Guide concis d'Erable
Logiciel de calcul formel pour HP48

Bernard Parisse
Institut Fourier (CNRS UMR 5582)
Université de Grenoble I
F-38402 St Martin d'Hères Cédex
Bernard.Parisse@ujf-grenoble.fr

15 Juillet 1998

Qu'est-ce qu'`Erable`? C'est un logiciel capable d'effectuer des calculs exacts par opposition au calcul numérique où les résultats sont approximatifs. Les logiciels les plus connus qui en sont capables sont Maple, Mupad (disponible gratuitement: cf. [11]), Mathematica, Axiom, Reduce, Derive, ... qui tournent sur ordinateur personnel ou station de travail. Une version affaiblie de Derive tourne d'ailleurs sur les calculatrices TI92 et 89.

La HP48 dispose de quelques instructions pour faire du calcul formel: par exemple `ISOL`, `QUAD` (pour résoudre des équations du 1er et 2ème degré), `TAYLR` (développement de Taylor), ∂ (dérivée), $\int$ (quelques intégrales sont reconnues), `COLCT` et `EXPAND` (pour "arranger" des expressions). Mais elles sont assez inefficaces. On trouve sur le réseau Internet des programmes qui permettent d'améliorer les capacités de la HP48: les plus utilisés sont `ALG48` ([9]) et `Erable`. Le premier est le plus rapide alors que le second est le plus complet. L'utilisation conjointe des deux logiciels transforme votre HP48 en la calculatrice formelle la plus puissante qui existe.

Attention, l'utilisation d'un logiciel de calcul formel ne vous dispensera pas de réfléchir! Dans de nombreuses situations, il faut avoir une idée de la façon dont le programme fonctionne pour bien présenter le problème. Plus on sait de mathématiques, mieux on se sert d'un logiciel de calcul formel. D'autre part, le calcul formel ne permet en général pas la résolution des problèmes qu'on rencontre dans la vie courante, **il ne jamais négliger les solutions numériques**.

La distribution du logiciel `Erable` comprend le logiciel de calcul formel proprement dit et une version modifiée du logiciel `EQSTK` qui propose un environnement de travail mieux adapté. La documentation que vous lisez s'organise de la manière suivante:

- Le chapitre 1 explique comment installer `Erable` sur votre calculatrice.
- Le chapitre 2 explique comment utiliser sa HP48 pour faire efficacement du calcul formel, il ne nécessite aucune connaissance préalable de la HP48. Il se termine par une petite feuille d'exercices qui vous permettra de vous familiariser avec `Erable`. Ce chapitre a été rédigé en collaboration avec Renée de Graeve et testé par des étudiants de Deug de l'Université de Grenoble I (à l'occasion du module calculatrices 1997/98).
- Le chapitre 3 reprend les principales commandes d'`Erable` par domaine mathématique
- Enfin, on trouvera en appendice A.2 et A.3 la table des commandes d'`Erable`, ainsi que les tables de référence de l'interface (touches redéfinies)

Chapitre 1

Avant de commencer

1.1 License d'utilisation

Le logiciel `Erable` ([19]) et sa documentation sont © Bernard Parisse pour la plus grande partie, Claude-Nicolas Fiechter et Mika Heiskanen pour certaines routines du noyau et pour `EQSTK`, André Schoorl pour la librairie `UFL` et Jean-Yves Avenard pour l'éditeur `Miniwriter`. Vous devez lire et accepter la licence d'utilisation d'`Erable` (fichiers `copying.doc` et `license.txt`) avant d'utiliser `Erable`. La licence d'utilisation d'`Erable` s'applique également à la documentation que vous lisez, ce qui signifie essentiellement que vous pouvez reproduire le logiciel et sa documentation par quelque moyen que ce soit tant que vous n'en faites pas d'exploitation commerciale. La doucmentation est fournie en ligne au format PDF que vous pouvez lire avec le logiciel `Acrobat Reader` de la société `Adobe`, ce logiciel est disponible gratuitement ([1])

1.2 Installation

Pour installer `Erable`, vous devez disposer d'un cable de liaison série avec un ordinateur. Pour utiliser l'installation automatique, vous devez également disposer du logiciel `kermit` ([13]) sur votre ordinateur. Ce logiciel est disponible gratuitement sur Internet (`http://www.kermit.columbia.edu`) pour de nombreuses plateformes, par exemple PC Dos, PC Windows 3.1, Mac, PC Linux, ...

Allumez votre ordinateur et connectez-y physiquement la HP48

1. Décompressez le fichier `erable.zip`

2. Lancez le programme `kermit` sur votre ordinateur. Si vous utilisez Windows 95, vous devrez probablement revenir en mode Dos auparavant.

3. Indiquez à `kermit` la ligne série à laquelle est connectée la HP48: par exemple si elle est connectée au port `COM2` sous MS-DOS, tapez:
   `set port 2`
   sous Linux, tapez:
   `set line /dev/cua1`

4. Tapez ensuite
   `set file type binary`
   `set speed 9600`
   `serv`

5. Prenez votre HP48. Vous devez avoir 120K de libre pour que l'installation réussisse. Si vous n'avez pas de données importantes, le mieux est de taper simultanément sur les touches $\boxed{\text{ON}}$, $\boxed{\text{A}}$ et $\boxed{\text{F}}$ puis sur la touche $\boxed{\text{F}}$ pour effacer la mémoire.

6. Tapez sur la touche shift gauche puis sur la touche $\boxed{1}$, tapez ensuite sur la touche blanche $\boxed{\text{B}}$ pour sélectionner `IOPAR` puis sur la touche blanche $\boxed{\text{A}}$ jusqu'au moment où vous verrez en première ligne d'affichage:
   `IR/wire: wire`

7. Enfoncez la touche $\boxed{\alpha}$ et tapez les touches $\boxed{\text{S}}$ $\boxed{\text{E}}$ $\boxed{\text{T}}$ $\boxed{\text{U}}$ $\boxed{\text{P}}$ $\boxed{\text{SPC}}$ $\boxed{\text{K}}$ $\boxed{\text{G}}$ $\boxed{\text{E}}$ $\boxed{\text{T}}$ puis relâchez la touche $\boxed{\alpha}$ et tapez la touche $\boxed{\text{ENTER}}$. Ceci doit charger le programme `SETUP` sur votre HP48. Si cela ne fonctionne pas, vérifiez la configuration de `kermit` et recommencez.

8. Pressez la touche $\boxed{\text{VAR}}$, puis la touche blanche $\boxed{\text{A}}$ ce qui lance le programme `SETUP`. Vous devrez alors patienter dix bonnes minutes puis la HP48 va “rebooter”.

9. Après le “reboot”, tapez la touche $\boxed{\text{VAR}}$, puis la touche $\boxed{\text{E}}$ (pour lancer à nouveau le programme `SETUP`). Cela lance la deuxième phase de l'installation qui se termine après environ une minute à nouveau par un “reboot”.

10. Pressez la touche $\boxed{\text{VAR}}$ puis la touche blanche $\boxed{\text{A}}$ (pour `INIT`). Cela lance l'environnement modifié `EQSTK`. L'installation est terminée.

Chapitre 2

Introduction à la HP48.

Le but de ce chapitre est de permettre à un utilisateur novice de prendre en main rapidement sa HP48 à l'aide de l'environnement proposé avec `Erable`. Une expérience menée à l'Université de Grenoble I montre qu'il faut environ une heure pour se débrouiller dans cet environnement. Ce chapitre sera également utile aux autres utilisateurs pour utiliser, voire adapter, l'interface proposée.

2.1 Allumer et éteindre la HP48.

Appuyer sur la touche `ON`. En cours de travail, cette touche `ON` permet de sortir d'une application: elle joue le rôle de EXIT ou de CANCEL.

Pour éteindre, appuyer sur la touche shift droit puis sur `ON`.

2.1.1 Que voit-on?

De haut en bas:

1. l'état de la calculatrice: `REAL VX='X' ...`
2. des lignes numérotées de 1 à 2, 3, 4 ou 5: c'est la pile
3. un bandeau contenant des commandes,
4. le clavier

1. L'état de la calculatrice décrit les modes mis en oeuvre:
    - mode `REAL` ou `COMPLEX`: selon la nature des calculs que l'on veut faire

- le nom de la variable courante `VX`: en général c'est `X`
- `SYMBOLIC` ou `NUMERIC`: selon que l'on veut faire du calcul symbolique (exact) ou numérique (approché)
- `POLY` ou `INT`: selon que l'on fait des calculs sur les polynômes ou sur les entiers
- `USER` ou `NORM`: selon que l'on veut utiliser des raccourcis claviers ou pas (en mode `USER` certaines touches ont été redéfinies pour améliorer la mise en oeuvre du calcul symbolique, c'est pourquoi il est préférable de travailler en mode `USER`)

Pour changer l'état de la calculatrice, on se reportera à la section 2.6 (configuration de la machine), p. 14.

2. La pile:
Principe: sur la pile, l'utilisateur met ses données et les résultats des différents opérations numériques ou symboliques s'y empilent au fur et à mesure: ceci permet de réutiliser *facilement* les résultats obtenus.
Exemple: si l'utilisateur veut afficher l'expression $X^2 + 1$ au niveau 1 de la pile, il peut taper:

   - ['] [DEL] [y^x] [2] [+] [1] [ENTER]
   (['] débute une expression, [DEL] est une touche redéfinie pour taper `X` sans être en mode alphabétique)
   - [DEL] [2] [y^x] [1] [+]
   on remarquera alors que les opérateurs y^x et + sont postfixés.

3. Le bandeau:
Les commandes du bandeau sont accessibles par les touches blanches [A] [B] [C] [D] [E] [F].

   Lorsque le bandeau comporte plus de 6 commandes, la suite du bandeau est visible lorsqu'on appuie sur la touche [NXT]. Le bandeau peut contenir des ensembles de commandes, ils sont repérables par leur forme de valise. Pour activer une commande du bandeau, il suffit de taper sur la touche blanche correspondante.

   On peut changer de bandeau en utilisant le menu général (touche [PRG]) et en validant la rubrique désirée ou en utilisant un raccourci clavier (lorsqu'il en existe un).

   Exemple: si on veut utiliser la fonction $\cosh(x)$ (cosinus hyperbolique), on peut taper:

- touche $\boxed{\texttt{PRG}}$, puis $\boxed{\texttt{7}}$ (pour `NUMERIC`) puis $\boxed{\texttt{ENTER}}$ puis $\boxed{\texttt{1}}$ (pour `MATHS FNCS`) puis $\boxed{\texttt{ENTER}}$ puis $\boxed{\texttt{D}}$ pour ouvrir la valise `HYP` puis $\boxed{\texttt{C}}$ pour `COSH`.
- si on connait le raccourci: $\boxed{\alpha}$, shift droit, $\boxed{\texttt{MTH}}$ puis $\boxed{\texttt{D}}$ pour la valise `HYP` puis $\boxed{\texttt{C}}$ pour `COSH`.
- directement le nom alphabétique de la commande:
  $\boxed{\alpha}$ $\boxed{\texttt{C}}$ $\boxed{\texttt{O}}$ $\boxed{\texttt{S}}$ $\boxed{\texttt{H}}$

4. Le clavier:
   Il faut repérer:

   - la touche $\boxed{\texttt{ON}}$ pour la mise en route ou pour arrêter un calcul en cours. Pour éteindre, taper la touche shift droit puis $\boxed{\texttt{ON}}$.
   - la touche $\boxed{\texttt{ENTER}}$ qui sert à valider une commande.
   - la touche $\boxed{\alpha}$ pour taper du texte en majuscules. Pour sortir du mode saisie alphabétique, taper à nouveau sur $\boxed{\alpha}$. Pour basculer entre l'écriture en majuscules ou en minuscules, taper sur shift gauche puis $\boxed{\alpha}$.
   - Les deux touches shift.
   - Les quatre flèches (gauche, haut, bas, droite) qui permettent de déplacer le curseur lorsqu'on est dans un éditeur ou dans un menu ou dans l'éditeur de la pile
   - La touche d'effacement $\boxed{\leftarrow}$
   - Les 6 touches blanches $\boxed{\text{A}}$, ..., $\boxed{\text{F}}$ qui permettent de faire exécuter les commandes qui sont dans le bandeau et la touche $\boxed{\texttt{NXT}}$ pour voir la suite du bandeau.
   - $\boxed{\alpha}$ $\boxed{\texttt{DEL}}$ ou shift gauche $\boxed{\texttt{DEL}}$ pour effacer la pile
   - $\boxed{\texttt{VAR}}$ donne accès au bandeau des noms de variables stockées dans la mémoire de la HP48.

   Les principales touches qui ont été redéfinies sont:

   - $\boxed{\texttt{DEL}}$ qui renvoie la variable `X`,
   - shift droit $\boxed{\texttt{DEL}}$ qui renvoie $-\infty$,
   - $\boxed{\alpha}$ shift droit $\boxed{\texttt{DEL}}$ qui renvoie $+\infty$ ou ∞ selon les circonstances
   - $\boxed{\texttt{CST}}$ renvoie le nombre imaginaire i (on peut aussi taper $\boxed{\alpha}$ shift gauche $\boxed{\texttt{CST}}$ comme pour n'importe quelle lettre minuscule)

- $\boxed{\texttt{PRG}}$ donne accès au menu général
- $\boxed{\texttt{MTH}}$ donne accès au menu du logiciel de calcul formel `Erable`
- $\boxed{\alpha}$ shift droit $\boxed{\texttt{MTH}}$ donne accès au bandeau regroupant les fonctions numériques de la HP48
- $\boxed{\alpha}$ shift droit $\boxed{1}$ donne accès au bandeau des commandes les plus utilisées d'`Erable`
- $\boxed{\alpha}$ shift droit $\boxed{\texttt{CST}}$ permet d'accéder directement à la configuration de la HP48

Remarques:

- Une même $\boxed{\text{touche}}$ peut avoir jusqu'à 6 fonctions différentes: $\boxed{\text{la touche}}$, shift gauche $\boxed{\text{la touche}}$, shift droit $\boxed{\text{la touche}}$, $\boxed{\alpha}$ $\boxed{\text{la touche}}$, $\boxed{\alpha}$ shift gauche $\boxed{\text{la touche}}$, $\boxed{\alpha}$ shift droit $\boxed{\text{la touche}}$
- ∞ et $-+\infty$ sont interprétés comme des infinis non signés ($|x| \to +\infty$).
  Pour obtenir par exemple $X = +\infty$ sur la pile, on peut taper
  $\boxed{\texttt{DEL}}$ $\boxed{\alpha}$ shift droit $\boxed{\texttt{DEL}}$ shift gauche $\boxed{0}$ (pour `X` $+\infty$ `=`)
  et pour obtenir $X = -\infty$, on tapera
  $\boxed{\texttt{DEL}}$ shift droit $\boxed{\texttt{DEL}}$ shift gauche $\boxed{0}$ (pour `X` $-\infty$ `=`)

5. La ligne de commande:
   Elle apparaît chaque fois que l'on tape sur le clavier: la pile remonte et libère une ligne sur laquelle s'inscrit ce que l'on tape. Tant qu'on n'a pas appuyé sur une commande ou sur $\boxed{\texttt{ENTER}}$, cette ligne peut être modifiée. Lorsqu'on appuie sur $\boxed{\texttt{ENTER}}$ ou sur une commande, les éléments de la ligne de commande ou le résultat de la commande sont rangés sur la pile. Il y a différents modes de saisie:

   - le mode immédiat: ce que l'on tape agit directement sur la pile
   - le mode algébrique: l'appui sur $\boxed{\texttt{'}}$ au début de la ligne de commande indique que l'on entre une expression algébrique: cette expression sera rentrée avec l'écriture mathématique habituelle. Pour des expressions algébriques compliquées, on aura plutôt intérêt à utiliser l'éditeur d'équations (shift gauche $\boxed{\texttt{ENTER}}$ pour `EQUATION`), dans ce cas $\boxed{\texttt{'}}$ devient inutile.

