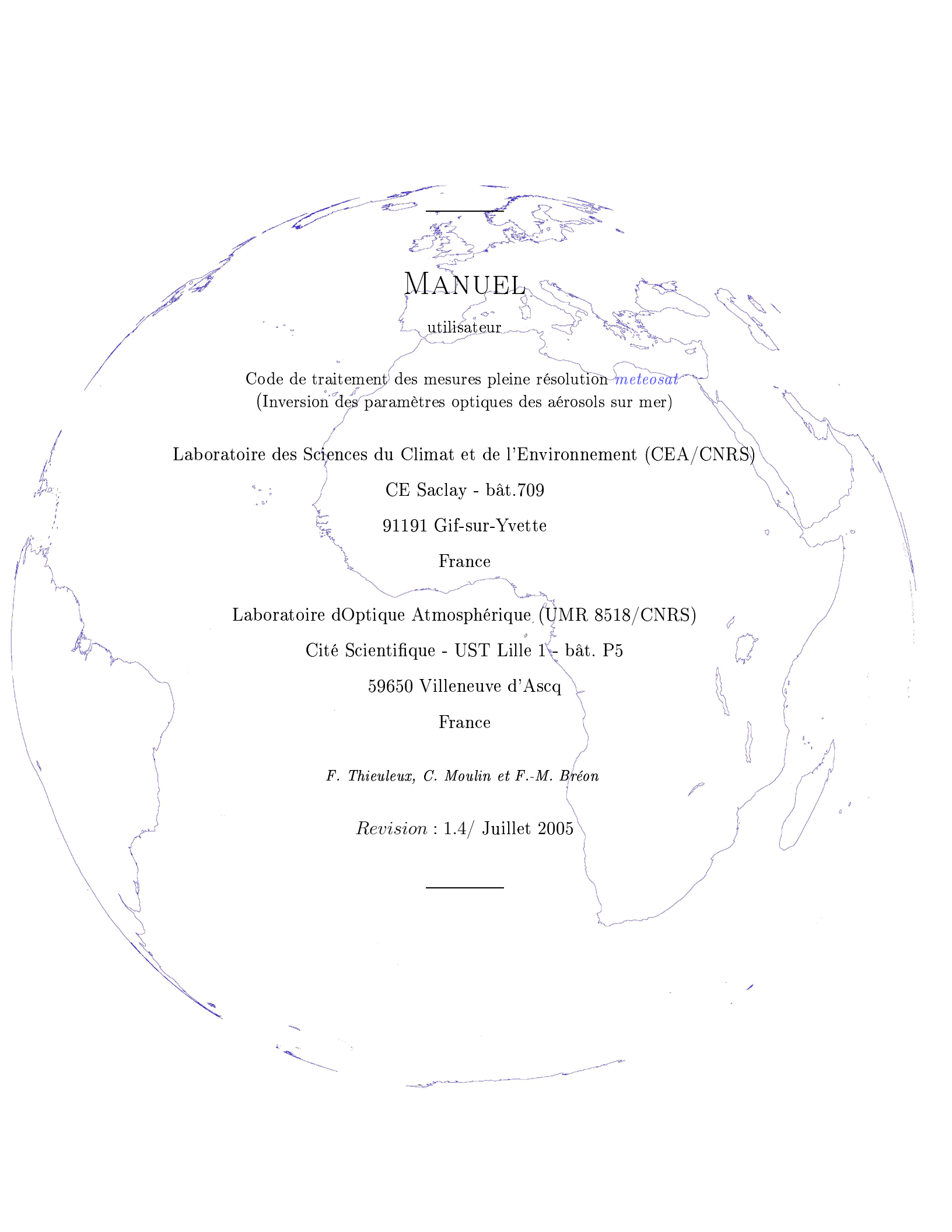




MANUEL

utilisateur

Code de traitement des mesures pleine résolution *meteosat*
(Inversion des paramètres optiques des aérosols sur mer)

Laboratoire des Sciences du Climat et de l'Environnement (CEA/CNRS)

CE Saclay - bât.709

91191 Gif-sur-Yvette

France

Laboratoire d'Optique Atmosphérique (UMR 8518/CNRS)

Cité Scientifique - UST Lille 1 - bât. P5

59650 Villeneuve d'Ascq

France

F. Thieuleux, C. Moulin et F.-M. Bréon

Revision : 1.4/ Juillet 2005

Je travaille face à l'océan, auquel je suis promis et déjà, livré.
Mais adossé à une autre mer, sur laquelle je n'ai pas le droit de m'embarquer encore,
mais que cependant je regarde parfois, à la dérobée, par dessus l'épaule.
L'ourlet d'écume, qui frange la baie du Lévrier, et celui des sables, l'Atlantique et le Sahara.
L'équilibre, sur cette mince cloison, est instable :
de quel côté vais-je tomber ?

Théodore Monod, *Méharées*.

BORDEREAU D'INDEXATION

Caractéristiques du document	
Volume	1
Nombre de pages liminaires	8
Nombre total de pages	105
Nombre d'annexes	0



DIFFUSION

Sigle	Nom
LSCE	C. Moulin

[illegible]



Code de traitement des mesures pleine résolution *meteosat*

Inversion des paramètres optiques des aérosols sur mer

F. Thieuleux et al.

Juillet 2005 — Page iv — *Revision* : 1.4



Sommaire

Sommaire	1
Introduction	7
1 Documentation	9
1.1 Format des données.	9
1.2 Code 6S	10
2 Installation du programme	11
2.1 Environnement matériel et logiciel	11
2.1.1 Langages de programmation utilisés	11
2.1.2 Système d'exploitation supporté	11
2.1.3 Standard et conformité	12
2.1.4 Règles syntaxiques	12
2.2 Arborescence des répertoires	12
2.3 Génération des exécutables	14
3 Mode opératoire	17
3.1 Définition des variables d'environnement	17
3.2 Paramètres de simulation - Arguments du programme	17
3.3 Constantes de simulation	27
3.3.1 Constantes du module <code>M_PARAMETRES_CHAINES</code>	30
3.3.2 Constantes du module <code>M_CONSTANTES</code>	30
3.4 Exemple de shell de lancement du code <i>meteosat</i>	30
3.5 Temps d'exécution.	30
4 Descriptif du logiciel	31
4.1 Structure du code	31
4.2 Format des fichiers de données en entrée	31
4.3 Liste des programmes	31
4.4 Liste des fonctions du répertoire <code>met</code>	31
4.4.1 Rappel sur les apports de fiabilité de programmation en Fortran 95	34
4.4.2 Fonction <code>addslash</code>	34
4.4.3 Fonction <code>allouer</code>	35
4.4.4 Fonction <code>allouer_synthese</code>	35
4.4.5 Fonction <code>desallouer</code>	35
4.4.6 Fonction <code>desallouer_synthese</code>	36
4.4.7 Fonction <code>decoder_argument</code>	36



4.4.8	Fonction <code>decoder_arguments</code>	36
4.4.9	Fonction <code>decoder_plage_horaire</code>	37
4.4.10	Fonction <code>decoder_imodele</code>	37
4.4.11	Fonction <code>decoder_bornes</code>	37
4.4.12	Fonction <code>decoder_date</code>	38
4.4.13	Fonction <code>controler_arguments</code>	38
4.4.14	Fonction <code>afficher_arguments</code>	38
4.4.15	Fonction <code>initialiser_arguments</code>	39
4.4.16	Fonction <code>initialiser_ligne_bandeau</code>	39
4.4.17	Fonction <code>initialiser_bandeau</code>	39
4.4.18	Fonction <code>reporter_bandeau</code>	39
4.4.19	Fonction <code>charger_bandeau</code>	40
4.4.20	Fonction <code>calculer_angstrom</code>	40
4.4.21	Fonction <code>calculer_coeff_interp_geom</code>	40
4.4.22	Fonction <code>calculer_fit</code>	40
4.4.23	Fonction <code>calculer_param_pixel</code>	41
4.4.24	Fonction <code>charger_table_mol</code>	41
4.4.25	Fonction <code>charger_table_frn</code>	41
4.4.26	Fonction <code>charger_table_tot</code>	41
4.4.27	Fonction <code>charger_table_fit</code>	42
4.4.28	Fonction <code>charger_tables</code>	42
4.4.29	Fonction <code>coder_tau550</code>	42
4.4.30	Fonction <code>coder_angstrom</code>	42
4.4.31	Fonction <code>coder_rho</code>	43
4.4.32	Fonction <code>compresser_fichiers</code>	43
4.4.33	Fonction <code>decompresser_fichiers</code>	43
4.4.34	Fonction <code>compresser_fichiers_sortie</code>	43
4.4.35	Fonction <code>ecrire_ligne</code>	44
4.4.36	Fonction <code>etalonner</code>	44
4.4.37	Fonction <code>extraire_pixel_produit</code>	45
4.4.38	Fonction <code>extraire_produit</code>	45
4.4.39	Fonction <code>fermer_fichiers1</code>	45
4.4.40	Fonction <code>fermer_fichiers2</code>	45
4.4.41	Fonction <code>filtrer_nuage</code>	45
4.4.42	Fonction <code>f_poly</code>	45
4.4.43	Fonction <code>pythag</code>	45
4.4.44	Fonction <code>svdfit</code>	46
4.4.45	Fonction <code>gausslegendre</code>	46
4.4.46	Fonction <code>glitter</code>	46
4.4.47	Fonction <code>tester_glitter</code>	47
4.4.48	Fonction <code>initialiser_variables</code>	47
4.4.49	Fonction <code>initialiser_parametres_codage</code>	48
4.4.50	Fonction <code>ouvrir_fichier</code>	48
4.4.51	Fonction <code>ouvrir_fichiers_ancillaire</code>	48
4.4.52	Fonction <code>ouvrir_fichiers_tables</code>	48
4.4.53	Fonction <code>initialiser</code>	48
4.4.54	Fonction <code>initialiser_day</code>	48

4.4.55	Fonction <code>initialiser_mth</code>	49
4.4.56	Fonction <code>initialiser_ext</code>	49
4.4.57	Fonction <code>interpoler_aerosol</code>	49
4.4.58	Fonction <code>interpoler</code>	49
4.4.59	Fonction <code>interpoler_moleculaire</code>	49
4.4.60	Fonction <code>inverser_aerosol</code>	49
4.4.61	Fonction <code>faire_nom_fichier_header</code>	49
4.4.62	Fonction <code>lire_header</code>	50
4.4.63	Fonction <code>faire_nom_fichier_sortie</code>	50
4.4.64	Fonction <code>ouvrir_fichiers_sortie</code>	50
4.4.65	Fonction <code>lire_fichier_entree</code>	50
4.4.66	Fonction <code>lire_fichier_mie_6S</code>	50
4.4.67	Fonction <code>faire_nom_fichier_image</code>	51
4.4.68	Fonction <code>lire_ligne</code>	51
4.4.69	Fonction <code>faire_liste</code>	51
4.4.70	Fonction <code>ouvrir_fichier_log</code>	51
4.4.71	Fonction <code>fermer_fichier_log</code>	52
4.4.72	Fonction <code>ouvrir_fichier_message</code>	52
4.4.73	Fonction <code>fermer_fichier_message</code>	52
4.4.74	Fonction <code>ecrire_message</code>	52
4.4.75	Fonction <code>polynome2</code>	53
4.4.76	Fonction <code>polynome3</code>	53
4.4.77	Fonction <code>polynome4</code>	53
4.4.78	Fonction <code>polynome5</code>	53
4.4.79	Fonction <code>polynome6</code>	53
4.4.80	Fonction <code>faire_nom_produit</code>	53
4.4.81	Fonction <code>ouvrir_fichier_produit</code>	54
4.4.82	Fonction <code>lire_fichier_produit</code>	54
4.4.83	Fonction <code>ecrire_fichier_produit</code>	54
4.4.84	Fonction <code>fermer_fichier_produit</code>	54
4.4.85	Fonction <code>read_header_msg</code>	55
4.4.86	Fonction <code>read_header_mfg</code>	55
4.5	Liste des fonctions du répertoire <code>com</code>	56
4.5.1	Fonction <code>bunzip2file</code>	57
4.5.2	Fonction <code>bzip2file</code>	57
4.5.3	Fonction <code>gunzipfile</code>	57
4.5.4	Fonction <code>gzipfile</code>	58
4.5.5	Fonction <code>jj2jma</code>	58
4.5.6	Fonction <code>jma2jj</code>	58
4.5.7	Fonction <code>a2j</code>	58
4.5.8	Fonction <code>julday</code>	58
4.5.9	Fonction <code>caldat</code>	59
4.5.10	Fonction <code>cds2jma</code>	59
4.5.11	Fonction <code>cds2jj</code>	59
4.5.12	Fonction <code>cds2jmahms</code>	59
4.5.13	Fonction <code>dsol</code>	60
4.5.14	Fonction <code>localiser1</code>	60



4.5.15	Fonction <code>localiser2</code>	60
4.5.16	Fonction <code>ligcol2latlon_geo</code>	61
4.5.17	Fonction <code>latlon2ligcol_geo</code>	61
4.5.18	Fonction <code>ligcol2tu_geo</code>	61
4.5.19	Fonction <code>calc_grid</code>	62
4.5.20	Fonction <code>elevation_geo</code>	62
4.5.21	Fonction <code>azimut_geo</code>	63
4.5.22	Fonction <code>ordre_octets.c</code>	63
4.5.23	Fonction <code>possol</code>	63
4.5.24	Fonction <code>swap_float</code>	63
4.5.25	Fonction <code>swap_float_buf</code>	64
4.5.26	Fonction <code>swap_double</code>	64
4.5.27	Fonction <code>swap_double_buf</code>	64
4.5.28	Fonction <code>swap_int</code>	65
4.5.29	Fonction <code>swap_int_buf</code>	65
4.5.30	Fonction <code>swap_uint</code>	65
4.5.31	Fonction <code>swap_uint_buf</code>	65
4.5.32	Fonction <code>swap_long</code>	65
4.5.33	Fonction <code>swap_long_buf</code>	65
4.5.34	Fonction <code>swap_short</code>	66
4.5.35	Fonction <code>swap_short_buf</code>	66
4.5.36	Fonction <code>swap_ushort</code>	66
4.5.37	Fonction <code>swap_ushort_buf</code>	66
4.6	Liste des fonctions du répertoire <code>mfg</code> .	66
4.7	Liste des fonctions du répertoire <code>msg</code> .	67
4.7.1	Fonction <code>msg_read</code>	67
4.7.2	Fonction <code>read_bsq_data</code>	68
4.7.3	Fonction <code>read_nat_data</code>	68
4.7.4	Fonction <code>swap_gp_pk_msg</code>	68
4.7.5	Fonction <code>swap_header_msg</code>	68
4.7.6	Fonction <code>swap_trailer_msg</code>	68
4.7.7	Fonction <code>tenbit2short</code>	69
4.7.8	Fonction <code>write_log_msg</code>	69
4.8	Contrôle des erreurs	69
5	Principe physique de l'algorithme.	71
5.1	Introduction.	71
5.2	Sélection des modèles d'aérosols.	71
5.3	Principe de l'algorithme en diffusion multiple.	71
6	Perspectives.	75
6.1	Améliorations du code.	75
6.1.1	Calcul précis de l'absorption gazeuse	75
6.1.2	Utilisation du canal à $1.64 \mu\text{m}$ pour l'inversion des propriétés des aérosols.	75
6.1.3	Basculement vers un module d'inversion "poussières désertiques".	75
6.1.4	Lecture du format <i>HRIT/LRIT</i>	75

A Exemple de script de lancement du code <i>meteosat</i>	77
B Précisions sur le masque Terre/Mer/Côte.	83
C Génération des tables précalculées.	85
C.1 Introduction	85
C.2 Utilisation du code de transfert radiatif <i>6S</i>	85
C.3 Hypothèses de simulations.	86
C.3.1 Géométrie de visée.	86
C.3.2 Conditions spectrales.	86
C.3.3 Modèles d'aérosols.	86
C.3.4 Tables de $\rho_{tot}^{mod}(\theta_s, \theta_v, \phi, \tau, \Delta\lambda)$	86
C.3.5 Table de la transmission gazeuse $t_{gaz}(\theta_s, \theta_v, k, \Delta\lambda)$	89
D Exemple de résultats en sortie du code <i>meteosat</i>.	91
D.1 Introduction	91
D.2 Épaisseur optique aérosol, instrument <i>Meteosat-05</i> , 5 Septembre 1994.	91
D.3 Épaisseur optique aérosol, instrument <i>Meteosat-07</i> , 7 Juin 2003.	92
D.4 Épaisseur optique aérosol, instrument <i>Meteosat-07</i> , 7 Juin 2003.	92
D.5 Produits aérosol dérivés de l'instrument <i>Meteosat-08</i> , 7 Juin 2003.	94
D.6 Produits aérosol mensuel dérivés de l'instrument <i>Meteosat-08</i> , Juin 2003.	94
Acronymes	98
Listes des symboles	100
Bibliographie	102
Documents de référence	105



Introduction

Ce manuel est destiné essentiellement à servir de référence pour l'installation et l'utilisation du code *meteosat*. Il permet de décrire à l'attention des développeurs sa structure interne, et la manière dont il est organisé, dans le but de favoriser des améliorations futures.

Le code *meteosat* permet de réaliser l'inversion des paramètres aérosols, essentiellement l'épaisseur optique à une longueur d'onde, et le coefficient d'Angström si possible¹. L'absorption gazeuse, la contribution du signal moléculaire et des aérosols, tous deux en diffusion simple et multiple, sont pris en compte et calculés grâce au code de transfert radiatif 6S² [*Vermote et al. 1997*].

Le premier chapitre renseigne l'utilisateur sur le support de documentations annexes servant à comprendre le format des données Meteosat, et le code de transfert radiatif 6S. Le chapitre 2 a pour objectif de présenter la démarche d'installation et de mise en application du code. Le chapitre 3 s'attache à décrire de manière précise les variables d'environnement, arguments du programme, ainsi que les constantes de simulation. Le quatrième chapitre précise la structure du code et s'attache à la description des fonctions. Enfin, différentes annexes renseignent l'utilisateur sur des points particuliers, A montre un exemple de shell de lancement du code *meteosat*.

Enfin, il est à noter que ce présent manuel a été construit en utilisant L^AT_EX³. L^AT_EX est un formateur de texte d'une très grande polyvalence et qui produit des documents d'une excellente qualité. Il est utilisé par un nombre toujours plus grand d'étudiants, de chercheurs, et d'éditeurs à travers le monde. C'est pour ces différentes raisons que le choix s'est donc porté sur L^AT_EX pour la réalisation de ce document. La version utilisée est L^AT_EX 2_ε 2000/06/01.

¹Le calcul du coefficient d'Angstrom n'est possible que si le capteur satellite mesure la réflectance totale dans au moins deux bandes spectrales dans le domaine visible / proche infrarouge, ce qui est le cas à partir du capteur METEOSAT-8 et des capteurs à venir

²Second Simulation of th Satellite Signal in th Solar Spectrum

³L^AT_EX est une extension d'un formateur de texte appelé T_EX, nom qui vient du mot grec $\eta\tau\epsilon\kappa\nu\eta$, signifiant "art"



Chapitre 1

Documentation

Le présent document doit normalement se trouver dans le répertoire `$METrac/met/doc/man`. La dernière version aux formats *Postscript*[®] et *PDF*[®] s'y trouve. ¹.

1.1 Format des données.

Des documents de référence fournis par l'agence *EUMETSAT* sur la description du format des données Meteosat (format *OpenMTP* pour les données antérieures à Meteosat-8, format *NATIVE* ou *BSQ*), ont été utilisés pour l'élaboration de ce code. Une version électronique² de ces documents au format *PDF* est disponible en local dans le répertoire `$METrac/met/doc/anc/`³.

- `$METrac/met/doc/anc/mtp/pdf/fg01.pdf` : Description complète du format *OpenMTP*,
- `$METrac/met/doc/anc/msg/pdf/icd_105_20.pdf` : Description complète du format 1.5 natif *MSG*,
- `$METrac/met/doc/anc/msg/pdf/MSG_native_format_1_5.pdf` : Description succincte du format 1.5 natif *MSG*,
- `$METrac/met/doc/anc/msg/pdf/MSG_bsq_format_1_5.pdf` : Description succincte du format 1.5 *BSQ MSG*.
- `$METrac/met/doc/anc/msg/pdf/ten_093_11.pdf` : Segment sol *MSG*.
- `$METrac/met/doc/anc/msg/pdf/spe_055_13.pdf` : Spécification détaillée du segment sol *MSG*.
- `$METrac/met/doc/anc/msg/pdf/cgms_03_26.pdf` : Spécification globale du format *LRIT/HRIT*.
- `$METrac/met/doc/anc/msg/pdf/ten_247_40.pdf` : Glossaire.

Une version en ligne de ces fichiers est disponible sur le site de l'agence Eumetsat aux adresses suivantes :

- <http://www.eumetsat.de/en/dps/archive/documentation/pdf/fig01.pdf>
- http://www.eumetsat.de/en/area2/publications/icd_105_20.pdf
- http://www.eumetsat.de/en/area2/publications/MSG_native_format_1_5.pdf
- http://www.eumetsat.de/en/area2/publications/MSG_bsq_format_1_5.pdf
- http://www.eumetsat.de/en/area2/publications/ten_093_11.pdf

¹Respectivement dans les sous-répertoires *ps* et *pdf*

²Ils sont également consultables sous forme papier (classeur rouge intitulé "Meteosat Second Generation - Documentation").

³*anc* pour *ancillaire*



- http://www.eumetsat.de/en/area2/publications/cgms_03_26.pdf
- http://www.eumetsat.de/en/area2/publications/spe_055_13.pdf
- http://www.eumetsat.de/en/area2/publications/ten_247_40.pdf

1.2 Code 6S

Le code *meteosat* s'appuie en partie sur l'utilisation d'un code de transfert radiatif, 6S (Second Simulation of the Satellite Signal in the Solar Spectrum, [Vermote et al. 1997]). 6S permet de simuler de manière précise dans le domaine spectral $[0,2\mu\text{m}-4\mu\text{m}]$, les processus de diffusion et absorption d'une atmosphère surmontant une surface ayant des propriétés radiatives quelconque. Nous reviendrons plus en détails dans l'annexe C sur l'utilisation de ce code à propos de ce qui nous intéresse ici.

La version 4.2 de la documentation de 6S est disponible en version électronique (format Postscript) en local dans le répertoire `$METrac/6s/doc/`. Une version papier est également consultable.

Chapitre 2

Installation du programme

2.1 Environnement matériel et logiciel

2.1.1 Langages de programmation utilisés

L'essentiel du code de traitement *meteosat* est écrit en langage Fortran 95. Néanmoins, les fonctions de lecture et décodage des fichiers bruts (format OpenMTP ou format NATIVE L1.5) sont écrites en C, ainsi que certaines fonctions communes génériques. Le choix de ces langages se justifie. D'une part, comme cela est décrit au paragraphe 2.1.2, ils offrent un haut niveau de portabilité. D'autre part, ils sont parfaitement adaptés au traitement scientifique¹.

2.1.2 Système d'exploitation supporté

Le code d'inversion *meteosat* a été compilé et testé avec succès, et peut être exécuté sans limites pour les environnements suivants :

- Systeme AIX v4.3 (*IBM*®), Compilateurs xlf90/xlc,
- Systeme AIX v5.2 (*IBM*®), Compilateurs IBM xlf90/xlc,
- Systeme Linux RedHat 8.x,9.x (*PC/Intel*™), Compilateurs Intel 7.1 ifc/icc,
- Systeme Linux Fedora Core 2 (*PC/Intel*™), Compilateurs Intel 8.1 ifort/icc,
- Systeme Linux Fedora Core 3 (*PC/Intel*™), Compilateurs GNU g95/gcc,
- Systeme Linux Fedora Core 4 (*PC/Intel*™), Compilateurs GNU icc/gcc,
- Systeme Linux YellowDogLinux 3.x,4.x (*Macintosh/Apple*®), Compilateurs NAGware f95/gcc,
- Systeme Darwin 2.x (*Macintosh/Apple*®), Compilateurs NAGware f95/gcc,
- Systeme OSF1 (*DEC*®), Compilateurs f95/cc

Le code prend en compte la convention d'arrangements d'octets en mémoire "Big Endian"/"Little Endian", la portabilité de ce point de vue est complète².

¹De plus, il est possible de trouver des versions sous licence GNU de ces compilateurs quasiment pour tous les systèmes d'exploitations, à la différence de logiciels payants comme IDL qui apportent indéniablement une souplesse dans l'écriture du code source, mais deviennent inopérants dans le cadre de traitements scientifiques à grande échelle

²Il est de plus possible d'exporter des résultats de traitement avec la convention souhaitée, ceci pour rendre directement lisible ces résultats sur une machine ayant une convention différente.



2.1.3 Standard et conformité

L'écriture du code source en C est conforme à la norme standard ANSI/ISO ISO/IEC 9899 :1999 (<http://www.sics.se/~pd/ISO-C-FDIS.1999-04.pdf>). Des tests stricts de compilations ont été effectués sur plusieurs architectures. L'écriture du code source en Fortran est conforme à la norme ISO/Fortran 95.

2.1.4 Règles syntaxiques

- Par souci de portabilité, les accents et autres caractères non ASCII ne sont pas utilisés dans les fichiers sources du code.
- Dans le code en Fortran, les variables locales ou globales sont en minuscules, les mots clés du langage en majuscules. De plus, les variables ayant l'attribut `PARAMETER` sont également en majuscules.
- L'indentation choisie est généralement de deux caractères.
- Plutôt que de privilégier la compacité du code, on choisit de l'aérer, en laissant fréquemment des lignes vides.

2.2 Arborescence des répertoires

Vous pouvez copier l'arborescence du code *meteosat* où vous le souhaitez. Il faudra toutefois penser à faire pointer la variable d'environnement `$METrac` vers le répertoire approprié.

L'arborescence est composée de plusieurs répertoires. Le répertoire `6s` contient les sources du code du même nom.³, les fichiers de génération des différents exécutables se trouvent dans le répertoire `gen`, les fichiers de lancement des simulations servant à générer les tables qui sont utiles au code *meteosat* dans le répertoire `inp`, la documentation du code dans le répertoire `doc`.

Le répertoire `com` contient les fichiers sources en C communs aux différents programmes rassemblés dans le répertoire `src/c`, les fichiers de génération des archives associés à ces fichiers se trouvent dans le répertoire `gen`, les fichiers objets sont stockés dans le répertoire `obj`, et finalement les fichiers archives dans le répertoire `lib`.

