

THESIS / THÈSE

MASTER EN SCIENCES INFORMATIQUES

Conception et réalisation d'un logiciel de calcul et de tracé de trajectoires aéronautiques à partir d'instructions de vol SID

Canart, Jean-Yves

Award date:
1985

Awarding institution:
Université de Namur

[Link to publication](#)

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal ?

Take down policy

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Facultés universitaires

Notre-Dame de la Paix

Namur

Année académique 1984-1985

Conception et réalisation
d'un logiciel de calcul et de tracé
de trajectoires aéronautiques
à partir d'instructions de vol SID

Jean-Yves CANART

Mémoire présenté en septembre 1985
en vue de l'obtention du titre de
Licencié et Maître en informatique

Je voudrais remercier ici mon promoteur,
Mr Jean-Paul Leclercq pour les nombreux
conseils qu'il a bien voulu me prodiguer.
Merci aussi pour l'humour et la gentil-
lesse qui ne lui firent jamais défaut.

Merci également à Mr Hoang Hoa Dung
pour les routines graphiques qu'il m'a
aimablement permis d'utiliser.

PLAN DU MEMOIRE

I.	Introduction	I
II.	Spécifications du programme	4
I.	Input	4
-	décollage	7
-	mouvement droit	7
-	cap	7
-	point	8
-	radiale	10
-	sturn	13
2.	Output	15
III.	Conception du programme SIDCAN	17
	Schéma général	17
I.	Traduction	19
I.I.	Lecture et analyse	20
I.I.I.	Analyseur lexical	21
I.I.2.	Analyseur syntaxique	24
I.I.3.	Méthodes d'analyse	25
I.I.3.I.	Méthode directe	25
I.I.3.2.	Méthode des masques	25
I.I.3.3.	Comparaison	28
I.2.	Traduction	30
I.2.I.	Principes généraux	30
I.2.I.I.	Stratégie générale de traduction	31
I.2.I.2.	Localisation des points dans l'espace	32
I.2.I.3.	Représentation des droites et des angles	33
I.2.I.4.	Unités et constantes	34

I.2.I.5. Conversion coord. géogr.→ coord. cart.	35
I.2.2. Présentation des routines spécialisées	39
I.2.2.1. Envol	39
I.2.2.2. Droite	39
I.2.2.3. Cap	40
I.2.2.4. Radiale	42
I.2.2.5. Sturn	47
I.2.2.6. Point	51
2. Plotting	60
IV. Présentation des principales routines	61
I. Sidcan	61
2. Traduction	61
3. Initialisations	62
4. Lirephrase	65
5. Liremot	66
6. Analysephrase	67
7. Traducphrase	67
8. Straight	69
9. Radiale	70
10. Survol	72
11. Sturn	74
12. Cap	75
13. Savecurve	76
14. Savestraight	77
15. Plotting	79
V. Evolution du produit	84
VI. Tests et conclusion	89

I N T R O D U C T I O N

La C.E.E dispose d'un programme informatique dont le but est de calculer les courbes d'exposition au bruit des avions autour des aéroports. Ce programme est appelé CANAR : Consequences of Aircraft Noise Abatement Regulation.

Il utilise d'une part les données relatives aux caractéristiques des avions telles que le bruit dégagé, les vitesses et performances, d'autre part de données spécifiques à chaque aéroport :

- données géographiques : Longitude, Latitude, Déclinaison magnétique
- d'implantation : position et longueur des pistes, altitude...

Pour établir les courbes de bruit, le programme doit disposer des trajectoires suivies par les avions en phase de décollage ou d'atterrissage. Ces trajectoires sont fournies au programme sous forme codée dans un formalisme précis.

Ces mêmes trajectoires (appelées SID) sont définies de manière précise par les autorités de l'aviation civile et publiées dans les documents aéronautiques. Y figurent également les principaux points de repère et aides à la navigation disponibles aux alentours des aéroports et facilitant les manoeuvres de décollage. Citons notamment :

- Radio-Balises
- VOR (Visual Omni Range)
- DME
- Middle Marker
- ...

La traduction manuscrite d'une trajectoire en sa forme codée utilisable par le programme CANAR s'avère une tâche particulièrement fastidieuse et sujette à erreurs.

Pour remédier à cette situation, il peut être intéressant de concevoir un programme utilitaire se chargeant de cette traduction.

Ce programme utilitaire constitue l'objectif de mon Mémoire.

Un tel programme existe déjà : il s'appelle SIDCAN et est opérationnel sur le DEC 2060 des Facultés de Namur.

Il possède cependant quelques lacunes qui rendent son utilisation parfois problématique :

1. Les trajectoires de vol admissibles par SIDCAN ne sont jamais définies avec précision.
2. La sémantique attachée à une trajectoire particulière est parfois ambiguë.
3. Quels sont les mouvements de base autorisés et refusés par Sidcan ?
4. L'exécution même de SIDCAN aboutit parfois à des résultats aberrants. Il semble bien que certains cas pourtant valides au point de vue syntaxique ne soient pas envisagés par SIDCAN et soient donc traités erronément.
5. L'extension éventuelle de SIDCAN est difficile. Le programme est parfois insuffisamment documenté. Un manuel de programmation et/ou d'utilisation serait le bienvenu .

J'essaierai donc, dans le cadre de ce mémoire, de rebâtir un programme cohérent.

J'insisterai dans mon étude sur les points suivants :

1. Adopter une syntaxe et une sémantique stricte.

J'utiliserai à cet effet le méta-langage BNF qui permet un formalisme rigoureux nécessaire à la définition précise et à la levée de toute ambiguïté syntaxique.

2. Obtenir un programme opérationnel détectant et traitant tous, si pas la plupart, des cas d'exceptions.

3. Adopter une découpe de programme et une stratégie de travail permettant une extension éventuelle du produit.

S P E C I F I C A T I O N S D U P R O G R A M M E

Le programme utilitaire SIDCAN assure deux tâches :

D'une part, la traduction d'un texte-SID en une suite de segments constitutifs d'une trajectoire sous forme normalisée et imposée par le programme CANAR,
d'autre part, la visualisation de cette trajectoire sur un écran de terminal.

I. INPUT

Les données "source" de la traduction sont groupées dans quelques fichiers que consultent le programme :

- les caractéristiques de l'aéroport concerné.
- les coordonnées et caractéristiques des aides à la navigation utilisées par le SID traduit.
- le texte-SID à traduire.

II. OUTPUT

La sortie du programme est constituée par deux fichiers :

- le premier constitue la sortie "officielle" du programme et est tel quel utilisable par le programme CANAR.
- Le second contient un certain nombre de paramètres utiles à l'élaboration de la sortie graphique. Sa syntaxe est plus libre puisque son utilisation est en fait interne au programme.

I. INPUT

Le texte-SID : syntaxe et sémantique

Les éléments d'un texte-SID sont constitués

- de mots-clé : AHEAD, TURN, INTERCEPT, CLIMB ...
indiquant la nature du mouvement de base à effectuer.
- de nombres indiquant des distances, des caps (magnétiques),
des altitudes...
- d'identificateurs : points de repère, radio-balises...
- de mots de liaison : assurent au texte-SID une certaine
cohésion et une tournure de phrase anglaise : AT, UNTIL ...

Tous ces éléments mis ensemble forment des phrases : les "textes-SID".

ex. AHEAD UNTIL 1.5 NM THEN RIGHT TO INTERCEPT TR 085 M TO
SEVENNOAKS THEN LEFT ONTO SEDOI5R TO HORNCHURCH

Pour assurer un traitement informatique rigoureux, il convient de fixer une syntaxe précise, notamment les mots admis ou refusés : c'est l'aspect syntaxique.

D'autre part, il faut convenir d'une représentation du texte sous forme d'une suite de caractères : c'est l'aspect lexical de l'analyse. Enfin, il faut convenir d'une signification bien précise de chaque phrase traduite, c'est l'aspect sémantique de l'analyse.

Ces aspects sont développés en page 19

La syntaxe du langage SID figure p 22 & 24

Dans les trajectoires SID sont susceptibles de figurer un certain nombre de mouvements de base.

Dans le cadre du programme SIDCAN, on en retiendra six. Les autres présentent moins d'intérêt pour le programme CANAR et ne seront donc pas envisagés. Il est à noter que ces six mouvements de base peuvent être représentés par un certain nombre de phrases-SID sémantiquement équivalentes. Il conviendra donc de définir pour chaque mouvement quelle syntaxe adopter.

Conventions :

Dans la présentation des mouvements de base qui suit, "A" représente la position de l'avion avant d'effectuer la trajectoire spécifiée par le texte-SID indiqué.

I. DECOLLAGE

Syntaxe : AHEAD UNTIL <NOMBRE> NM

L'avion quitte la piste et garde son cap sur une certaine distance
(à compter à partir du début de la piste)

Cette distance peut s'exprimer soit par un nombre, soit par un
point de repère à survoler.

On conviendra de ne garder que le premier cas.

Chaque nombre est suivie de son unité. On n'admet comme unité de
mesure de distance que le NM (mille nautique = 1.85318 km).

2. MOUVEMENT DROIT

Syntaxe : AHEAD UNTIL <NOMBRE> NM

L'avion conserve son cap sur une certaine distance et se dirige
vers un point donné. Le point est défini soit par un point de
repère, soit par une certaine distance à parcourir depuis A.

On ne retiendra que le second cas.

On remarquera que la syntaxe du mouvement droit est la même que
la syntaxe du décollage. Il est cependant aisé de distinguer les
deux mouvements :

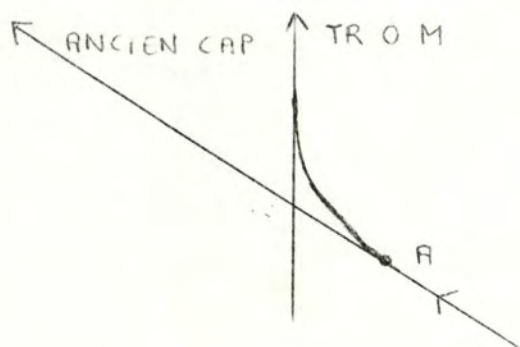
le décollage est toujours le premier mouvement d'une trajectoire.
La première phrase du texte-SID doit donc toujours être le mouve-
ment DECOLLAGE, toutes les autres phrases du texte répondant à la
même syntaxe étant des mouvements droits.

3. CAP

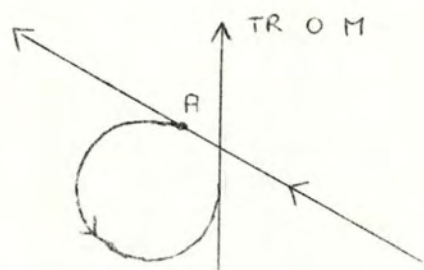
Syntaxe : {TURN} <DIRECTION> ONTO TR <NOMBRE> M

Il est à remarquer que le texte-SID doit obligatoirement mentionner la direction à prendre pour négocier le virage.

Ex 1. TURN RIGHT ONTO TR O M



Ex 2. LEFT ONTO TR O M



On peut remarquer que le sens adopté ne correspond pas toujours au sens "normal" du virage (l'angle le plus petit). Ceci est du notamment au fait que l'avion doit parfois éviter certaines zones (par ex. résidentielles), ce qui le contraint à effectuer le grand virage.

4. POINT

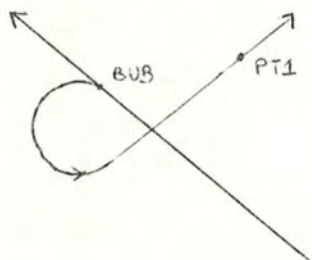
Syntaxe : {AT <NAVAID>} {TURN} <DIRECTION> TO <NAVAID>

L'avion vire dans la direction donnée pour rejoindre un point donné. Il n'y a pas d'indication de cap à prendre.

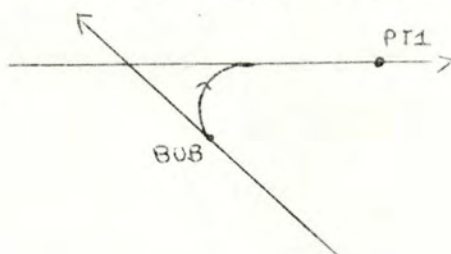
Le texte-SID permet deux options :

1. L'avion est contraint de commencer son virage en A (at navaid)
 2. L'avion peut commencer son virage plus tôt ou plus tard.
- (Ceci peut arriver dans le cas où l'avion ne sait pas atteindre le point étant donné le rayon minimal auquel il est contraint)

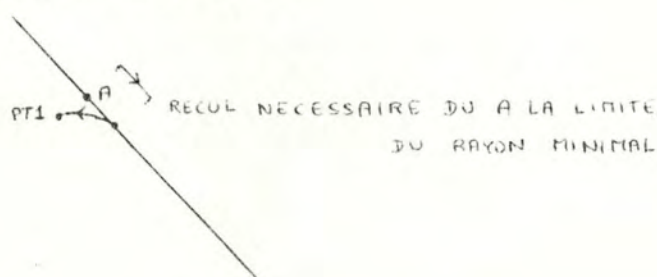
Ex 1. AT BUB TURN LEFT TO PTI



Ex 2. AT BUB TURN RIGHT TO PTI



Ex 3. TURN LEFT TO PTI



Remarques :

1. L'avion peut ne pas pouvoir commencer son virage en A (rayon minimum du virage). Dans ce cas, il commencera son virage soit plus tôt, soit plus tard (si toutefois cela lui est permis par l'option {AT NAVAID})
2. L'avion peut être contraint de tourner à gauche pour rejoindre un point situé à sa droite ou vice-versa, ceci est dû aux contraintes locales (réglementations anti-bruits, pylone ...).

5. RADIALE

Syntaxe : {AT<NAVAID>} [TURN] <DIRECTION> ONTO TR<NOMBRE> M TO<NAVAID>

L'avion doit rejoindre un point tout en adoptant un nouveau cap. Pour cela, il doit s'aligner sur la radiale de ce point. (la radiale DEG d'un point P est la droite passant par P et faisant un angle de DEG degrés avec le nord (magnétique))

L'avion doit virer dans le sens indiqué par le texte-SID.

Selon l'option {AT NAVAID}, il peut soit commencer son virage en A, soit entamer le virage plus tôt ou plus tard de manière à adopter le rayon minimal.

Hyp. DI radiale d'origine

D2 radiale à atteindre

DI et D2 sécantes

5.I. Il y a une contrainte sur le point de départ

5.I.I. le point d'intersection entre les droites DI et D2
se trouve devant l'avion

L'avion commence son virage en A dans le sens indiqué de manière à rejoindre la radiale.

Conditions : 1. Le sens indiqué par le texte-SID doit être "le bon" (angle du virage $< 180^\circ$)

2. L'avion ne doit pas être "trop près" de la radiale (ce qui contraindrait l'avion à adopter un rayon inférieur au rayon min.)

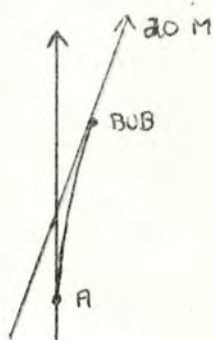
3. Le point à survoler doit être "bien situé"
sur la radiale (lorsque l'avion a rejoint la
radiale, le point à survoler doit être devant lui)

Ex.1. RIGHT ONTO TR 20 M TO BUB

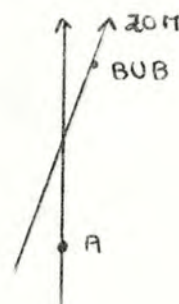
Ex.2. LEFT ONTO TR 20 M TO BUB (erreur 1)

Ex.3. RIGHT ONTO TR 20 M TO BUB (erreur 2)

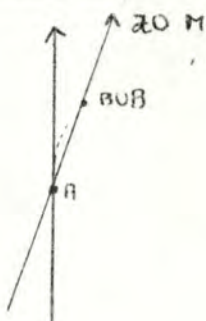
Ex.4. LEFT ONTO TR 230 M TO BUB (erreur 3)



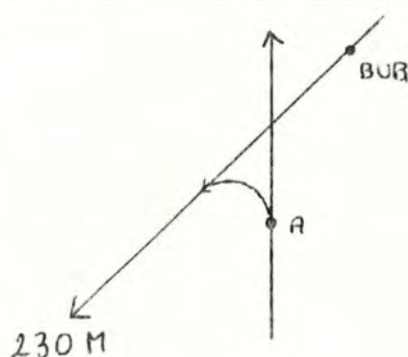
EX 1



EX 2



EX 3

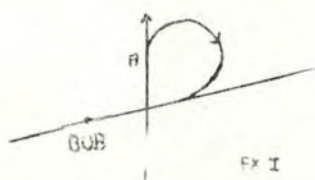


EX 4

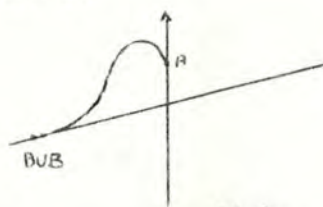
5.1.2. Le point d'intersection se trouve derrière l'avion

Deux cas sont possibles :

1. Le sens indiqué par le texte-SID est contraire au sens "normal" du virage : l'avion effectue alors un virage dans le sens indiqué pour s'orienter sur le bon cap (ex 1)
2. Le sens indiqué par le texte-SID est identique au sens "normal" du virage : l'avion est contraint d'effectuer le mouvement de base STURN (ex 2)



EX 1



EX 2

Conditions : 1. L'avion ne doit pas être "trop près" de la radiale.

2. Le point à survoler doit être "bien situé" sur la radiale.

5.2. Il n'y a pas de contrainte sur le point de départ

5.2.I. Le point d'intersection se trouve devant l'avion

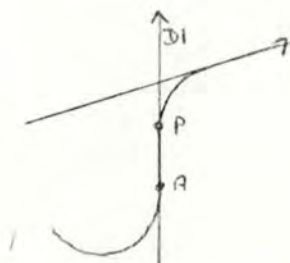
L'avion est libre de choisir le point de départ qui lui convient le mieux pour négocier son virage. Normalement, il choisira le point qui lui permettra de négocier le virage avec un rayon minimal. Il y a cependant des cas où soit c'est impossible, soit ce n'est pas réaliste :

c'est impossible, si l'avion doit virer de beaucoup plus loin qu'il ne lui est possible (ex 2), ce n'est pas réaliste si l'avion est contraint de parcourir une distance énorme afin de négocier son virage avec un rayon minimum (ex 3)

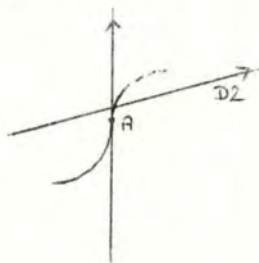
Dans ce cas, il négociera directement son virage en A, on est ramené au cas 5.I.I.

Condition : Le point à survoler doit être bien situé sur la radiale.

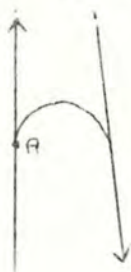
Ex I. L'avion se trouve sur la radiale DI et peut choisir son point de départ, il choisira le point P qui lui permet de négocier son virage avec le rayon minimal.



Ex 2. L'avion ne peut rejoindre la radiale D2 avec le rayon minimum. Le texte-SID correspondant n'est pas correct.



Ex 3. L'avion devrait parcourir une distance beaucoup trop grande, il va donc directement le négocier en A.

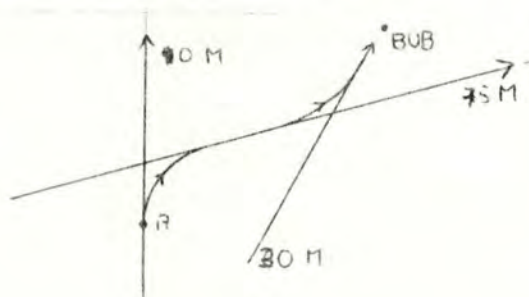


6. STURN

Syntaxe : { AT<NAVAID> } { TURN } <DIRECTION> TO INTERCEPT
TR <NOMBRE> M TO <NAVAID>

Le STURN (Virage en forme de "S") est un cas particulier du virage vers une radiale d'un point donné. Il est utilisé lorsque l'angle entre les deux radiales est assez faible (normalement $< 45^\circ$).

Il permet d'éviter à l'avion de devoir utiliser un rayon de virage beaucoup trop grand qui ne lui permettrait pas d'atteindre le point souhaité.



Le mécanisme du STURN est d'adopter un cap intermédiaire faisant un angle de 45° avec le cap final avant d'adopter le cap final lui-même.

Conditions :

1. La direction indiquée par le texte-SID doit être la bonne :
virer à gauche si l'avion se trouve à droite de la radiale
virer à droite si l'avion se trouve à gauche de la radiale
2. L'avion ne doit pas être "trop près" de la radiale.
3. Le point à survoler doit être bien situé sur la radiale.

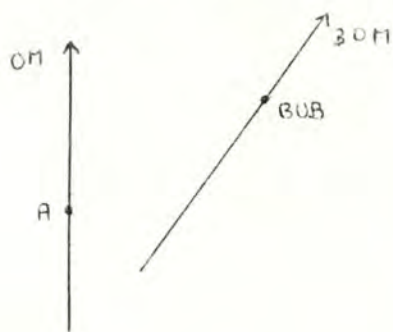
Il est à remarquer que l'utilisation d'un STURN hors du cas pour lequel il est prévu ($\text{angle} > 45^\circ$) peut conduire à des trajectoires qui, bien que n'étant pas erronées, défient le bon sens.

Ex 1. LEFT TO INTERCEPT TR 30 M TO BUB (erreur 1)

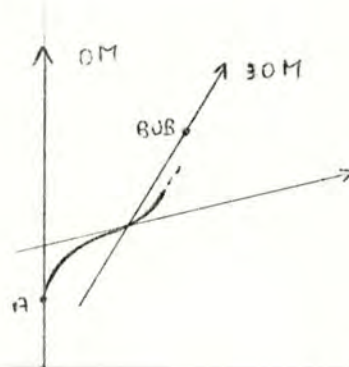
Ex 2. RIGHT TO INTERCEPT TR 30 M TO BUB (erreur 2)

Ex 3. RIGHT TO INTERCEPT TR 30 M TO BUB (erreur 3)

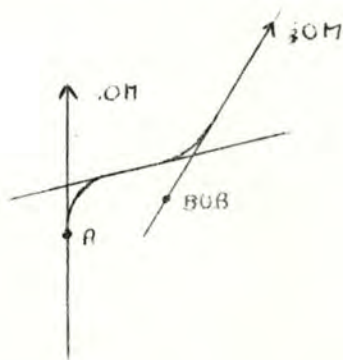
Ex 4. Utilisation abusive d'un STURN, un virage radiale simple serait beaucoup plus approprié.



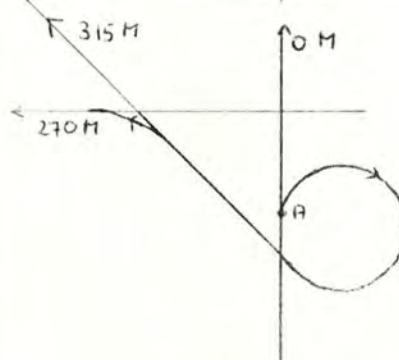
EX 1



EX 2



EX 3



EX 4

2. OUTPUT

La sortie du programme est la trajectoire CANAR ou TRACK, c'est la conversion sous forme numérique d'un texte-SID. Un track est une suite de segments de vol. Il existe deux types de segments :

- un segment droit caractérisé par une longueur.
- un segment courbe caractérisé par le rayon du virage, le sens du virage (gauche ou droite) et l'angle du virage.

La succession des éléments d'un TRACK correspond donc à la trajectoire suivie par un avion parcourant successivement les segments de vol.

Ex. TRACK I segment droit 3 NM

I segment courbe gauche, 50°, 5 NM

I segment droit 2 NM

correspond à la trajectoire d'un avion parcourant 3 NM avec le cap d'origine, virant à gauche d'un angle de 50° avec un rayon de 5 NM et continuant sur le même cap pendant 2 NM.

Le point de départ de la trajectoire est par convention la fin de la piste de décollage, le cap initial étant l'orientation de la piste de décollage. Ces paramètres figurent également dans la sortie du programme SIDCAN.

Conventions concernant les segments de vol

Ces conventions sont liées au programme CANAR.

- I. Le TRACK est constitué d'un maximum de 15 segments.
2. Chaque segment est soit courbe, soit droit.
3. Un segment est constitué de deux nombres réels :

segment droit : Le premier nombre est la longueur du segment en NM, le second est obligatoirement 0.

segment courbe : Le premier nombre est l'angle du virage en degrés. L'angle est toujours une valeur comprise entre 0 et 360°. Le second nombre est le rayon du virage en NM. Si ce nombre est négatif (positif), il s'agit d'un virage vers la gauche (droite).

4. Les premier et dernier segments du TRACK sont obligatoirement des segments droits. De plus, les segments courbes et droits doivent alterner.

Les segments impairs seront donc droits, les segments pairs, courbes.