   Exemple: on veut entrer l'expression $(X+1)(X-1)$:

   - 1ère méthode: on tape:
     $\boxed{\texttt{'}}$ shift gauche $\boxed{\div}$ $\boxed{\texttt{DEL}}$ $\boxed{\texttt{+}}$ $\boxed{\texttt{1}}$ $\boxed{\triangleright}$ $\boxed{\times}$ shift gauche $\boxed{\div}$ $\boxed{\texttt{DEL}}$ $\boxed{\texttt{-}}$ $\boxed{\texttt{1}}$ $\boxed{\texttt{ENTER}}$

- 2ème méthode (avec l'éditeur d'équation):
  `PRG` `1` (`EDITORS`) `ENTER` `4` (`NEW EQUATION`) `ENTER`
  ou shift gauche `ENTER`
  On tape la même séquence que celle de la première méthode mais sans le quote `'` initial.

  L'éditeur d'équation sert surtout à rentrer des expressions plus compliquées (intégrales, fractions, ...), par exemple pour $\int_1^2 \frac{dx}{x}$, on lance l'éditeur d'équation puis on tape:
  shift droit `COS` `1` `▷` `2` `▷` `1` `÷` `DEL` `▷` `▷` `DEL` `ENTER`

  On remarquera que `▷` permet de déplacer le curseur jusqu'au bon endroit.

- 3ème méthode: on entre $X+1$ en tapant `'` `DEL` `+` `1` `ENTER` puis on entre $X-1$ en tapant `'` `DEL` `-` `1` `ENTER`
  $X+1$ se trouve au niveau 2 de la pile et $X-1$ au niveau 1 de la pile, on appuie sur la touche `×` (multiplication) pour obtenir $(X+1)(X-1)$ au niveau 1

2.2 Les différents opérateurs arithmétiques

Avant de traiter cette section, tapez la commande `VER` ou choisissez la commande `RESET ERABLE` dans l'option `4.SETUP` du menu principal `PRG`.

Pour faire du calcul numérique, les opérations $+$, $-$, $\times$, $\div$, $\sqrt{}$... sont réalisées:

1. soit en tapant les arguments puis la touche correspondante, par exemple, pour $5+3$:
   `5 3 +`

2. soit en tapant la touche `'`, l'opération sous forme algébrique, la touche `ENTER` puis la touche `EVAL` (pour les expressions contenant les constantes π, e ou i ou des intégrales définies, il faut taper shift gauche `EVAL` au lieu d'`EVAL`), par exemple pour $5+3$:
   `' 5 + 3 ENTER EVAL`

Pour faire du calcul symbolique, ces mêmes opérations peuvent aussi être réalisées de deux manières:

1. soit en tapant les arguments suivi par α, shift droit et la touche de l'opération, par exemple pour $\frac{2}{4}$:
   `2 4` α shift droit $\div$
   ce qui renvoie $\frac{1}{2}$ sur la pile.

2. soit en tapant ', l'opération sous forme algébrique, la touche ENTER puis la α, shift droit, SPC (qui exécute la commande EXPA), par exemple pour $\frac{2}{4}$:
   '2÷4 ENTER
   qui renvoie $\frac{2}{4}$ sur la pile, puis:
   α shift droit SPC
   qui simplifie $\frac{2}{4}$ en $\frac{1}{2}$

La différence entre calcul numérique et symbolique est que les résultats sont exacts dans le second cas. En contrepartie, les calculs symboliques sont souvent plus longs, et certaines opérations ne peuvent être effectuées que numériquement. Exemples:

- $\frac{1}{3} \times 3$.
  En mode numérique, on tape:
  1 3÷ 3 ×
  et on obtient 0.9999999999
  En mode symbolique, on tape:
  1 3 α shift droit ÷ 3 α shift droit ×
  et on obtient 1.
- $\int_1^2 e^{x^2}\, dx$:
  Saisissez l'intégrale avec EQUATIONWRITER. En mode numérique, il suffit de taper shift droit EVAL pour en obtenir une approximation numérique. En mode symbolique, on n'obtient rien car e^{x^2} n'admet pas de primitive exprimée en termes des fonctions usuelles.

Les opérateurs symboliques reconnaissent les nombres entiers, les fractions, les réels, les infinis ($+\infty$, $-\infty$ et ∞ non signé), les complexes (écrit sous forme symbolique comme $1 + 2i$ ou sous forme numérique comme (1,2)), les polynômes et fractions rationnelles, les expressions algébriques ainsi que les vecteurs et les matrices formés à partir de ces éléments.

En résumé pour effectuer une opération symbolique, on a donc deux possibilités le mode de saisie algébrique (' ou EQUATIONWRITER) où l'on utilise les touches usuelles suivi par la commande de simplification α shift droit SPC (raccourci clavier d'EXPA), ou le mode de saisie directe, où l'on précède les touches de fonction par α et shift droit.

2.3 Comment accéder aux commandes

Principe général: les commandes peuvent être tapées en toutes lettres (en mode α), par la touche menu blanche correspondant à la commande du bandeau ou, pour certaines commandes, par un raccourci clavier.

Les commandes agissent sur les objets qui se trouvent sur la pile: certaines commandes ont besoin de plusieurs arguments de différentes natures.

Exemple:
La commande `TAYLR` a besoin de 3 arguments, à placer sur la pile dans cet ordre:

- l'expression à développer (au niveau 3), par exemple `'SIN(X)'`,
- la variable (au niveau 2), par exemple `X`,
- l'ordre au niveau 1, par exemple 3

On tape alors les touches α `T A Y L R` ou on fait apparaître `TAYLR` dans le bandeau (`MTH` puis `1.BASE ALGEBRA`) et on tape la touche menu `F`. Les 3 arguments sont enlevés de la pile et remplacés ici par 1 argument qui est le développement de Taylor de $\sin(X)$ en $X = 0$ à l'ordre 3: $X - \frac{X^3}{3!}$.

Les commandes de calcul symbolique (du logiciel `Erable`) sont accessibles dans les différents bandeaux obtenus grâce au menu `MTH` suivi du numéro de menu puis de `ENTER` ou par le raccourci clavier α, shift droit, numéro de menu (par exemple `MTH`, 1 `ENTER` ou α, shift droit, 1 pour le bandeau `1.BASE ALGEBRA`). On peut aussi taper le nom de commande, les commandes sont listées alphabétiquement dans le résumé des commandes en appendice A.3.2)

Pour les autres commandes, il faut se reporter au menu principal (touche `PRG`) ou aux raccourcis claviers repérables par les annotations bleues du clavier de la HP48 (par exemple shift droit-`8` lance le menu `PLOT` de tracé de graphes de la HP48).

2.4 Les avantages de la pile

2.4.1 Un exemple

On veut tracer sur un même graphique $\sin(x)$ et son développement de Taylor au voisinage de $x = 0$ aux ordres 3,5 et 7.

1. On fait apparaitre `TAYLR` dans le bandeau en tapant `MTH` 1 `ENTER` ou α shift droit 1.
2. On entre $\sin(X)$ en tapant les touches `' SIN DEL ENTER`, puis X en tapant la touche `DEL` puis 3 puis on appuie sur la touche blanche `F` correspondant à `TAYLR`.

3. Reprendre l'étape 2 pour 5 et 7[1]. On obtient sur la pile les développements P_3, P_5 et P_7.

4. Taper à nouveau $\sin(X)$

5. Taper les touches shift droit `+ ENTER` pour entrer une liste vide au niveau 1 de la pile

6. Taper 4 fois sur `+` pour constituer une liste contenant $\sin(x)$, P_7, P_5 et P_3 à partir de la liste vide entrée précédemment.

7. Lancer le menu principal (`PRG`), choisissez l'option `8. GRAPH` puis `PLOT LEVEL 1`

8. Déplacer le curseur sur `X-VIEW` pour indiquer par exemple -4 et 4 comme intervalle en X, puis sur `Y-VIEW` pour indiquer -2 et 2 en Y.

9. Taper alors sur la touche menu `F` correspondant à l'option `DRAW` du bandeau. Les graphes de P_3, P_5, P_7 et $\sin(x)$ se tracent alors successivement.

2.4.2 Manipulation de la pile.

On peut déplacer des objets sur la pile sans les modifier à l'aide de l'application pile interactive. Elle se lance par le menu principal `PRG`, `0.EDITORS`, `3.EDIT STACK` ou par le raccourci clavier $\boxed{\uparrow}$ (flèche vers le haut). Le niveau courant de la pile est alors marqué par une flèche vers la droite. On se déplace dans la pile à l'aide des flèches de déplacement vers le haut et vers le bas. Il est possible d'effectuer un mouvement circulaire des niveaux 1 jusqu'au niveau courant à l'aide des commandes `ROLL` et `ROLLD`, on peut recopier le niveau courant au niveau 1 avec `PICK`, on peut créer une liste avec les objets du niveau 1 au niveau courant en tapant `→LIST`. On peut enfin éditer l'objet situé au niveau courant de la pile avec la commande `EDIT` (shift gauche $\pm$).

Pour quitter la pile interactive, taper la touche `ENTER` ou la touche `ON`.

Deux raccourcis claviers bien utiles:

- L'appui sur la touche `ENTER` duplique le niveau 1 de la pile
- L'appui sur la touche $\rightarrow$ échange les niveaux 1 et 2 de la pile.

[1] On peut aussi utiliser la fonction `LASTARG` en tapant shift droit $\boxed{\text{EEX}}$, puis on modifie l'ordre au niveau 1

2.5 Variables, variable courante, fonctions

2.5.1 Création et destruction de variables

Lorsqu'on veut utiliser plusieurs fois la même expression, il est plus commode de la placer dans une variable que de la laisser sur la pile. En général, la pile sert à stocker les résultats des commandes mais pas les arguments des commandes. Pour stocker une expression dans une variable, on place sur la pile successivement l'expression à conserver et le nom de la variable à créer. Par exemple, pour placer `'SIN(X)'` dans la variable `'ABC'` on tape sur les touches `' SIN DEL ENTER ' A B C ENTER`. L'expression à stocker est alors au niveau 2 et le nom de variable au niveau 1. Il reste alors à taper la touche `STO`.

En appuyant sur la touche `VAR`, on fait apparaitre un bandeau qui contient les noms des variables (du répertoire courant). Pour détruire une variable, il faut placer son nom au niveau 1 et taper sur les touches shift gauche `EEX` (repérable à la légende `PURG`). Par exemple, si on tape `' A B C` shift gauche `EEX`, on fait disparaitre `ABC` du bandeau.

Raccourcis claviers lorsque le bandeau `VAR` est affiché:

- touche du bandeau: évalue l'objet contenu dans la variable, donc le place sur la pile si ce n'est pas un programme.

- shift gauche, touche du bandeau: stocke le niveau 1 de la pile dans la variable

- shift droit, touche du bandeau: rappelle le contenu de la variable sur la pile (sans l'évaluer s'il s'agit d'un programme)

Attention, veillez à ne pas confondre les shifts si vous utilisez ces raccourcis claviers.

2.5.2 Exemple

Revenons à l'exemple précédent: graphe de $\sin(X)$ et de ses développements de Taylor au voisinage de 0. On peut stocker $\sin(X)$ dans `F0`, $X - \frac{X^3}{3}$ dans `F3`, $X - \frac{X^3}{3} - \frac{X^5}{5}$ dans `F5`, etc. On lance alors `PRG`, `8.GRAPH`, `PLOT MENU` (raccourci shift droit 8). Pour entrer la liste des fonctions à tracer, choisissez la commande `CHOOSE` du bandeau. Ceci affiche la liste des variables du répertoire courant: il suffit alors de cocher avec la commande `CHK` du bandeau les variables contenant les expressions dont on veut le graphe. On sort de `CHOOSE` avec `OK` puis on fait `ERASE` et `DRAW` pour tracer les graphes.

2.5.3 Les variables d'`Erable`

`Erable` possède 3 variables réservées:

- la variable "courante" qui est contenue dans `VX`, généralement il s'agit `X`. C'est par rapport à cette variable qu'`Erable` dérive, intègre ou factorise
- la variable `MODULO`: c'est un entier n. Les instructions modulaires sont effectuées dans $\mathbb{Z}/n\mathbb{Z}$.
- la variable `EPS`: c'est un réel positif ε qui contrôle les erreurs d'arrondis. Tout nombre flottant de module inférieur à ε est considéré comme nul (en mode numérique)

D'autre part, l'instruction `RISCH` crée la variable `PRIMIT` pour conserver la dernière primitive calculée et `RISCH` (respectivement `DSOLVE`) modifie la variable `ERABLMSG` (`ODETYPE`) qui donnent des informations sur l'intégration (l'équation différentielle).

2.5.4 Définir une fonction

Pour créer une fonction, il existe une méthode assez intuitive: on tape
`'F(X)=`*expression*`'`
où *expression* est la définition de la fonction. Par exemple `'F(X)=SIN(X)/X'`. Puis on tape les touches shift gauche `STO` (légende `DEF`).

Faites apparaitre le bandeau `VAR` en tapant sur la touche `VAR`. Pour utiliser la fonction `F`, il suffit de placer la valeur de `X` au niveau 1 et de taper la touche menu correspondant à `F` dans le bandeau.

