méthode de Lenstra

1. Nous cherchons à factoriser un entier impair donné que nous savons composé (par exemple pour casser une clé RSA).

Dans notre procédure de calcul de somme dans $\mathbf{Z}/n\mathbf{Z}$, des divisions peuvent ne pas être effectuées. Maple renvoie alors un message d'erreur :

```
> 7/3 mod 21;
'Error, the modular inverse does not exist'
```

Pourquoi ? Si tel est le cas, que pouvons-nous dire de la factorisation de n ? Comment modifier alors la procédure d'addition sur $\mathcal{E}(\mathbf{Z}/n\mathbf{Z})$ pour obtenir un facteur de n ?

2. On dit qu'un nombre n est B-super friable si toutes les puissances entières de nombres premiers divisant n sont inférieures à B.

Par exemple, $2^4 \times 3^2 \times 11$ est 16-super friable.

3. Si le mathématicien allemand Helmut HASSE n'a pas été très inspiré dans ses choix politiques en soutenant le régime nazi [e], il a été un très grand algébriste et théoricien des nombres. Il a en particulier démontré le théorème suivant :

$$\left|\#\mathcal{E}(\mathbb{F}_p) - (p+1)\right| \leqslant 2\sqrt{p}$$

Sa démonstration fait référence à la fonction ζ de RIEMANN qui est a priori un outil d'analyse mais qui a eu une énorme influence en théorie des nombres comme nous le verrons dans un prochain TP.

4. Soit p un diviseur premier de n. On suppose que le cardinal c_p de $\mathcal{E}(\mathbb{F}_p)$ est B-super friable. Soit P un point de $\mathcal{E}(\mathbf{Z}/n\mathbf{Z})$ différent de $\mathcal{O}$ et soit k le PPCM de $(1,2,..,B)$. Comme c_p divise k et que $P \in \mathcal{E}(\mathbb{F}_p)$ car p divise n, alors, d'après le théorème de LAGRANGE, on sait que $kP = \mathcal{O}$ dans $\mathcal{E}(\mathbb{F}_p)$.

Cependant, on ne connaît pas p donc on va calculer kP dans $\mathcal{E}(\mathbf{Z}/n\mathbf{Z})$. Or des problèmes peuvent apparaître car il va y avoir des diviseurs de 0 : le point précédent nous permet de trouver un facteur de n. Si c'est un facteur trivial, on choisit une autre courbe. Sinon, on a un diviseur de n ! Si le calcul de kP ne pose pas de problème, on choisit un autre point P ou une autre courbe elliptique.

Le problème est dans le choix du bon point et de la bonne courbe...

e. Il avait pourtant un nom de famille pouvant plaire aux nazis mais ceux-ci ont toujours refusé son adhésion au parti car on le suspectait d'avoir des origines juives...

Voici un exemple proposé par Marc JOYE.

On veut factoriser $n = 540143$ par la méthode de LENSTRA. On va travailler sur les courbes elliptiques

$$\mathcal{E}_a : y^2 = x^3 + ax + 1$$

Le point $P(0,1) \neq \mathcal{O}$ appartient toujours à $\mathcal{E}_a$. Prenons $a = 1$. On vérifie que le discriminant est non nul modulo n.

On sait qu'un diviseur premier p de n est majoré par $\sqrt{n} \approx 735$

Le nombre de points de la courbe est majoré, d'après le théorème de HASSE, par $735 + 1 + 2\sqrt{735} < 792$. On choisit donc $B = 792$.

On choisit un nombre 792-super friable, par exemple $k = 2^9 \times 3^6$.

Si le calcul n'aboutit pas, on a gagné, sinon, on augmente a de 1 ou on augmente B (par exemple $B = 2^9 \times 3^6 \times 5^4$).

Il a été montré que la complexité de cet algorithme est en $\mathcal{O}\left(e^{(1+\varepsilon)\log p \log\log p}\right)$, où p est le plus petit facteur premier de n : c'est bien plus efficace que les méthodes classiques.