CONCEPTION DU PROGRAMME SIDCAN

Introduction : schéma général

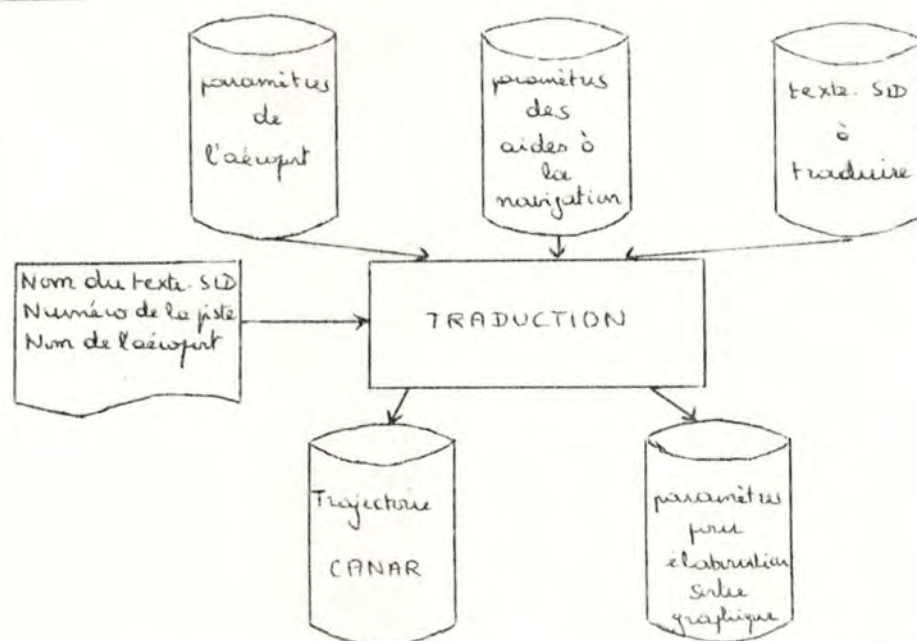
Le programme SIDCAN comporte deux parties :

1. Sous-routine TRADUCTION
2. Sous-routine GRAPHIQUE

Ces deux parties sont totalement indépendantes l'une de l'autre et peuvent être exécutées isolément. Elles sont accessibles par l'intermédiaire d'un menu.

I. TRADUCTION

La sous-routine traduction assure la lecture d'un texte-SID sur fichier externe, la traduction de celui-ci en un TRACK selon les conventions du programme CANAR et le sauvetage de celui-ci sur fichier externe. Les contrôles sont faits sur la validité lexicale, syntaxique et sémantique du texte-SID. Toute détection d'erreur provoque l'arrêt du programme avec un message d'erreur approprié.



Les renseignements que doit fournir l'utilisateur pour l'exécution de la routine sont :

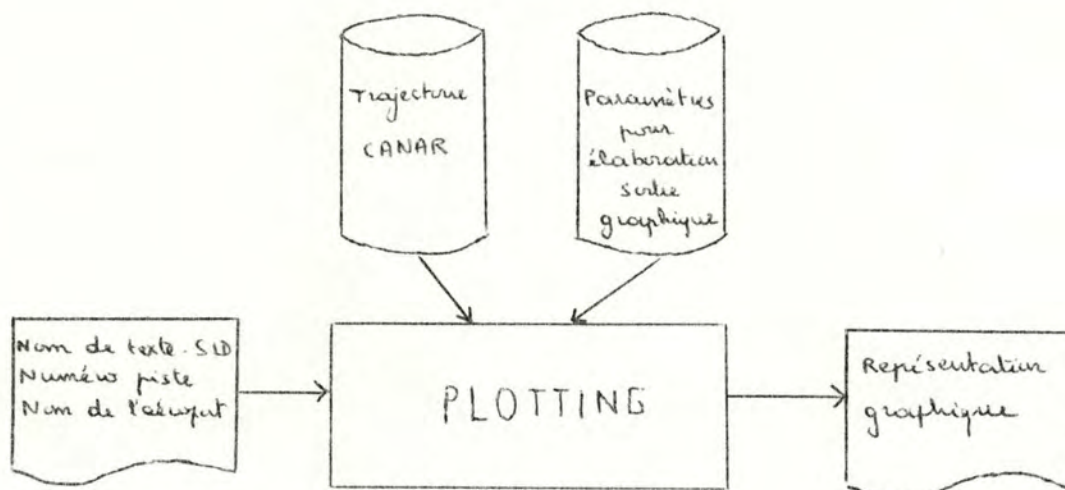
1. Le nom de l'aéroport
2. Le nom du texte-SID à traduire
3. Le numéro de la piste de décollage

2. PLOTTING

La sous-routine PLOTTING assure la lecture d'une trajectoire CANAR et de ses paramètres graphiques et une représentation de celle-ci sur écran graphique.

Les renseignements que doit fournir l'utilisateur pour l'exécution de la routine sont :

1. Le nom de l'aéroport
2. Le nom du texte-SID à traduire
3. Le numéro de la piste de décollage



I MODULE TRADUCTION

A. Conception

On entend par TRADUCTION, le fait de "lire" un texte-SID et de le convertir en une trajectoire numérique.

Il y a donc deux aspects assez différents de la TRADUCTION :

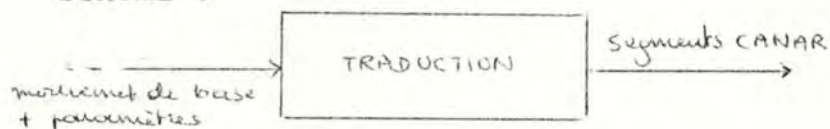
I. LECTURE et ANALYSE : il s'agit, à partir d'une chaîne de caractères (le texte-SID), de déterminer des mouvements de base et des paramètres.

Schéma :



2. TRADUCTION : convertir ces mouvements de base et paramètres en données numériques, les segments CANAR.

Schéma :



Stratégie générale : on traduit le texte-SID phrase par phrase (def. p.14). Chaque phrase est successivement lue, analysée et traduite.

B. Réalisation

Le module TRADUCTION répondra aux spécifications I+2.

Algorithme : Tant qu'il y a des "phrases" à traiter

	[	- traiter une phrase
Traiter une phrase	[	- lire une phrase P
		- Analyser la phrase P
		- Traduire la phrase P

I.I. LECTURE et ANALYSE

Le processus de lecture et d'analyse est celui qui consiste à transformer un flot de caractères en :

- un type de mouvement de base
- les paramètres de ce mouvement

Ex. ^TURN^^RIGHT^TO^^INTERCEPT^^TR^223^M^TO^^NAV^^

L'analyseur fournira les résultats suivants :

mouvement de base STURN

paramètres RIGHT 223 NAV

(le souligné permet de distinguer le paramètre en tant que tel de sa représentation sous forme d'une suite de caractères)

La correspondance entre un flot de caractères et les résultats de l'analyse (type du mouvement + paramètres) se fait par l'utilisation de règles (ou productions).

Celles-ci fixent la syntaxe correcte de chaque mouvement de base. On peut distinguer deux sortes de règles :

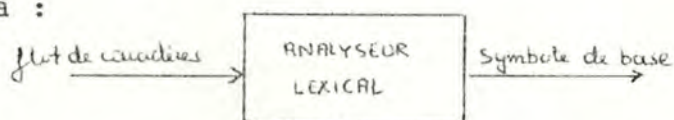
- les règles syntaxiques qui déterminent la syntaxe d'un texte-SID en termes de symboles de base.
- les règles lexicales qui déterminent la syntaxe d'un symbole de base en terme de caractères.

Le module ANALYSE peut, suivant cette dichotomie, être séparé en deux sous-modules :

- un module ANALYSEUR LEXICAL
- un module ANALYSEUR SYNTAXIQUE

ANALYSEUR LEXICAL

Schéma :



Les règles lexicales permettent de "voir" un texte-SID comme une suite de caractères. Dans la grammaire SIDCAN, un symbole de base peut être un mot-clé, un identificateur, un nombre ou une "fin-de-fichier".

Les symboles de base correspondent aux symboles terminaux dans les règles syntaxiques exprimées en BNF.

La fonction de l'analyseur lexical (ou Scanner) est de transformer le flot de caractères provenant de la source (SIDCAN : un fichier transféré lors de l'initialisation en une chaîne de caractères) en un flot de symboles disponible pour l'analyseur syntaxique. La façon concrète dont cette information est disponible est la suivante :

On peut voir le flot de symboles comme une suite ordonnée de symboles de base ; à chaque invocation par l'analyseur syntaxique, l'analyseur lexical fournit le symbole "courant". Il faut comprendre par symbole courant le *i*ème symbole de la suite (c'est à dire le *i*ème symbole le plus à gauche) lors de la *i*ème invocation.

Concrètement, l'analyseur lexical est une fonction dont chaque appel donne comme résultat un "symbole de base".

Ce résultat constitue l'interface AN. LEXICAL → AN. SYNTAXIQUE. L'interface est donc un paramètre résultat de la fonction

ANALYSEUR LEXICAL. La valeur de ce paramètre est

1. Le type de symbole de base
2. Une valeur pour ce symbole de base

Certains symboles de base n'ont cependant qu'un type et pas de valeur (Ex. Fin-de-fichier).

Pour convertir un flot de caractères transmis par la source en une suite de symboles de base, l'analyseur dispose

1. De quelques règles BNF lui donnant la syntaxe exacte de chaque symbole de base
2. De quelques conventions liant un symbole de base à sa représentation physique (en termes de caractères)

I. Règles BNF

```

<symbole de base> ::= <symbole> <blancs> | <fin-de-fichier>
<symbole>          ::= <mot-reservé> | <identificateur> | <nombre>
<mot-reservé>      ::= AHEAD | UNTIL | NM | ONTO | TR | TO |
                        M | INTERCEPT | RIGHT | LEFT | TURN | . | THEN
<identificateur>   ::= <lettre> { <lettre> | <chiffre> } *
<nombre>           ::= <entier> | <entier> . <entier>
<entier>           ::= <chiffre> | <entier> <chiffre>
<lettre>           ::= A | B | C | ... | Y | Z
<chiffre>          ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
<blancs>           ::= <blanc> { <blanc> } *
<blanc>            ::=

```

2. Conventions de représentation physique

- a) Le flot de caractères se lit de la gauche vers la droite.

- b) Chaque symbole est exclusivement constitué de caractères
- . alphabétiques (Majuscules de A à Z)
 - . numériques (de 0 à 9)
 - . du signe spécial .
- c) Le caractère " " (espace) sert de délimiteur.
Il ne peut pas figurer à l'intérieur d'un symbole (b).
deux symboles successifs doivent être obligatoirement
séparés par un ou plusieurs blancs.
- d) Le symbole de base FF dépend de l'implémentation physique du flot de caractères. C'est à l'analyseur lexical qu'il appartient de détecter la fin de fichier et de renvoyer un symbole de base FF
(dans SIDCAN, la fin du fichier est détectée par le fait que le pointeur courant positionne sur le dernier caractère de la chaîne de caractères SID)
- e) Un identificateur ne peut contenir qu'un maximum de dix caractères.

Deux méthodes d'application de ces règles ont retenu mon attention :

I. Méthode directe

L'analyseur syntaxique se sert directement des règles de syntaxe pour déterminer le mouvement de base à effectuer. Il y a donc une correspondance presque directe entre l'algorithme et la syntaxe elle-même.

La lecture (lire les mots de la phrase un par un) et l'analyse (déterminer le mouvement de base) se font donc conjointement.

Ex. Si le symbole de base lu est AHEAD

Alors

"le mouvement de base est droite "

Si le symbole de base suivant n'est pas UNTIL
ou nombre

Alors "il y a une erreur"

·
·
·

Je n'ai pas utilisé cette méthode (cfr p.18).

2. Utilisation des masques

La lecture d'une phrase se fait mot par mot.

Les mots constitutifs d'une phrase-SID sont essentiellement de deux types :

- I. les mots-clés : ils permettent de "reconnaître" le type de mouvement de base. Ils peuvent également assurer

la syntaxe de la phrase. Dans cette dernière fonction, ils seraient à la limite inutiles. Ils servent simplement à conserver à la phrase-SID la forme d'un texte anglais :

(ON AT ONTO ...).

2. les paramètres : ceux-ci serviront de données aux routines spécialisées qui traduisent les différents mouvements.

Paramètres de direction : LEFT, RIGHT

distance : nombre

cap : degrés, navaid

Cette distinction est importante car le traitement diffère en fonction du type du mot rencontré.

Le traitement d'une phrase se fait en deux temps :

- phase de lecture et codage : les paramètres sont stockés et chaque symbole de base est codé.
- phase d'analyse : la suite de symboles de base ainsi codée est comparée à toutes les formes possibles de phrase-SID afin de déterminer le mouvement de base à utiliser.

Les règles de syntaxe ne figurent donc plus dans l'algorithme mais sous forme de données dans un tableau. (MASK)

I) Phase de lecture et codage

a) codage

Chaque symbole de base est codé par une lettre de l'alphabet.

Un code existe pour chaque valeur du type de symbole de base "mot réservé". Ce code correspond à l'ordre de ce mot

dans la liste des mots réservés. (dans SIDCAN, AHEAD → A, AT → B ...). Cette liste figure sur un fichier (DICT.DAT) et est classée par ordre alphabétique (pour accroître la vitesse de recherche du mot).

Pour les autres types de symboles de base (c-à-d dans SIDCAN, identificateur , nombre), un code existe pour chaque type de symbole de base (sauf pour le symbole de base

fin-de-fichier qui ne fait pas à proprement parler partie du texte-SID). (dans SIDCAN, <nombre> → R <identificateur> → S) On appelle masque d'une phrase-SID le codage de cette phrase selon cette technique.

Ex. phrase RIGHT TO INTERCEPT TR 100 M TO NAV

masque H I C K R E I S

b) stockage

Chaque valeur lue est stockée en vue de son utilisation lors de la traduction. Plusieurs tableaux sont nécessaires pour stocker des valeurs de type différent.

Dans SIDCAN, deux tableaux sont prévus :

un pour les valeurs numériques : TNOMBRE

un pour les valeurs alphanumériques: TALPHA

Le fait d'utiliser des tableaux doit permettre une adaptation aisée du programme en permettant aux mouvements de base futurs de compter autant de variables que désirées.

Actuellement deux classes de symboles de base demandent le stockage de données (<nombre> et <identificateur>).

Il n'est pas difficile d'augmenter le nombre de classes de symboles de bases et de tableaux de stockage.

2) Phase d'analyse

Le stockage termine la première phase de l'analyse (phase de lecture). La deuxième phase (reconnaissance du mouvement de base) est assez simple.

Il suffit de comparer le masque obtenu avec l'ensemble de tous les masques possibles, ce qui permet de déterminer le mouvement de base. Le tableau MASK reprend pour chaque masque possible, le numéro du mouvement de base correspondant (voir ex. en Annexes p. 2).

Le fait de coder les symboles de base par une lettre permet

- de classer les masques de phrases possibles par ordre alphabétique ce qui permet d'utiliser la recherche dichotomique.
- d'utiliser les primitives FORTRAN permettant de comparer deux chaînes de caractères, ce qui a pour résultat une recherche très rapide du masque correspondant à la phrase analysée.

Comparaison des deux méthodes

I) Méthode directe

- avantages :
- Elle est plus naturelle et se prête mieux à la méthode de programmation Top-Down, la structure de la grammaire utilisée correspond en effet de façon plus ou moins directe avec la structure du programme d'analyse.
 - Cette méthode permet d'éviter l'utilisation de fichiers externes (DICT, MASK).

inconvénients : - il est assez malaisé de modifier la grammaire utilisée car ceci impliquerait une modification assez radicale du programme puisque les deux structures sont liées.

- la grammaire doit posséder un certain nombre de propriétés assez restrictives, notamment elle doit être LL(I), ce qui n'est pas le cas de la grammaire SID

2) Méthode des masques

Avantages :-il est assez aisé de modifier la grammaire. Il est même possible d'automatiser la création des fichiers externes DICT et MASK (voir p. 25)

-La grammaire SID peut être plus souple (bien qu'elle ne puisse pas être récursive)

Inconvénients :-Méthode moins naturelle

-Risque d'explosion combinatoire du nombre de masques (cependant la grammaire actuelle bien que ne permettant qu'un nombre limité de phrases peut déjà rendre de nombreux services.)

3) Choix

Il fallait choisir la méthode qui réponde le mieux aux critères d'extensibilité et de simplicité du programme.

Il fallait prévoir une stratégie d'analyse suffisamment souple pour permettre au programmeur (ou à l'utilisateur) une évolution assez aisée du produit.

C'est en fonction de ces critères qu'a été choisie la deuxième méthode.

I.2. TRADUCTION

I.2.I. Principes généraux

Introduction

La traduction d'un mouvement de base en une trajectoire numérique se fait par la routine TRADUCPHRASE.

Celle-ci s'occupe de

- faire les conversions nécessaires, notamment :
 - conversion coord. géogr. → coord. cartésiennes
 - conversion NM → KM
 - conversion cap magnétique → cap trigonométrique
- appeler les différentes routines spécialisées

Il existe une routine spécialisée pour chaque mouvement de base. Un mouvement de base pouvant cependant être constitué de plusieurs mouvements de base plus élémentaires, certaines routines peuvent le cas échéant s'appeler l'une l'autre.

Donnons d'abord quelques règles générales nécessaires à la compréhension des principes qui régissent la traduction des mouvements de base.

J'aborderai successivement :

- stratégie générale de traduction
- localisation des points dans l'espace
- représentation des droites et des angles
- unités et constantes
- conversion coordonnées géographiques → cartésiennes

I.2.I.I. Stratégie générale de traduction

Chaque mouvement de base représente un ou plusieurs segments dans le format CANAR. Ces segments sont créés au fur et à mesure de la traduction de chaque mouvement de base.

Chaque routine spécialisée s'occupe donc elle-même du calcul, de la création et du stockage des segments représentant la trajectoire. Ces segments sont stockés provisoirement, en cours de traduction, dans un tableau. Ce tableau sera sauvé sur fichier externe lorsque le texte-SID sera entièrement traduit.

Chaque routine peut être appelée à modifier ce qui a déjà été créé dans les routines exécutées précédemment avec les restrictions suivantes :

- seul le dernier segment stocké peut être modifié
- on ne peut modifier que des segments droits pour en prolonger ou raccourcir la longueur.

Il existe donc plusieurs variables globales accessibles à n'importe quel moment au cours de la traduction et par n'importe quelle sous-routine.

Ce sont :

1. Les tableaux contenant les segments de vol CANAR
2. La position de l'avion
3. Le cap de l'avion

Certaines routines modifient la valeur de ces variables, d'autres pas :

Les différents points de repère nécessaires à la traduction sont fournis au programme en coord. géogr.

Une sous-routine COORD se charge de la conversion avant l'utilisation de ces repères.

(Il est à remarquer que cette conversion entraîne une erreur puisque la surface d'une sphère n'est pas développable, on tiendra ces erreurs pour négligeables, étant donné les faibles distances mises en jeu)

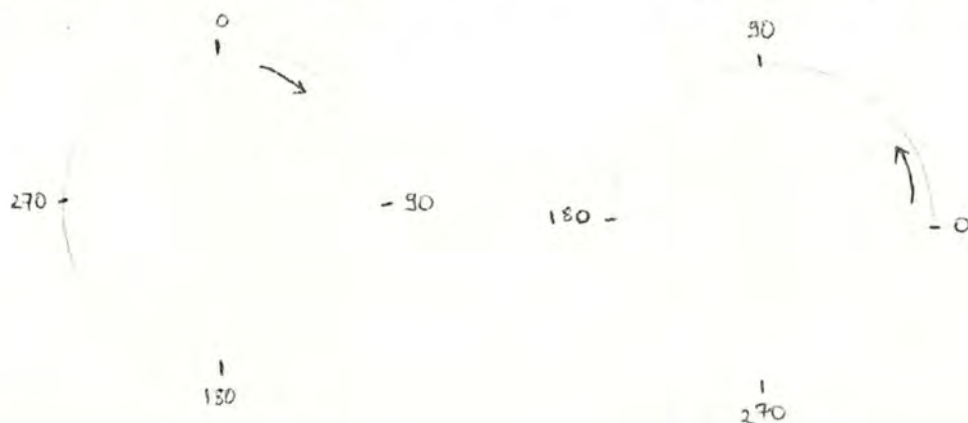
La présentation mathématique et informatique de cette conversion figure p. 35

I.2.I.3 Représentation des droites et des angles

a) conversion cap magnétique $\rightarrow$ cap trigonométrique

Le cercle magnétique des angles — le 0° est vers le Nord et le sens de parcours est le sens des aiguilles d'une montre — ne correspond pas au cercle trigonométrique où le 0° se situe 90° à l'est et le sens du parcours est inversé. Tous les calculs trigonométriques se font en utilisant les angles trigonométriques : une conversion (assez simple) est donc nécessaire pour passer d'un système à l'autre.

EX. Le cap magnétique 180° correspond au cap trigono. 270° .



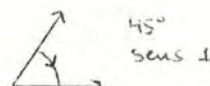
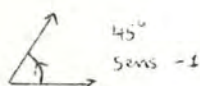
b) déviation magnétique

Une autre correction à faire est celle de la déviation magnétique. Le nord géométrique (= direction de la droite qui joint le point où l'on se trouve à l'axe de rotation de la terre) ne correspond pas au nord magnétique (= direction des lignes de force du champ magnétique terrestre).

Une correction est nécessaire, elle dépend de la latitude de l'aéroport et figure dans les paramètres de chaque aéroport. Pour nos latitudes (50° N), la déviation magnétique est d'environ 2° .

I.2.I.4 Unités et constantes

Tous les angles sont exprimés en degrés et compris entre 0° et 360° . La différence entre deux angles est toujours exprimée par un angle positif et un sens (gauche = -I, droite = I).



Les distances sont exprimées en NM ($=1.85318$ KM).

Rayon minimum du virage

Le rayon minimum du virage sera souvent employé dans les calculs. Il correspond au rayon du virage le plus serré que l'avion puisse prendre compte tenu de sa vitesse, de ses caractéristiques...

Il se calcule comme suit :

données : vitesse de l'avion 250 KTS (NM/h)

taux du virage I (correspond à un virage de 360° en 2 m)

distance parcourue en deux minutes (=circonférence) =

$$2 \times \frac{250 \times 1.85318}{60} = 15,443 \text{ KM}$$

$$\text{Rayon minimum du virage} : \frac{15,443}{2 \times \pi} = 2,458 \text{ KM}$$

I.2.I.5. Conversion coord. géographiques → coord. cartés.

a) résolution mathématique

données : un point P dont on connaît les coordonnées géographiques. Ce point P est le centre de l'aéroport et sera pris comme point d'origine du système d'axes du plan cartésien que l'on se propose de construire.

I) conversions

latitude N +

S -

+ conversion système sexagésimal → décimal

longitude E angle

O 360° - angle

+ conversion système sexagésimal → décimal

Ex. LAT : N51:27:34 +51,459444..

LON : E10:15:20 +10,255555..

LAT : S10:30:59 -10,5163888..

LON : O15:20:35 +344,656944..

2) La terre peut être assimilée à un ellipsoïde de révolution

ayant pour axes principaux r_I, r_I, r_3 avec $r_I = 6392,015$ KM
et $r_3 = 6374,567$ KM

Tout point de la surface de la terre est le résultat de l'équation

$$\frac{x^2}{r_I^2} + \frac{y^2}{r_I^2} + \frac{z^2}{r_3^2} = 1$$

3) conversion coord. géo. → coord. cart. (3 dimensions)

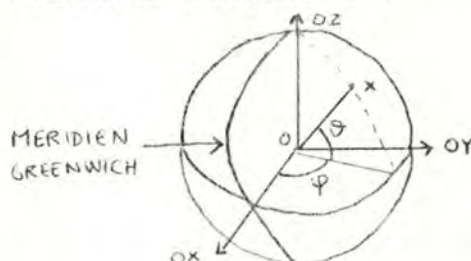
On construit un système d'axes ayant pour centre le centre de la terre, pour axe oz l'axe de rotation de la terre et pour plan (ox, oz) le plan du méridien de Greenwich.

On peut trouver les formules suivantes :

$$x = r_I \cos \varphi \cos \theta$$

$$y = r_I \sin \varphi \cos \theta$$

$$z = r_3 \sin \theta$$



Ex. Le point (N051:08:42, W000:11:48) a pour coordonnées cartésiennes (4010,017; -13,764; 4964,106) en KM

4) construction d'un système cartésien à deux dimensions

-équation des points du plan tangent au point P

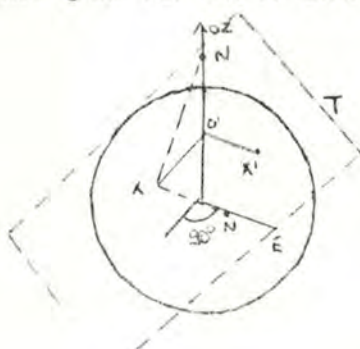
on a $OX \perp XA \forall A \in \text{Plan}$ (a,b,c coord. de A)

donc $(x,y,z) \perp (a-x, b-y, c-z)$

donc (produit scalaire nul) $\underline{ax + by + cz = x^2 + y^2 + z^2}$

-construction des deux axes

I. Axe Nord : donné par XN ou N est l'intersection du plan avec OZ



$$\text{coordonnées} \begin{cases} a = 0 \\ b = 0 \\ ax + by + cz = x^2 + y^2 + z^2 \end{cases} \quad \text{point N est sur l'axe OZ}$$

$$N = (0, 0, \frac{x^2 + y^2 + z^2}{z})$$

$$XN = (-x, -y, \frac{x^2 + y^2 + z^2}{z} - z)$$

Il faut encore normer l'axe et on obtient $XN(1), XN(2), XN(3)$

2. Axe Est : il est donné par XE

Soit le point P' de coordonnées $(\theta, \varphi + \frac{\pi}{2})$

Il a pour coordonnées cartésiennes $(-y, x, z)$ (rotation de 90° vers l'est)

On a $XE // O'X'$ avec $O' = (0, 0, z) \Rightarrow O'X' = (-y, x, 0) \Rightarrow$

$$XE = (-y, x, 0)$$

(le point $E = XE + X = (-y, x, 0) + (x, y, z) = (x-y, x+y, z)$

remplit bien l'équation du plan T)

On norme XE et on obtient $XE(1), XE(2), XE(3)$

5) Coordonnées dans le repère (XN, XE) d'un point z

on va prendre la projection de ce point (car le point z à la surface de la terre n'appartient pas au plan T)

Soit (m, n, p) les coord. cartésiennes de z

coord. cartésiennes de z' (projection de z sur plan T) :

$$z' = \beta \cdot z \text{ avec } \beta = \frac{x^2 + y^2 + z^2}{mx + ny + pz} \quad (\text{car } z' \in T)$$

On exprime z' ($\in T$) en fonction des axes :

$$Xz' = A(XE) + B(XN)$$

$$\begin{cases} z'_1 = Ae_1 + Bn_1 \\ z'_2 = Ae_2 + Bn_2 \\ z'_3 = Ae_3 + Bn_3 \end{cases}$$

On ne garde que les 2 premières (la 3ème est redondante)

$$A = \frac{z'_1 n_2 - z'_2 n_1}{n_2 e_1 - n_1 e_2} \quad B = \frac{z'_1 e_2 - z'_2 e_1}{n_1 e_2 - n_2 e_1}$$

Cas d'exception à traiter séparément :

I) $\theta = 0$ (aéroport sur l'équateur)

$$XN(1) = 0 \quad XE(1) = \cos(\theta + 90)$$

$$XN(2) = 0 \quad XE(2) = \sin(\theta + 90)$$

$$XN(3) = 1 \quad XE(3) = 0$$

b) résolution informatique

La conversion coord. géogr. $\rightarrow$ coord. cartés. regroupent en fait deux problèmes distincts :

1. Le calcul des axes (XN,XE) étant donné un point X à la surface du globe. Ce calcul est fait une fois pour toute à l'initialisation de SIDCAN.