Pour définir la fonction composée de 2 fonctions, par exemple $F \circ G$ composée de F et G, il suffit de taper un programme:
`<< G F >>`
et de le stocker dans une variable, par exemple `FoG`.

Attention, un programme est exécuté de gauche à droite, donc dans cet exemple `G` est exécuté avant `F`.

2.6 Configuration

2.6.1 Le menu de configuration.

1. Appuyer sur la touche `PRG` pour avoir le menu principal
2. Aller à la rubrique `4.SETUP` de ce menu en tapant `4` puis `ENTER` (ou en utilisant la flèche vers le bas puis `ENTER`)

On peut aussi utiliser le raccourci clavier α-shift droit-`CST` (légende `MODES`). On obtient alors le menu:

1. `CALC MODES` (raccourci α `CST`):
   Permet de changer les unités d'angles, le système de coordonnées (rectangulaire, cylindrique ou sphérique) et les "drapeaux" (`FLAGS`)

2. `RESET ERABLE` (commande `VER`):
   remet `Erable` dans son état standard

3. `REAL MODE` (commande `13 CF`):
   passage en mode réel

4. `COMPLEX MODE` (commande `13 SF`):
   passage en mode complexe

5. `INTEGER ARIT` (commande `10 SF`):
   les commandes arithmétiques d'`Erable` telles que `GCD3` (identité de Bezout) considèrent que les arguments sont des entiers

6. `POLYN ARIT` (commande `10 CF`):
   Inverse de la précédente, les arguments sont considérés comme étant des polynomes (les entiers sont considérés comme des polynçomes constants)

7. `NUMERIC MODE` (raccourci shift gauche `EVAL`):
   Mode numérique. Les réels (ou complexes) même entiers sont considérés comme des réels (inexacts)

8. `SYMBOLIC MODE` (raccourci α shift droit `Q`):
   Mode symbolique. Les réels (ou complexes) entiers sont consiérés comme des entiers c'est-à-dire exacts.

9. `TIME` (raccourci shift droit `TIME`)
   Réglage de l'heure

ATTENTION, certaines commandes d'`Erable` font passer la machine automatiquement dans un mode. Par exemple, si un réel non entier est rencontré dans une expression, la HP48 passe en mode numérique.

2.6.2 Problèmes de configuration.

Toute la description qui vient d'être faite suppose que la machine est en mode `USER` et utilise l'affichage modifié de la librairie `EQSTK`. Il se peut que par une erreur de manipulation, la machine ne soit plus dans cet état. Voici comment remettre la machine dans l'état `USER/EQSTK`.

Enfoncez la touche `ON`, maintenez-la enfoncée et tapez sur la touche `C`, relachez la touche `ON`. Ceci a pour effet de "rebooter" la HP48. Après quelques secondes, tapez les touches shift gauche puis α (pour remettre le mode `USER`), puis tapez les touche α `A S T K` pour relancer l'affichage algébrique de la pile.

2.7 Les commandes les plus utilisées d'`Erable`

Outre la redéfinition des opérations arithmétiques usuelles, voici la liste de quelques instructions bien utiles.

2.7.1 Simplification, dérivation, intégration, développement de Taylor.

- `EXPA`: développe une expression, exemple:
  $(x + 12)^{16}$ `EXPA`
  Pour des expressions transcendentes, utiliser `TEXPA`.

- `COLC`: factorise une expression, exemple: essayez sur l'expression précédente!

- `der1` (raccourci clavier α-shift droit-∂) dérive une expression par rapport à la variable courante, exemple:
  `'1/2*SIN(X)' der1`

- `RISCH` (raccourci α-shift droit-$\int$) intégre une expression par rapport à la variable courante. Exemples :

$$\begin{array}{rcl} \dfrac{1}{x^2-4} & \texttt{RISCH} & \dfrac{1}{4}\ln(x-2) - \dfrac{1}{4}\ln(x+2) \\ x\ln(x) & \texttt{RISCH} & \dfrac{1}{2}x^2\ln(x) - \dfrac{1}{4}x^2 \\ (1+2x^2)\exp(x^2) & \texttt{RISCH} & x\exp(x^2) \end{array}$$

- `LIMIT`: par exemple:
  `'X*LN(X)' 'X=0' LIMIT`

- `SERIES`: calcule le développement de Taylor d'une fonction en un point à un ordre. Par exemple:
  `'EXP(SIN(X))' X 4 SERIES`: développement de Taylor de $e^{\sin(x)}$ en $x = 0$ à l'ordre 4.

SERIES sait aussi lever des indéterminations et effectue aussi des développements asymptotiques:
'SIN(X)/(EXP(X)-1)' SERIES
'X^X' 'X=0+0' SERIES (développement asymptotique en 0^+).

2.7.2 Changement de variable.

L'instruction EXEC (raccourci α-shift droit-EVAL) permet de faire des substitutions et des changements de variables. On place l'expression au niveau 2 et une équation au niveau 1 qui représente le changement à effectuer, par exemple:
'X^2+Y^2' 'X=1' EXEC

2.7.3 Arithmétique.

Les instructions GCD1 et LCM1 renvoient respectivement le plus grand diviseur commun et le plus petit multiple commun de leurs 2 arguments (entiers, complexes, polynômes). La commande PF décompose une fraction en éléments simples.

2.7.4 Algèbre linéaire

Pour résoudre un système linéaire: tapez le système suivi de la liste des variables puis de SYST. Pour réduire une matrice (pivot de Gauß): rref. Pour diagonaliser une matrice JORDAN.

2.7.5 Exercices

1. Développer les expressions suivantes:

$$(x+3)^7 \times (x-5)^6, \quad \sqrt{3+2\sqrt{2}}, \quad \frac{1+\sqrt{2}}{1+2\sqrt{2}}, \quad e^{i\pi/6}, \quad \ln(1+i)$$

2. Factoriser sur $\mathbb{R}$ et $\mathbb{C}$:

$$x^8 - 3x^7 - 25x^6 + 99x^5 + 60x^4 - 756x^3 + 1328x^2 - 960x + 256$$

$$x^6 - 2x^3 + 1, \quad (-y+x)z^2 - xy^2 + x^2y$$

3. Calculez les intégrales et simplifiez le résultat:

$$\int \frac{1}{e^x - 1}\, dx, \quad \int \frac{1}{x\ln(x)} \ln(\ln(x)) dx, \quad \int e^{x^2}\, dx, \quad \int x\sin(x)e^x\, dx$$

Vérifiez en dérivant les expressions obtenues.

4. Déterminer la valeur de:
$$\int_1^2 \frac{1}{(1+x^2)^3}, \quad \int_1^2 \frac{1}{x^3+1}\,dx$$
5. Vérifier que
$$\sin(2x) - 2\sin(x)\cos(x) = 0$$
en passant en exponentielles complexes.
6. Déterminer la valeur de
$$4\arctan(1/5) - \arctan(1/239)$$
7. Calculer le développement de Taylor en $x = 0$ à l'ordre 4 de:
$$\ln(1+x+x^2), \quad \frac{\exp(\sin(x))-1}{x+x^2}, \quad \sqrt{1+e^x}, \quad \frac{\ln(1+x)}{\exp(x)-\sin(x)}$$
8. Résoudre le système linéaire:
$$\left\{\begin{array}{ccccccc} x & + & y & + & az & = & 1 \\ x & + & ay & + & z & = & 2 \\ ax & + & y & + & z & = & 3 \end{array}\right.$$
9. Déterminer la liste des diviseurs de 45768.
   Factoriser 100!
10. Déterminer la primitive de:
$$\frac{-e^x\ln(x)^2 + \frac{2(e^x+1)}{x}\times\ln(x) + e^x + e^{2x}}{1+2e^x+e^{2x}}$$
11. Déterminer l'inverse et le déterminant de:
$$\begin{pmatrix} 1 & 1 & 1 & a \\ 1 & 1 & a & 1 \\ 1 & a & 1 & 1 \\ a & 1 & 1 & 1 \end{pmatrix}$$

Indications:

1. Utiliser `EXPA` (raccourci α-shift droit-$\boxed{\text{SPC}}$)

2. Utiliser COLC (raccourci α-shift droit-$\boxed{1}$ $\boxed{\text{B}}$)

3. Utiliser RISCH (raccourci α-shift droit-$\boxed{\text{COS}}$: c'est la touche $\int$). Vérifiez le résultat en dérivant la primitive: rappelez la variable PRIMIT sur la pile puis tapez der1 (raccourci α-shift droit-$\boxed{\text{SIN}}$ i.e. la touche ∂). La troisieme expression n'admet pas de primitive exprimée à l'aide des fonctions élémentaires: c'est le sens du texte "No closed form" dans la variable ERABLMSG.

4. Comme ci-dessus RISCH puis EXPA

5. EXPLN puis EXPA.

6. TSIMP puis EXPA.

7. Utiliser l'instruction SERIES: tapez la fonction, la variable et l'ordre désiré.

8. Taper la matrice symbolique du système puis rref. Ou tapez:
   `{ 'X+Y+A*Z=1' 'X+A*Y+Z=2' 'A*X+Y+Z=3' { X Y Z } }`
   puis SYST ou SOLGEN.

9. Tapez DIVIS.
   100 fact COLC.

10. TSIMP RISCH donne la réponse en une bonne minute.

11. Taper la matrice puis INVL (raccourci α-shift droit-$1/x$) ou RDET selon le cas. Pour faire les deux opérations à la suite, pensez à dupliquez la matrice en tapant ENTER puis échangez les niveaux 1 et 2 de la pile (raccourci clavier flèche vers la droite, fonction SWAP).

Chapitre 3

Les commandes d'`Erable`

3.1 Simplifications

3.1.1 Simplification rationnelle: EXPA.

La première des fonctionnalités d'un système de calcul formel est de simplifier des expressions. On simplifie en calculant le pgcd du numérateur et du dénominateur (par l'algorithme d'Euclide ou des algorithmes plus efficaces, ici le pgcd heuristique) et en utilisant des opérations arithmétiques élémentaires (pour simplifier une fraction contenant des racines carrées ou des complexes, on multiplie le dénominateur par l'expression conjuguée: par exemple $\sqrt{2} - 1/(1+\sqrt{2})$, $1/(1+i\sqrt{3})^3$). Par exemple, pour simplifier 15/10 on recherche le plus grand diviseur commun de 15 et 10, on trouve 5, et on divise 15 et 10 par 5 pour obtenir 3/2. Cet algorithme se généralise à des fractions de polynômes à une ou plusieurs variables. Par exemple $(x^2+2x+1)/(x^2-1)$.

La commande corrspondante est `EXPA`

3.1.2 Développer des expressions transcendentes: TEXPA

La fonction `TEXPA` applique les règles de développement usuelles des fonctions exponentielles, logarithme et trigonométriques:

$$e^{x+y} \to e^x e^y, \ln(xy) \to \ln(x) + \ln(y), \sin(x+y) \to \sin(x)\cos(y) + \cos(x)\sin(y), ...$$

Transformations trigonométriques

La fonction TRIGCOS convertit les fonctions tangentes et sinus en cosinus en utilisant les formules:

$$\sin(x)^2 = 1 - \cos(x)^2$$

puis exécute une simplification rationnelle et convertit à nouveau en sinus s'il reste des racines carrées. La fonction TRIGSIN agit de manière similaire en échangeant les rôles de sin et cos. Les fonctions TAN2SC et TAN2SC2 permettent d'éliminer les fonctions tangentes en appliquant respectivement les règles:

$$\tan(x) = \frac{\sin(x)}{\cos(x)}, \quad \tan(\frac{x}{2}) = \frac{1-\cos(x)}{\sin(x)}$$

La fonction HALFTAN exprime les fonctions trigonométriques en fonction de la tangente de l'angle moitié. Enfin, la fonction TRIG additionne les effets de TAN2SC2 et de TRIGCOS (ou TRIGSIN selon l'état du drapeau 26).

3.1.3 Formules d'Euler

La fonction SINCOS transforme les exponentielles complexes en fonctions trigonométriques. L'opération inverse est réalisée par EXPLN.

3.1.4 Linéarisation.

EXPLIN linéarise des exponentielles alors que TRIGLIN linéarise des fonctions trigonmétriques. *Exemples:*

```
'SIN(X)*COS(X)'  TRIGLIN  '1/2*SIN(2*X)'
'EXP(X)^2'       EXPLIN   'EXP(2*X)'
```

3.1.5 Autres transformations.

La fonction LNCOLC exécute l'opération:

$$\ln(x) + \ln(y) \to \ln(xy)$$

Elle est donc inverse de TEXPA pour des logarithmes.

Il se peut qu'en appliquant les transformations ci-dessus on n'arrive pas à simplifier complètement une expression contenant des fonctions transcendantes (exponentielle, logarithme, fonctions trigonmétriques et trigonométriques inverses) La simplification générale de telles expressions n'est pas triviale. La fonction TSIMP essaie de minimiser le nombre de “variables” algébriquement indépendentes (ce qui permet ensuite de n’avoir que des simplifications rationnelles à effectuer). En général, on applique `EXPA` après `TSIMP`. L’algorithme de `TSIMP` est le suivant:

- ramener les fonctions trigonométriques (directes et inverses) sous forme d’exponentielles et de logarithmes.

- appliquer ensuite les règles:

  $$e^{x+y} = e^x e^y, \quad \ln(xy) = \ln(x) + \ln(y), \quad e^{\ln x} = x, \quad \ln e^x = x, \quad x, y \in \mathbb{R}^{+*} \tag{3.1}$$

  (ce sont en fait les seules règles permettant de simplifier des expressions contenant des logarithmes et des exponentielles). On regarde d’abord les arguments des fonctions ln, s’ils ont un diviseur commun, on factorise ce diviseur et on applique une des règles (3.1). Pour les exponentielles, il faut regarder si un argument est combinaison linéaire à coefficients rationnels des arguments des autres exponentielles et des autres logarithmes de l’expression.

Exemples: testez l’effet de `TSIMP` sur les expressions suivantes:

$$\frac{e^{2x}-1}{e^x-1}, \quad \ln(x^2-1) - \ln(x^2+2x+1), \quad e^{x^2+3x+y/2} - e^{x^2+2x+y} e^{1/2\times x^2+2x}$$

De nouveaux problèmes apparaissent lorsque les arguments de (3.1) sont complexes, les égalités deviennent fausses en général, par exemple:

$$\ln(e^{2i\pi}) = \ln(\cos(2\pi) + i\sin(2\pi)) = \ln(1) = 0 \neq 2i\pi \tag{3.2}$$

Attention donc, chaque logarithme complexe peut entrainer une erreur de $2i\pi$ dans le résultat renvoyé (cf. l’équation (3.2)). De même une fonction arctangente peut entrainer une erreur de π car elle est transformée en un logarithme complexe multiplié par $1/(2i)$. Ceci est également vrai pour la fonction `EXPA`. Par exemple si on applique `EXPA` à:

$$\arctan(x) + \arctan(1/x)$$

on trouve $\pi/2$ qui est incorrect si $x < 0$. Il est donc judicieux d’évaluer numériquement le résultat pour déterminer ce type d’erreurs.

