RETURN BIDS TO:
RETOURNER LES SOUMISSIONS À:
Réception des soumissions - TPSGC / Bid Receiving - PWGSC
1550, Avenue d'Estimauville
1550, D'Estimauville Avenue
Québec
Québec
G1J 0C7

SOLICITATION AMENDMENT
MODIFICATION DE L'INVITATION

The referenced document is hereby revised; unless otherwise indicated, all other terms and conditions of the Solicitation remain the same.

Ce document est par la présente révisé; sauf indication contraire, les modalités de l'invitation demeurent les mêmes.

Comments - Commentaires

Vendor/Firm Name and Address
Raison sociale et adresse du fournisseur/de l'entrepreneur

Issuing Office - Bureau de distribution
TPSGC/PWGSC
601-1550, Avenue d'Estimauville
Québec
Québec
G1J 0C7

Title - Sujet Amélioration Préd.	
Solicitation No. - N° de l'invitation W7701-145784/A	**Amendment No. - N° modif.** 001
Client Reference No. - N° de référence du client W7701-14-5784	**Date** 2014-01-16
GETS Reference No. - N° de référence de SEAG PW-$QCN-015-15777	
File No. - N° de dossier QCN-3-36195 (015)	**CCC No./N° CCC - FMS No./N° VME**
Solicitation Closes - L'invitation prend fin **at - à 02:00 PM** **on - le 2014-01-24**	**Time Zone** **Fuseau horaire** Heure Normale du l'Est HNE
F.O.B. - F.A.B. **Plant-Usine:** ☐ **Destination:** ☑ **Other-Autre:** ☐	
Address Enquiries to: - Adresser toutes questions à: Legendre, Sylvie	**Buyer Id - Id de l'acheteur** qcn015
Telephone No. - N° de téléphone (418) 649-2860 ( )	**FAX No. - N° de FAX** (418) 648-2209
Destination - of Goods, Services, and Construction: **Destination - des biens, services et construction:**	

Instructions: See Herein

Instructions: Voir aux présentes

Delivery Required - Livraison exigée	**Delivery Offered - Livraison proposée**
Vendor/Firm Name and Address **Raison sociale et adresse du fournisseur/de l'entrepreneur** **Telephone No. - N° de téléphone** **Facsimile No. - N° de télécopieur**	
Name and title of person authorized to sign on behalf of Vendor/Firm (type or print) **Nom et titre de la personne autorisée à signer au nom du fournisseur/ de l'entrepreneur (taper ou écrire en caractères d'imprimerie)** **Signature** **Date**	

This amendment 001 to the request for proposal is to extend the solicitation period, answer questions from the industry, clarify the maximum budget, correct rated criteria, amend the financial bid preparation instructions, the financial bid presentation sheet and the basis of payment.

The request for proposal is amended as follows:

1) Extension of the solicitation period

DELETE : The solicitation ends on 2014-01-21

INSERT : **The solicitation ends on 2014-01-24**

2) QUESTIONS AND ANSWERS

QUESTION 1

Section 7 “Maximum Funding” of Part 2 - Bidder Instructions (page 10 of 68) does not specify if the amount is applicable to the firm tasks or to the tasks performed on an "as and when requested basis" using a Task Authorization or to both. The same is applicable to the amounts indicated at section 7.5 Funding by fiscal year of Part 7 Resulting Contract Clauses (page 25 of 68). Could you specify to which tasks the amounts are applicable, and if these amounts are applicable to both kinds of tasks, if the Bidder must respect a certain division between the budget for the firm tasks and the the tasks performed on an "as and when requested basis" using a Task Authorization? Also, it would be difficult to provide an accurate estimate of the work to be performed on an "as and when requested basis" using a Task Authorization as the “Description of services” at section 3.2 of Annex A - Statement of Work is short and mentions that “the deliverables and timelines will be specified once the tasks are assigned”; furthermore, we don’t know the quantity nor the span of the tasks authorizations that could be issued, not even the moment when the orders will be received.

ANSWER 1

Please refer to the enclosed amendments regarding the maximum funding, the financial proposal and the basis of payment.

QUESTION 2
At Annex A of the request for proposal, paragraph 2 “Applicable References”, we refer to 6 documents. Would it be possible to get a copy of the last 3 references (4), (5), and (6)? (page 51 of 68 of the french version)

ANSWER 2
Yes, please see the attached documents to this amendment.

QUESTION 3
Would it be possible to extend the solicitation period for an additional 10 days?

ANSWER 3
A 10 day extension is not possible, but the solicitation period is extended until January 24.

QUESTION 4
P.5. The RFP mentions that "the work of this contract must be performed at DRDC Valcartier". Are there tasks or parts of tasks that could be performed at the contractor location?

ANSWER 4
As mentioned in the contract, all code development work ‘’must be done at DRDC Valcartier, as it must be integrated to the EOSPEC’s SVN software development environment, which can only be accessed internally’’. In addition the lab characterization of sensors (Taks3A and TA-A) must also be performed in-house. (We do not want the sensors to be sent outside DRDC Valcartier). The accompanying planning and design work can be carried out externally, but this accounts for a very limited portion of the overall work (e.g. Task 3B1., 3B2, 3E1).

QUESTION 5
Does Canada accept the experience and expertise of subcontractors when the term "Bidder" is used?

ANSWER 5
No, this experience will not be considered, as detailed at section 4 - Definition of Bidder of the Standard instructions 2003 included in the request for proposal.

QUESTION 6
P. 36. Requiring more than 3 years of demonstrated experience with the "correlated-K approach" over the last 5 years is an (very) overly restrictive requirement in our opinion. Can Canada accept experience with similar state of the art atmospheric radiative transfer methods?

ANSWER 6
Any provider with recent relevant experience in atmospheric radiative transfer calculation is likely to be knowledgeable of the correlated-K approach. This approach is in the heart of most modern radiative transfer calculation methods and it is in particular employed in the core of EOSPEC/SMART, the DRDC Valcartier Library. I think that without a solid knowledge of the correlated-K approach, the provider cannot respond adequately to Task 3C2 and TA-D.

QUESTION 7
P. 41 and 42. Atmospheric science is not evolving at a very fast rate. Why restrict the date of publications to more recent than 10 years? Can Canada accept older publications?

ANSWER 7
No, because the main purpose here is to demonstrate recent experience in the field.

QUESTION 8
P. 42 and 43. Why restrict the date of presentation to more recent than 3 years? Can Canada accept older presentations?

ANSWER 8
Same point here. The purpose is to show recent work in the field.

QUESTION 9
P. 43. How is Canada going to verify that the provided sample reports have not been revised outside of the company?

ANSWER 9
Good point. This is why we expect the Bidders to put emphasis on internal software documentations. Excerpt of sample reports can provide a nice complement, as it may show the professionalism of the company in terms of final report production.

QUESTION 10
P. 43. Why does Canada request that the C++ and FORTRAN expertise reside in the same programmer? Can Canada separate the evaluation criterion so that more than one programmers be used to respond to this need?

ANSWER 10
It is not mandatory for the C++ and FORTRAN expertise to reside in the same programmer. This requirement is found under the Experience of the company.

QUESTION 11
P. 47. "The work shall be completed one month following the date when the sensors are made available, no later than the end of the first fiscal year." Which fiscal is this?

ANSWER 11
2014-2015.

3) At section 2 Summary of Part 1 - General Information

DELETE section (g) in its entirety
INSERT :

The maximum funding available for the contract resulting from the bid solicitation is $200,000.00 CAD for the firm portion of the work (Tasks under 3.1 of the statement of Work at Annex A) and is $300,000.00 CAD for the portion of the work to be performed on an as and when requested basis using a task authorization (tasks under 3.2 of the Statement of Work at Annex A) (applicable taxes extra). Bids valued in excess of this amount will be considered non-responsive. This disclosure does not commit Canada to pay the maximum funding available.

4) At section 7. Maximum funding of Part 2 - Bidder Instructions

DELETE in its entirety
INSERT :

The maximum funding available for the contract resulting from the bid solicitation is $200,000.00 CAD for the firm portion of the work (Tasks under 3.1 of the statement of Work at Annex A) and is $300,000.00 CAD for the portion of the work to be performed on an as and when requested basis using a task authorization (tasks under 3.2 of the Statement of Work at Annex A) (applicable taxes extra). Bids valued in excess of this amount will be considered non-responsive. This disclosure does not commit Canada to pay the maximum funding available.

5) At section II Financial Bid of Part 3 - Bid Preparation Instructions

DELETE in its entirety
INSERT :

Bidders must submit their financial bid in accordance with the following :

1.1 For the firm portion of the Work (section 3.1 of the Statement of Work in Annex A) :

(a) A Total Cost to a Limitation of Expenditure, which must not exceed the maximum funding specified in Part 2. The total amount of applicable taxes is to be shown separately, if applicable. The information should be provided in accordance with the Financial Bid Presentation Sheet at Attachment 1.

(b) Prices must be in Canadian funds, Canadian customs duties and excise taxes included, and applicable taxes excluded.

1.2 For the portion of the Work to be performed on an "as and when requested basis" using a Task Authorization (Section 3.2 of the Statement of Work in Annex A) :

(a) A firm all-inclusive hourly rate for each category of resources listed in attachment 1 - Financial Bid Presentation Sheet, for each year of the contract period. The information should be provided in accordance with the Financial Bid Presentation Sheet at Attachment 1.

(b) Prices must be in Canadian funds, Canadian customs duties and excise taxes included, and applicable taxes excluded.

6) At section 1.2 Financial Evaluation of Part 4 - Evaluation Procedures and Basis of Selection

DELETE in its entirety
INSERT :

1.2.1 Mandatory financial Criteria for the firm portion of the Work (section 3.1 of the Statement of Work in Annex A) :

The Bidder must submit a Basis of Payment to a Limitation of Expenditure that does not exceed the maximum funding specified at Part 2 of the Statement of work (Annex A), FOB Destination (for goods), all applicable customs duty and excise taxes included, but applicable taxes extra.

1.2.2 Mandatory financial Criteria for the portion of the Work to be performed on an "as and when requested basis" using a Task Authorization (Section 3. 2 of the Statement of Work in Annex A) :

The Bidder must submit a firm all-inclusive hourly rate for each category of resources listed in attachment 1 - Financial Bid Presentation Sheet, for each year of the contract period. FOB Destination (for goods), all applicable customs duty and excise taxes included, but applicable.

7) At section 7.5 Funding by fiscal year of Part 7 Resulting Contract clauses

DELETE in its entirety
INSERT :

7.5 Funding by fiscal year for the firm portion of the Work (Section 3.1 of the statement of Work in Annex A):

Despite the Total Estimated Cost (Limitation of Expenditure) specified in the Contract, and unless otherwise authorized in writing by the Contracting Authority, the maximum amount which may be paid for work completed, in the period ending March 31 of the year specified is as follows:

Period of award to March 31st 2014:	$ 20 000.00
Period of April 1st 2014 to March 31st 2015:	$ 180 000.00

8) At the group of rated criteria "1. Technical Proposal" in the Attachement 2 - Point Rated Techical Criteria

DELETE : Min : 52
INSERT : Min: 54

9) At the rated criteria 1.3 in the Attachement 2 - Point Rated Techical Criteria

DELETE : Min : 10, max : 6
INSERT : Min : 5, Max : 10

10) At the rated criteria 3.3 in the Attachement 2 - Point Rated Techical Criteria

DELETE : Min : 20, max : 8
INSERT : Min : 5, Max : 10

11) At the attachment 1 - Financial Bid Presentation Sheet, and at Annex B - Basis of Payment

DELETE in its entirety
INSERT :

1. LABOUR : Firm all-inclusive rates (including profit and overhead, applicable taxes extra), in accordance with the following:

Note to bidders :

- **Bidders must submit an all-inclusive firm hourly rate, per resource, and for each period.**
- **If the resource works for a subcontractor, the bidder should clearly indicate the name of the subcontractor.**

- **The proposed firm hourly rates are applicable to the firm portion of the work (Tasks under 3.1 of the statement of Work at Annex A) and to the portion of the work to be performed on an as and when requested basis using a task authorization (tasks under 3.2 of the Statement of Work at Annex A) for the periods between contract award to March 31, 2015. The proposed firm hourly rates for the period between April 1, 2015 and March 31, 2017, are only applicable to the portion of the work to be performed on an as and when requested basis using a task authorization (tasks under 3.2 of the Statement of Work at Annex A)**

- **The estimated number of hours is only applicable to the firm portion of the work (Tasks under 3.1 of the statement of Work at Annex A)**

- **The Bidder must propose the same resources for the work to be performed on an as and when requested basis using a task authorization (tasks under 3.2 of the Statement of Work at Annex A) and for the corresponding firm portion of the work (Tasks under 3.1 of the statement of Work at Annex A)**

Labour Categories and name of the proposed resources	**Firm All inclusive hourly rates**						
	Contract Periods						
	A From date of Contract to 2014-03-31	B Total Est. Number of hours	C From 2014-04-01 to 2015-03-31	D Total Est. Number of hours	**E** **Total Cost** **(A X B) + (C X D)**	F From 2015-04-01 to 2016-03-31	G From 2016-04-01 to 2017-03-31
Labour Category : Project Manager Name of the proposed resource: _____	$ _____ / hour	_____ hours	$ _____ / hour	_____ hours	$ _____	$ _____ / hour	$ _____ / hour
Labour Category: _____ Name of the proposed resource: _____	$ _____ / hour	_____ hours	$ _____ / hour	_____ hours	$ _____	$ _____ / hour	$ _____ / hour
Labour Category: _____ Name of the proposed resource: _____	$ _____ / hour	_____ hours	$ _____ / hour	_____ hours	$ _____	$ _____ / hour	$ _____ / hour

Labour Categories and name of the proposed resources	Firm All inclusive hourly rates						
	Contract Periods						
	A From date of Contract to 2014-03-31	B Total Est. Number of hours	C From 2014-04-01 to 2015-03-31	D Total Est. Number of hours	**E** **Total Cost** **(A X B) + (C X D)**	F From 2015-04-01 to 2016-03-31	G From 2016-04-01 to 2017-03-31
Labour Category: _____ Name of the proposed resource: _____	$ _____ / hour	_____ hours	$ _____ / hour	_____ hours	$ _____	$ _____ / hour	$ _____ / hour
Labour Category: _____ Name of the proposed resource: _____	$ _____ / hour	_____ hours	$ _____ / hour	_____ hours	$ _____	$ _____ / hour	$ _____ / hour
Labour Category: _____ Name of the proposed resource: _____	$ _____ / hour	_____ hours	$ _____ / hour	_____ hours	$ _____	$ _____ / hour	$ _____ / hour

TOTAL ESTIMATED LABOUR: $_____

Items 2 through 7 should be completed if the Bidder's rates in 1. above do not include the following:

If an item is not applicable, write "NA" on the total estimated line.