Un cas (hautement improbable !) est détecté mais n'est pas traité : si l'aérodrome se trouve sur un des pôles ; un message d'erreur s'ensuit.

2. Le calcul des coord. cartés. d'un point par rapport au système d'axes (XN,XE). Ce calcul est fait par une procédure spéciale COORD dont les arguments sont la longitude et la latitude du point et le résultat les coord. cartés.

I.2.2 Présentation des routines spécialisées

I.2.2.1. ENVOL

<u>Données</u>	(A_x, A_y)	Position de l'avion
	α	Cap de l'avion
	DIST	Distance à parcourir
	RWYLENGHT	Longueur de la piste de décollage
<u>Résultats</u>	(B_x, B_y)	Position de l'avion
	β	Cap de l'avion
	I segment droit CANAR	

Calculs

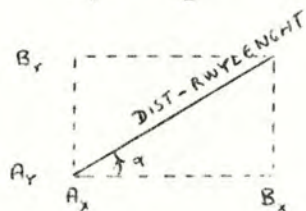
L'envol correspond au parcours d'un segment de droite dont la longueur est la longueur à parcourir depuis le début de la piste (DIST) - la longueur de la piste elle-même (RWYLENGHT) (car la trajectoire CANAR commence en fin de piste).

Segment droit : longueur DIST - RWYLENGHT (+ conversion KM/NM)

β : $\beta = \alpha$ car segment droit, l'avion conserve son cap.

$$(B_x, B_y) : B_x = A_x + (DIST - RWYLENGHT) \cos(\alpha)$$

$$B_y = A_y + (DIST - RWYLENGHT) \sin(\alpha)$$



I.2.2.2 DROITE

<u>Données</u>	(A_x, A_y)	Position de l'avion
----------------	--------------	---------------------

	α	Cap de l'avion
	DIST	Distance à parcourir
<u>Résultats</u>	(B_x, B_y)	Position de l'avion
	β	Cap de l'avion
	I segment CANAR droit	

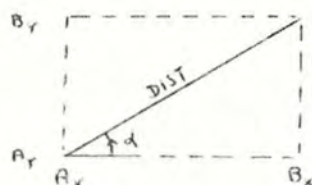
Calculs

Segment droit : longueur DIST (+ conversion KM / NM)

$\beta: \beta = \alpha$ car segment droit, l'avion conserve son cap.

$$(B_x, B_y) : B_x = A_x + \text{DIST} \cos(\alpha)$$

$$B_y = A_y + \text{DIST} \sin(\alpha)$$



I.2.2.3 CAP

<u>Données</u>	(A_x, A_y)	Position de l'avion
	α	Cap de l'avion
	β	Nouveau cap à atteindre
	SENS	Sens du virage (I=droit, -I=gauche)
	RMIN	Rayon minimum du virage
<u>Résultats</u>	(B_x, B_y)	Position de l'avion
	β	Cap de l'avion
	I segment courbe CANAR	

Calculs

Segment courbe : I. sens : SENS

2. rayon : rayon minimum (converti en NM)

3. angle du virage : DIFF

Calcul de DIFF

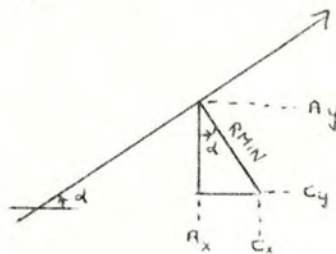
- Si virage à gauche : $\text{DIFF} = \beta - \alpha$
- Si virage à droite : $\text{DIFF} = \alpha - \beta$
- Si $\text{DIFF} < 0^\circ$: $\text{DIFF} = \text{DIFF} + 360^\circ$

(B_x, B_y) : Calcul

1. Calcul du centre du virage

$$C_x = A_x + R_{\text{MIN}} \sin(\alpha) * \text{SENS}$$

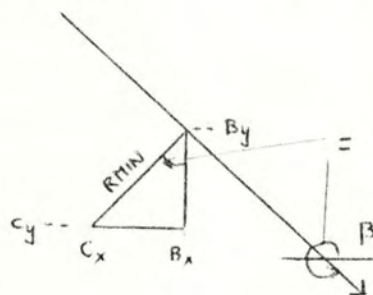
$$C_y = A_y - R_{\text{MIN}} \cos(\alpha) * \text{SENS}$$



2. Calcul de la fin du virage

$$B_x = C_x - R_{\text{MIN}} \sin(\beta) * \text{SENS}$$

$$B_y = C_y + R_{\text{MIN}} \cos(\beta) * \text{SENS}$$



Cas d'exception Si $\alpha = \beta$ alors pas de virage

I.2.2.4 RADIALE

<u>Données</u>	(A_x, A_y)	Position de l'avion
	α	Cap de l'avion
	β	Cap à atteindre
	NAV	Point à survoler
	CONTPD	Indicateur si il y a contrainte sur le point de départ (vrai, faux)
	RMIN	Rayon minimum du virage
	SENS	Sens du virage (I=droit, -I=gauche)
<u>Résultats</u>	(B_x, B_y)	Position de l'avion
	β	Cap de l'avion
	<u>Si</u> CONTPD <u>alors</u>	2 segments CANAR (C,D) (*) (cas 1)
		4 segments CANAR (C,D,C,D) (*) (cas 2)
	<u>sinon</u>	3 segments CANAR (D,C,D) (cas 1)
		2 segments CANAR (C,D) (*) (cas 2)
		4 segments CANAR (C,D,C,D) (*) (cas 3)
		(*) + I segment droit si dernier segment traduit courbe

Calculs

Hyp. : les radiale de départ et d'arrivée sont sécantes

I. Calcul du point d'intersection (I_x, I_y)

A) Il y a contrainte sur le point de départ

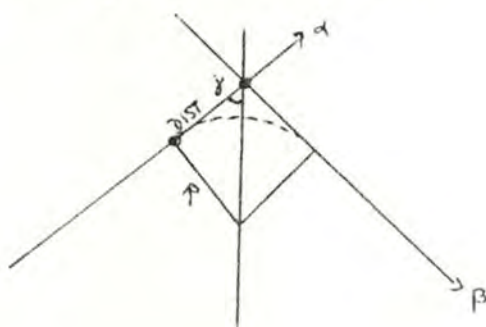
. Le point d'intersection se trouve devant l'avion (cas I)

- Calcul de la bissectrice : angle $\gamma = \frac{180 - \text{DIFF}}{2}$

(Calcul de DIFF p. 41)

- Calcul de la distance entre (A_x, A_y) et $(I_x, I_y) = \text{DIST}$

- Calcul du rayon du virage = R



$$R = \text{DIST} * \text{TAN}(\gamma)$$

1er segment CANAR :

1. sens : SENS

2. rayon : R

3. angle du virage : DIFF (calcul p 41)

- calcul du point d'arrivée du 1er segment courbe = P

- calcul du centre du virage (voir p 41)

- calcul de la fin du virage (voir p 41)

- l'avion doit encore parcourir la distance séparant P de NAV.
= DIST

2ème segment CANAR : droit de longueur DIST

Cas d'exception

1. Le dernier segment CANAR traduit est courbe : il faut avant tout calcul insérer un segment droit de longueur arbitrairement petite.

2. Le sens du virage doit être adapté à la direction à prendre. Calcul : si DIFF > 180 alors erreur
traitement : message d'erreur et arrêt du programme.

3. Le rayon du virage R doit être compatible avec les possibilités de l'appareil. Calcul : si R < RMIN alors erreur
traitement : message d'erreur et arrêt du programme.

4. Le point à survoler NAV doit se trouver devant l'appareil une fois que celui-ci se trouve sur la radiale à atteindre.
Calcul : si NAV précède P alors erreur
traitement : message d'erreur et arrêt du programme.

. Le point d'intersection se trouve derrière l'avion

Deux cas sont possibles :

a) $\text{DIFF} > 180$ (cas 1)

- calcul de la bissectrice : $\text{angle} = \frac{\text{DIFF} - 180}{2}$
- calcul de la distance entre (A_x, A_y) et $(I_x, I_y) = \text{DIST}$
- calcul du rayon du virage = R (voir p. 42)

1er segment CANAR :

1. sens : SENS

2. rayon : R

3. angle du virage : DIFF

- calcul du point d'arrivée P du 1er segment courbe
 - calcul du centre du virage (voir p. 41)
 - calcul de la fin du virage (voir p. 41)
- l'avion doit encore parcourir la distance séparant P de NAV.
= DIST

2ème segment CANAR : droit de longueur DIST

Cas d'exception : 1, 3, 4 p. 43

b) $\text{DIFF} < 180$ (cas 2)

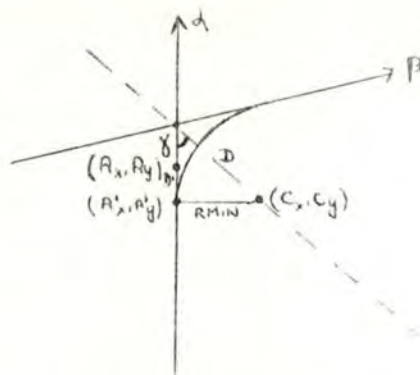
Dans ce cas, l'avion ne peut atteindre le point et le cap désirés par un simple virage : un S-TURN est nécessaire.

Voir le détail des calculs p. 41 (I.2.2.5)

B) Il n'y a pas de contrainte sur le point de départ.

. Le point d'intersection se trouve devant l'avion

- calcul du point de départ d'un virage avec rayon minimum



Hyp. On a deux droites sécantes A et B avec $A = (\text{angle } \alpha, \text{point } (A_x, A_y))$ et $B = (\text{angle } \beta, \text{point NAV})$. $\text{DIFF} < 180^\circ$ et (A_x, A_y) précède (I_x, I_y) .

Il faut calculer le point $(A'_x, A'_y) \in A$.

Soit (C_x, C_y) le point d'intersection de la bissectrice des deux droites A et B et de la perpendiculaire au point (A_x, A_y) à la droite A.

Par construction, (C_x, C_y) est le centre du virage à rayon minimum d'un avion commençant son virage en (A_x, A_y) et le terminant sur la radiale B.

Calcul

1. Calcul de l'angle $\gamma = \frac{180 - \text{DIFF}}{2}$, on a $0 < \gamma < 90$
2. Calcul de $D = \text{distance entre } (I_x, I_y) \text{ et } (C_x, C_y)$.
On a $D = \frac{R}{\sin(\gamma)}$
3. Calcul de $D' = \text{distance entre } (I_x, I_y) \text{ et } (A'_x, A'_y)$
on a $D' = D * \cos(\gamma)$
4. Calcul de $A'_x = I_x - \cos(\alpha) D'$
 $A'_y = I_y - \sin(\alpha) D'$

- La modification du point de départ est-elle possible ?

Calcul

1. Si (A_x, A_y) précède (A'_x, A'_y) ,
alors si $\text{dist}((A_x, A_y), (A'_x, A'_y)) < 5 \text{ KM}$
alors la modification est possible

2. Si (A_x, A_y) suit (A'_x, A'_y) ,
alors si (dernier segment traduit est droit) ET
 (sa longueur $\text{dist}((A_x, A_y), (A'_x, A'_y))$)
alors la modification est possible

Si la modification est possible alors

(cas 1)

1er segment CANAR droit : longueur $\text{dist}((A_x, A_y), (A'_x, A'_y))$
 (à ajouter ou retrancher du segment précédent)

2ème segment courbe :

1. sens : SENS

2. rayon : RMIN

3. angle : DIFF

- calcul de la fin du deuxième segment courbe P

- calcul du centre du virage (voir p. 41)

- calcul de la fin du virage (voir p. 41)

- l'avion doit encore parcourir la distance séparant P de NAV.

= DIST

3ème segment droit : longueur DIST

sinon

(cas 2)

On considère qu'il y a une contrainte sur le point de départ,
 voir calculs p.

Cas d'exception

voir points 2, 4 p. 43

. Le point d'intersection se trouve derrière l'avion (cas 3)

On considère qu'il y a malgré tout une contrainte sur le point de départ. Voir calculs p. 42

Calcul des autres résultats

(B_x, B_y) la nouvelle position de l'avion est le point NAV

I.2.2.5 STURN

<u>Données</u>	(A_x, A_y)	Position de l'avion
	α	Cap initial de l'avion
	β	Cap final à atteindre
	NAV	Point à survoler
	RMIN	Rayon minimum du virage
	SENS	Sens du virage (I=droit, -I=gauche)
<u>Résultats</u>	(B_x, B_y)	Position de l'avion
	β	Cap de l'avion
	4 segments CANAR (C, D, C, D)	
	+ I segment droit si dernier segment traduit courbe	

Calculs

Le STURN consiste à adopter un cap faisant un angle de 45° avec avant d'adopter lui-même.

Calcul du cap intermédiaire :

Si SENS = -I alors $\gamma = \beta + 45^\circ$ (si $\gamma \geq 360^\circ$ alors $\gamma = \gamma - 360^\circ$)

Si SENS = I alors $\gamma = \beta - 45^\circ$ (si $\gamma < 0^\circ$ alors $\gamma = \gamma + 360^\circ$)

- calcul du 1er virage :

ce calcul revient au mouvement de base CAP avec comme

données : $((A_x, A_y), \alpha, \gamma, \text{SENS}, R_{\text{MIN}})$.

les résultats de ce calcul sont :

Position de l'avion (PT_x, PT_y) , cap intermédiaire γ ,

un segment CANAR courbe

- calcul du segment de droite joignant les deux virages et du virage terminal :

Ce calcul revient au mouvement de base RADIALE avec comme

données : $((PT_x, PT_y), \gamma, \beta, \text{NAV}, \text{CONTPD}=\text{Faux}, R_{\text{MIN}}, -\text{SENS})$

les résultats de ce calcul sont :

Position de l'avion NAV, cap final β , trois segments CANAR (D, C, D)

- résultats de STURN :

$(B_x, B_y) = \text{NAV}$

Cap de l'avion $\beta = \text{cap final}$

4 segments CANAR (1 de CAP + 3 de RADIALE)

Cas d'exception

1. Le dernier segment CANAR traduit est courbe : il faut avant tout calcul insérer un segment droit de longueur arbitrairement petite.

2. La direction du virage doit être la bonne.

Si " (A_x, A_y) est à gauche" de la radiale à atteindre, alors la direction doit être droite (I)

Si " (A_x, A_y) est à droite" de la radiale à atteindre, alors la direction doit être gauche (-I)

traitement : message d'erreur et arrêt du programme

Comment déterminer la position d'un point par rapport à une droite géométrique ? Notions de "droite" et de "gauche"

Soit une droite géométrique dont on connaît l'équation. Cette équation est de la forme $Ax + By + C = 0$. Considérons la fonction $Ax + By + C$. L'équation d'une droite admet comme solution l'ensemble des couples de points (x,y) qui annulent la fonction.

Toute droite partage le plan en deux demi-plans. L'ensemble des points appartenant à un demi-plan donnent à la fonction une valeur négative : on appelle ce demi-plan la région négative de la droite. L'ensemble des points appartenant à l'autre demi-plan donnent à la fonction une valeur positive : on appelle ce demi-plan la région positive de la droite. On peut raccrocher ce concept à celui de droite et de gauche. Le demi-plan gauche de la droite est la région positive. Le demi-plan droit de la droite est la région négative.

Ce concept ne s'applique qu'aux droites orientées. L'équation de la droite doit donc permettre de distinguer deux droites identiques du point de vue géométrique mais dont l'orientation est inverse l'une de l'autre.

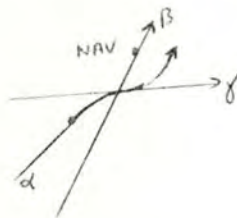
Une telle équation $A = -\sin(\alpha)$, $B = \cos(\alpha)$, $C = PT_x \sin(\alpha) - PT_y \cos(\alpha)$ permet de distinguer le sens de la droite.

3. Le STURN doit être possible.

Le STURN est impossible si, malgré l'utilisation du rayon minimal, l'avion se trouve trop près de la radiale

pour la rejoindre par un STURN normal.

Ex.



L'utilisation du rayon minimal ne permet à l'avion que de rejoindre une radiale parallèle à la radiale du point NAV.

Calcul :

On calcule successivement :

- centre du 1er virage (avec rayon minimum)
- fin du 1er virage
- centre du 2nd virage (avec rayon minimum)
- fin du 2nd virage PT

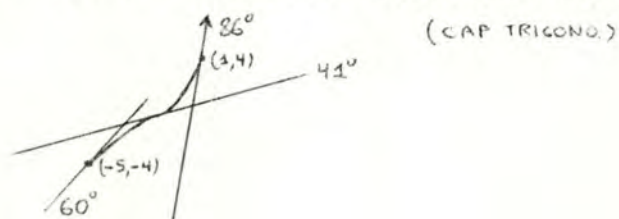
On compare la position de (A_x, A_y) et de PT.

S'ils sont situés du même côté de la radiale (d'arrivée), c'est que le STURN est possible.

traitement : si le STURN est impossible, message d'erreur et arrêt du programme.

Exemple de calcul du STURN

Soit le texte : RIGHT TO INTERCEPT TR 002 M TO BUB



I. Le sens est-il le bon ?

Le point de départ $(-5, -4)$ est à gauche de la droite d'arrivée (car la fonction $-\sin(86)(-5) + \cos(86)(-4) + (1)\sin(86) - (4)\cos(86)$ est positive)

Le sens proposé est la droite → OK

2. Le STURN est-il possible ?

Le calcul du point d'arrivée d'un double virage à rayon minimum donne ($B_x = -3.645$, $B_y = -2.590$).

La fonction de la droite d'arrivée est positive en ce point, les points (A_x, A_y) et (B_x, B_y) sont donc situés du même côté de la droite d'arrivée, le STURN est donc possible.

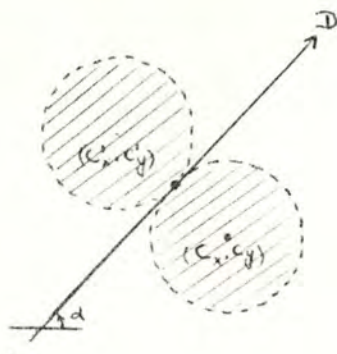
I.2.2.6 POINT

<u>Données</u>	(A_x, A_y)	Position de l'avion
	α	Cap de l'avion
	NAV	Point à survoler
	CONTPD	Indicateur si il y a contrainte sur le point de départ (vrai, faux)
	R MIN	Rayon minimum du virage
	SENS	Sens du virage (I=droit, -I=gauche)
<u>Résultats</u>	(B_x, B_y)	Position de l'avion
	β	Cap de l'avion
	2 segments CANAR	(C,D) (*) (cas 1)
	2 segments CANAR	(D,C) (cas 2)
	(*) + I segment droit si dernier segment traduit courbe	

Calculs

Deux cas sont à distinguer :

1. L'avion peut atteindre le point à partir de l'endroit où il se trouve.
2. Le rayon minimum de l'avion l'empêche d'atteindre ce point à partir de l'endroit où il se trouve.



Soit un point (A_x, A_y) et une droite D d'angle α .

Tout point situé dans une zone hachurée est inaccessible à partir du point (A_x, A_y) .

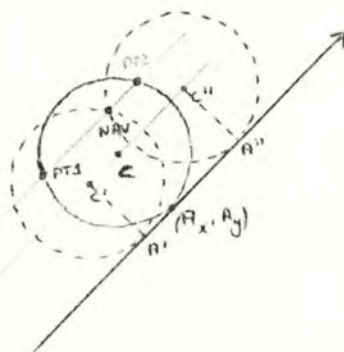
Cette zone hachurée se détermine comme suit :

soit (C_x, C_y) le centre d'un cercle de rayon R_{MIN} (virage à droite)

soit (C'_x, C'_y) le centre situé de l'autre côté de la droite (virage vers la gauche). Tout point situé dans les aires de ces deux cercles est inaccessible par un vol normal.

- 1. Calcul du centre du virage C (voir p. 41)
- 2. Si (la distance entre C et NAV est inférieure à R_{MIN})
 alors procédure de calcul n° 1
 sinon procédure de calcul n° 2

Procédure de calcul n° I (cas 2)



- soit (A_x, A_y) la position de l'avion
- soit NAV la position à atteindre
- soit C le centre d'un virage à rayon minimum dans la direction proposée

La distance (C, NAV) est R_{MIN}

Calcul de la trajectoire

Considérons la droite D définie par l'angle α (angle de la droite de départ) et le point NAV. Cette droite est parallèle à la droite de départ. D coupe le cercle de centre C et de rayon RMIN en deux points PTI et PT2.

Considérons la distance (NAV,PT2) et (NAV,PTI).

Si l'avion entame son virage plus tôt sur la droite de départ d'une distance (NAV,PT2), NAV sera à ce moment situé sur le cercle de son virage (centre C', rayon RMIN).

Si par contre, l'avion entame son virage plus tard sur la droite de départ d'une distance (NAV,PTI), NAV sera à nouveau situé sur le cercle de son virage (centre C'', rayon RMIN).

Choix de la trajectoire : Si l'avion peut reculer d'une distance (NAV,PT2), il commence son virage en A', sinon il commencera son virage en A''

Résultats :

$$(B_x, B_y) = \text{NAV}$$

Calcul de β

soit γ = angle de la droite passant par les points C' (C''), NAV

si virage à gauche : $\beta = \gamma + 90^\circ$ (-360° si $\beta \geq 360^\circ$)

si virage à droite : $\beta = \gamma - 90^\circ$ ($+360^\circ$ si $\beta < 0^\circ$)

I segment CANAR droit de longueur (NAV,PTI) ou (NAV,PT2)

(Si segment CANAR précédant droit, à ajouter ou retrancher de ce segment selon le cas)

I segment CANAR courbe

1. sens : SENS

2. rayon : RMIN

3. angle : DIFF (voir p. 43)

Calcul des points d'intersection PTI et PT2 de la droite définie par α et NAV et du cercle de centre C et de rayon RMIN

L'équation d'une droite qui passe par un point NAV et qui fait un angle avec l'axe OX est :

$$y - y_I = (x - x_I) \operatorname{tg}(\alpha)$$

$$y = \operatorname{tg}(\alpha) x - (y_I - x_I \operatorname{tg}(\alpha))$$

Posons $A = \operatorname{tg} \alpha$

$$B = y_I - x_I \operatorname{tg} \alpha$$

Pour trouver les deux points PTI et PT2, je dois résoudre

$$\begin{cases} y = Ax + B & (I) \text{ équation de la droite} \\ (y - C_2)^2 + (x - C_I)^2 - R^2 = 0 & (2) \text{ équation du cercle} \end{cases}$$

J'injecte (I) dans (2)

$$(Ax + B - C_2)^2 + (x - C_I)^2 - R^2 = 0$$

$$(Ax + B)^2 - 2(Ax + B)C_2 + C_2^2 + x^2 - 2xC_I + C_I^2 - R^2 = 0$$

$$A^2x^2 + 2ABx + B^2 - 2AC_2x - 2BC_2 + C_2^2 + x^2 - 2xC_I + C_I^2 - R^2 = 0$$

$$(A^2 + I)x^2 + (2AB - 2AC_2 - 2C_I)x + (B^2 - 2BC_2 + C_2^2 + C_I^2 - R^2) = 0$$

Posons $CA = A^2 + I$

$$CB = 2AB - 2AC_2 - 2C_I$$

$$CC = (B - C_2)^2 + C_I^2 - R^2$$

$$\text{Il vient } PTI_x = \frac{-CB - \sqrt{(CB)^2 - 4CA \cdot CC}}{2 \cdot CA}$$

$$PT2_x = \frac{-CB + \sqrt{(CB)^2 - 4CA \cdot CC}}{2 \cdot CA}$$

$$PTI_y = A \cdot PTI_x + B$$

$$PT2_y = A \cdot PT2_x + B$$

Cas d'exception

1. Le dernier segment CANAR traduit est courbe.

Traitement : avant tout calcul, insérer un segment droit de longueur arbitrairement petite.

2. Le point NAV se trouve sur la circonférence.

Traitement : Le même calcul peut être employé, on obtiendra $PT2 = NAV$, ce qui ne pose aucun problème.