Aidez-vous de Maple pour factoriser n avec cette méthode.

```
> facto_p(nextprime(123456789)*nextprime(98765432));

12193264407559831 = 98765441 * 123456791 avec a = 49
```

On notera cependant qu'un concurrent est arrivé récemment et a remisé la méthode de factorisation par courbes elliptiques au deuxième rang des méthodes de factorisation les plus efficaces : il s'agit du crible algébrique dont la complexité est en $\mathcal{O}\left(e^{\left(\frac{64}{9}\log n\right)^{\frac{1}{3}}(\log\log n)^{\frac{2}{3}}}\right)$.

Cependant, la méthode « elliptique » est plus rapide pour trouver de petits facteurs.

5 Cryptographie

En fait, on reprend des méthodes introduites ces dernières années (EL GAMAL) sur $(\mathbb{F}_p^\star, \times)$ mais en travaillant sur des courbes elliptiques. La difficulté tient au fait qu'il est difficile de calculer un logarithme discret. Or les groupes définis sur des courbes elliptiques sont très divers, on ne sait même pas trouver rapidement leur ordre et encore moins leur structure : le problème du logarithme discret devient beaucoup plus compliqué et inversement les calculs du codeur se font très rapidement sur des courbes elliptiques définies sur $\mathbb{F}_{2^m}$.

Par exemple, on estime actuellement qu'une clé de 256 bits avec des courbes elliptiques assure la même sécurité qu'une clé de 3072 bits avec RSA...

Cela permet donc d'assurer une grande sécurité sur les petits processeurs des cartes à puces.

Un problème extra-mathématique existe cependant : les algorithmes liés à ces méthodes sont cachés sous d'horribles brevets qui rend le coût de leur utilisation exorbitant...

6 Jokers

Si vous éprouvez quelques difficultés...

```
> proc1 := proc(x,p)
  local i,r;
  r:=NULL;
  for i from 0 to p-1 do
    if i^2 mod p = x
    then r := r,i;
    fi;
  od;
  RETURN([r]);
  end:
```

```
> proc2 := proc(P1,P2,a,b,p)
   local x1,x2,y1,y2,x3,y3,lambda;
   if (4*a^3+27*b^2) mod p = 0  then ERROR(‘Delta nul‘);fi;
   if  (P1=[0] or appart_p(a,b,P1,p)) and (P2=[0] or appart_p(a,
      b,P2,p))   then
     if P1 = [0] then RETURN(P2); fi;
     if P2 = [0] then RETURN(P1); fi;
     x1:=P1[1] mod p;y1:=P1[2] mod p;x2:=P2[1] mod p;y2:=P2[2]
        mod p;
     if x1 = x2 mod p and y1 = -y2 mod p then RETURN([0]);fi;
     if P1 = P2 then
        if y1 = 0 mod p then RETURN([0]);fi;
           if igcd(y1,p) <  1 then RETURN([‘Diviseur‘,igcd(y1,p)
              ]);fi;
        lambda := (3*x1^2+a)/(2*y1) mod p;
     else
        if igcd(x1-x2,p) <  1 then RETURN([‘Diviseur‘,igcd(x1-x2
           ,p)]);fi;
        lambda := (y1-y2)/(x1-x2) mod p;
     fi;
     x3 := (lambda^2-x1-x2) mod p;
     y3 := (-y1 +lambda*(x1-x3)) mod p;
     RETURN([x3,y3]):
   else ERROR(‘Mauvais points‘);
   fi;
 end:
```

```
> proc3 := proc(x,a,b,p)
    local X;
    X := x;
    while sqrt_p((X^3+a*X+b) mod p,p) = [] and X < p do
      X := X+1;
    od;
    RETURN([X,sqrt_p((X^3+a*X+b) mod p,p)[1]]);
  end:
```

```
> proc4 := proc(x1,x2,a,b,p)
    local y2,E,x,i,P1,P2,P,lambda;
    if (4*a^3+27*b^2) mod p = 0 then ERROR(‘Delta nul‘);fi;
```

```
    P1 := point_courbe_p(x1,a,b,p);
    P2 := point_courbe_p(x2,a,b,p);
    E := []; P:= [P1,P2];
    lambda := (P2[2]-P1[2])/(P2[1]-P1[1]) mod p;
    for x from 0 to p-1 do
      y2 := (x^3 + a*x + b) mod p;
      for i in sqrt_p(y2,p) do
        E := [op(E),[x,i]];
        P := [op(P),[x,(lambda*(x-P1[1])-P1[2]) mod p]];
      od;
    od;
    E := plot(E,style=point,symbol=diamond,color=red);
    P := plot(P,style=point,symbol=circle,color=blue);
    plots[display]([E,P]);
   end:
```

```
> proc5 := proc(P,k,a,b,p)
  local K,M,Q,r;
  K:=k; M:=[0]; Q := P mod p;
  while K   0 do
    r := K mod 2;
    if r = 1 then M := somme_p(M,Q,a,b,p); fi;
    if member('Diviseur',M) then RETURN(M);fi;
    Q := somme_p(Q,Q,a,b,p);
    if member('Diviseur',Q) then RETURN(Q);fi;
    K := iquo(K,2);
    od;
  RETURN(M);
  end:
```

```
> proc6 := proc(n)
  local deux,trois,sept,onze,h;
  deux:=2;trois:=3;sept:=7;onze:=11;
  h:=rand(n);
  while deux <= h() do deux := deux*2;od;
  while trois <= h() do trois := trois*3;od;
  while sept <= h() do sept := sept*7;od;
  while onze <= h() do onze := onze*11;od;
  RETURN(deux*trois*sept*onze/462);
  end:
```

```
> proc7 := proc(n)
  local kP,a,b,P,B,s,k;
  if isprime(n) then ERROR('premier!');fi;
  P:=[0,1];
  s := ceil(sqrt(n)):
  B:=s+1+2*ceil(sqrt(s));
  k:=friable(B);
  a:=1;b:=1;
  kP:=k_multiple_p(P,k,a,b,n);
  while not member('Diviseur',kP) do
    a := a+1; # on peut choisir de jouer aussi sur B
    # k := friable(B);
    kP:=k_multiple_p(P,k,a,b,n);
    od;
```

```
printf('%a = %a * %a avec a = %a',n,kP[2],n/kP[2],a);
end:
```