Notez enfin qu’en mode complexe (`13 SF`) les exponentielles et logarithmes ayant un argument complexe sont laissés tels quels par `TSIMP`. Par contre en mode réel (`13 CF`) les exponentielles et les logarithmes complexes sont convertis en fonctions trigonométriques.

3.2 Arithmétique entière, modulaire et polynomiale

`Erable` effectue de l'arithmétique sur les entiers, les entiers de Gauß (c'est-à-dire des complexes dont la partie réelle et la partie imaginaire sont des entiers), les polynômes de une ou plusieurs variables à coefficients entiers ou entiers de Gauß et les matrices et vecteurs à coefficients entiers, entiers de Gauß ou polynômiaux. Certaines opérations arithmétique donnent des résultats différents selon qu'on considère un entier comme élément de l'anneau $\mathbb{Z}$ ou comme polynôme constant. Par défaut `Erable` suppose que les arguments sont des polynômes, si vous travaillez avec des entiers, tapez `10 SF`. Enfin, `Erable` reconnait les symboles $\pm\infty$, ∞ (égal à $+\infty$ en valeur absolue) et ? (indéterminé) et exécute les opérations arithmétiques usuelles sur ces arguments.

3.2.1 Arithmétique usuelle.

Les fonctions suivantes sont disponibles:

- Les opérations usuelles sont implémentées (addition: `add`, soustraction: `SUBT`, multiplication: `MULT`, division: `DIV1`, racine carrée: `SQRT`, puissance entière: `POWER`, opposé `CHS`,
- pour les nombres complexes: partie réelle `re`, imaginaire `im`, conjugué `conj`, module `abs`, argument `arg`
- pgcd (`GCD1`), ppcm (`LCM1`) des arguments situés aux niveaux 1 et 2. La fonction `LGCD` renvoie le pgcd de tous les éléments de la liste située au niveau 1.
- factorisation: voir la section 3.3.
- `SIMP2`: simplifie les éléments situés aux niveaux 1 et 2 en les divisant par leur pgcd.
- `GCD3`: pgcd étendu. Renvoie u, v et d tels que $au+bv = d$ (où a et b désignent les valeurs des niveaux 2 et 1 de la pile). Rappel: par défaut `GCD3` considère ses arguments comme étant polynomiaux.
- `DIV2`: division euclidienne. Renvoie le quotient au niveau 2 et le reste au niveau 1. Comme `GCD3`, `DIV2` suppose que les arguments sont des polynômes sauf si vous tapez `10 SF` avant d'appeler `DIV2`.

- `ABCUV`: résoud une équation de type Bezout pour des polynômes:

  $$ax + by = c$$

  où a, b et c sont trois polynômes (tels que le PGCD de a et b divise c sinon il n'y a pas de solutions). Vous entrez dans l'ordre a, b, c sur la pile et le programme renvoie x et y aux niveaux 2 et 1. S'il n'y a pas de solutions, `ABCUV` provoque l'erreur `Bad Argument type`.

  Exemples:

  ```
  'X^2+2*X+1' 'X^2+3*X+2' 'X+1'    ABCUV  -1 1 1
  'X^2+2*X+1' 'X^2+3*X+2' 1        ABCUV  erreur
  ```

  Dans le premier cas, on a:

  $$(X + 1) = (X^2 + 2X + 1) \times (-1) + (X^2 + 3X + 2) \times 1$$

  alors que dans le second cas il n'y a pas de solution, car le PGCD de x^2+x+1 et $x^2 + 3x + 2$ ne divise pas 1.

- `fact` et `comb`: identiques aux instructions `FACT` (factorielle) et `COMB` (combinaisons de p éléments parmi n) mais avec tous les chiffres significatifs.

- `PA2B2` (librairie `kernel.lib`): décompose un nombre premier p congru à 1 modulo 4 en somme de 2 carrés $p = a^2 + b^2$, `PA2B2` renvoie le nombre complexe $a + ib$. Cet algorithme est utilisé pour factoriser des entiers de Gauß .

- `EULER` retourne l'indicatrice d'Euler d'un entier (i.e. le nombre d'entiers naturels $p \in [1, n]$ premiers avec n), par exemple l'indicatrice d'Euler de 12 est 4 car seuls 1,5,7,11 sont inférieurs à 12 et premiers avec 12.

- `XFRC`: transforme un objet algébrique, réel ou complexe ou une liste contenant des objets de ce type à la manière de l'instruction intégrée →`Q` mais en détectant les nombres quadratiques. Par exemple: 1.20710678118 devient `'1/2+1/2*√2'` et 1.5 `'3/2'`.

3.2.2 HORN, PTAYL et polynômes orthogonaux.

Pour les polynômes, `Erable` propose en complément des opérations usuelles les instructions spécifiques suivantes:

- HORN (méthode de Horner):
  on place le polynome P puis un objet r sur la pile. HORN renvoie le quotient euclidien de $P(X)$ par $X - r$ au niveau 3, r au niveau 2 et $P(r)$ au niveau 1. Par exemple:
  'X^2+2*X+3' 5 HORN
  renvoie 'X+7' au niveau 3, 5 au niveau 2 et 38 au niveau 1.

- PTAYL (développement de Taylor):
  on place le polynôme P puis un objet r sur la pile, PTAYL renvoie $P(X - r)$. Par exemple:
  'X^3+2*X+2' PTAYL
  renvoie 'X^3+6*X^2+14*X+12' ce qui signifie que:

$$P(x) = (x-2)^3 + 6(x-2)^2 + 14(x-2) + 12$$

- LEGENDRE, TCHEB,HERMITE: prennent en argument un entier n et renvoient le n-ième polynôme de Legendre ou de Tchebycheff ou de Hermite.

3.2.3 Arithmétique modulaire

Les fonctions MODADD, MODSUBT, MODMULT, MODDIV, MODPOW, MODINV implémentent les opérations arithmétiques usuelles mais dans $\mathbb{Z}/n\mathbb{Z}$ où n est l'entier placé dans la variable MODULO.

Une des applications bien connue du calcul modulaire est par exemple la cryptographie (à clef publique), on trouvera une implémentation de l'algorithme RSA dans les répertoires crypt et decrypt. Le calcul modulaire intervient également dans de très nombreux algorithmes de calcul formel.

3.3 Factorisation.

3.3.1 Factorisation d'entiers: COLC, FACTO, DIVIS

Le logiciel ALG48 (et certaines versions d'Erable où ces routines sont reprises) contiennent quelques méthodes avancées de factorisation d'entier (méthode de Pollard, Brent, ...). On peut ainsi factoriser en un temps raisonnable des nombres de 18 à 20 chiffres. Des tests de pseudo-primalité permettent de tester des nombres jusqu'à 30 chiffres.

Les routines de factorisation sont COLC (sous forme symbolique) et FACTO qui renvoie la liste des facteurs et multiplicité. La fonction DIVIS renvoie la liste des diviseurs d'un entier.

3.3.2 Factorisation d'entiers de Gauß : COLC, FACTO, DIVIS.

A partir de la version 3.1, `Erable` peut factoriser des nombres complexes avec les mêmes commandes. Il commence par factoriser sur les entiers le pgdc de la partie réelle et de la partie imaginaire du complexe puis, en prenant le module au carré du complexe et en appliquant `PA2B2` aux facteurs premiers congrus à 1 modulo 4 de ce module au carré, il termine la factorisation sur les entiers de Gauß .

3.3.3 Factorisation de polynômes et de fractions rationnelles: COLC, FACTO, DIVIS, FROOTS, FCOEF.

On sait trouver toutes les racines d'un polynôme numériquement sur $\mathbb{C}$, mais pour un polynômes symbolique de degré supérieur ou égal à 5 [1], on peut montrer qu'il est génériquement impossible d'exprimer les racines en fonction des coefficients (résultat dû à Galois). Il existe par contre des algorithmes qui factorisent des polynômes symboliques à coefficients rationnels en produit de polynômes irréductibles sur les rationnels. Par exemple, $x^3 - 1 = (x-1)(x^2+x+1)$. Ce sont des algorithmes p-adiques qui cherchent à remonter des factorisations de $\mathbb{Z}/p^k\mathbb{Z}[X_1, ..., X_n]$ à $\mathbb{Z}[X_1, ..., X_n]$. Sur la HP48, la fonction `FCTR` d'`ALG48` permet de factoriser des polynômes à coefficients rationnels en produits de facteurs irréductibles. Cet algorithme n'est pas (encore) implémenté dans `erable` (par exemple `COLC` ne factorise pas la version développée de $(x^4+x^3+1)(x^4+x+1)$). On se contente de déterminer la factorisation *square-free* et les racines évidentes. Ce qui devrait néanmoins suffire dans les applications rencontrées par les utilisateurs de la HP48. Par contre `erable` peut factoriser certains polynômes à coefficients complexes non factorisables par `ALG48` (par exemple $(x+i)(x+1+i)(x+1-i)$).

Les commandes de factorisation sont `COLC` (sous forme symbolique), `FACTO` (liste des facteurs), `FROOTS` (liste alternant les racines et leur multiplicité), `FCOEF` (l'inverse de `FROOTS`), `DIVIS` (liste des diviseurs). Attention, la variable contenue dans `VX` est considérée comme variable principale lors d'une factorisation.

Exemples:
On tape une liste alternant les racines et leur multiplicité respective:
`{ 1 2 -1 3 2 -1}` `FCOEF` renvoie:

$$\frac{(x - -1)^3 (x-1)^2}{x-2}$$

[1] En degré 4, les formules existent mais sont tellement abonimables qu'elles ne sont pas utilisées par les logiciels de calcul formel

(Attention aux drapeaux: tapez VER avant de commencer). Si on applique FROOTS à cette expression, on obtient la liste donnée en argument de FCOEF. Par contre FACTO renverra la liste des facteurs au lieu de la liste racines.

3.3.4 Décomposition en éléments simples.

L'une des applications de la factorisation est la recherche de la décomposition en éléments simples d'une fraction rationnelle sur $\mathbb{R}$ ou sur $\mathbb{C}$. La fonction PF effectue cette décomposition (sur $\mathbb{R}$ ou $\mathbb{C}$ selon l'état du drapeau 13).

3.4 Intégration et dérivation: RISCH, der, der1.

On apprend au lycée comment dériver une fonction, et ce sont ces règles que tous les systèmes de calcul formel appliquent:

$$(f+g)' = f' + g', \quad (fg)' = f'g + fg', \quad (1/g)' = -g'/g^2, ...$$

Il suffit donc de connaître la dérivée des fonctions usuelles.

Le problème inverse est autrement plus difficile. Étant donné une fonction f, il s'agit de savoir si elle admet une primitive g qui s'exprime au moyen des fonctions élémentaires (donc on veut résoudre $g' = f$ et exprimer g à l'aide de fonctions usuelles). On apprend en premier cycle un grand nombre de méthodes heuristiques permettant d'intégrer certaines classes de fonctions (fractions rationnelles, linéarisation de polynômes trigonométriques, fraction rationnelles en $\sin(x)$, $\cos(x)$, racines carrées de polynômes du second degré, intégration par parties, ...). Ces méthodes ont l'avantage de la rapidité mais lorsqu'elles échouent (si la fonction à intégrer n'est pas dans un classe connue), on ne sait pas si la fonction admet une primitive exprimable à l'aide de fonctions élémentaires. Il existe pourtant un algorithme (Risch-Bronstein: *cf.* [5]) qui permet de le faire lorsque les fonctions usuelles autorisées sont les logarithmes, les exponentielles, et les extensions algébriques, mais il n'est actuellement complètement programmé que sur le système Scratchpad II (IBM). L'idée de l'algorithme est une généralisation de la méthode de décomposition en éléments simples des fractions rationnelles. Sur la HP48, la version actuelle d'erable est capable de déterminer g si elle existe lorsque f ne contient que des extensions emboitées purement transcendantes (ce qui exclut par exemple les racines carrées) et une seule extension exponentielle sur la variable.

Quelle que soit la méthode utilisée (heuristique ou Risch), avant d'intégrer, il faut savoir factoriser le dénominateur des fractions rencontrées.

3.4.1 Dérivation et gradient: der, der1

La fonction `der1` calcule la dérivée du niveau 1 de la pile par rapport à la variable courante. La fonction `der` calcule la dérivée d'une (liste de) fonction(s) par rapport à 1 variable ou le gradient d'une (liste de) fonction(s) par rapport à une liste de variables.

Exemple:
`{ 'X^2+LN(X)' 'X*Y' } der1`
renvoie `{ '2*X+INV(X)' 'Y' }`,
`'X^2+2*X*LN(Y)-1/Y' { X Y } der`
renvoie `{ '2*X+2*LN(Y)' '2*X*(1/Y)+1/Y^2' }`

Ces deux fonctions de dérivation sont presque identiques au programme de la ROM de la HP48, la différence principale est que les expressions comme $\sqrt{2}$ ou $\frac{1}{3}$ ne sont pas évaluées. Une autre différence apparaît lorsqu'on dérive des fonctions utilisateurs. La notation de la HP `'derZ(X,1)'` ne me paraît pas naturelle pour la dérivée de la fonction `'Z(X)'`, elle le devient encore moins pour des fonctions à plusieurs variables. J'ai décidé d'introduire les notations usuelles dans le domaine des équations aux dérivées partielles: $\partial_k f$ désigne la dérivée partielle de f par rapport à la k-ième variable. Donc `'Z(X)' X der` renvoie `d1Z(X)`. Cette notation est reconnue par l'instruction `EXEC` et permet de faire des changements de fonction dans des équations différentielles. Exemples:
`'d1Z(X)' 'Z(X)=4*X^3+1' EXEC` renvoie $12x^2$.
`'d1Z(X)+Z(X)' 'Z(X)=EXP(-X)*Y(X)' EXEC EXPA` renvoie `'d1Y(X)*EXP(-X)'`.

3.4.2 Intégration: RISCH, IPP, EXEC

La commande `RISCH` permet de traiter un assez grand nombre d'intégrands. La dernière primitive calculée est conservée dans la variable `PRIMIT`. Pour l'évaluer entre deux bornes, rappelez si nécessaire la primitive sur la pile, puis tapez la borne inférieure et la borne supérieure et enfin `PREVAL`.