2. EQUIPMENT: at laid down cost without markup

Description	Firm unit price	Total est. Qty.
(a)	$	
(b)	$	
(c)	$	
(d)	$	

(e)	$	

TOTAL ESTIMATED EQUIPMENT: $ _____

3. RENTALS : at actual cost without markup

Description	Firm unit price	Total est. Qty.
(a)	$	
(b)	$	
(c)	$	
(d)	$	
(e)	$	

TOTAL ESTIMATED RENTALS: $ _____

4. MATERIALS AND SUPPLIES: at laid down cost without markup

Description	Firm unit price	Total est. Qty.
(a)	$	
(b)	$	
(c)	$	
(d)	$	
(e)	$	

TOTAL ESTIMATED MATERIALS AND SUPPLIES: $ _____

5. SUBCONTRACTS: at actual cost without markup

Support for the proposed subcontractor's price is required in the same details as that required for the Bidder's price. The estimated price for subcontracts should include all direct charges and travel & living expenses which would be to the account of the subcontractor.

Details are to be provided on a separate sheet.

The bidder can include the labour cost of the subcontracts in the table of section 1.

TOTAL ESTIMATED SUBCONTRACTS: $ _____

6. TRAVEL & LIVING: at actual cost without markup but not to exceed the limits of the Treasury Board Travel Directive. With respect to the TB Travel Directive, only the meal, private vehicle and incidental allowances specified in Appendices B, C and D of the TB Travel Directive http://www.tbs-sct.gc.ca/hr-rh/gtla-vgcl/ and the other provisions of the directive referring to "travellers" rather than those referring to "employees", are applicable.

Details are to be provided on a separate sheet.

TOTAL ESTIMATED TRAVEL & LIVING: $ _____

7. OTHER DIRECT CHARGES : at actual cost without markup

Description	Price
(a)	
(b)	
(c)	

TOTAL ESTIMATED OTHER DIRECT CHARGES: $ _____

TOTAL ESTIMATED COST TO A LIMITATION OF EXPENDITURE for the firm part :
$_____
(applicabe taxes extra)

TOTAL ESTIMATED COST TO A LIMITATION OF EXPENDITURE for the work done on request, by means of a task authorization $ 300,000.00
(applicabe taxes extra)

All other terms and conditions remain the same.

On the blending of surface layer and upper air profiles

Guy Potvin
Denis Dion
DRDC Valcartier

Gilles Fortin
Aerex Avionique

[Non-Standard Informative Statement text - if not required, delete this text]

Defence R&D Canada – Valcartier

Technical Report
DRDC TR [enter number only: 9999-999]

Commentaire [DRDC1]:
Title page
The title page provides information for description, bibliographic control and release. It must contain all the information that appears on the front cover, including security markings, warning terms and informative statements, but not copy numbers.

More information:
Users' Guide: Sections 2.4.1.1 and 4.4.
Standard: Sections 5.3.3, 6.6.2, 7.5.1 and Figures 6, 12 – 18.

Commentaire [DRDC2]:
Subtitle
Inclusion of a subtitle on the Executive Summary, Sommaire and DCD is at the discretion of the author.

More information:
Users' Guide: Section 4.4.1.
Standard: Sections 5.3.2, 5.3.6, 6.6.2 and 6.6.4.

Commentaire [DRDC3]:
Title page
It is very likely that some of the information on the title page will run off the bottom of the page and end up at the top of the Verso page.

See Section 4.4.2 of the **Users' Guide** for instructions on how to manually make everything fit on the title page.

Principal Author

Original signed by Guy Potvin

Guy Potvin

Defence Scientist / ESG

Approved by

Original signed by Alexandre Jouan

Alexandre Jouan

Section Head / SGE

Approved for release by

Original signed by Christian Carrier

Christian Carrier

Chief Scientist

[Include the sponsor of the work or a reference to a thrust or work unit, when deemed appropriate by author or CSA; relevant patent number(s), relating to protected intellectual property, should be noted. If there is no relevant information for this document, delete this text.]

Commentaire [DRDC4]:
"Original signed by"
Your original signature is only required on the final printed version of your document. For all other versions, including the electronic version which will be sent to DRDKIM, your name, with the words 'originally signed by", is to appear on the signature line.

More information:
Users' Guide: Section 4.5.1.
Standard: Sections 5.3.4, 6.6.3, and Figure 7.

Abstract

We expose a profile blending method that combines surface layer data, typically taken from a ground station or buoy, with boundary layer data such as that obtained from a radiosonde. The results are potential temperature and specific humidity profiles ranging from the ground up to the top of the upper layer profile. The surface data is used to create a profile using Monin-Obukhov Similarity theory. The M-O profile extends from the ground to the linking (or connection) height, which is included in the boundary layer profile aloft (which for a radiosonde typically begins at about 50 m AGL). Our method modifies the M-O profile in such a way that both its value and derivative with respect to height agrees with the boundary layer profile at the linking height, while keeping the deviation with respect to the M-O profile as small as possible by using a minimum action principle. Our method is essentially linear, but we also examine an exponential variant in order to avoid negative values in the blended profile (especially relevant for the specific humidity). We also examine ways to prevent the blended specific humidity profile from becoming supersaturated. Since the potential temperature and specific humidity combine to form the modified refractive index profile, we perform an error analysis to determine the effect of the blending method on the refractive index RF duct height perturbations. We then show a profile blending program designed using the principles explained in this report. We suggest possible improvements and conclude that the resulting software performs well for most realistic situations.

Résumé

Nous exposons ici une méthode de mélange de profil pour agencer des données météorologiques de la couche de surface atmosphérique, typiquement obtenues de stations terrestres ou de bouées, avec des données dans la couche limite planétaire telles qu'obtenues par radiosondage. Il en résulte des profils de température potentielle et d'humidité spécifique allant du sol jusqu'au haut du profil élevé. Les données de surface sont utilisées pour créer un profil vertical utilisant la théorie de similarité de Monin-Obukhov. Le profil M-O s'étend de la surface jusqu'à la hauteur de connexion, comprise dans le profil élevé (lequel profil débute typiquement à environ 50 m du sol). Notre méthode modifie le profil M-O de telle sorte que sa valeur et sa dérivée en fonction de la hauteur s'accorde avec le profil élevé à la hauteur de raccordement, tout en maintenant, à l'aide du principe d'action minimal, des déviations aussi faibles que possible par rapport au profil M-O original. Quoique notre méthode soit essentiellement linéaire, nous avons considéré aussi une forme exponentielle permettant d'éviter des valeurs négatives lors du mélange (ce qui est de mise principalement pour l'humidité spécifique). Nous avons aussi mis en place des façons d'empêcher que le profil d'humidité devienne sursaturé lors du mélange. Comme le profil de réfraction modifié est le produit de la combinaison des profils de température et d'humidité, nous avons réalisé une analyse d'erreur nous permettant de mesurer l'effet de la méthode sur les perturbations créées sur la hauteur du conduit RF. Enfin, nous présentons un logiciel effectuant les mélanges utilisant les principes présentés dans ce rapport. Nous suggérons des améliorations possibles à la méthode de mélange et concluons que le logiciel actuel offre d'assez bonne performance dans la plupart des situations réalistes.

Commentaire [DRDC5]:
Abstract/Résumé
The abstract should briefly describe:
- what was done and why (introduction);
- how it was done (methods);
- what was found (results); and
- what these results mean (discussion).

Requirements:
- must appear in both English and French;
- should not exceed 200 words;
- should be unclassified, whenever possible, for **classified/designated** documents;
- begins on a new, odd-numbered, right-hand page; and
- is numbered as page 'i'.

More information:
Users' Guide: Section 4.6.
Standard: Section 5.3.5.

Insert Abstract/Résumé Bookmarks:
The abstract and résumé text are the only sections of the DCD that will not automatically copy once you have completed filling out the Info Forms. The DRDC toolbar provides you with an easy way to copy your abstract and résumé text onto the DCD sheet – it does this by assigning bookmarks to the text you have keyed in.

More information:
Users' Guide: Section 4.6.2.

Commentaire [DRDC6]:
Résumé
If the abstract text is too long, the résumé will have to be placed on its own page – see Section 4.6.1 of the **Users' Guide** for instructions on how to do this.

TIP: Don't forget to use the Translation Info Form to keep track of your translation.

This page intentionally left blank.

Commentaire [DRDC7]:
Blank page
This page is unnecessary if the preceding page has an even page number.

More information:
Users' Guide: Section 4.20.
Standard: Section 6.6.8.

Executive summary

On the blending of surface layer and upper air profiles

Guy Potvin; Denis Dion; Gilles Fortin; DRDC TR [enter number only: 9999-999]; Defence R&D Canada – Valcartier; .

Introduction: The modern battlespace may require the detection of targets close to the ground and a large distance from EO and RADAR sensors that are also near the ground. The proper simulation or performance prediction of such a scenario requires unbroken profiles of the refractive index going from the ground to the upper atmosphere. This demands a method for combining surface data, such as meteorological measurements from a surface weather station or buoy, with upper air data, such as from a radiosonde or a numerical model. In this work, we propose such a method.

Results: We have devised a blending method where the surface data is used to create a profile of potential temperature or specific humidity using the well-known Monin-Obukhov (or M-O) similarity theory. This M-O profile extends up to the lowest point of the upper air profile, where there is often a discrepancy between the profiles. Our method uses physical principles to modify the M-O profile so that it matches the upper air profile at a specified linking height, thus creating a single blended profile extending from the ground to the upper atmosphere. We also performed an error analysis on the blending method and investigated ways of fusing data from various sources and locations. We then integrated this knowledge into a demonstrator program that produces blended profiles of potential temperature, specific humidity and refractive index based on user inputs and preferences. We found that the program produces good results in most realistic situations.

Significance: Our profile blending method fills a gap as no general purpose blending method can be found in the scientific literature. We have devised a method that is stable, reliable and physically motivated. It produces refractive index profiles that will be significant in the study of the propagation of radio, IR and visible waves through the atmosphere. This, in turn, will help in predicting sensor performance in various atmospheric conditions and operational situations.

Future plans: We intend to develop the profile blending program further to increase its robustness and to include it in the profile module of EOSPEC function library, where it can be used for the purpose of simulation studies of sensor performance. We also intend to develop a version of the program that can be used in an operational context for *in situ* prediction of sensor performance, in both maritime and land environments.

Commentaire [DRDC8]:
Executive Summary/*Sommaire*
Executive summary should describe:
- general problem being addressed, what led to the work being done;
- who did the work and indicate the effort and resources applied;
- facilities or bodies of background knowledge; and
- final results.

Requirements:
- must appear in both English and French;
- must normally be one page or less (in each language), but may be longer at the discretion of the author;
- must be able to stand alone because it may be distributed on its own or published with other executive summaries;
- should be unclassified, whenever possible, for **classified/designated** documents; and
- begins on a new, odd-numbered, right-hand page.

More information:
Users' Guide: Section 4.7.
Standard: Sections 5.3.6, 6.6.4, 7.4.1.1.2, 7.4.1.2.5 and Figure 8.

Commentaire [DRDC9]:
Subtitle
Inclusion of a subtitle on the executive summary and *sommaire* is at the discretion of the author.

More information:
Users' Guide: Section 4.7.1.

Commentaire [DRDC10]:
TIP: Include military significance, if any.

Sommaire

On the blending of surface layer and upper air profiles

Guy Potvin; Denis Dion ; Gilles Fortin; DRDC TR [enter number only: 9999-999] ;R & D pour la défense Canada – Valcartier; .

Introduction : Les opérations militaires requièrent le plus souvent le besoin de détecter des cibles près de la surface et à des distances appréciables des capteurs RADAR ou EO-IR qui sont aussi près de la surface. L'étude par simulation ou la prédiction des performances dans de tels scénarios nécessite le plus souvent la connaissance des profils de réfractivité de façon continue et lisse du sol jusqu'au haut de l'atmosphère. Ceci nécessite une méthode permettant de combiner les informations de surface, telles les données météorologiques provenant de stations à la surface ou de bouées, avec les données en élévation, telles qu'obtenues par radiosondage ou à partir de modèles numériques. Une telle méthode est présentée dans ce rapport.

Résultats : Nous avons mis au point une méthode de combinaison (ou de mélange) de données dans laquelle des données de surface sont utilisées pour générer des profils de température potentielle et d'humidité spécifique à l'aide de la théorie de similarité de Monin-Obukhov. Ces profils M-O s'étendent jusqu'à la partie inférieur du profil élevé, où des écarts subsistent le plus souvent entre les profils. Notre méthode utilise des principes physiques pour modifier le profil M-O afin qu'il s'harmonise au profil élevé à la hauteur de raccordement souhaitée, créant ainsi un profil mélangé unifié allant de la surface jusqu'au haut de l'atmosphère. Nous avons aussi réalisé une analyse d'erreur sur la méthode de mélange et étudié des façons de fusionner des données provenant de différentes sources. Nous avons intégré l'ensemble de ces connaissances dans un logiciel de démonstration qui réalise des mélanges de profils à partir des entrées et des options choisies par l'usager. Nous avons pu vérifier que le logiciel produit de bons résultats dans la plupart des situations d'étude réalistes.

Importance : Notre méthode de mélange de profil répond à un manque important car aucune méthode de mélange d'ordre générale n'est présentée dans la littérature scientifique. Nous avons mis au point une méthode qui est stable, fiable et construite sur des bases physiques. Elle permet de générer des profils atmosphériques et de réfractivité nécessaires aux études des effets de propagation atmosphérique dans différentes bandes de fréquence et de longueurs d'onde. Ceci contribue en outre à la mise au point de système de prédiction de portée en fonction des conditions atmosphériques et des conditions d'opération.