Le répertoire `doc` contient la totalité des sources en $\text{\LaTeX}$ 2_ε de ce présent manuel ainsi que les fichiers permettant de reproduire celle-ci dans le répertoire *man*. Celui-ci contient plusieurs répertoires, `dvi`, `html`, `pdf`, `ps`, contenant respectivement les versions au format *dvi*, *html*, *pdf* et *PostScript* de cette documentation, et le répertoire `tex` rassemblant les sources en $\text{\LaTeX}$ 2_ε de ce manuel. Les documentations fournies par l'agence *Eumetsat*, concernant le format des données *Meteosat* sont également présents dans le répertoire `anc`. Enfin, des documentations annexes sont présentes dans le répertoire `misc`.

Le répertoire `met` contient les fichiers sources du code *meteosat*, qui permet l'inversion des paramètres optique des aérosols au dessus des océans à partir des canaux visibles des satellites *Meteosat*. L'essentiel des fonctions sont écrites en *Fortran*, et se trouvent dans le répertoire

³La version du code `6s` utilisée est 4.2. Ces sources ont été quelque peu modifiées pour permettre de simuler la réflectance au sommet de l'atmosphère dans les bandes spectrales des instruments *Meteosat-5*, *Meteosat-7* et *MSG-1*. Les modifications apportées à ce code permettent également de simuler en une exécution du code `6s` la réflectance TOA dans tout le demi-espace montant. Une autre modification est celle portant sur le nombre d'angles de Gauss qui a été volontairement augmenté. Nous renvoyons le lecteur à l'annexe C de ce document pour plus de précisions.

`met/src/f`, les fonctions en *C* se trouvent quant à elles dans le répertoire `met/src/c`, les programmes principaux en *Fortran* sont rassemblés dans le répertoire `met/src`. Le répertoire `gen` rassemble les différents scripts en shell (`bash`) qui servent à la compilation du code. L'espace de génération est constitué par le répertoire `obj` qui contient les fichiers objets (`*.o`), le répertoire `lib` qui contient les bibliothèques d'objets utilisées lors de l'édition des liens, le répertoire `mod` qui contient les différents fichiers modules *Fortran*. Le répertoire `inc` contient le fichier recensant les messages d'information ou d'erreur que renvoie le code *meteosat* lors de son exécution. Le répertoire `inp` rassemble plusieurs fichiers de lancement du code.

Le répertoire `mfg` contient les fichiers sources permettant le décodage des fichiers au format OpenMTP.

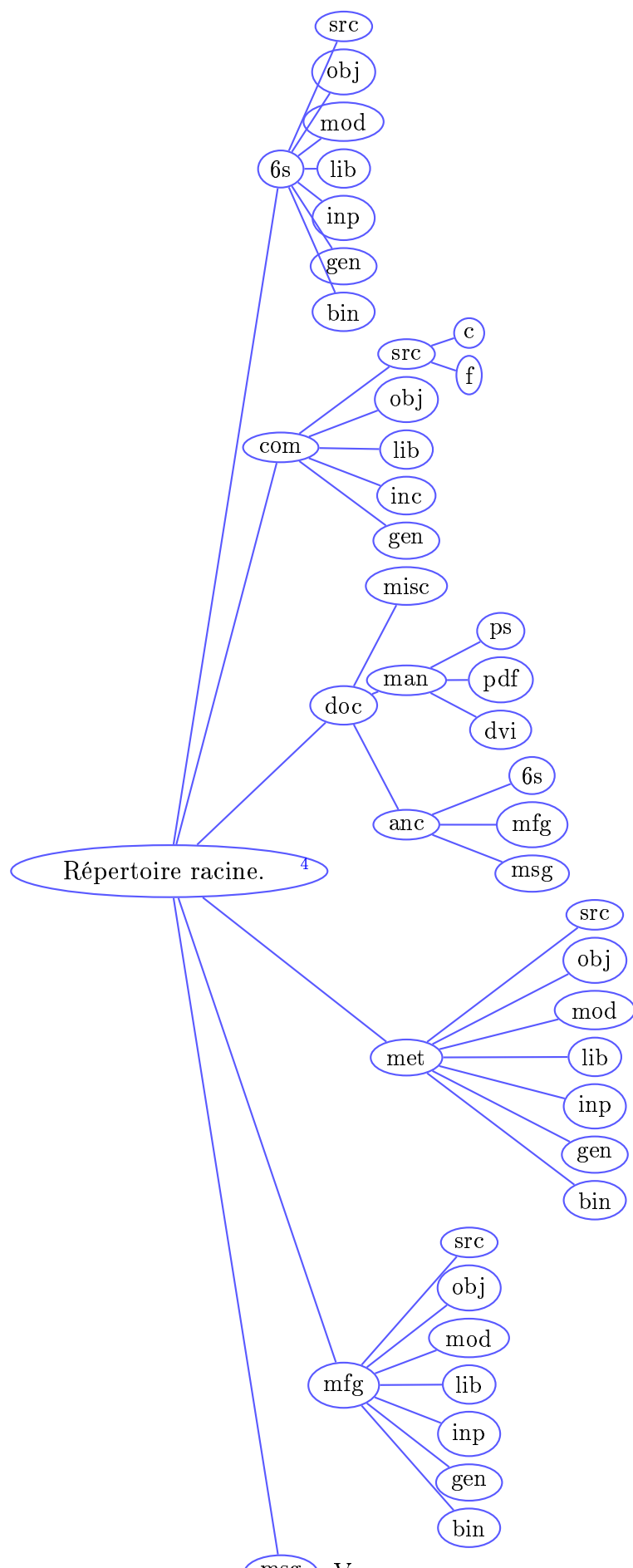
Le répertoire `msg` contient les fichiers sources permettant le décodage des fichiers au format *natif* ou *BSQ*.

La figure 2.1 de la page 15 représente sous forme d'un arbre, l'arborescence complète qui vient d'être décrite ici sommairement.



2.3 Génération des exécutables

D'une façon générale, la compilation des différents exécutables se fait au moyen de scripts interprétés par le shell (`/bin/bash`), et se trouvent dans les répertoires `gen`. Les scripts commençant par le préfixe `test` désactivent toute optimisation de compilation, et incluent des options de vérification intensive de la syntaxe du code source, de vérification étendue d'erreur à l'exécution, des informations pour le débogage du code, etc... ; les scripts commençant par le préfixe `comp` ont pour but d'optimiser au mieux le code compilé pour l'architecture machine concernée. Ces scripts sont complémentaires : l'utilisateur du code qui ne souhaite pas modifier celui-ci préférera optimiser un code *a priori* stable ; le développeur devrait plutôt préférer des compilation et exécution successives du code à l'aide des scripts préfixés `test`, puis `comp`.





Chapitre 3

Mode opératoire

3.1 Définition des variables d'environnement

La seule et unique variable d'environnement qui soit nécessaire à l'utilisation du code est `$METrac`.

En pratique, il est conseillé et recommandé de définir la variable `$METrac` dans l'un des fichiers `~/.bashrc`, `~/.kshrc`, `~/.cshrc`, ou `~/.tcshrc`, suivant que le shell utilisé est le `bash`, `ksh`, `csh`, ou `tcsh`. Si vous utilisez le `bash` ou le `ksh`, il suffit d'ajouter dans le fichier correspondant la ligne suivante :

```
export METrac=/mon/chemin/a/moi
```

De même, pour le `csh` ou `tcsh` :

```
setenv METrac /mon/chemin/a/moi
```

Puis, au prompt, exécuter la commande

```
source ~/.bashrc(ou encore ~/.cshrc, ~/.tcshrc, ou ~/.kshrc)
```

pour lire et exécuter la modification des commandes contenues dans le fichier dans le shell en cours.

3.2 Paramètres de simulation - Arguments du programme

Une grande liberté est donnée à l'utilisateur pour spécifier les paramètres du code *meteosat*. L'utilisateur peut lancer le code directement en passant comme argument les paramètres au fichier exécutable¹, ou bien utiliser un fichier de lancement. Un exemple de shell de lancement du code *meteosat* est proposé page 77 dans l'annexe A.

Le décodage des arguments, ainsi que l'intercohérence de ceux-ci fait appel à une gestion rigoureuse des erreurs, et nous renvoyons l'utilisateur au paragraphe 4.8 pour plus de détails.

Nous décrivons dans le tableau 3.1 l'ensemble de ces paramètres. La première colonne correspond à l'indice du paramètre, la seconde est une description précise de celui-ci, la troisième est le type et sous-type de la variable *Fortran* qui contient l'argument une fois celui-ci décodé, la dernière le format *Fortran* de décodage de l'argument.

¹*i.e.* mode ligne de commande



No	Nom	Définition	TYPE (SOUS-TYPE)	FORMAT
1	isat	Indice du satellite. Valeur correspondant au satellite dont on veut traiter un fichier : -1 : Satellite par défaut (lecture de l'en-tête du fichier) 5 : Meteosat-5 7 : Meteosat-7 8 : Meteosat-8 (MSG-1) 9 : Meteosat-9 (MSG-2) [Non encore testé]	INTEGER(int32)	I2.2
2	iraw	Indice du format des données Valeur correspondant au format du fichier en entrée du code : -1 : Format de données déterminé par le programme 0 : Format de données : - OpenMTP pour les données MFG (isat <= 7) - NATIF ou BSQ pour les données MSG (isat >= 8) 1 : Données contenues dans un (ou plusieurs) fichier(s) image(s) associés à un (ou plusieurs) fichier(s) en-tête.	INTEGER(int32)	I2.2
3	racine	Répertoire contenant les fichiers utiles au traitement des images Meteosat. Si racine vaut -1, les arguments qui suivent contiennent les noms des différents répertoires utiles ; Sinon, racine contient le répertoire (absolu) des différents sous-répertoire utiles.	CHARACTER	A
4	rep_raw	Répertoire (absolu) contenant les fichiers bruts (format OpenMTP, NATIVE, BSQ, HRIT/LRIT).	CHARACTER	A
5	rep_img	Répertoire (absolu) contenant les fichiers images brutes (niveau 1 ou 1.5).	CHARACTER	A
6	rep_hdr	Répertoire (absolu) contenant les fichiers en-tête (binaire) associés aux images brutes.	CHARACTER	A
7	rep_log	Répertoire (absolu) contenant le fichier log du traitement.	CHARACTER	A
8	rep_tmc	Répertoire (absolu) contenant les fichiers masque Terre/Mer/Cote.	CHARACTER	A

9	rep_mie	Répertoire (absolu) contenant les fichiers des paramètres spectrales des modèles d'aérosols obtenus par des simulations du code <i>6S</i> .	CHARACTER	A
10	rep_lst	Répertoire (absolu) contenant les fichiers répertoriant la liste des fichiers à traiter.	CHARACTER	A
11	rep_lut	Répertoire (absolu) contenant les fichiers des tables de couleur.	CHARACTER	A
12	rep_tab	Répertoire (absolu) contenant les fichiers précalculés.	CHARACTER	A
13	rep_msq	Répertoire (absolu) contenant les fichiers de masque.	CHARACTER	A
14	rep_tau	Répertoire (absolu) contenant les fichiers de produits aérosols.	CHARACTER	A
15	spare15	Argument non utilisé.	CHARACTER	A
16	spare16	Argument non utilisé.	CHARACTER	A
17	rep_gif	Répertoire (absolu) des fichiers de sortie au format GIF.	CHARACTER	A
18	rep_tif	Répertoire (absolu) des fichiers de sortie au format TIFF.	CHARACTER	A
19	rep_jpg	Répertoire (absolu) des fichiers de sortie au format JPEG.	CHARACTER	A
20	rep_hdf	Répertoire (absolu) des fichiers de sortie au format HDF.	CHARACTER	A
21	rep_rho	Répertoire (absolu) des fichiers de sortie de composition colorée RGB.	CHARACTER	A



22	fichtmc	Nom du fichier de masque utilisé pour discriminer les pixels de type terre, mer, côte, et espace. Le fichier a une taille de 5000x5000 pixels lorsque les données à traiter sont de type MFG , de 3712x3712 pixels lorsqu'elles sont de type MSG . Les valeurs possibles du masque TMC par convention codées sur 1 octet (8 bits) sont les suivantes : - 10 : Mer - 20 : Côte - 30 : Terre - 0 : Espace Si fichtmc = "-1", dans ce cas on considère le fichier de masque par défaut ("MET-07.tmc" pour Meteosat-07, "MET-08.tmc" pour Meteosat-08), etc... , se trouvant dans le répertoire pointé par l'argument rep_tmc.	CHARACTER	A
23	fichlog	Nom du fichier de dialogue ou sont dirigés les commentaires du programme.	CHARACTER	A
24	fichlst	Nom du fichier contenant la liste des fichiers Meteosat à traiter, ou encore nom d'un fichier unique à traiter. Dans ce dernier cas, l'utilisateur peut spécifier le nom absolu du fichier, ou bien le nom simple du fichier, le répertoire par défaut le contenant est supposé pointé par l'argument du programme rep_img. Si fichlst vaut -1, dans ce cas on considère les fichiers images dont la plage horaire est contenue dans plage, dont la date est comprise entre annee_deb/mois_deb/jour_deb et entre annee_fin/mois_fin/jour_fin.	CHARACTER	A
25	fichlut_rho	Nom du fichier contenant la table de couleurs utilisée pour la mise en forme des fichiers de la réflectance mesurée au sommet de l'atmosphère. Ce fichier est supposé se trouver dans le répertoire pointé par l'argument rep_lut Si fichlut_rho vaut -1, dans ce cas on considère la table de couleur par défaut, GREY_bg=white.lut.	CHARACTER	A
26	fichlut_tau	Nom du fichier contenant la table de couleurs utilisée pour la mise en forme des fichiers de l'épaisseur optique aérosol inversée. Ce fichier est supposé se trouver dans le répertoire pointé par l'argument rep_lut Si fichlut_tau vaut -1, dans ce cas on considère la table de couleur par défaut, METEOSAT_bg=white.lut.	CHARACTER	A

27	fichlut_ang	Nom du fichier contenant la table de couleurs utilisée pour la mise en forme des fichiers du coefficient d'Angström inversé. Ce fichier est supposé se trouver dans le répertoire pointé par l'argument rep_lut Si fichlut_ang vaut -1, dans ce cas on considère la table de couleur par défaut, METEOSAT_bg=white.lut .	CHARACTER	A
28	date_debut	Date de début de traitement (ssi fchlst=-1) Le format de décodage est AAAA-MM-JJ, AAAA est l'année, MM le mois, JJ le jour du mois.	CHARACTER	A
29	date_fin	Date de fin de traitement (ssi fchlst=-1) Le format de décodage est AAAA-MM-JJ, AAAA est l'année, MM le mois, JJ le jour du mois.	CHARACTER	A
30	plage	Plage(s) horaire(s) à traiter par jour (ssi fchlst=-1). Spécifier un nombre, ou une liste de nombres. Par exemple : 22-24, sera interprété comme 22, 23, et 24 ; 20,24 sera interprété comme 20 et 24 .	CHARACTER	A
31	iatm	Indice correspondant à la répartition et à la quantité des gaz absorbants (H ₂ O,O ₃) du modèle d'atmosphère retenu pour calculer les transmissions gazeuses sur le double trajet montant et descendant. Ces modèles d'atmosphère pour lesquels sont précalculées les transmissions sont au nombre de 6. Le cas d'étude sans absorption est également envisageable (iatm=0). Si la valeur est négative, le modèle d'atmosphère retenu est <i>US62</i> . La numération utilisée ici reprend celle du code <i>6S</i> . - <0 : Mer - 0 : pas d'absorption gazeuses - 1 : modèle "Tropical" (TRO), $u_{H_2O} = 4.12g/cm^2, u_{O_3} = .247cm - atm,$ - 2 : modèle "Mid Latitude Summer" (MLS), $u_{H_2O} = 2.93g/cm^2, u_{O_3} = .319cm - atm,$ - 3 : modèle "Mid Latitude Winter" (MLW), $u_{H_2O} = .853g/cm^2, u_{O_3} = .395cm - atm,$ - 4 : modèle "Sub Artic Summer" (SAS), $u_{H_2O} = 2.10g/cm^2, u_{O_3} = .480cm - atm,$ - 5 : modèle "Sub Artic Winter" (SAW), $u_{H_2O} = .419g/cm^2, u_{O_3} = .480cm - atm,$ - 6 : modèle "US Standard 1962" (US62), $u_{H_2O} = 1.42g/cm^2, u_{O_3} = .344cm - atm.$	INTEGER(int32)	I1



32	lambda	Valeur de la longueur d'onde [μm] pour laquelle on souhaite inverser la valeur de l'épaisseur optique aérosol. Par défaut, <i>i.e.</i> si cette est négative, l'épaisseur optique aérosol est inversée à $.55\mu\text{m}$.	CHARACTER	A
33	ietal	Indice correspondant à la provenance des coefficients d'étalonnage utiliser. Pour les instruments de type <i>MFG</i> , les informations concernant l'étalonnage radiométrique des mesures, incluses dans l'en-tête des fichiers au format <i>OpenMTP</i> ne sont pas renseignées, elles le sont pour les instruments de type <i>MSG</i> dans l'en-tête des fichiers de niveau 1.5 (tous formats.) Si <i>ietal</i> vaut 0, les informations recueillies dans l'en-tête sont utilisées pour étalonner les comptes numériques des images. Si <i>ietal</i> vaut 1, on procède à l'étalonnage des mesures au moyen de coefficients utilisateurs, avec dérive temporelle possible (<i>c.f.</i> fonction <i>f_etalonner</i>).	INTEGER(int32)	I1
34	swap_coord	Drapeau du retournement de l'image. On veut que l'image soit orientée de telle manière que les coordonnées les plus petites correspondent au coin NW, les coordonnées les plus grandes correspondent au coin SE. La valeur (1) retourne ainsi l'image d'origine, la valeur (0) laisse l'orientation de l'image identique à celle d'origine. Une connaissance de l'orientation de l'image d'origine est "a priori" nécessaire.	INTEGER(int32)	I1
35	ifltnua	Indice du canal spectral sur lequel est basé le filtre nuage. Pour les versions antérieures à <i>Meteosat-8</i> (MSG-1), un seul canal (VIS) est disponible. Par défaut (-1), le second canal ($0.810\mu\text{m}$) est choisi pour <i>Meteosat-8</i> et ses successeurs, le premier pour les versions antérieures.	INTEGER(int32)	I2
36	typ_coord_img	Type de coordonnées de l'image à lire. Les coordonnées NW et SE de l'image peuvent être entrées en valeurs de latitude, longitude (0), ou encore en coordonnées image(1); Par défaut (-1), l'image globale est traitée.	INTEGER(int32)	I2
37	xcoord_NW	Coordonnées ligne/latitude du pixel NW de l'image. Le choix est laissé à l'utilisateur d'entrer les coordonnées de l'image en valeurs de longitude (valeur réelle), ou indice de ligne (entière).	INTEGER(int32) ²	I4 ³

²ou encore REAL(real32) si l'argument *type_coord_img* vaut 0³ou encore F9.5 si l'argument *type_coord_img* vaut 0

38	ycoord_NW	Coordonnées colonne/longitude du pixel NW de l'image. Le choix est laissé à l'utilisateur d'entrer les coordonnées de l'image en valeurs de latitude (valeur réelle), ou indice de ligne (entière).	INTEGER(int32) ¹	I4 ²
39	xcoord_SE	Coordonnées ligne/latitude du pixel SE de l'image. Le choix est laissé à l'utilisateur d'entrer les coordonnées de l'image en valeurs de longitude (valeur réelle), ou indice de colonne (entière).	INTEGER(int32) ¹	I4 ²
40	ycoord_SE	Coordonnées colonne/longitude du pixel SE de l'image. Le choix est laissé à l'utilisateur d'entrer les coordonnées de l'image en valeurs de latitude (valeur réelle), ou indice de ligne (entière).	INTEGER(int32) ¹	I4 ²
41	nlig_bnd	Nombres de lignes du bandeau permettant de conserver les informations sur le voisinage d'un pixel lors du filtre nuageux. Ce nombre entier doit être impair. On choisit de limiter sa valeur maximale à 9. Par défaut (-1), le bandeau est constitué de 3 lignes successives.	INTEGER(int32)	I1



42	iaer	Type de l'inversion que l'on souhaite obtenir. Si l'on souhaite forcer une inversion avec un modèle d'aérosol unique, on spécifie ici son indice. Pour les images de type MFG , qui ne dispose que d'une seule bande spectrale exploitable dans le visible, seule une inversion avec forçage du modèle est possible. La liste des modèles par ordre croissant d'indice sont les suivants (indice variant de 0 à 15) : B00,099,M99,C99,M90,C90,M70,M50,C70,C50,T99,T90,T50,W00, W01,W02 ⁴ . Pour les images de type MSG , une inversion avec forçage du modèle est possible, comme pour les images de type MFG . Une inversion multimodale est demandée lorsque l'utilisateur spécifie une liste d'indices de modèles. Par exemple, 2-4, sera interprété comme 2, 3, et 4; 2,4,6-10 sera interprété comme 2 4, 6, 7, 8, 9 et 10. La valeur -1 signifie que l'utilisateur souhaite faire une inversion multimodale, i.e. obtention d'une épaisseur optique et d'un coefficient d'Angström associé, en utilisant tous les modèles disponibles (hormis le modèle B00).	INTEGER(int32)	I1
43	ilmb1	Indice du premier canal spectral sur lequel se fait l'inversion de l'épaisseur optique aérosol, dans le cas de figure ou le modèle d'aérosol n'est pas forcé (i.e. imodele=-1). Pour les instruments de type MFG , l'unique canal disponible est utilisé. Pour les instruments de type MSG , deux des trois premiers canaux spectraux (0.635, 0.810, ou 1.640 μm) au choix sont possibles.	INTEGER(int32)	I2
44	ilmb2	Indice du deuxième canal spectral sur lequel se fait l'inversion de l'épaisseur optique aérosol, dans le cas de figure ou le modèle d'aérosol n'est pas forcé (i.e. imodele=-1). Pour les instruments de type MFG , l'unique canal disponible est utilisé. Pour les instruments de type MSG , deux des trois premiers canaux spectraux (0.635, 0.810, ou 1.640 μm) au choix sont possibles.	INTEGER(int32)	I2
45	typ_coord_ext	Type des coordonnées du point d'extraction. Les coordonnées de ce point peuvent être entrées en coordonnées géographiques (0), ou bien directement en coordonnées image (1).	INTEGER(int32)	I1

⁴Le modèle 'B00' correspondant au modèle intégré par défaut dans 6S, "Shettle model for background desert aerosol", est un modèle de type aérosol désertique, ayant une distribution en taille constituée par 3 log-normales, qui peut servir à inverser les données de type **MFG**.

46	<code>xcoord_ext</code>	Coordonnée suivant l'axe x d'un pixel pour lequel on souhaite extraire les paramètres aérosol inversés. Cette coordonnée est soit en valeur de longitude, ou bien en indice de colonnes.	REAL(real32) ⁵ .	F9.5 ⁶
47	<code>ycoord_ext</code>	Coordonnée suivant l'axe y d'un pixel pour lequel on souhaite extraire les paramètres aérosol inversés. Cette coordonnée est soit en valeur de latitude, ou bien en indice de lignes.	REAL(real32) ³	F9.5 ⁴
48	<code>dialogue</code>	Drapeau pour l'écriture de commentaires du programme dans le fichier de dialogue. Plus la valeur du drapeau est élevée, plus le flot de commentaires redirigés vers le fichier <code>fichlog</code> est important.	INTEGER(int32)	I1
49	<code>ifit</code>	Calcul des coefficients du fit polynomial d'ordre 3, reliant la réflectance totale au sommet de l'atmosphère en fonction de l'épaisseur optique aérosol, $\rho_{tot}^{TOA}(\lambda) = f(\tau_a(\lambda))$. 0 : lecture de la table précalculée des coefficients du fit polynomial d'ordre 3 1 : calcul des coefficients à partir de la table précalculée reliant $\rho_{tot}^{TOA}(\lambda)$ et $\tau_a(\lambda)$ -1 : par défaut, calcul des coefficients à partir de la table précalculée.	INTEGER(int32)	I1
50	<code>ths_lim</code>	Angle zénithal solaire (degrés) maximal au-delà duquel on ne procède plus à l'inversion des aérosols.	REAL(real32)	F9.5
51	<code>thv_lim</code>	Angle zénithal de visée (degrés) maximal au-delà duquel on ne procède plus à l'inversion des aérosols.	REAL(real32)	F9.5
52	<code>seuil_rho_TOA</code>	Seuil sur la réflectance au sommet de l'atmosphère au-delà de laquelle on considère qu'un pixel marin est nuageux	REAL(real32)	F9.5
53	<code>seuil_etrho_TOA</code>	Seuil sur l'écart-type de la réflectance au sommet de l'atmosphère d'un super-pixel nxn, au-delà de laquelle on considère qu'un pixel marin est nuageux ⁷ .	REAL(real32)	F9.5
54	<code>vent_lim</code>	Valeur limite de la vitesse de vent utilisée pour calculer la réflectance analytique du glitter par la formule de [Cox et Munk, 1954].	REAL(real32)	F9.5

⁵ou encore INTEGER(int32) si l'argument `typ_coord_ext` vaut 1

⁶ou encore I4 si l'argument `type_coord_ext` vaut 1

⁷Ceci est basé sur l'hypothèse de plus grande hétérogénéité horizontale des nuages comparée aux panaches d'aérosols.