TD N°

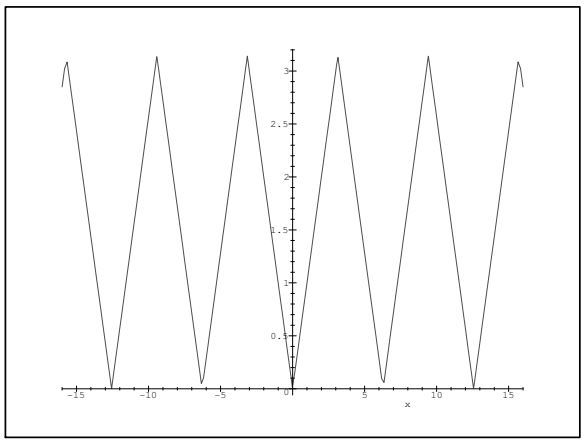
6

Séries de Fourier

Enfin un peu d'analyse...Joseph Fourier (1868-1830) le mathématicien, à ne pas confondre avec son homonyme et contemporain Charles Fourier(1872-1837), philosophe socialiste...

1 Un exemple

Soit f la fonction paire, 2π-périodique et définie sur $\mathbb{R}$ par $f(x) = x$ pour tout x appartenant à $[0,\pi]$.
La première étape consiste à observer la tête de cette fonction.
On peut utiliser la commande :
`piecewise(cond1,f_1,cond2,f_2,...,f_sinon)`
et la représenter sur une période.
Comment la représenter sur plus d'une période ? On voudrait pouvoir utiliser `plot(signal(x),x=r`
avec `sigma(x)` une expression définie à partir de f.



On calcule ensuite la valeur moyenne de la fonction et sa valeur efficace sur une période (i.e. $\sqrt{\frac{1}{2\pi}\int_{-\pi}^{\pi}\left(f(x)\right)^2\ \mathrm{d}x}$).
Calculez les coefficients du développement en série de FOURIER. On utilisera à bon escient la commande `assume(hypothèse)`.
Créez une fonction `sf := (n,x) ->` sa somme partielle de FOURIER.
Créez une animation où apparaîtra la représentation du signal et ses approximations successives par les sommes de FOURIER partielles.
L'énergie d'un signal est donnée par $\mathcal{E}(s) = \frac{1}{\mathrm{T}}\int_{[a,a+\mathrm{T}]} f^2(x)\ \mathrm{d}x$ et l'énergie de l'harmonique de rang n par $\mathcal{E}_n(s) = \frac{1}{\mathrm{T}}\int_{[a,a+\mathrm{T}]} (a_n\cos(n\omega x) + b_n\sin(n\omega x))^2(x)\ \mathrm{d}x$.
La formule de BESSEL-PARSEVAL donne donc