Perspectives : Nous prévoyons raffiner la technique davantage pour la rendre plus robuste et l'inclure au module de génération de profil de la librairie EOSPEC-LIB, pour un usage généralisé dans des études de simulation et de performance de système. Nous projetons aussi en développer une version qui pourrait être utilisée dans un contexte opérationnel de *prédiction in situ* de performance de capteur, dans des environnements maritimes ou terrestres.

Commentaire [DRDC11]:
Sommaire
Should describe: general problem being addressed; what led to the work being done; who did the work and indicate the effort and resources applied; facilities or bodies of background knowledge; and final results.

TIP: Use simple, non-technical language.

More information:
Users' Guide: Section 4.7.
Standard: Section 5.3.6.

Commentaire [DRDC12]:
TIP: Don't forget to use the Translation Info Form to keep track of your translation.

Commentaire [DRDC13]:
Sommaire
Document numbers are always in English, even when the principal language of the document is French.

Table of contents

Commentaire [DRDC14]:
Table of contents
A table of contents is mandatory for all formal client-oriented documents. It outlines the organisation and scope of the document.

TIP: Always use the DRDC toolbar to update the table of contents.

IMPORTANT REMINDER:
Don't forget to use 'hidden' markers to properly format the headers/footers when the table
of contents contains 'sensitive' headings – see Section 6.2.2.3.4
of the **Users' Guide**.

More information:
Users' Guide: Section 4.8.
Standard: Sections 5.3.7 and 7.5.1.

This page intentionally left blank.

Commentaire [D15]: Blank page
TIP: This page is unnecessary if the preceding page has an even page number. To remove the page, delete the entire sentence.

List of figures

Commentaire [DRDC16]:
List of figures
Required if the document contains more than five figures; optional otherwise. Can be combined with the list of tables if both fit on the same page.

TIP: Always use the DRDC toolbar to update the list of figures.

IMPORTANT REMINDER:
Don't forget to use 'hidden' markers to properly format the headers/footers when the listing contains 'sensitive' captions – see Section 6.2.2.3.4 of the **Users' Guide**.

More information:
Users' Guide: Sections 4.9 and 4.11.6.
Standard: Sections 5.3.8 and 7.5.1.

This page intentionally left blank.

Commentaire [DRDC17]:
Blank page
This page is unnecessary if the preceding page has an even page number.

WARNING: Be VERY careful not to delete the section break at the bottom of this page when removing this page.

More information:
Users' Guide: Section 4.20.
Standard: Section 6.6.8.

1 Introduction

Characterizing the atmosphere typically requires the combination of data from different sources, whether this is in an operational context or as part of a measurement trial. It is therefore necessary to find a way of combining the data into a coherent picture that takes into account the various characteristics, accuracies and reliabilities of the data sources. This is known as data fusion. In this work, we describe a method for blending surface layer data with boundary layer data so as to create a single, smooth potential temperature or specific humidity profiles from the ground to any arbitrary height. Since we wish to use this method for the creation of refractivity profiles, we must ensure that the thermodynamic profiles are continuous with respect to their values and to their first-order derivative.

In order to establish the physical principles behind our approach, we review atmospheric surface layer theory in Section 2. We then expose the theory of our profile blending method in Section 3 and examine issues relating to its error analysis in Section 4. We present a basic data fusion method in Section 5 in an attempt to better relate our method to real-world applications. We then examine a profile blending Demonstrator in Section 6 and address some practical issues in profile blending in Section 7. We conclude in Section 8.

Commentaire [DRDC18]:
First page of body
Must start on an odd-numbered, right-hand page.

More information:
Users' Guide: Section 4.11.
Standard: Sections 5.4, 6.2 – 6.6, 7.5.1 and Figures 9, 19 – 24.

Commentaire [DRDC19]:
Styles
Always use customised, built-in DRDC styles for formatting headings and text – see Section 3.7 of the **Users' Guide**.

This page intentionally left blank.

Commentaire [D20]: Blank page
TIP: This page is unnecessary if the preceding page has an even page number. To remove the page, delete the entire sentence.

2 Surface Layer Theory

Commentaire [DRDC21]:
Styles
Always use customised, built-in DRDC styles for formatting headings and text – see Section 3.7 of the **Users' Guide**.

In this section we review some of the major points regarding the turbulent surface layer. In particular, we review the turbulent budgets of the surface layer and the restrictions that lead to the Monin-Obukhov theory.

2.1 Turbulent Budgets

We shall adopt the approach by Garratt (1992), where we introduce the primitive equations using index notation, and then show the turbulent generalization of those equations. The primitive equations are fundamentally about the conservation of three quantities: mass, momentum and thermal energy.

2.1.1 The conservation of mass

The conservation of mass is expressed as

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x_j}\left(\rho v_j\right) = 0 \tag{1}$$

where ρ is the density of the moist air and v_j is the wind velocity field (where $j = 1, 2, 3$ and the index 3 represents the vertical direction, $x_3 = z$). Unless otherwise indicated, repeated indices imply a summation. Equation (1) can be written in an alternative form,

$$\frac{1}{\rho}\frac{d\rho}{dt} = -\frac{\partial v_j}{\partial x_j} \tag{2}$$

where $d/dt = \partial/\partial t + v_j\, \partial/\partial x_j$ is the material time derivative with respect to the instantaneous fluid flow. Compressibility effects can be neglected in the surface layer, which means that $d\rho/dt \approx 0$ and that

$$\frac{\partial v_j}{\partial x_j} \approx 0 \tag{3}$$

to a good approximation. In order to find the turbulent generalization, we use Reynolds convention to decompose the velocity field, $v_j = \bar{v}_j + v_j'$, where $\bar{v}_j$ is the mean with respect to the ensemble of turbulent fluctuations, and v_j' is the fluctuating component that averages to zero. Due to the linear nature of Eq (3), both the average and fluctuating velocity fields are incompressible.

$$\frac{\partial v_j}{\partial x_j} \approx 0, \quad \frac{\partial v_j'}{\partial x_j} \approx 0 \tag{4}$$

The other mass conservation that concerns us is that of the water vapour density, ρ_v. However, it more convenient to express water vapour in terms of the specific humidity, which is the ratio of the mass of water vapour per unit mass of moist air (Rogers and Yau, 1989), $q = \rho_v/\rho$. It obeys the following equation,

$$\frac{dq}{dt} = \kappa_v \frac{\partial^2 q}{\partial x_j \partial x_j} \tag{5}$$

where κ_v is the molecular diffusivity for water vapour in dry air, which we take to be a constant. For the average specific humidity field we obtain

$$\frac{D\bar{q}}{Dt} = -\frac{\partial}{\partial x_j}\left(\overline{v_j' q'}\right) + \kappa_v \frac{\partial^2 \bar{q}}{\partial x_j \partial x_j} \tag{6}$$

where $D/Dt = \partial/\partial t + \bar{v}_j \, \partial/\partial x_j$ is the material time derivative with respect to the average fluid flow, not the instantaneous fluid flow. The quantity in the parentheses of the first term in the right-hand-side of Eq (6) is the turbulent flux of the average specific humidity, which the covariance between the velocity and specific humidity fluctuations. Therefore, the equation of motion for the first-order statistic (the mean) depends on the second-order statistic, and the equation of motion for the second-order statistic depends on the third-order statistic and so on. This is called the closure problem in turbulence theory. The first-order solution to this problem will play an important role in our profile blending method.

2.1.2 The conservation of momentum

The momentum density is the mass density times the velocity. It obeys the well-known Navier-Stokes equation,

$$\frac{\partial}{\partial t}(\rho v_i) + \frac{\partial}{\partial x_j}\left(\rho v_j v_i\right) = -\frac{\partial p}{\partial x_i} + \eta \frac{\partial^2 v_i}{\partial x_j \partial x_j} - \rho g \delta_{i3} \tag{7}$$

where p is the total pressure, g is the acceleration of gravity, and η is the dynamic viscosity, which is independent of the density. The Navier-Stokes equation can also be written as,

$$\frac{dv_i}{dt} = -\frac{1}{\rho}\frac{\partial p}{\partial x_i} + \nu \frac{\partial^2 v_i}{\partial x_j \partial x_j} - g\delta_{i3} \tag{8}$$

where $\nu = \eta/\rho$ is the kinematic viscosity. Assuming that the density is approximately constant, we obtain an equation for the average velocity field,

$$\frac{D\bar{v}_i}{Dt} = -\frac{\partial}{\partial x_j}\left(\overline{v_j' v_i'}\right) - \frac{1}{\rho}\frac{\partial \bar{p}}{\partial x_i} + \nu \frac{\partial^2 \bar{v}_i}{\partial x_j \partial x_j} - g\delta_{i3} \tag{9}$$

where the first term on the right-hand side is clearly the divergence of the turbulent stress (momentum flux).

2.1.3 The conservation of thermal energy

The thermal energy equation in its full form is rather complicated. However, we can formulate an acceptable substitute by considering only the potential temperature θ,

$$\theta = \left(\frac{p_0}{p}\right)^{\gamma} T \tag{10}$$

where T is the temperature in Kelvins, $p_0 = 1000$ mb is the reference pressure, and $\gamma = 0.285$. If a parcel of air is raised or lowered adiabatically, its pressure will change to match the pressure at its new altitude. This will cause the parcel to expand or contract, and the temperature will drop or increase so as to conform to the first law of thermodynamics (conservation of energy). The potential temperature, on the other hand, remains constant during this process and so is a better measure of thermal energy than the temperature proper (Rogers & Yau, 1989).

Assuming the radiative heating is negligible, we obtain an equation similar to that for specific humidity (6),

$$\frac{d\theta}{dt} = \kappa_T \frac{\partial^2 \theta}{\partial x_j \partial x_j} \tag{11}$$

where κ_T is the molecular thermal diffusivity, which we also take to be a constant. We also obtain a familiar equation for the average potential temperature.

$$\frac{D\bar{\theta}}{Dt} = -\frac{\partial}{\partial x_j}\left(\overline{v_j' \theta'}\right) + \kappa_T \frac{\partial^2 \bar{\theta}}{\partial x_j \partial x_j} \tag{12}$$

2.2 Monin-Obukhov Similarity Theory

The Monin-Obukhov Similarity theory (or M-O theory for short), is supposed to describe the atmospheric surface layer (typically the first 100 m or less above ground level (AGL)) subject to certain restrictions. These are that the surface layer is statistically homogeneous and stationary. Homogeneity implies that all the statistical moment fields (average fields, covariance fields, etc.) are a function of height only. Stationary moment fields mean that they are time-independent. The average conservation equations of specific humidity and potential temperature become,

$$\frac{\partial}{\partial z}\left(\overline{w'q'}\right) = \kappa_v \frac{\partial^2 \bar{q}}{\partial z^2} \tag{13}$$

$$\frac{\partial}{\partial z}\left(\overline{w'\theta'}\right) = \kappa_T \frac{\partial^2 \bar{\theta}}{\partial z^2} \tag{14}$$

where w is the vertical wind. The fact that we have a more or less rigid boundary at $z = 0$ implies that the average vertical wind vanishes, $\bar{w} = 0$. Furthermore, if we take the average horizontal wind to be in the x-direction ($\bar{v}_1 = \bar{u}$ and $\bar{v}_2 = 0$), the average momentum conservation equations become,

$$\frac{\partial}{\partial z}\left(\overline{w'u'}\right) = \upsilon \frac{\partial^2 \bar{u}}{\partial z^2} \tag{15}$$

for the horizontal component and,

$$\frac{\partial}{\partial z}\left(\overline{w'w'}\right) + \frac{1}{\rho}\frac{\partial \bar{p}}{\partial z} + g = 0 \tag{16}$$

for the vertical component. If we assume that the first term on the left hand side of Eq (16) is negligible with respect to the two other terms, we obtain the hydrostatic equation for the average pressure.

$$\frac{\partial \bar{p}}{\partial z} \approx -\rho g \tag{17}$$

Since we consider the molecular diffusivities and the kinematic viscosity to be approximately constant, Eqs (13), (14) and (15) can take on the following form.

$$\overline{w'q'} - \kappa_v \frac{\partial \bar{q}}{\partial z} = \text{Const.} \tag{18}$$

$$\overline{w'\theta'} - \kappa_T \frac{\partial \bar{\theta}}{\partial z} = \text{Const.} \tag{19}$$

$$\overline{w'u'} - \nu \frac{\partial \bar{u}}{\partial z} = \text{Const.} \tag{20}$$

There is almost no turbulence very close to the ground, so the molecular diffusion dominates there. However, almost everywhere else in the surface layer the turbulent diffusion dominates. It is therefore convenient to consider them constant and to define characteristic scale with respect to the vertical turbulent fluxes.

$$u_*^2 = -\overline{w'u'}, \qquad u_*\theta_* = -\overline{w'\theta'}, \qquad u_*q_* = -\overline{w'q'} \tag{21}$$

It is important to note that the velocity characteristic scale u_* in Eq (21) is real. This is because the average horizontal wind at the ground vanishes ($\bar{u}(0) = 0$) due to the no-slip condition on fluid motion. Since the wind is nonzero aloft, this means that there is a transfer of momentum from the air to the ground which entails a negative turbulent momentum flux. The minus signs in the definitions for the potential temperature and specific humidity characteristic scales are merely conventions, but they have some justification when we consider a frequently used turbulence closure scheme: the flux-gradient relation.

Let us consider the potential temperature first, since the generalization to the specific humidity is straightforward. It is natural to assume that turbulence mixes and diffuses conserved quantities until they are evenly spread out. To first-order, this means that the turbulence transports the quantity from where it is, on average, very concentrated to where it is not. Mathematically, this means that the turbulence transports the quantity in the direction opposite to the direction of the average gradient. Therefore, the flux-gradient relation is,

$$\overline{w'\theta'} = -\mathrm{K}_T \frac{\partial \bar{\theta}}{\partial z} \tag{22}$$

where $\mathrm{K}_T > 0$ is the eddy thermal diffusivity. Unlike the molecular diffusivities, the eddy diffusivity is not a property of the fluid, but may depend on a variety of quantities such as position and average velocity. Using the definition of the potential temperature characteristic scale, we have

$$\theta_* = \frac{\mathrm{K}_T}{u_*} \frac{\partial \bar{\theta}}{\partial z} \tag{23}$$

where it is clear that the characteristic scale has the same sign as the average gradient.