Vous pouvez également évaluer des intégrales définies (avec des bornes). Par exemple, tapez:

$$\int_3^{15} \frac{1}{X^2-1}\, dX,$$

puis `EXPA`. La primitive formelle $-1/2\ln(x+1)+1/2\ln(x-1)$ stockée dans la variable `PRIMIT`.

Fractions rationnelles

Normalement (!) `RISCH` intègre toutes les fractions rationnelles à coefficients réels ou complexes, mais en cas de factorisation numérique, les erreurs de calcul s'additionnant, les résultats pourraient être mauvais. De plus en intégration formelle, les calculs peuvent s'allonger. Enfin, en calcul avec paramètres, il n'est pas sûr que la calculatrice arrive à factoriser le polynôme. Vous pouvez par exemple intégrer:

$$\frac{1}{X^2 - A^2} \quad \text{ou} \quad \frac{1}{X^3 - A^3} \quad \text{(ce sera long)}$$

mais pas:

$$\frac{1}{X^4 + A^4}$$

En général la factorisation du dénominateur n'est pas assurée au-delà du degré 2 (le programme de factorisation détectera néanmoins les racines évidentes). Pour ce genre de polynôme, il est souvent intéressant de laisser le dénominateur sous forme la plus factorisée connue. Par exemple pour $x^4 - a^4$, le programme s'exécutera plus rapidement si vous entrez au dénominateur `(X^2-A^2)(X^2+A^2)` au lieu de `X^4-A^4`.

Enfin, il est bien sûr impossible de prendre un degré qui ne soit pas de type entier (inutile d'essayer $\frac{1}{x^{2n}+1}$ car l'exposant `'2*N'` est un objet symbolique).

Remarque 1
- *Le rôle de l'indicateur utilisateur 13 est ici primordial. Si vous intégrez une fraction à coefficients réels et que vous désirez obtenir un résultat formel ne faisant pas intervenir les nombres complexes, désarmez le flag 13. Par exemple* `'1/(X^2+1)' RISCH` *donne avec le flag 13 armé:*
  `'i/2*LN(X+i)-i/2*LN(X-i)'`
  et avec le flag 13 désarmé `'ATAN(X)'`*.*
- *Le drapeau 13 peut passer à l'état 1 (*set*) lors de l'appel de* `RISCH`*.*

Exemples:

`13 CF '1/((X^2+1)*(X^2+2))' RISCH`

renvoie

$$\operatorname{atan}(x) - \frac{1}{2}\sqrt{2}\operatorname{atan}(\frac{x}{2}\sqrt{2}$$

alors que

`13 SF '1/((X^2+1)*(X^2+2))'RISCH` renvoie

$$\frac{-i}{2}\ln(x-i) + \frac{i}{2}\ln(x+i) + \frac{i}{4}\sqrt{2}\ln(x-i\sqrt{2}) - \frac{i}{4}\sqrt{2}\sqrt{2}\ln(x+i\sqrt{2}$$

Intégrands non fractionnaires

Si l'intégrand n'est pas une fraction, `RISCH` détermine s'il se ramène à la forme $P(x)e^{ax+b}$ où a et b peuvent être complexes. Ceci permet aussi de traiter les fonctions trigonométriques, mais uniquement en mode réel (`13 CF`), en mode complexe, il faut d'abord les transformer en exponentielles complexes en utilisant la commande `EXPLN`.

Fractions trigonométriques.

Le programme d'intégration `RISCH` permet également d'intégrer des fractions rationnelles contenant soit des $\sin x$ et $\cos x$, soit des $\sinh x$ et $\cosh x$. Attention, ils ne reconnaissent pas $\tan x$, ni $\tanh x$, ni $\sin 2x$, ... A vous de faire les conversions trigonométriques nécessaires avant d'appeler `RISCH` (par exemple en appelant `TEXPA`).

La méthode d'intégration est le changement de variables $y = \tan \frac{x}{2}$ ou $y = e^x$ selon le cas et le résultat que vous obtiendrez sera une fraction rationnelle de y. Pour revenir à des sinus et cosinus (hyperboliques), les formules sont les suivantes:

$$\tan \frac{x}{2} = \frac{1-\cos x}{\sin x}, \qquad e^x = \cosh x + \sinh x$$

Par exemple, pour trouver la primitive de la cosécante $\int \frac{1}{\sin x}\, dx$, on tape `'1/SIN(X)'` puis `RISCH` et on obtient en 10 secondes: `'LN(TAN(X/2))'`.

Remarque 2 *Attention: dans l'intégration de fractions rationnelles fonction de* $\sin(x), \cos(x)$*, le changement de variables effectué n'est pas bijectif. La primitive obtenue n'est donc valable que localement. Pour la recoller sur* $\mathbb{R}$ *tout entier, il faut recoller les arctangentes (lorsque cela est possible) en* $x = (2k+1)\pi$ *(*k *entier). Par exemple:*

- *pour* $\frac{1}{\sin x}$*, c'est impossible (car l'intégrale diverge en* $x = k\pi$*),*
- *pour* $\frac{1}{\sin x + 2}$*, on trouve comme primitive (*`13 CF`*):*

$$\frac{2}{3}\sqrt{3}\arctan\left(\frac{\tan\frac{x}{2}+\frac{1}{2}}{\frac{\sqrt{3}}{2}}\right),$$

pour $x = (2k+1)\pi$*, la tangente n'est pas définie, alors que la primitive l'est. Plaçons-nous par exemple en* $x = \pi^-$*. Notre résultat tend vers* $\frac{2}{3}\sqrt{3}\frac{\pi}{2}$*. Alors qu'en* $x = \pi^+$*, il tend vers* $-\frac{2}{3}\sqrt{3}\frac{\pi}{2}$*. Il faut donc rajouter* $2 \times \frac{2}{3}\sqrt{3}\frac{\pi}{2}$ *à notre résultat sur l'intervalle* $[\pi, 3\pi[$*. Le même raisonnement en* $(2k+1)\pi$ *pour* k *entier conduit à la primitive sur* $\mathbb{R}$ *tout entier:*

$$\int \frac{dx}{\sin x + 2} = \frac{2}{3}\sqrt{3}\arctan\left(\frac{\tan\frac{x}{2}+\frac{1}{2}}{\frac{\sqrt{3}}{2}}\right) + \frac{2\sqrt{3}}{3}\pi E(\frac{x+\pi}{2\pi}) + C.$$

Notez que le recollement des arctangentes peut provoquer des erreurs d'évaluation par la HP, la moindre erreur d'arrondi pour x voisin de $(2k+1)\pi$ est fatale!

Racines carrées de trinômes du second degré.

Le programme `RISCH` peut aussi intégrer des fractions rationnelles en deux variables: x et un trinôme du second degré en x. Attention, les trinômes elliptiques sont traités dans le complexe. Par exemple,
`'X/sqrt(X^2+X-2)' RISCH`
renvoie:

$$\frac{1}{2}(\sqrt{x^2+x-2}-x)+\frac{1}{2}\ln(2\sqrt{x^2+x-2}-1)-\frac{\frac{\frac{9}{2}}{2}}{2\sqrt{x^2+x-2}-1}$$

qu'on simplifie par `EXPA` en:

$$\frac{2\ln(-2x-1+2\sqrt{x^2+x-2})+1}{4}+\sqrt{x^2+x-2}$$

Autres intégrands

Lorsque `RISCH` ne reconnaît pas une des formes précédents, l'algorithme de Risch est exécuté. Pour l'instant l'implémentation ne traite pas les extensions multiples de type exponentiel qui ont une partie polynomiale. Ainsi, les exemples suivants sont acceptés:

$$\ln\ln(x), \quad \frac{1}{\exp^{\ln(x)^2+1}-1}, \quad x^3 e^{\left(\frac{x+1}{x+2}\right)}$$

Par contre, les fonctions suivantes:

$$\ln(x)\sqrt{x^2-1}, \quad e^{\ln(x)^2+1}$$

ne peuvent être intégrées par `RISCH` car la première est une extension algébrique et la deuxième est un polynôme par rapport à une extension exponentielle sur un logarithme.

Avant d'appeler `RISCH`, il peut être nécessaire d'utiliser `TSIMP` en mode complexe (`13 SF`) pour transformer les fonctions transcendentes en exponentielles et en logarithmes indépendants. Après calcul de la primitive par `RISCH`, il est souvent nécessaire de refaire la conversion inverse à nouveau en invoquant `TRIG`. Exemple:

$$\frac{\sin(x)-x\cos(x)}{x^2}, \quad \arctan(x)$$

3.4.3 Intégration par parties: IPP

Le programme `IPP` permet d'intégrer par parties en utilisant la formule:

$$\int_a^b u'v\ dx = [uv]_a^b - \int_a^b uv'\ dx$$

Il faut placer l'intégrale au niveau 2 et la fonction u au niveau 1 (attention il s'agit de la fonction u et non de la fonction u'). On peut réitérer le procédé, `IPP` applique l'intégration par parties à la première intégrale rencontrée après un éventuel signe =.

Exemple: calcul de la primitive de $\arcsin(x)^2$ qui s'annule en 0. Tapez:

$$\int_0^t \arcsin(x)^2\ dx$$

puis x puis `IPP` qui renvoie

$$\int_0^t \arcsin(x)^2\ dx = t\arcsin(t)^2 - \int_0^t \frac{-2x\arcsin(x)}{x^2-1}\sqrt{-x^2+1}\ dx$$

Pour terminer l'intégration, on tape $\sqrt{1-x^2}$ et on appelle `IPP` qui renvoie après simplification (`OBJ→` pour décomposer l'égalité en deux morceaux, `DROP2` pour supprimer 2 et = puis `EXPA EXPA`):

$$t\arcsin(t)^2 - 2t + 2\arcsin(t)\sqrt{1-t^2}$$

que l'on peut vérifier (en tapant `t der EXPA`).

Attention, le contenu de la variable `VX` du répertoire courant est modifié si on appelle `IPP`. Si vous intégrez systématiquement par rapport à x vous ne risquez pas d'avoir de mauvaises surprises.

3.4.4 Changement de variables dans une intégrale: EXEC

Taper simplement le changement sous la forme:
'variable=nouvelle valeur'
et appelez l'instruction `EXEC` (raccourci α-shift droit-EVAL). Par exemple pour poser $x = y^2$ dans:

$$\int_0^2 e^x\ dx$$

tapez l'intégrale puis `'X=Y^2' EXEC`. On peut aussi faire un changement implicite dans les bons cas, par exemple ci-dessus on peut revenir en x en tapant
`'Y^2=X' EXEC`.

3.5 Équations différentielles: DSOLVE, LDEC

Les problèmes reliés à l'intégration et à la dérivation qui ont énormément d'applications dans les sciences sont les équations différentielles. La résolution exacte d'équations différentielles est encore un secteur de recherches très actives. Sur la HP48, le programme `DSOLVE` est en cours de développement, la version actuelle permet de résoudre certaines équations du 1er ordre (linéaire, homogène, incomplète, variables séparables, Bernouilli, ...). Les instructions `LDEC`, `LAP` et `ILAP` permettent de résoudre des équations ou des systèmes linéaires à coefficients constants par transformation de Laplace et tranformée de Laplace inverse.

3.5.1 Équations linéaires à coefficients constants: LDEC

Pour résoudre ce type d'équations: on place au niveau 2 de la pile le second membre de l'équation et au niveau 1 l'équation caractéristique puis on appelle `LDEC` qui renvoie au niveau 1 la solution particulière dont toutes les dérivées s'annulent en 0. Les autres solutions s'obtiennent en ajoutant la solution de l'équation homogène. Les niveaux 2 et 3 sont sans intérêt ici, ils sont présents pour des raisons de compatibilité avec les systèmes différentiels linéaires à coefficients constants (section 3.5.4). Par exemple, pour résoudre l'équation

$$y'' + y = \sin(x)$$

dont l'équation caractéristique est $x^2 + 1$, on tape `'SIN(X)' 'X^2+1' LDEC` qui renvoie après simplifications (par `SINCOS EXPA`):

$$y(x) = -\frac{x\cos(x) - \sin(x)}{2}$$

La solution générale est donc:

$$y(x) = A\cos(x) + B\sin(x) - \frac{x\cos(x) - \sin(x)}{2}$$

3.5.2 Équations différentielles du premier ordre: DSOLVE

Par exemple pour résoudre:

$$y'(x) = xy(x) + 1 - x^2$$

on tape le second membre puis `DSOLVE`:
`'X*Y(X)+1-X^2' DSOLVE`

On obtient alors la solution passant par le point (x_0, y_0). Donc, si vous traitez un problème avec condition initiale, placez les valeurs initiales dans les variables `X0` et `Y0`. Le résultat renvoyé n'est pas simplifié, il faudra invoquer plusieurs fois `EXPA` pour calculer les intégrales et simplifier.

Pour les équations homogènes, `DSOLVE` renvoie une solution en paramétriques, rappelons que les autres solutions s'obtiennent par homothétie. Les solutions singulières ($y/x =$constante) ne sont pas calculées.

La vérification d'une solution peut se faire en utilisant l'instruction de changement de fonction `EXEC` (taper l'équation différentielle complète suivi du changement de variable, dans l'exemple $y(x) = \exp(x^2/2) + x$ est solution:
`'-d1Y(X)+X*Y(X)+1-X^2' 'Y(X)=EXP(X^2)+X' EXEC EXPA`)
L'instruction `EXEC` permet aussi de changer de fonction inconnue, par exemple ici on peut taper `'Y(X)=Z(X)+X'` pour se ramener à une équation linéaire homogène.

3.5.3 Transformée de Laplace et transformée inverse: LAP et ILAP

Pour résoudre les équations différentielles linéaires à coefficients constants, `LDEC` utilise les transformations de Laplace et de Laplace inverse définies par:

$$Y(p) = \mathcal{L}(y)(p) = \int_0^\infty e^{-px} y(x)\ dx, \quad \mathcal{L}^{-1}(Y)(t) = \frac{1}{2i\pi} \int_C e^{zt}\ dz \qquad (3.3)$$

où C est une courbe fermée contenant les pôles de Y. Les instructions `LAP` et `ILAP` permettent de calculer les transformées de Laplace et de Laplace inverse des fonctions qui interviennet habituellement dans les équations différentielles linéaires à coefficients constants issues de la physique.