$$\mathcal{E}(f) = f_{\text{moy}}^2 + \sum_{n=1}^{+\infty}\mathcal{E}_n$$

Déterminer le plus petit entier n_0 telle que $f_{\text{moy}}^2 + \sum_{n=1}^{n_0}\mathcal{E}_n$ soit égal à 99,9% de l'énergie du signal.
Refaites les calculs avec la fonction impaire, de période 1, et telle que $f(t) = 1$ pour tout $t \in \left]0;\frac{1}{2}\right[$ et $f(0) = f(1/2) = 0$ et visualisez un résultat bien connu.

2 À Centrale

Exercice 6 - 1 **Centrale 2010**

On pose g : $\begin{array}{ccc} \mathbb{R} & \to & \mathbb{R} \\ x & \mapsto & \begin{cases} \frac{\text{ch}(x)}{\text{sh}(x)} - \frac{1}{x} \text{ si } x \neq 0 \\ 0 \text{ sinon} \end{cases} \end{array}$

1. La fonction g est-elle continue ? Soit $\alpha \in \mathbb{R}^*$. Soit f une fonction définie sur $\mathbb{R}$, 2π-périodique, telle que $f(x) = \text{ch}(\alpha x)$ pour $x \in [-\pi, \pi]$.
2. Représenter f sur $[-3\pi, 3\pi]$ pour $\alpha = 2$.
3. Calculer la série de Fourier de f et étudier sa convergence.
4. En déduire une expression de g comme somme de série de fonctions rationnelles.

Exercice 6 - 2 Centrale 2010

Soit, pour $f \in \mathcal{C}^0([0,\pi],\mathbb{R})$ et $n \in \mathbb{N}$, $\quad \text{I}_n(f) = \displaystyle\int_0^\pi f(t)|\sin nt|\,\text{d}t$

1. Soit $f : t \mapsto 1$. Calculer $\text{I}_n(f)$ pour $n \in [\![1, 10]\!]$. Prouver le résultat dans le cas général.
2. Soit $f : t \mapsto t$. Calculer $\text{I}_n(f)$ pour $n \in [\![1, 10]\!]$. Prouver le résultat dans le cas général.
3. Soit $f : t \mapsto t^2$. Calculer $\text{I}_n(f)$ pour $n \in [\![1, 10]\!]$. Que peut-on conjecturer ?
4. Soit $\varphi : t \mapsto |\sin t|$. Montrer que φ est somme de sa série de Fourier.
   En déduire, si $f \in \mathcal{C}^0([0,\pi],\mathbb{R})$, que $\left(\text{I}_n(f)\right)_{n\geq 0}$ tend vers une limite que l'on précisera.

TD N°

7

Transformée de LAPLACE

Pierre-Simon de LAPLACE (1749-1827) a été très éclectique dans ses explorations scientifiques (qui furent majeures) et politiques (qui furent souvent médiocres). Fils d'un ouvrier agricole, il fut ministre de l'intérieur sous le Consulat, comte de l'Empire, marquis après la Restauration. Son œuvre mathématique est très riche mais c'est paradoxalement une découverte d'EULER dévéloppée un siècle plus tard par HEAVISIDE qui a été nommée en son honneur et que nous allons étudier aujourd'hui

1 Apéritif : convergence dominée d'une suite de fonctions

On considère une suite de fonctions (f_n) définie sur un intervalle I.
Calculez $\lim\limits_{n\to+\infty} f_n(x)$, le maximum de $f_n(x)$ sur I. Dites si la convergence est uniforme et vérifiez si l'égalité suivante a lieu :

$$\lim_{n\to+\infty}\int_a^b f_n(x)\,\mathrm{d}x = \int_a^b \lim_{n\to+\infty} f_n(x)\,\mathrm{d}x$$

Ah, j'oubliai : tout ceci avec Maple, évidemment...
Vous commencerez par `f:=(n,x)->...`. Vous utiliserez `limit`, `int`, `solve`. Vous finirez par une animation des représentations graphiques des f_n pour différentes valeurs de n avec `plots[display]([P],insequence=true)`.
Testez avec $f_n(x)=\frac{nx}{(1+n^2x^2)^2}$ et $g_n(x)=\frac{n^2x}{(1+n^2x^2)^2}$ sur $[0,1]$ par exemple.