The characteristic scales, the height, the average potential temperature, the average specific humidity and the gravitational acceleration give us all we need to define a length scale which we use to scale all the profiles in the surface layer. The Monin-Obukhov (M-O) length scale is

$$L = \frac{u_* \bar{\theta}_v}{kg\theta_{v*}} \tag{24}$$

where $k = 0.41$ is the von Karman constant, $\bar{\theta}_v$ is the average virtual potential temperature over the entire surface layer and θ_{v*} is the characteristic scale of the virtual potential temperature. The virtual potential temperature is defined as

$$\theta_v = \theta[1 + 0.61q] \tag{25}$$

and it is better at describing the buoyancy and stability of moist air than the potential temperature. To a good approximation we therefore have the following estimates.

$$\bar{\theta}_v \approx \bar{\theta}[1 + 0.61\bar{q}], \qquad \theta_{v*} \approx \theta_* + 0.61\bar{\theta}q_* \tag{26}$$

It should be note that although the M-O length is defined using the average virtual potential temperature over the entire surface layer, in practice using the virtual potential temperature at one point in the surface layer, often close to the ground, will suffice. This is because the virtual potential temperature is in degrees Kelvin and therefore does not vary appreciably over the surface layer.

We can now start by defining the gradients of the M-O profiles.

$$\frac{\partial \theta_{MO}}{\partial z} = \frac{\theta_*}{kz}\phi_T\left(\frac{z}{L}\right) \tag{27}$$

$$\frac{\partial q_{MO}}{\partial z} = \frac{q_*}{kz}\phi_q\left(\frac{z}{L}\right) \tag{28}$$

$$\frac{\partial U_{MO}}{\partial z} = \frac{u_*}{kz}\phi_U\left(\frac{z}{L}\right) \tag{29}$$

The functions ϕ_T, ϕ_q and ϕ_U are called the similarity functions. There are a variety of possible similarity functions available in the literature (Forand, 2008), but we will make the following choice. First, we state that the similarity functions for potential temperature and specific humidity are identical and we use the same symbol for both, $\phi = \phi_T = \phi_q$.

$$\phi(\xi) = \begin{cases} 1 + \dfrac{6\xi}{1+\xi}, & \xi \geq 0 \\ (1 - 16\xi)^{-1/2}, & \xi < 0 \end{cases} \tag{30}$$

Whereas the similarity function for the wind has the following form.

$$\phi_U(\xi) = \begin{cases} 1 + \dfrac{6\xi}{1+\xi}, & \xi \geq 0 \\ (1 - 16\xi)^{-1/4}, & \xi < 0 \end{cases} \tag{31}$$

Combining Eqs (23) and (27), we obtain an expression for the eddy thermal diffusivity in the M-O theory.

$$\mathrm{K}_T = \frac{kzu_*}{\phi(z/L)} \tag{32}$$

It is interesting to note both that the eddy diffusivity is a function of height, M-O length and the characteristic velocity scale, and that it does not depend explicitly on the potential temperature characteristic scale. And since potential temperature and specific humidity have the same similarity function, it follows that $\mathrm{K}_q = \mathrm{K}_T$.

The integration of Eqs (27), (28) and (29) yield expressions for the M-O profiles themselves (Forand, 2008),

$$\theta_{MO}(z) = \theta_0 + \frac{\theta_*}{k}\left[\ln\left(\frac{z}{z_T}\right) + \Psi\left(\frac{z}{L}\right) - \Psi\left(\frac{z_T}{L}\right)\right] \tag{33}$$

$$q_{MO}(z) = q_0 + \frac{q_*}{k}\left[\ln\left(\frac{z}{z_q}\right) + \Psi\left(\frac{z}{L}\right) - \Psi\left(\frac{z_q}{L}\right)\right] \tag{34}$$

where z_T and z_q are the roughness lengths for potential temperature and specific humidity, respectively, and θ_0 and q_0 are the values of the potential temperature and specific humidity evaluated at their respective roughness lengths. The stability function takes on the following form,

$$\Psi(\xi) = \begin{cases} 6\ln(1+\xi), & \xi \geq 0 \\ 2\ln 2 - 2\ln(1+x^2), & \xi < 0 \end{cases} \tag{35}$$

where $x = (1 - 16\xi)^{1/4}$. We have a similar form for the wind, except we set $U_0 = 0$.

$$U_{MO} = \frac{u_*}{k}\left[\ln\left(\frac{z}{z_U}\right) + \Psi_U\left(\frac{z}{L}\right) - \Psi_U\left(\frac{z_U}{L}\right)\right] \tag{36}$$

$$\Psi_U(\xi) = \begin{cases} 6\ln(1+\xi), & \xi \geq 0 \\ 2\tan^{-1}(x) - \frac{\pi}{2} + 3\ln 2 - 2\ln(1+x) - \ln(1+x^2), & \xi < 0 \end{cases} \tag{37}$$

Roughness lengths are typically very small, in the order of 1.5×10^{-4} m over a smooth surface. It is for this reason that there is little practical difference between evaluating a value at the surface or at the roughness length.

3 Blending the Surface Layer with the Upper Air

Commentaire [DRDC22]:
Styles
Always use customised, built-in DRDC styles for formatting headings and text – see Section 3.7 of the **Users' Guide**.

In this section, we will demonstrate a method for combining incompatible surface layer and boundary layer profiles based on an eddy diffusivity hypothesis. The problem is as follows: suppose we have surface layer data (ASTD, ASQT, wind near the surface, etc.), from which we deduce characteristic scales (θ_*, q_*, u_*) and M-O profiles of potential temperature ($\theta_{MO}(z)$), specific humidity ($q_{MO}(z)$) and wind ($U_{MO}(z)$). In addition, we have profiles of the boundary layer from radiosonde data or numerical model output, and these profiles don't match at the lowest radiosonde or model datum (typically at around 50 m AGL). How, therefore, do we combine these two profiles into one?

3.1 The Basic Theory

We do so by pointing out that a real atmospheric profile is almost always changing (non-zero time derivative) and that M-O similarity theory is built on the assumption of a perfectly stationary surface layer. We therefore partition a profile into a time-independent M-O component and a time-dependent residual component (which we denote with a prime). For the potential temperature, we have

$$\theta(z,t) = \theta_{MO}(z) + \theta'(z,t)\,. \tag{38}$$

Therefore, the rate-of-change of the profile is

$$\frac{\partial\theta}{\partial t} = \frac{\partial\theta'}{\partial t}\,. \tag{39}$$

We further assume that the residual component obeys a budget equation

$$\frac{\partial\theta'}{\partial t} = -\frac{\partial H_\theta}{\partial z} \tag{40}$$

where H_θ is the turbulent heat flux. We also assume that this flux conforms to the flux-gradient relationship (22).

$$H_\theta = -\mathrm{K}'_T \frac{\partial\theta'}{\partial z} \tag{41}$$

Placing Eq (41) in Eq (40), we obtain

$$\frac{\partial\theta'}{\partial t} = \mathrm{K}'_T \frac{\partial^2\theta'}{\partial z^2} \tag{42}$$

If we assume the time rate-of-change is constant, $\partial\theta'/\partial t = \alpha$, then the residual has a quadratic form,

$$\theta' = A_\theta + B_\theta z + C_\theta z^2 \tag{43}$$

where $C_\theta = \alpha/2\mathrm{K}'_T$. Now suppose that $\theta_R(h_0)$ is the radiosonde measurement of potential temperature at its lowest height $h_0 \approx 50$ m AGL. Suppose further that the M-O profile at that height does not match that value: $\theta_{MO}(h_0) \neq \theta_R(h_0)$. As was mentioned previously, we assume the residual profile makes up the difference.

$$\theta'(h_0) = \theta_R(h_0) - \theta_{MO}(h_0) \tag{44}$$

Equation (44) enables us to determine one of the constant in the quadratic form (43). Also, since we want the M-O profile to dominate near the ground, we set $A_\theta = 0$. This still leaves us with one constant to determine. We do so by assuming that the derivatives match at the linking point.

$$\left.\frac{\partial\theta'}{\partial z}\right|_{h_0} = \left.\frac{\partial\theta_R}{\partial z}\right|_{h_0} - \left.\frac{\partial\theta_{MO}}{\partial z}\right|_{h_0} \tag{45}$$

Assuming we know the next lowest point of the radiosonde, $\theta_R(h_1)$, we estimate its derivative as

$$\left.\frac{\partial\theta_R}{\partial z}\right|_{h_0} \approx \frac{\theta_R(h_1) - \theta_R(h_0)}{h_1 - h_0}. \tag{46}$$

We now have enough equations to determine all the constants. A little algebra gives,

$$B_\theta = \frac{2}{h_0}\theta'(h_0) - \frac{\partial\theta'(h_0)}{\partial z}, \tag{47}$$

$$C_\theta = \frac{1}{h_0}\frac{\partial\theta'(h_0)}{\partial z} - \frac{\theta'(h_0)}{h_0^2}. \tag{48}$$

We perform the same procedure for the specific humidity to find B_q and C_q ($A_q = 0$).

3.2 The Minimum Action Method

The quadratic formula shown in the previous subsection is simple and ensures a continuous blending between the M-O profile and the radiosonde profile, both in terms of values and their first derivative. However, the quadratic formula also suffers from instability. Errors in the radiosonde values can create potentially large residual profiles that are spurious. To counter this,

we introduce a method that relies on a cubic residual profile and a minimum action principle to both determine the extra coefficient and stabilize the residual.

The new form of the residual is then

$$\theta' = A_\theta + B_\theta z + C_\theta z^2 + D_\theta z^3 \tag{49}$$

where, as before, we set $A_\theta = 0$. The boundary conditions now yield,

$$B_\theta = \frac{2}{h_0}\theta'(h_0) - \frac{\partial \theta'(h_0)}{\partial z} + D_\theta h_0^2 \,, \tag{50}$$

$$C_\theta = \frac{1}{h_0}\frac{\partial \theta'(h_0)}{\partial z} - \frac{\theta'(h_0)}{h_0^2} - 2D_\theta h_0 \,. \tag{51}$$

We now see that the coefficients B_θ and C_θ depend on the coefficient D_θ as well as on the boundary conditions. We therefore have a family of residual profiles, each one characterized by a different value of D_θ. This is to be expected, since we have three unknowns and two equations, the problem is underdetermined. We therefore choose the value of D_θ that gives the straightest possible profile. We do this by defining an 'action'.

$$S = \int_0^{h_0} \frac{1}{2}\left(\frac{\partial \theta'}{\partial z}\right)^2 \mathrm{d}z \tag{52}$$

The expression in the square brackets is our 'Lagrangian'. The action depends on the boundary conditions, which are kept fixed, and the coefficient D_θ. The straightest possible profile is the one where the action is a minimum with respect to D_θ, or where

$$\frac{\partial S}{\partial D_\theta} = 0 \,. \tag{53}$$

After some extensive algebra, Eqs (52) and (53) give,

$$D_\theta = \frac{5}{4h_0^2}\frac{\partial \theta'(h_0)}{\partial z} - \frac{5}{4h_0^3}\theta'(h_0) \,. \tag{54}$$

We then place Eq (54) in Eqs (50) and (51). By finding the straightest possible line, we give the residual profile some rigidity.

Returning to the budget equation (42), we obtain the following with the cubic residual.

$$\frac{\partial \theta'}{\partial t} = \mathrm{K}'_T[2C_\theta + 6D_\theta z] \tag{55}$$

This result can mean either that the rate-of-change varies with height (perhaps due to a height dependent potential temperature advection) or that the eddy thermal diffusivity is not constant. If the second hypothesis is true, then the budget equation becomes

$$\frac{\partial \theta'}{\partial t} = \frac{\partial \mathrm{K}'_T}{\partial z}\frac{\partial \theta'}{\partial z} + \mathrm{K}'_T\frac{\partial^2 \theta'}{\partial z^2}. \tag{56}$$

By assuming a constant rate-of-change (α), we can solve it and obtain the following solution,

$$\mathrm{K}'_T = (\alpha z + \beta)\left(\frac{\partial \theta'}{\partial z}\right)^{-1} = \frac{\alpha z + \beta}{B_\theta + 2C_\theta z + 3D_\theta z^2} \tag{57}$$

where β is a constant of integration which we set to $\beta = \alpha B_\theta / 2C_\theta$ so as to ensure compatibility with the quadratic case ($D_\theta = 0$).

$$\mathrm{K}'_T = \left(\frac{\alpha}{2C_\theta}\right)\frac{B_\theta + 2C_\theta z}{B_\theta + 2C_\theta z + 3D_\theta z^2} \tag{58}$$

Of course, the same treatment applies to the specific humidity. However, since the eddy diffusivities depend explicitly on the coefficients which may be different for the potential temperature and the specific humidity, we have in general $\mathrm{K}'_T \neq \mathrm{K}'_q$, unlike the M-O eddy diffusivities.