55	rho_glt_lim	Valeur de la réflectance seuil pour le glitter. Lorsque la réflectance du glitter calculée par la formule de Cox et Munk dépasse cette valeur seuil, on arrête le traitement du pixel considéré.	REAL(real32)	F9.5
56	tau550_min	Valeur de l'épaisseur optique aérosol minimale inversée à 550 nm en deçà de laquelle le coefficient d'Angström n'est pas inversé.	REAL(real32)	F9.5
57	angstrom_bg	Valeur du coefficient d'Angström imposée lorsque l'épaisseur optique aérosol inversée à 550 nm est inférieure à la valeur limite tau550_min.	REAL(real32)	F9.5
58	rho_mes_minmax	Valeurs minimale et maximale de la réflectance mesurée au sommet de l'atmosphère à coder sur l'échelle 8 bits. ⁸	REAL(real32)	F9.5
59	ang_day_minmax	Valeurs minimale et maximale du coefficient d'Angström inversé sur la journée à coder sur l'échelle 8 bits. ⁸	REAL(real32)	F9.5
60	tau550_day_minmax	Valeurs minimale et maximale de l'épaisseur optique aérosol inversée sur la journée à coder sur l'échelle 8 bits. ⁸	REAL(real32)	F9.5
61	ang_mth_minmax	Valeurs minimale et maximale du coefficient d'Angström inversé sur le mois à coder sur l'échelle 8 bits. ⁸	REAL(real32)	F9.5
62	tau550_mth_minmax	Valeurs minimale et maximale de l'épaisseur optique aérosol inversée sur le mois à coder sur l'échelle 8 bits. ⁸	REAL(real32)	F9.5
63	iordre	Drapeau permettant de spécifier la manière dont les octets seront rangés en mémoire, ceci concernant uniquement les fichiers binaires créés lors de l'exécution du programme. 0 : les octets sont rangés dans la représentation BIG ENDIAN ⁹ . 1 : les octets sont rangés dans la représentation LITTLE ENDIAN ⁹ . -1 : par défaut, les octets sont rangés en accord avec la représentation de la machine.	INTEGER(int32)	I1

⁸Les deux valeurs doivent être séparées par le symbole " : "

⁹ La représentation BIG ENDIAN d'un arrangement d'un groupe d'octets en mémoire signifie que l'octet le plus significatif est localisé à l'adresse la plus petite, l'octet le moins significatif est localisé à l'adresse la plus grande. Dans la représentation LITTLE ENDIAN, c'est la convention inverse qui s'applique, l'octet le plus significatif étant localisé à l'adresse la plus grande. Nous rappelons à l'utilisateur que les arrangements d'octets sont rangés par convention LITTLE ENDIAN pour les ordinateurs de type INTEL/PC, DEC, VAX ; la convention BIG ENDIAN est adoptée pour les machines SGI, IBM, Macintosh ;

64	<code>icomp_in</code>	Drapeau pour le choix de la compression des fichiers d'entrée	INTEGER(int32)	I1
65	<code>icomp_out</code>	Drapeau pour le choix de la compression des fichiers de sortie	INTEGER(int32)	I1
66	<code>rescale_ref</code>	Facteur de redimensionnement de l'image de la réflectance TOA. Ce coefficient doit être compris entre 0 et 1. Si la valeur est négative, aucun redimensionnement n'est appliqué.	REAL(real32)	F9.5
67	<code>rescale_tau</code>	Facteur de redimensionnement de l'image de l'épaisseur optique aérosol. Ce coefficient doit être compris entre 0 et 1. Si la valeur est négative, aucun redimensionnement n'est appliqué.	REAL(real32)	F9.5

TAB. 3.1: Description des arguments du programme

3.3 Constantes de simulation

Nous avons également besoin de définir des paramètres intervenant dans le calcul du code, associés aux paramètres de simulation que nous venons de voir au paragraphe 3.2. Ceux-ci se distinguent d'une part en un ensemble de constantes dimensionnant la taille des tableaux utilisés dans les calculs et d'autre part en des valeurs seuils nécessaires pour les tests d'arrêt du code. Ces paramètres sont introduits dans le code en *Fortran* à la compilation par l'attribution de valeurs aux constantes, incluses dans des modules, par le biais du fichier `m_meteosat.f`, dans lequel sont définis une partie importante des modules du code¹⁰.

Les différentes constantes de simulation utilisées sont décrites dans le tableau 3.2, page 30. Les constantes qui sont les plus susceptibles d'être modifiées se trouvent dans le module `m_constants`. Une modification de la valeur des autres constantes demande une bonne connaissance du fonctionnement du code.

NOM	Définition	TYPE (SOUS-TYPE)	Valeur
MODULE <code>m_parametres_chaines</code>			
<code>LENREP</code>	Longueur des chaînes contenant les noms de répertoires.	INTEGER(int32)	150
<code>LENFIC1</code>	Longueur des chaînes contenant les noms de fichiers ¹¹ .	INTEGER(int32)	150

¹⁰Le fichier se trouve dans le répertoire `$METrac/src/f`

¹¹sans arborescence.



LENFIC2	Longueur des chaînes contenant les noms de fichiers ¹²	INTEGER(int32)	201
LENPARAM	Longueur de la chaîne servant à decoder les arguments du programme.	INTEGER(int32)	201
MODULE m constantes			
IND_SAT_MIN	Indice minimal du satellite utilisable.	INTEGER(int32)	1
IND_SAT_MAX	Indice maximal du satellite utilisable.	INTEGER(int32)	8
NB_LIG_BND_MAX	Nombre de lignes maximales du bandeau.	INTEGER(int32)	7
NB_PLG_MAX	Tableau du nombre de plages horaires maximales dans une journée ¹³ .	INTEGER(int32)	48,96
NB_LMB_MAX	Nombre de canaux spectraux ¹³ .	INTEGER(int32)	1,12
NB_LIG_MIN	Nombre de lignes minimales d'une image.	INTEGER(int32)	1
NB_COL_MIN	Nombre de colonnes minimales d'une image.	INTEGER(int32)	1
NB_LIG_MAX	Nombre de lignes maximales d'une image ¹³ .	INTEGER(int32)	5000,3712
NB_COL_MAX	Nombre de colonnes maximales d'une image de type ¹³ .	INTEGER(int32)	5000,3712
NB_LIG_MSG_HRV_MAX	Nombre de lignes maximales d'une image de type MSG , canal HRV.	INTEGER(int32)	11136
NB_COL_MSG_HRV_MAX	Nombre de colonnes maximales d'une image de type MSG , canal HRV.	INTEGER(int32)	11136

¹²arborescence incluse.¹³Cette constante est définie en *Fortran* comme un tableau dont la taille dépend du nombre de type d'instruments accepté par le code. Actuellement, les instruments de type **MFG** (0), et **MSG** (1) sont possibles. Par la suite, les instruments de type **MTG** (Meteosat Third Generation) pourront également l'être.

ANNEE_MIN	Année calendaire minimale.	INTEGER(int32)	1990
ANNEE_MAX	Année calendaire maximale.	INTEGER(int32)	2004
MOIS_MIN	Nombre de mois minimal.	INTEGER(int32)	1
MOIS_MAX	Nombre de mois maximal.	INTEGER(int32)	12
JOUR_MIN	Nombre de jours du mois minimal.	INTEGER(int32)	1
JOUR_MAX	Nombre de jours du mois maximal.	INTEGER(int32)	31
LATITUDE_MIN	Valeur minimale que peut prendre la latitude (degrés).	REAL(real32)	-90.
LATITUDE_MAX	Valeur maximale que peut prendre la latitude (degrés).	REAL(real32)	+90.
LONGITUDE_MIN	Valeur minimale que peut prendre la longitude (degrés).	REAL(real32)	-180.
LONGITUDE_MAX	Valeur maximale que peut prendre la longitude (degrés).	REAL(real32)	+180.
THS_MIN	Valeur minimale que peut prendre l'angle zénithal solaire (degrés).	REAL(real32)	0.
THS_MAX	Valeur maximale que peut prendre l'angle zénithal solaire (degrés).	REAL(real32)	90.
THV_MIN	Valeur minimale que peut prendre l'angle zénithal de visée (degrés).	REAL(real32)	0.
THV_MAX	Valeur maximale que peut prendre l'angle zénithal de visée (degrés).	REAL(real32)	90.



REFLECTANCE_TOA_MIN	Valeur minimale que peut prendre la réflectance totale au sommet de l'atmosphère.	REAL(real32)	0.
REFLECTANCE_TOA_MAX	Valeur maximale que peut prendre la réflectance totale au sommet de l'atmosphère.	REAL(real32)	1.

TAB. 3.2: Description des constantes du programme

3.3.1 Constantes du module M_PARAMETRES_CHAINES

3.3.2 Constantes du module M_CONSTANTES

3.4 Exemple de shell de lancement du code *meteosat*

Le code *meteosat* peut être directement lancé au prompt du shell au moyen d'une ligne de commande, mais le nombre de paramètres étant important, il est préférable d'utiliser un shell qui rassemble tous les paramètres du programme, puis les passe en arguments à l'exécutable concerné, et enfin lance cet exécutable. Pour un tel exemple de shell, l'utilisateur est prié de se reporter à l'annexe A.

3.5 Temps d'exécution.

À titre illustratif, le temps moyen d'exécution du code *meteosat* nécessaire pour traiter une image de type *MSG* pleine résolution, sur une machine possédant un processeur *PentiumIV*TM à 2,66 GHz, est légèrement inférieur à 2 minutes¹⁴. Ce temps est similaire pour traiter une image de type *MFG* pleine résolution (5000 pixels de côtés), où toutefois seule l'épaisseur optique aérosol est inversée.

¹⁴Notez que lorsque l'argument `ifit` vaut 1, le pré-calcul et l'écriture sur disque des tables des coefficients des relations polynomiales reliant la réflectance totale atmosphérique TOA pour les différents canaux spectraux à l'épaisseur optique aérosol, prend 2 minutes sur cette même machine.

Chapitre 4

Descriptif du logiciel

4.1 Structure du code

4.2 Format des fichiers de données en entrée

Le code *meteosat* attend que les données d'entrée, de type *MFG* ou *MSG*, soient sous un format particulier si l'argument *iraw* vaut 1. Dans ce cas, pour chaque acquisition, il doit exister dans le répertoire pointé par l'argument *rep_hdr* un fichier contenant l'en-tête binaire, et dans le répertoire pointé par l'argument *rep_img* autant de fichiers contenant chacun une image binaire brute de l'acquisition. Dans le cas où *iraw* vaut 0, le code s'attend à trouver dans le répertoire pointé par l'argument *rep_raw*, un fichier unique au format *NATIVE* ou *BSQ* pour les acquisitions de type *MSG*, ou au format *OpenMTP* pour les acquisitions de type *MFG*.

4.3 Liste des programmes

Voici la liste des exécutable généré par les différents scripts :

- *mtp_main*, programme qui génère les fichiers en-tête et image, à partir d'un fichier brut au format OpenMTP,
- *msg_main*, programme qui génère les fichiers en-tête, image, et remorque à partir d'un fichier brut au format natif de niveau 1.5,
- *meteosat*, programme d'inversion des propriétés optiques des aérosols sur mer, pour les images Meteosat pleine résolution,
- *meteosat_day*, programme pour moyenner les inversions générées par le programme *meteosat* à l'échelle de la journée,
- *meteosat_mth*, programme pour moyenner les inversions générées par le programme *meteosat* à l'échelle du mois

4.4 Liste des fonctions du répertoire met

Le tableau 4.1 recense les fonctions du code et renvoie l'utilisateur aux différents paragraphes.

Fichier source	Module	Fonction	Section
----------------	--------	----------	---------



Fonctions en Fortran			
f_addslash.F	f_addslash	addslash	4.4.2
f_allouer.F	f_allouer	allouer	4.4.3
		allouer_synthese	4.4.4
		desallouer	4.4.5
		desallouer_synthese	4.4.6
f_arguments.F	f_arguments	byte_decoder_argument	4.4.7
		twobyte_decoder_argument	
		word_decoder_argument	
		char_decoder_argument	
		int8_decoder_argument	
		int16_decoder_argument	
		int32_decoder_argument	
		real32_decoder_argument	
		real64_decoder_argument	
		decoder_argument	4.4.8
		decoder_plage_horaire	4.4.9
		decoder_imodele	4.4.10
f_bandeau.F	f_bandeau	decoder_bornes	4.4.11
		decoder_date	4.4.12
		controler_arguments	4.4.13
		afficher_arguments	4.4.14
f_calculer_angstrom.F	f_calculer_angstrom	initialiser_arguments	4.4.15
		initialiser_ligne_bandeau	4.4.16
		initialiser_bandeau	4.4.17
		reporter_bandeau	4.4.18
f_calculer_coeff_interp_geom.F	f_calculer_coeff_interp_geom	charger_bandeau	4.4.19
		calculer_angstrom	4.4.20
		calculer_coeff_interp_geom	4.4.21
		calculer_fit	4.4.22
f_calculer_param_pixel.F	f_calculer_param_pixel	calculer_param_pixel	4.4.23
		charger_table_mol	4.4.24
		charger_table_frn	4.4.25
		charger_table_tot	4.4.26
f_charger_tables.F	f_charger_tables	charger_table_aer	4.4.27
		charger_table_fit	4.4.28
		charger_tables	
f_coder.F	f_coder	coder_tau550	4.4.29
		coder_angstrom	4.4.30
		coder_rho	4.4.31
f_compresser_fichiers.F	f_compresser_fichier	compresser_fichier	4.4.32
		decompresser_fichier	4.4.33
		compresser_fichiers_sortie	4.4.34
f_ecrire_ligne.F	f_ecrire_ligne	ecrire_ligne	4.4.35
f_etalonner.F	f_etalonner	etalonner	4.4.36
f_extraire_produit.F	f_extraire_produit	extraire_pixel_produit	4.4.37
		extraire_produit	4.4.38

<code>f_fermer_fichiers.F</code>	<code>f_fermer_fichiers</code>	<code>fermer_fichiers1</code> <code>fermer_fichiers2</code>	4.4.39 4.4.40
<code>f_filtrer_nuage.F</code>	<code>f_filtrer_nuage</code>	<code>filtrer_nuage</code>	4.4.41
<code>f_fitter.F</code>	<code>f_fitter</code>	<code>f_poly</code> <code>pythag</code> <code>svdfit</code>	4.4.42 4.4.43 4.4.44
<code>f_GaussLegendre.F</code>	<code>f_GaussLegendre</code>	<code>gausslegendre_real32</code> <code>gausslegendre_real64</code>	4.4.45
<code>f_glitter.F</code>	<code>f_glitter</code>	<code>glitter_real32</code> <code>glitter_real64</code> <code>tester_glitter</code>	4.4.46 4.4.47
<code>f_initialiser.F</code>	<code>f_initialiser</code>	<code>initialiser_variables</code> <code>initialiser_parametres_codage</code> <code>ouvrir_fichiers_ancillaire</code> <code>ouvrir_fichiers_tables</code> <code>initialiser</code> <code>initialiser_day</code> <code>initialiser_mth</code> <code>initialiser_ext</code>	4.4.48 4.4.49 4.4.51 4.4.52 4.4.53 4.4.54 4.4.55 4.4.56
<code>f_interpoler_aerosol.F</code>	<code>f_interpoler_aerosol</code>	<code>interpoler_aerosol</code>	4.4.57
<code>f_interpoler.F</code>	<code>f_interpoler</code>	<code>real32_interpoler1</code> <code>real64_interpoler1</code> <code>real32_interpoler2</code> <code>real64_interpoler2</code> <code>real32_interpoler3</code> <code>real64_interpoler3</code> <code>real32_interpoler4</code> <code>real64_interpoler4</code> <code>real32_interpoler5</code> <code>real64_interpoler5</code> <code>real32_interpoler6</code> <code>real64_interpoler6</code>	?? ?? ?? ?? ?? ??
<code>f_interpoler_moleculaire.F</code>	<code>f_interpoler_moleculaire</code>	<code>interpoler_moleculaire</code>	4.4.59
<code>f_inverser_aerosol.F</code>	<code>f_inverser_aerosol</code>	<code>inverser_aerosol</code>	4.4.60
<code>f_lire_fichier_entree.F</code>	<code>f_lire_fichier_entree</code>	<code>faire_nom_fichier_header</code> <code>lire_header</code> <code>faire_nom_fichier_sortie</code> <code>ouvrir_fichiers_sortie</code> <code>lire_fichier_entree</code>	4.4.61 4.4.62 4.4.63 4.4.64 4.4.65
<code>lire_fichier_mie_6S.F</code>	<code>f_lire_fichier_mie_6S</code>	<code>lire_fichier_mie_6S</code>	4.4.66
<code>lire_ligne.F</code>	<code>f_lire_ligne</code>	<code>faire_nom_fichier_image</code> <code>lire_ligne</code>	4.4.67 4.4.68
<code>f_liste_fichiers.F</code>	<code>f_liste_fichiers</code>	<code>faire_liste</code>	4.4.69



<code>f_message.F</code>	<code>f_message</code>	<code>ouvrir_fichier_log</code> <code>fermer_fichier_log</code> <code>ouvrir_fichier_message</code> <code>fermer_fichier_message</code> <code>ecrire_message</code>	4.4.70 4.4.71 4.4.72 4.4.73 4.4.74
<code>f_polynome.F</code>	<code>f_polynome</code>	<code>real32_polynome2</code> <code>real64_polynome2</code> <code>real32_polynome3</code> <code>real64_polynome3</code> <code>real32_polynome4</code> <code>real64_polynome4</code> <code>real32_polynome5</code> <code>real64_polynome5</code> <code>real32_polynome6</code> <code>real64_polynome6</code>	4.4.75 4.4.76 4.4.77 4.4.78 4.4.79
<code>f_produit.F</code>	<code>f_produit</code>	<code>faire_nom_produit</code> <code>ouvrir_fichier_produit</code> <code>lire_fichier_produit</code> <code>ecrire_fichier_produit</code> <code>fermer_fichier_produit</code>	4.4.80 4.4.81 4.4.82 4.4.83 4.4.84
Fonctions en C			
<code>read_header_mfg.c</code>		<code>read_header_mfg</code>	4.4.86
<code>read_header_msg.c</code>		<code>read_header_msg</code>	4.4.85

TAB. 4.1: Description sommaire des fonctions du répertoire `met`

4.4.1 Rappel sur les apports de fiabilité de programmation en Fortran 95

Pour améliorer la fiabilité générale du programme et s'assurer d'une parfaite homogénéité du contrôle des arguments, il faut insérer le même bloc interface dans toutes les unités de programme faisant référence à la procédure concernée.

C'est là le rôle du module et de l'instruction `USE` permettant l'accès à son contenu dans une unité de programme quelconque.

Un module est une unité de programme particulière introduite en *Fortran 90* pour encapsuler entre autres :

- des données et des définitions de types dérivés,
- des blocs interfaces,
- des procédures (après l'instruction `CONTAINS`),

Quel que soit le nombre d'accès (`USE`) au même module, les entités ainsi définies sont uniques¹. Enfin, ce module doit être compilé séparément avant de pouvoir être utilisé.

Dans le code, nous choisissons toujours d'utiliser un module pour réaliser une interface *explicite* qui contient la procédure, car c'est la solution la plus sûre.

4.4.2 Fonction `addslash`

Prototype

¹Ceci remplace avantageusement la notion de `COMMON` chère au *Fortran 77*.

```

FUNCTION addslash(dir)
IMPLICIT NONE
!----- Declarations des variables en argument -----
CHARACTER(LEN=*), INTENT(IN)          :: dir
CHARACTER(LEN=LEN(dir))                :: addslash

```

Description

4.4.3 Fonction allouer

Prototype

```

FUNCTION allouer( )
USE m_constants_math
USE m_nul
USE m_satellite
USE m_etalonnage
USE m_bande_spectrale
USE m_dimension_image
USE m_dimension_spectrale
USE m_dimension_bandeau
USE m_dimension_modele
USE m_dimension_angulaire
USE m_bandeau
USE m_image
USE m_codage
USE m_modele_aerosol
USE m_interpolation_geometrie
USE m_interpolation_modele
USE m_interpolation_spectrale
USE m_6s
USE m_angles_gauss
USE m_angles_diffusion
USE f_gausslegendre
IMPLICIT NONE
INTEGER(KIND=int16)                :: allouer

```

Description

4.4.4 Fonction allouer_synthese

Prototype

```

FUNCTION allouer_synthese( )
USE m_synthese
USE m_dimension_image
IMPLICIT NONE
INTEGER(KIND=int16)                :: allouer_synthese

```

Description

4.4.5 Fonction desallouer

Prototype

```

USE m_nul
USE m_dimension_bandeau
USE m_bandeau
USE m_image
USE m_codage
USE m_tables
USE m_interpolation_geometrie
USE m_interpolation_modele
USE m_interpolation_spectrale
USE m_liste

```



```
USE m_chemins
USE m_angles_gauss
USE m_angles_diffusion
IMPLICIT NONE
INTEGER(KIND=int16) :: desallouer
```

Description

4.4.6 Fonction desallouer_synthese

Prototype

```
FUNCTION desallouer_synthese( )
USE m_synthese
IMPLICIT NONE
INTEGER(KIND=int16) :: desallouer_synthese
```

Description

4.4.7 Fonction decoder_argument

Prototype

```
INTERFACE decoder_argument
MODULE PROCEDURE int8_decoder_argument, int16_decoder_argument, int32_decoder_argument, &
    real32_decoder_argument, real64_decoder_argument, &
    byte_decoder_argument, twobyte_decoder_argument, word_decoder_argument, &
    char_decoder_argument
ENDINTERFACE

FUNCTION byte_decoder_argument(argc, argf, arge, argo)
IMPLICIT NONE
INTEGER(KIND=int16) :: byte_decoder_argument !
!----- Declarations des variables en argument -----
INTEGER(KIND=int32), INTENT(IN) :: argc ! Indice de l'argument
CHARACTER(LEN=*), INTENT(IN) :: argf ! Format de decodage de l'argument
INTEGER(KIND=int32), INTENT(IN) :: arge ! Numero d'erreur
LOGICAL(KIND=byte), INTENT(OUT) :: argo ! Argument du programme decode
```

Description

4.4.8 Fonction decoder_arguments

Prototype

```
FUNCTION decoder_arguments( )
USE m_programme
USE m_decodage
USE m_satellite
USE m_dialogue
USE m_type_fichier_entree
USE m_date
USE m_chemins
USE m_coordonnees_image
USE m_indices_image
USE m_spec_satellite
USE m_etalonage, ONLY : ietal
USE m_modele_aerosol
USE m_tempo
USE m_filtre_nuage
USE m_date_plage
USE m_compression
USE m_coordonnees_extraction
USE m_parametres_chaines
```

```
USE m_dimension_bandeau
USE m_globales
USE m_fit
USE m_parametres_techniques
USE m_parametres_codage
USE m_ordre_octet
USE m_lambda_tau_sortie
!-----
IMPLICIT NONE
INTEGER(KIND=int16)                                :: decoder_arguments
```

Description

4.4.9 Fonction decoder_plage_horaire

Prototype

```
FUNCTION decoder_plage_horaire(buf_,ntab_,tab_,n_)
USE f_message, ONLY : ecrire_message
!-----
IMPLICIT NONE
INTEGER(KIND=int16)                                :: decoder_plage_horaire
!----- Declarations des variables en argument -----
CHARACTER(LEN=*), INTENT(IN)                      :: buf_
INTEGER(KIND=int32), INTENT(IN)                    :: ntab_
INTEGER(KIND=int32), DIMENSION(:), INTENT(INOUT) :: tab_
INTEGER(KIND=int32), INTENT(OUT)                   :: n_
```

Description

4.4.10 Fonction decoder_imodele

Prototype

```
FUNCTION decoder_imodele(buf_,ntab_,tab_,n_)
USE f_message, ONLY : ecrire_message
!-----
IMPLICIT NONE
INTEGER(KIND=int16)                                :: decoder_imodele
!----- Declarations des variables en argument -----
CHARACTER(LEN=*), INTENT(IN)                      :: buf_
INTEGER(KIND=int32), INTENT(IN)                    :: ntab_
INTEGER(KIND=int32), DIMENSION(:), INTENT(INOUT) :: tab_
INTEGER(KIND=int32), INTENT(OUT)                   :: n_
```

Description

4.4.11 Fonction decoder_bornes

Prototype

```
FUNCTION decoder_bornes(buf_,min_,max_)
!-----
IMPLICIT NONE
INTEGER(KIND=int16)                                :: decoder_bornes
!----- Declarations des variables en argument -----
CHARACTER(LEN=*), INTENT(IN)                      :: buf_
REAL(KIND=real32), INTENT(OUT)                   :: min_
REAL(KIND=real32), INTENT(OUT)                   :: max_
```

Description



4.4.12 Fonction decoder_date

Prototype

```
FUNCTION decoder_date(buf_,date_)
USE m_date_plage, ONLY : date
!-----
IMPLICIT NONE
INTEGER(KIND=int16)                :: decoder_date
!----- Declarations des variables en argument -----
CHARACTER(LEN=*), INTENT(IN)       :: buf_
TYPE(date), INTENT(OUT)            :: date_
```

Description

4.4.13 Fonction controler_arguments

Prototype

```
FUNCTION controler_arguments( )
USE m_constantes
USE m_precision
USE m_programme
USE m_satellite
USE m_etalonnage, ONLY : ietal
USE m_type_fichier_entree
USE m_nul
USE m_dialogue
USE m_version
USE m_date
USE m_decodage
USE m_chemins
USE m_coordonnees_image
USE m_indices_image
USE m_spec_satellite
USE m_dimension_angulaire
USE m_dimension_aerosol, ONLY : ntau
USE m_dimension_spectrale
USE m_dimension_polynome, ONLY : npol_fit
USE m_dimension_modele
USE m_modele_aerosol
USE m_tempo
USE m_filtre_nuage
USE m_date_plage
USE m_compression
USE m_coordonnees_extraction
USE m_dimension_bandeau
USE m_globales
USE m_dimension_image
USE m_fit
USE m_parametres_techniques
USE m_parametres_codage
USE m_ordre_octet
USE f_addslash
USE f_liste_fichiers
USE f_message, ONLY : ouvrir_fichier_log,ecrire_message
!-----
IMPLICIT NONE
INTEGER(KIND=int16)                :: controler_arguments
```

Description

4.4.14 Fonction afficher_arguments

Prototype

```
FUNCTION afficher_arguments( )
USE m_dialogue
USE f_message, ONLY : ecrire_message
```

```
!-----
IMPLICIT NONE
INTEGER(KIND=int16) :: afficher_arguments
```

Description

4.4.15 Fonction initialiser_arguments

Prototype

```
FUNCTION initialiser_arguments( )
```

```
!-----
IMPLICIT NONE
INTEGER(KIND=int16) :: initialiser_arguments
```

Description

4.4.16 Fonction initialiser_ligne_bandeau

Prototype

```
FUNCTION initialiser_ligne_bandeau(ibnd)
```

```
IMPLICIT NONE
INTEGER(KIND=int16) :: initialiser_ligne_bandeau
```

```
!----- Declarations des variables en argument -----
INTEGER(KIND=int32), INTENT(IN) :: ibnd ! Indice relatif sur les lignes du bandeau
```

Description

4.4.17 Fonction initialiser_bandeau

Prototype

```
FUNCTION initialiser_bandeau( )
```

```
IMPLICIT NONE
INTEGER(KIND=int16) :: initialiser_bandeau
```

Description

4.4.18 Fonction reporter_bandeau

Prototype

```
FUNCTION reporter_bandeau( )
```

```
IMPLICIT NONE
INTEGER(KIND=int16) :: reporter_bandeau
```

Description



4.4.19 Fonction charger_bandeau

Prototype

```
FUNCTION charger_bandeau(ilig)
USE m_nul
USE m_spec_satellite
USE m_dimension_image
USE m_chemins
USE m_coordonnees_image
USE f_message, ONLY : ecrire_message
```

Description

4.4.20 Fonction calculer_angstrom

Prototype

```
FUNCTION calculer_angstrom( )
IMPLICIT NONE
INTEGER(KIND=int16) :: calculer_angstrom
```

Description

4.4.21 Fonction calculer_coeff_interp_geom

Prototype

```
FUNCTION calculer_coeff_interp_geom()
USE m constantes_math
USE m_angles_gauss
USE m_angles_diffusion
USE m_interpolation_geometrie
USE m_dimension_angulaire
USE m_dimension_image
USE m_bandeau
USE m_pixel_type
!-----
IMPLICIT NONE
INTEGER(KIND=int16) :: calculer_coeff_interp_geom
```

Description

4.4.22 Fonction calculer_fit

Prototype

```
FUNCTION calculer_fit(imod)
USE m_dialogue
USE m_nul
USE m_precision
USE m_dimension_angulaire
USE m_dimension_aerosol, ONLY : ntau
USE m_dimension_spectrale, ONLY : nlmb_inv
USE m_dimension_modele
USE m_dimension_polynome, ONLY : npol_fit
USE m_modele_aerosol
USE m_tau_aerosol
USE m_tables
USE f_fitter, ONLY : svdfit
!-----
IMPLICIT NONE
INTEGER(KIND=int16) :: calculer_fit
```

Description

4.4.23 Fonction calculer_param_pixel

Prototype

```
FUNCTION calculer_param_pixel()
USE m_ligne
USE m_satellite
USE m_dimension_image, ONLY : ncol
USE m_spec_satellite
USE m constantes_math
USE m_cooronnees_image
USE m_bandeau
USE m_tempo
USE m_interpolation_geometrie
USE m_dimension_bandeau
USE m_parametres_techniques
USE m_pixel_type
USE m_tmp_parametres_pixel
USE m_projection, ONLY : test_projection
USE f_bandeau, ONLY : initialiser_ligne_bandeau
USE f_message, ONLY : ecrire_message
!-----
IMPLICIT NONE
!----- Declaration de la fonction -----
INTEGER(KIND=int16) :: calculer_param_pixel
```

Description

La fonction `calculer_param_pixel` a pour objectif de calculer les paramètres temporels et géométriques pour les pixels utiles, après application du masque `tmc`. Le traitement est effectué à chaque appel de la fonction pour les pixels de la dernière ligne du bandeau courant.