3. Si $\alpha = 90^\circ$ ou 270° alors $TAN(\alpha) = \infty$

Traitement : $PT1_x = NAV_x$

$$PT2_x = NAV_x$$

$$PT1_y = C_y - \sqrt{-(PT1_x - C_x)^2 + RMIN^2}$$

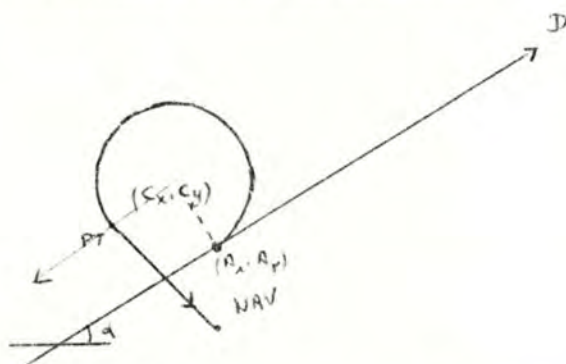
$$PT2_y = C_y + \sqrt{-(PT2_x - C_x)^2 + RMIN^2}$$

4. $CA = 0$; impossible car $CA = A^2 + I$

5. Il y a une contrainte sur le point de départ

Traitement : Message d'erreur et arrêt du programme.

Procédure de calcul n° 2



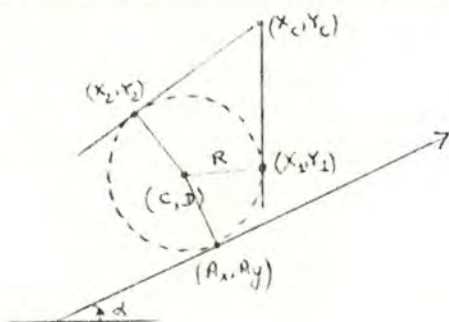
- soit une droite $D ((A_x, A_y),)$
- soit un point $NAV \in C, \in D$
- soit un cercle C de centre (C_x, C_y) et de rayon $RMIN$

déterminer (PT_x, PT_y) , un des deux points de tangence du cercle avec une droite passant par NAV.

Une fois déterminées les coordonnées de ce point PT, il vient la trajectoire CANAR est composée :

d'un segment courbe de rayon RMIN, de sens SENS et d'angle DIFF
d'un segment droit de longueur (PT, RMIN).

Calcul des deux points de tangence PT1 et PT2



Les points (x_I, y_I) et $(x_2, y_2) \in$ droite, $\in$ cercle.

1. équation de la droite $Y = Ax + B$

le point $(x_c, y_c) \in$ droite $\Rightarrow y_c = Ax_c + B$

$$y = Ax - Ax_c + y_c$$

$$y = A(x - x_c) + y_c$$

2. équation du cercle $(x - c)^2 + (y - d)^2 = R^2$

Il faut donc résoudre cette équation en sachant que $y \in$ droite $\Rightarrow$ on remplace y par sa valeur dans l'équation de la droite.

$$(x - c)^2 + (A(x - x_c) + y_c - d)^2 = R^2$$

Il reste une inconnue A . Une fois fixée la valeur de A , l'équation admet une solution unique (car la droite est tangente au cercle) : réécrivons l'équation en fonction des puissances de x ($Ax^2 + Bx + C = 0$)

$$x^2 - 2xc + c^2 + (Ax - Ax_c + y_c - d)^2 = R^2$$

$$\left[A^2 + I\right] x^2 + \left[-2(c + A(Ax_C - y_C + d))\right] x + \left[(-Ax_C + y_C - d)^2 - R^2 + c^2\right] = 0$$

Le discriminant de cette équation est nul (car une seule solution en x) $b^2 - 4ac = 0$

$$\left[-2(c + A(Ax_C - y_C + d))\right]^2 - 4\left[A^2 + I\right]\left[(-Ax_C + y_C - d)^2 - R^2 + c^2\right] = 0$$

Il faut réécrire l'équation avec A comme variable

$$4(A^2 x_C + c + A(d - y_C))^2 - 4[A^2 + I]\left[A^2 x_C^2 - 2Ax_C(y_C - d) + (y_C - d)^2 - R^2 + c^2\right] = 0$$

$$(A^2 x_C + c + A(d - y_C))^2 - (I + A^2)(A^2 x_C^2 - 2Ax_C(y_C - d) + (y_C - d)^2 - R^2 + c^2) = 0$$

$$A^4 x_C^2 + (c + A(d - y_C))^2 + 2A^2 x_C(c + A(d - y_C)) - A^2 x_C^2 + 2Ax_C(y_C - d) - (y_C - d)^2$$

$$+ R^2 - c^2 - A^4 x_C^2 + 2A^3 x_C(y_C - d) - A^2(y_C - d)^2 + A^2 R^2 - A^2 c^2 = 0$$

$$c^2 + A^2(d - y_C)^2 + 2cA(d - y_C) + 2A^2 x_C c + 2A^3 x_C(d - y_C) - A^2 x_C^2 + 2Ax_C(y_C - d)$$

$$- (y_C - d)^2 + R^2 - c^2 + 2A^3 x_C(y_C - d) - A^2(y_C - d)^2 + A^2 R^2 - A^2 c^2 = 0$$

$$\left[2x_C c - x_C^2 - c^2 + R^2\right]A^2 + \left[2(d - y_C)(c - x_C)\right]A + \left[R^2 - (y_C - d)^2\right] = 0$$

$$p = \left[2(d - y_C)(c - x_C)\right]^2 - 4\left[R^2 - (x_C - c)^2\right]\left[R^2 - (y_C - d)^2\right]$$

$$A = \frac{(y_C - d)(c - x_C) \pm \sqrt{(y_C - d)^2(c - x_C)^2 - (R^2 - (x_C - c)^2)(R^2 - (y_C - d)^2)}}{R^2 - (x_C - c)^2}$$

Maintenant qu'on a les deux valeurs de A, il suffit de résoudre l'équation en x :

$$PTI_x = \frac{c + A_I(A_I x_C - y_C + d)}{A_I^2 + I}$$

$$PTI_y = A_I(PTI_x - x_C) + y_C$$

$$PT2_x = \frac{c + A_2(A_2 x_C - y_C + d)}{A_2^2 + I}$$

$$PT2_y = A_2(PT2_x - x_C) + y_C$$

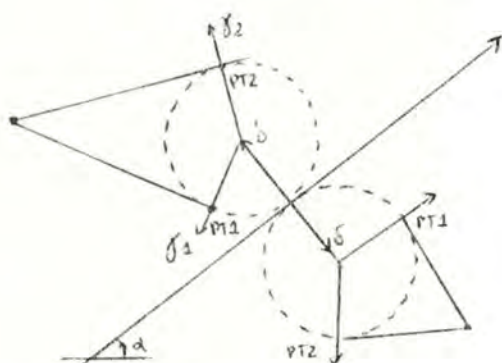
Quel point est le bon de PTI et PT2 ?

Soit γ_1 = l'angle de la droite allant du centre vers PTI

Soit γ_2 = l'angle de la droite allant du centre vers PT2

Soit δ = $- 90^\circ$ si virage à gauche

+ 90° si virage à droite



Deux distinctions sont à faire :

-1. Sens du virage

-2. Position du point à survoler par rapport à la radiale
d'origine (gauche ou droit)

Calculons $\text{DIFF1} = \gamma_1 - \delta$

$\text{DIFF2} = \gamma_2 - \delta$

Si (virage à gauche) et (point à gauche) et ($\text{DIFF1} \neq \text{DIFF2}$) alors $\text{PT} = \text{PT2}$

Si (virage à gauche) et (point à gauche) et ($\text{DIFF1} = \text{DIFF2}$) alors $\text{PT} = \text{PTI}$

Si (virage à gauche) et (point à droite) et ($\text{DIFF1} \neq \text{DIFF2}$) alors $\text{PT} = \text{PTI}$

Si (virage à gauche) et (point à droite) et ($\text{DIFF1} = \text{DIFF2}$) alors $\text{PT} = \text{PT2}$

Si (virage à droite) et (point à droite) et ($\text{DIFF1} \neq \text{DIFF2}$) alors $\text{PT} = \text{PTI}$

Si (virage à droite) et (point à droite) et ($\text{DIFF1} = \text{DIFF2}$) alors $\text{PT} = \text{PT2}$

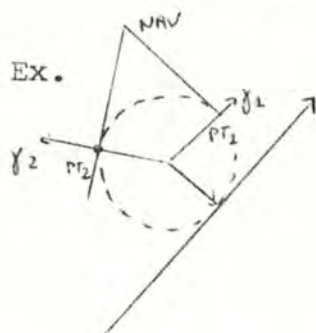
Si (virage à droite) et (point à gauche) et ($\text{DIFF1} \neq \text{DIFF2}$) alors $\text{PT} = \text{PT2}$

Si (virage à droite) et (point à gauche) et ($\text{DIFF1} = \text{DIFF2}$) alors $\text{PT} = \text{PTI}$

Conclusion : le sens du virage importe peu :

Si le point est à gauche de la droite, on prend le point dont la différence est la plus petite.

Si le point est à droite de la droite, on prend le point dont la différence est la plus grande.



$$\delta = 45 - 90 = -45 (+360) = \underline{315}$$

$$\text{DIFF1} = \gamma_1 - \delta = 40 - 315 = -275 (+360) = \underline{85}$$

$$\text{DIFF2} = \gamma_2 - \delta = 135 - 315 = -180 (+360) = \underline{180}$$

$\text{DIFF1} < \text{DIFF2}$ et NAV est à gauche de la droite $\Rightarrow$ le bon point est PTI

Résultats

$$(B_x, B_y) = \text{NAV}$$

Calcul de β

soit γ = angle de la droite passant par C et PT

si virage à gauche : $\beta = \gamma + 90^\circ$ (-360° si $\beta \geq 360^\circ$)

si virage à droite : $\beta = \gamma - 90^\circ$ ($+360^\circ$ si $\beta < 0^\circ$)

I segment CANAR courbe

I. sens : SENS

rayon : RMIN

angle : DIFF (voir p.43)

I segment CANAR droit de longueur (PT,NAV)

Cas d'exception

I. Le dernier segment CANAR traduit est courbe.

Traitement : avant tout calcul, insérer un segment droit de longueur arbitrairement petite.

2 MODULE PLOTTING

Les calculs effectués par ce module étant directement dépendants de la programmation et des routines graphiques utilisées, j'ai préféré détailler ceux-ci dans la partie "informatique" de la conception, soit en p. 79

P R E S E N T A T I O N D E S R O U T I N E S

I. SIDCAN

Spécifications

Le programme principal SIDCAN affiche un menu à l'écran.

L'utilisateur a le choix entre

- la traduction d'un texte-SID particulier
- la sortie graphique d'un texte-SID particulier
- sortie du programme

Algorithme

Répéter

- | | |
|---|-------------------|
| -affichage du menu | |
| -lecture de la réponse de l'utilisateur | |
| -selon la réponse de l'utilisateur | -appel TRADUCTION |
| | -appel PLOTTING |

jusqu'à ce que l'utilisateur désire sortir du programme

Principales variables

REPONSE	Character*I	réponse de l'util.	Var. locale
---------	-------------	--------------------	-------------

2. TRADUCTION

Spécifications

La S-R TRADUCTION effectue la traduction d'un texte-SID spécifié par l'utilisateur en une trajectoire CANAR.

Algorithme

- effectuer les initialisations	INITIALISATIONS
- lire un mot du Texte-SID	LIREMOT
- <u>tant que</u> 'on n'a pas tout lu'	
- lire une phrase	LIREPHRASE
- analyser la phrase	ANALYSEPHRASE
- traduire la phrase	TRADUCPHRASE
- <u>fin tant que</u>	
- clôture	CLOTURE

Principales variables

PHRASE	Character*20	Phrase en cours de traduc.	Var. locale
NUMSUB	Integer	Numéro du mouvement de base de la phrase traduite	Var. locale

3. INITIALISATIONSSpécifications

La S-R effectue les initialisations nécessaires au bon fonctionnement de la sous-routine TRADUCTION, c-à-d :

- I. Sur demande de l'utilisateur, initialisation du
 - nom du texte-SID à traduire
 - nom de l'aéroport
 - numéro de la piste de décollage
2. Lecture des fichiers et stockage des données dans des tableaux accessibles aux différentes routines spécialisées (variables COMMON)
3. initialisations nécessaires relatives aux transformations

COORD. GEO. → COORD. CART.

Calculs effectués conformément aux principes expliqués p.

4. Initialisation des structures de données permettant le
calcul et le stockage des résultats -trajectoire CANAR
-graphiques

Principales variables

1. Structures de données externes voir p. 1 (Annexes)

2. Structures de données internes

AIRPORT	Character*15	Nom de l'aéroport	commune	GEN
SIDLAB	Character*10	Libellé du texte-SID à traduire	commune	GEN
RWY	Integer	Numéro de la piste de décollage	commune	GEN
RECIP	Integer	Numéro de la piste de décollage inverse	locale	
DVM	Real*8	Déviations magnétique	commune	GEN
CTRLONG	Character*10	Coordonnées géograph. du centre de l'aéroport	locale	
CTRLAT	Character*10		locale	
RWYLONG	Character*10	Coordonnées géographiques du début de la piste	locale	
RWYLAT	Character*10		locale	
RECLONG	Character*10	Coordonnées géographiques de la fin de la piste	locale	
RECLAT	Character*10		locale	
SID	Character*250	Texte-SID à traduire	commune	INITSCAN
NAVDAT	tableau(1:34, 1:4) de Character*10	Nom, type et c.g. de tous les points de repère	commune	GEN
MASK	Tableau(1:99) de character*20	Masques de toutes les phrases possibles(p.)	commune	INITANAL

TRAJ	tableau(1:99) de integer	Numéro du mouvement de base de chaque masque	commune	INITANAL
DICT	Tableau(1:13) de character*10	mots-clés possibles d'un texte-SID	commune	INITSCAN
X, Y, Z	Real*8	paramètres de conversion	commune	COORD
XN, XE	tableau(1:3) de real*8	coord. géog. → coord. car. selon technique p.		
LASTPT	tableau(0:1) de Real*8	Coord. Cart. de la posi- tion courante de l'avion	commune	TRACK
LASTANG	Real*8	Cap courant de l'avion	commune	TRACK
RWYLENGHT	Real*8	longueur de la piste	commune	TRACK
TRACK	tableau(1:15, 1:2) de Real*8	trajectoire traduite selon format CANAR	commune	TRACK
ITRACK	Integer	Nombre de segments CANAR déjà traduits	commune	TRACK
POINTS	tableau(1:16, 0:1) de real*8	Coord. Cart. des points de début et fin des seg- ments CANAR	commune	TRACK
IPOINT	Integer	Nombre de points déjà stockés	commune	TRACK
CENT	tableau(1:10, 0:1) de real*8	Coord. Cart. des centres des segments courbes	commune	TRACK
ICTR	Integer	Nombre de centres de segments courbes traduits	commune	TRACK
NBNAV	Integer	Nombre courant de points de repères utilisés	commune	TRACK
NAVUSED	tableau(1:30, 1:2) de character*10	noms et types des points de repères utilisés	commune	TRACK

NAVCOORD	tableau (4:34,0:1) de real*8	Coord. Cartés. des points de repère utilisés	commune	TRACK
MAXTRACK	Integer	Nombre maximum de segments CANAR	commune	TRACK
RMIN	Real*8	Rayon minimum du virage	commune	TRACK

4. LIREPHRASE

Spécifications

La S-r LIREPHRASE effectue la lecture d'une phrase, la fabrication du masque de cette phrase et le stockage des paramètres nécessaires à la traduction de cette phrase.

Algorithme

Tant que (il y a encore des mots à lire) ET (on n'a pas lu un dé-
limiteur de phrase)

- Si le mot lu est un nombre alors stocker le nombre
- Si le mot lu est un identificateur alors stocker
l'identificateur
- coder le mot-lu et le rajouter au masque
- lire un mot

Fin tant que

Principales variables

PHRASE	Character*20	Masque de la phrase en cours de lecture	paramètre
TNOMBRE	tableau (4:10) de Real*8	table des nobres se trouvant dans la phrase traduite	commune TRADUC

TIDENTIF	tableau (1:10) de character*10	Table des identificateurs se trouvant dans la phrase traduite	commune	TRADUC
----------	-----------------------------------	---	---------	--------

5. LIREMOT

Spécifications

La S-R LIREMOT effectue la lecture du prochain mot sur le support d'entrée et renvoie le type du symbole de base (TYPMOT) et éventuellement sa valeur (MOTLU,NBRELU).

Algorithme

- sauter les caractères ' '
- Si le caractère lu est un point alors typmot = 'point'
sinon si le caractère lu est une lettre alors
 - lire MOTLU
 - si MOTLU est dans DICT alors typmot = 'mot-réservé'
sinon typmot = 'identificateur'
 - sinon si le caractère lu est un chiffre alors
 - lire NBRELU
 - typmot = 'nombre'
 - sinon erreur et arrêt du programme

Principales variables

SID (voir p.63)

PS	Integer	Pointeur vers caractère courant du SID	commune	INITSCAN
TYPMOT	Integer	Type du symbole de base lu	commune	SCAN

MOTLU	Character*10	Identificateur lu	commune	SCAN
NBRELU	Real*8	Nombre lu	commune	SCAN
DICT	(voir p. 63)			

6. ANALYSEPHRASE

Spécifications

La S-R ANALYSEPHRASE détermine à partir du masque d'une phrase-SID PHRASE le mouvement de base correspondant NUMSUB

Algorithme

MASK(nbrephrases) = PHRASE

I = I

Tant que (MASK(I) = PHRASE) I=I+1 Fin tant que

Si (I=nbrephrases) alors erreur et arrêt du programme

NUMSUB = TRAJ(I)

Variables principales

MASK voir p. 63

TRAJ voir p. 63

NBREPHRASES	integer	Nombre de masques possibles	var. locale
PHRASE	Character*20	Masque de la phrase en cours d'analyse	paramètre (Arg)
NUMSUB	integer	Numéro du mouvement de base correspondant à PHRASE	paramètre (Rés)

7. TRADUCPHRASE

Spécifications

La S-R TRADUCPHRASE effectue la traduction d'une phrase-SID

dont le mouvement de base est NUMSUB, les paramètres de mouvement étant stockés dans TNOMBRE et TIDENTIF.

Algorithme

Avant d'appeler la sous-routine correspondante à la valeur de NUMSUB, TRADUCPHRASE effectue les conversions nécessaires (par appel des S-R correspondantes).

1. CONVNAV

conversion (coord. géo. $\rightarrow$ coord. cart.) et stockage dans NAVUSED et NAVCOORD des points de repère utilisés.

2. CONVCAP

conversion cap magnétique $\rightarrow$ cap trigonométrique selon la formule p.34

3. CONVDIR

conversion du sens du virage selon la convention :

RIGHT I

LEFT -I

4. CONVKM

conversion NM $\rightarrow$ KM des distances

Principales variables

NUMSUB	integer	Numéro du mouvement de base	paramètre (arg)
		1. Segment droit	
		2. Radiale sans contr.	
		3. Radiale avec contr.	
		4. Survol sans contr.	

5. Survol avec contr.
6. Sturn sans contr.
7. Sturn avec contr.
8. Cap sans contr.
9. Cap avec contr.

TNOMBRE	voir p. 65		
TIDENTIF	voir p. 65		
CONTPD	logical	variable logique indiquant s'il y a contrainte sur le point de départ	var. locale
CP	Real*8	Cap à atteindre (par. d'un M.B)	var. locale
NAV	tableau(0:1) de real*8	Point de repère à survoler (paramètre d'un M.B.)	var. locale
SENS	Integer	Sens du virage (par. d'un M.B.)	var. locale

8. STRAIGHT

Spécifications

La S-R STRAIGHT assure la traduction du mouvement de base
"Droite"

Algorithme

- Si (le M.B est un décollage) alors DIST = DIST - RWYLENGHT
- Calcul du point d'arrivée du segment CANAR correspondant
- Sauvetage du segment droit et m-à-j des variables globales
(S.R. SAVESTRAIGHT)

Principales variables

DIST	Real*8	Distance à parcourir (KM)	paramètre (arg)
------	--------	---------------------------	-----------------

RWYLENGHT	Real*8	Longueur de la piste (KM)	commune GEN
ENDPT	tableau(0:4) de Real*8	Point d'arrivée du segment droit	var. locale

9. RADIALE

Spécifications

La S-R RADIALE effectue la traduction du mouvement de base RADIALE conformément aux principes de calcul expliqués p.42

Algorithme

- calcul de l'angle du virage DIFF DIFFANG
- si (il y a contrainte sur le pt de dép.) alors calcul-I
sinon calcul-2
- calcul du centre du virage C CTRVIR
- calcul de la fin du virage PTFIN FINVIR
- stockage du segment courbe CANAR et m-à-j SAVECURVE
des variables globales
- calcul du segment droit et stockage, m-à-j SAVESTRAIGHT
des variables globales
- CALCUL-I

Si (le point d'intersection I se trouve derrière l'avion) alors

Si (l'angle du virage < 180°) alors

effectuer le mouvement de base Sturn

sinon

calcul du rayon du virage R

si R < RMIN alors erreur !

sinon

Si (l'angle du virage $> 180^\circ$) alors erreur !

calcul du rayon du virage R RAYON

si ($R < R_{MIN}$) alors erreur !

- CALCUL-2

R = RMIN

Si (le point d'intersection I se trouve derrière l'avion) alors

mouvement de base RADIALE avec contrainte

sinon

si (l'angle du virage $> 180^\circ$) alors erreur !

calcul du point de départ du virage et modification

éventuelle du dernier segment CANAR PTDEPART

si modification impossible alors

mouvement de base RADIALE avec contrainte

Principales variables

CONTPD	logical	Variable logique indiquant s'il y a contrainte sur le pt de dép.	paramètre (arg)
SENS	Integer	Sens du virage	paramètre (arg)
DIFF	Real*8	Angle du virage	var. locale
BETA	Real*8	Cap à atteindre	paramètre (arg)
LASTPT	tableau(0:4) de real*8	Position courante de l'avion	commune TRACK
LASTANG	Real*8	Cap courant de l'avion	commune TRACK
NAVAID	tableau(0:4) de real*8	Point de repère à survoler	paramètre (arg)
R	Real*8	Rayon du virage (KM)	var. locale

C	tableau(0:1) de real*8	Centre du virage (KM)	var. locale
PTFIN	tableau(0:1) de real*8	Fin du virage (KM)	var. locale

IO. SURVOL

Spécifications

La S-R SURVOL effectue la traduction du mouvement de base SURVOL conformément aux principes de calcul expliqués p. 51

Algorithme

- Calcul du centre du virage C CTRVIR
- Calcul de DIST = C - NAVAID DIST
- Si (DIST $\leq$ RMIN)
 - alors calcul-1
 - sinon calcul-2

CALCUL-1

- Calcul des deux points d'intersection PT1, PT2 de la droite (NAVAID, LASTANG) avec le cercle du virage. INTERSECT
- Calcul de la distance (NAVAID, PT1) (NAVAID, PT2) DIST
- Choix du pt PT et modification du point de départ
- Calcul du nouveau cap NEWANG
- Calcul de l'angle du virage DIFF DIFF
- Stockage du segment courbe CANAR et m-à-j des variables globales SAVECURVE

CALCUL-2

- Calcul des deux points PT1 et PT2 de tangence du cercle

avec une droite passant par NAVAID

- Choix du bon point PT
- Calcul du nouveau cap NEWANG
- Calcul de l'angle du virage DIFF DIFF
- Stockage du segment courbe CANAR et m-à-j
des variables globales SAVECURVE
- Calcul de la distance (PT, NAVAID) DIST
- Stockage du segment droit CANAR et m-à-j
des variables globales STRAIGHT

Principales variables

NAVAID	tableau (0 4) of real*8	point à survoler	paramètre (Arg)
SENS	Integer	Sens du virage	paramètre (Arg)
C	tableau (0 1) of real*8	Centre du virage	paramètre (Arg)
LASTPT	tableau (0 1) of real*8	Position courante de l'avion	commune TRACK
LASTANG	Real*8	Cap courant de l' avion	commune TRACK
PT1, PT2	tableau (0 1) of real*8	Points du cercle du virage	var. locale
PT	tableau (0 1) of real*8	Point de fin du virage	var. locale
DIFF	Real*8	Angle du virage	var. locale
NEWANG	Real*8	Nouveau cap de l' avion	var. locale

II. STURN

Spécifications

La sous-routine STURN assure la traduction du mouvement de base STURN conformément aux principes de calcul p.41

Algorithme

- contrôle-I
- contrôle-2
- calcul du cap intermédiaire GAMMA
- appel de la S-R CAP avec paramètres :
(GAMMA, SENS)
- appel de la S-R RADIALE avec paramètres
(BETA, NAVAIID, -SENS, .FALSE.)

Contrôle I

(contrôle si le sens du virage est conforme au cap à atteindre)

Si (LASTPT est situé à gauche de la droite (NAVAID, BETA)

Alors S = I

Sinon S = -I

Si (SENS = -S) alors erreur !

Contrôle 2

(contrôle si le STURN est possible)

- calcul du point d'arrivée du second virage
- Si (les point de départ et d'arrivée ne sont pas
situés du même coté de la droite d'arrivée)
alors erreur !