3.3 The Exponential Method

Although the minimum action method tends to stabilize the blended profile (M-O plus residual profiles), there still exists the possibility that it might take on negative values somewhere between the ground and the linking height for some exceptional boundary conditions. Such a scenario is unlikely in the case of potential temperature given that the potential temperature differences between the surface and the linking height (less than 10^o K) tend to be much smaller than the potential temperatures themselves (around 300^o K). The situation is different for specific humidity since the humidity differences can be of the same order of magnitude as the humidity values themselves. A possible solution would be to adopt an exponential form for the blended profile, as opposed to the linear form of the previous subsections.

$$\theta'(z) = \theta_{MO}(z)\exp[\beta_\theta z + \chi_\theta z^2 + \delta_\theta z^3] \tag{59}$$

It is clear that no matter what the values of the coefficients β_θ, χ_θ and δ_θ may be, the blended profile is never negative. Using the same boundary conditions as before, we obtain

$$\beta_\theta = \frac{2}{h_0}\ln\left(\frac{\theta_R(h_0)}{\theta_{MO}(h_0)}\right) - \frac{\partial}{\partial z}\ln\left(\frac{\theta_R(h_0)}{\theta_{MO}(h_0)}\right) + \delta_\theta h_0^2\,, \tag{60}$$

$$\chi_\theta = \frac{1}{h_0}\frac{\partial}{\partial z}\ln\left(\frac{\theta_R(h_0)}{\theta_{MO}(h_0)}\right) - \frac{1}{h_0^2}\ln\left(\frac{\theta_R(h_0)}{\theta_{MO}(h_0)}\right) - 2\delta_\theta h_0\,. \tag{61}$$

If we set $\delta_\theta = 0$, we obtain the exponential equivalent of the quadratic theory. If we perform the same variation as in Eqs (52) and (53), we obtain

$$\delta_\theta = \frac{5}{4h_0^2}\frac{\partial}{\partial z}\ln\left(\frac{\theta_R(h_0)}{\theta_{MO}(h_0)}\right) - \frac{5}{4h_0^3}\ln\left(\frac{\theta_R(h_0)}{\theta_{MO}(h_0)}\right). \tag{62}$$

This method avoids negative values, but at a price. Unlike the linear method, a physical interpretation in terms of budgets and eddy diffusivities for the exponential method is not as straightforward. Furthermore, the exponential method is prone to greater variability than the linear. However, we may take comfort in the fact that in the limit $\theta_R(h_0) - \theta_{MO}(h_0) \ll \theta_{MO}(h_0)$, the linear and exponential methods are roughly equivalent.

3.4 Correction for Saturation

When linking specific humidity profiles, there exists a possibility that the blended profile exceeds the saturation (100% Relative Humidity (R.H.)) value. We must therefore find a way of modifying the blended profile so that it does not exceed 100% R.H. If $q(z)$ is the blended profile, we want to create from it another profile $q_m(z)$ that does not exceed $q_{sat}(z)$, the saturation profile,

$$q_{sat} = \frac{\varepsilon}{p}e_{sat}(T)\,, \tag{63}$$

where $e_{sat}(T)$ is the saturation water vapour pressure and is a function of temperature only, and $\varepsilon = 0.622$. Rogers and Yau (1989) give the following formula for the saturation water vapour pressure,

$$e_{sat}(T) = 6.112\exp\left(\frac{17.67T}{T + 243.5}\right) \tag{64}$$

where the water vapour is given in millibars and the temperature is in degrees centigrade. This formula is accurate to within 0.1% over the temperature range $-30°\text{C} \le T \le 35°\text{C}$.

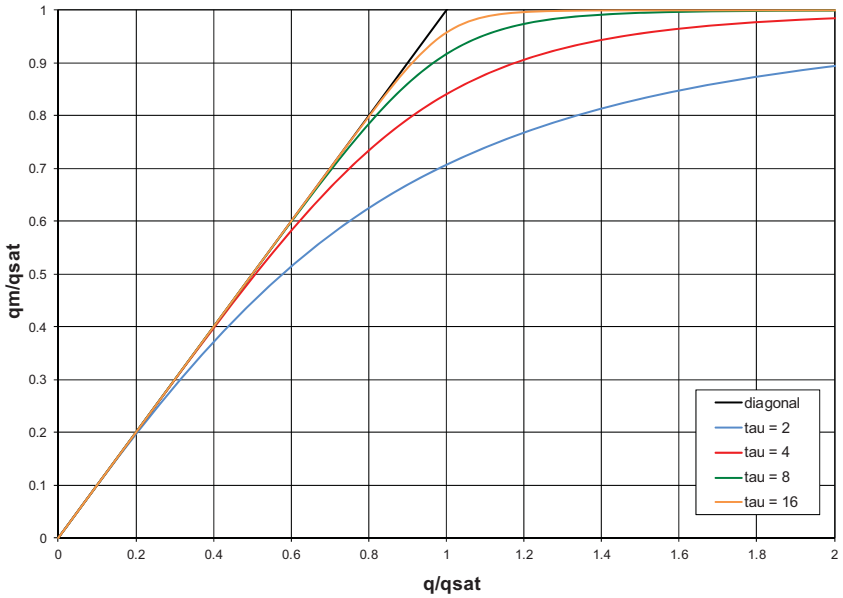

Figure 1: The correction function for the specific humidity for various values of τ.

A possible correction function is given by the following,

$$q_m = \frac{q}{[1 + (q/q_{sat})^{\tau}]^{1/\tau}} \tag{65}$$

where the parameter τ determines the 'sharpness' of the correction function. This is evident in Figure 1, where the curves corresponding to the higher values of the parameter stay closer to the diagonal for unsaturated humidity values, and closer to the asymptotic saturation value for supersaturated humidity values. This implies that we put some thought into the choice of τ we wish to use. A very high value will create less distortion for specific humidity values below saturation, but will also create an abrupt change where the original specific humidity meets and surpasses the saturation value. Such an abrupt change can create a spike in the specific humidity gradient profile, which can adversely influence refractivity profiles for radio propagation.

On the other hand, a low value of the parameter will certainly create distortion at or very near the sea surface (where the R.H. is typically around 98-99%) in the case of a maritime atmospheric surface layer. A possible solution would be to adopt a height-dependent parameter (z), such that

$$\tau(z) = \tau_h \exp(-z/l_\tau) + \tau_l[1 - \exp(-z/l_\tau)] \quad (66)$$

where τ_h is the high value of the parameter, τ_l is the low value and l_τ is the length scale that controls the transition from the high value at the surface and the low value aloft. The optimal parameter values can be obtained by trial and error. This can be done using the Demonstrator described in Section 6.

This page intentionally left blank.

Commentaire [D23]: Blank page
TIP: This page is unnecessary if the preceding page has an even page number. To remove the page, delete the entire sentence.

4 Error Analysis and Evaluation

We now undertake the task of evaluating the suitability of the proposed blending method for the purposes of blending actual surface data with boundary layer data. However, suitability depends greatly on the particular application one has in mind. One application of particular importance is the modelling of radio wave propagation in the boundary layer. We therefore begin this section with a brief overview of the modified refractivity profile and one of its defining features: the duct height. We then develop a first-order linear theory of the sensitivity of the duct height with respect to the boundary conditions at the linking height. We define a root-mean-squared average deviation between the M-O thermodynamic profile and the blended profile and we find its behaviour with respect to the residual value and gradient at the linking height. Finally, we study the effect of the blending method on numerical model outputs.

4.1 The Modified Refractivity Profile

For radio waves, the refractivity is (Battan, 1973),

$$N = 77.6\frac{p}{T} - 5.6\frac{e}{T} + 3.75 \times 10^5 \frac{e}{T^2} \tag{67}$$

where p is the total pressure in millibars, e is the partial pressure of water vapor (mb) and T is the temperature in Kelvins. It should be noted that meteorological data is usually in terms of pressure, temperature and relative humidity. Since the relative humidity in percentage is given as,

$$H = 100\frac{e}{e_{sat}(T)} \tag{68}$$

we can write Eq (67) as

$$N = 77.6\frac{p}{T} - G_N(T)H\,, \tag{69}$$

where G_N is a function that depends only on temperature.

$$G_N(T) = \left[\frac{5.6}{100} - \frac{3750}{T}\right]\frac{e_{sat}(T)}{T} \tag{70}$$

However, we are concerned with the modified refractivity M that takes into account the curvature of the earth,

$$M = N + 10^6\frac{z}{R} \tag{71}$$

where $R = 6368$ km is the radius of the earth and the height z is in kilometres. Of particular importance is the duct height z_c, where the derivative of the modified refractivity vanishes.

$$\frac{dM(z_c)}{dz} = 0 \tag{72}$$

It should be pointed out that not all physically possible modified refractivity profiles have a duct height. For what follows it will be convenient to rewrite the refractivity in terms of the M-O variables.

$$N(p,\theta,q) = \frac{p_0^{\gamma} p^{1-\gamma}}{\theta}\left[77.6 - 5.6\frac{q}{\varepsilon} + 3.75 \times 10^5 \frac{p_0^{\gamma} q}{\varepsilon p^{\gamma}\theta}\right] \tag{73}$$

4.2 Linear Error Analysis

Let us suppose that we find M-O profiles corresponding to our surface data, and that those profiles produce a modified refractivity profile possessing a duct height. Suppose further that we wish to preserve that duct height as much as possible after we perform the blending with the boundary layer data. In this subsection we perform an analysis using small first-order variations to determine the linking height that will least perturb the duct height using the linear method.

We start by defining a derivative function *F*.

$$F = \frac{dM}{dz} = \frac{dN}{dz} + \frac{10^6}{R} \tag{74}$$

Since $N = N(p,\theta,q)$, we can also state the following.

$$F = \frac{\partial N}{\partial \theta}\frac{\partial \theta}{\partial z} + \frac{\partial N}{\partial q}\frac{\partial q}{\partial z} + \frac{\partial N}{\partial p}\frac{\partial p}{\partial z} + \frac{10^6}{R} \tag{75}$$

As stated before, the duct height is where $F(z_c) = 0$. We take F_{MO} as the derivative function for an M-O profile and z_{c0} as its corresponding duct height. If A is a parameter relating to the residual profile, then a small variation of it, δA, will cause a small variation of the conduit height such that,

$$\left.\frac{\delta z_c}{\delta A}\right|_{A=0} = -\left(\frac{dF}{dz}\right)^{-1}\left.\frac{dF}{dA}\right|_{z=z_{c0}}. \tag{76}$$

A variation with respect to one of the potential temperature coefficients gives us

$$\left.\frac{\delta z_c}{\delta B_\theta}\right|_{B_\theta=0} = -\left(\frac{d^2N}{dz^2}\right)^{-1}\frac{\partial N}{\partial \theta}\left.\frac{\partial^2\theta}{\partial B_\theta \partial z}\right|_{z=z_c}, \tag{77}$$

which leads to

$$\left.\frac{\delta z_c}{\delta B_\theta}\right|_{B_\theta=0} = -\left(\frac{d^2N}{dz^2}\right)^{-1}\left.\frac{\partial N}{\partial \theta}\right|_{z=z_c}. \tag{78}$$

Likewise, for the other coefficients we have

$$\left.\frac{\delta z_c}{\delta C_\theta}\right|_{C_\theta=0} = -2z\left(\frac{d^2N}{dz^2}\right)^{-1}\left.\frac{\partial N}{\partial \theta}\right|_{z=z_c}, \tag{79}$$

$$\left.\frac{\delta z_c}{\delta D_\theta}\right|_{D_\theta=0} = -3z^2\left(\frac{d^2N}{dz^2}\right)^{-1}\left.\frac{\partial N}{\partial \theta}\right|_{z=z_c}. \tag{80}$$

We use the chain rule once more to find how the conduit here varies with the residual potential temperature at the linking height (denoted as θ_h'),

$$\left.\frac{\delta z_c}{\delta \theta_h'}\right|_0 = -\left(\frac{d^2N}{dz^2}\right)^{-1}\frac{\partial N}{\partial \theta}\left[\frac{\partial B_\theta}{\partial \theta_h'} + 2z\frac{\partial C_\theta}{\partial \theta_h'} + 3z^2\frac{\partial D_\theta}{\partial \theta_h'}\right], \tag{81}$$

$$\left.\frac{\delta z_c}{\delta \theta_h'}\right|_0 = -\frac{3}{4h_0}\left(\frac{d^2N}{dz^2}\right)^{-1}\frac{\partial N}{\partial \theta}\left[1 + 4\frac{z}{h_0} - 5\left(\frac{z}{h_0}\right)^2\right]\Bigg|_{z=z_{c0}}. \tag{82}$$

Likewise, for the gradient of the residual at the linking height (which we shall call $d\theta_h'$), we obtain this expression.

$$\left.\frac{\delta z_c}{\delta d\theta_h'}\right|_0 = \frac{1}{4}\left(\frac{d^2N}{dz^2}\right)^{-1}\frac{\partial N}{\partial \theta}\left[1 + 12\frac{z}{h_0} - 15\left(\frac{z}{h_0}\right)^2\right]\Bigg|_{z=z_{c0}} \tag{83}$$

Similar expressions hold for the humidity. Since the square brackets in (82) and (83) are quadratic, we can rewrite these expressions as

$$\left.\frac{\delta z_c}{\delta \theta_h'}\right|_0 = -\frac{3}{4h_0^3}\left(\frac{d^2N}{dz^2}\right)^{-1}\frac{\partial N}{\partial \theta}(h_0 - a_+ z)(h_0 - a_- z)|_{z=z_{c0}}\,, \tag{84}$$

where $a_+ = 1$ and $a_- = -5$. This means that if the linking height is equal to the duct height, then the duct height is insensitive to the residual potential temperature at the linking height to first order. We also have

$$\left.\frac{\delta z_c}{\delta d\theta_h'}\right|_0 = \frac{1}{4h_0^2}\left(\frac{d^2N}{dz^2}\right)^{-1}\frac{\partial N}{\partial \theta}(h_0 - b_+ z)(h_0 - b_- z)|_{z=z_{c0}} \tag{85}$$

where $b_+ = -6 + \sqrt{51} \approx 1.14$ and $b_- = -6 - \sqrt{51} \approx -13.14$. Similarly, this means that if the linking height is equal 1.14 times the duct height, then the duct height is insensitive to the gradient of the residual potential temperature at the linking height to first order. If the linking height is much greater that the duct height, $h_0 \gg z_{c0}$, then we obtain the limits,

$$\left.\frac{\delta z_c}{\delta \theta_h'}\right|_0 \approx -\frac{3}{4h_0}\left(\frac{d^2N}{dz^2}\right)^{-1}\left.\frac{\partial N}{\partial \theta}\right|_{z=z_{c0}}\,, \tag{86}$$

$$\left.\frac{\delta z_c}{\delta d\theta_h'}\right|_0 \approx \frac{1}{4}\left(\frac{d^2N}{dz^2}\right)^{-1}\left.\frac{\partial N}{\partial \theta}\right|_{z=z_{c0}}\,, \tag{87}$$

where the sensitivity with respect to the residual potential temperature decreases inversely with respect to the linking height, and the sensitivity with respect to the gradient of the residual potential temperature becomes constant.