2 Entrée : une drôle de fonction intégrable au sens de Riemann

Bernhard RIEMANN (1826-1866)

Cet exemple, cité dans l'excellent « *L'analyse au fil de l'histoire* » de HAIRER et WANNER, a été proposé par RIEMANN en 1854 pour démontrer la portée de sa théorie de l'intégration. Soit

$$f(x)=\sum_{n=1}^{+\infty}\frac{\mathrm{B}(nx)}{n^2} \quad \text{où} \quad \mathrm{B}(x)\begin{cases} x-\langle x\rangle & \text{si } 2x\notin\mathbb{Z} \\ 0 & \text{sinon}\end{cases}$$

avec $\langle x\rangle$ l'entier le plus proche de x (`round` sous Maple).
Que pensez-vous de la continuité de cette fonction sur $[0,1]$? La série converge-t-elle uniformément ? La fonction f est-elle intégrable sur $[0,1]$?
Créez une procédure `B:=x->...` qui calcule $\mathrm{B}(x)$.
Vous utiliserez `type` et une conditionnelle `if...then...else...`.
Créez ensuite une procédure `fn:=(n,x)->...` qui calcule $\frac{\mathrm{B}(nx)}{n^2}$.
Tracez la représentation graphique de plusieurs f_n.
Créez `F:=(n,x)->sum(fn(k,x),k=1..n)` et tracez la représentation graphique de `F(100,x)` par exemple.

3 Plat de résistance : transformée de Laplace

3.1 Rôle

La transformée de LAPLACE permet de convertir une équation différentielle dont on connaît les conditions initiales en une équation algébrique : on résout l'équation algébrique puis on effectue la transformation inverse pour obtenir la solution de l'équation différentielle.

3.2 Définitions et premières propriétés

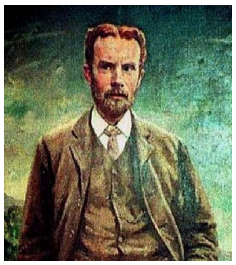
Si f est une fonction définie sur $\mathbb{R}$ en étant identiquement nulle sur $]-\infty;0[$, on appelle transformée de Laplace de f la fonction F définie, sous certaines conditions, par :

$$\mathcal{L}\big(f(x)\big)=\mathrm{F}(s)=\int_0^{+\infty}\mathrm{e}^{-sx}f(x)\,\mathrm{d}x$$

La fonction $\mathcal{L}$ qui à f associe sa transformée F est la **transformation de LAPLACE**.
Pour assurer l'existence de la transformée, voici deux conditions suffisantes :

— f est continue par morceaux sur tout compact $[0,\mathrm{A}]$;
— f est d'ordre exponentiel à l'infini : il existe $\mathrm{M}>0$ et $a\in\mathbb{R}$ tels que $\big|f(x)\big|<\mathrm{M}\mathrm{e}^{ax}$ pour tout x assez grand.

Si ces conditions sont vérifiées, pour quelles valeurs de s la transformée est-elle définie ?
On obtient facilement les transformées de Laplace des fonctions polynomiales, trigonométriques, exponentielles, etc. qu'on regroupe dans une table.
Créez une fonction `lap := (f,s) ->...`.
Déterminez les transformées de LAPLACE des fonctions définies par les expressions suivantes et vérifiez avec Maple en utilisant `assuming` à bon escient :
$f_1(x)=1$, $f_2(x)=x$, $f_3(x)=x^n$, $f_4(x)=\sqrt{x}$, $f_5(x)=\frac{1}{\sqrt{x}}$, $f_6(x)=\mathrm{e}^{-ax}$, $f_7(x)=\cos(\omega x)$, $f_8(x)=\sin(\omega x)$, $f_9(x)=\mathrm{sh}(ax)$, $f_{10}(x)=\mathrm{ch}(ax)$.
Vous donnerez vos résultats sous forme d'un tableau.