Organigramme

4.4.24 Fonction charger_table_mol

Prototype

```
FUNCTION charger_table_mol( )
USE m_dimension_angulaire
USE m_dimension_spectrale
IMPLICIT NONE
INTEGER(KIND=int16) :: charger_table_mol
```

Description

4.4.25 Fonction charger_table_frn

Prototype

```
FUNCTION charger_table_frn( )
IMPLICIT NONE
INTEGER(KIND=int16) :: charger_table_frn
```

Description

4.4.26 Fonction charger_table_tot

Prototype

```
FUNCTION charger_table_tot( )
USE m_dimension_angulaire
USE m_dimension_spectrale
USE m_dimension_aerosol
USE m_dimension_modele
IMPLICIT NONE
INTEGER(KIND=int16) :: charger_table_tot
```



Description

4.4.27 Fonction charger_table_fit

Prototype

```
FUNCTION charger_table_fit( )  
USE m_dimension_angulaire  
USE m_dimension_spectrale  
USE m_dimension_modele  
USE m_dimension_polynome  
USE m_modele_atmosphere, ONLY : iatm  
USE f_calculer_fit  
IMPLICIT NONE  
INTEGER(KIND=int16)          :: charger_table_fit
```

Description

4.4.28 Fonction charger_tables

Prototype

```
FUNCTION charger_tables( )  
IMPLICIT NONE  
INTEGER(KIND=int16)          :: charger_tables
```

Description

4.4.29 Fonction coder_tau550

Prototype

```
FUNCTION coder_tau550(tau550_min,tau550_max,pente,ordorig)  
USE m_image, ONLY : tau550  
USE m_bandeau  
USE m_codage, ONLY : tau550_int8  
IMPLICIT NONE  
INTEGER(KIND=int16)          :: coder_tau550  
!----- Declarations des variables en argument -----  
REAL(KIND=real32), INTENT(IN)      :: tau550_min  !  
REAL(KIND=real32), INTENT(IN)      :: tau550_max  !  
REAL(KIND=real32), INTENT(IN)      :: pente       !  
REAL(KIND=real32), INTENT(IN)      :: ordorig     !
```

Description

4.4.30 Fonction coder_angstrom

Prototype

```
FUNCTION coder_angstrom(angstrom_min,angstrom_max,pente,ordorig)  
USE m_image, ONLY : angstrom  
USE m_bandeau  
USE m_codage, ONLY : angstrom_int8  
IMPLICIT NONE  
INTEGER(KIND=int16)          :: coder_angstrom  
!----- Declarations des variables en argument -----  
REAL(KIND=real32), INTENT(IN)      :: angstrom_min !  
REAL(KIND=real32), INTENT(IN)      :: angstrom_max !  
REAL(KIND=real32), INTENT(IN)      :: pente       !  
REAL(KIND=real32), INTENT(IN)      :: ordorig     !
```

Description

4.4.31 Fonction coder_rho

Prototype

```
FUNCTION coder_rho(ilmb,rho_min,rho_max,pente,ordorig)
USE m_bandeau
USE m_codage, ONLY : rho_mes_int8
IMPLICIT NONE
INTEGER(KIND=int16)                :: coder_rho
!----- Declarations des variables en argument -----
INTEGER(KIND=int32), INTENT(IN)    :: ilmb      ! Indice de la longueur d'onde
!----- Declarations des variables en argument -----
REAL(KIND=real32), INTENT(IN)      :: rho_min   !
REAL(KIND=real32), INTENT(IN)      :: rho_max   !
REAL(KIND=real32), INTENT(IN)      :: pente     !
REAL(KIND=real32), INTENT(IN)      :: ordorig   !
```

Description

4.4.32 Fonction compresser_fichiers

Prototype

```
FUNCTION compresser_fichier(fic)
IMPLICIT NONE
INTEGER(KIND=int16)                :: compresser_fichier
!----- Declarations des variables en argument -----
CHARACTER(LEN=*), INTENT(IN)       :: fic      !
```

Description

4.4.33 Fonction decompresser_fichiers

Prototype

```
FUNCTION decompresser_fichier(fic)
IMPLICIT NONE
INTEGER(KIND=int16)                :: decompresser_fichier
!----- Declarations des variables en argument -----
CHARACTER(LEN=*), INTENT(IN)       :: fic      !
```

Description

4.4.34 Fonction compresser_fichiers_sortie

Prototype

```
FUNCTION compresser_fichiers_sortie( )
USE m_chemins
USE m_compression, ONLY : comp_out
USE m_dimension_spectrale, ONLY : nlmb_inv
USE m_test_parametres, ONLY : sortie
USE m_modele_aerosol, ONLY : imodele
!-----
IMPLICIT NONE
INTEGER(KIND=int16)                :: compresser_fichiers_sortie
```

Description



4.4.35 Fonction `ecrire_ligne`

Prototype

```
FUNCTION ecrire_ligne(ilig)
IMPLICIT NONE
INTEGER(KIND=int16)                :: ecrire_ligne
!----- Declarations des fonctions externes -----
INTEGER                            :: swap_float_buf ! Fonction en C
!----- Declarations des variables en argument -----
INTEGER(KIND=int32), INTENT(IN)    :: ilig ! Indice de la ligne courante de l'image courante
```

Description

4.4.36 Fonction `etalonner`

Prototype

```
FUNCTION etalonner(ilig,ilmb)
USE m_constants_math
USE m_tempo
USE m_header
USE m_etalonnage
USE m_bande_spectrale
USE m_satellite
USE m_dimension_spectrale, ONLY : tab_ilmb, nlmb_tot
USE m_dimension_bandeau
USE m_bandeau
USE m_image, ONLY : CN
USE m_pixel_type
USE m_parametres_techniques
USE m_parametres_codage
USE f_message, ONLY : ecrire_message
USE f_coder, ONLY : coder_rho
USE m_dimension_image
IMPLICIT NONE
INTEGER(KIND=int16)                :: etalonner
!----- Declarations des variables en argument -----
INTEGER(KIND=int32), INTENT(IN)    :: ilig ! Indice de la ligne courante de l'image courante
INTEGER(KIND=int32), INTENT(IN)    :: ilmb ! Indice de la longueur d'onde de l'image a etalonner
```

Cette fonction étalonne les comptes numériques de la dernière ligne du bandeau uniquement, de manière à obtenir la réflectance mesurée au sommet de l'atmosphère correspondante. Seuls les pixels de type `mer` sont étalonnés. Un test est effectué pour forcer les valeurs négatives de réflectance à une valeur nulle.

Si le variable `ietal` vaut 0, les informations concernant l'étalonnage radiométrique des mesures, fournies dans l'en-tête des données, sont utilisées.

Pour les instruments MSG, les informations concernant l'étalonnage radiométrique des mesures, sont fournies dans l'en-tête principale des données de niveau 1.5 . La quantité physique mesurée (canal `i`) s'exprime grâce à la relation :

Le coefficient d'étalonnage² pour le canal spectral `i` `coef_etal(i)%Cft` estimé pour une date donnée, se calcule par la formule

$$\text{coef_etal}(i)\%Cft = \text{coef_etal}(i)\%Cf + (\text{coef_etal}(i)\%Dj * Nj * 10^{-5}) \quad (4.1)$$

ou

`coef_etal% Cf(i)` est le coefficient d'étalonnage à la date du lancement de l'instrument considéré, `coef_etal(i)%Dj` est le coefficient de dérive journalière³, `Nj` est le nombre de jours depuis la date du lancement.

²dont l'unité est $W.m^{-2}.sr^{-1}.CN^{-1}$

³dont l'unité est $W.m^{-2}.sr^{-1}.CN^{-1}.jour^{-1}$

4.4.37 Fonction `extraire_pixel_produit`

Prototype

Description

4.4.38 Fonction `extraire_produit`

Prototype

Description

4.4.39 Fonction `fermer_fichiers1`

Prototype

Description

4.4.40 Fonction `fermer_fichiers2`

Prototype

Description

4.4.41 Fonction `filtrer_nuage`

Prototype

Description

4.4.42 Fonction `f_poly`

Prototype

Description

4.4.43 Fonction `pythag`

Prototype

Description



4.4.44 Fonction svdfit

Prototype

Description

4.4.45 Fonction gausslegendre

Prototype

```
INTERFACE gausslegendre
  MODULE PROCEDURE gausslegendre_real32,gausslegendre_real64
ENDINTERFACE

FUNCTION gausslegendre_real32(mu1,mu2,n)
  IMPLICIT NONE
!----- Declarations des variables en argument -----
  REAL(KIND=real32),      INTENT(IN)      :: mu1
  REAL(KIND=real32),      INTENT(IN)      :: mu2
  INTEGER(KIND=int32),     INTENT(IN)      :: n
!----- Declaration de la fonction -----
  TYPE(gauss), DIMENSION(n)              :: gausslegendre_real32

FUNCTION gausslegendre_real64(mu1,mu2,n)
  IMPLICIT NONE
!----- Declarations des variables en argument -----
  REAL(KIND=real64),      INTENT(IN)      :: mu1
  REAL(KIND=real64),      INTENT(IN)      :: mu2
  INTEGER(KIND=int32),     INTENT(IN)      :: n
!----- Declaration de la fonction -----
  TYPE(gauss), DIMENSION(n)              :: gausslegendre_real64
```

Description

Cette fonction à valeurs tableau de types dérivés renvoie les *n* angles calculés aux points de Gauss entre les valeurs *mu1* et *mu2*, ainsi que leurs poids associés. L'interface explicite de la fonction est déclarée dans un bloc MODULE PROCEDURE.

4.4.46 Fonction glitter

Prototype

```
INTERFACE glitter
  MODULE PROCEDURE glitter_real32,glitter_real64
ENDINTERFACE

PURE FUNCTION glitter_real32(vent,mus,muv,thd,rF)
!-----
  IMPLICIT NONE
!----- Declaration de la fonction -----
  REAL(KIND=real32)              :: glitter_real32
!----- Declarations des variables en argument -----
  REAL(KIND=real32), INTENT(IN)  :: vent ! Vitesse du vent (m/s)
  REAL(KIND=real32), INTENT(IN)  :: mus  ! Cosinus de l'angle zenithal solaire
  REAL(KIND=real32), INTENT(IN)  :: muv  ! Cosinus de l'angle zenithal de visée
  REAL(KIND=real32), INTENT(IN)  :: thd  ! Angle de diffusion (degrés)
  REAL(KIND=real32), INTENT(IN)  :: rF   ! Coefficient de reflexion de Fresnel

PURE FUNCTION glitter_real64(vent,mus,muv,thd,rF)
!-----
  IMPLICIT NONE
!----- Declaration de la fonction -----
  REAL(KIND=real64)              :: glitter_real64
!----- Declarations des variables en argument -----
```

```
REAL(KIND=real64), INTENT(IN)      :: vent ! Vitesse du vent (m/s)
REAL(KIND=real64), INTENT(IN)      :: mus ! Cosinus de l'angle zenithal solaire
REAL(KIND=real64), INTENT(IN)      :: muv ! Cosinus de l'angle zenithal de visée
REAL(KIND=real64), INTENT(IN)      :: thd ! Angle de diffusion (degres)
REAL(KIND=real64), INTENT(IN)      :: rF ! Coefficient de reflexion de Fresnel
```

Description

Cette fonction permet de calculer la réflectance du glitter $\rho_g(\theta_s, \theta_v, \phi)$ modélisée de manière analytique par [Cox et Munk, 1954] telle que :

$$\rho_g(\theta_s, \theta_v, \phi) = \frac{\pi r_F(\theta)}{4 \mu_s \mu_v \cos^4(\beta)} p_s(\theta_s, \theta_v, \phi) \quad 4.4.47$$

où l'angle β est l'angle entre la normale locale et la normale de la facette, et $p_s(\theta_s, \theta_v, \phi)$ la fonction de densité de probabilité donnée par la relation :

$$p_s(\theta_s, \theta_v, \phi) = \frac{1}{\pi \sigma^2} \exp\left(\frac{-\tan^2(\beta)}{\sigma^2}\right)$$

et où σ , l'écart-type des pentes de vagues, est une fonction de la vitesse du vent, v tel que :

$$\sigma^2 = 0.003 + 5.12 \cdot 10^{-3} v$$

4.4.47 Fonction `tester_glitter`

Prototype

```
FUNCTION tester_glitter()
USE m_types
USE m_constantes
USE m_dimension_image
USE m_tables, ONLY : tab_frn
USE m_bandeau
USE m_parametres_techniques
USE m_interpolation_geometrie
USE f_interpoler, ONLY : interpoler1
!-----
IMPLICIT NONE
!----- Declaration de la fonction -----
INTEGER(KIND=int16)      :: tester_glitter
```

Description

La fonction `tester_glitter` permet de déterminer, connaissant la géométrie $(\theta_s, \theta_v, \phi)$ de visée d'un pixel donné, si la réflectance du glitter $\rho_g(\theta_s, \theta_v, \phi)$ estimée par la relation , est supérieure à n certain seuil pointé par l'argument `rho_glt_lim`⁴. Dans la modélisation précédente, on prend pour la vitesse de vent la valeur pointée par l'argument `vent_lim`⁵. Cette manière de procéder permet de s'affranchir simplement des pixels contaminés par le phénomène du glitter, sans connaissance *a priori* par une source exogène des direction et vitesse du vent de surface.

4.4.48 Fonction `initialiser_variables`

Prototype

Description

⁴La valeur par défaut de cet argument est de 0.005

⁵La valeur par défaut de cet argument est de 8 $m.s^{-1}$



4.4.49 Fonction initialiser_parametres_codage

Prototype

Description

4.4.50 Fonction ouvrir_fichier

Prototype

```
FUNCTION ouvrir_fichier(nul,dir,fic,frm,acc,act,rcl,sts,pos,ios)
IMPLICIT NONE
!----- Declaration de la fonction -----
INTEGER(KIND=int16)                                :: ouvrir_fichier
!----- Declarations des variables en argument -----
INTEGER(KIND=int32), INTENT(IN)                    :: nul  !
CHARACTER(LEN=*), INTENT(IN), OPTIONAL              :: dir  !
CHARACTER(LEN=*), INTENT(IN)                        :: fic  !
CHARACTER(LEN=*), INTENT(IN), OPTIONAL              :: frm  !
CHARACTER(LEN=*), INTENT(IN), OPTIONAL              :: acc  !
CHARACTER(LEN=*), INTENT(IN), OPTIONAL              :: act  !
INTEGER(KIND=int32), INTENT(IN), OPTIONAL           :: rcl  !
CHARACTER(LEN=*), INTENT(IN), OPTIONAL              :: sts  !
CHARACTER(LEN=*), INTENT(IN), OPTIONAL              :: pos  !
INTEGER(KIND=int32), INTENT(OUT), OPTIONAL          :: ios  !
```

Description

4.4.51 Fonction ouvrir_fichiers_ancillaire

Prototype

Description

4.4.52 Fonction ouvrir_fichiers_tables

Prototype

Description

4.4.53 Fonction initialiser

Prototype

Description

4.4.54 Fonction initialiser_day

Prototype

Description

4.4.55 Fonction initialiser_mth

Prototype

Description

4.4.56 Fonction initialiser_ext

Prototype

Description

4.4.57 Fonction interpoler_aerosol

Prototype

Description

4.4.58 Fonction interpoler

Prototype

Description

4.4.59 Fonction interpoler_moleculaire

Prototype

Description

4.4.60 Fonction inverser_aerosol

Prototype

Description

4.4.61 Fonction faire_nom_fichier_header

Prototype

Description



4.4.62 Fonction `lire_header`

Prototype

Description

Fonction permettant de lire le fichier contenant l'en-tête associée au(x) fichier(s) image(s) à traiter. Deux fonctions distinctes en C sont appelées pour lire ce fichier, selon que l'on souhaite traiter des images de type *MFG* ou bien *MSG*. Des informations utiles comme le jour Julien, la date, l'heure d'acquisition, les valeurs des longitudes nominale et effective, le nombre de lignes et de colonnes de chaque image, la position éventuelle d'une scène dans une grille de référence, les coefficients d'étalonnage, etc ... sont décodés et conservés dans une structure *ad-hoc* pour la suite du traitement.

4.4.63 Fonction `faire_nom_fichier_sortie`

Prototype

Description

4.4.64 Fonction `ouvrir_fichiers_sortie`

Prototype

Description

4.4.65 Fonction `lire_fichier_entree`

Prototype

Description

4.4.66 Fonction `lire_fichier_mie_6S`

Prototype

Description

Fonction permettant la lecture du fichier formaté obtenu en sortie du code 6S, rassemblant les résultats des calculs de Mie obtenus pour un modèle d'aérosol (distribution(s) en taille, indice de réfraction) donné. Ces résultats recensent les principales caractéristiques radiatives du modèle d'aérosol, à savoir les coefficients d'extinction et de diffusion, paramètre d'asymétrie, fonction de phase, pour les 10 longueurs d'ondes imposées de manière interne dans 6S⁶. Ce fichier⁷ est utilisé pour calculer les propriétés radiatives de l'atmosphère, et notamment les

⁶Les valeurs de ces 10 longueurs d'ondes sont contenues dans le tableau `spct_6s(imod,ilmb)%lmb`

⁷Un fichier pour chacun des modèles d'aérosols

tables précalculées de ce code. La lecture de ce fichier permet donc de récupérer aux 10 longueurs d'ondes λ_i utilisées par 6S, le coefficient d'extinction $\sigma_{ext}^{mod}(\lambda_i)$, le coefficient de diffusion $\sigma_{ext}^{mod}(\lambda_i)$, l'albedo de diffusion simple $\omega_0^{mod}(\lambda_i)$, le paramètre d'asymétrie $g^{mod}(\lambda_i)$. On récupère également les valeurs de la fonction de phase normalisée pour les 83 angles de diffusion et 10 longueurs d'ondes utilisées par 6S $P(\Theta_d, \lambda_i)$.

4.4.67 Fonction faire_nom_fichier_image

Prototype

Description

4.4.68 Fonction lire_ligne

Prototype

Description

4.4.69 Fonction faire_liste

Prototype

Description

Fonction qui génère la liste des année, mois, jour, plage horaire contenus dans la plage de temps définies par les arguments du programme `date_debut%annee`, `date_debut%mois`, `date_debut%jour`, `date_fin%annee`, `date_fin%mois`, `date_fin%jour`, `plage_horaire`. Une entrée de cette liste correspond à un quadruplet `annee,mois,jour,plage_horaire`. Cette fonction calcule également le nombre d'année, mois, jour, ainsi que le nombre d'entrées total de la liste, qui doit rester plus petit que la valeur contenue dans le paramètre `NB_IMAGE_MAX`⁸. La fonction retourne la valeur 0 en cas d'exécution nominale.

4.4.70 Fonction ouvrir_fichier_log

Prototype

```
\begin{verbatim}
FUNCTION ouvrir_fichier_log( )
IMPLICIT NONE
INTEGER(KIND=int16)                                :: ouvrir_fichier_log
```

Description

Fonction permettant l'ouverture en écriture du fichier de commentaires du code. La fonction retourne la valeur 0 en cas d'exécution nominale.

⁸c.f. module `m_constants` dans le fichier `m_meteosat.F`



4.4.71 Fonction fermer_fichier_log

Prototype

```
FUNCTION fermer_fichier_log( )  
IMPLICIT NONE  
INTEGER(KIND=int16) :: fermer_fichier_log
```

Description

Fonction permettant la fermeture du fichier de commentaires du code. La fonction retourne la valeur 0 en cas d'exécution nominale.

4.4.72 Fonction ouvrir_fichier_message

Prototype

```
FUNCTION ouvrir_fichier_message( )  
IMPLICIT NONE  
INTEGER(KIND=int16) :: ouvrir_fichier_message
```

Description

Fonction permettant l'ouverture en lecture du fichier des messages d'erreur du code. La fonction retourne la valeur 0 en cas d'exécution nominale.

4.4.73 Fonction fermer_fichier_message

Prototype

```
FUNCTION fermer_fichier_message( )  
IMPLICIT NONE  
INTEGER(KIND=int16) :: fermer_fichier_message
```

Description

Fonction permettant la fermeture du fichier des messages d'erreur du code. La fonction retourne la valeur 0 en cas d'exécution nominale.

4.4.74 Fonction ecrire_message

Prototype

```
FUNCTION ecrire_message(no_,typ_,msg_,nul_)  
IMPLICIT NONE  
INTEGER(KIND=int16) :: ecrire_message  
!----- Declarations des variables en argument -----  
INTEGER, INTENT(IN) :: no_ ! Numero du message d'erreur  
INTEGER, INTENT(IN), OPTIONAL :: typ_ ! Type du message :  
! - 0 : message d'erreur (valeur par omission)  
! - 1 : message d'information  
CHARACTER(LEN=*), INTENT(IN), OPTIONAL :: msg_ ! Message optionnel supplémentaire a ajouter a la fin  
! du message (entre coquets)  
INTEGER, INTENT(IN), OPTIONAL :: nul_ ! Numero d'unité logique optionnel associe a un fichier  
! vers lequel forcer la redirection du message
```

Description

Fonction permettant, connaissant le numéro `no_` du message, de décoder, puis de retourner la chaîne contenant le message d'erreur associé à ce numéro, présent dans le fichier des messages \$METrac/met/inc/erreur.meteosat Cette fonction possède en outre trois arguments optionnels permettant :

- - **typ_** : de préciser le type du message, *erreur* ou *information*,
- - **msg_** : d'ajouter à la fin du message d'erreur un message entre crochets,
- - **nul_** : de forcer la redirection de l'écriture du message d'erreur vers une unité logique, avec test sur l'existence du fichier associé à cette unité.

La fonction retourne le numéro d'erreur associé au message lorsque le traitement est nominal, une valeur négative sinon.

4.4.75 Fonction polynome2

Prototype

```
INTERFACE polynome2
  MODULE PROCEDURE real32_polynome2, real64_polynome2
ENDINTERFACE
INTERFACE polynome3
  MODULE PROCEDURE real32_polynome3, real64_polynome3
ENDINTERFACE
INTERFACE polynome4
  MODULE PROCEDURE real32_polynome4, real64_polynome4
ENDINTERFACE
INTERFACE polynome5
  MODULE PROCEDURE real32_polynome5, real64_polynome5
ENDINTERFACE
INTERFACE polynome6
  MODULE PROCEDURE real32_polynome6, real64_polynome6
ENDINTERFACE
```

Description

4.4.76 Fonction polynome3

Idem que la fonction `polynome2` décrite au paragraphe 4.4.75, renvoie la valeur d'une fonction polynomiale d'ordre 3 à l'abscisse x connaissant les coefficients c_i .

4.4.77 Fonction polynome4

Idem que la fonction `polynome2` décrite au paragraphe 4.4.75, renvoie la valeur d'une fonction polynomiale d'ordre 4 à l'abscisse x connaissant les coefficients c_i .

4.4.78 Fonction polynome5

Idem que la fonction `polynome2` décrite au paragraphe 4.4.75, renvoie la valeur d'une fonction polynomiale d'ordre 5 à l'abscisse x connaissant les coefficients c_i .

4.4.79 Fonction polynome6

Idem que la fonction `polynome2` décrite au paragraphe 4.4.75, renvoie la valeur d'une fonction polynomiale d'ordre 6 à l'abscisse x connaissant les coefficients c_i .

4.4.80 Fonction faire_nom_produit

Prototype

Description



4.4.81 Fonction ouvrir_fichier_produit

Prototype

```
FUNCTION ouvrir_fichier_produit(cext,cstatus)
USE m_dimension_image, ONLY : ncol
!-----
IMPLICIT NONE
INTEGER(KIND=int16)                                :: ouvrir_fichier_produit
!----- Declarations des variables en argument -----
CHARACTER(LEN=*), INTENT(IN), OPTIONAL              :: cext
CHARACTER(LEN=*), INTENT(IN), OPTIONAL              :: cstatus
```

Description

Cette fonction permet de contrôler l'ouverture nominale en lecture (`cstatus = 'OLD'`) ou en écriture (`cstatus = 'REPLACE'`) d'un fichier de produit⁹ pour une acquisition (SLT), pour une moyenne journalière (DAY) ou mensuelle (MTH).