La méthode de résolution de `LDEC` est la suivante: on a

$$\mathcal{L}(y')(p) = -y(0) + p\mathcal{L}(y)(p), \quad \mathcal{L}(y'')(p) = -py(0) - y'(0) + p^2\mathcal{L}(y)(p)$$

ce qui transforme l'équation différentielle en une équation linéaire. Par exemple, pour résoudre:

$$y'' + 2y' + y = \cos(x)$$

on prend la transformée de Laplace des deux membres de l'équation:

$$(p^2 + 2p + 1)Y(p) = \frac{p}{p^2 + 1} + (p + 2)y(0) + y'(0)$$

ici la transformée de Laplace de $\cos(x)$ se calcule facilement par la définition (3.3).

Pour finir la résolution, il suffit de calculer séparément les transformées inverses y_c, y_y et $y_{y'}$ de:

$$\frac{1}{p^2+2p+1}\frac{p}{p^2+1}, \quad \frac{p+2}{p^2+2p+1}, \quad \frac{1}{p^2+2p+1}$$

en appelant à chaque fois `ILAP`.

Attention, le nom de la variable du coté Laplace est identique à celui de la variable du coté x: c'est le contenu de `VX`. Pour tester l'exemple ci-dessus, tapez `X` au lieu de p ou stockez p dans `'VX'`.

La solution générale de l'équation est alors: $y_c + y(0)y_y + y'(0)y_{y'}$. Lorsque $y(0)$ et $y'(0)$ sont connus, il est plus simple de calculer directement:

$$\frac{1}{p^2+2p+1}\left(\frac{p}{p^2+1} + (p+2)y(0) + y'(0)\right)$$

et d'appeler une seule fois `ILAP`.

Ce procédé se généralise sans difficultés à des équations linéaires de tous ordres.

Dans l'exemple, on tapera `'COS(X)' LAP 'X^2+2*X+1' / ILAP` pour obtenir la solution particulière de l'équation différentielle qui s'annule ainsi que sa dérivée en 0. On peut simplifier le résultat par exemple à l'aide de `SINCOS EXPA`.

3.5.4 Système linéaire à coefficient constant: LDEC

Soit $y = (y_1, ..., y_n)$ un vecteur de fonctions de x et supposons qu'on souhaite résoudre:

$$y' = Ay + b$$

où A est une matrice constante $n \times n$ et b un vecteur de n fonctions de x. Par exemple pour:

$$A = \begin{pmatrix} 1 & -1 \\ 2 & 4 \end{pmatrix}, \quad b = \begin{pmatrix} 1 \\ e^x \end{pmatrix}$$

il s'agit du système:

$$\left\{ \begin{array}{rcrcrcl} y_1'(x) & = & y_1(x) & - & y_2(x) & + & 1 \\ y_2'(x) & = & 2y_1(x) & + & 4y_2(x) & + & e^x \end{array} \right.$$

avec conditions initiales $y1(0) = y2(0) = 0$. Soit $\mathcal{L}(b)$ le vecteur des n transformées de Laplace des n fonctions de b et $y(0)$ le vecteur de données initiales en $x = 0$. Alors, on a:

$$(pI_n - A)\mathcal{L}(y) = \mathcal{L}(b) + y(0)$$

d'où:

$$\mathcal{L}(y) = (pI_n - A)^{-1}(\mathcal{L}(b) + y(0))$$

Il suffit donc de calculer $(pI_n - A)^{-1}$, faire le produit avec $L(b) + y(0)$ et calculer les transformées de Laplace inverses.

L'instruction `LDEC` est prévue pour faciliter le travail. Tapez le vecteur b puis la matrice A puis `LDEC`. Le programme renvoie $\mathcal{L}^{-1}(pI_n - A)^{-1}v$ au niveau 1, le niveau 2 contient le déterminant factorisé de $(pI_n - A)$ et le niveau 3 contient la comatrice de $pI_n - A$ (où p est le nom de la variable courante): ils permettent le calcul de $\mathcal{L}^{-1}[(pI_n - A)^{-1}y(0)]$.

Reprenons l'exemple ci-dessus. Plaçons b sur la pile: `{ 1 'EXP(X)' }` (ici la variable courante contenue dans `VX` est `'X'`). On place A sur la pile:
`{ { 1 -1 } { 2  4 } }`
Après `LDEC`, on obtient:

$$\left\{ \begin{array}{lcl} y_1 & = & -1/2e^x - 2/3 + 2e^{2x} - 5/6e^{3x} \\ y_2 & = & 1/3 - 2e^{2x} + 5/3e^{3x} \end{array} \right.$$

Au niveau 2, on a $\det(sI_2 - A)$ (avec $s = X$):

$$\det(sI_2 - A) = (2 - s)(3 - s)$$

et au niveau 3 la comatrice de $sI - A$:

$$\begin{pmatrix} -4+s & -1 \\ 2 & -1+s \end{pmatrix}$$

Les niveaux 2 et 3 sont utiles si on cherche une solution avec donnée initiale non-nulle, par exemple $y_1(0) = 1, y_2(0) = 2$. Nous devons ajouter
$\mathcal{L}^{-1}\left[(sI_2 - A)^{-1}y(0)\right]$ à la solution précédente. Pour cela, tapez `{ 1 2 }`, multipliez par la comatrice du niveau 3 puis par l'inverse du déterminant du niveau 2:

$$(sI_2 - A)^{-1}(1,2) = (\frac{-6+s}{6-5s+s^2}, \frac{2s}{6-5s+s^2})$$

il reste à appeler `ILAP`:

$$(4e^{2x} - 3e^{3x}, -4e^{2x} + 6e^{3x})$$

et à additionner au résultat précédent:

$$\left\{ \begin{array}{lcl} y_1 & = & -1/2e^x - 2/3 + 6e^{2x} - 23/6e^{3x} \\ y_2 & = & 1/3 - 6e^{2x} + 23/3e^{3x} \end{array} \right.$$

3.6 Développements de Taylor et limites: SERIES, LIMIT

Pour calculer une limite, entrez l'expression, puis une équation de la forme '*variable=point*' puis tapez `LIMIT`. Par exemple:
`'SIN(X)/X' 'X=0' LIMIT`

Le programme `SERIES` effectue l'équivalent de l'instruction intégrée `TAYLR` mais avec deux extensions: la possibilité de faire un développement en un autre point que $x = 0$ et de faire des développements asymptotiques. L'instruction `LIMIT` est une version de `SERIES` qui ne renvoie que la limite au lieu de renvoyer le développement, l'ordre du reste et la limite.

Rappelons la syntaxe de la commande `TAYLR`: au niveau 3 on entre une expression algébrique (la fonction), au niveau 2 un nom global (la variable) et au niveau 1 un entier positif (l'ordre souhaité). `SERIES` accepte également au niveau 2 une expression du type '*nom global=valeur*' pour effectuer un développement limité ailleurs qu'en 0. Il renvoie au niveau 3 la limite, au niveau 2 une liste et au niveau 1 une équation. Cette équation exprime une variable h qui tend vers 0 au point considéré en fonction de la variable de départ. Cette variable h est utilisée dans la liste du niveau 2 dont le premier élément est le développement limité (ou asymptotique) et le deuxième élément est l'ordre du reste de ce développement

Exemples:

- Déterminer le développement limité de $\sin(x)$ en $x = 0$ à l'ordre 3:
  `'SIN(X)' 'X' 3 SERIES`
  renvoie au niveau 3 la limite 0, au niveau 2 la liste:
  $$\{h(1 - \frac{h^2}{6}) \quad h.h^4\}$$
  et au niveau 1 $h = X$. Ici la petite variable h est égale à X, le développement est $h - h^3/6$ et le reste est un $O(h^5)$.

- Déterminer le développement limité de e^x en $x = 1$ à l'ordre 4:
  `'EXP(X)' 'X=1' SERIES`
  On obtient au niveau 2 la liste:
  $$\{e(1 + h + \frac{1}{2}h^2 + \frac{1}{3}h^3 + \frac{1}{24}h^4) \quad eh^5\}$$
  et au niveau 1 $h = X - 1$ (qui tend bien vers 0 lorsque X tend vers 1).

L'instruction `SERIES` calcule par défaut des développements bidirectionnels à l'ordre 4. On peut spécifier l'ordre et la direction simultanément en plaçant au niveau 1 un réel: son signe indique alors la direction de la limite et sa valeur absolue l'ordre. On peut également indiquer le point limite sous la forme $X = a + 0$ ou $X = a - 0$. Pour une limite bidirectionnelle d'ordre spécifié, on utilise un entier système (`#5d` par exemple pour l'ordre 5).

On peut effectuer des développements asymptotiques en l'infini en indiquant $X = \pm\infty$ comme point limite (pour obtenir $X = +\infty$, entrez $X = \infty$ puis tapez la touche $\boxed{\text{EVAL}}$).

Exemple: déterminez le développement de:
$(\ln(-\ln(x + x^2)) - \ln(-\ln(x))) * \ln(x)/x$ en 0^+
`'( LN(-LN(X+X^2))-LN(-LN(X)) )*LN(X)/X' 'X=0+0' SERIES`

Les instructions `DIVPC` et `TRUNC` ont pour objectif d'aider à comprendre comment calculer un développement limité:

- `DIVPC` effectue la division en puissances croissantes du niveau 3 par le niveau 2 à l'ordre spécifié par un réel au niveau 1
- `TRUNC` tronque un développement limité au niveau 2 en supprimant tous les termes qui peuvent être absorbés par le reste du niveau 1

Remarque 3 *L'ordre demandé à* `Erable` *est un ordre relatif et non absolu. Donc l'ordre renvoyé par* `SERIES` *n'est pas affecté par des simplifiations dans des produits ou des divisions (par exemple pour* $\sin(x)/x$ *en* $x = 0$*). Mais comme en calcul numérique,* `Erable` *ne peut déterminer à l'avance les pertes d'ordre lorsque des cancellations interviennent dans des sommes ou des différences. Par exemple pour* $\sin(x) - x$ *en 0, on perd deux ordres de précision relative, donc pour avoir un DL à l'ordre relatif 3, il faut demander au départ un ordre relatif de 5.*

3.7 SOLV et EXEC

La commande `SOLV` permet de résoudre des équations polynomiales par rapport à une variable ou à une fonction d'un polynôme en une variable. Il renvoie une liste de solutions. On peut par exemple résoudre $\sin(x)^2 - 1 = 0$ de cette manière:
`'SIN(X)^2-1' X SOLV`

Selon le type de l'objet placé au niveau 1, l'instruction `EXEC` permet d'effectuer soit une ou plusieurs substitutions dans une expression, soit un changement de variables, soit d'isoler une variable (comme la commande `ISOL` de la ROM), soit d'exécuter un même programme sur une liste d'objets.

- si le niveau 1 est un nom de variable, on isole cette variable par exemple:
  `'X^2+2*X=3' X EXEC` renvoie $X = 1$ qui est solution de l'équation $x^2 + 2x =$

3. Attention, il peut y avoir d'autres solutions que la solution renvoyée. Utilisez `SOLV` si vous les cherchez.

- changement de variables (si le niveau 1 est une équation):
  `{ 1 'M+2' 'M-3' } 'M=4' EXEC { 1 6 1 }`
  On peut aussi mettre une équation définissant implicitement la nouvelle variable, par exemple `'M^2=16'`. Si l'expression est une intégrale, `EXEC` modifiera les bornes si nécessaire, s'il s'agit d'une équation différentielle `EXEC` effectue aussi les changements de fonction, par exemple:
  `'d1d1Z(X)-Z(X)=0' 'Z(X)=EXP(-X)*Y(X)'`

- Exécution d'un programme sur une liste, si le niveau 1 est un programme. Exemple: pour changer le signe d'une matrice symbolique, on peut utiliser l'instruction intégrée `NEG`:
  `{ { 1 2 } { 3 4 } } << NEG >> EXEC`
  qui équivaut ici à l'instruction `CHS` d'`erable`:
  `{ { 1 2 } { 3 4 } } CHS`

- Plusieurs substitutions, si les niveaux 2 et 1 sont des listes. `EXEC` remplace dans l'expression au niveau 3 les éléments de la liste du niveau 2 par les éléments correspondants de la liste de niveau 1. Exemple: remplacement des fonctions trigonométriques par des exponentielles complexes:

```
'COS(X)+i*SIN(X)'
{ SIN COS }
{ << i * EXP DUP INV - i 2 * / >>
  << i * EXP DUP INV + 2 / >>
}
EXEC EXPA
```

  renvoie `'EXP(i*X)'`.

3.8 Algèbre linéaire

3.8.1 Systèmes linéaires.

Deux commandes permettent de résoudre des systèmes linéaires: `SYST` et `SOLGEN`. Elles prennent en argument une liste contenant les équations et dont le dernier élément est la liste des inconnues. Par exemple pour résoudre le système

$$\left\{ \begin{array}{ccccc} 2x & + & 3y & = & 5 \\ x & - & 5y & = & 4 \end{array} \right.$$

d'inconnues x et y, on entre d'abord:
`{ '2*X+3*Y=5' 'X-5*Y=4' { X Y } }`
puis on appelle `SYST` ou `SOLGEN`. Lorsque le système n'a pas de paramètres, on peut omettre de taper la liste des inconnues, par exemple ici on aurait pu taper:
`{ '2*X+3*Y=5' 'X-5*Y=4' }`

La commande `SYST` renvoie au niveau 1 le système réduit, au niveau 2 la liste des pivots utilisés dans la réduction du système et au niveau 3 le système. La commande `SOLGEN` renvoie au niveau 1 la solution exprimée de manière paramétrique et renvoie les mêmes résultats que `SYST` mais décalés d'un niveau. Les étapes intermédiaires de la réduction sont affichés si drapeau 1 est *set*(donc en tapant `1 SF` on affiche les étapes intermédiaires et en tapant `1 CF` on ne les affiche pas). Dans ce cas, l'utilisateur doit appuyer sur `CONT` (la touche-menu en haut à droite) à chaque étape. Il peut également interrompre la réduction en tapant `HALT` par exemple pour visualiser complètement une grande matrice. La reprise du programme s'effectue alors par shift gauche-ON (commande `CONT`).