Oliver HEAVISIDE (1850-1925)

On a l'habitude de nommer la fonction $\mathcal{U}:x\mapsto\begin{cases}1 \text{ si } x\geqslant 0\\ 0 \text{ sinon}\end{cases}$ **échelon de HEAVISIDE** du nom du scientifique anglais qui créa le calcul opérationnel que nous allons bientôt introduire.
La transformation de LAPLACE est clairement linéaire.
Exprimez $\mathcal{L}(f(ax))$ et $\mathcal{L}(f(x-a))$ en fonction de $\mathcal{L}(f(x))$.
Que peut-on dire de la transformée d'une fonction périodique ?
On suppose que f' vérifie les mêmes conditions que f. Démontrez que

$$\mathcal{L}(f'(x))=s\mathrm{F}(s)-f(0^+)$$

puis que

$$\mathcal{L}(f''(x))=s^2\mathrm{F}(s)-sf(0^+)-f'(0^+)$$

si f'' est suffisamment régulière.
En fait, aux valeurs initiales près (!), la transformée de LAPLACE remplace le produit formel de $f(x)$ par $\frac{\mathrm{d}}{\mathrm{d}x}$ par le produit algébrique de $\mathrm{F}(s)$ par s.
Que vaut $\lim\limits_{s\to+\infty}\int_0^{+\infty}\mathrm{e}^{-sx}f'(x)\,\mathrm{d}x$?
Déduisez-en $\lim\limits_{s\to+\infty}s\mathrm{F}(s)$ (C'est le théorème de la valeur initiale).
Montrez de même que

$$\mathcal{L}\left(\int_0^x f(t)\,\mathrm{d}t\right)=\frac{\mathrm{F}(s)}{s}$$

3 3 Transformation inverse

Mathias LERCH (1860-1922)

On sent, d'après les résultats précédents, quelles pourraient être les applications de cette transformation.
Pour cela, il faudrait pouvoir passer de la transformée à **l'original**. Si $\mathrm{F}(s)=\mathcal{L}\big(f(x)\big)$ alors f est une transformée inverse de F : $f(x)=\mathcal{L}^{-1}(\mathrm{F}(s))$.
On parle alors d'**original** de $\mathrm{F}(s)$.
On fera attention au fait que deux fonctions distinctes peuvent avoir la même transformée.
Toutefois, un théorème dû à LERCH (1903) permet d'avoir une idée du problème :
Si la fonction résultat $\mathrm{F}(s)$ *est identiquement nulle, ou même si l'on sait seulement que* $\mathrm{F}(s_0+mh)=0$ $(m=1,2,3,\ldots)$ *où* s_0 *est un point quelconque du demi-plan de convergence et* h *un nombre réel quelconque, la fonction* $f(x)$ *est nulle presque partout.*
La démonstration est fondée sur le théorème d'approximation de WEIERSTRASS.
On retiendra donc que l'original est unique sur tout sous-ensemble où il est continu.
La transformation inverse est linéaire comme réciproque d'une application linéaire.
Nous abuserons donc des décompositions en éléments simples.
Déterminez les originaux de $\mathrm{F}(as)$ et $\mathrm{F}(s+a)$, de $\mathrm{F}'(s)$ et de $\int_s^{+\infty}\mathrm{F}(u)\,\mathrm{d}u$.
Montrez que $\mathcal{L}^{-1}(\mathrm{F}(s)\times\mathrm{G}(s))=\int_0^x f(t)g(x-t)\,\mathrm{d}t$ (produit de convolution de f par g).