4.4.82 Fonction lire_fichier_produit

Prototype

```
FUNCTION lire_fichier_produit(cext)
USE m_dimension_image, ONLY : nlig
USE m_synthese
!-----
IMPLICIT NONE
INTEGER(KIND=int16)                                :: lire_fichier_produit
!----- Declarations des variables en argument -----
CHARACTER(LEN=*), INTENT(IN)                       :: cext
```

Description

Cette fonction permet de lire un fichier produit pour une acquisition

4.4.83 Fonction ecrire_fichier_produit

Prototype

Description

4.4.84 Fonction fermer_fichier_produit

Prototype

Description

⁹Indifféremment épaisseur optique ou coefficient d'Angström

4.4.85 Fonction read_header_msg

Prototype

```
int read_header_msg(char *file, int *pos, int *SatelliteId, float *NominalLongitude,  
                    int *SatelliteStatus, int *annee, int *jj, float *TrueRepeatCycleStart,  
                    float *PlannedForwardScanEnd,  
                    int *SouthernLinePlanned, int *NorthernLinePlanned,  
                    int *EasternLinePlanned, int *WesternLinePlanned,  
                    int *LowerSouthLinePlanned, int *LowerNorthLinePlanned,  
                    int *LowerEastColumnPlanned, int *LowerWestColumnPlanned,  
                    int *UpperSouthLinePlanned, int *UpperNorthLinePlanned,  
                    int *UpperEastColumnPlanned, int *UpperWestColumnPlanned,  
                    double Cal_Slope[], double Cal_Offset[],  
                    float *LongitudeOfSSP, float *LineDirGridStepVIS_IR,  
                    float *ColumnDirGridStepVIS_IR, int *GridOriginVIS_IR,  
                    float *LineDirGridStepHRV, float *ColumnDirGridStepHRV,  
                    int *GridOriginHRV);
```

Description

Cette fonction décode l'en-tête principale au format natif 1.5 d'un fichier de type *MSG*. Elle récupère les informations pertinentes contenues dans cette en-tête utile dans la suite du traitement,

4.4.86 Fonction read_header_mfg

Prototype

```
int read_header_mfg(char *file, int *pos, int *year, int *jday, int *plage, int *date,  
                    int *heure, float *ssp, int *line1, int *pixel1, int *nlines,  
                    int *npixels);
```

Description

Cette fonction décode l'en-tête principale au format *OpenMTP* d'un fichier de type *MFG*. Elle récupère les informations pertinentes contenues dans cette en-tête utile dans la suite du traitement,



4.5 Liste des fonctions du répertoire com

Fichier source	Fonction	Section
compressionfile.c	bunzip2file	4.5.1
	bzip2file	4.5.2
	gunzip2file	4.5.3
	gzip2file	4.5.4
date.c	jj2jma	4.5.5
	jma2jj	4.5.6
	a2j	4.5.7
	julday	4.5.8
	caldat	4.5.9
	cds2jma	4.5.10
	cds2jj	4.5.11
	cds2jmahms	??
dsol.c	dsol	4.5.13
locate.c	localiser1	4.5.14
	localiser2	4.5.15
meteosat_geo.c	ligcol2latlon_geo	4.5.16
	latlon2ligcol_geo	4.5.17
	ligcol2tu_geo	4.5.18
	calc_grid	4.5.19
visee_geo.c	elevation_geo	4.5.20
	azimut_geo	4.5.21
ordre_octets.c	ordre_octets	4.5.22
possol.c	possol	4.5.23

swap.c	swap_float	4.5.24
	swap_float_buf	4.5.25
	swap_double	4.5.26
	swap_double_buf	4.5.27
	swap_int	4.5.28
	swap_int_buf	4.5.29
	swap_uint	4.5.30
	swap_uint_buf	4.5.31
	swap_long	4.5.32
	swap_long_buf	4.5.33
	swap_short	4.5.34
	swap_short_buf	4.5.35
	swap_ushort	4.5.36
	swap_ushort_buf	4.5.37

TAB. 4.2: Description sommaire des fonctions du répertoire com

4.5.1 Fonction bunzip2file

Prototype

```
int bunzip2file(char *file);
```

Description

Décompression d'un fichier en invoquant la commande **bunzip2** (extension **.bz2**).

Cette fonction teste premièrement si le fichier **file** existe; Si ce dernier existe, elle teste alors l'existence de l'extension **.bz2**. Si celui-ci n'existe pas, elle teste alors l'existence du fichier **file.<ext>**; En cas de succès à l'un de ces tests, elle exécute par l'intermédiaire de la primitive **system** la commande appropriée.

La fonction retourne 0 si nominale, 1 sinon.

4.5.2 Fonction bzip2file

Prototype

```
int bzip2file(char *file);
```

Description

Compression d'un fichier en invoquant la commande **bzip2** (extension **.bz2**).

Cette fonction teste si le fichier **file** existe; s En cas de succès, elle exécute par l'intermédiaire de la primitive **system** la commande **bzip2**.

La fonction retourne 0 si nominale, 1 sinon.

4.5.3 Fonction gunzipfile

Prototype



```
int gunzipfile(char *file);
```

Description

Idem que la fonction `bunzip2file`, pour des fichiers compressés au moyen de la commande système `gzipfile`.

4.5.4 Fonction `gzipfile`

Prototype

```
int gzipfile(char *file);
```

Description

Idem que la fonction `bzip2file`, pour compresser des fichiers au moyen de la commande système `gzipfile`.

4.5.5 Fonction `jj2jma`

Prototype

```
int jj2jma(int j, int *jday,int *month,int *ia);
```

Description

4.5.6 Fonction `jma2jj`

Prototype

```
int jma2jj(int jour,int mois,int annee);
```

Description

4.5.7 Fonction `a2j`

Prototype

```
int a2j(int annee);
```

Description

4.5.8 Fonction `julday`

Prototype

```
int julday(int annee,int mois,int jour);
```

Description

La fonction `julday` provient de *Numerical Recipes*. La valeur retournée par cette fonction est le numéro du jour Julien qui commence à midi de la date calendaire définie par l'année `annee`, le numéro du mois `mois`, et le jour du mois `jour`.

4.5.9 Fonction caldat

Prototype

```
int caldat(int jjulien,int *jour,int *mois,int *annee);
```

Description

La fonction `caldat` provient de *Numerical Recipes*. Cette fonction est l'inverse de la fonction `julday` précédemment décrite. Connaissant la valeur du jour Julien `jjulien`, la fonction retourne l'année `annee`, le numéro du mois `mois`, et le jour du mois `jour`.

4.5.10 Fonction cds2jma

Prototype

```
int cds2jma(int cdsday,int *jour,int *mois,int *annee);
```

Description

Calcul de l'année courante `annee`, ainsi que du mois `mois`, et du jour du mois `jour`, connaissant le nombre de jour écoulés depuis le 1er Janvier 1958. Le compteur journalier est défini à 0 pour cette dernière date. Cette fonction fait appel aux fonctions 4.5.8[julday] et 4.5.9[caldat] définies précédemment.

4.5.11 Fonction cds2jj

Prototype

```
int cds2jj(int cdsday,int *jjulien);
```

Description

Calcul du jour julien `jjulien`, connaissant le nombre de jour `cdsday` écoulés depuis le 1^{er} Janvier 1958. Le compteur journalier est défini à 0 pour cette dernière date. Cette fonction fait appel aux fonctions `julday` et `jma2jj` définies précédemment.

4.5.12 Fonction cds2jmahms

Prototype

```
int cds2jmahms(int cdsday,int cdsmsec,int *jour,int *mois,int *annee,int *heure,int *min,int *sec)
```

Description

Calcul de l'année courante `annee`, ainsi que du mois `mois`, et du jour du mois `jour`, de l'heure `heure`, minutes `min` et secondes `sec`, connaissant le nombre de jour écoulés depuis le 1^{er} Janvier 1958, `cdsday`, et le nombre de millisecondes du jour `cdsmsec`. Le compteur journalier est défini à 0 pour cette dernière date. Cette fonction fait appel aux fonctions `julday` et `caldat` définies précédemment.



4.5.13 Fonction `dsol`

Prototype

```
float dsol(int jj);
```

Description

Fonction permettant de prendre en compte la variation du flux solaire extraterrestre comme une fonction de l'indice du jour dans le calendrier Julien [*Paltridge et Platt, 1977*]. La fonction `dsol` renvoie un coefficient multiplicatif f qui s'applique à la constante solaire E_s tel que :

$$f = \frac{1}{(1 - e \cos(omm))^2}$$

avec

$$omm = 0.9856(jday - 4) \frac{\pi}{180}$$

où e , excentricité de l'orbite elliptique terrestre vaut 0,01673 et `jday` est l'indice du jour dans le calendrier Julien.

4.5.14 Fonction `localiser1`

Prototype

```
int localiser1(float xxx[], int *n, float *x, int *j);
```

Description

Recherche d'une valeur dans une table ordonnée par une méthode dichotomique. Soit un tableau `xxx` de `n` valeurs réelles, et une valeur `x` réelle. La fonction `localiser1` retourne une valeur entière `j`, telle que `x` soit compris entre `xx[j]` et `xx[j+1]`. Le tableau `xx` doit être constitué de valeurs monotones croissantes ou décroissantes. Les indices `j=0` ou `j=n` sont retournés pour indiquer que `x` est en dehors de la plage de valeurs.

4.5.15 Fonction `localiser2`

Prototype

```
int localiser2(float xxx[], int *n, float *x, int *jlo);
```

Description

Recherche d'une valeur dans une table ordonnée par une méthode dichotomique, lorsque des appels successifs sont corrélés. Soit un tableau `xxx` de `n` valeurs réelles, et une valeur `x` réelle. La fonction `localiser1` retourne une valeur entière `jlo`, telle que `x` soit compris entre `xx[j]` et `xx[j+1]`. `jlo` en entrée est considérée comme indice initial. Le tableau `xx` doit être constitué de valeurs monotones croissantes ou décroissantes. Les indices `jlo=0` ou `jlo=n` sont retournés pour indiquer que `x` est en dehors de la plage de valeurs.

4.5.16 Fonction `ligcol2latlon_geo`

Prototype

```
int ligcol2latlon_geo(int isat,int ires,float lon_prj,float lig,float col,  
                     float *lat,float *lon,int *utile);
```

Description

Cette fonction permet de calculer les coordonnées géographiques d'un point de la surface terrestre associées aux coordonnées image d'un pixel, pour les satellites géosationnaires de type *MFG* ou *MSG*. Connaissant les coordonnées de ligne `lig` et de colonne `col` dans une grille de référence ayant pour origine l'angle Sud-Est, d'un pixel d'une image de type *MFG* (`isat = 0`) ou *MSG* (`isat = 1`), ainsi que la longitude¹⁰ du satellite `lon_prj` en degrés décimaux, la fonction `ligcol2latlon_geo` calcule les coordonnées géographiques correspondantes, respectivement la latitude¹¹ `lat` et la longitude¹², `lon` tous deux en degrés décimaux. La fonction retourne la valeur 0 si tout se déroule de façon nominale, -1 si les coordonnées géographiques du point trouvé sont en dehors du disque visible par le satellite, -3 si les coordonnées images du pixel sont incohérentes.

4.5.17 Fonction `latlon2ligcol_geo`

Prototype

```
int latlon2ligcol_geo(int isat,int ires,float lon_prj,float lat,float lon,  
                     float *lig,float *col,int *utile);
```

Description

Cette fonction permet de calculer les coordonnées images dans une grille de référence d'un point de la surface terrestre, pour les satellites géosationnaires de type *MFG* ou *MSG*. Cette grille de référence est constituée d'un nombre variable de lignes et de colonnes suivant l'instrument et la résolution spatiale de l'image considérée. Elle est définie telle que les lignes soient orientées dans la direction Est-Ouest, et les colonnes dans la direction Nord-Sud. Connaissant les coordonnées géographiques, latitude `lat` et longitude `lon` d'un point de la surface terrestre, ainsi que la longitude du satellite `lon_prj` en degrés décimaux, la fonction `latlon2ligcol_geo` calcule les coordonnées images de ligne `lig` et de colonne `col` dans la grille de référence ayant pour origine l'angle Sud-Est, du pixel correspondant. La fonction retourne -1 si le point géographique n'est pas visible par le satellite, -3 si les coordonnées géographiques en entrée sont incohérentes, 0 sinon.

4.5.18 Fonction `ligcol2tu_geo`

Prototype

```
float ligcol2tu_geo(float tu_img,int isat,int ires,float lig,float col,float tim_tot);
```

¹⁰Orientée positive vers l'Est

¹¹Orientée positive au Nord-Est

¹²Convention de signe identique à celle de `lon_prj`



isat \ ires	Résolution image	
	Haute	Basse
	0	1
0	5000	2500
1	11136	3712

TAB. 4.3 – Nombre de lignes et colonnes de la grille de référence en fonction du type de l'instrument (variable `isat`) et de la résolution de l'image (variable `ires`).

Description

Cette fonction retourne la valeur réelle correspondant à l'heure d'acquisition d'un pixel, connaissant les coordonnées images (`lig,col`) de celui-ci, ainsi que l'heure de fin d'acquisition de l'image `tu_img` et la durée totale d'acquisition de l'image `tim_tot`. Ces deux dernières informations sont récupérées dans l'en-tête principale. On fait l'hypothèse ici de la linéarité du temps d'acquisition d'un pixel en fonction de sa position sur la grille.

4.5.19 Fonction `calc_grid`

Prototype

```
void calc_grid(int isat,int ires,float *nblig,float *nbcol);
```

Description

Cette fonction calcule le nombre de lignes `nblig` et de colonnes `nbcol` de la grille de référence, en fonction du type *MFG* (`isat` = 0) ou *MSG* (`isat` = 1) de l'instrument, et de la résolution spatiale de l'image. Le tableau 4.3 ci-dessous recense les différents cas possibles.

4.5.20 Fonction `elevation_geo`

Prototype

```
float elevation_geo(float lat,float lon,float lon_prj);
```

Description

La fonction `elevation_geo` renvoie dans le cas nominal la valeur en degrés de l'angle d'élevation¹³ du satellite, une valeur négative sinon. Pour effectuer cette opération, la fonction doit connaître la latitude `lat`, orientée positive vers le Nord, et la longitude `lon` orientée positive vers l'Est, du point de la surface terrestre choisi, ainsi que la longitude du point sub-satellite `lon_prj`. Les trois arguments de la fonction sont en degrés.

¹³Angle qui est par définition complémentaire de l'angle zénithal de visée

4.5.21 Fonction `azimut_geo`

Prototype

```
float azimut_geo(float lat,float lon);
```

Description

La fonction `azimut_geo` renvoie dans le cas nominal l'angle azimutal de visée du satellite en degrés, une valeur négative sinon. Pour effectuer cela, la fonction doit connaître la latitude `lat`, orientée positive vers le Nord, ainsi que la longitude `lon` orientée positive vers l'Est, de la cible terrestre. Les deux arguments de la fonction sont en degrés.

4.5.22 Fonction `ordre_octets.c`

Prototype

```
int ordre_octets(void);
```

Description

Le but de cette fonction est de déterminer simplement la façon dont sont ordonnés les octets d'information en mémoire sur la machine support du traitement. La valeur retournée est 0 si la machine range les octets en mémoire à la manière `BIG ENDIAN`, est 1 si la machine range les octets en mémoire à la manière `LITTLE ENDIAN`.

4.5.23 Fonction `possol`

Prototype

```
int possol(int *jj,float *tu,float *xlon,float *xlat,float *asol,float *phi0);
```

Description

Fonction permettant de calculer les angles azimutal solaire `phi0` et zénithal solaire `asol`¹⁴, pour un jour `jday` dans le calendrier Julien, pour un point du globe terrestre défini par ses latitude `xlat` et longitude `xlon`¹⁵, et pour une heure `tu` *GMT*¹⁶ quelconque.

4.5.24 Fonction `swap_float`

Prototype

```
int swap_float__(float *x);
```

Description

La fonction `swap_float` permet de retourner les octets d'un objet de type `float` en mémoire. La représentation *BIG ENDIAN* d'un arrangement d'un groupe d'octets en mémoire signifie que l'octet le plus significatif est localisé à l'adresse la plus petite, l'octet le moins significatif est localisé à l'adresse la plus grande. Dans la représentation *LITTLE ENDIAN*, c'est la convention inverse qui s'applique, l'octet le plus significatif étant localisé à l'adresse la plus grande. Ainsi, sur le schéma 4.1 suivant, l'octet `n` est plus significatif que l'octet `n+1` dans la

¹⁴Tous deux en degrés.

¹⁵Également en degrés.

¹⁶Heure décimale



représentation *BIG ENDIAN*. On rappelle à l'utilisateur que les arrangements d'octets sont rangés par convention *LITTLE ENDIAN* pour les ordinateurs de type INTEL/PC, DEC, VAX ; la convention *BIG ENDIAN* est adoptée pour les machines SGI, IBM, Macintosh.

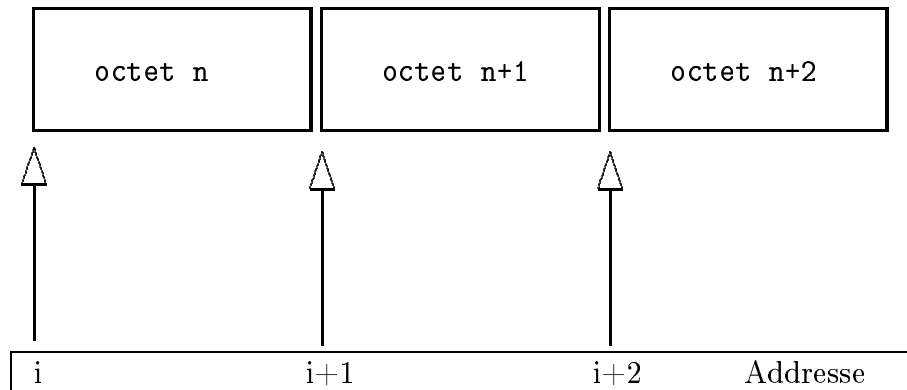


FIG. 4.1 – Schéma de l'arrangement d'un groupe d'octets en mémoire.

On choisit délibérément de perdre en généricité de la fonction, en n'utilisant pas le type `void` défini en *C*. Il en résulte une augmentation du nombre de fonctions, mais une plus grande robustesse du code lors du passage d'une variable à l'une de ces fonctions.

4.5.25 Fonction `swap_float_buf`

Prototype

```
int swap_float_buf(float *x,int *siz);
```

Description

La fonction `swap_float_buf` permet de retourner les octets d'un tableau d'objets de type `float` `x`, ayant `siz` éléments en mémoire.

4.5.26 Fonction `swap_double`

Prototype

```
int swap_double(double *x);
```

Description

Idem que la fonction `swap_float` décrite au paragraphe 4.5.24 pour un objet de type `double`.

4.5.27 Fonction `swap_double_buf`

Prototype

```
int swap_double_buf(double *x,int *siz);
```

Description

Idem que la fonction `swap_float_buf` décrite au paragraphe 4.5.25 pour un tableau d'objets de type `double`.

4.5.28 Fonction swap_int

Prototype

```
int swap_int(int *x);
```

Description

Idem que la fonction `swap_float` décrite au paragraphe 4.5.24 pour un objet de type `int`.

4.5.29 Fonction swap_int_buf

Prototype

```
int swap_int_buf(int *x,int *siz);
```

Description

Idem que la fonction `swap_float_buf` décrite au paragraphe 4.5.25 pour un tableau d'objets de type `int`.

4.5.30 Fonction swap_uint

Prototype

```
int swap_uint(uint *x);
```

Description

Idem que la fonction `swap_float` décrite au paragraphe 4.5.24 pour un objet de type `uint`.

4.5.31 Fonction swap_uint_buf

Prototype

```
int swap_uint_buf(uint *x,int *siz);
```

Description

Idem que la fonction `swap_float_buf` décrite au paragraphe 4.5.25 pour un tableau d'objets de type `uint`.

4.5.32 Fonction swap_long

Prototype

```
int swap_long(long *x);
```

Description

Idem que la fonction `swap_float` décrite au paragraphe 4.5.24 pour un objet de type `long`.

4.5.33 Fonction swap_long_buf

Prototype

```
int swap_long_buf(long *x,int *siz);
```

Description

Idem que la fonction `swap_float_buf` décrite au paragraphe 4.5.25 pour un tableau d'objets de type `long`.



4.5.34 Fonction `swap_short`

Prototype

```
int swap_short(short *x);
```

Description

Idem que la fonction `swap_float` décrite au paragraphe 4.5.24 pour un objet de type `short`.

4.5.35 Fonction `swap_short_buf`

Prototype

```
int swap_short_buf(short *x,int *siz);
```

Description

Idem que la fonction `swap_float_buf` décrite au paragraphe 4.5.25 pour un tableau d'objets de type `short`.

4.5.36 Fonction `swap_ushort`

Prototype

```
int swap_ushort(ushort *x);
```

Description

Idem que la fonction `swap_float` décrite au paragraphe 4.5.24 pour un objet de type `ushort`.

4.5.37 Fonction `swap_ushort_buf`

Prototype

```
int swap_ushort_buf(ushort *x,int *siz);
```

Description

Idem que la fonction `swap_float_buf` décrite au paragraphe 4.5.25 pour un tableau d'objets de type `ushort`.

4.6 Liste des fonctions du répertoire `mfg`.

Fichier source	Fonction	Section
<code>mfg_read.c</code>	<code>mfg_read</code>	??
<code>read_image_mfg.c</code>	<code>read_image_mfg</code>	??
<code>swap_header_mfg.c</code>	<code>swap_header_mfg</code>	??

swap_image_header_mfg.c	swap_image_header_mfg	??
write_log_mfg	write_log_mfg	??

TAB. 4.4: Description sommaire des fonctions du répertoire mfg

4.7 Liste des fonctions du répertoire msg.

Fichier source	Fonction	Section
msg_read.c	msg_read	4.7.1
read_bsqa_data.c	read_bsqa_data	4.7.2
read_nat_data.c	read_nat_data	4.7.3
swap_gp_pk_msg.c	swap_gp_pk_msg	4.7.4
swap_header_msg.c	swap_header_msg	4.7.5
swap_trailer_msg.c	swap_trailer_msg	4.7.6
tenbit2short.c	tenbit2short	4.7.7
write_log_msg.c	write_log_msg	4.7.8

TAB. 4.5: Description sommaire des fonctions du répertoire com

4.7.1 Fonction msg_read

Prototype

```
int msg_read(char *file_in, char *dir_out, const int ichan_out[]);
```

Description



4.7.2 Fonction `read_bsq_data`

Prototype

```
int read_bsq_data(FILE *fsi, const char *fileout, const int NbLines, const int ichan_out[], FILE *flog);
```

Description

Cette fonction lit les enregistrements d'un fichier de type *MSG* au format *BSQ*, et écrit les `NbLines` des images brutes correspondants aux canaux spectraux spécifiés par l'utilisateur dans le tableau `ichan_out`, en conservant l'ordre d'acquisition des lignes et colonnes de l'image initiale ainsi que les différents enregistrements de l'en-tête et de la remorque dans des fichiers distincts.

4.7.3 Fonction `read_nat_data`

Prototype

```
int read_nat_data(FILE *fsi, const char *fileout, const int NbLines, const int ichan_out[], FILE *flog);
```

Description

Cette fonction lit les enregistrements d'un fichier de type *MSG* au format *NATIVE*, et écrit les `NbLines` des images brutes correspondants aux canaux spectraux spécifiés par l'utilisateur dans le tableau `ichan_out`, en conservant l'ordre d'acquisition des lignes et colonnes de l'image initiale ainsi que les différents enregistrements de l'en-tête et de la remorque dans des fichiers distincts.

4.7.4 Fonction `swap_gp_pk_msg`

Prototype

```
int swap_gp_pk_msg(GP_PK_HEADER *gp_pk_hd, GP_PK_SH1 *gp_pk_sh1, GP_PK_SH2 *gp_pk_sh2);
```

Description

4.7.5 Fonction `swap_header_msg`

Prototype

```
int swap_header_msg(header_msg *header);
```

Description

4.7.6 Fonction `swap_trailer_msg`

Prototype

```
int swap_trailer_msg(trailer_msg *trailer);
```

Description

4.7.7 Fonction `tenbit2short`

Prototype

```
void tenbit2short(int nb, unsigned char *ca, unsigned short *sh);
```

Description

La fonction `tenbit2short` permet la conversion d'un tableau `ca` de `nb` pixels de type *MSG*, dans lequel les comptes numériques de chacun des pixels sont codés sur 10 bits, en un tableau dans lequel les comptes numériques sont codés sur 16 bits, soit 2 octets. Pour réaliser cette opération, on lit une suite de 5 `unsigned char` correspondant à 4 champs de 10 bits, pour les transformer en 4 `unsigned short int`. Pour cela, on applique différents masques, et on utilise les opérateurs de décalage de bit `<<` et `>>`. On réitère cette lecture autant de fois que nécessaire.

4.7.8 Fonction `write_log_msg`

Prototype

```
int write_log_msg(FILE *flog, int fileformat, nat_main_prod_header *nat_main_prod_hd,  
                 bsq_main_prod_header *bsq_main_prod_hd,  
                 secd_prod_header *secd_prod_hd, GP_PK_HEADER *gp_pk_hd,  
                 GP_PK_SH1 *gp_pk_sh1, GP_PK_SH2 *gp_pk_sh2,  
                 header_msg *header, trailer_msg *trailer);
```

Description

Cette fonction écrit une sortie formatée dirigée vers le fichier pointé par le flux `flog` des enregistrements importants des structures `nat_main_prod_hd` ou `bsq_main_prod_hd` suivant le type de fichier en entrée donné par l'indice `fileformat`, `secd_prod_hd`, `gp_pk_hd`, `gp_pk_sh1`, `gp_pk_sh2`, `header` et `trailer`.

4.8 Contrôle des erreurs

Afin de permettre un contrôle rapide et efficace des erreurs d'exécution du code, un fichier formaté (`$METrac/met/inc/erreur.meteosat`) recense tous les messages d'erreur rencontrés lors de l'exécution du code. De plus, des corrections peuvent être apportées à ce fichier sans qu'il soit nécessaire de recompiler le code. En outre, cette manière de procéder permet d'associer de manière biunivoque un code d'erreur à un message.

Toute ligne valide de ce fichier doit comporter un nombre composé de 5 chiffres, suivi du message d'erreur associé à ce numéro. Seuls les 256 premiers caractères du message seront décodés. Toute autre ligne n'est pas considérée, et peut être utilisée à des fins de commentaires du fichier, à condition que le premier caractère de la ligne soit différent d'un chiffre. Cette manière de procéder permet à l'utilisateur d'adapter, d'ajouter ou de supprimer facilement des messages d'erreur.

La recherche, dans le code source, de la ligne où s'est produite l'erreur `YYYYY` est facilitée par l'inclusion de commentaires du type `! E-YYYYY17`, par l'auteur, et devrait être maintenue, lors de développements ultérieurs du code.

¹⁷E comme Erreur



Chapitre 5

Principe physique de l'algorithme.

5.1 Introduction.

Nous décrivons brièvement ici le principe d'inversion des mesures dans deux canaux *MSG* pour retrouver l'épaisseur optique et le coefficient d'Angström des aérosols. Nous donnons en premier lieu quelques précisions concernant les modèles d'aérosols retenus.