Principales variables

SENS	Integer	Sens du premier virage	paramètre (Arg)
BETA	Real*8	Cap final à atteindre	paramètre (Arg)
NAVAID	tableau(0:1) de real*8	Point à survoler	paramètre (Arg)
GAMMA	Real*8	Cap intermédiaire à atteindre	var. locale

I2. CAP

La sous-routine CAP effectue la traduction du mouvement de base CAP selon les calculs expliqués p. 40

Algorithme

- Créer un segment droit CANAR de longueur arbitrairement petite (si nécessaire) INSERERDR
- Calcul de l'angle du virage DIFF DIFF
- Calcul du centre du virage C CTRVIR
- Calcul de la fin du virage NEWPT FINVIR
- Stockage du segment courbe CANAR et m-à-j des variables globales SAVECURVE

Principales variables

BETA	Real*8	Cap final à atteindre	Paramètre (Arg)
SENS	integer	Sens du virage	Paramètre (Arg)
LASTANG	Real*8	Cap courant de l'avion	commune TRADUC
LASTPT	tableau(0:1) de real*8	Position courante de l'avion	commune TRADUC

C	tableau(0:1) de Real*8	Centre du virage	Var locale
NEWPT	tableau(0:1) de Real*8	Point de fin du virage	Var. locale
RMIN	Real*8	Rayon minimum	commune TRADUC
DIFF	Real*8	Angle du virage	Var. locale

I3. SAVECURVE

Spécifications

La S-R SAVECURVE effectue le stockage

1. des éléments constitutifs d'un segment CANAR courbe
selon le format CANAR (voir p. 15)

2. d'informations additionnelles :

Centre du virage en C.C. et NM

Fin du segment courbe

et la m-à-j des variables globales LASTPT et LASTANG.

Algorithme

- 1er élément du TRACK = ANGLE
- 2nd élément du TRACK = SENS*RAYON (converti en NM)
- LASTPT = ENDPOINT
- LASTANG = ENDANG
- Stocker CTR dans le tableau CENT
- Stocker ENDPOINT dans le tableau POINT

Variables principales

RAYON	Real*8	Rayon du virage (KM)	paramètre (Arg)
ANGLE	Real*8	Angle du virage	paramètre (Arg)

SENS	Integer	Sens du virage	Paramètre (Arg)
ENDPOINT	tableau (0:4) de Real*8	Fin du virage	Paramètre (Arg)
ENDANG	Real*8	Cap final	Paramètre (Arg)
CTR	tableau (0:4) de Real*8	Centre du virage	Paramètre (Arg)
TRACK	Voir p. 63		
LASTPT	tableau (0:4) de Real*8	Position courante de l'avion	commune TRADUC
LASTANG	Real*8	Cap courant de l'avion	commune TRADUC
CENT	Voir p. 63		
POINT	Voir p. 63		

I4. SAVESTRAIGHT

Spécifications

La S-R SAVESTRAIGHT effectue le stockage

1. des éléments constitutifs d'un segment CANAR droit
selon le format CANAR (voir p. 16)

2. d'informations additionnelles :

Fin du segment droit

et la m-à-j des variables globales LASTPT et LASTANG.

Algorithme

Si (le dernier segment CANAR stocké est un segment droit)

alors - convertir DIST en NM

- prolonger le dernier segment traduit de DIST

sinon - créer un nouveau segment CANAR

1er élément = DIST (converti en NM)

2nd élément = 0

- stocker ENDPOINT dans le tableau POINTS

- LASTPT = ENDPOINT

Variables principales

DIST	Real*8	Distance à stocker	paramètre (Arg)
ENDPOINT	tableau(0:1) de Real*8	Point d'arrivée du segment droit	paramètre (Arg)
TRACK	voir p. 63		
POINTS	voir p. 63		
LASTPT	tableau(0:1) de Real*8	Position courante de l'avion	commune TRADUC
LASTANG	Real*8	Cap courant de l'avion	commune TRADUC

I5. PLOTTING

Spécifications

La S-R PLOTTING effectue le tracé sur écran graphique

- de la trajectoire CANAR désignée par l'utilisateur
- de certains renseignements complémentaires :
 - nom de la trajectoire
 - nom de l'aéroport
 - piste de décollage

Algorithme

La sortie graphique des résultats fait appel aux routines graphiques suivantes : INGRAF, CLS, WINDOW, VIEWPORT, CIRCLE, VECT, POSIT, INK, CLIPPER, TEXT, EXGRAF. Les spécifications de ces routines sont données en Annexes p. 7

L'utilisation de ces routines donnent lieu à un certain nombre de calculs préalables.

Détermination du système d'axe utilisateur

Un point sur l'écran peut être spécifié en donnant ses coord. cartés. Il y a deux systèmes de coord. cartés. :

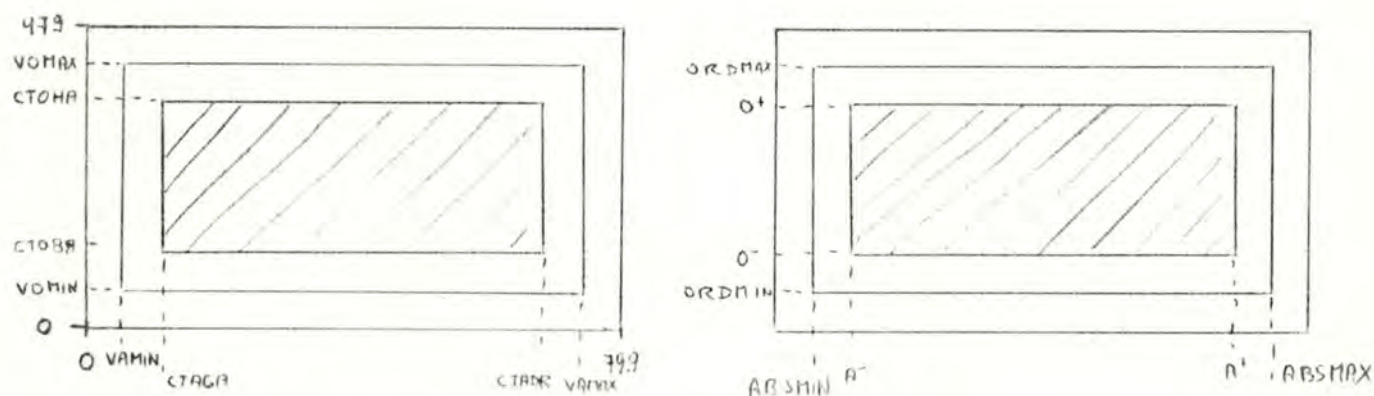
Le système de C.C. "écran". Il est fixe et est lié aux caractéristiques de l'écran. Dimensions : $0 \leq x \leq 799$, $0 \leq y \leq 479$ avec x et y comme entrées.

Le système de C.C. "utilisateur". C'est celui dans lequel sont exprimées les coordonnées des points à représenter.

Dans SIDCAN, le système est en KM avec comme centre du plan, le centre de l'aéroport.

Le système d'axes "utilisateur" est spécifié par la S-R WINDOW. Cette S-R demande les valeurs min. et max. des abs. et ord. Ces valeurs seront associées avec les valeurs min. et max. des abs. et ord. du système de coordonnées "écran" (VAMIN, VAMAX, VOMIN, VOMAX). Ces valeurs sont déclarées par l'intermédiaire de la S-R VIEWPORT.

Correspondance Syst. écran Syst. utilisateur



Il existe trois zones :

- zone 1 : 0..799, 0..479 l'écran complet
- zone 2 : VAMIN..VAMAX, VOMIN..VOMAX
la partie de l'écran accessible aux routines graphiques
- zone 3 : CTAGA..CTADR, CTOBA..CTOHA
la partie de l'écran effectivement utilisée
pour dessiner la trajectoire

La zone 2 est spécifiée par l'instruction 'VIEWPORT'

Dans SIDCAN, la totalité de l'écran est accessible aux routines graphiques CALL VIEWPORT (0,799,0,479)

La sous-routine WINDOW permet d'associer aux valeurs (VAMIN, VAMAX, VOMIN, VOMAX) les valeurs correspondantes en

système "utilisateur".

Soient ABSMIN, ABSMAX, ORDMIN, ORDMAX les valeurs à déclarer.

I. Calcul de A^- , A^+ , O^- , O^+

a) Tous les points de la trajectoire doivent figurer dans ce rectangle. On prendra donc pour valeurs de A^- , A^+ , O^- , O^+ les valeurs minimum et maximum (abs et ord) de tous les points à représenter. Ceci peut être fait assez simplement par une simple boucle sur tous les points (TABLEAUX : POINTS, CENT, et NAVUSED + RWYCAR, RECCAR)

b) Pour éviter toute distorsion dans les tracés, il faut que le rapport longueur/largeur soit le même dans les deux systèmes de coordonnées. Soit $RAPFEN = (CTADR - CTAGA) / (CTOHA - CTOBA)$ ce rapport dans le système "écran". Il faut donc adopter le même rapport pour le système utilisateur.

Si $\frac{A^+ - A^-}{O^+ - O^-} < RAPFEN$ alors il faut augmenter la longueur
longueur = longueur $RAPFEN$
sinon il faut augmenter la largeur
largeur = largeur/ $RAPFEN$

On conviendra de répartir de façon égale à gauche et à droite (en haut et en bas) l'excédent de longueur (de largeur).

Ex. syst. écran $RAPFEN = 4/3$

syst. utilisateur $A^- = 0$, $A^+ = 100$, $O^- = 50$, $O^+ = 150$

$RAP = 1 < RAPFEN$

longueur = largeur $4/3 = 133.33$

à répartir $133.33 - 100 = 33.33$

$A^- = A^- - \frac{33}{2}$, $A^+ = A^+ + \frac{33}{2}$

2. Calcul de ABSMIN, ABSMAX, ORDMIN, ORDMAX

En plus du tracé de la trajectoire, l'écran doit encore contenir un certain nombre d'informations (Libellé du SID, nom de l'aéroport...).

Il faut donc encore élargir A^- , A^+ , O^- , O^+ pour obtenir ABSMIN, ABSMAX, ORDMIN, ORDMAX

Dans le système écran, la différence est de

CTAGA - VAMIN	Abcisse à gauche	AG
VAMAX - CTADR	Abcisse à droite	AD
CTOBA - VOMIN	Ordonnée en bas	OB
VOMAX - CTOHA	Ordonnée en haut	OH

Il suffit de transposer ces différences dans le système utilisateur.

$$\text{Soit le rapport RAPP} = \frac{\text{CTADR} - \text{CTAGA}}{A^+ - A^-}$$

On a :

$$\text{ABSMIN} = A^- - \text{AG}/\text{RAPP}$$

$$\text{ABSMAX} = A^+ + \text{AD}/\text{RAPP}$$

$$\text{ORDMIN} = O^- + \text{OB}/\text{RAPP}$$

$$\text{ORDMAX} = O^+ + \text{OH}/\text{RAPP}$$

ALGORITHME

- initialiser routines graphiques	<u>INGRAF</u>
- effacer l'écran	<u>CLS</u>
- afficher cadre	<u>CLIPPER</u>
- afficher titres	<u>TEXT</u>
- afficher trajectoire	<u>POSIT, VECT, CIRCLE</u>

- afficher aides à la navigation TEXT
- sortir routines graphiques EXGRAF

afficher trajectoire

- se positionner sur la fin de piste POSIT
- afficher le premier segment droit VECT
- tant qu'il y a encore des segments à afficher
 - afficher un segment courbe CIRCLE
 - afficher un segment droit POSIT, VECT

fin tant que

Variables principales

TRACK	voir p. 63		
POINTS	voir p. 63		
CENT	voir p. 63		
ITRACK	Integer	Nombre de segments CANAR	commune TRACK
IPOINT	Integer	Nombre de points (=ITRACK+1)	commune TRACK
ICENT	Integer	Nombre de centres	commune TRACK
VAMIN, VAMAX	Integer	Coordonnées de la fenêtre	var locale
VOMIN, VOMAX		(système "écran")	
ABSMIN, ABSMAX	Real*8	Coordonnées de la fenêtre	var locale
ORDMIN, ORDMAX		(système "utilisateur")	
RAPFEN	Real*8	Rapport longueur/largeur	var locale
RAPP	Real*8	Rapport Coord écran / Coord utilisateur	var locale
AIRPORT	Character*15	Nom de l'aéroport	commune GEN
SIDLAB	Character*10	Nom du SID	commune GEN
RWY	Integer	N° de la piste de décollage	commune GEN

E V O L U T I O N D U P R O D U I T

Possibilités d'extension

Les mouvements de base traduits par le programme SIDCAN ne représentent pas l'ensemble des mouvements de base possibles d'une trajectoire. De même, il existe un certain nombre de mots-clé non repris dans le dictionnaire des mots-clés (DICT.DAT)

Il peut donc être intéressant de vouloir modifier le programme et/ou les fichiers de façon à augmenter les possibilités du programme SIDCAN.

L'utilisation de la technique des "masques" permet une évolution assez facile du produit. Ces facilités d'évolution ne concernent que la partie "syntaxique" de la traduction, les routines spécialisées de traduction devant être réécrites ou modifiées le cas échéant.

- I) on désire modifier la syntaxe d'un mouvement de base
- par l'ajout de nouveaux mots-clés

Il suffit de modifier le fichier DICT.DAT de façon à autoriser le décodage par la procédure LIREMOT de ces mots.

- par l'ajout d'un nouveau type de symbole de base

Il faut attribuer à ce nouveau type un code non encore utilisé (dans SIDCAN, codes de 'V' à 'Z')

Il faut modifier la S-R LIREMOT pour lui permettre de reconnaître ce nouveau type

Il faut modifier la S-R LIREPHRASE pour pouvoir stocker la valeur de ce type.

Il faudra peut-être rajouter un tableau TXXX pour recueillir ces valeurs.

2) On désire ajouter un mouvement de base

- il faut modifier le fichier MASK.DAT de façon à inclure les différents masques possibles pour ce mouvement.
- attribuer un numéro de trajectoire non encore repris.
(dans SIDCAN, les numéros sont de 1 à 9)
- rajouter une routine spécialisée pour traduire ce mouvement.

Automatisation

la génération des différents masques possibles pour chaque phrase SID peut être automatisée de la manière suivante :

Soit une grammaire SID décrite en BNF.

Le symbole distingué est $\langle \text{phrase} \rangle$

La première production de la grammaire est

$\langle \text{PHRASE} \rangle ::= \langle P1 \rangle | \langle P2 \rangle | \langle P3 \rangle | \dots | \langle Pn \rangle$

Chaque alternative correspond à un type de mouvement de base.

Pour pouvoir générer l'ensemble des phrases possibles pour chaque type de mouvement de base, il faut que la grammaire possède certaines propriétés.

Soit $S_1 = \{ \text{symbole distingué} \}$

$S_2 = \{ \text{symboles terminaux} \}$, c'est-à-dire l'ensemble des symboles de la grammaire BNF non délimités par les caractères ' $\langle$ ' et ' $\rangle$ '.

$S_3 = \{ \text{symboles non-terminaux sans fils} \} = \text{ensemble des symboles délimités par } \langle \text{ et } \rangle$.

mais qui ne figurent dans aucun membre de gauche d'aucune production.

$S_4 = \{\text{symboles non-terminaux avec fils}\} = \text{ensemble des symboles délimités par '<' et '>' et tel que pour chaque symbole, il existe une production dont ce symbole constitue le membre de gauche.}$

Considérons la relation 'a pour fils' que nous définissons comme suit : SYM_1 a pour fils $\text{SYM}_2 \Leftrightarrow \exists \text{ production } P : \text{SYM}_1 \text{ figure dans le membre de gauche et } \text{SYM}_2 \text{ dans le membre de droite.}$

Si on élabore le graphe de cette relation, la condition pour que la grammaire soit automatisable est que le graphe n'ait pas de circuit. Autrement dit, chaque symbole ne peut être son propre fils, chaque symbole ne peut compter parmi ses fils un de ses ascendants. La grammaire n'est donc pas récursive et l'ensemble des phrases possibles est fini.

Le programme chargé de l'automatisation de cette grammaire fonctionnerait comme suit :

I) Construire S_2, S_3, S_4

L'ensemble S_2 constitue le dictionnaire des mots-réservés classé par ordre alphabétique et sauvé sur le fichier DICT.DAT. Il est utilisable tel quel par SIDCAN

L'ensemble S_3 constitue la grammaire BNF et servira à générer l'ensemble des phrases possibles.

L'ensemble S_4 constitue le dictionnaire des types de symboles de base (autre que mot-réservé)

- 2) Construire une représentation interne de la grammaire.
 Chaque symbole $\in S_2$ est codé par une lettre (de A à Z, $Z \in R$) suivant l'ordre alphabétique.

Chaque symbole $\in S_3$ est codé par une lettre (de R à Z, $Z \in Z$).

Chaque symbole $\in S_4$ est codé par un chiffre (numéro de la production dont S_4 constitue le membre de gauche).

- 3) Construire une procédure récursive GENERERPHRASE

Conventions :

préfixe : chaîne de caractères alphabétiques de longueur 0.

α : un caractère alphanumérique

suffixe : chaîne de caractères alphanumériques de longueur 0.

() : chaîne de caractères vide

(c) : chaîne de caractères contenant le seul caractère c

($c_1.c_2$) : chaîne de caractères résultant de la concaténation de c_1 et c_2

procedure GENERERPHRASE(prefixe, α , suffixe)

Si α est une lettre

alors si suffixe = ()

alors éditer (prefixe. α)

sinon soit suffixe = ($S_1.S_2 \dots S_m$)

GENERERPHRASE(prefixe. α , S_1 , ($S_2 \dots S_m$))

sinon si production = $\alpha_1 | \alpha_2 | \dots | \alpha_n$

alors GENERERPHRASE(prefixe, α_1 , suffixe)

GENERERPHRASE(prefixe, α_2 , suffixe)

.

.

.

GENERERPHRASE(prefixe, α_n , suffixe)

sinon = ($\alpha_1.\alpha_2 \dots \alpha_n$)

GENERERPHRASE(prefixe, α_1 , ($\alpha_2 \dots \alpha_n$..suffixe))

Résultats de l'automatisation

un fichier MASK.DAT = ensemble des phrases générées par
GENERERPHRASE

un fichier DICT.DAT = ensemble des symboles terminaux
= ensemble des valeurs du type de Symbole
de base "mot-réservé"

un fichier COD.DAT = ensemble des codes
(symboles terminaux A → ..)
(symboles non-terminaux sans fils R → ..)

Il ne reste plus qu'à modifier la procédure LIREMOT pour lui
permettre de reconnaître les symboles non-terminaux.

T E S T S E T C O N C L U S I O N

Tests

Les tests de correction des sous-routines ont été réalisées de la manière suivante :
chaque sous-routine spécialisée a été isolée, compilée et incluse dans un programme de test.
Tous les programmes de test (un pour chaque sous-routine testée) possédaient la structure :

- entrer les valeurs pour les arguments de la procédure testée
- appel de la procédure
- écriture des résultats de la procédure
 - les paramètres résultats de la procédure
 - les variables globales qui ont pu être modifiées lors de l'exécution de la procédure

Toute procédure dont les tests avaient prouvé la validité fut incorporée au programme principal.

Celui-ci fut alors exécuté sur base d'un jeu de tests reprenant toutes les phrases-SID possibles. Des exemples d'exécution figurent en Annexes p.

Conclusion

Bien que pouvant être amélioré sur un certain nombre de points (analyse, routines graphiques), je crois avoir rempli, en écrivant ce programme, les conditions que je m'étais imposées en début de mémoire.

J'espère ainsi avoir pu contribuer dans la mesure de mes moyens à une (petite) partie du projet CANAR.

Bibliographie

1. "Adaptation et gestion du programme 'CANAR' de calcul,
de tracé des courbes d'exposition au bruit des aéronefs"
Rapport 2 - Programme utilitaire SIDCAN, O. Henkes,
A. Simonetti.
2. "Les techniques de la navigation aérienne", G. Coutand
et L. Andlauer, Presses universitaires de France, 1978
3. "Structured system programming", Addison-wexley, A. Welch,
1981.

Conception et réalisation
d'un logiciel de calcul et de tracé
de trajectoires aéronautiques
à partir d'instructions de vol SID

ANNEXES

P L A N D E L A N N E X E

Fichiers du programme SIDCAN	I
Déclarations 'COMMON' (variables globales FORTRAN	6
Spécifications des sous-routines graphiques	7
Tests et résultats de SIDCAN	8
Manuel de l'utilisateur	10
Source du programme SIDCAN	12

FICHIERS DU PROGRAMME SIDCAN

Tous les fichiers utilisés par le programme SIDCAN sont des fichiers à organisation et accès séquentiels.

On peut distinguer les fichiers de lecture (entrée du programme) et d'écriture (sortie du programme).

I. Les fichiers en lecture

A. Les fichiers 'programme'

Ces fichiers contiennent des paramètres nécessaires à la traduction des textes-SID. Ces fichiers ne varient pas d'une exécution à l'autre du programme. Ces paramètres sont cependant placés sur support externe pour pouvoir être modifiés aisément (en cas de changement de la grammaire SID).

I) Fichier DICT

Nom DICT.DAT

Contenu L'ensemble des mots réservés classés par ordre alphabétique.

Correspondance Format externe - Structure interne

n enregistrements AIO DICT(I:NBDICT) Character*IO
(n ≤ NBDICT)

Utilisation lecture séquentielle par S-R INITIALISATIONS

Ex.

```

AHEAD
AT
INTERCEPT
LEFT
M
NM
UNTO
RIGHT
THEN
TO
TR
TURN
UNTIL

```


2) Fichier MasqueNom MASK.DATContenu La liste des masques possibles des phrases de la
grammaire SID classés par ordre alphabétiqueCorrespondance Format externe - Structure interne

n enregistrements (A20,I2)	MASK(I:99)	Character*20
(n ≤ 99)	TRAJ(I:99)	Integer

Utilisation lecture séquentielle par S-R INITIALISATIONSEx.

AMTF	01
ATF	01
BUDGKTE	09
BUDGKTEJU	03
BUDJU	05
BUDJCKTEJU	07
BUHGKTE	09
BUHGKTEJU	03
BUHJU	05
BUHJCKTEJU	07
BULGKTE	09
BULGKTEJU	03
BULJU	05
BULDJCKTEJU	07
BULHGKTE	09
BULHGKTEJU	03
BULHJU	05
BULHJCKTEJU	07
DGKTE	08
DGKTEJU	02
DJU	04
DJCKTEJU	06
LDGKTE	08
LHGKTEJU	02
LHJU	04
LHJCKTEJU	06
HGKTE	08
HGKTEJU	02
HJU	04
HJCKTEJU	06
LHGKTE	08
LDGKTEJU	02
LDJU	04
LDJCKTEJU	06

B. Les fichiers 'utilisateur'

Ces fichiers sont créés par l'utilisateur avant toute exécution du programme SIDCAN.

Les trois premières lettres de ces fichiers correspondent aux trois premières lettres du nom de l'aéroport concerné.

I) Fichier AEROPORT

Nom XXXAIR.DAT

Contenu Tous les paramètres concernant l'aéroport dont les trois premières lettres sont XXX :

- déclinaison magnétique
- Coordonnées géographiques du centre de l'aéroport
- Nombre de pistes
- Pour chaque piste :
 - numéro de la piste
 - Coordonnées géographiques du début de la piste

Correspondance Format externe - Structure interne

1er enregistrement	F4.I	DVM	Real*8
2ème	"	AIO, IX, AIO	CTRLAT Real*8
			CTRLONG Real*8
3ème	"	I2	NUMRNY Integer
NUMRNY enregistre-	I2, IX, 2 (AIO, IX)	RWYN (0:36)	Integer
ments suivants		LAT (0:36)	Character*10
		LONG (0:36)	Character*10

II) Utilisation lecture séquentielle par S-R INITIALISATIONS

Ex.

```
-2.0
N001:00:42 W000:11:40
2
00 N001:00:30 W000:13:12
20 N001:00:54 W000:11:24
```


Ex. p. 8

2. Les fichiers en écriture

Les résultats de la traduction sont stockés sur deux fichiers. Le premier reprend strictement les résultats officiels nécessaires à l'utilisation du programme CANAR. L'autre fichier contient des informations facilitant le tracé des résultats. Si on change le logiciel graphique, il suffit de changer l'organisation de ce fichier et on pourra garder intact le premier fichier (XXXCAN.PLO).

Les deux fichiers sont cependant nécessaires à l'élaboration des trajectoires graphiques. Afin d'éviter toute erreur, au début de chaque fichier sont repris les renseignements: Nom du SID, N° de la piste de décollage.