4.3 Deviation of the Thermodynamic Profile

We wish to find some measure of the deviation of the blended profile with respect to the M-O profile. To this end, we use the average squared difference between these profiles between the ground and the linking height. We use the potential temperature as an example, but the results can easily be extended to specific humidity.

$$E^2 = \frac{1}{h_0}\int_0^{h_0} \mathrm{d}z\,[\theta'(z)]^2 \tag{88}$$

We use the linear minimum action method, which means that the function in the squared brackets is a third order polynomial (Eq (49)). The integral in Eq (88) can therefore be evaluated and yields the expression,

$$E^2 = \frac{h_0^2}{3}B_\theta^2 + \frac{h_0^3}{2}B_\theta C_\theta + \frac{h_0^4}{5}\left[2B_\theta D_\theta + C_\theta^2\right] + \frac{h_0^5}{3}C_\theta D_\theta + \frac{h_0^6}{7}D_\theta^2 \,. \tag{89}$$

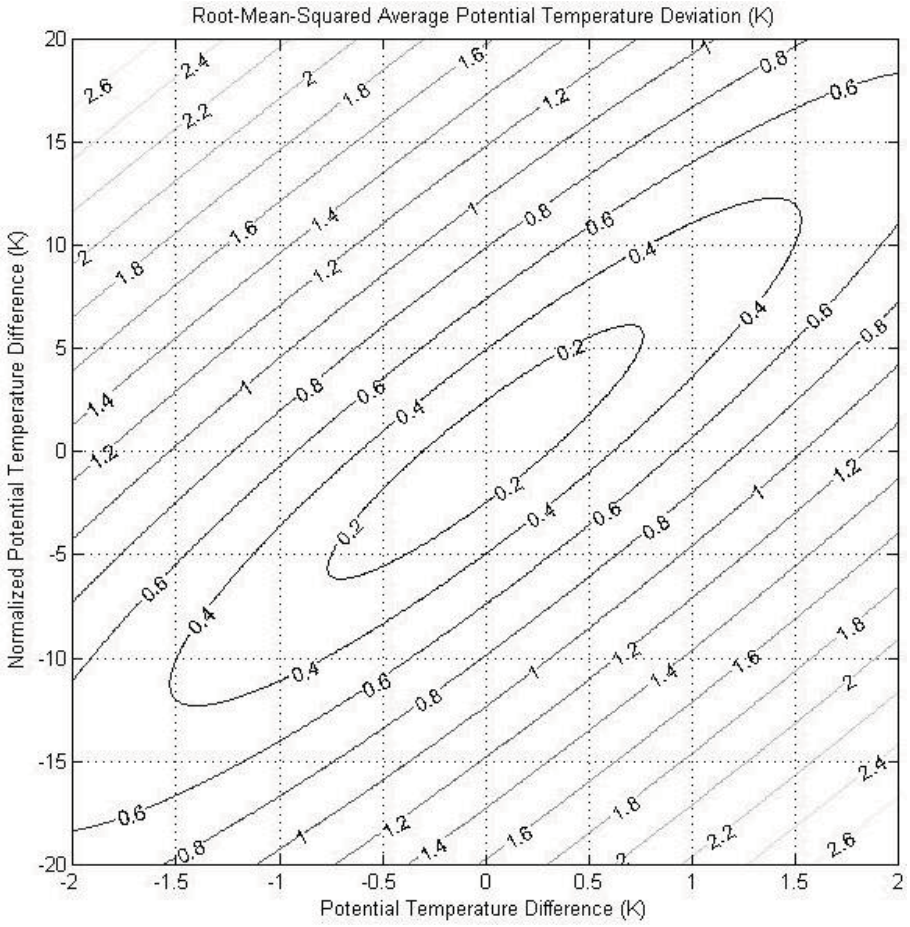

Figure 2: Contour plot of the root-mean-squared average deviation of the potential temperature profile as a function of the potential temperature difference at the linking height and the normalized potential temperature gradient difference at that height.

Given the expressions in Eqs (50), (51) and (54), this reduces to

$$E^2 = \frac{237}{560}(\theta_h')^2 - \frac{27}{280}\theta_h'(h_0 d\theta_h') + \frac{11}{1680}(h_0 d\theta_h')^2 \,. \tag{90}$$

We define the root-mean-squared average deviation by taking the square root of expression (90), which we plot in Figure 2. The quadratic form of Eq (90) causes the contours to take an elliptical form. The major axis of these ellipses is described by a linear relationship.

$$d\theta_h' = \frac{81}{11}\frac{\theta_h'}{h_0} \approx 7.36\frac{\theta_h'}{h_0} \tag{91}$$

Equation (91) can be interpreted in the following manner: for a given linking height and a potential temperature difference, the gradient difference that minimizes the deviation is given by that equation.

4.4 Method Evaluation using Case Studies

In this subsection, we evaluate the blending method using outputs from the Global Environmental Multiscale (GEM) model from the Canadian Meteorological Centre (CMC). The output files are called HPAC files and were used for the Northern Watch Technical Demonstration Program (TDP) (Forand, 2009), as well as for the Shipboard Integration of Sensor and Weapon Systems (SISWS) TDP (Ford *et al*, 2005). We shall use the HPAC files to perform two distinct tasks: 1) evaluate the eddy diffusivity value obtained by the quadratic linear method, 2) evaluate the effect of the linear minimum action method on the duct height. The first task is meant to determine if the blending method is physically reasonable from the point of view of the budget equation. The second is to determine if the method is adequate for radio propagation applications.

4.4.1 Eddy diffusivity evaluation

For this purpose we use HPAC data used for a SISWS field trial off the coast of Halifax, N.S., on the 12th of September 2005. The data consists of an array of meteorological profiles. Each profile is located on a point of a lattice of points about 10 km apart, and there is an array of profiles on the hour every hour from 12h00 GMT to 23h00 GMT. Each profile consists of 10 points unevenly spaced along the vertical from the sea surface to about 1 km. The spacing in space and time is a bit large, but is sufficient for our purpose.

Before examining the GEM output, we must clarify some mathematical issues. First, the budget equation (42) assumes a horizontally homogeneous surface layer. The HPAC file, on the other hand, includes a large-scale horizontal potential temperature gradient. We must modify the budget equation so as to take this into account.

$$\frac{\partial\theta'}{\partial t} + U\frac{\partial\theta'}{\partial x} = \mathrm{K}_T'\frac{\partial^2\theta'}{\partial z^2} \tag{92}$$

A second problem is: once we have estimated the thermal eddy diffusivity, by what standard to we judge its value? We take the M-O diffusivity as our standard. However, the expression for the M-O diffusivity (32) is rather complicated. We therefore make two simplifying assumptions: 1) we are close to neutrality ($z/L \to 0$) and 2) we take the average diffusivity between the ground and the linking height. We consequently obtain the following rough estimate,

$$\mathrm{K}_{MO} \approx \frac{k u_*}{2} h_0 \tag{93}$$

which should be good enough for order-of-magnitude estimates. Finally, we recall that the expression for the thermal eddy diffusivity is

$$\mathrm{K}'_T = \frac{\alpha_0}{2C_\theta} \tag{94}$$

where $\alpha_0 = h_0^{-1} \int_0^{h_0} \mathrm{d}z \, (\partial\theta'/\partial t + U \, \partial\theta'/\partial x)$ is the average warming rate between the ground and the linking height.

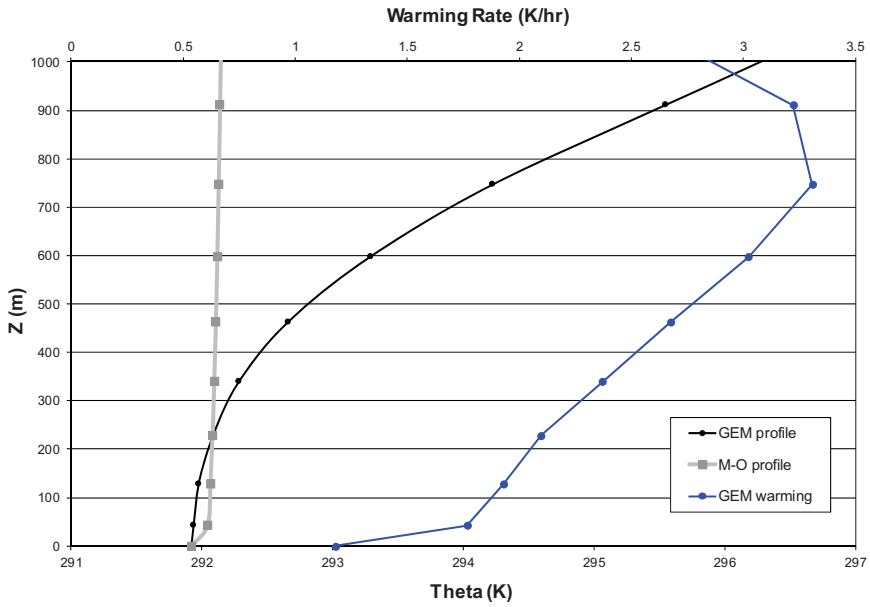

Figure 3: Potential temperature profiles from the GEM and M-O similarity theory. Also shown is the warming rate estimated from the GEM output.

The sample GEM profiles are shown in Figure 3. There we see the GEM potential temperature profile, the GEM warming rate profile and the M-O potential temperature profile, which we obtained using the characteristic scales provided in the HPAC file (,). We use the first point above the sea surface as our linking height ( ). Upon performing the linear quadratic blending, we obtain a thermal eddy diffusivity of 3.36 , whereas the M-O diffusivity is 5.1 . This demonstrates that the thermal eddy diffusivity corresponding to the residual profile is comparable to the M-O eddy diffusivity for this case. More work on this topic remains to be done, but we have at least demonstrated a framework in which we can judge the physical reasonableness of the blending method.

4.4.2 Duct height perturbation

For the purpose of the SISWS field trial, a special HPAC file was created with very fine spatial resolution. We can thus construct an M-O profile from the characteristic scales provided, and then perform a blending with the actual GEM profile at a given height. This gives us a test case to examine the effect of the blending procedure on the duct height. We choose the linking height at 50 m AGL, and we use a linear quadratic blending method. The potential temperature blending is shown in FigureFigure 4 and the specific humidity blending in Figure 5.

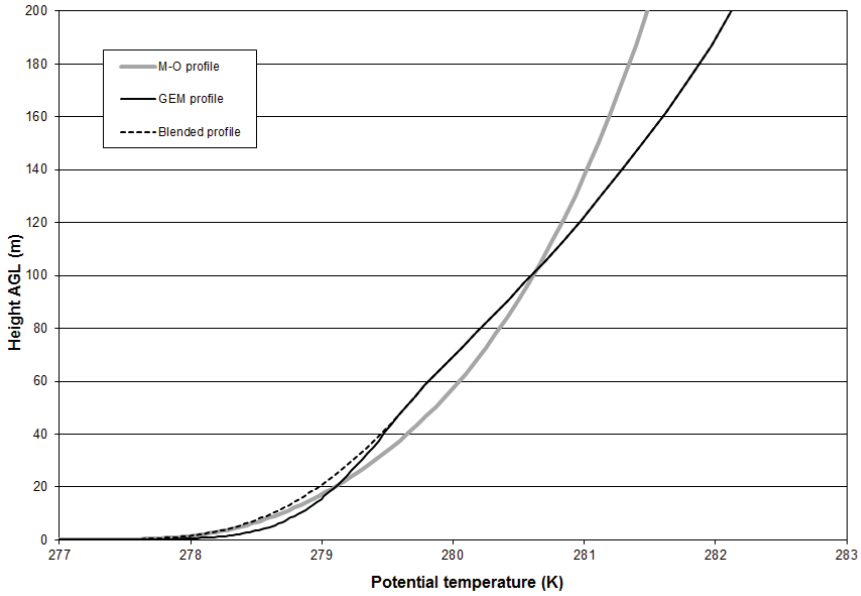

Figure 4: Potential temperature profiles from the GEM and M-O similarity theory along with the blended potential temperature profile.

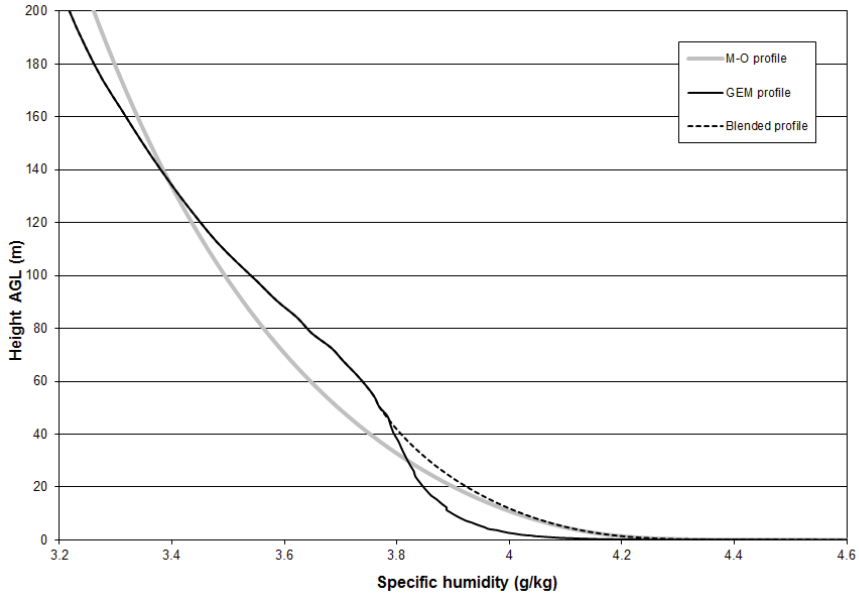

Figure 5: Specific humidity profiles from the GEM and M-O similarity theory along with the blended specific humidity profile.