5.2 Sélection des modèles d'aérosols.

Les propriétés optiques des aérosols au dessus des bassins Atlantique et Méditerranéen peuvent varier largement en fonction de la distribution en taille et de l'indice de réfraction des particules, lesquelles à leur tour dépendent de la source et nature des aérosols. Le coefficient d'Angström α est une propriété optique qui peut être utilisé comme approchant la distribution en taille. α est calculé à partir de la variation spectrale de τ entre deux longueurs d'ondes et devient grand (jusqu'à 2.5) pour les très petits types d'aérosols comme les sulfates des régions polluées et faible (0) pour les types d'aérosols contenant des grosses particules comme les poussières désertiques. Nous définirons ici α entre $0.635 \mu\text{m}$ et $0.810 \mu\text{m}$. Les mesures autant *in situ* que satellitaires en Méditerranée ont montré que le coefficient d'Angström calculé dans le visible varie grossièrement entre 0 et 2 [Holben et al., 2001], [Jamet, 2004]. 15 modèles d'aérosols avec un coefficient d'Angström allant de -0.1 à 2.4 ont été utilisé dans cette étude. Ceux-ci incluent 12 modèles développés pour la correction atmosphérique des images SeaWiFS [Gordon et Wang, 1994], [Mc Clain, et al., 2000]. Ces modèles ont une distribution en taille bimodale obtenus en faisant varier les proportions respectives des modes "tropospheric" (petites particules) et "oceanic" (grosses particules) ' [Shettle et Fenn, 1979], de même que l'humidité relative, laquelle a pour effet principal de changer le rayon moyen des deux modes élémentaires. Puisque cet ensemble de modèles couvre des coefficients d'Angström jusqu'à 1.5, il a été complété de 3 modèles additionnels de très petites particules dans le domaine 1.8 - 2.4 . La table 5.1 fournit des renseignements complémentaires au sujet de ces modèles.

5.3 Principe de l'algorithme en diffusion multiple.

Tant que la fonction de phase moléculaire est stable, le régime en diffusion multiple qui résulte du mélange des aérosols et des molécules est gouverné par la fonction de phase et l'albedo de diffusion simple des aérosols. L'effet des diffusions multiples sur le signal TOA est



Nom	Type	Humidité relative (%)	Contribution en nombre (petit mode %)	Contribution en nombre (gros mode %)	Coefficient d'Angström
O99	Oceanic	99	100	0	-0.09
M99	Maritime	99	1	99	0.08
C99	Coastal	99	0.5	99.5	0.21
M90	Maritime	90	1	99	0.22
C90	Coastal	90	0.5	99.5	0.42
M70	Maritime	70	1	99	0.43
M50	Maritime	50	1	99	0.51
C70	Coastal	70	0.5	99.5	0.66
C50	Coastal	50	0.5	99.5	0.78
T99	Tropospheric	99	0	100	1.29
T90	Tropospheric	90	0	100	1.49
T50	Tropospheric	50	0	100	1.61
W01	Tropospheric	–	0	100	1.79
W02	Tropospheric	–	0	100	1.97
W03	Tropospheric	–	0	100	2.25

TAB. 5.1 – Caractéristiques des modèles d'aérosols utilisés dans cette étude. Ceux-ci sont tous construits par mélange de distribution lognormales de modes "oceanic" et "tropospheric" d'après [Shettle et Fenn, 1979], à l'exception des modèles W01, W02 et W03. Ces trois modèles ont été obtenus à partir du mode "tropospheric" avec une humidité relative de 0% en diminuant le rayon moyen de la distribution de $0.027 \mu\text{m}$ (valeur d'origine) respectivement à $0.021 \mu\text{m}$, $0.017 \mu\text{m}$ et $0.015 \mu\text{m}$ respectivement. Il est à noter que le coefficient d'Angström est défini entre $0.635 \mu\text{m}$ et $0.810 \mu\text{m}$.

donc difficile à appréhender, précisément parce que les propriétés optiques des aérosols sont les inconnues du problème. Une solution revient à établir une relation non ambiguë entre un modèle d'aérosol donné et l'effet des diffusions multiples sur le signal TOA, lorsque ce modèle intervient. Pour faire l'hypothèse d'une telle relation, le modèle d'aérosol doit être auparavant déterminé.

- (1), Le rapport $\frac{\rho_{tot}}{\rho_{mol}}$ est calculée à $0.635 \mu m$ et $0.810 \mu m$, dans lequel ρ_{tot} est la réflectance TOA mesurée¹, et ρ_{mol} est la réflectance TOA d'une atmosphère sans aérosols, précalculée et stockée.
- (2), Comme les effets en diffusion multiple sont dépendants du modèle d'aérosol, plusieurs valeurs de $\tau_a(810)$ sont possibles avec la valeur de $\frac{\rho_{tot}}{\rho_{mol}}$ à $0.810 \mu m$, et chacune d'elles correspond à un modèle d'aérosol donné.
- (3), Le jeu de valeurs de $\tau_a(810)$ est alors convertit à $0.635 \mu m$, en utilisant les coefficients d'extinctions spectraux σ_a^{ext} de chacun des modèles d'aérosols.
- (4), À ces valeurs de $\tau_a(635)$ correspondent plusieurs valeurs du rapport $\frac{\rho_{tot}}{\rho_{mol}}$ à $0.635 \mu m$, qui diffère d'après le modèle d'aérosol.
- (5), En comparant ces valeurs du rapport $\frac{\rho_{tot}}{\rho_{mol}}$ à $0.635 \mu m$ à la valeur précédemment calculée à l'étape (1) permet de sélectionner les deux modèles d'aérosols qui encadrent le mieux le rapport $\frac{\rho_{tot}}{\rho_{mol}}$.

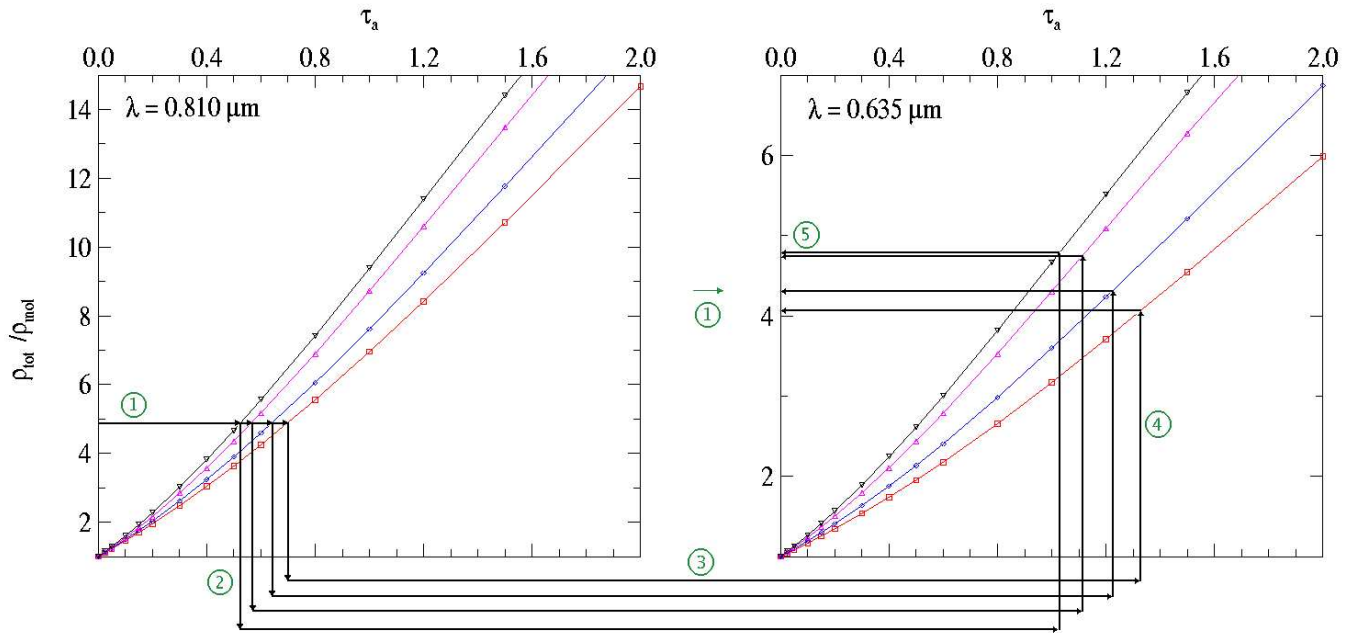


FIG. 5.1 – Variation de la réflectance simulée au sommet de l'atmosphère pour les canaux de l'instrument *Meteosat-08* centrés sur $0.635 \mu m$ et $0.810 \mu m$, en fonction de τ_a , et exprimée comme le rapport $\frac{\rho_{tot}}{\rho_{mol}}$, lorsque $\theta_s = 40.6^\circ$, $\theta_v = 29.4^\circ$ et $\Delta\phi = \frac{\pi}{2}$. Le modèle d'aérosol maritime est utilisé dans cet exemple, pour 4 valeurs de l'humidité relative (50%, 70%, 90%, et 99% de bas en haut). Les flèches indiquent un chemin possible pour identifier un couple de modèles d'aérosols encadrant au mieux la mesure. Les nombres encadrés en vert repèrent les étapes successives de l'algorithme d'inversion, décrites au paragraphe 5.3. .

¹Puisque l'on fait l'hypothèse ici que la réflectance marine est négligeable à ces deux longueurs d'ondes.



Chapitre 6

Perspectives.

6.1 Améliorations du code.

6.1.1 Calcul précis de l'absorption gazeuse

6.1.2 Utilisation du canal à $1.64 \mu\text{m}$ pour l'inversion des propriétés des aérosols.

6.1.3 Basculement vers un module d'inversion "poussières désertiques".

6.1.4 Lecture du format *HRIT/LRIT*



Annexe A

Exemple de script de lancement du code *meteosat*

Il est souhaitable de lancer le code *meteosat* au moyen d'un script¹ qui rassemble tous les paramètres du programme, puis les passe en arguments à l'exécutable concerné, et enfin lance cet exécutable. Un tel exemple de script est donné dans les pages qui suivent.

¹Ce script sera interprété ensuite par un shell, en l'occurrence celui qui est choisi ici est `/bin/bash`.



```
#!/bin/sh
#-----#
#          COMMISSARIAT A L'ENERGIE ATOMIQUE          #
#          LABORATOIRE DES SCIENCES DU CLIMAT ET DE L'ENVIRONNEMENT          #
#          91191 Gif-sur-Yvette FRANCE                #
#-----#
#
# $Id: test.sh,v 1.1.1.1 2005/09/01 08:57:23 thieul Exp $
#
#-----#
#
# BUT: Fichier de commande regroupant les valeurs des parametres de la simulation pour le code meteosat.
#
# AUTEUR: Francois Thieuleux
#
# NOTE: Pour utiliser ce shell(sh) , seule la variable d'environnement suivante doit etre definie :
#
#       METrac = chemin absolu du repertoire contenant le code meteosat
#
#-----#
METcod=$METrac/met
#-----#
#
# Parametres de simulation
#
#-----#

## Indice du satellite
## 5 : Meteosat-5
## 7 : Meteosat-7
## 8 : Meteosat-8 (MSG-1)
## 9 : Meteosat-9 (MSG-2)
## D'autres sont possibles...
##-----#
isat=8

## Type de donnees en entree du code
## 0 : Donnees contenues dans un fichier au format
##   - "OpenMTP" pour les donnees Meteosat (isat <= 7)
##   - "natif" ou "bsq" pour les donnees MSG (isat >= 8)
## 1 : Donnees contenues dans un (ou plusieurs) fichier(s) image(s) associes a un (ou plusieurs) fichier(s) en-tete.
##-----#
iraw=1

## Noms des repertoires contenant les fichiers utiles au traitement des images Meteosat/MSG et des donnees. Si l'argument du
## programme racine vaut -1, on considere les arguments suivants pour obtenir les noms des differents repertoires;
## Sinon, on suppose que l'argument contient le repertoire (absolu) racine des differents sous-repertoire utiles
## ( img, hdr, log, lst, msq, tab, ... )
##-----#
##racine="$METcod"
racine="-1"

## Si l'argument precedent vaut -1, les arguments suivants, qui representent le nom des repertoires contenant les fichiers utiles
## au traitement des images Meteosat/MSG, sont pris en compte.
##
## rep_raw : Repertoire (absolu) des fichiers natif
## rep_img : Repertoire (absolu) des fichiers images brutes (niveau 0)
## rep_hdr : Repertoire (absolu) des fichiers en-tete (binaire) associes aux images brutes
## rep_log : Repertoire (absolu) du fichier log du traitement
## rep_tmc : Repertoire (absolu) des fichiers masque Terre/Mer/Cote
## rep_lst : Repertoire (absolu) du fichier contenant la liste des fichiers image a traiter
## rep_lut : Repertoire (absolu) des fichiers des tables de couleur
## rep_tab : Repertoire (absolu) des fichiers precalcules
## rep_msq : Repertoire (absolu) des fichiers de masque
## rep_tau : Repertoire (absolu) des fichiers aerosols
## rep_gif : Repertoire (absolu) des fichiers de sortie au format GIF
## rep_tif : Repertoire (absolu) des fichiers de sortie au format TIF
## rep_jpg : Repertoire (absolu) des fichiers de sortie au format JPG
## rep_hdf : Repertoire (absolu) des fichiers de sortie au format HDF
##-----#
rep_raw="$WORK2/METarc"
rep_img="$WORK2/METout/image"
rep_hdr="$WORK2/METout/header_bin"
rep_log="$WORK1/METlog"
rep_tmc="$WORK1/METtmc"
rep_mie="$WORK1/METmie"
rep_lst="$WORK1/METlst"
rep_tab="$WORK1/METtab"
rep_lut="$WORK1/METlut"
rep_msq="$WORK1/METmsq"
rep_tau="$WORK1/METtau"
spare15="ArgumentNonUtilise"
spare16="ArgumentNonUtilise"
rep_gif="$WORK1/METgif"
rep_tif="$WORK1/METtif"
rep_jpg="$WORK1/METjpg"
rep_hdf="$WORK1/METHdf"
rep_rho="$WORK1/METrho"

## Fabrication des repertoires de sortie du code, s'ils n'existent pas deja.
##-----#
if [ ! -d $rep_log ]; then mkdir $rep_log; fi
```

```

if [ ! -d $rep_msq ]; then mkdir $rep_msq; fi
if [ ! -d $rep_tau ]; then mkdir $rep_tau; fi
if [ ! -d $rep_gif ]; then mkdir $rep_gif; fi
if [ ! -d $rep_tif ]; then mkdir $rep_tif; fi
if [ ! -d $rep_jpg ]; then mkdir $rep_jpg; fi
if [ ! -d $rep_hdf ]; then mkdir $rep_hdf; fi
if [ ! -d $rep_rho ]; then mkdir $rep_rho; fi

## Nom du fichier de masque utilise pour discriminer les pixels de type terre, mer, cote, et espace. Le fichier doit etre oriente
## de telle maniere que les coordonnees les plus petites correspondent au coin NW, les coordonnees les plus grandes correspondent
## au coin SE. Le fichier doit egalement avoir une taille de 5000x5000 pixels si les donnees a traiter sont de type MFG,
## de 3712x3712 pixels si les donnees a traiter sont de type MSG. Les valeurs du masque sont par convention codees sur 1 octet
## (8bits), dont les valeurs de codage sont les suivantes :
## - 10 : Mer
## - 20 : Cote
## - 30 : Terre
## - 0 : Espace
## Si fichtmc = -1, dans ce cas on considere le fichier de masque par default ( "MET-07.tmc" pour Meteosat-07,
## "MET-08.tmc" pour Meteosat-08 (MSG-1) ), etc... , situe dans le repertoire pointe par l'argument rep.tmc.
##-----
fichtmc="-1"

## Nom du fichier de dialogue ou sont diriges les commentaires du programme.
## Si fichlog = -1, dans ce cas le nom du fichier de log est construit a partir du nom de ce fichier de commande (+ extension .log),
## et se situe dans le repertoire pointe par l'argument rep.log .
##-----
fichlog="-1"

## Nom du fichier contenant la liste des fichiers images MFG ou MSG a traiter, ou encore nom d'un fichier image unique
## a traiter. Dans ce dernier cas, l'utilisateur peut specifier le nom absolu du fichier a traiter, ou bien le nom simple du
## fichier, le repertoire par default le contenant est supposee pointe par l'argument du programme racine_img. Si fichlst = -1,
## dans ce cas on considere les fichiers images dont la plage horaire est contenue dans plage, dont la date est comprise
## entre annee_deb/mois_deb/jour_deb et entre annee_fin/mois_fin/jour_fin;
##-----
fichlst="-1"

## Nom des fichiers des tables de couleurs utilisees pour la mise en forme des fichiers de la reflectance mesuree au sommet de l'
## l'atmosphere, de l'epaisseur optique aerosol inversee a 550 nm, du coefficient d'Angstrom. Ces fichiers sont supposes se trouver
## dans le repertoire pointe par l'argument rep_lut. Si fichlut_rho = -1, dans ce cas on considere la table de couleur par default
## ("GREY_bg=white.lut"). Si fichlut_tau = -1, dans ce cas on considere la table de couleur par default ("METEOSAT_bg=white.lut").
## Si fichlut_ang = -1, dans ce cas on considere la table de couleur par default ("METEOSAT_bg=white.lut"). Ces trois derniers
## fichiers se trouvent dans le repertoire pointe par l'argument rep_lut.
##-----
fichlut_rho="-1"
fichlut_tau="-1"
fichlut_ang="-1"

## Date de debut de traitement (ssi fichlst=-1)
## Date de fin de fin de traitement (ssi fichlst=-1)
## Le format de decodage est AAAA-MM-JJ, AAAA est l'annee, MM le mois, JJ le jour du mois
##-----
date_debut=2003-06-07
date_fin=2003-06-07

## Plage(s) horaire(s) a traiter par jour (ssi fichlst=-1). Specifier un nombre, ou une liste de nombres.
## (ex : 22-24, sera interprete comme 22, 23, et 24; ex : 20,24 sera interprete comme 20 et 24 ).
##-----
plage=45

## Indice correspondant a la repartition et au contenu des gaz absorbants (H2O,03) pour differents modeles d'atmosphere.
## Ces modeles sont au nombre de 6. Le cas sans absorption est egalement envisageable (0). Si la valeur est negative,
## le modele d'atmosphere retenu est US62.
## <0 : Valeur par default, modele US62
## 0 : pas d'absorption
## 1 : modele "Tropical" (TRO), u(H2O)=4.12g/cm2,u(O3)=.247cm-atm
## 2 : modele "Mid Latitude Summer" (MLS), u(H2O)=2.93g/cm2,u(O3)=.319cm-atm
## 3 : modele "Mid Latitude Winter" (MLW), u(H2O)=.853g/cm2,u(O3)=.395cm-atm
## 4 : modele "Sub Artic Summer" (SAS), u(H2O)=2.10g/cm2,u(O3)=.480cm-atm
## 5 : modele "Sub Artic Winter" (SAW), u(H2O)=.419g/cm2,u(O3)=.480cm-atm
## 6 : modele "US Standard 1962" (US62), u(H2O)=1.42g/cm2,u(O3)=.344cm-atm
##-----
iatm=6

## Valeur de la longueur d'onde (en microns) a laquelle on souhaite obtenir la valeur de l'epaisseur optique inversee.
## Par default (i.e. si cette valeur est negative), l'epaisseur optique en aerosol est inversee a 550 nm .
##-----
lambda=.865
#lambda=-1.

## Drapeau de la source des coefficients d'etalonnage a utiliser.
## Pour les instruments de type MFG, les informations concernant l'etalonnage radiometrique des mesures, fournies dans l'en-tete
## des images au format OpentMTP ne sont pas renseignees, elles le sont pour les instruments de type MSG dans l'en-tete des fichiers
## de niveau 1.5 (natif ou BSQ).
## Si ietal vaut 0, ces informations sont utilisees pour etalonner les comptes numeriques des images si elles sont disponibles.
## Si ietal vaut 1, on procede a l'etalonnage grace a des coefficients utilisateurs, avec derive temporelle possible (voir fonction
## f_etalonner.f).

```



Code de traitement des mesures pleine résolution *meteosat*
Inversion des paramètres optiques des aérosols sur mer
F. Thieuleux et al.

Juillet 2005 — Page 80 — *Revision* : 1.4



```
##-----
ietal=1
#ietal=0

## Drapeau du retournement de l'image. On veut que l'image soit orientee de telle maniere que les coordonnees les plus petites
## correspondent au coin NW, les coordonnees les plus grandes correspondent au coin SE. La valeur (1) retourne ainsi l'image
## d'origine, la valeur (0) laisse l'orientation de l'image identique a celle d'origine.
## Une connaissance de l'orientation de l'image d'origine est "a priori" necessaire.
##-----
retourne_coordonnees=1

## Indice du canal spectral sur lequel est base le filtre nuage. Pour les versions Meteosat anterieures a Meteosat-8 (MSG-1), un
## seul canal (VIS) est disponible.
## -1 : Par défaut, canal 2 (870 nm) pour Meteosat-8 et ses successeurs, canal 1 pour les autres plateformes.
##-----
ifltua=-1

## Drapeau du type de coordonnees de l'image a lire. Les coordonnees NW et SE de l'image peuvent etre entrees en valeurs de
## latitude,longitude (0), ou bien directement en coordonnees image(1); (-1) signifie un traitement global de l'image.
##-----
type_coord_image=-1

## Coordonnees de l'image a traiter
## ycoord_NW : coordonnees ligne/latitude du pixel NW de l'image
## xcoord_NW : coordonnees colonne/longitude du pixel NW de l'image
## ycoord_SE : coordonnees ligne/latitude du pixel SE de l'image
## xcoord_SE : coordonnees colonne/longitude du pixel SE de l'image
## Le choix est laisse a l'utilisateur d'entrer les coordonnees de
## l'image en valeurs de latitude/longitude, ou bien en ligne/colonne.
## Exemple : ycoord_NW=45. pour une latitude du pixel NW de 45 degres.
##           ycoord_NW=1000 pour la ligne
##-----
ycoord_NW=1
xcoord_NW=1
ycoord_SE=3712
xcoord_SE=3712

## Nombres de lignes du bandeau permettant de conserver les informations sur le voisinage d'un pixel lors du filtre nuageux.
## Ce nombre entier doit etre impair. On choisit de limiter sa valeur maximale a 9.
## -1 : Par défaut, le bandeau est constitue de 3 lignes successives.
##-----
nlig_bnd=3

## Type de l'inversion que l'on souhaite obtenir. Si l'on souhaite forcer une inversion avec un modele unique, on specifie ici
## son indice. Pour les images de type MFG, seule une inversion avec forçage du modele est possible (un seul canal spectral).
## La liste des modeles par ordre croissant d'indice sont les suivants (indice variant de 0 a 15) :
## 'B00','099','M99','C99','M90','C90','M70','M50','C70','C50','T99','T90','T50','W00','W01','W02' .
## Le modele 'B00' correspondant au modele integre par défaut dans 6S, "shettle model for background desert aerosol", est un modele
## de type aerosol desertique, qui peut servir a inverser les donnees de type MFG.
## Pour les images de type MSG, une inversion avec forçage du modele est possible, comme pour les images de type MFG. Une inversion
## multimodale est possible, lorsque l'utilisateur specifie une liste d'indices de modeles. Par exemple, 2-4, sera interprete comme
## 2, 3, et 4; 2,4,6-10 sera interprete comme 2, 4, 6, 7, 8, 9 et 10 . La valeur -1 signifie que l'on souhaite faire une inversion
## multimodale (obtention d'une epaisseur optique et d'un coefficient d'Angstrom associe), en utilisant tous les modeles disponibles.
## (hormis le modele B00).
##-----
#iaer=-1
iaer=0

## Indices des canaux spectraux sur lesquels se fait l'inversion de l'epaisseur optique aerosol, dans le cas de figure ou le modele
## d'aerosol n'est pas force (i.e. iaer=-1). Pour les instruments de type MFG, l'unique canal disponible est utilise.
## Pour les instruments de type MSG, deux des trois canaux spectraux (670, 870, ou 1360 nm) au choix sont possibles.
##-----
ilmb1=1
ilmb2=2

## Drapeau du type des coordonnees du point d'extraction. Les coordonnees de ce point peuvent etre entrees en valeurs de
## latitude,longitude (0), ou bien directement en coordonnees image (1).
##-----
type_coordonnees_extraction=1

## Coordonnees de latitude et longitude d'un pixel pour lequel on souhaite extraire les parametres aerosol inverses.
## Ces coordonnees peuvent etre entrees en valeurs de latitude,longitude, ou bien directement en coordonnees image.
##-----
# Sal Island
xcoord_extraction=1073
ycoord_extraction=1268
# Erdemli (Cote)
#xcoord_extraction=2768
#ycoord_extraction=680
# Porth Crete (Cote) 25.282W 35.33N
#xcoord_extraction=2567
#ycoord_extraction=686
# Nes Zion (Cote) 34.789W 31.922N
#xcoord_extraction=2822
#ycoord_extraction=808
# Acres 28.63E 38.53N
#xcoord_extraction=1093
```



```
#ycoord_extraction=611
# Dahkla 15.95W 23.717N
#xcoord_extraction=1279
#ycoord_extraction=1113

## Drapeau pour ecriture de commentaires du programme dans le fichier de dialogue. Plus la valeur du drapeau est elevee,
## plus le flot de commentaires rediriges vers le fichier fichlog est important.
##-----
dialogue=7

## Calcul des coefficients du fit polynomial d'ordre 3, de rho=f(tau)
## -1 : par defaut, calcul des coefficients a partir de la table precalculee de rho.
## 0 : lecture table precalculee des coefficients du fit polynomial d'ordre 3, de rho=f(tau)
## 1 : calcul des coefficients a partir de la table precalculee de rho
##-----
ifit=0
ifit=1

## Angle zenithal solaire (degres) maximal au dela duquel on ne procede plus a l'inversion des aerosols.
## Angle zenithal de visee (degres) maximal au dela duquel on ne procede plus a l'inversion des aerosols.
##-----
ths_lim=75.00
thv_lim=75.00

## Seuil sur la reflectance au sommet de l'atmosphere au dela duquel on considere qu'un pixel marin est nuageux.
##-----
SeuilReflectanceT0A=0.180

## Seuil sur l'ecart-type de la reflectance au sommet de l'atmpsphere
## d'un super-pixel nxn,
## au dela de laquelle on considere qu'un pixel marin est nuageux.
## (Basee sur l'hypothese de plus grande heterogeneite horizontale des
## nuages comparee aux panachees d'aerosols).
##-----
SeuilEtReflectanceT0A=0.0045

## Valeur limite de la vitesse de vent utilisee pour calculer la reflectance analytique du glitter par la formule de Cox et Munk.
##-----
vent_lim=8.00

## Valeur de la reflectance seuil pour le glitter. Lorsque la reflectance du glitter calculee par la formule de Cox et Munk
## depasse cette valeur seuil, on arrete le traitement du pixel considere.
##-----
rho_glt_lim=0.005

## Valeur de l'epaisseur optique aerosol inversee a 550 nm minimale en deca de laquelle le coefficient d'Angstrom n'est pas inverse.
##-----
tau550_min_angstrom=0.07

## Valeur du coefficient d'Angstrom imposee lorsque l'epaisseur optique aerosol inversee a 550 nm est inferieure a la valeur
## limite tau550_min_angstrom
##-----
angstrom_background=-0.08

## Valeur minimale et maximale de la reflectance mesuree au sommet de l'atmosphere a coder sur l'echelle 8 bits.
## Valeur minimale et maximale du coefficient d'Angstrom inverse sur la journee a coder sur l'echelle 8 bits.
## Valeur minimale et maximale de l'epaisseur optique aerosol inversee sur la journee a coder sur une echelle 8 bits.
## Valeur minimale et maximale du coefficient d'Angstrom inverse sur le mois a coder sur l'echelle 8 bits.
## Valeur minimale et maximale de l'epaisseur optique aerosol inversee sur le mois a coder sur une echelle 8 bits.
##-----
rho_mes_minmax=0.00:1.00
angstrom_jour_minmax=-0.10:3.00
tau550_jour_minmax=0.00:2.00
angstrom_mois_minmax=-0.10:3.00
tau550_mois_minmax=0.00:1.00

## Drapeau permettant de specifier la maniere dont les octets seront ranges en memoire, ceci concernant uniquement les fichiers
## binaires crees lors de l'execution du programme.
## -1 : Par defaut, les octets sont ranges en accord avec la representation de la machine (voir ci-dessous).
## 0 : octets ranges dans la representation BIG ENDIAN (voir ci-dessous).
## 1 : octets ranges dans la representation LITTLE ENDIAN (voir ci-dessous).
## La representation BIG ENDIAN d'un arrangement d'un groupe d'octets en memoire signifie que l'octet le plus significatif est
## localise a l'adresse la plus petite, l'octet le moins significatif est localise a l'adresse la plus grande. Dans la representation
## LITTLE ENDIAN, c'est la convention inverse qui s'applique, l'octet le plus significatif etant ocalise a l'adresse la plus grande.
## Ainsi, sur le schema suivant, l'octet n est plus significatif que l'octet n+1 dans la representation BIG ENDIAN.
## On rappelle a l'utilisateur que les arrangements d'octets sont ranges par convention 'LITTLE ENDIAN' pour les ordinateurs de type
## INTEL/PC, DEC, VAX; la convention 'BIG ENDIAN' est adoptee pour les machine SGI, IBM, Macintosh;
##-----
iordre=-1

## Drapeau pour le choix de la compression des fichiers d'entree et de sortie
## 0 : pas de compression, 1 : compression
##-----
icomp_in=0
```



```
icomp_out=0
```

```
## Lancement du programme meteosat
```

```
##-----  
if [ ! $dialogue = 0 ]  
then  
    echo " Execution meteosat "  
fi  
  
compil="$METcod/bin/meteosat \  
$isat \  
$siraw \  
$racine \  
$rep_raw $rep_img $rep_hdr $rep_log $rep_tmc $rep_mie $rep_lst \  
$rep_tab $rep_lut $rep_msq $rep_tau $spare15 $spare16 \  
$rep_gif $rep_tif $rep_jpg $rep_hdf $rep_rho \  
$fichtmc $fichlog $fichlst $fichlut_rho $fichlut_tau $fichlut_ang \  
$date_debut $date_fin $plage $iatm $lambda $ietal \  
$retourne_coordonnees \  
$ifltua \  
$type_coord_image $ycoord_NW $xcoord_NW $ycoord_SE $xcoord_SE \  
$nlig_bnd $iaer $ilmb1 $ilmb2 \  
$type_coordonnees_extraction $ycoord_extraction $xcoord_extraction \  
$dialogue \  
$ifit \  
$ths_lim $thv_lim \  
$SeuilReflectanceTOA $SeuilEtReflectanceTOA \  
$vent_lim $rho_glt_lim $tau550_min_angstrom $angstrom_background \  
$rho_mes_minmax $angstrom_jour_minmax $tau550_jour_minmax $angstrom_mois_minmax $tau550_mois_minmax \  
$iordre $icomp_in $icomp_out $0"  
if [ ! $dialogue = 0 ]  
then  
    echo $compil  
fi  
$compil  
  
## Si erreur lors du calcul, arret du code meteosat  
##-----  
if [ "$?" != "-0" ]  
then  
    echo " Detruisez les fichiers resultats"  
    echo " Arret du code meteosat "  
    exit 1  
fi  
## Fin du shell de lancement code meteosat  
##-----
```