I) Fichier CANAR

Nom XXXCAN.DAT

Contenu Ce fichier contient la trajectoire d'un texte SID dans le format imposé par le programme CANAR :

- segments droits et courbes
- coord. cart. des aides à la navigation utilisées
- libellé du SID
- Coord. cart. des points de début et de fin de la piste utilisée
- numéro de la piste utilisée

Correspondance Format externe - Structure interne

1er article	AIO	SIDLAB	Character*10
2ème article	I2	RWY	Integer
3ème article	4 (IX, FI5.8)	RWYCAR (0:1)	Real*8
		RECCAR (0:1)	Real*8
4ème article	I2	ITRACK	Integer
ITRACK articles	(FI5.8, IX, FI5.8)	TRACK (I:16, I:2)	Real*8
(ITRACK+5) ^e article	I2	NBNAV	Integer
NBNAV articles	(2 (AIO, IX), 2 FI5.8)	NAVUSED (I:3I, I:2)	Character*10
		NAVCOORD (I:3I, 0:1)	Real*8

Ex. p 8

2) Fichier Sortie graphique

Nom XXXPLO.DAT

Contenu Ensemble des paramètres utiles à l'élaboration d'une trajectoire graphique :

- les points de début et fin des segments CANAR
- Les centres des segments courbes CANAR

Correspondance Format externe - Structure interne

1er article	AIO	SIDLAB	Character*10
2ème article	I2	IPOINT	Integer
IPOINTS articles	2 FI5.8	POINTS (I:16, 0:1)	Real*8
(IPOINTS+3) ^e article	I2	ICTR	Integer
ICTR articles	2 FI5.8	CENT (I:10, 0:1)	Real*8

Ex. p 8

COMMON /TRADUC/ TIDENTIF(1:10),TNOMBRE(1:10)

CHARACTER*10 TIDENTIF
REAL*8 TNOMBRE

1 COMMON /TRACK/ TRACK(1:15,1:2),ITRACK,MAXTRACK,LASTANG,
1 LASTPT(0:1),POINTSC(1:15,0:1),IPOINT,
1 CENT(1:10,0:1),ICTR,RMIN,NBNAV,RWYLENGHT,
NAVUSED(1:30,1:2),NAVCOORD(1:31,0:1)

CHARACTER*10 NAVUSED
1 REAL*8 TRACK,LASTANG,LASTPT,POINTS,CENT,RMIN,NAVCOORD,
RWYLENGHT
1 INTEGER ITRACK,MAXTRACK,IPOINT,ICTR,NBNAV

COMMON /SCAN/ TYPMOT,NBRELU,MOTLU

REAL*8 NBRELU
CHARACTER*10 MOTLU
INTEGER TYPMOT

1 COMMON /GEN/ AIRPORT,SIDLAB,RWY,DVM,NAVDAT(1:31,1:4),
RWYCAR(0:1),RECCAR(0:1)

CHARACTER*15 AIRPORT
CHARACTER*10 SIDLAB,NAVDAT
INTEGER RWY
REAL*8 DVM,RWYCAR,RECCAR

COMMON /COORD/ XNC(1:3),XSC(1:3),X,Y,Z

REAL*8 XN,XS,X,Y,Z

COMMON /WIN/ TABWIN

REAL TABWIN(4)

COMMON /VIEW/ TABVIEW

INTEGER TABVIEW(4)

COMMON /PIC/ TABNAME,TABPICK,TABPICY,TABNSP

CHARACTER*5 TABNAME(16)

REAL TABPICK(16,16),TABPICY(16,16)

COMMON /INITANAL/ MASK(1:99),TRAJ(1:99)

CHARACTER*20 MASK
INTEGER TRAJ

COMMON /INITSCAN/ SID,DICT(1:13),PS

INTEGER PS
CHARACTER*250 SID
CHARACTER*10 DICT

subroutine INGRAF

action: fait entrer dans reGIS

subroutine CLS

action: efface l'ecran

subroutine EXGRAF

action: exit reGIS

subroutine POSIT(x,y)

action: positionne le curseur aux coordonnees (x,y)

arguments:
x,y: real

subroutine WINDOW(xmin,xmax,ymin,ymax)

action: definir une fenetre (vue de l'utilisateur) limitee par
(xmin,ymin) le bord inferieur gauche et
(xmax,ymax) le bord exterieur droite .

arguments:
xmin,xmax,ymin,ymax: real

subroutine VIEWPORT(amin,amax,omin,omax)

action: definir une cloture (zone de l'ecran) limitee par
(amin,amax) le bord inferieur gauche et
(omin,omax) le bord superieur droite .

arguments:
amin,amax: [0..79,1]
omin,omax: [0..47,1]

subroutine CLIPPER(x1,y1,x2,y2)

action: decoupe le segment de droite de coordonnees (x1,y1,x2,y2)
a travers la fenetre courante.
affiche la partie du segment inclus dans la fenetre

argument:
x1,y1,x2,y2: real

subroutine TEXT(string)

action: affiche la chaine de caracteres a partir de la position
du curseur

arguments:
string: string,128

subroutine VECT(x,y)

action: affiche le segment de droite determine par la position
actuelle du curseur et le point (x,y)

arguments:

x,y: real

subroutine CIRCLE(xc,yc,x,y,arcdeg,strigo)

action: affiche l'arc de cercle de centre (xc,yc)
(x,y) est un point sur la circonference
(arcdeg) degre de l'arc
(strigo) sens du trace

arguments:
xc,yc,x,y: real
arcdeg: [0..360]
strigo: boolean

subroutine INK(valink)

action: determine l'intensite du trace.

valink=0 -> noir
valink=1 -> gr
valink=2 -> blanc
valink=3 -> blanc clair

arguments:

valink: 0..3

26 S
 AHEAD UNTIL 3 NM THEN RIGHT ONTO TR 085 M TO SEVENNOAKS . AT SEVENNOAKS
 LEFT ONTO TR 015 M TO HORNCURCH THEN LEFT ONTO TR 322 M TO BPK
 26 S1
 AHEAD UNTIL 3 NM THEN RIGHT TO BPK THEN AHEAD UNTIL 3 NM
 26 S2
 AHEAD UNTIL 5 NM THEN RIGHT ONTO TR 085 M . LEFT TO BPK
 26 S3
 AHEAD UNTIL 15 NM THEN LEFT TO INTERCEPT TR 085 M TO SEVENNOAKS
 26 S4
 AHEAD UNTIL 5 NM THEN RIGHT ONTO TR 085 M THEN LEFT TO SEVENNOAKS
 26 S5
 AHEAD UNTIL 3 NM THEN RIGHT TO SEVENNOAKS THEN LEFT TO HORNCURCH THEN
 RIGHT TO BPK THEN LEFT TO SEVENNOAKS
 26 S6
 AHEAD UNTIL 5 NM THEN LEFT TO SEVENNOAKS . AT SEVENNOAKS LEFT ONTO TR 015 M TO
 THEN RIGHT TO INTERCEPT TR 224 M TO BPK
 26 S7
 AHEAD UNTIL 10 NM THEN LEFT ONTO TR 333 M THEN RIGHT ONTO TR 228 M

6
 1.63294545 0.37173053 -1.63318081 -0.37121312

9
 1.19253421 0.00000000
 185.81523646 2.21895655
 16.04773331 0.00000000
 70.00000000 -0.5310515
 14.82437420 0.00000000
 98.00000000 -0.26980649
 0.05396130 0.00000000
 45.00000000 6.47332232
 16.25584984 0.00000000

3
 SEVENNOAKS MARKER 27.58370834 11.53976755
 HORNCURCH MARKER 34.51373419 39.12334087
 BPK MARKER 6.56241261 67.05928494

1
 0
 1.63294545 0.37173053 -1.63318081 -0.37121312

3
 1.19253421 0.00000000
 109.40036494 0.26980649
 39.19473624 0.00000000

1
 PK MARKER 6.56241261 67.05928494

1
 0
 4
 -1.63318081 -0.37121312
 -0.52198555 -0.11848040
 -1.12959087 0.42642331
 7.20013005 72.53212216
 1
 -0.63288945 0.36908480

4
 3
 1.63294545 0.37173053 -1.63318081 -0.37121312

5
 3.19253421 0.00000000
 185.81523646 0.26980649
 0.05396130 0.00000000
 74.17454538 -0.26980649
 36.36021423 0.00000000

1
 PK MARKER 4.564 - 4.51 67.05928494

S2
26
0

-1.63318081	-0.37121312
-4.13602149	-0.94055995
-4.30786006	0.04325332
-4.20360545	0.05544526
-3.77543972	0.47500742
6.56261828	67.05925228

2

-4.24692539	-0.45301475
-4.26954012	0.55171333

S3
26

1.63294545	0.37173058	-1.63318081	-0.37121312
------------	------------	-------------	-------------

5

13.19253349	0.00000000
219.18476354	-0.26980649
0.05396130	0.00000000
45.00000000	20.58314839
12.87634945	0.00000000

1

EVENNOAKS MARKER	27.58370834	11.53976755
------------------	-------------	-------------

S3
26
6

-1.63318081	-0.37121312
-22.20620117	-5.05105774
-21.70129139	-5.84843367
-21.63972574	-5.76753260
3.49937841	8.68159731
27.58370729	11.58976723

2

-22.09529727	-5.83830294
8.57171045	-29.37139370

S4
26
0

-1.63318081	-0.37121312
-4.13602149	-0.94055995
-4.30735006	0.04325332
-4.20360545	0.05544526
-4.09383784	0.00176011
27.58399231	11.58898543

2

-4.24692539	-0.45301475
-4.26954012	0.55171333

S4
26

1.63294545	0.37173058	-1.63318081	-0.37121312
------------	------------	-------------	-------------

5

3.19253421	0.00000000
185.81523646	0.26980649
0.05396130	0.00000000
12.96103617	-0.26980649
18.18913228	0.00000000

1

EVENNOAKS MARKER	27.58370834	11.53976755
------------------	-------------	-------------

S5
26

1.63294545	0.37173058	-1.63318081	-0.37121312
------------	------------	-------------	-------------

9

1.19253421	0.00000000
172.07423456	0.26980649
16.38354301	0.00000000
55.37176395	-0.26980649
15.11198521	0.00000000
301.34092704	0.26980649
21.59287643	0.00000000
156.91162131	-0.26980649
31.89924622	0.00000000

3

EVENNOAKS MARKER	27.58370834	11.53976755
ORCHURCH MARKER	34.61373419	39.12334087
PK	6.56241251	67.05928494

Manuel de l'utilisateur

Pour pouvoir utiliser correctement le programme SIDCAN, l'utilisateur doit être particulièrement attentif aux points suivants :

- I) Connaître la syntaxe exacte qui lui est permise dans l'écriture du texte-SID ainsi que la sémantique y afférente.

Ces renseignements figurent dans le mémoire p. 7 à 14, 22 et 24.

- 2) L'utilisation de SIDCAN est très facile puisque le programme n'est pratiquement pas interactif.
 - a) Créer (ou modifier), au moyen d'un éditeur de texte un fichier SID (à sauver dans le fichier XXXSID.DAT, XXX étant les trois premières lettres de l'aéroport concerné.)
 - b) Créer, si nécessaire, un fichier AIRPORT, contenant les paramètres de l'aéroport concerné.
 - c) Créer, si nécessaire, un fichier NAVAID contenant les paramètres des aides à la navigation de l'aéroport concerné.

L'utilisateur est donc vivement prié de consulter (et de respecter rigoureusement !) les spécifications des différents fichiers "utilisateur". Ces spécifications se trouvent en Annexes p.

d) lancer l'exécution du programme SIDCAN

Après affichage du menu, l'utilisateur se voit proposé trois choix.

- traduction d'un texte SID déterminé
- affichage d'un texte SID déterminé (sortie graphique)
- sortie du programme

Si l'utilisateur désire afficher un texte-SID, il sera très souvent contraint à demander d'abord la traduction, le programme SIDCAN, dans sa version actuelle ne permettant la traduction et le stockage que d'un seul texte SID à la fois.

3) Ecriture d'un texte SID

Cette opération étant la plus courante et probablement la seule à effectuer par l'utilisateur, il convient de la décrire plus profondément :

- le texte-SID doit être écrit en MAJUSCULES.
- Aucun retour-chariot ne peut figurer dans le texte-SID aussi long soit-il.
- Tous les mots du texte-Sid doivent être séparés par au moins un espace.

Ces consignes (et d'autres) figurent dans le mémoire p. 23

PROGRAM SIDCAN

CHARACTER*1 REPOSE
LOGICAL SORTIE

```

SORTIE = .FALSE.
DOWHILE (.NOT. SORTIE)
0      WRITE(5,10)
      FORMAT(///' SID-CANAR TRANSLATION'//IX,25('*')///)
0      WRITE(5,20)
      FORMAT(/' MENU'//5X,
1      'SID-CANAR TRANSLATION (T)'//5X
1      'SID-TRACK PLOTTING (P)'//5X
1      'STOP (S)')
5      READ(5,25)REPOSE
      FORMAT(A1)
      IF (REPOSE.EQ.'T') CALL TRADUCTION
      IF (REPOSE.EQ.'P') CALL PLOTTING
      IF (REPOSE.EQ.'S') SORTIE = .TRUE.
ENDDO
STOP
END

```


SUBROUTINE TRADUCTION

```

*****
***      DECLARATIONS      ***
*****

***      1. DEC EXTERNES *****

CHARACTER*20 PHRASE
INTEGER      NUMSUB

***      2. DEC COMMON *****

INCLUDE      "SCAN.CMN"

*****

CALL INITIALISATIONS
CALL LIREMOT
DO WHILE (TYPMDT.NE.27)
    CALL LIREPHRASE(PHRASE)
    CALL ANALYSEPHRASE(PHRASE,NUMSUB)
    CALL TRADUCPHRASE(NUMSUB)
ENDDO
CALL INSERERORR
CALL CLOTURE
RETURN
END

```

SUBROUTINE INITIALISATIONS

```
*****
***      DECLARATIONS      ***
*****
```

```
***** 1. DEC EXTERNES *****
```

```
CHARACTER*10  CTRLAT,CTRLONG
REAL*8        PHI,THETA
```

```
***** 2. DEC COMMON *****
```

```
INCLUDE      'GEN.CMN'
INCLUDE      'COORD.CMN'
INCLUDE      'INITANAL.CMN'
INCLUDE      'INITSCAN.CMN'
INCLUDE      'TRACK.CMN'
```

```
***** 3. DEC INTERNES *****
```

```
CHARACTER*250  SIDF
CHARACTER*10  LON, (0:36), LAT(0:36), RWYLONG, RWYLAT, RECLONG, RECLAT
CHARACTER*10  FAIR, FNAV, FSID, SIDLF
REAL*8        TOT, TIT, PI, N
INTEGER        RECIP, NUMRNY, RWYN(0:36), I, J, JK1, JK2, RWYF, K
LOGICAL        TROUVE
```

```
*****
```

```
WRITE(5,*) 'NOM DE L' "AEROPORT ?'
READ(5,10) AIRPORT
FORMAT(A15)
WRITE(5,*) 'NOM DU SID A TRADUIRE ?'
READ(5,20) SIDLAB
FORMAT(A10)
WRITE(5,*) 'NUMERO DE LA PISTE DE DECOLLAGE ?'
READ(5,40) RWY
FORMAT(I2)
IF ((RWY.LT.0).OR.(RWY.GT.36)) GOTO 30
```

```
*****
```

PREMIERES INITIALISATIONS

```
*****
```

```
*** NOM DES FICHIERS ***
```

```
FAIR = AIRPORT(1:3)//'AIR.DAT'
FNAV = AIRPORT(1:3)//'NAV.DAT'
FSID = AIRPORT(1:3)//'SID.DAT'
```

```
*** NUMERO DE PISTE ***
```

```
RECIP = RWY+18
IF (RECIP.GT.36) RECIP = RECIP-36
```



```

*** OUVERTURE DES FICHIERS ***
*** STOCKAGE DES DONNEES ***
*** FERMETURE DES FICHIERS ***

```

```

OPEN(UNIT=11,FILE=FAIR,STATUS='OLD',ERR=170)
READ(11,35)DVM
FORMAT(F4.1)
READ(11,50)CTRLAT,CTRLONG
FORMAT(A10,1X,A10)
READ(11,50)NUMRNY
FORMAT(I2)
DO I=1,NUMRNY
  READ(11,70,END=170)RWYN(I),LAT(I),LONG(I)
  FORMAT(I2,1X,2(A10,1X))
ENDDO
OK1=0
OK2=0
DO I=1,NUMRNY
  IF (RWY.EQ.RWYN(I)) THEN
    RWYLONG=LONG(I)
    RWYLAT=LAT(I)
    OK1=1
  ENDIF
  IF (RECIP.EQ.RWYN(I)) THEN
    RECLONG=LONG(I)
    RECLAT=LAT(I)
    OK2=1
  ENDIF
ENDDO
IF (OK1*OK2.EQ.0) CALL ERREUR(1)
CLOSE(UNIT=11)

```

```

*****
***** FICHIER SID *****
*****

```

```

OPEN(UNIT=12,FILE=FSID,STATUS='OLD',ERR=170)
TROUVE = .FALSE.
DOWHILE (.NOT. TROUVE)
  READ(12,90,END=95)RWYF,SIDLF,SIDF
  FORMAT(I2,1X,A10/A250)
  IF ((SIDLAB.EQ.SIDLF).AND.(RWY.EQ.RWYF)) TROUVE=.TRUE.
ENDDO
IF (.NOT. TROUVE) CALL ERREUR(2)
SID=SIDF//
PS=1
CLOSE(UNIT=12)

```

```

*****
*****
***** FICHIER NAVAID *****
*****

```

```

OPEN(UNIT=13,FILE=FNAV,STATUS='OLD',ERR=170)
READ(13,110,END=120)((NAVDAT(I,J),J=1,4),I=1,31)
FORMAT(4(A10,2X))
CLOSE(UNIT=13)

```

```

*****
***      FICHIER MASK      ***
*****

OPEN(UNIT=14,FILE='MASK.DAT',STATUS='OLD',ERR=170)
READ(14,130,END=140) (MASK(I),TRAJ(I),I=1,99)
FORMAT(A20,I2)
CLOSE(UNIT=14)

```

```

*****
***      FICHIER DICT      ***
*****

OPEN(UNIT=15,FILE='DICT.DAT',STATUS='OLD',ERR=170)
READ(15,150,END=160) (DICT(I),I=1,13)
FORMAT(A10)
CLOSE(UNIT=15)

```

```

*****
***      INITIALISATIONS      ***
*****

PI=3.14159265
CALL BRDGS(CTRLONG,PHI)
CALL BRDGS(CTRLAT,THETA)
IF ((THETA.EQ.90).OR.(THETA.EQ.-90)) CALL ERREUR(3)
X=6392.015*COSD(PHI)*COSD(THETA)
Y=6392.015*SIND(PHI)*COSD(THETA)
Z=6374.567*SIND(THETA)
IF (THETA.EQ.0) THEN
    XN(1)=0
    XN(2)=0
    XN(3)=0
    XE(1)=COSD(PHI+90)
    XE(2)=SIND(PHI+90)
    XE(3)=0
ELSE
    N=(X*X+Y*Y+Z*Z)/Z
    XN(1)=-X
    XN(2)=-Y
    XN(3)=N-Z
    XE(1)=-Y
    XE(2)=X
    XE(3)=0
ENDIF
TOT=0
TIT=0
DO K=1,3
    TOT=TOT+XE(K)*XE(K)
    TIT=TIT+XN(K)*XN(K)
ENDDO
DO K=1,3
    XE(K)=XE(K)/SQRT(TOT)
    XN(K)=XN(K)/SQRT(TIT)
ENDDO
CALL COORD(RECLAT,RECLONG,RECCAR)

```



```
CALL COORD(RWYLAT,RWYLONG,RWYCAR)
LASTPT(0)=RWYCAR(0)
LASTPT(1)=RWYCAR(1)
RWYLENGHT=DIST(RECCAR,RWYCAR)
CALL ANGLE(RWYCAR,RECCAR,LASTANG)
TRACK(1,1)=0
TRACK(1,2)=0
ITRACK=1
POINTS(1,0)=RECCAR(0)
POINTS(1,1)=RECCAR(1)
IPPOINT=2
ICTR=0
NBNAV=0
MAXTRACK=15
RMIN=0.5
RETURN
CALL ERREUR(4)
END
```

SUBROUTINE LIREPHRASE(PHASE)

```

*****
***      LIRE UNE PHRASE      ***
*****

```

```

*****
***      DECLARATIONS      ***
*****

```

```

*****  1. DEC EXTERNES *****

```

```

CHARACTER*20  PHRASE
INTEGER      NBRE, ID

```

```

***      2. DEC COMMON      ***

```

```

INCLUDE      'TRADUC.CMN'
INCLUDE      'SCAN.CMN'

```

```

***      3. DEC INTERNES      ***

```

```

INTEGER      NBREMOT

```

```

*****

```

```

PHRASE = '
NBREMOT = 0
NBRE = 0
ID = 0
DO WHILE ((TYPMOT.NE.27).AND.(TYPMOT.NE.22).AND.(TYPMOT.NE.9))
  IF (TYPMOT.EQ.20) THEN
    NBRE = NBRE + 1
    TNDMBRE(NBRE) = NBRELU
  ENDIF
  IF (TYPMOT.EQ.4) THEN
    ID = ID + 1
    TIDENTIF(ID) = 'LEFT'
  ENDIF
  IF (TYPMOT.EQ.8) THEN
    ID = ID + 1
    TIDENTIF(ID) = 'RIGHT'
  ENDIF
  IF (TYPMOT.EQ.21) THEN
    ID = ID + 1
    TIDENTIF(ID) = MOTLU
  ENDIF
  NBREMOT = NBREMOT + 1
  PHRASE(NBREMOT:NBREMOT) = CHAR(TYPMOT+64)
  CALL LIREMOT
ENDDO
IF (TYPMOT.NE.27) CALL LIREMOT
RETURN
END

```



```

*****
***      LIRE MOT      ***
*****

```

```

*****
***      DECLARATIONS  ***
*****

```

```

*****  1. DEC COMMON *****

```

```

INCLUDE      "INITSCAN.CMN"
INCLUDE      "SCAN.CMN"

```

```

***      2. DEC INTERNES      ***

```

```

REAL*8      MULOIX
INTEGER     P1,Q,BINF,BSUP

```

```

*****

```

```

***      ON SAUTE LES BLANCS      ***

```

```

IF (PS.EQ.251) THEN
  TYPMOT = 27
  RETURN
ENDIF
IF (SID(PS:PS).EQ.' ') THEN
  PS = PS+1
  GOTO 10
ENDIF

```

```

***      EST-CE LE CARACTERE SPECIAL "." ?      ***

```

```

IF (SID(PS:PS).EQ.'.') THEN
  TYPMOT = 22
  PS = PS+1
  RETURN
ENDIF

```

```

***      EST-CE UN IDENTIF OU UN MOT RESERVE ?      ***

```

```

IF ((SID(PS:PS).GE.'A').AND.(SID(PS:PS).LE.'Z')) THEN
  P1=PS
  DOWHILE (SID(PS:PS).NE.' ')
    IF (PS-P1.GT.9) CALL ERREUR(5)
    MOTLU = SID(P1:PS)
    PS = PS+1
  ENDDO

```

```

***      RECHERCHE DU MOT DANS LA TABLE      ***

```

```

BINF=1
BSUP=13
DOWHILE (BINF.LE.BSUP)
  Q=INT(REAL(BINF+BSUP)/2)
  IF (COICT(Q).EQ.MOTLU) THEN
    TYPMOT=Q

```

```

      RETURN
    ENDIF
    IF (DICT(Q).GT.MOTLU) THEN
      BSUP=Q-1
    ELSE
      BINF=Q+1
    ENDIF
  ENDDO

```

```

***      ON EST SORTI DE LA BOUCLE SANS PASSER PAR      ***
***      RETURN, LE MOT N'EST DONC PAS DANS LA        ***
***      TABLE, C'EST DONC UN IDENTIFI=              ***

```

```

      TYPMOT = 21
      RETURN
    ENDIF

```

```

***      EST-CE UN NOMBRE ?                            ***

```

```

IF ((SID(PS:PS).LT."0").OR.(SID(PS:PS).GT."9")) CALL ERREUR(6)
TYPMOT=20
NBRELU=0.0
DOWHILE ((SID(PS:PS).GE."0").AND.(SID(PS:PS).LE."9"))
  NBRELU=NBRELU*10+ICHAR(SID(PS:PS))-48
  PS=PS+1

```

```

ENDDO
IF (SID(PS:PS).NE."") RETURN
PS=PS+1
IF ((SID(PS:PS).LT."0").OR.(SID(PS:PS).GT."9")) CALL ERREUR(7)
MULDIX=1.0
DOWHILE ((SID(PS:PS).GE."0").AND.(SID(PS:PS).LE."9"))
  MULDIX=MULDIX*10.0
  NBRELU=NBRELU+(ICHAR(SID(PS:PS))-48)/MULDIX
  PS=PS+1

```

```

ENDDO
RETURN
END

```


SUBROUTINE ANALYSEPHRASE(PHASE,NUMSUB)

```

*****
***      ANALYSE PHRASE      ***
*****

*****
***      DECLARATIONS      ***
*****

***** 1. DEC EXTERNES *****
CHARACTER*20 PHASE
INTEGER      NUMSUB

***      2. DEC COMMON      ***
INCLUDE      'INITANAL.CMN'

***      3. DEC INTERNES    ***
INTEGER      I,NBREPHRASE

*****
NBREPHRASE=35
MASK(NBREPHRASE)=PHASE
I=1
DO WHILE (PHASE.NE.MASK(I))
    I=I+1
ENDDO
IF (I.EQ.NBREPHRASE) CALL ERREUR(8)
NUMSUB=TRAJ(I)
RETURN
END

```

SUBROUTINE PLOTTING

- 22 -

```

INCLUDE      'TRACK.CMN'
INCLUDE      'GEN.CMN'
INCLUDE      'LYS.CMN'
INCLUDE      'PLOT.CMN'

CHARACTER*10  SIDF,FCAN,FPLO
CHARACTER*2   CRWY
CHARACTER*1   RP
1  REAL        TOTABS,TOTCRD,NEWABS,NEWCRD,
1  INTEGER     DIFF,RAPFEN,R
1  LOGICAL      RWYF,I,J,
               CTAGA,CTADD,CTDBA,CTDHA
               X,SENS

```

```

WRITE(5,*) 'SORTIE GRAPHIQUE D'UNE TRAJECTOIRE'
WRITE(5,*) 'NOM DE L'AEROPORT'
READ(5,1) AIRPORT
FORMAT(A15)
WRITE(5,*) 'NOM DU SID'
READ(5,2) SIDLAB
FORMAT(A10)
WRITE(5,*) 'NUMERO DE LA PISTE DE DECOLLAGE'
READ(5,3) RWY
FORMAT(I2)

```

```

FCAN=AIRPORT(1:3)//'CAN.DAT'
FPLO=AIRPORT(1:3)//'PLO.DAT'