For both types of profiles, the linking height is chosen where the GEM profile takes a rather abrupt turn. The blended profiles are both smoother than the corresponding GEM profiles but stay closer to the corresponding M-O profiles. The potential temperature and specific humidity profiles result in the modified refractivity profiles shown in Figure 6. The duct height for the GEM profile is about 7.1 m AGL, the M-O profile has a duct height at about 15.4 m AGL, and the blended profile has a duct height at about 12.3 m AGL. As expected, the blended modified refractivity profile has a duct height that is a compromise between the GEM and M-O duct heights, although it is somewhat closer to the M-O value than to the GEM value. While the blended profiles in Figures Figure 4 to Figure 6 have a smooth appearance, despite the fact that we did not use the cubic method with its stabilizing properties, this may be due to the fact that we used the same surface values for the M-O profiles than for the GEM profiles. Had we attempted to blend profiles with difference surface values, the result might have been less appealing. In such a case, we would have had to fuse data from different sources. The next section will examine some techniques for data fusion.

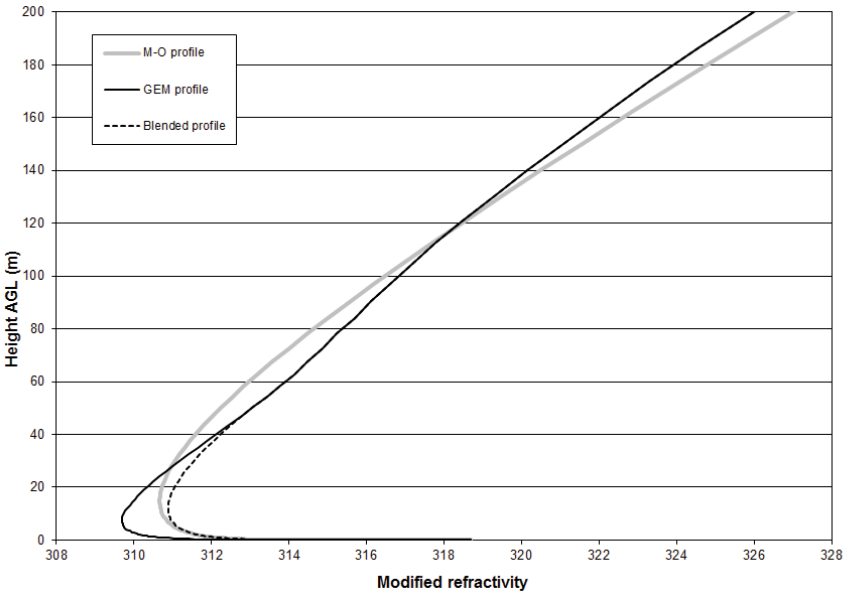

Figure 6: Modified refractivity profiles from the GEM and M-O similarity theory along with the blended modified refractivity profile.

5 Data Fusion

We show how to use the profile linking method in the presence of measurement uncertainty. Specifically, we will use the linear minimum-action method for profile linking. Also, we will assume that the surface value of potential temperature (or specific humidity) may be in error but not the estimate of the surface heat (or specific humidity) flux. In other words, we assume that the surface potential temperature, θ_0, in equation (33) is a normally distributed random variable with a mean $\langle\theta_0\rangle$ and a variance σ_θ^2. Therefore, the M-O profile is translated by a random shift. This, in turn, entails that the residual potential temperature difference at the linking height is a normally distributed random variable with the same variance and a mean $\langle\theta'(h_0)\rangle$.

We have now introduced a degree of uncertainty in the linking method. However, if we were to find the most likely value of the surface potential temperature now, we would trivially obtain the mean value. We need a cost function that will guide us towards a surface potential temperature value that results in a residual profile which is desirable with respect to some criteria. The cost function and the probability density will play off each other, resulting in a surface potential temperature value representing a compromise between the measured value (the mean of the distribution) and the ideal value (corresponding to a cost of zero). We want the vertical derivative of the linked profile to be as close as possible to the vertical derivative of the M-O profile. Put another way, we want the maximum absolute value of derivative of the residual to be as small as possible. Since the residual profile is cubic, the derivative is quadratic.

$$F = \frac{\partial\theta'}{\partial z} = B_\theta + 2C_\theta z + 3D_\theta z^2 \tag{95}$$

This means that we have an extreme value where $F' = 0$, which occurs at the point z_e.

$$z_e = -\frac{C_\theta}{3D_\theta} = \frac{2}{5}h_0 \tag{96}$$

After a bit of algebra, we obtain an expression for the derivative at this point.

$$F(z_e) = \frac{27}{20}\frac{\theta'(h_0)}{h_0} - \frac{7}{20}\frac{\partial\theta'(h_0)}{\partial z} \tag{97}$$

However, we now have a bit of a problem. There is the possibility that the absolute value of the derivatives at the extremities (the surface and linking height) may be larger than that at the extreme point. We can create a cost function by first finding the maximum of these values,

$$\gamma\left(\theta'(h_0)\right) = \max(|F(0)|, |F(z_e)|, |F(h_0)|)\,, \tag{98}$$

where

$$F(0) = \frac{3}{4}\frac{\theta'(h_0)}{h_0} + \frac{1}{4}\frac{\partial\theta'(h_0)}{\partial z}, \tag{99}$$

and

$$F(h_0) = \frac{\partial\theta'(h_0)}{\partial z}\,. \tag{100}$$

We combine the cost function and the probability distribution to create a fitness function.

$$G\left(\theta'(h_0)\right) = \frac{1}{\sqrt{2\pi}\sigma_\theta}\exp\left[-\frac{(\theta' - \langle\theta'\rangle)^2}{2\sigma_\theta^2} - \frac{\gamma^2(\theta')}{2\sigma_{d\theta}^2}\right] \tag{101}$$

The parameter $\sigma_{d\theta}$ represents a tolerance on the derivative we are willing to accept. The next step is to numerically evaluate the fitness function and find the value of $\theta'(h_0)$ that maximizes it. That value represents our compromise between the uncertainty of the surface potential temperature measurement and our desire to minimize the difference in derivative. The new value of surface potential temperature is given as,

$$\tilde{\theta}_0 = \theta_0 + \theta'_{max}(h_0) - \langle\theta'(h_0)\rangle\,. \tag{102}$$

Note that while the potential temperature variance represents the accuracy of the measuring instruments, it can also represent other factors such as the distance between the surface instruments and the launch point of the radiosonde. For instance, if the measuring instrument has an accuracy of $\sigma_{\theta 0}$, but the radiosonde is launched a distance L from it and we estimate the horizontal gradient of potential temperature to be in the order of g_r, then we obtain the following estimate.

$$\sigma_\theta^2 \approx \sigma_{\theta 0}^2 + g_r^2 L^2 \tag{103}$$

6 MATLAB™ Demonstrator

A MATLAB™ Demonstrator was developed for experimentation and testing of profile blending based on the approach developed in this report. The Demonstrator allows one to experiment all options exposed in previous Sections, in particular, the fitness *G* function derived in Section 5. The Graphical User Interface (GUI) of the Demonstrator is shown in Figure 8. The process applies on vertical atmospheric profiles stored in a specific ASCII (text) format. The format is shown in Figure 7. The profile input files may contain more than one atmospheric profile, the first row giving the profile identification number. It is suitable to include in the first two rows (the first two elevations) the surface condition parameters to be blended with the upper-air data. Separation height between the surface conditions and the upper-air conditions must be specified in the "cut under" field of the GUI.

1	0	0	0	0	0
0	0	0	1020.3	16.2	99.8
4	67.5	2	1015.9	17	95
42	20	6	1011	15.1	96.2
127	22	6	1001	14.4	98.1
227	34	5	989	14.4	98.1
337	48	4	976	14.6	98.1
459	65	3	962	14.2	98.1
594	80	2	947	13.6	98.7
743	104	2	931	13	99.3
907	143	2	913	12.6	100
1087	167	3	893	12.1	99.3
1287	185	5	872	11.6	99.3
1507	197	6	850	10.5	98.7
1749	197	6	825	9	98.7
2015	194	7	799	7.3	97.3
2308	192	9	771	6.1	94.6
2630	191	11	742	5.1	96.6
2983	186	13	710	3.1	100
3350	182	14	679	1.2	100
3727	180	15	648	-0.4	97.8
4112	183	16	617	-1.9	94.9
---	---	---	---	---	---
---	---	---	---	---	---
2	0	0	0	0	0
0	0	0	1018.8	16.1	99.8
4	67.5	3	1014.4	16.7	96.2
42	25	8	1009	15.6	97.5
127	26	9	999	14.9	98.7
228	27	9	987	14.7	97.5
338	26	8	974	14.5	98.1
460	27	7	960	14.1	98.7

Figure 7: Example of atmospheric profile input file; the columns are: elevation (m), wind dir. (deg.), wind speed (m/s), pressure (mbar), temperature (degC) and humidity (%)

The connection between surface and upper-air conditions is performed according to the Minimum Action Method exposed in Subsection 3.2, based on a cubic correction profile $\theta'(z)$. The surface conditions are defined in the "Meteo parameters" field. The surface input parameters can be either taken from the profile input file, by clicking on the two "get form UL" buttons, or directly specified by the user. The next two fields, allows one to:

- Select the linear or the exponential method for temperature and humidity (as described in Subsection 3.3); we recall that the exponential method is highly recommended for the humidity;

- Select or not the correction for saturation (exposed in Subsection 3.4), and define the associated parameters (Equation (6)).

The only other parameters that need to be defined for the blending are the connection height (h_0) and the optional shifts of θ and q to improve alignment of profiles before blending. The profiles are blended automatically upon any parameter change.

The grey area of the GUI contains the input parameters of the Demonstrator optimization Options. The process consists in optimizing the fitness function G at various elevations and determining the elevation corresponding to the best G and Duct Height Ratio (DHR) (e.g. highest G and lowest ratio). The DHR is defined as the duct height (d) change through the blending relative to the duct height as obtained from Monin-Obukhov (d_{MO}) in the surface layer. By checking the box "lowest height", one requests to stop iteration at the lowest height where the specified "Tolerances" on both G and the ratio are met. Standard deviations of the G parameters must be defined as input in the appropriate fields. The starting elevation of the optimization process can be specified in meters AGL, or as a multiplicative factor of the stability height (L). The G function used in the Demonstrator is:

$$G = \exp\left[- \frac{\Delta\theta_{MO}^2}{2\sigma_\theta^2} - \frac{\gamma_{\theta'}^2}{2\sigma_{d\theta}^2} - \frac{\Delta q_{MO}^2}{2\sigma_q^2} - \frac{\gamma_{q'}^2}{2\sigma_{dq}^2} - \frac{r^2}{2\sigma_r^2} \right], \tag{104}$$

where $\Delta\theta_{MO}^2$ is the square of temperature displacement (= $((\theta' - \langle\theta'\rangle)^2)$); Δq_{MO}^2 is the square of the humidity displacement; σ_q is the authorized standard deviation (tolerance) of humidity displacement; $\sigma_{d\theta}$ and σ_{dq} are the tolerance imposed on the derivatives of θ and q respectively (see section 5); $r = (d - d_{MO})/d_{MO}$ is the DHR and σ_r is the tolerance on the duct height ratio. In the Demonstrator, the optimization is performed using the SIMPLEX algorithm (Nelder and Mead, 1965).

The graphical pane of the GUI contains four graphics depicting the vertical variation of temperature, humidity, pressure and duct height respectively. The "T" and "H" buttons (on the right-hand side of the pane) allows one to switch to various representations/units of the temperature (T) and humidity (H) profiles respectively. The "Z" button (toggle switch) switches the Z axis to focus on the lowest part of the profiles (e.g. up to h_0). Multiple curves can be displayed at the same time on each graphic using the "curves" buttons: the upper-layer profile

(red); the surface layer M-O profile (blue); the residual surface layer profile (green) and the result of the blending for the overall profile (black – not shown in Figure 8Figure 8).

It is worth noting that the first step of the blending process is the match of pressures and elevations of the upper-air profiles with pressures and elevations in the surface layer. In the Demonstrator, pressure and elevation in the surface layer (at reference height) is used as the ground truth reference and all upper-layer elevations are modified with respect to this reference as a function of the layer pressure. The original upper-layer data (as provided as input) can be displayed in the graphics using the "UL_orig" button.

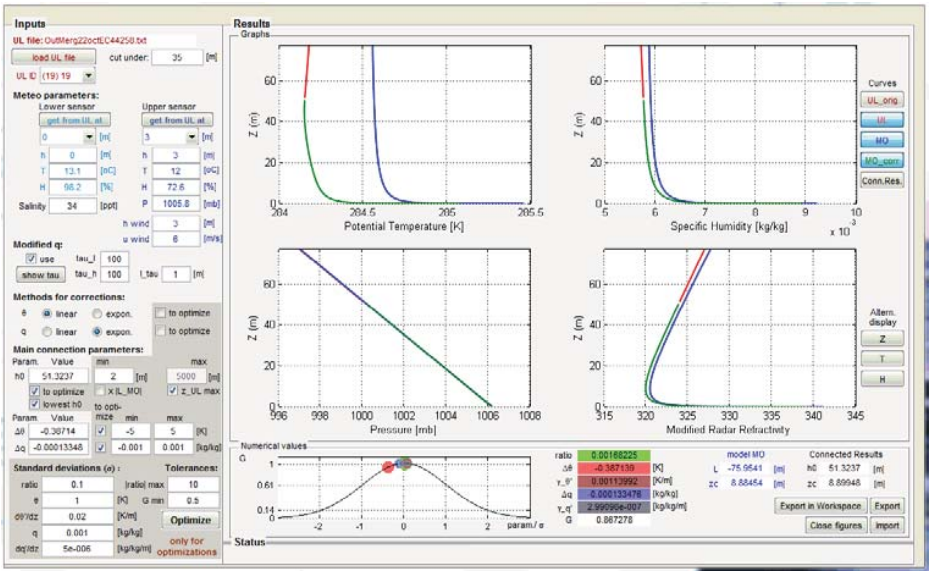

Figure 8: The Demonstrator Graphical User Interface; graphics show results of the optimization process

This page intentionally left blank.

Commentaire [D24]: Blank page
TIP: This page is unnecessary if the preceding page has an even page number. To remove the page, delete the entire sentence.