3 4 Application à la résolution d'équations différentielles linéaires

On veut résoudre l'équation différentielle $y''-3y'-4y=\sin(x)$ en utilisant les transformées de Laplace avec les conditions initiales $y(0)=1$ et $y'(0)=-1$.
Alors :

$$\mathcal{L}\left(y''-3y'-4y\right)=\mathcal{L}\left(\sin(x)\right)$$

c'est-à-dire :

$$\left(s^2-3s-4\right)\mathcal{L}(y)-sy(0)-y'(0)+3y(0)=\mathcal{L}(\sin(x))$$

Déterminez $\mathcal{L}(y)$. Vérifiez avec Maple.
Utilisez `convert(L(y),parfrac,s)` pour vérifier votre décomposition en éléments simples.
Vous pourrez vérifier votre résultat avec `inttrans[invlaplace](%,s,x)`.
Appliquez cette méthode pour résoudre le système différentiel :

$$\begin{cases} y_1'=4y_1+4y_2 \\ y_2'=y_1+4y_2 \end{cases} \quad \text{avec} \quad \begin{cases} y_1(0)=0 \\ y_2(0)=1 \end{cases}$$

4 Fromage : exercices divers

Exercice 7 - 1

Soit la fonction $\mathrm{J}(x)=\sum_{n=0}^{+\infty}(-1)^n\frac{1}{(n!)^2}\left(\frac{x}{2}\right)^{2n}$.
Vérifiez que sa transformée est $\frac{1}{\sqrt{s^2+1}}$.
Résolvez l'équation différentielle $xy''+y'+xy=0$ telle qe $y(0)=1$ et $y'(0)=0$ en utilisant les transformées de LAPLACE.

Exercice 7 - 2

Développez $\frac{\sin x}{x}$ en série entière. En déduire sa transformée de LAPLACE.
Retrouvez ce résultat à l'aide de l'original de $\int_s^{+\infty}\mathrm{F}(u)\,\mathrm{d}u$.
Résolvez ensuite $xy''+2y'+xy=0$, $y(0)=1$, $y'(0)=0$.

5 Dessert : directement avec Maple (pour vérifier..)

`dsolve({equa,cond},fonc(var),method=laplace)` permet de résoudre directement l'équation différentielle.
Pour utiliser des outils plus détaillés, on ouvre la bibliothèque `inttrans`.

```
> with(inttrans);
> equadiff := diff(x(t),t$2)-4*diff(x(t),t)+3*x(t)=exp(3*t);
> L:=laplace(equadiff,t,s);
> L:=subs({x(0)=4,D(x)(0)=8},L);
> L:=subs(laplace(x(t),t,s)=K(s),L);
> K(s):=solve(L,K(s));
> sol := invlaplace(K(s),s,t);
```

6 Digestif : une démonstration du théorème de Lerch dans un cas particulier

Soit f continue et bornée sur $[0,+\infty[$.
On note $\mathrm{F}(s)=\displaystyle\int_0^{+\infty} f(t)\mathrm{e}^{-st}\,\mathrm{d}t$ et $\mathrm{G}(s)=\displaystyle\int_0^{+\infty} tf(t)\mathrm{e}^{-st}\,\mathrm{d}t$.

1. Montrez que F est de classe $\mathcal{C}^\infty$ sur $]0,+\infty$.

2. Soit $h : t\mapsto \mathrm{e}^t\mathrm{e}^{-\mathrm{e}^t}$.

a. Montrez que $h(t)=\displaystyle\sum_{k=0}^{+\infty}\frac{(-1)^k}{k!}\mathrm{e}^{(k+1)t}$.

b. Montrez que h est intégrable sur $\mathbb{R}$ et que $\displaystyle\int_{\mathbb{R}} h=1$.

c. Pour $n\in\mathbb{N}^*$, on définit $h_n : t\mapsto nh_n(t)$. Montrez que

$$\forall a>0,\quad \lim_{n\to+\infty}\int_{-a}^{a} h_n(t)\,\mathrm{d}t=1$$

d. En déduire que pour f continue et bornée sur $\mathbb{R}_+$ et pour tout $x>0$:

$$\lim_{n\to+\infty}\int_0^{+\infty} h_n(x-t)f(t)\,\mathrm{d}t=f(x)$$

3. Soit F la transformée de LAPLACE de f continue et bornée sur $\mathbb{R}_+$.
Montrez que pour tout $x>0$ et tout $n\in\mathbb{N}^*$:

$$n\sum_{k=0}^{+\infty}\frac{(-1)^k}{k!}\mathrm{e}^{n(k+1)x}\mathrm{F}(n(k+1))=\int_0^{+\infty} h_n(x-t)f(t)\,\mathrm{d}t$$

4. Que pouvez-vous en déduire s'il existe A > 0 tel que F soit identiquement nulle sur $[\mathrm{A},+\infty[$?