Annexe B

Précisions sur le masque Terre/Mer/Côte.

Le masque Terre/Mer/Côte, appelé `tmc` dans le code, est utilisé pour définir le géotype d'un pixel de l'image. Puisque les images de type *MFG* et *MSG* sont distribuées par *Eumetsat* dans un format géolocalisé corrigé des effets géométriques, un fichier contenant le masque Terre/Mer/Côte a été créé pour chacun des instrument et résolution d'images employés. Ce fichier est unique pour chaque instrument et résolution, puisque les satellites dont il est question ici sont géostationnaires¹.

Les informations du masque sont stockées pour chaque pixel sur un 1 octet (`unsigned char`). Les différents fichiers de masque ont été réalisés à partir des fichiers de granules *MODIS* *MOD03*², et le tableau [B.1](#) ci-dessous recense les principales caractéristiques de ce masque.

Dans le code *meteosat*, on se contente pour l'instant de distinguer les pixels de type surface continentale, marine sans distinction de la profondeur de la colonne d'eau ou de l'éloignement à la côte, et accessoirement ceux qui sont de type côtier et espace.

¹Ceci est vrai tant que les images du satellite restent géolocalisées pour une même longitude sub-satellite.

²de type "*MODIS/Terra Geolocation Fields 5-Min L1A Swath 1km*"



Masque Terre/Mer Meteosat	
Valeur	Description
0	Océan peu profond (Océan à moins de 5 km du bord de côtes ou de moins de 50 m de profondeur).
1	Terre (et uniquement cela).
2	Bord de côtes océanique et lacustre.
3	Surface d'eaux à l'intérieur des terres peu profonde (moins de 5 km du rivage
4	Zone recouverte par les eaux de façon intermittente.
5	Surface d'eaux à l'intérieur des terres profonde (plus de 5 km du rivage et plus de 50 m de profondeur).
6	Océan peu profond ou plateau continental (au-delà de 5 km des côtes compris entre 50 m et 500m de profondeur.).
7	Océan profond (plus de 500m de profondeur).

TAB. B.1: Valeurs possibles du masque Terre/Mer/Côte

Annexe C

Génération des tables précalculées.

C.1 Introduction

Le code *meteosat* s'appuie sur des résultats de codes de transfert radiatif qui sont conservés dans des fichiers lus en entrée du code. Ces fichiers contiennent des tables précalculées de grandeurs physiques telles que :

- $\rho_{tot}^{mod}(\theta_s, \theta_v, \phi, \tau, \Delta\lambda)$, la réflectance totale au sommet de l'atmosphère pour différentes géométries de visée $(\theta_s, \theta_v, \phi)$, modèle d'aérosol (*mod*), quantité d'aérosol (τ) et bande spectrale $(\Delta\lambda)$,
- $\rho_{mol}(\theta_s, \theta_v, \phi, \Delta\lambda)$, la réflectance moléculaire pour différentes géométries de visée et bande spectrale,
- $t_{gaz}(\theta_s, \theta_v, k, \Delta\lambda)$, la transmission gazeuse¹ totale pour différents trajets atmosphériques, profils d'atmosphère k et bande spectrale,
- $r_F(\Theta_d)$, le coefficient de réflexion de Fresnel pour différents géométries angulaires (Θ_d) .

Nous essaierons ici de préciser les méthodes utilisées pour générer ces différentes tables, et nous ferons également une description de la manière dont celles-ci sont stockées.

C.2 Utilisation du code de transfert radiatif *6S*.

Plusieurs améliorations ont été apportées par rapport au code *5S*. Le code de transfert radiatif *6S* permet de calculer les propriétés optiques d'une cible à une altitude variable, de prendre en compte les transmissions gazeuses de nouveaux gaz absorbants ($\text{CH}_4, \text{CO}, \text{NO}_2$), de prendre en compte les conditions aux limites avec des surfaces non lambertiennes, d'améliorer encore la précision au regard de la diffusion par les aérosols et les molécules en utilisant la méthode des diffusions multipliss. Le pas utilisé pour l'intégration spectrale a été portée à 2.5 nanomètres.

Ce code permet à l'utilisateur de réaliser simplement une simulation du transfert radiatif dans l'atmosphère en spécifiant comme nous venons de le voir un grand nombre de propriétés de l'atmosphère et de la surface. Toutefois, ce code est adapté à des simulations au "cas par cas" : les résultats seront fournis à l'utilisateur dans la géométrie de visée demandée, et cette manière de procéder devient rhédibitoire lorsque l'on est amené à effectuer un nombre de simulations importantes. Nous avons donc été amené à modifier quelque peu le code *6S* de manière à récupérer pour un angle solaire donné, le champ de luminances au sommet de l'atmosphère.

¹Ozone et vapeur d'eau uniquement



Numéro du canal	1	2	3	4	5	6	7	8	9	10	11	12
Longueur d'onde [μm]	0.635	0.810	1.640	3.920	6.250	7.350	8.700	9.660	10.800	12.000	13.400	0.750

TAB. C.1 – Longueurs d'ondes équivalentes pour les bandes spectrales *MSG* utilisées dans le code.

De plus, nous avons intégré les informations concernant les bandes spectrales des instruments *Meteosat-07* (1 bande spectrale) et *Meteosat-08* (bandes spectrales 1,2,3 et 12).

C.3 Hypothèses de simulations.

C.3.1 Géométrie de visée.

Le code *6S* a été modifié de façon à pouvoir simuler le signal TOA pour un plus grand nombre d'angles zénithal solaire et zénithal de visée. Dans ce code, les calculs peuvent être obtenus désormais pour 48 valeurs des angles de Gauss entre 0° et 90°. La table C.2 suivante donne les valeurs de ces différents angles, ainsi que leurs poids associés. De même, la table C.3 contient les valeurs des 49 angles azimutaux relatifs entre 0° et 180°.

C.3.2 Conditions spectrales.

Les graphiques des figures C.1 page 87 et C.2 page 88 représentent les réponses spectrales des canaux 1,2,3 et 12 du capteur *METEOSAT-08* qui ont été incluses dans le code *6S* pour calculer de manière précise pour chaque canal spectral les propriétés d'absorption et de diffusion de l'atmosphère.

Nous donnons à titre indicatif dans le tableau C.1 les longueurs d'ondes équivalentes pour les bandes spectrales de *METEOSAT-08*.

C.3.3 Modèles d'aérosols.

Les propriétés de diffusion atmosphérique sont simulées en considérant qu'un seul modèle d'aérosol est présent, et que la distribution verticale de celui-ci suit une fonction de l'altitude telle que l'épaisseur optique aérosol soit de la forme :

$$\tau_a(z) = \tau_a e^{-z/H_a}$$

où H_a , l'échelle de hauteur des aérosols vaut 2 km.

C.3.4 Tables de $\rho_{tot}^{mod}(\theta_s, \theta_v, \phi, \tau, \Delta\lambda)$.

Pour rendre le code plus modulable, une table contenant la réflectance totale au sommet de l'atmosphère² a été générée pour chacun des modèles d'aérosols choisis. Ceci permet entre autres de substituer de manière plus aisée un modèle d'aérosol actuel par un nouveau modèle que l'on jugerait par la suite plus approprié.

²hors les effets d'absorption gazeuse vus au paragraphe C.3.5

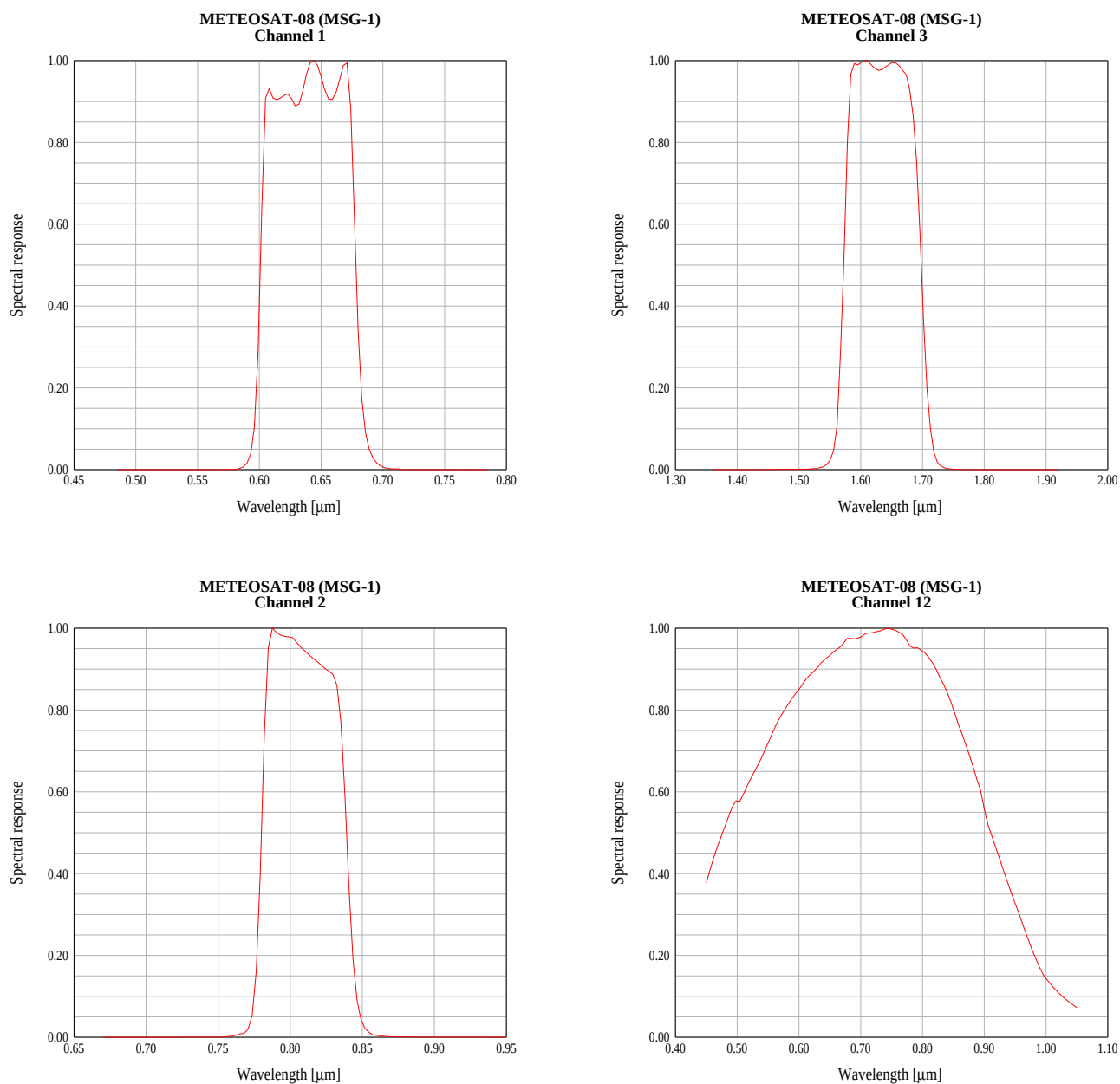


FIG. C.1 – Réponses des bandes spectrales 1,2,3 et 12 du capteur *METEOSAT-08* intégrées dans le code *6S*.

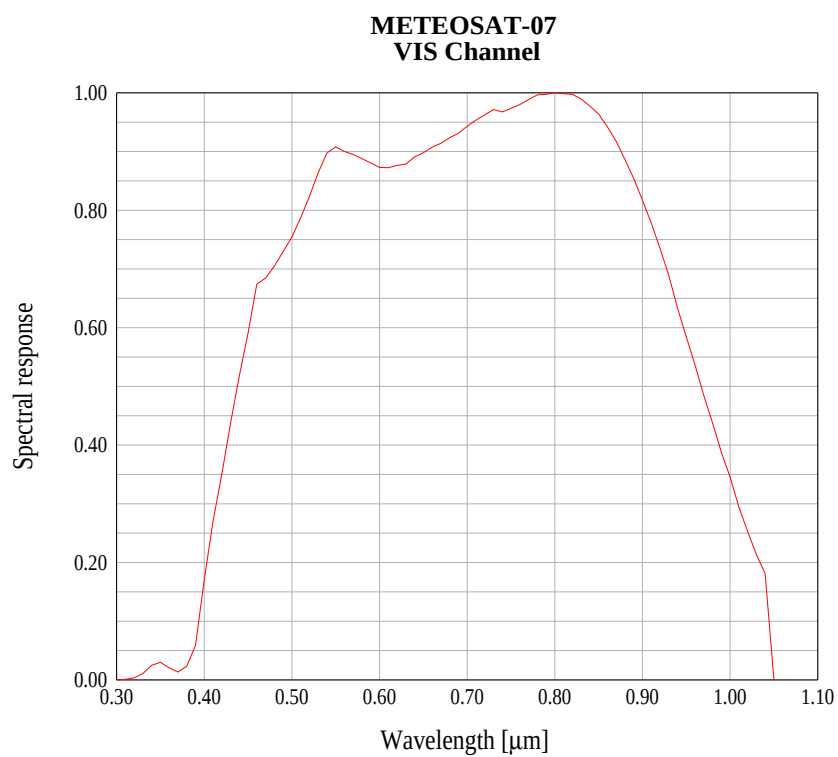


FIG. C.2 – Réponses de la bande visible du capteur *METEOSAT-07* intégrées dans le code *6S*.



C.3.5 Table de la transmission gazeuse $t_{gaz}(\theta_s, \theta_v, k, \Delta\lambda)$.



Indice	Angle [°]	Poids
1	1.427796	2.0356797E-02
2	3.277456	2.1172941E-02
3	5.138011	2.1966644E-02
4	7.001070	2.2737069E-02
5	8.865034	2.3483399E-02
6	10.72945	2.4204841E-02
7	12.59412	2.4900634E-02
8	14.45894	2.5570037E-02
9	16.32387	2.6212340E-02
10	18.18887	2.6826866E-02
11	20.05392	2.7412962E-02
12	21.91901	1.2151605E-02
13	23.78413	1.3128229E-02
14	25.64927	1.4090942E-02
15	27.51443	1.5038721E-02
16	29.37960	1.5970564E-02
17	31.24479	1.6885480E-02
18	33.10999	1.7782502E-02
19	34.97519	1.8660679E-02
20	36.84040	1.9519081E-02
21	38.70562	2.0356797E-02
22	40.57085	2.1172941E-02
23	42.43607	2.1966644E-02
24	44.30131	2.2737069E-02
25	46.16654	2.3483399E-02
26	48.03178	2.4204841E-02

Indice	Angle [°]	Poids
1	5.4748803E-02	2.4521174E-03
2	0.2884057	5.7055922E-03
3	0.7085180	8.9579662E-03
4	1.314743	1.2201460E-02
5	2.106462	1.5432372E-02
6	3.082856	1.8647287E-02
7	4.242914	2.1842854E-02
8	5.585430	2.5015749E-02
9	7.109012	2.8162677E-02
10	8.812078	3.1280369E-02
11	10.69286	3.4365591E-02
12	12.74940	3.7415132E-02
13	14.97958	4.0425837E-02
14	17.38106	4.3394569E-02
15	19.95136	4.6318255E-02
16	22.68782	4.9193859E-02
17	25.58759	5.2018389E-02
18	28.64766	5.4788921E-02
19	31.86485	5.7502571E-02
20	35.23583	6.0156528E-02
21	38.75709	6.2748037E-02
22	42.42498	6.5274402E-02
23	46.23569	6.7733005E-02
24	50.18527	7.0121288E-02
25	54.26961	7.2436780E-02
26	58.48447	7.4677072E-02

Annexe D

Exemple de résultats en sortie du code *meteosat*.

D.1 Introduction

Nous illustrerons ici quelques résultats obtenus à partir de mesures de différents instruments. Nous montrerons tout d'abord des cartes d'épaisseur optique aérosol inversée à 550 nm pour des instruments de type *MFG* (*Meteosat-05* et *Meteosat-07*). Nous rappelons ici une nouvelle fois que pour ce type d'instrument seul le canal large bande dans le visible peut être utilisé, ce qui oblige l'utilisateur à faire l'hypothèse *a priori* d'un modèle d'aérosol, et en définitive restreint l'algorithme à l'inversion d'une unique épaisseur optique à une longueur d'onde donnée ; Puis, nous montrerons les produits obtenus à partir de l'instrument *Meteosat-08* ; Pour ce type d'instrument, trois bandes spectrales peuvent être utilisées, et en plus de l'épaisseur optique aérosol, le coefficient d'Angström entre 0.635 μm et 0.810 μm associée peut également être obtenu. Nous illustrerons également ces mêmes produits en moyenne journalière (code *meteosat_day*) et mensuelle (code *meteosat_mth*).

Toutes les images illustrées ici ont été mise en forme en utilisant le logiciel *Msphinx* développé au LOA depuis de nombreuses années. Un simple fichier de commande permet ainsi de créer des images de manière sûre et efficace. Ce logiciel est disponible pour un nombre varié d'architectures ¹, et de plus il est librement téléchargeable à l'adresse suivante :

<http://www-loa.univ-lille1.fr/Msphinx/index.html> Une documentation complète de ce logiciel est disponible également à cette adresse.

D.2 Épaisseur optique aérosol, instrument *Meteosat-05*, 5 Septembre 1994.

La figure D.1 de la page 92 montre l'épaisseur optique aérosol inversée à 550 nm à l'échelle du globe. Le modèle d'aérosol choisi est de type "poussières désertiques", inclus dans le code 6S². On peut nettement distinguer un panache de poussières désertiques expulsé au large des côtes de Mauritanie, aux environs de Nouackchott. Vous pouvez noter également dans le golfe

¹Architecture/Système : PC Intel/Linux, Power PC/Linux, Power PC/OSX Darwin, Silicon Graphics/I-RIX64, Sun Solaris V5.7 et ultérieur, IBM RISC V4.3.2 et ultérieur. Hewlett Packard V10.2 et ultérieur

²Dans ce même code, le modèle d'aérosol correspond à l'indice *iaer*=5, il s'agit plus exactement du modèle de poussières désertiques décrit dans [Shettle, 1984].

de Guinée, la partie grisée où l'épaisseur optique aérosol n'est pas déterminée : cela est dû au masque des pixels qui sont contaminés par le glitter.

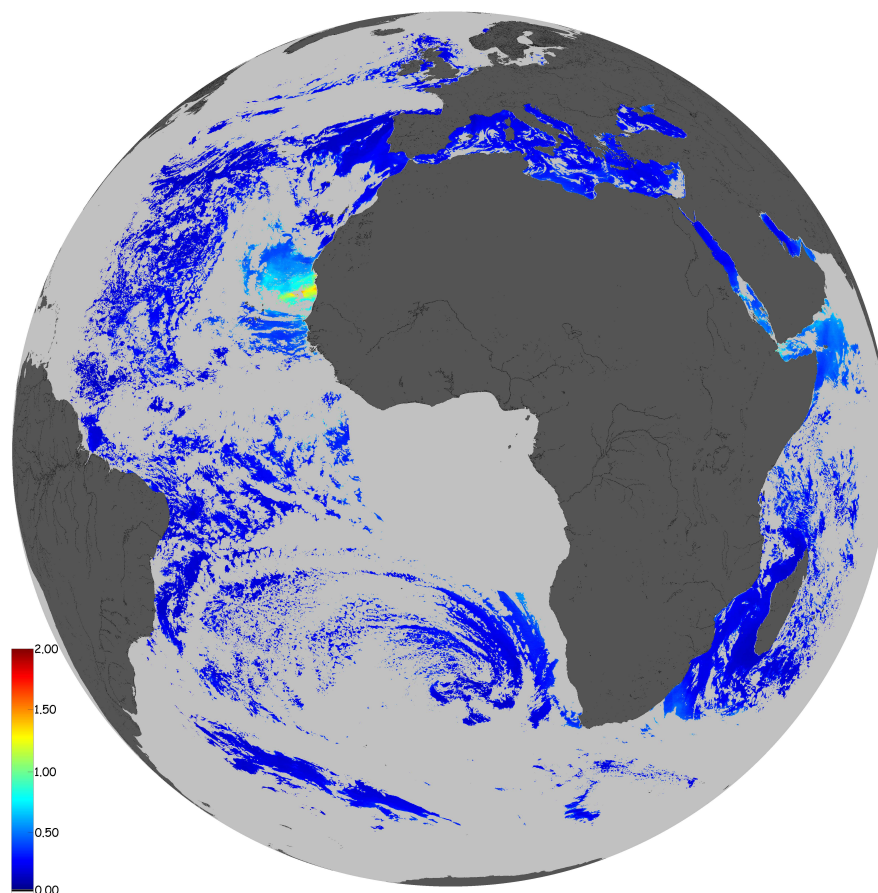


FIG. D.1 – Épaisseur optique aérosol inversée à 550 nm, à partir de l'image du 5 Septembre 1994 acquise par l'instrument *Meteosat-05* à 11h00 UTC.

D.3 Épaisseur optique aérosol, instrument *Meteosat-07*, 7 Juin 2003.

La figure D.2 suivante montre l'épaisseur optique aérosol à 550 nm inversée grâce à l'image acquise par l'instrument *Meteosat-07* le 7 Juin 2003. Le code *meteosat* permet actuellement l'inversion des propriétés optiques des aérosols des instruments *Meteosat-05* (lancé le 2 Mars 1991), et *Meteosat-07* (lancé le 2 Septembre 1997).

D.4 Épaisseur optique aérosol, instrument *Meteosat-07*, 7 Juin 2003.