```

```

OPEN (UNIT=11,FILE=FCAN,STATUS='OLD',ERR=100)
READ(11,5) SIDF
FORMAT(A10)
IF (SIDF.NE.SIDLAB) CALL ERREUR(9)
READ(11,10) RWYF
FORMAT(I2)
IF (RWYF.NE.RWY) CALL ERREUR(10)
READ(11,15) RWYCAR(0),RWYCAR(1),RECCAR(0),RECCAR(1)
FORMAT(4(1X,F15.3))
READ(11,20) ITRACK
FORMAT(I2)
READ(11,25) (TRACK(I,1),TRACK(I,2),I=1,ITRACK)
FORMAT(F15.8,1X,F15.8)
READ(11,30) NBNVAV
FORMAT(I2)
READ(11,35) (NAVUSED(I,1),NAVUSED(I,2),NAVCDORD(I,0),
1  NAVCDORD(I,1),I=1,NBNVAV)
FORMAT(2(A10,1X),2F15.3)
CLOSE(UNIT=11)

```

```

OPEN(UNIT=12,FILE=FPLO,STATUS='OLD',ERR=100)
READ(12,40) SIDF
FORMAT(A10)
IF (SIDF.NE.SIDLAB) CALL ERREUR(11)
READ(12,42) RWYF
FORMAT(I2)
IF (RWYF.NE.RWY) CALL ERREUR(12)

```



```

READ(12,45) IPOINT
FORMAT(12)
READ(12,50) (POINTS(I,0), POINTS(I,1), I=1, IPOINT)
FORMAT(2F15.8)
READ(12,55) ICTR
FORMAT(12)
READ(12,60) (CENT(I,0), CENT(I,1), I=1, ICTR)
FORMAT(2F15.8)
CLOSE(UNIT=12)

```

***** INITIALISATIONS *****

```

VAMIN = 0
VAMAX = 799
VOMIN = 0
VOMAX = 479
CTAGA = 20
CTADR = 789
CTOBA = 50
CTOHA = 439
RAPFEN = (CTADR-CTAGA)/(CTOHA-CTOBA)

```

***** CALCUL DES VALEURS MIN ET MAX DES ABSC ET ORDON ***

```

ABSMIN=10E10
ABSMAX=-10E10
ORDMIN=10E10
ORDMAX=-10E10
DO I=1, IPOINT
    CALL MAX(POINTS(I,0), POINTS(I,1))
ENDDO
CALL MAX(RWYCAR(0), RWYCAR(1))
CALL MAX(RECCAR(0), RECCAR(1))
DO I=1, NBNAV
    CALL MAX(NAVCOORD(I,0), NAVCOORD(I,1))
ENDDO
DO I=1, ICTR
    CALL MAX(CENT(I,0), CENT(I,1))
ENDDO

```

```

TOTABS=ABSMAX-ABSMIN
TOTORD=ORDMAX-ORDMIN
IF(TOTABS/TOTORD.LT.RAPFEN) THEN
    NEWABS=TOTORD*RAPFEN
    DIFF = NEWABS-TOTABS
    ABSMIN=ABSMIN-DIFF/2
    ABSMAX=ABSMAX+DIFF/2
ELSE
    NEWORD=TOTABS/RAPFEN
    DIFF = NEWORD-TOTORD
    ORDMIN=ORDMIN-DIFF/2
    ORDMAX=ORDMAX+DIFF/2
ENDIF

```

```

RAPP = (CTADR-CTAGA)/(ABSMAX-ABSMIN)

```

```

ABSINF=(CTAGA-VAMIN)/RAPP
ABSSUP=(VAMAX-CTADR)/RAPP
ORDINF=(CTDBA-VOMIN)/RAPP
ORDSUP=(VOMAX-CTJHA)/RAPP

```

```

ABSMIN=ABSMIN-ABSINF
ABSMAX=ABSMAX+ABSSUP
ORDMIN=ORDMIN-ORDINF
ORDMAX=ORDMAX+ORDSUP

```

```

CALL INGRAF
CALL CLS
CALL VIEWPORT(0,799,0,479)
CALL WINDOW(ABSMIN,ABSMAX,ORDMIN,ORDMAX)

```

```

CALL CLIPPER(ABSMIN,ORDMAX,ABSMAX,ORDMAX)
CALL CLIPPER(ABSMIN,ORDMIN,ABSMAX,ORDMIN)
CALL CLIPPER(ABSMIN,ORDMIN,ABSMIN,ORDMAX)
CALL CLIPPER(ABSMAX,ORDMIN,ABSMAX,ORDMAX)
CALL CLIPPER(WAB( 0),WOR( 30),WAB(799),WOR( 30))
CALL CLIPPER(WAB( 0),WOR(449),WAB(799),WOR(449))
CALL POSIT(WAB(10),WOR(469))
CALL TEXT ("AIRPORT")
CALL POSIT(WAB(100),WOR(469))
CALL TEXT ("AIRPORT")
CALL POSIT(WAB(300),WOR(469))
CALL TEXT ("SID")
CALL POSIT(WAB(400),WOR(469))
CALL TEXT ("SIDLAB")
CALL POSIT(WAB(600),WOR(469))
CALL TEXT ("RUNWAY")
CALL POSIT(WAB(700),WOR(469))
I=0
DO WHILE (RWY.GE.10)
    RWY=RWY-10
    I=I+1
ENDDO
CRWY(1:1)=CHAR(I+48)
CRWY(2:2)=CHAR(RWY+48)
CALL TEXT(CRWY)
CALL POSIT(WAB(300),WOR(25))
CALL TEXT ("<R>RETURN TO MENU")

```

```

CALL INK(3)
CALL CLIPPER(RWYCAR(0),RWYCAR(1),RECCAR(0),RECCAR(1))
CALL POSIT (POINTS(1,0),POINTS(1,1))

CALL INK(2)
CALL VECT(POINTS(2,0),POINTS(2,1))
I=2
DO WHILE (I.LE.IPOINT-2)
    IF (TRACK(1,2).LT.0) THEN
        J=I

```


ELSE

J=I+1

ENDIF
CALL CIRCLE(CENT(I/2,0),CENT(I/2,1),POINTS(J,0),
POINTS(J,1),NINT(TRACK(I,1)),.TRUE.)
CALL POSIT(POINTS(I+1,0),POINTS(I+1,1))
CALL VECT(POINTS(I+2,0),POINTS(I+2,1))
I=I+2

ENDDO

CALL INK(3)

DO I=1,NBNAV

CALL POSIT(NAVCOORD(I,0),NAVCOORD(I,1))

CALL TEXT(NAVUSED(I,1)(1:1))

ENDDO

READ(5,95)RP

FORMAT(A1)

CALL EXGRAF

RETURN

00 CALL ERREUR(13)

END

```
*****  
SUBROUTINE  MAX(A,B)
```

```
INCLUDE "PLOT.CMN"
```

```
REAL      A,B
```

```
IF (A.LT.ABSMIN) ABSMIN=A
```

```
IF (A.GT.ABSMAX) ABSMAX=A
```

```
IF (B.LT.ORDMIN) ORDMIN=B
```

```
IF (B.GT.ORDMAX) ORDMAX=B
```

```
RETURN
```

```
END
```

REAL FUNCTION WAB(X)

INCLUDE "PLOT.CMN"

INTEGER X

WAB = ABSMIN+X/RAPP

RETURN

END

REAL FUNCTION WOR(Y)

INCLUDE "PLOT.CMN"

INTEGER Y

WOR = ORDMIN+Y/RAPP

RETURN

END

SUBROUTINE ERREUR(ERR)

INTEGER ERR
CHARACTER*50 MESS

WRITE(5,10)ERR
FORMAT(///'ERREUR !!!!!!!!!!!!!!!'/'ERREUR NO ',I2/)

1 FORMAT(' ',A50//'PROGRAMME INTERROMPU, VEUILLEZ CORRIGER
L'ERREUR ET RECOMMENCER L'EXECUTION')
STOP
END

```

*****
***      CONVERSION COORDONNEES GEOGRAPHIQUES      ***
***      -->      COORDONNEES CARTESIENNES      ***
*****

```

```

*****
***      DECLARATIONS      ***
*****

```

```

IMPLICIT REAL*8 (A-Z)
CHARACTER*10    LAT, LONG
DIMENSION      A(0:1)

```

```

INCLUDE 'COORD.CMN'

```

```

*****

```

```

CALL BRDGS(LONG, LONGD)
CALL BRDGS(LAT, LATD)
XG=6392.015*CCSD(LONGD)*COSD(LATD)
YG=6392.015*SIND(LONGD)*COSD(LATD)
ZG=6374.567*SIND(LATD)
BETA=(X*X+Y*Y+Z*Z)/(X*XG+Y*YG+Z*ZG)
XGP=XG*BETA
YGP=YG*BETA
ZGP=ZG*BETA
XVP=XGP-X
YVP=YGP-Y
ZVP=ZGP-Z
A(0)=(XVP*XN(2)-YVP*XN(1))/(XN(2)*XE(1)-XN(1)*XE(2))
A(1)=-(XVP*XE(2)-YVP*XE(1))/(XN(2)*XE(1)-XN(1)*XE(2))
RETURN
END

```


SUBROUTINE BRDGS(A,ADEC)

```

*****
***      CONVERSION  DEG:MIN:SEC -> DEG.DECIMAL      ***
*****

```

```

*****
***      DECLARATIONS      ***
*****

```

```

*****  1. DEC EXTERNES *****

```

```

CHARACTER*10      4
REAL*8           ADEC

```

```

*****  2. DEC INTERNES *****

```

```

CHARACTER*1      ORIENT
INTEGER          SIGN,DEG,MIN,SEC

```

```

*****

```

```

ORIENT=A(1:1)
SIGN=0
IF(ORIENT.EQ.'N') SIGN=1
IF(ORIENT.EQ.'S') SIGN=-1
IF(ORIENT.EQ.'E') SIGN=1
IF(ORIENT.EQ.'W') SIGN=-1
IF(SIGN.EQ.0) CALL ERREUR(14)
DEG=IVAL(A(2:4))
IF((ORIENT.EQ.'N').OR.(ORIENT.EQ.'S')) THEN
  IF (DEG.GT.90) CALL ERREUR(14)
ENDIF
IF((ORIENT.EQ.'E').OR.(ORIENT.EQ.'W')) THEN
  IF (DEG.GT.180) CALL ERREUR(14)
ENDIF
IF (A(5:5).NE.':') CALL ERREUR(14)
MIN=IVAL(A(6:7))
IF((MIN.LT.0).OR.(MIN.GT.59)) CALL ERREUR(14)
IF(A(8:8).NE.':') CALL ERREUR(14)
SEC=IVAL(A(9:10))
IF((SEC.LT.0).OR.(SEC.GT.59)) CALL ERREUR(14)
ADEC=(DEG+DBLE(MIN)/60+DBLE(SEC)/3600)*SIGN
RETURN
END

```

```
FUNCTION          IVAL(A)
CHARACTER*(*)     A
INTEGER           IVAL,I

IVAL=0
DO I=1,LEN(A)
  IF((A(I:I).LT.'0').OR.(A(I:I).GT.'9'))CALL ERREUR(14)
  IVAL=IVAL*10+(ICHAR(A(I:I))-48)
ENDDO
RETURN
END
```


SUBROUTINE TRADUCPHRASE(NUMSUB)

```

INCLUDE      'TRACK.CMN'
INCLUDE      'GEN.CMN'
INCLUDE      'TRADUC.CMN'

```

```

REAL*8      KM,CP,NAV(0:1)
INTEGER     NUMSUB,SENS
LOGICAL     CONTPD

```

```

GOTO (1,2,3,4,5,6,7,8,9) , NUMSUB

```

```

*** STRAIGHT ***

```

```

CALL STRAIGHT(CONVKM(TNOMBRE(1)))
RETURN

```

```

*****

```

```

*** RADIALE ***

```

```

I=1
CONTPD = .FALSE.
GOTO 10
I=2
CONTPD = .TRUE.
CALL CONVDIR(TIDENTIF(I),SENS)
CALL CONVCAP(TNOMBRE(1),CP)
CALL CONVNAV(TIDENTIF(I+1),NAV)
CALL RADIALE(CP,NAV,SENS,CONTPD)
RETURN

```

```

*****

```

```

*** SURVOL ***

```

```

I=1
CONTPD = .FALSE.
GOTO 15
I=2
CONTPD = .TRUE.
CALL CONVDIR(TIDENTIF(I),SENS)
CALL CONVNAV(TIDENTIF(I+1),NAV)
CALL SURVOL(NAV,SENS,CONTPD)
RETURN

```

```

*****

```

```

*** STURN ***

```

```

I=1
GOTO 20
I=2
CALL CONVDIR(TIDENTIF(I),SENS)
CALL CONVCAP(TNOMBRE(1),CP)
CALL CONVNAV(TIDENTIF(I+1),NAV)
CALL STURN(CP,NAV,SENS)
RETURN

```

```

*****

```

*** CAP ***

```

I=1
GOTO 25
I=2
CALL CONVDIR(TIDENTIF(I),SENS)
CALL CONVCAP(TNOMBRE(I),CP)
CALL CAP(CP,SENS)
RETURN
END

```

SUBROUTINE CONVNAV(NAV,CC)

```

INCLUDE      "GEN.CMN"
INCLUDE      "TRACK.CMN"

```

```

CHARACTER*10 NAV
REAL*8      CC(0:1)
INTEGER I

```

```

NAVUSED(NBNAV+1,1)=NAV
I=1
DO WHILE (NAVUSED(I,1).NE.NAV)
  I=I+1
ENDDO
IF (I.EQ.NBNAV+1) THEN
  NAVDAT(31,1)=NAV
  I=1
  DO WHILE (NAVDAT(I,1).NE.NAV)
    I=I+1
  ENDDO
  IF (I.EQ.31) CALL ERREUR(15)
  NBNAV=NBNAV+1
  NAVUSED(NBNAV,1)=NAV
  NAVUSED(NBNAV,2)=NAVDAT(I,2)
  CALL COORD(NAVDAT(I,3),NAVDAT(I,4),CC)
  NAVCOORD(NBNAV,0)=CC(0)
  NAVCOORD(NBNAV,1)=CC(1)
  ELSE
    CC(0)=NAVCOORD(I,0)
    CC(1)=NAVCOORD(I,1)
  ENDIF
RETURN
END

```

SUBROUTINE CONVDIR(DIRECT,SENS)

```

CHARACTER*10 DIRECT
INTEGER      SENS

```

```

IF (DIRECT.EQ.'LEFT') SENS=-1
IF (DIRECT.EQ.'RIGHT') SENS=1
RETURN

```

END

SUBROUTINE CONVCAP(MAGN,TRIGO)

INCLUDE "GEN.CMN"

REAL*8 MAGN,TRIGO

TRIGO=450-MAGN-DVM

IF (TRIGO.GE.360) TRIGO=TRIGO-360

RETURN

END

SUBROUTINE STRAIGHT(DIST)

```

*****
* ARGUMENT
*
* DIST          Longueur du segment droit          REAL*8
*
* RESULTAT
*
* (Effet de bord) mise-a-jour des  -segments TRACK
*                               -var. globales
*
* SPECIFICATIONS
*
* La procedure effectue la creation et le stockage d'un segment
* droit correspondant a la trajectoire suivie par un avion cou-
* vrant la distance DIST en conservant son cap initial.
* Elle met a jour les variables globales (position de l'avion)
*
*****
*
*****
* DECLARATIONS
*
* NOM          TYPE DE VARIABLE          TYPE FORTRAN
*
* DIST          PARAMETRE(ARG)          REAL*8
* LASTPT        COMMON                  LASTPT(0:1) REAL*8
* LASTANG        COMMON                  REAL*8
* ENDPT         VAR LOCALE              ENDPT(0:1) REAL*8
*****
INCLUDE 'TRACK.CMN'

REAL*8 DIST,ENDPT
DIMENSION ENDPT(0:1)

IF (DIST.EQ.0) RETURN
IF (ITRACK.EQ.1) DIST=DIST - RWYLENGHT
ENDPT(0) = LASTPT(0)+DIST*COSD(LASTANG)
ENDPT(1) = LASTPT(1)+DIST*SIND(LASTANG)
CALL SAVESTRAIGHT(DIST,ENDPT)
RETURN
END

```

```

*****
* ARGUMENTS
*
* BETA      Angle de la droite D      Real*8
* NAVAIID   Pt de la droite D        Array[0:1] of Real*8
* SENS      Sens du virage            INTEGER
* CONTPD    Contrainte sur le pt de dep. BOOLEAN
*
* RESULTAT
*
* Mise-a-jour du tableau des segments et des variables globales
*
* SPECIFICATIONS
*
* La procedure effectue la creation et le stockage de 1 a plusieurs
* segments droits ou courbes correspondant a la trajectoire suivie
* par un avion effectuant un virage dans le sens indique par SENS
* pour s'aligner sur le cap BETA et poursuivant sa route jusqu'au
* point NAVAIID.
* L'avion commence son virage :
* -en LASTPT (Si la valeur de CONTPD est TRUE)
* -en un point situe sur la droite D(LASTANG,LASPT)
*   (Si la valeur de CONTPD est FALSE)
*****
* DECLARATIONS
*
* NOM                TYPE DE VARIABLE                TYPE FORTRAN
*
* BETA               PARAMETRE(ARG)                   REAL*8
* NAVAIID            PARAMETRE(ARG)                   REAL*8
* SENS               PARAMETRE(ARG)                   INTEGER
* CONTPD            PARAMETRE(ARG)                   BOOLEAN
* LASTPT            COMMON                             REAL*8
* LASTANG           COMMON                             REAL*8
* R                 VAR LOCALE                         REAL*8
* TROPLOIN          COMMON                             REAL*8
* RMIN              COMMON                             REAL*8
* PTDEP             VAR LOCALE                         REAL*8
* POSSIBLE          VAR LOCALE                         BOOLEAN
* C                 VAR LOCALE                         REAL*8
* PTFIN            VAR LOCALE                         REAL*8
*****
INCLUDE "TRACK.CHN"

REAL*8 BETA,NAVAID,R,PTDEP,C,PTFIN,ICO:1)
INTEGER SENS
LOGICAL CONTPD,TROPLOIN,POSSIBLE,PREC
DIMENSION NAVAIID(0:1),PTDEP(0:1),PTFIN(0:1)
CALL INSEREROR
CALL DIFFANG(LASTANG,BETA,DIFF,SENS)
CALL INTERSECT(LASTANG,PTFIN,BETA,NAVAID,I)
IF (CONTPD) THEN
  IF (PREC(I,PTFIN,PTDEP)) THEN
    IF (DIFF.LT.180) THEN
      CALL STURN(BETA,NAVAID,SENS)

```


1

```

      RETURN
      ELSE
        CALL RAYON(LASTPT, LASTANG, NAVAID, BETA, DIFF, SENS, R)
        IF (R.LT.RMIN) CALL ERREUR(16)
      ENDIF
      ELSE
        IF (DIFF.GT.180) CALL ERREUR(17)
        CALL RAYON(LASTPT, LASTANG, NAVAID, BETA, DIFF, SENS, R)
        IF (R.LT.RMIN) CALL ERREUR(18)
      ENDIF
      ELSE
        R=RMIN
        IF (.NOT.(PREC(LASTPT, I, LASTANG))) THEN
          CONTPD = .TRUE.
          GOTO 10
        ENDIF
        IF (DIFF.GT.180) CALL ERREUR(19)
        CALL PTDEPART(LASTPT, LASTANG, NAVAID, BETA,
                     DIFF, I, SENS, PTDEP)
        CALL MODIF(PTDEP, POSSIBLE)
        IF (.NOT.(POSSIBLE)) THEN
          CONTPD = .TRUE.
          GOTO 10
        ENDIF
      ENDIF
    ENDIF
    CALL CTRVIR(C, LASTANG, LASTPT, R, SENS)
    CALL FINVIR(C, BETA, PTFIN, R, SENS)
    CALL DIFFANG(LASTANG, BETA, DIFF, SENS)
    CALL SAVECURVE(C, DIFF, SENS, PTFIN, BETA, C)
    IF (PREC(NAVAID, LASTPT, LASTANG)) CALL ERREUR(20)
    CALL STRAIGHT(DIST(LASTPT, NAVAID), NAVAID)
    RETURN
  END

```



```

SUBROUTINE SURVOL(NAVAID,SENS,CONTPD)
INCLUDE "TRACK.CMN"

REAL*8      C(0:1),A,B,CA,CB,CC,PT1(0:1),PT2(0:1),
1          TEMP(0:1),D,NEWPT(0:1),ALPHA,NEWANG,DIFF,ABS,
1          ORD,R1,R2,A1,A2,AX,BX,CX,V,BETA,GAMMA,DIFF1,
1          DIFF2,PT(0:1),NAVAID(0:1)

INTEGER      SENS

LOGICAL      PREC,CONTPD

CALL INSERERDR
CALL CTRVIRCC(LASTANG, LASTPT, RMIN, SENS)
IF (DIST(C,NAVAID).LE.RMIN) THEN
  IF (CONTPD) CALL ERREUR(21)
  IF ((LASTANG.EQ.90).OR.(LASTANG.EQ.270)) THEN
    PT1(0)=NAVAID(0)
    PT2(0)=NAVAID(1)
    R=-(PT1(0)-C(0))*2+RMIN**2
    PT1(1)= C(2)-SQRT(R)
    PT2(1)= C(2)+SQRT(R)
  ELSE
    A = TAND(LASTANG)
    B = NAVAI(1)-NAVAID(0)*A
    CA = A**2+1
    CB = 2*A*B-2*A*C(1)-2*C(0)
    CC = (B-C(1))**2+C(0)**2-RMIN**2
    PT1(0)=(-CB-SQRT(CB**2-4*CA*CC))/(2*CA)
    PT2(0)=(-CB+SQRT(CB**2-4*CA*CC))/(2*CA)
    PT1(1)=A*PT1(0)+B
    PT2(1)=A*PT2(0)+B
  ENDIF
  IF (.NOT.(PREC(PT1,PT2, LASTANG))) THEN
    TEMP(0)=PT1(0)
    TEMP(1)=PT1(1)
    PT1(0)=PT2(0)
    PT1(1)=PT2(1)
    PT2(0)=TEMP(0)
    PT2(1)=TEMP(1)
  ENDIF
  IF((DIST(NAVAID,PT2).GT.(CONVKM(TRACK(ITRACK,1))))).OR.
  1 (TRACK(ITRACK,2).NE.0)) THEN
    D=DIST(NAVAID,PT1)
  ELSE
    D=-DIST(NAVAID,PT2)
  ENDIF
  C(0)=C(0)+D*COSD(LASTANG)
  C(1)=C(1)+D*SIND(LASTANG)
  CALL STRAIGHT(D)
  CALL ANGLE(C,NAVAID,ALPHA)
  NEWANG=ALPHA-90*SENS
  IF (NEWANG.GE.360) NEWANG = NEWANG-360
  IF (NEWANG.LT.0) NEWANG = NEWANG+360
  CALL DIFFANG(LASTANG,NEWANG,DIFF,SENS)
  CALL SAVECURVE(RMIN,DIFF,SENS,NAVAID,NEWANG,C)
  ELSE
  **** LE 2EME CAS D'EXCEPTION EST DETECTE MAIS NON TRAITE
  IF((ABS.EQ.RMIN).OR.(ABS.EQ.-RMIN)) CALL ERREUR(22)

```

```

ABS = C(0)-NAVAID(0)
ORD = NAVAID(1)-C(1)
R1 = RMIN**2-ORD**2
R2 = RMIN**2-ABS**2
A1 = (ABS*ORD+SQRT(ABS**2*ORD**2-R1*R2))/R2
A2 = (ABS*ORD-SQRT(ABS**2*ORD**2-R1*R2))/R2
PT1(0) = (C(0)+A1*(A1*NAVAID(0)-ORD))/(1+A1**2)
PT2(0) = (C(0)+A2*(A2*NAVAID(0)-ORD))/(1+A2**2)
PT1(1) = NAVAID(1)+A1*(PT1(0)-NAVAID(0))
PT2(1) = NAVAID(1)+A2*(PT2(0)-NAVAID(0))
CALL EQUATION(LASTANG, LASTPT, AX, BX, CX)
V = AX*NAVAID(0)+BX*(NAVAID(1))+CX
ALPHA = LASTANG+90*SENS
IF (ALPHA.LT.0) ALPHA=ALPHA+360
IF (ALPHA.GE.360) ALPHA=ALPHA-360
CALL ANGLE(C, PT1, BETA)
CALL ANGLE(C, PT2, GAMMA)
DIFF1=BETA-ALPHA
IF (DIFF1.LT.0) DIFF1=DIFF1+360
DIFF2=GAMMA-ALPHA
IF (DIFF2.LT.0) DIFF2=DIFF2+360
IF (V*(DIFF1-DIFF2).GT.0) THEN
    PT(0)=PT2(0)
    PT(1)=PT2(1)
ELSE
    PT(0)=PT1(0)
    PT(1)=PT1(1)
ENDIF
CALL ANGLE(C, PT, ALPHA)
NEWANG = ALPHA-90*SENS
IF (NEWANG.GE.360) NEWANG = NEWANG-360
IF (NEWANG.LT.0) NEWANG = NEWANG+360
CALL DIFFANG(LASTANG, NEWANG, DIFF, SENS)
CALL SAVECURVE(RMIN, DIFF, SENS, PT, NEWANG, C)
CALL STRAIGHT(DIST(PT, NAVAID))
ENDIF
RETURN
END

```

SUBROUTINE ANGLE(A,B,ALPHA)

REAL*8 ALPHA,A,B,PI,SALPHA,CALPHA,D
DIMENSION A(0:1),B(0:1)