La liste des pivots est très utile lorsque le système à résoudre possède des paramètres. En effet, si un pivot s'annule pour une valeur du paramètre, alors la solution renvoyée par `SYST` ou `SOLGEN` n'est pas correcte. Il faut reprendre la résolution du système pour cette valeur particulière du paramètre. Pour faciliter ce travail, `SYST` et `SOLGEN` enregistrent le dernier système résolu contenant des paramètres dans la variable `SYSTEM`. Il suffit donc de rappeler le système sur la pile en appuyant sur la touche `VAR` puis la touche blanche du bandeau correspondant à `SYSTEM` (en général la touche A), taper l'équation '*paramètre=valeur particulière*' puis `EXEC` pour faire le remplacement et appeller à nouveau `SYST` ou `SOLGEN`

Par exemple, prenons le système:

$$\left\{ \begin{array}{cccc} mx & + & y & = & -2 \\ mx & + & (m-1)y & = & 2 \end{array} \right. \tag{3.4}$$

d'inconnues x et y, où m est un paramètre. On tape:
`{ 'M*X+Y=-2' 'M*X+(M-1)*Y=2' { X Y } } SYST`
Au niveau 2, on remarque que la liste des pivots s'annule lorsque $m = 0$ ou $m = 2$ (on peut ici chercher les valeurs de m en tapant `M SOLV`). Donc la solution renvoyée au niveau 1 est correcte lorsque $m \neq 0$ et $m \neq 2$. Pour résoudre le système dans ces deux cas particuliers, on rappelle le système sur la pile (`SYSTEM`) puis on tape:
`'M=0' EXEC SYST`
pour $m = 0$ ou:
`'M=2' EXEC SYST`
pour $m = 2$.

3.8.2 Création de matrices: LCXM, idn, VAND, HILBERT

Si la matrice ne contient pas de paramètres, vous pouvez utiliser *MatrixWriter*, sinon il faut rentrer une liste de listes, en d'autres termes remplacer les crochets [] par des accolades {}: c'est ce que l'on appelle une matrice symbolique. Voici quelques méthodes permettant de créer des matrices symboliques:

1. On tape les délimiteurs { et } et on entre les coefficients un à un. On a vu qu'il est impossible d'éditer une matrice symbolique avec l'application *MatrixWriter* mais les programmes `AIO` ([7]) et `XCELL` ([15]) permettent de créer et d'éditer des matrices symboliques dans un environnement de type tableur, de même que le Metakernel ([16]).

2. Il est aussi possible d'entrer une matrice dont on connait le coefficient a_{ij} en fonction de i et j. Le programme `LCXM` prend en arguments:

   - au niveau 3 et 2 le nombre de lignes et colonnes de la matrice à construire,
   - au niveau 1 un programme RPL qui prend 2 arguments (le numéro de ligne et de colonne) et qui renvoie a_{ij}.

   Par exemple, pour créer une matrice 3×4 de coefficients $a_{ij} = (i - j)^2$, on tapera `3 4` et le programme:
   `<< - SQ >>`
   puis `LCXM`

3. `idn` permet de créer une matrice identité symbolique en plaçant sur la pile, un entier (la taille de la matrice) ou une matrice carrée (dans ce cas la matrice est remplacée par une matrice identité de taille identique).

4. `VAND` crée une matrice de Vandermonde. On place sur la pile la liste des éléments. Par exemple `{ 1 a b } VAND`

5. `HILBERT` crée une matrice de Hilbert de la taille spécifiée. On place en argument le nombre de lignes et de colonnes.

3.8.3 Opérations simples: XY, TR, TRAN

Les opérations arithmétiques usuelles acceptent les matrices et vecteurs comme arguments. Les opérations plus spécifiques sont:

- `XY`: produit scalaire de deux vecteurs
- `TR`: trace d'une matrice ($\sum_{i=1}^{n} a_{ii}$)

- `TRAN`: transposée d'une matrice $(({}^tA)_{ij} = A_{ji})$. Attention, il ne s'agit pas de l'adjointe de la matrice de départ (contrairement à l'instruction intégrée `TRN` qui équivaut à `TRAN conj`).

3.8.4 Réduction des matrices par l'algorithme du pivot partiel: rref, RANG

Pour réduire une matrice, entrez la matrice sur le niveau 1 de la pile puis lancez le programme `rref` ou `RANG`.

`rref` crée des 0 de part et d'autre de la diagonale alors que `RANG` crée des 0 sous la diagonale.

Attention: les programmes `rref` et `RANG` considèrent que la dernière colonne est une colonne de constantes (2ème membre d'un système par exemple), et ne crée pas de 0 dans cette colonne. Si nécessaire, ajoutez une colonne de zéros!

Le pivot choisi est l'élément du reste de la colonne dont le "module" est maximal. Par convention:

- un scalaire nul a comme module -8,
- un scalaire non nul a comme module sa valeur absolue en mode numérique, 1 pour un complexe et 2 pour un réel en mode symbolique,
- un nom de variable a pour module $\frac{-1}{180}$,
- une expression algébrique $-\frac{l}{180}$, où l est le nombre d'éléments de l'algébrique.

En mode symbolique (flag 15 armé), on limite ainsi les cas particuliers et on facilite leur résolution, alors qu'en mode numérique on minimise les erreurs d'arrondi (meilleure stabilité).

A la fin de la réduction, vous voyez au niveau 2 la liste des pivots. Chaque fois qu'un pivot de type symbolique s'annule, il faut reprendre la réduction dans ce cas particulier. Si on sait résoudre l'équation *pivot=0* (par exemple en appelant `SOLV`), on rappele la matrice originale sur la pile (tapez `MATRIX`) puis on tape une équation du type `'<parametre>=valeur'` puis `EXEC` (raccourci α-shift droit-EVAL) et enfin à nouveau `rref` ou `RANG`.

Par exemple, supposons que l'on résolve matriciellement le système 3.4. Pour $m \neq 2$ et $m \neq 0$, il y a une solution unique obtenue en tapant

```
{ {M 1 -2} { M 'M-1' 2 } }
```

puis `rref` qui donne au niveau 2:

```
 Pivots: { 1 'M-2' }
```

et au niveau 1:

`{ { 'M^2-2*M' 0 '-2*M' } { 0 'M-2' 4 } }`.
C'est-à-dire que:

$$x = \frac{-2m}{m^2 - 2m} = \frac{-2}{m-2}, \qquad y = \frac{4}{m-2}.$$

Comme pour les systèmes, il faut regarder pour quelles valeurs du paramètre m la liste des pivots est susceptible de s'annuler. La réduction de la matrice pour une valeur particulière du paramètre s'effectue comme pour les systèmes, mais ici on rappelera le contenu de `MATRIX` au lieu du contenu de `SYSTEM` avant de taper par exemple `'M=2' EXEC`.

3.8.5 Autres applications de la réduction.

On peut également inverser une matrice carrée, calculer le déterminant d'une matrice carrée ou déterminer le noyau et l'image d'une application par l'algorithme de réduction. Les commandes sont les suivantes:

- `INVL`:
  permet d'inverser une matrice carrée placée au niveau 1 de la pile par la méthode du pivot.
- `RDET` calcule le déterminant d'une matrice carrée par réduction. Ce qui permet parfois d'obtenir le déterminant sous forme factorisée, contrairement au programme `det` qui renvoie toujours un résultat non factorisé. Vous pouvez naturellement poursuivre la factorisation en tapant `COLC`.
- `KERN` renvoie les équations paramétriques du noyau d'une application linéaire dont la matrice est au niveau 1.
- `<< TRAN rref TRAN >>` donne une base de l'image d'une application linéaire (on prend les vecteurs colonnes non nuls de la matrice renvoyée).

3.8.6 Diagonalisation.

La commande `JORDAN` permet de diagonaliser une matrice carrée. Elle permet également de calculer la forme normale de Jordan d'une matrice non diagonalisable.

`JORDAN` prend en argument la matrice et renvoie aux niveaux:

- 1: la liste de vecteurs propres taggués par la valeur propre correspondante. Si la valeur propre est multiple, elle est suivie d'une liste de vecteurs caractéristiques ordonnées en cycles de Jordan, chaque cycle se termine par un vecteur propre taggué par le mot "Eigen" (les vecteurs caractéristiques sont taggués par "Char").

- 2: le polynôme minimal,
- 3: le polynôme caractéristique, c'est un multiple du polynôme minimal,
- 4: la liste des racines du polynôme caractéristique (suivies de leur multiplicté)
- 5: l'inverse de A
- 6: le déterminant de A si A est d'ordre pair ou son opposé si l'ordre est impair,

Exemple:

$$A = \begin{pmatrix} 1 & -1 & 0 \\ 0 & 1 & -1 \\ -1 & 0 & 1 \end{pmatrix}$$

le programme renvoie:

```
6: 0
5: { { infini infini infini }
     { infini infini infini }
     { infini infini infini } }
4: {0 1 '3/2+i/2*sqrt(3)' 1 '3/2-i/2*sqrt(3)' 1 }
3: 'X^3-3*X^2+3*X'
2: 'X^3-3*X^2+3*X'
1: { :0: {1 1 1}
     :'3/2+i/2*sqrt(3)': {1 '-1/2-i/2*sqrt(3)' '-1/2+i/2*sqrt(3)'}
     :'3/2-i/2*sqrt(3)': {1 '-1/2+i/2*sqrt(3)' '-1/2-i/2*sqrt(3)'}
   }
```

Donc A possède 3 valeurs propres de mulitplicité 1: 0 et $\frac{3\pm\sqrt{3}i}{2}$. La base de vecteurs propres coorespondante est:

$$\{\ (1,1,1), (1, \frac{-1 \mp i\sqrt{3}}{2}, \frac{-1 \pm i\sqrt{3}}{2})\ \}$$

correspondants à $0, (3+\sqrt{3}i)/2, (3-\sqrt{3}i)/2$. Le polynôme minimal et le polynôme caractéristique sont ici identiques (c'est le cas générique) et valent X^3-3X^2+3X. La matrice n'est pas inversible, son déterminant au niveau 6 est nul.

3.8.7 Autres commandes.

- Déterminant: `det`
  Pour obtenir le déterminant d'une matrice, tapez la matrice puis `det`. L'algorithme appliqué ici est le développement récursif par rapport à une colonne. Il y a donc trois commandes pour calculer le déterminant d'une matrice: `det`, `RDET` et `MAD` (qui renvoie le déterminant au niveau 4).
  Exemple, testez les 3 méthodes pour la matrice:
  `{ { 1 1 A } { 1 A 1 } { A 1 1 } }`

- Polynôme caractéristique: `PCAR` et `MAD`
  Vous pouvez calculer le polynôme caractéristique d'une matrice en tapant `PCAR` ou `MAD`.

- Produit vectoriel: `cross`
  Le programme `cross` effectue le produit vectoriel de deux vecteurs symboliques ou numériques de $\mathbb{C}^3$,

- Formes quadratiques:
  `QXA` et `AXQ` permettent de passer de la représentation algébrique à la représentation matricielle d'une forme quadratique. Le programme `GAUSS` du répertoire `other` permet de réduire une forme quadratique sous forme de somme de carrés indépendants. Pour plus de détails, veuillez lire la documentation en anglais.

Annexe A

Références et remerciements

A.1 Remerciements

Je remercie tout d'abord Claude-Nicolas Fiechter et Mika Heiskanen, les auteurs du logiciel `ALG48` de m'avoir laissé utiliser une partie de leur code source dans `Erable` et de m'avoir autorisé à diffuser une version modifiée de `EQSTK` ([8]) pour afficher correctement les objets mathématiques. Je remercie plus spécialement Mika pour `JAZZ` ([12]) et son débuggueur RPL-système `SDB`.

Je remercie aussi:

- les développeurs du système Linux ([21]) et des utilitaires GNU ([10]), en particulier R. Stallman pour `emacs`, M. Mikocevic pour les GNU-tools ([17]), Eddie C. Dost pour `x48`, l'émulateur HP48 sous Unix/X11.
- D. Knuth et la communauté TEX pour le logiciel [14] qui a permis la mise en page de ce document,
- Jean-Yves Avenard [2] pour l'éditeur `MINWRITER`, André Schoorl pour `UFL` et `JAVA` ([20]), les 5 membres du MDG pour le Metakernel ([16])
- Gilles Virone de m'avoir montré ce qu'était capable de faire une HP28/48 et de m'avoir prêté son Pentium durant l'été 1995,
- Renée De Graeve qui a rédigé une partie de ce manuel l'a longuement relu et a testé de nombreux exemples,
- HP Corvallis et W. Wickes pour `TOOLS.EXE` ([3]), D. Müller et R. Hellstern pour `RPL48` ([18]), P. Courbis et S. Lalande pour [4],

- les administrateurs des sites `ftp` anonymes, en particulier ceux de `ftp.funet.fi`, `ftp.cis.com`, `hplyot.obspm.fr`, `hpcvbbs.cv.hp.com` et `wuarchive.wustl.edu`,
- J.M. Ferrard pour la source d'inspiration qu'a été son livre [6],
- les utilisateurs d'`ALGB` puis d'`Erable` sur le réseau (en particulier Maurice Al-Khaliedy, Craig Clifford, David Czinczenheim, Eduardo, Eric Gorka, Scott Guth, John Wilson, ...)
- mes (ex-)étudiants de l'Université de Grenoble, plus particulièrement Christophe Burdin, Jérôme Coss, Ludovic Dumaine, Frédéric Hermann, Stéphane Monboisset, Lionel Pilot, Éric Saya, Quan Tong Duc, Samy Venin...

A.2 Référence des commandes.