Les figures suivantes de la page 95 illustrent l'épaisseur optique aérosol inversée à 550 nm, pour l'instrument *Meteosat-07*, pour différentes images acquises heure par heure, de 11h00 à

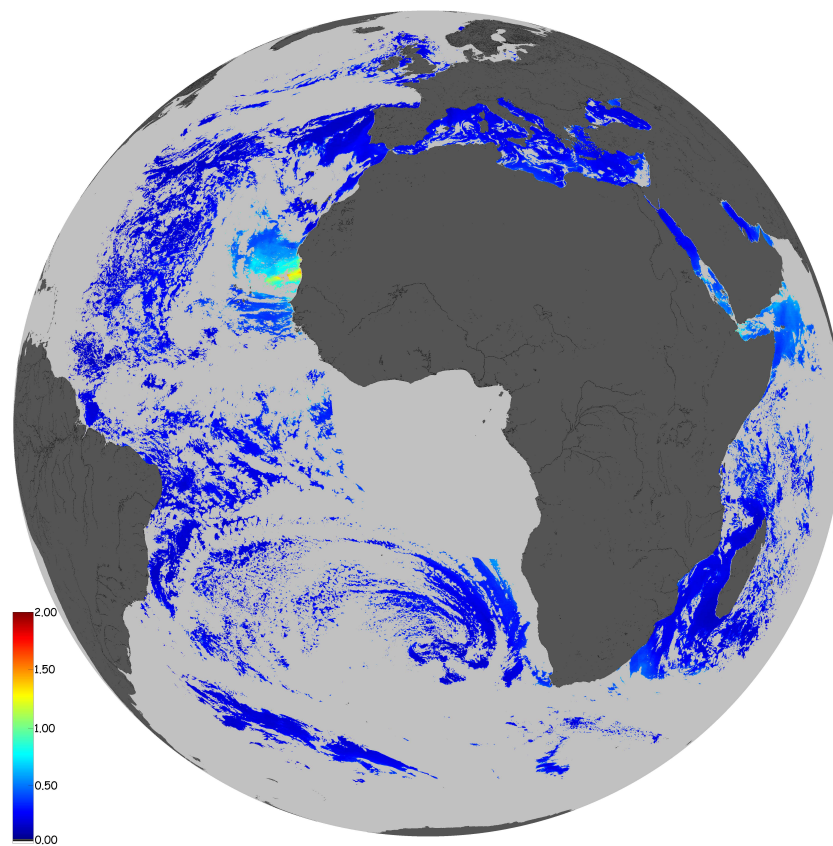


FIG. D.2 – Épaisseur optique aérosol inversée à 550 nm, à partir de l'image du 7 Juin 2003 acquise par l'instrument *Meteosat-07* à 11h00 UTC.



16h00 UTC. Cela met en évidence que la couverture spatiale du produit final augmente de façon significative lorsque l'on combine comme c'est le cas dans cet exemple les mesures de 9 images successives. L'utilisation de plusieurs images par jour conduit à une augmentation significative de la couverture spatiale pour trois raisons principales :

- Cela permet de s'affranchir du problème de la contamination des pixels par le glitter (localisée ici dans le Golf de Guinée pour les images successives),
- Cela permet de remplir une grande partie des zones nuageuses où la couverture des nuages demeure partielle,
- Enfin, cela permet de traiter des pixels qui ne sont pas suffisamment illuminés certaines heures de la journée.

La carte journalière de la figure D.4 page 96 obtenue en utilisant les 9 images a une remarquable couverture spatiale au regard de ce qui peut être obtenue par des capteurs ayant une résolution spatiale comparable telle que *MODIS* (1 km) ou encore *POLDER* (6 km).

D.5 Produits aérosol dérivés de l'instrument *Meteosat-08*, 7 Juin 2003.

Les figures D.5 de la page 96 présentent l'épaisseur optique aérosol (à gauche) inversée à 865 nm, ainsi que le coefficient d'Angström associé (à droite) à 11h15 UTC. On peut noter sur ces deux images la dissymétrie qui apparaît dans la forme du glitter : cela est dû au fait que le satellite était au cours du mois de Juin 2003 à une longitude (10 W) différente de celle utilisée pour la projection des images de niveau 1.5 *MSG*.

On peut noter sur ces deux images la dissymétrie qui apparaît dans la forme du glitter : cela est dû au fait que le satellite était au cours du mois de Juin 2003 à une longitude (10. W) différente de celle utilisée pour la projection des images de niveau 1.5 *MSG*.

D.6 Produits aérosol mensuel dérivés de l'instrument *Meteosat-08*, Juin 2003.

La moyenne mensuelle (tout comme la moyenne journalière) des produits aérosol est effectuée en donnant un poids égal, pour un pixel donné, à chacune des valeurs inversées de toutes les images du produit du mois en question. Il convient donc, avant de lancer le code *meteosat_mth*, de s'assurer que toutes les images disponibles pour le ou les mois désirés ont été traitées³.

³Toutefois, il est possible d'exclure certaines images de la moyenne mensuelle en ajustant convenablement les arguments *date_debut*, *date_fin*, et *plage* en entrée du programme *meteosat_mth*.

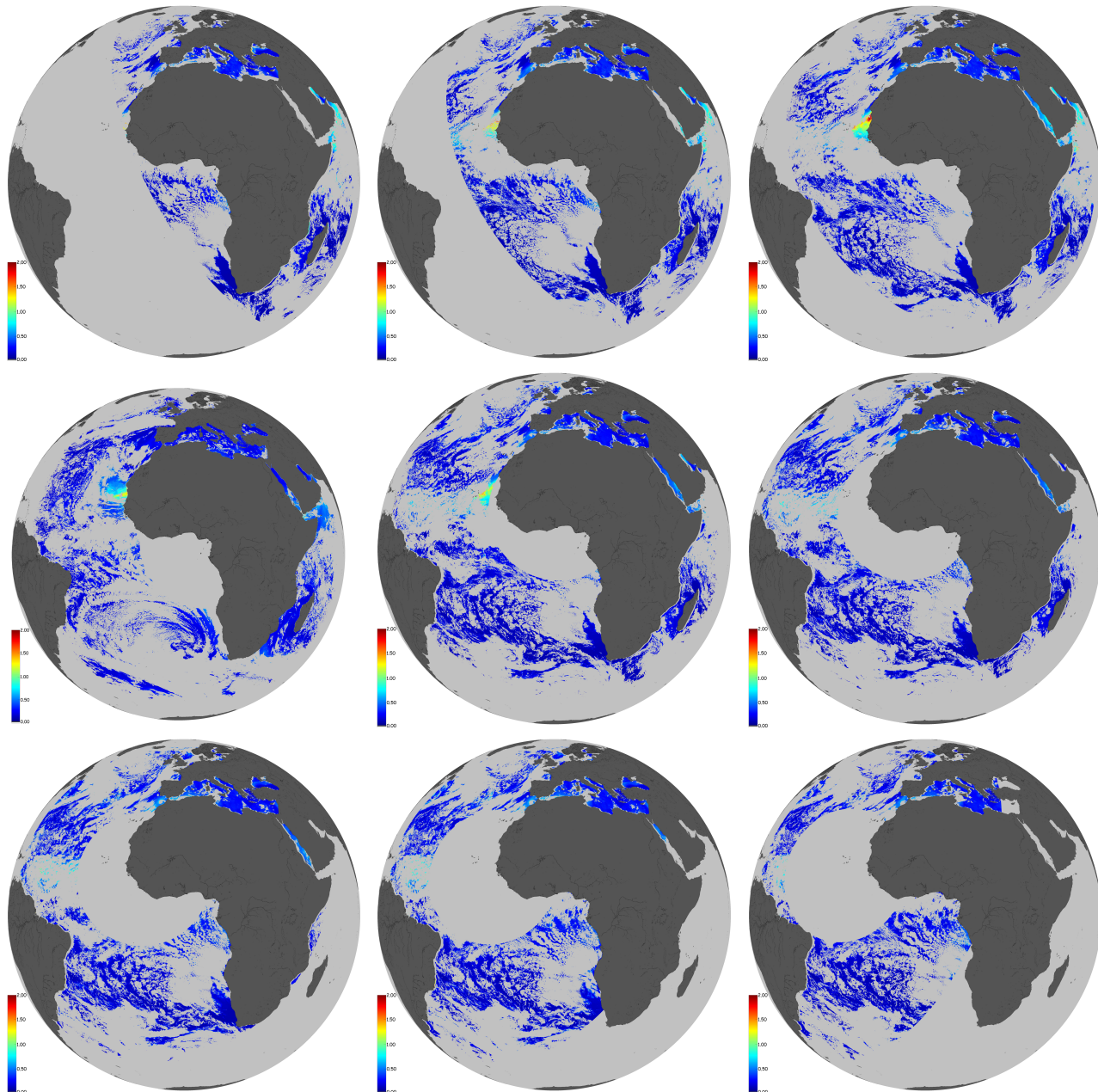


FIG. D.3 – Épaisseur optique aérosol inversée à 550 nm, pour l'instrument *Meteosat-07*, pour différentes images acquises heure par heure, de 11h00 UTC en haut à gauche, jusque 16h00 UTC en bas à droite.

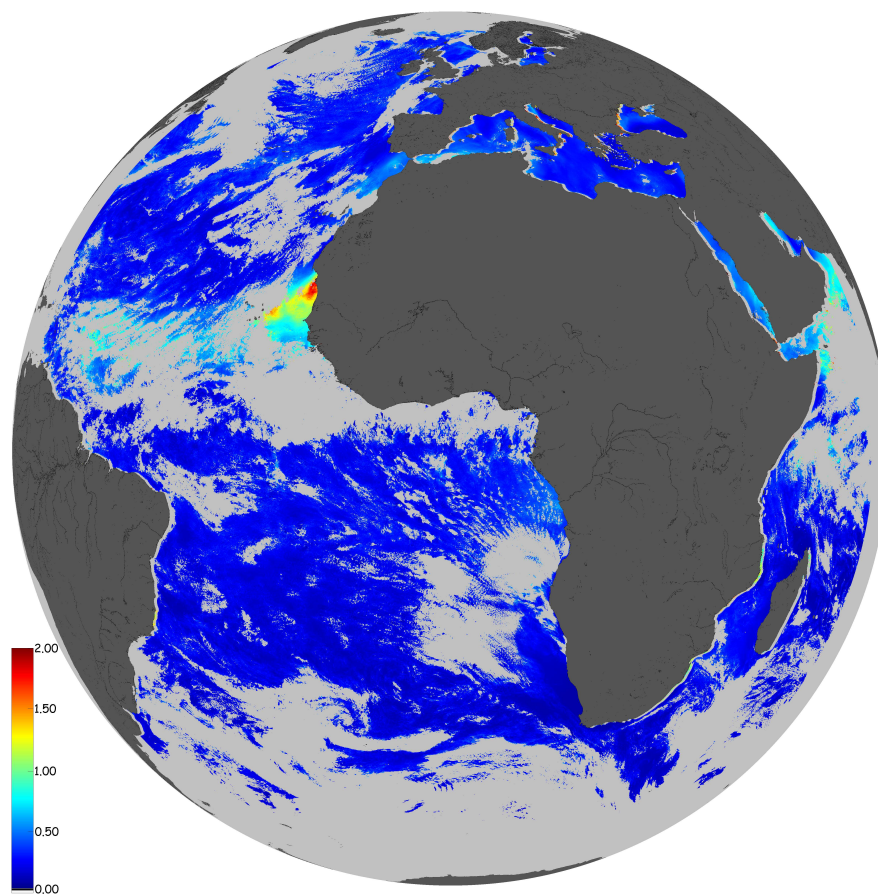


FIG. D.4 – Épaisseur optique aérosol inversée à 550 nm, à partir des 9 images précédentes.

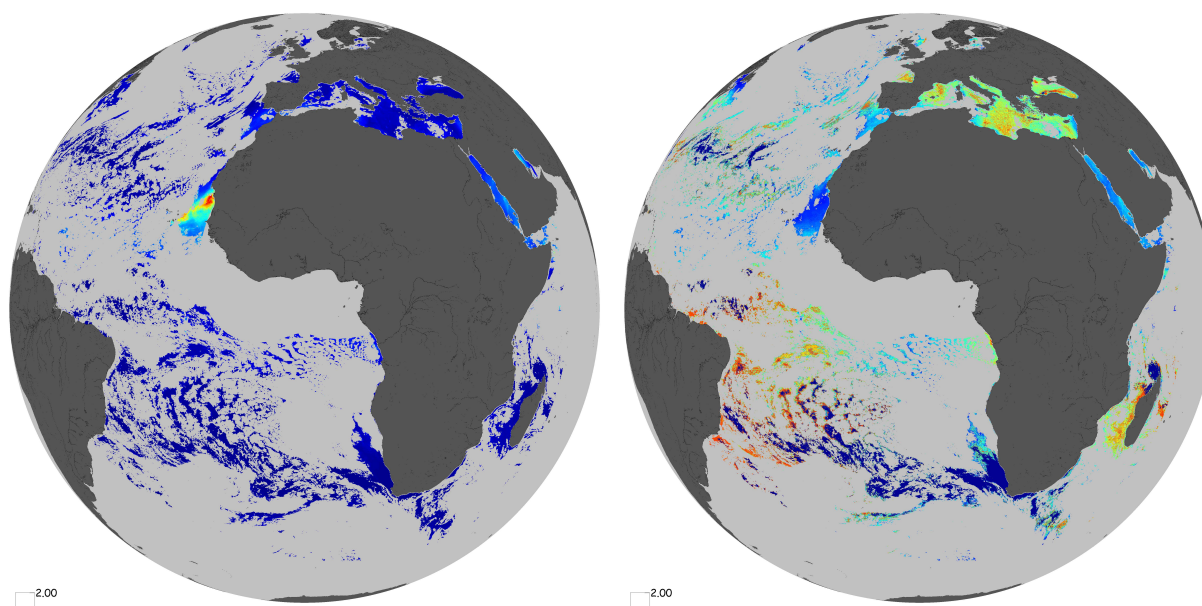


FIG. D.5 – Épaisseur optique aérosol (à gauche) inversée à 550 nm, pour l'instrument *Meteosat-08*, et coefficient d'Angström associé (à droite) à 11h15 UTC

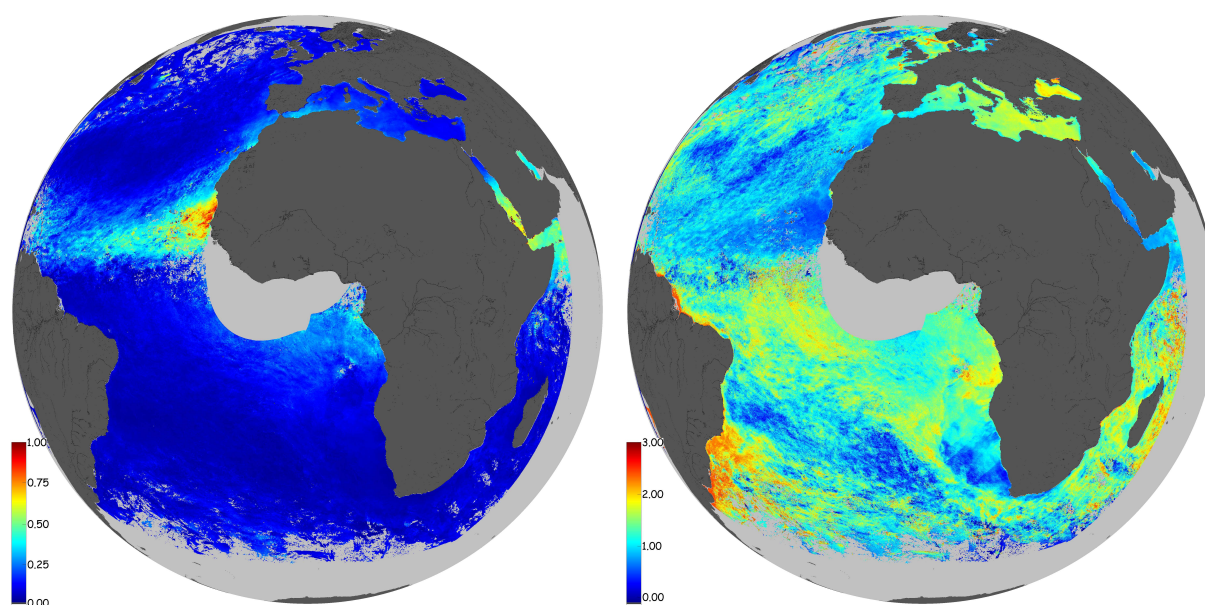


FIG. D.6 – Épaisseur optique aérosol (à gauche) inversée à 865 nm, pour l'instrument *Meteosat-08*, et coefficient d'Angström associé (à droite) pour le mois de Juin 2003.



Acronymes

<i>ANSI</i>	: American National Standards Institute
<i>ASCII</i>	: American Standard Code for Information and Interchange
<i>BSQ</i>	: Band SeQuential (format de données MSG)
<i>EUM</i>	: Eumetsat
<i>EUMETSAT</i>	: European Organisation for the Exploitation of Meteorological Satellites
<i>FTP</i>	: File Transfer Protocol
<i>GB</i>	: Gigabyte
<i>GIF</i>	: Graphic Interchange Format
<i>GMT</i>	: Greenwich Meridian Time
<i>GNU</i>	: Gnu is Not Unix
<i>HRIT</i>	: High Rate Information Transmission
<i>HRV</i>	: High Resolution Visible
<i>JPEG</i>	: Joint Photographic Experts Group
<i>IDL</i>	: Interface Description Language
<i>IR</i>	: Infra-Red (channel)
<i>LOA</i>	: Laboratoire d'Optique Atmosphérique
<i>LRIT</i>	: Low Rate Information Transmission
<i>LSCE</i>	: Laboratoire des Sciences du Climat et de l'Environnement
<i>MARF</i>	: Meteorological Archive and Retrieval Facility
<i>MFG</i>	: Meteosat First Generation
<i>MODIS</i>	: MODerate Resolution Image Spectroradiometer
<i>MPEF</i>	: Meteorological Products Extraction Facility
<i>MSG</i>	: Meteosat Second Generation
<i>MTG</i>	: Meteosat Third Generation
<i>PNG</i>	: Portable Network Graphics
<i>MTP</i>	: Meteosat Transition Program



RGB : Red Green Blue

SEVIRI : Spinning Enhanced Visible and Infra-Red Imager

TIFF : Tagged Image File Format

TOA : Top Of Atmosphere

VIS : Visible (channel)

WV : Water Vapor (channel)



Listes des symboles

Caractéristiques spectrales, géométrie angulaire

λ	:	Longueur d'onde	$[\mu\text{m}]$
θ_s	:	Angle zénithal solaire	degrés
θ_v	:	Angle zénithal de visée	degrés
ϕ_s	:	Angle azimutal solaire	degrés
ϕ_v	:	Angle azimutal de visée	degrés
Θ_d	:	Angle de diffusion	degrés
ϕ	:	Angle azimutal relatif	degrés

Atmosphère et propriétés des aérosols

$\sigma_a^{ext}(\lambda)$	:	Coefficient d'extinction spectral	
τ	:	Épaisseur optique totale de l'atmosphère	sans dimension
$\tau_m(\lambda)$	:	Épaisseur optique moléculaire	sans dimension
$\tau_a(\lambda)$	:	Épaisseur optique aérosol	sans dimension
$\tau_a^{diff}(\lambda)$	:	Épaisseur optique de diffusion des aérosols	sans dimension
$\tau_a^{abs}(\lambda)$	:	Épaisseur optique d'absorption des aérosols	sans dimension
ω_{0a}	:	Albedo de diffusion simple des aérosols	sans dimension
u_{O_3}	:	Quantité d'Ozone intégrée	cm-atm
u_{H_2O}	:	Quantité de vapeur d'eau intégrée	cm-atm

Interface air/eau

β	:	Angle entre la normale locale et la la normale d'une facette de vagues	sans dimension
$p_s(\theta_s, \theta_v, \phi)$	:	Densité de probabilité des pentes de vagues pour la géométrie angulaire $(\theta_s, \theta_v, \phi)$	sans dimension



- $r_F(\theta)$: Coefficient de réflexion de Fresnel pour l'angle d'incidence θ sans dimension
- $\rho_g(\theta_s, \theta_v, \phi)$: Réflectance analytique du glitter d'après le modèle de
[*Cox et Munk, 1954*]
- σ : Écart-type de la distribution des pentes de facettes des vagues sans dimension



Bibliographie

- Cox, C., and W. Munk, 1954 : "Measurements of the roughness of the sea surface from photographs of the sun's glitter", *J. Opt. Soc. Am.*, 11838-11850.
- Cox, C., and W. Munk, 1956 : "Slopes of the sea surface deduced from photographs of sun glitter", *Bull. Scripps Inst. Oceanogr.*, vol. 6, pp 401-488.,
- Govaerts, Y. M., 1999 : "Correction of the Meteosat-5 and -6 radiometer solar channel spectral response with the Meteosat-7 sensor spectral characteristics", *Int. J. Remote Sensing*, vol. 20, no. 18, 3677-3682.
- Gordon, H. R. and M. Wang, 1994 : "Retrieval of water-leaving radiance and aerosol optical thickness over the oceans with SeaWiFS : A preliminary algorithm". *App. Opt.*, vol. 33, 7754-7763.
- Gordon, H. R., 1997 : "Atmospheric correction of ocean color imagery in the Earth Observing System Area", *J. Geophys. Res.*, vol. 102, n° D14, 17081-17106.
- Holben, B. N., D. Tanré, A. Smirnov, T. F. Eck, I. Slutsker, N. Abuhassan, W. W. Newcomb, J. S. Schafer, B. Chatenet, F. Lavenue, Y. J. Kaufman, J. V. Castle, A. Setzer, B. Markham, D. Clark, R. Frouin, R. Halthore, A. Karneli, N. T. O'Neill, C. Pietras, R. T. Pinker, K. Voss, G. Zibordi, 2001 : An emerging ground-based aerosol climatology : Aerosol optical depth from AERONET, *J. Geophys. Res.*, vol. 106, 12067-12098, 10.1029/2001JD900014.
- Jamet, C., C. Moulin et S. Thiria, 2004 : Monitoring aerosol properties over the Mediterranean from SeaWiFS images using a neural network inversion, *Geophys. Res. Lett.*, vol. 31, L13107, doi :10.1029/2004GL019951.
- Lenoble, J., 1993 : "Atmospheric Radiative Transfer", *A. Deepak Publishing, Hampton, Virginia*, 532 p.
- Mc Clain, C. R. E. J. Ainsworth, R. A. Barnes, R. E. Eplee, Jr, F. S. Patt, W. D. Robinson, M. Wang, and S. W. Bailey, 2000 : SeaWiFS Postlaunch Calibration and Validation Analyses, Part 1., *NASA Tech. Memo 2000-206892*, vol. 9, S. B. Hooker and E. R. Firestone, Eds, NASA Goddard Space Flight Center, 82p. .
- Moulin, C., F. Guillard, F. Dulac, and C. E. Lambert, 1997 : "Long-term daily monitoring of Saharan dust load over ocean using Meteosat ISCCP-B2 data, 1, Methodology and preliminary results for 1983-1984 in the Mediterranean", *J. Geophys. Res.*, vol. 102, 16947-16958.
- Moulin, C., F. Dulac, C. E. Lambert, P. Chazette, I. Jankowiak, B. Chatenet, and F. Lavenue, 1997 : "Long-term daily monitoring of Saharan dust load over ocean using Meteosat ISCCP-B2 data, 2, Accuracy of the method and validation using sunphotometer measurements", *J. Geophys. Res.*, vol. 102, 16959-16969.
- Paltridge G.W., and C.M.R Platt, 1977 : "Radiative Processes in Meteorology and Climatology", *Development in Atmospheric Science*, Elsevier Scientific Publishing Company, New-York, N.Y. 10017.



Vermote, E. F., Tanré, D., Deuzé, J.-L., Herman, M., and Morcrette, J.-J., 1997 : "Second simulation of the satellite in the solar spectrum, 6S : An overview", *IEEE Transactions on Geoscience and Remote Sensing*, 35, 675-686.

Shettle, E. P., and R. W. Fenn, 1979 : "Models for the aerosols of the Lower Atmosphere and effects of Humidity Variations on Their Optical Properties", *Rep. AFGL-TR-79-0214*, U.S. Air Force Geophysics Laboratory, Hanscom Air Force Base, Mass.

Shettle, E. P., 1984 : "Optical and radiative properties of a desert aerosol model", *IRS'84, Proceedings of the International Radiation Symposium, Perugia, Italy*, 74-77.

Thieuleux, F., C. Moulin, F.-M. Bréon, F. Maignan, J. Poitou, and D. Tanré, 2005 : "Remote sensing of aerosols over the oceans using MSG/SEVIRI imagery", Submitted to *Ann. Geophys.*.

Documents de références

The Meteosat Archive, Doc. No. EUM/FG/1, Format Guide No. 1 - Basic imagery - OpenMTP Format, *EUMETSAT*, Revision 2.1, April 2000, 24 pages.

(<http://www.eumetsat.de/en/dps/archive/documentation/pdf/fg01.pdf>)

Meteosat Second Generation Level 1.5 Image Data Format Description, Doc. No. EUM/M-SG/ICD/105, *EUMETSAT*, Issue 2, November 2001, 212 pages.

(http://www.eumetsat.de/en/area4/msg/news/us_doc/icd_105_20.pdf)

Meteosat Second Generation Level 1.5 Image Data Format Description, Doc. No. EUM/M-SG/ICD/105, *EUMETSAT*, Issue 3, 4 February 2005, .

(http://www.eumetsat.de/en/area4/msg/news/us_doc/icd_105_30.pdf)

Meteosat Second Generation Level 1.5 Native Format File Definition, *EUMETSAT*, December 2002, 15 pages.

(http://www.eumetsat.de/en/dps/news/pdf/MSG_native_format_1_5.pdf)

MSG Level 1.5 BSQ Format File Definition, *EUMETSAT*, December 2002, 9 pages.

(http://www.eumetsat.de/en/dps/news/pdf/MSG_bsq_format_1_5.pdf)

Overview of the Meteosat Second Generation Ground Segment, Doc. No. EUM/MSG/TEM/093, *EUMETSAT*, Issue 1.1, September 1997, 20 pages.

(http://www.eumetsat.de/en/area4/msg/news/us_doc/ten_093_11.pdf)

MSG Ground Segment Design Specification (GSDS), Doc. No. MSG/SPE/055, *EUMETSAT*, Issue 1.3, 5 January 1999, 55 pages.

(http://www.eumetsat.de/en/area4/msg/news/us_doc/spe_055_13.pdf)

MSG Ground Segment LRIT/HIRT Mission Specific Implementation, DOc. N° EUM/MSG/SPE/057, *EUMETSAT*, Issue 4.1, 19 March 2001, 140 pages.

(http://www.eumetsat.de/en/area4/msg/news/us_doc/spe_057_41.pdf)

MSG Ground Segment LRIT/HIRT Mission Specific Implementation, DOc. N° EUM/MSG/SPE/057, *EUMETSAT*, Issue 5, 4 February 2005, 171 pages.

(http://www.eumetsat.de/en/area4/msg/news/us_doc/spe_057_5.pdf)

DADF Facility DADF Glossary, Doc. No. EUM/MSG/TEN/247, *EUMETSAT*, Issue 4.0, 23 August 1999, 15 pages.

(http://www.eumetsat.de/en/area4/msg/news/us_doc/ten_247_40.pdf)