PI=3.141529654

D=DIST(A,B)

IF (D.EQ.0) CALL ERREUR(23)

SALPHA=(B(1)-A(1))/D

CALPHA=(B(0)-A(0))/D

ALPHA=ASIN(SALPHA)*180/PI

IF (CALPHA.LT.0) ALPHA=180-ALPHA

IF (ALPHA.LT.0) ALPHA=360+ALPHA

RETURN

END


```

*****
* ARGUMENTS
* BETA          Angle de la droite D          Real*8
* NAVAIID       Point de la droite D          Array[0:1] of Real*8
* SENS          Sens du virage                 Integer
* RESULTATS
* Mise-a-jour des segments TRACK et des variables globales
* SPECIFICATIONS
* La sous-routine STURN assure la creation et le stockage des
* segments correspondant a la trajectoire suivie par un avion in-
* terceptant une radiale d'un angle BETA et rejoignant un point ap-
* pele NAVAIID par un virage dans le sens indique par SENS.
* L'interception se fait par deux virages de sens opposes.
* Le premier pour s'aligner sur un cap faisant un angle de 45 Deg
* avec le cap final, le deuxieme pour s'aligner sur le cap final.
*****
* DECLARATIONS
* NOM          TYPE DE VARIABLE          TYPE FORTRAN
* BETA          PARAMETRE(ARG)            REAL*8
* NAVAIID       PARAMETRE(ARG)            NAVAIID(0:1) REAL*8
* SENS          PARAMETRE(ARG)            INTEGER
* GAMMA         VAR LOCALE                REAL*8
*****
INCLUDE      "TRACK.CMN"

1 REAL*8      BETA,GAMMA,NAVAID,A,B,C,V,W,CT(0:1),
              PTA(0:1),PTB(0:1)
INTEGER      SENS,S
DIMENSION    NAVAIID(0:1)

CALL EQUATION(BETA,NAVAID,A,B,C)
V=A*LASTPT(0)+B*LASTPT(1)+C
IF (V.GT.0) THEN
    S=1
ELSE
    S=-1
ENDIF
IF (SENS.NE.S) CALL ERREUR(24)
GAMMA=BETA-45*SENS
IF (GAMMA.GE.360) GAMMA=GAMMA-360
IF (GAMMA.LT.0) GAMMA=GAMMA+360
CALL CTRVIR(CT,LASTANG,LASTPT,RMIN,SENS)
CALL FINVIR(CT,GAMMA,PTA,RMIN,SENS)
CALL CTRVIR(CT,BETA,PTA,RMIN,-SENS)
CALL FINVIR(CT,BETA,PTB,RMIN,-SENS)
W = A*PTB(0)+B*PTB(1)+C
IF (W*V.LE.0) CALL ERREUR(35)

```

CALL CAP(GAMMA,SENS)
CALL RADIALE(BETA,NAVAID,-SENS,.TRUE.)
RETURN
END


```

*****
* ARGUMENTS
*
* BETA          Angle (cap a atteindre)          Real*8
* SENS          Direction a prendre              Integer
*
* RESULTAT
* (Effet de bord) mise-a-jour des -segments TRACK
*                               -var globales
* SPECIFICATIONS
*
* La procedure effectue la creation et le stockage d'un segment
* courbe correspondant a la trajectoire suivie par un avion ef-
* fectuant un virage dans le sens indique par SENS (ou le sens
* oppose si il est plus approprie et si l'utilisateur est d'ac-
* cord) pour suivre le cap indique par l'angle BETA.
* Elle met egalement a jour les variables globales :
* cap de l'avion et position.
*
* PRECONDITIONS
*
* BETA = Angle(deg) 0<=BETA<360
* SENS = 1 ou -1
*****
* DECLARATIONS
*
* NDM          TYPE DE VARIABLE          TYPE FORTRAN
*
* BETA          PARAMETRE(ARG)          REAL*8
* SENS          PARAMETRE(ARG)          INTEGER
* LASTANG       COMMON                  REAL*8
* LASTPT        COMMON                  REAL*8 LASTPT(0:1)
* RMIN          COMMON                  REAL*8
* DIFF          VAR LOCALE              REAL*8
* C             VAR LOCALE              REAL*8 C(0:1)
* NEWPT         VAR LOCALE              REAL*8 NEWPT(0:1)
*****
INCLUDE          "TRACK.CMN"
REAL*8          BETA,C,NEWPT
INTEGER         SENS
DIMENSION       NEWPT(0:1),C(0:1)

CALL INSEREROR
IF (LASTANG.EQ.BETA) RETURN
CALL DIFFANG(LASTANG,BETA,DIFF,SENS)
CALL CTRVIR(C,LASTANG,LASTPT,RMIN,SENS)
CALL FINVIR(C,BETA,NEWPT,RMIN,SENS)
CALL SAVECURVE(RMIN,DIFF,SENS,NEWPT,BETA,C)
RETURN
END

SUBROUTINE EQUATION(CALPHA,PT,A,B,C)

```



```
*****
* ARGUMENTS
* ALPHA          Angle de la droite          Real*8
* PT             Point de la droite          Array[0..1] of Real*8  *
* RESULTATS
* A,B,C          Coefficients A,B,C de l'equation Ax+By+C=0 de la droite definie par ALPHA et PT , Real*8  *
* SPECIFICATIONS
* La sous-routine EQUATION calcule les coefficients A,B,C de l'equation Ax+By+C=0 de la droite definie par un angle ALPHA et un point PT
* Les relations suivantes sont appliquees :
*   A = -SIN(ALPHA)
*   B =  COS(ALPHA)
*   C =  PT(0)*SIN(ALPHA)-PT(1)*COS(ALPHA)
* PRECONDITIONS
* 0<= ALPHA < 360
*
*****
* DECLARATIONS
*
* NOM            TYPE DE VARIABLE            TYPE FORTRAN
* ALPHA          PARAMETRE(ARG)              REAL*8
* A,B,C          PARAMETRES(RES)              REAL*8
* PT             PARAMETRE(ARG)              PT(0:1) REAL*8
*****
REAL*8          ALPHA,PT,A,B,C
DIMENSION      PT(0:1)

A = -SIND(ALPHA)
B =  COSD(ALPHA)
C =  PT(0)*SIND(ALPHA)-PT(1)*COSD(ALPHA)
RETURN
END
```

```

*****
* ARGUMENTS
*
* ALPHA      Angle de la droite A      Real*8      *
* BETA       Angle de la droite B      Real*8      *
* PTA        Point appart. a A         Array[0..1] of Real*8 *
* PTB        Point appart. a B         Array[0..1] of Real*8 *
*
* RESULTATS
*
* I          Point d'intersect. de A et B   Array[0..1] of Real*8 *
*
* SPECIFICATIONS
*
* La sous-routine INTERSECT calcule le point d'intersection I *
* de deux droites A et B definies par
* A : angle ALPHA   pt PTA
* B : angle BETA    pt PTB
*
* PRECONITIONS
*
* 0 <= ALPHA < 360      0 <= BETA < 360
* ALPHA-BETA <> 180, <> -180, <> 0
*****
* DECLARATIONS
*
* NOM          TYPE DE VARIABLE          TYPE FORTRAN      *
*
* ALPHA,BETA   PARAMETRES(ARG)           REAL*8          *
* PTA,PTB,I    PARAMETRES(ARG)           ARRAY[0..1] OF REAL*8 *
* ADA,BDA,CDA, VARIABLES LOCALES         REAL*8          *
* ADB,BDB,CDB
*****

REAL*8  ALPHA,BETA,PTA,PTB,I,ADA,BDA,CDA,AOB,BOB,CDB,DEN
DIMENSION PTA(0:1),PTB(0:1),ICO(1)

CALL EQUATION(ALPHA,PTA,ADA,BDA,CDA)
CALL EQUATION(BETA,PTB,ADB,BDB,CDB)
DEN = ADA*BDB-ADB*BDA
IF (DEN.EQ.0) CALL ERREUR(26)
ICO = (BDA*CDB-BDB*CDA)/DEN
IC1 = (ADB*CDA-ADA*CDB)/DEN
RETURN
END

```



```

*****
* ARGUMENTS      A      point de la droite D      Array[0..1] of Real*8
*                B      point de la droite d      Array[0..1] of Real*8
*                ANGLE  angle de la droite D      Real*8
*
* RESULTAT      PREC                                Logical
*
* SPECIFICATIONS
*
* La sous-routine PREC est une fonction qui attribue a la variable
* PREC la valeur .TRUE. si A "precede" B sur la droite D et la valeur
* .FALSE. si A "ne precede pas" B sur la droite D
*
* soit D defini par deux points A et B et un angle ALPHA
* PREC(A,B,ANGLE) <=> (B(0)-A(1))*COS(ANGLE) >= 0 ET
*                    (B(1)-A(1))*SIN(ANGLE) >= 0
*
* PRECONDITION  Il existe une droite D definie par les 2 points A et
*                B t.q. D fait un angle ALPHA avec l'axe des X
*****
* DECLARATIONS
*
* NOM                TYPE DE VARIABLE                TYPE FORTRAN
*
* A,B                PARAMETRE(ARG)                    REAL*8 A(0:1) B(0:1)
* ANGLE              PARAMETRE(ARG)                    REAL*8
* PREC               FONCTION                          LOGICAL
*****
REAL*8  ANGLE,A,B
DIMENSION  A(0:1),B(0:1)
PREC = (((B(0)-A(0))*COSD(ANGLE)).GE.0).AND.
       (((B(1)-A(1))*SIND(ANGLE)).GE.0)
1  RETURN
END

```



```
SUBROUTINE RAYON(A, ALPHA, B, BETA, DIFF, SENS, R)
REAL*8          GAMMA, DIFF, I, A, R, ALPHA, B, BETA
DIMENSION       I(0:1), A(0:1), B(0:1)
INTEGER         SENS
CALL INTERSECT(ALPHA, A, BETA, B, I)
IF (DIFF.LT.180) THEN
    GAMMA=(180-DIFF)/2
ELSE
    GAMMA=(DIFF-180)/2
ENDIF
R = DIST(A, I)*TAND(GAMMA)
RETURN
END
```

```

*****
* ARGUMENTS      ALPHA    un angle      REAL*8
*                BETA     un angle      REAL*8
*                SENS     Sens du virag. INTEGER
*
* RESULTATS      DIFF     un angle      REAL*8
*
* SPECIFICATIONS
*
* DIFF est un angle de 0 a 360 deg. qui correspond a la dif-
* ference : BETA - ALPHA.
* Le sens adopte est conventionnellement le sens inverse des
* aiguilles d'une montre, ce qui correspond a un angle positif.
*
* PRECONDITIONS
*
* 0 <= ALPHA <= 360
* 0 <= BETA <= 360
* ALPHA et BETA en degres
*****

```

```

*****
* DECLARATIONS
*
* NOM                TYPE DE VARIABLE          TYPE FORTRAN
*
* ALPHA,BETA         PARAMETRES(ARG)           REAL*8
* DIFF               PARAMETRES(RES)           REAL*8
* SENS               PARAMETRE (ARG)           INTEGER
*
*****

```

```

REAL*8 ALPHA,BETA,DIFF
INTEGER SENS

```

```

IF (SENS.EQ.-1) DIFF=BETA-ALPHA
IF (SENS.EQ.1) DIFF=ALPHA-BETA
IF (DIFF.LT.0) DIFF=DIFF+360
RETURN
END
*****

```

REAL*8 FUNCTION DIST(A,B)

```
*****
* ARGUMENTS      A      un point du plan      array[0..1] of real*8*
*                B      un point du plan      array[0..1] of real*8*
* RESULTATS      DIST    une longueur          Real*8          *
* SPECIFICATIONS
```

```
* DIST recoit lors de l'activation de la fonction DIST la valeur de *
* la distance qui separe les deux points A et B                      *
```

```
*****
* DECLARATIONS
```

NOM	TYPE DE VARIABLE	TYPE FORTRAN
A	PARAMETRE(ARG)	REAL*8 A(0:1)
B	PARAMETRE(ARG)	REAL*8 B(0:1)
DIST	FONCTION	REAL*8

REAL*8 A,B

DIMENSION A(0:1),B(0:1)

DIST = SQRT((B(0)-A(0))**2 + (B(1)-A(1))**2)

RETURN

END

SUBROUTINE PTDEPART(A, ALPHA, B, BETA, DIFF, I, SENS, PTDEP)

* ARGUMENTS

* A Pt de la droite D Array[0:1] of Real*8
 * ALPHA Angle de la droite D Real*8
 * B Pt de la droite D' Array[0:1] of Real*8
 * BETA Angle de la droite D' Real*8
 * SENS Sens du virage Integer
 * DIFF Angle du virage Real*8
 * I Point d'intersection Array[0:1] of Real*8

* RESULTATS

* PTDEP Point de depart du virage Array[0:1] of Real*8

* SPECIFICATIONS

* Si les preconditions sont respectees, PTDEP constitue le resultat de la procedure et represente le point de depart en coordonnees cartesiennes du virage d'un rayon R d'un avion se trouvant sur une droite D(A, ALPHA) et desirant rejoindre un point B situe sur la droite D'(B, BETA).

* PRECONDITIONS

* 0 <= ALPHA, BETA < 360

* SENS = 1 OU -1

* DECLARATIONS

NOM	TYPE DE VARIABLE	TYPE FORTRAN
A	PARAMETRE(ARG)	A(0:1) REAL*8
B	PARAMETRE(ARG)	B(0:1) REAL*8
ALPHA	PARAMETRE(ARG)	REAL*8
BETA	PARAMETRE(ARG)	REAL*8
SENS	PARAMETRE(ARG)	INTEGER
PTDEP	PARAMETRE(RES)	PTDEP(0:1) REAL*8
D	VAR LOCALE	REAL*8
DPRIME	VAR LOCALE	REAL*8
DIFF	PARAMETRE(ARG)	REAL*8
I	PARAMETRE(ARG)	I(0:1) REAL*8

INCLUDE "TRACK.CMN"

REAL*8 ALPHA, BETA, A, B, PTDEP, D, DPRIME, DIFF, I
 INTEGER SENS
 DIMENSION A(0:1), B(0:1), I(0:1), PTDEP(0:1)

GAMMA=(180-DIFF)/2
 D=RMIN/SIND(GAMMA)
 DPRIME=D*COSD(GAMMA)
 PTDEP(0)=I(0)-COSD(ALPHA)*DPRIME
 PTDEP(1)=I(1)-SIND(ALPHA)*DPRIME
 RETURN
 END

SUBROUTINE CTRVIR(C,ALPHA,PT,DIST,SENS)

```

*****
* ARGUMENTS
*
* ALPHA      Angle de la droite D      Real*8
* PT         Pt de la droite D         Array[0:1] of Real*8
* DIST       Rayon du virage            Real*8
* SENS       Sens du virage             Integer
*
* RESULTATS
*
* C          Centre du virage           Array[0:1] of Real*8
*
* SPECIFICATIONS
*
* CTRVIR calcule et affecte a C les coordonnees cartesiennes d'
* un point situe a une distance DIST d'un point PT et sur une
* droite perpendiculaire a la droite D (d'angle ALPHA et de point
* PT) du cote indique par sens.
*
* PRECONDITIONS
*
* ALPHA      angle en degres      0 <= ALPHA < 360
* DIST       positif
* SENS       1 ou -1
*****
* DECLARATIONS
*
* NOM          TYPE DE VARIABLE          TYPE FORTRAN
*
* C            PARAMETRE(RES)             C(0:1) REAL*8
* ALPHA        PARAMETRE(ARG)             REAL*8
* PT           PARAMETRE(ARG)             PT(0:1) REAL*8
* DIST         PARAMETRE(ARG)             REAL*8
* SENS         PARAMETRE(ARG)             INTEGER
*****

REAL*8 C,ALPHA,PT,DIST
INTEGER SENS
DIMENSION C(0:1),PT(0:1)

C(0)=PT(0)+SENS*SIND(ALPHA)*DIST
C(1)=PT(1)-SENS*COSD(ALPHA)*DIST
RETURN
END
*****

```



```

*****
* ARGUMENTS
*
* C      Point      Array(0:1) of Real*8
* BETA   Angle      Real*8
* DIST   Rayon du virage Real*8
* SENS   Sens du virage Integer
*
* RESULTAT
*
* NEWPT Point      Array(0:1) of Real*8
*
* SPECIFICATIONS
*
* NEWPT est le point situe a l'intersection de la droite D
* definie par l'angle BETA et la droite definie par un pt C
* et une distance R. Cette droite etant perpendiculaire a D
* et le point C se trouvant du cote de la droite D indique par sens.
*
* PRECONDITIONS
*
* BETA      Angle en degre      0<=BETA<360
* DIST      Positif
* SENS      1 ou -1
*****
*****
* DECLARATIONS
*
* NOM      TYPE DE VARIABLE      TYPE FORTRAN
*
* C      PARAMETRE(ARG)      C(0:1) REAL*8
* BETA   PARAMETRE(ARG)      REAL*8
* NEWPT  PARAMETRE(RES)      NEWPT(0:1) REAL*8
* DIST   PARAMETRE(ARG)      REAL*8
* SENS   PARAMETRE(ARG)      INTEGER
*****
REAL*8      C,BETA,NEWPT,DIST
INTEGER     SENS
DIMENSION   C(0:1),NEWPT(0:1)

NEWPT(0)=C(0)-DIST*SIN(BETA)*SENS
NEWPT(1)=C(1)+DIST*COS(BETA)*SENS
RETURN
END

```


SUBROUTINE SAVESTRAIGHT(DIST,ENDPOINT)

```

*****
* ARGUMENTS
* DIST          Une longueur          Real*8
* ENDPOINT      Un point              Array[0:1] of Real*8
* RESULTATS
* Mise-a-jour du tableau des segments
* SPECIFICATIONS
* DIST est une distance en KM = longueur d'un segment droit
* ENDPOINT est un point du plan en coord. cartésiennes repre-
* sentant la fin d'un segment droit
* 1. si le dernier segment stocké est un segment courbe
*    alors si il y a déjà 15 segments stockés alors erreur
*        alors erreur
*        sinon on stocke le segment droit dont la longueur est
*            DIST (converti en NM)
*    sinon on modifie le dernier segment en prolongeant sa lon-
*        gueur par DIST
* 2. on met à jour la variable globale LASTPT
*****
* DECLARATIONS
*
* NOM          TYPE DE VARIABLE          TYPE FORTRAN
* TRACK        COMMON                   TRACK(1:15,1:2) REAL*8
* ITRACK       COMMON                   INTEGER
* MAXTRACK     COMMON                   INTEGER
* LASTPT       COMMON                   LASTPT(0:1) REAL*8
* DIST         PARAMETRE(ARG)           REAL*8
* ENDPOINT     PARAMETRE(ARG)           ENDPOINT(0:1) REAL*8
*****
INCLUDE        "TRACK.CMN"

REAL*8        DIST,ENDPOINT
DIMENSION     ENDPOINT(0:1)

IF (TRACK(ITRACK,2).EQ.0) THEN
    TRACK(ITRACK,1)=CONVNM(DIST)+TRACK(ITRACK,1)
ELSE
    ITRACK=ITRACK+1
    IF (ITRACK.GT.MAXTRACK) CALL ERREUR(27)
    TRACK(ITRACK,1)=CONVNM(DIST)
    TRACK(ITRACK,2)=0.0
    IPOINT=IPOINT+1
ENDIF
POINTS(IPOINT,0)=ENDPOINT(0)
POINTS(IPOINT,1)=ENDPOINT(1)
LASTPT(0)=ENDPOINT(0)
LASTPT(1)=ENDPOINT(1)
RETURN
END

```

```

*****
* ARGUMENTS
*
* RAYON      Une longueur      Real*8
* ANGLE      Un angle          Real*8
* SENS       Sens du virage    Integer
* ENDPOINT   Un point          Array[0:1] of Real*8
* ENDANGLE   L'angle final     Real*8
* CTR        Le centre du virage Array[0:1] of Real*8
*
* RESULTATS
*
* Mise-a-jour du tableau des segments
*
* SPECIFICATIONS
*
* Si il y a deja 15 segments stockes
* alors erreur
* sinon 1. stockage du segment courbe avec
*         conversion du rayon en NM
*        2. mise-a-jour des variables globales
*****
* DECLARATIONS
*
* NOM          TYPE DE VARIABLE          TYPE FORTRAN
*
* TRACK        COMMON                    TRACK(1:15,1:2) REAL*8
* ITRACK        COMMON                    INTEGER
* MAXTRACK      COMMON                    INTEGER
* LASTANG       COMMON                    REAL*8
* LASTPT        COMMON                    LASTPT(0:1) REAL*8
* RAYON         PARAMETRE(ARG)            REAL*8
* ANGLE         PARAMETRE(ARG)            REAL*8
* ENDANGLE      PARAMETRE(ARG)            REAL*8
* SENS          PARAMETRE(ARG)            INTEGER
* ENDPOINT      PARAMETRE(ARG)            ENDPOINT(0:1) REAL*8
*****
INCLUDE        'TRACK.CMN'

REAL*8        RAYON,ANGLE,ENDPOINT,CONVNM,CTR(0:1)
INTEGER       SENS
DIMENSION     ENDPOINT(0:1)

ITRACK=ITRACK+1
IF (ITRACK.GT.MAXTRACK) CALL ERREUR(28)
TRACK(ITRACK,1)=ANGLE
TRACK(ITRACK,2)=SENS*CONVNM(RAYON)
LASTPT(0)=ENDPOINT(0)
LASTPT(1)=ENDPOINT(1)
LASTANG =ENDANGLE
ICTR=ICTR+1
CENT(ICTR,0)=CTR(0)
CENT(ICTR,1)=CTR(1)
IPOINT=IPOINT+1
POINTS(IPOINT,0)=ENDPOINT(0)

```



```
POINTS(IPOINT,1)=ENDPOINT(1)  
RETURN  
END
```

SUBROUTINE INSERERDR

```

*****
* ARGUMENTS
*
* AUCUN
*
* RESULTATS
*
* Mise-a-jour eventuelle du tableau des segments et de la
* variable globale LASTPT
*
* SPECIFICATIONS
*
* Si le dernier segment stocke est courbe
* alors on stocke un segment droit de longueur 0.1 KM
*****
*****
* DECLARATIONS
*
* NOM                TYPE DE PARAMETRE                TYPE FORTRAN
*
* TRACK              COMMON                            TRACK(1:15,1:2) REAL*8
* ITRACK             COMMON                            INTEGER
*****
INCLUDE              "TRACK.CMN"
IF (TRACK(ITRACK,2).NE.0) CALL STRAIGHT(0.1)
RETURN
END
*****

```

```

SUBROUTINE  MODIF(PTDEP,POSSIBLE)
INCLUDE      "TRACK.CMN"
REAL*8  PTDEP(0:1),AVDER(0:1)
LOGICAL POSSIBLE,PREC
D=DIS(TLASTPT,PTDEP)
IF (PREC(TLASTPT,PTDEP,LASTANG)) THEN
    IF (D.GE.5) THEN
        POSSIBLE = .FALSE.
    ELSE
        CALL SAVESTRAIGHT(D,PTDEP)
    ENDIF
    ELSE
    IF ((TRACK(ITRACK,2).EQ.0).AND.
        (D.LT.CONVKM(TRACK(ITRACK,1)))) THEN
        CALL SAVESTRAIGHT(-D,PTDEP)
    ELSE
        POSSIBLE=.FALSE.
    ENDIF
ENDIF
RETURN
END

```

REAL*8 FUNCTION CONVM(KM)

REAL*8 KM

CONVM=KM/1.85313

RETURN

END

REAL*8 FUNCTION CONVKM(NM)

-Go-

REAL*8 NM

CONVKM=NM*1.35313

RETURN

END

SUBROUTINE CLOTURE

-G1-

INCLUDE 'GEN.CMN'
INCLUDE 'TRACK.CMN'

CHARACTER*10 FCAN,FPLO

FCAN=AIRPORT(1:3)//'CAN.DAT'
FPLO=AIRPORT(1:3)//'PLD.DAT'

*****SAUVETAGE FICHIER FCAN *****

```

OPEN(UNIT=11,FILE=FCAN,STATUS='NEW',ERR=100)
WRITE(11,5) SIDLAB
FORMAT(A10)
WRITE(11,10) RWY
FORMAT(I2)
WRITE(11,15) RWYCAR(0),RWYCAR(1),RECCAR(0),RECCAR(1)
FORMAT(4(1X,F15.3))
WRITE(11,20) ITRACK
FORMAT(I2)
WRITE(11,25) (TRACK(I,1),TRACK(I,2),I=1,ITRACK)
FORMAT(F15.8,1X,F15.8)
WRITE(11,30) NBNVAV
FORMAT(I2)
WRITE(11,35) (NAVUSED(I,1),NAVUSED(I,2),NAVCOORD(I,0),
              NAVCOORD(I,1),I=1,NBNVAV)
FORMAT(2(A10,1X),2F15.3)
CLOSE(UNIT=11)

```


***** SAUVETAGE FICHIER FPLO *****

```
0 OPEN(UNIT=12,FILE=FPLO,STATUS='NEW',ERR=100)
  WRITE(12,40)SIDLAB
  FORMAT(A10)
  WRITE(12,42)RWY
  FORMAT(I2)
  WRITE(12,45)IPDINT
  FORMAT(I2)
  WRITE(12,50)(PDINTS(I,0),PDINTS(I,1),I=1,IPDINT)
  FORMAT(2F15.3)
  WRITE(12,55)ICTR
  FORMAT(I2)
  WRITE(12,60)(CENT(I,0),CENT(I,1),I=1,ICTR)
  FORMAT(2F15.3)
  CLOSE(UNIT=12)
  RETURN
00 CALL ERREUR(29)
  RETURN
  END
```