Liste des symboles utilisés dans les tableaux:

- %: réel
- C%: complexe
- n: entier
- [ ]: matrice ou vecteur numérique
- { l }: liste
- { m }: matrice ou vecteur symbolique
- p: polynome ({ p } pour un polynome liste),
- { v }: liste de variables
- s: objet symbolique
- v: “noyau” (nom de variable ou expression irrationnelle)
- f: fraction
- N, D: numérateur et dénominateur d’une fraction
- o: objet

Fonctions de la librairie `Erable`:

Nom	Explication	Arguments	Arguments renvoyés
ABCUV	Bezout $ax + by = c$	3,2,1:a, b, c	1:1 [3,2: x, y] ou 1: 0
AXL	tableau ↔ liste	[ ] ou { m }	{ m } ou [ ]
AXQ	matrice s forme quadratique	{ m }	s
		2: { m }, 1: { v }	s
C2P	Cycles → permutation	{ cycles }	p
CHS	Change le signe	o	$-o$
COLC	Factorisation	s	s
CIRC	Compose 2 permutations	2:p_2, 1:p_1	$p_2 \circ p_1$
CURL	Rotationnel	2: { s_1 s_2 s_3 } 1: { v }	{ s'_1 s'_2 s'_3 }
DIV	Divergence	2: { s_1 ... s_k } 1: { v }	s
DIV1	Division	2: o_2, 1: o_1	o_2/o_1
DIV2	Division euclidienne	2: o_2, 1: o_1	2: o_2 div o_1,1: o_2 mod o_1
DIVIS	Liste des diviseurs	o	{ l }
DIVPC	Division en puiss. croissante	3: s, 2: s', 1: n	s
DSOLVE	Résoudre $y'(x) = f(y(x), x)$	$f(y(x), x)$	$y(x)$
EPSX0	$\lvert . \rvert < \varepsilon \to 0$	o	o
EULER	Indicatrice d'Euler	n	$\varphi(n)$
EXEC	Substitution ou *doall*	2: { l }, 1: programme	1: { l }
		2: s, 1: $o_1 = o_2$	s
		3: s, 2: { l1 } 1: { l2 }	s
EXPA	Simplification	o	o'
EXPLN	Conversion en $\exp, \ln$	s	s
FACTO	Factorisation	o	3: { v } 2: f , 1: { $f_1 n_1$ ... }
FCOEF	racines/poles → Fraction	{ $r_1 n_1 r_2 n_2$ ... }	{ f }
FROOTS	Factorisation	o	3: { v } 2: f , 1: { $s_1 n_1$ ... }
FSIGN	Signe d'une frac. rationnelle	s	liste tagguée
FXND	Décompose une fraction	$f = N/D$	2: N, 1: D
GCD1	Plus grand commun diviseur	2: o_2, 1: o_1	GCD(o_2,o_1)
GCD3	PGCD étendu ($au + bv = d$)	2,1: a , b	GCD$(a, b) = d$, u, v
HALFTAN	En fonct. de $\tan(x/2)$	s	s
HERMITE	Polynôme de Hermite	integer n	H_n
HESS	Matrice hessienne	2: s, 1: { v }	matrix
HILB	Matrice de Hilbert	entier r	$r \times r$ matrice
HORN	Schéma de Horner	2:p , 1: r	3: $p/(X - r)$, 2: r, 1: $P(r)$
ILAP	Laplace inverse	s	$L^{-1}(s)$
INVL	Inversion	o	o^{-1}
IPP	Intégration par parties	$\int_a^b u'vdx$, u	$\int_a^b u'v\ dx = [uv]_a^b - \int_a^b uv'\ dx$
JORDAN	Diagonalisation	endomorphisme	7 à 1: *cf.* section algèbre linéaire
KERN	Noyau d'une appli. lin.	1: { m }	comme `SOLGEN`
LAP	Transformée de Laplace	2:f, 1:g	$L(f)/g$
LAPL	Laplacien	2: f, 1: { v }	Δf
LCM1	Plus petit multiple commun	2: o_2, 1: o_1	LCM(o_2,o_1)
LCXM	Création de matrice	3: r, 2: c, 1: prog	1: $r \times c$ matrice
LDEC	Équ. diff. lin. coeff. cst	2: { m }, 1: { v }	3,2: $(m - x)^{-1}$, 1: $(m - x)^{-1}v$
LEGENDRE	(Polynomes de)	entier n	L_n
LGCD	PGCD d'une liste	{ l }	o=PGCD
LIDNT	Liste des variables	s	2: s, 1: { v }
LIMIT	Limite	2:$f(x)$, 1:$x = a$	$\lim_{x \to a} f(x)$
LNCOLC	Log Collect	s	s
LU2	décomposition LU	M	2: $L^{-}1$, 1: U
LVAR	liste de variables	o	{ v }

Fonctions de la librairie `Erable` (suite):

Nom	Explication	Arguments	Arguments renvoyés
MAD	det, inverse, poly car	o	4: det, 3: $1/o$, 2: { p },1: p
MMULT	produit spécial	3: o_2, o_1, n	"$o_2 \times o_1$"
MULT	produit	2: o_2, o_1	$o_2 \times o_1$
NDXF	crée une fraction	2: N, 1: D	$f = N/D$
ORND	Arrondit un objet	2: o, 1: D	o
P2C	Permutation → cycles	p	
PCAR	Polynome caractéristique	endomorphism	s
PF	Déc. en éléments simples	f	$\sum_i f_i$
PFEXEX	`EXEC` entre + et −	2:$\sum_i f_i$, 1:prog	$\sum_i \text{prog}(f_i)$
POWER	Puissance	2: o, 1: n	o^n
PREVAL	Évaluation d'une primitive	3: $f(x)$, 2: b, 1:a	$f(b) - f(a)$
PTAYL	Taylor pour des polynomes	2: $P(X)$, 1: o	$P(X - o)$
QXA	forme quadratique → matrice	2: s, 1: { v }	{ m }
		s	{ m }
RANG	Réduction sous-diagonale	{ m }	{ m }
RDET	Déterminant (rref)	endomorphisme	déterminant
RISCH	Integration symbolique	s	s
SCROLL	Défilement d'un grob	grob	
SERIES	Taylor	3: s, 2: v, 1: n	3: n, 2: { p }, 1: s
SIMP2	Simplification	2: o_2, 1: o_1	2: o_2', 1: o_1'
SINCOS	$e^{ix} \to \cos(x) + i\sin(x)$	s	s
SOLGEN	Résolution de système	1:{ équations { v } }	
SOLV	Solve symbolique	2: s, 1: x	2: x, 1: solutions
SQRT	Racine carrée	n or C% or s	n or C% or s
STUDMULT	× de matrices "étudiant"	M, M'	"$M\dot{M}'$ "
SUBT	Soustraction	2: o_2, 1: o_1	$o_2 - o_1$
SYST	Résolution de système	1:{ équations { v } }	
TAN2SC	$\tan(x) \to \sin(x)/\cos(x)$		
TAN2SC2	$\tan(x) \to (1 - \cos(2x))/\sin(2x)$		
TCHEB	(Polynômes de)	entier n:	T_n
TEXPA	Développe les fonct. transc.	s	s
TR	trace	{ m }$= (a_{ij})_{1\leq i,j\leq n}$	$\sum_{i=1}^n a_{ii}$
TRAN	transposée	[ ] or { m }	[ ] ou { m }
TRIG	Trigo.: → $\sin, \cos, \arctan$	s	s
TRIGCOS	$\sin^2 \to 1 - \cos^2$	s	s
TRIGLIN	Linéarisation trigo.	s	s
TRIGSIN	$\sin^2 \to 1 - \cos^2$	s	s
TSIMP	Simplification (transcendentale)	s	s
VAND	matrice de Vandermonde	liste	matrice
VER	Version	rien	% 2.99
XFRC	Amélioration de →Q	o	o
XNUM	→ numérique	o	o
XQ	→ exact	o	o
XY	produit scalaire de 2 vecteurs	2: x 1: y	$x.y$
abs	Valeur absolue	s	s
add	Addition	2: o_2, 1: o_1	$o_2 + o_1$
arg	Argument	1: s	1: s
comb	Combinaisons	2: n, 1: n'	$C_n^{n'}$
conj	Conjugué	o	$\bar{o}$
cross	Produit vectoriel	2: x, 1: y	$x \wedge y$
der	derivée	2: s, 1: v	1: s
der1	derivée/var. courante	s	s
det	Déterminant (expand)	endomorphisme	déterminant
fact	Factorielle	n	$n!$
idn	identité	entier ou matrice	matrice identité
im	partie imaginaire	o	$\Im(o)$
re	partie réelle	o	$\Re(o)$
rref	Réduction (Gauß)	M	{ s }, matrice réduite

Fonctions de la librairie `kernel` (n désigne l'entier contenu dans la variable `MODULO`):

Nom	Fonction	Arguments	Arguments renvoyés
`{kernel.lib}`	`(0:787)`		
MODADD	addition modulaire	3: n_1, 2:n_2, 1: n	$(n_1 + n_2) \bmod n$
MODSUBT	soustraction modulaire	3: n_1, 2:n_2, 1: n	$(n_1 - n_2) \bmod n$
MODMULT	multiplication modulaire	3: n_1, 2:n_2, 1: n	$(n_1 * n_2) \bmod n$
MODDIV	Division modulaire	3: n_1, 2:n_2, 1: n	$(n_1/n_2) \bmod n$
MODPOW	Puissance modulaire	3: n_1, 2:n_2, 1: n	$n_1^{n_2} \bmod n$
MODINV	Inversion modulaire	2: n_1, 1: n	$n_1^{-1} \bmod n$
PA2B2	Résoudre $n = a^2 + b^2$	n	$a + ib$

A.3 Référence de l'interface.

A.3.1 Le menu principal

Accessible par la touche `PRG`. Dans les tableaux ci-dessous, d désigne le shift droit et g le shift gauche.

0. EDITORS	**1. VAR I/O**	**2. PROGRAMS**
1. EDIT LEVEL 1 (↓) 2. VIEW LEVEL 1 (g ↓) 3. EDIT STACK (↑) 4. SPEC CHARS (d PRG) 5. NEW EQU (g ENTER) 6. NEW TEXT 7. NEW MAT[] (d ENTER) 8. NEW LIST 9. NEW PROGRAM A. PICTURE 0. BACK	1. VAR. (d VAR) 2. SEND—GET (d 1) 3. PORTS (g 2) 0. BACK	1. USER PROG (α d PRG) 2. LIBRARIES (d 2) 0. BACK

3.PHYSICS	**4.SETUP** (α d CST)	**5. STATISTICS** (d 5)
1. EQ LIBRARY (d 3) 2. CONSTANTS (g 3) 3. DATE/TIME (g 4) 0. BACK	1. CALC MODES (d CST) 2. RESET ERABLE 3. REAL MODE 4. COMPLEX MODE 5. INTEGER ARIT 6. POLYN. ARIT 7. NUMERIC MODE 8. SYMBOLIC MODE 9. TIME SETUP 0. BACK	Single-var Frequencies Fit data Summary stats

6. UNITS (d 6) : bandeau unités.

7.NUMERIC	**8. GRAPH**	**9.ERABLE** (MTH)
1. MATH FNC (α d MTH) 2. SOLVE EQN (d 7) 0. BACK	1. PLOT MENU (d 8) 2. PLOT LEVEL 1 0.BACK	1.BASE ALGEBRA (α d 1) 2. COMPLEX (α d 2) 3. TRIGO. (α d 3) 4. MATRICES (α d 4) 5. CONVERSION (α d 5) 6. ARITHMETIC (α d 6) 7. ERABLE SOLV (α d 7) 8. EXP AND LN (α d 8) 9. DIFF CALC. (α d 9) 0. MAIN MENU

A.3.2 Le menu d'`Erable`.

Pour lancer ce menu, taper la touche `MTH`. Chaque rubrique propose un bandeau contenant des commandes d'`Erable`. Lorsqu'un raccourci clavier existe, il est indiqué entre parenthèses, d signifie shift droit et g signifie shift gauche.

1.BASE ALGEBRA (α d 1)	**2.COMPLEX** (α d 2)	**3.TRIGO.** (α d 3)
EXPA (α d SPC)	re	TEXPA
COLC	im	TRIGLIN
der1 (α d SIN)	conj	SINCOS
RISCH (α d COS)	abs	TRIG
LIMIT	arg	TRIGSIN
TAYLR	i (CST)	TRIGCOS
EXEC (α d EVAL)		TAN2SC2
SOLV		TAN2SC
IPP		
FSIGN		

4.MATRICES (α d EVAL)	**5.CONVERSION** (α d 5)	**6.ARITH.** (α d 6)
rref	AXL	DIV2
JORDAN	EXEC	GCD1
det	SINCOS	GCD3
PCAR	EXPLN	ABCUV
SYST	FXND	LCM1
SOLGEN	NDXF	PF
RDET	AXQ	COLC
RANG	QXA	DIVIS
idn		SIMP2
LCXM		EULER
HILBERT		fact
VAND		comb

7.ERABLE SOLV (α d 7)	**8.EXP & LN** (α d 8)	**9. DIFF CALC** (α d 9)
FROOTS	EXPLIN	der
FCOEF	EXPLN	RISCH
SOLV	LNCOLC	IPP
LNCOLC	TEXPA	SERIES
COLC	TEXPA	LIMIT
EXEC	TSIMP	DSOLV
		LDEC
		LAP
		ILAP

Bibliographie

[1] Adobe. `Acrobat Reader 3`. `http://www.adobe.com`, 1997.

[2] J.-Y. Avenard. `miniwriter`. `http://www.epita.fr/~avenar_j`, 1997.

[3] H. P. Corvallis. `TOOLS.EXE`. `hpcvbbs.cv.hp.com ftp.funet.fi wuarchive.wustl.edu`, 1991.

[4] P. Courbis and S. Lalande. *Voyage au centre de la HP 48 S/SX*. Angkor, 1993.

[5] J. Davenport, Y. Siret, and E. Tournier. *Calcul formel: Systèmes et algorithmes de manipulations algébriques*. Masson, 1989.

[6] J. Ferrard. *"Mathez" la HP 48 G/GX*. D31 Diffusion (HP48), 1993.

[7] C. N. Fiechter. `AIO48`. `http://www.hut.fi/~fiechter`, 1997.

[8] C. N. Fiechter and M. Heiskanen. `EQSTK92.ZIP`. `http://www.hut.fi/~mheiskan`, 1997.

[9] C. N. Fiechter and M. Heiskanen. `ALG48V42.ZIP`. `http://www.cs.pitt.edu/~fiechter/hp48` `http://www.hut.fi/~mheiskan`, 1998.

[10] Free Software Foundation. *The GNU project*. `http://www.fsf.org`, 1998.

[11] B. Fuchssteiner. `MuPAD`. `ftp://ftp.inria.fr/lang/MuPAD` `http://www.mupad.de`, 1998.

[12] M. Heiskanen. `JAZZV65.ZIP`. `http://www.hut.fi/~mheiskan`, 1996.

[13] Kermit. `kermit`. `ftp://kermit.columbia.edu`, 1998.

[14] Knuth & al. *TEX& LATEX*. `http://www.tug.org/teTeX/`, -1998.

[15] D. Lopez. XCELL. ftp://ftp.cis.com/pub/hp48g/uploads/xcell2.zip, 1998.

[16] Maubert Development Group. Metakernel, 48+. http://www.epita.fr/~avenar_j, 1998.

[17] M. Mikocevic. GNUTOOLS. ftp://srcm1.zems.fer.hr:/pub/hp48/tools2.1.9.zip, 1995.

[18] D. Müeller and R. Hellstern. RPL48V20.ZIP. ftp.funet.fi cbs.cis.com hplyot.obspm.fr, 1993.

[19] B. Parisse. Erable. http://www-fourier.ujf-grenoble.fr/~parisse http://perso.wanadoo.fr/bernard.parisse ftp://fourier.ujf-grenoble.fr/pub/hp48/erable.zip, 1998.

[20] R. Steventon, A. Schoorl, and W. Laughlin. JAVA34.ZIP. ftp://ftp.cis.com/pub/hp48g/uploads/java34.zip http://www.engr.uvic.ca/~aschoorl, 1998.

[21] L. Torvald. *Linux*. http://www.linux.org, 1998.

Table des matières