7 Practical Issues for Profile Blending

The blending technique was tried on several dozen cases using the Demonstrator. Profiles were obtained from the Environment Canada (EC) Limited-Area Model (LAM) High Resolution Deterministic Profiles System (HRDPS). These forecasts are made available in grib2 format on: http://www.weatheroffice.gc.ca/grib/grib2_HRDPS_HR_e.html. Profiles were chosen nearby locations of permanent EC buoys in the Atlantic, near Halifax, NS, and the surface data were taken from the buoys at forecast hours. Hence, the exercise consisted of blending forecast profiles with measured buoy data. For most cases the blending performed pretty well - even without G optimization most of time - as the profile and buoy data were in fairly good agreement under most situations.

Further tests were performed using hypothetical profiles where the air-sea temperature difference was varied from – 8 to +8 degrees C and the temperature at the buoy measurement height was shifted from -6 to +6 degrees C. These tests unveiled some difficulties of the method. The main problem is that G gets difficult to deal with given the number of parameters involved. One may thus wish to consider separately the different parameters of G. In particular the Duct Height Ratio (DHR), the tolerance on duct height change, should be considered separately. Furthermore, unlike with the potential temperature θ, there may be no real constraint on the q-derivative, depending on the application. In addition, value of G attainable for a satisfactory blending depends critically upon the chosen standard deviation of the parameters. A G threshold is thus difficult to define and may vary depending on the conditions.

Moreover, the user should be fully conscious of the impact of shifting q or θ. One has to keep in mind that changing q will directly influence the extinction at the surface in the IR. Besides, changing θ (for a fixed q) will modify the relative humidity which may be of concern in the EO/IR, as it can alter aerosol extinction. Consequently, it is recommended to review the design of G in order to deal efficiently with arbitrary-defined conditions, especially when the surface and upper air input data are taken from distant and dissimilar sources. Given the difficulties of adjusting G, and given the overall good performances obtained with forecasted profiles, one may choose a simpler approach where the blending (and possible shifting) of q and θ is applied iteratively for different connection heights with tests against quality criteria. In most applications, the key criteria are likely to be:

- a DHR as specified be the user
- a maximum $\gamma_{\theta'}^2$
- a maximum deviation of $q(z)$.

The range of possible shifting of q and θ must be a user input. It may conceivably correspond to the temperature and humidity measurement error. However, depending on the applications, it may be appropriate to set it larger. This is justified and desirable for instance when surface parameters are known to be far from the profile data. In the shifting process, precautions to prevent unrealistic specific humidity must be taken (as discussed in Section 3.4). Moreover, one should keep in mind that severe constraints on the DHR (reduced tolerance on the duct height) could prevent q from shifting.

This page intentionally left blank.

Commentaire [D25]: Blank page
TIP: This page is unnecessary if the preceding page has an even page number. To remove the page, delete the entire sentence.

8 Conclusion

In this work, we have developed a method for blending surface data with a boundary layer profile in such a way as to create a blended profile extending from the surface to the linking height with the upper boundary layer profile. Above this point, the blended profile is identical with the given boundary layer profile. The method starts by creating an M-O profile of potential temperature or specific humidity from the surface data (and assuming such data is sufficient). Then we add a 'residual' profile to the M-O profile that bridges the gap between it and the boundary layer profile at the linking height. The residual profile also bridges the gap between the derivatives of the profiles.

The residual profile is at first described by a quadratic polynomial which can be readily associated to a heat (or moisture) flux causing a warming or cooling of the surface layer, contrary to the M-O similarity theory that postulates a stationary surface layer. This creates an interesting physical interpretation where the difference between the M-O and boundary layer profiles is accounted for by the flux responsible for the evolution of the surface layer. While the quadratic form has a straightforward physical interpretation, it also suffers from stability problems. We then considered a cubic polynomial form which is fully determined using a minimum action principle that selects the cubic coefficient corresponding to the straightest possible residual profile. This produces more stable blended profiles, but the specific humidity profiles still present certain difficulties. In rather dry atmospheres, the additive residual profile might create negative humidity values, and in moist atmospheres there is the possibility of creating supersaturated values in the blended profile. We found ways of overcoming these problems (see Subsections 3.3 and 3.4).

In Section 4, we considered the specific case of the RF modified refractive index profile. An important feature of this profile is the duct height. We therefore performed an error analysis of the effect of the blending method on this height. We found a quadratic formula relating the deviation of the duct height with respect to the profile value differences and the profile gradient differences at the linking height. We evaluated our blending method using the outputs of a numerical weather model. First, we performed a test to determine if the diffusivity implied by our method was comparable to the one corresponding to the numerical model, and then we examined the effect on the duct height. In both cases, we found that our blending method performed satisfactorily.

After developing some theory regarding data fusion in Section 5, we present an application of it as a MATLAB^{TM} demonstrator in Section 6. After explaining the demonstrator, we examine some practical issues regarding its proper use in Section 7, but conclude that the demonstrator, and our blending method, performs well under most realistic conditions.

This page intentionally left blank.

Commentaire [D26]: Blank page
TIP: This page is unnecessary if the preceding page has an even page number. To remove the page, delete the entire sentence.

References

Battan, L. J. (1973), Radar Observations in the Atmosphere, Chicago: University Chicago Press, 324 p.

Forand, J. L. (2008), The LWKD Surface Boundary Layer Model (U), v. 8.10, (DRDC Valcartier TM 2007-467) Defence R&D Canada – Valcartier.

Forand, J. L. (2009), Nothern Watch: Barrow Strait Meteorological Data in 2008 (U), (DRDC Valcartier TR 2009-145) Defence R&D Canada – Valcartier.

Ford, B., Conway, L., Toulgoat, M., Forand, J. L., Thomson, A., Martelli, R., Ferland, G., Parry, T., Charland, S., Duprat, G., Jones, G., Cooper, R., Guven, E. and McCarthy, T. (2005), Shipboard Integration of Sensor and Weapon Systems (SISWS): Overview, Benefits, Progress (U), (DRDC Ottawa TR 2005-256) Defence R&D Canada – Ottawa.

Garratt, J. R. (1992), The Atmospheric Boundary Layer, Cambridge: Cambridge University Press, 316 p.

Nelder J. A. and R. Mead (1965), A Simplex method for function minimization, The Computer Journal, Vol. 7, No. 4, pp. 308-313

Rogers, R. R. and Yau, M. K. (1989), A Short Course in Cloud Physics, 3rd ed. Oxford: Pergamon Press, 293 p.

Commentaire [DRDC27]:
References
The purpose of the reference section is to help retrieve references.
It lists all reports, articles and other documents that were used to develop the document.
It must appear as the last section of the body (text), starting on a new page.
Requirements:
- must contain all references cited in the text;
- reference numbers must correspond with those listed in the body (text); and
- references must be listed in order of their appearance.

More information:
Users' Guide: Section 4.12.
Standard: Sections 5.4.5, 6.5.4 and 6.6.6.

This page intentionally left blank.

Commentaire [DRDC28]:
Blank page
This page is unnecessary if the preceding page has an even page number.

More information:
Users' Guide: Section 4.20.
Standard: Section 6.6.8.

Distribution list

Document No.: DRDC TR [enter number only: 9999-999]

LIST PART 1: Internal Distribution by Centre

1 Director General
3 Document Library
1 D.Dion (author)
1 G. Potvin (author)
1 L. Forand

7 TOTAL LIST PART 1

LIST PART 2: External Distribution by DRDKIM

1 Library and Archives Canada

1 DRDKIM (PDF File)
1 DRDC
1 Gilles Fortin (author)
Aerex Avionique Inc.
324 avenue Saint Augustin, Breakeyville, QC, G0S 1E1, Canada

4 TOTAL LIST PART 2

11 TOTAL COPIES REQUIRED

Commentaire [DRDC29]:
Distribution List
The distribution list is a permanent record of the initial distribution of a document and lists all the intended recipients of the document, along with an indication of how many copies of the document each recipient should receive.

More information:
Users' Guide: Section 4.18.
Standard: Section 5.5.7 and Figure 3.

This page intentionally left blank.

Commentaire [D30]: Blank page
TIP: This page is unnecessary if the preceding page has an even page number. To remove the page, delete the entire sentence.

DOCUMENT CONTROL DATA

(Security classification of title, body of abstract and indexing annotation must be entered when the overall document is classified)

1. ORIGINATOR (The name and address of the organization preparing the document. Organizations for whom the document was prepared, e.g. Centre sponsoring a contractor's report, or tasking agency, are entered in section 8.)

DRDC Valcatier
2459, Pie-XI Blvd North
Québec, QC
G3J 1X5 Canada

2. SECURITY CLASSIFICATION (Overall security classification of the document including special warning terms if applicable.)

UNCLASSIFIED

3. TITLE (The complete document title as indicated on the title page. Its classification should be indicated by the appropriate abbreviation (S, C or U) in parentheses after the title.)

On the blending of surface layer and upper air profiles

4. AUTHORS (last name, followed by initials – ranks, titles, etc. not to be used)

Potvin, G.; Dion, D.; Fortin, G.

5. DATE OF PUBLICATION (Month and year of publication of document.)

6a. NO. OF PAGES (Total containing information, including Annexes, Appendices, etc.)

54

6b. NO. OF REFS (Total cited in document.)

7

7. DESCRIPTIVE NOTES (The category of the document, e.g. technical report, technical note or memorandum. If appropriate, enter the type of report, e.g. interim, progress, summary, annual or final. Give the inclusive dates when a specific reporting period is covered.)

Technical Report

8. SPONSORING ACTIVITY (The name of the department project office or laboratory sponsoring the research and development – include address.)

9a. PROJECT OR GRANT NO. (If appropriate, the applicable research and development project or grant number under which the document was written. Please specify whether project or grant.)

9b. CONTRACT NO. (If appropriate, the applicable number under which the document was written.)

10a. ORIGINATOR'S DOCUMENT NUMBER (The official document number by which the document is identified by the originating activity. This number must be unique to this document.)

DRDC TR [enter number only: 9999-999]

10b. OTHER DOCUMENT NO(s). (Any other numbers which may be assigned this document either by the originator or by the sponsor.)

11. DOCUMENT AVAILABILITY (Any limitations on further dissemination of the document, other than those imposed by security classification.)

Unlimited

12. DOCUMENT ANNOUNCEMENT (Any limitation to the bibliographic announcement of this document. This will normally correspond to the Document Availability (11). However, where further distribution (beyond the audience specified in (11) is possible, a wider announcement audience may be selected.))

Unlimited

Commentaire [DRDC31]:
DCD
The DCD must be prepared for all client-oriented documents (except Letter Reports) and must be submitted with the report.

Note: Some of the information required for the DCD is obtained from the Doc Info tab. The DCD Info tab is designed to get the rest of the information required to complete the form (except for the abstract/résumé which is taken from the document itself).

More information:
Users' Guide: Section 4.19.
Standard: Section 5.5.8 and Figure 4.

Commentaire [DRDC32]:
Subtitle
Inclusion of a subtitle on the DCD is at the discretion of the author.

Commentaire [DRDC33]:
No. of pages
The total number of pages will be automatically generated by the template. If the count does not appear to be correct, use the DRDC toolbar to "Update Fields".

TIP: The total number of pages must include the front matter, body and back matter pages, including the Distribution List and the blank page that follows it – it does not include the DCD.

Commentaire [DRDC34]:
No. of references
The total number of references will be automatically generated by the template.
If the count does not appear to be correct, check to make sure that you have used the correct DRDC styles – see Section 4.12 of the **Users' Guide**.

13. ABSTRACT (A brief and factual summary of the document. It may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall begin with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (S), (C), (R), or (U). It is not necessary to include here abstracts in both official languages unless the text is bilingual.)

We expose a profile blending method that combines surface layer data, typically taken from a ground station or buoy, with boundary layer data such as that obtained from a radiosonde. The results are potential temperature and specific humidity profiles ranging from the ground up to the top of the upper layer profile. The surface data is used to create a profile using Monin-Obukhov Similarity theory. The M-O profile extends from the ground to the linking (or connection) height, which is included in the boundary layer profile aloft (which for a radiosonde typically begins at about 50 m AGL). Our method modifies the M-O profile in such a way that both its value and derivative with respect to height agrees with the boundary layer profile at the linking height, while keeping the deviation with respect to the M-O profile as small as possible by using a minimum action principle. Our method is essentially linear, but we also examine an exponential variant in order to avoid negative values in the blended profile (especially relevant for the specific humidity). We also examine ways to prevent the blended specific humidity profile from becoming supersaturated. Since the potential temperature and specific humidity combine to form the modified refractive index profile, we perform an error analysis to determine the effect of the blending method on the refractive index RF duct height perturbations. We then show a profile blending program designed using the principles explained in this report. We suggest possible improvements and conclude that the resulting software performs well for most realistic situations.

14. KEYWORDS, DESCRIPTORS or IDENTIFIERS (Technically meaningful terms or short phrases that characterize a document and could be helpful in cataloguing the document. They should be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location may also be included. If possible keywords should be selected from a published thesaurus, e.g. Thesaurus of Engineering and Scientific Terms (TEST) and that thesaurus identified. If it is not possible to select indexing terms which are Unclassified, the classification of each should be indicated as with the title.)

Surface layer,

profile blending

Commentaire [DRDC35]:
DCD
The abstract/résumé text is copied from the front matter of this document.

More information:
Users' Guide: Section 4.6.2.

Commentaire [DRDC36]:
LAST TIP: Don't forget to use the DRDC toolbar to "Update Fields" before saving and closing your document – see Section 4.21 of the **Users' Guide** for information on saving your document, deleting the embedded comments and updating the fields in the template.

SMART and SMARTI: visible and IR atmospheric radiative transfer libraries optimized for wide band applications

Vincent Ross [a*], Denis Dion[b]

[a]AEREX Avionique Inc., 36 du Ruisseau, Suite 102, Breakeyville, Québec, Canada, G0S 1E2;
[b]DRDC Valcartier, 2459 Pie-XI Boulevard, Val-Bélair, Québec, Canada, G3J 1X5

ABSTRACT

A new C++ library for radiative transfer calculations in the visible and infrared bands which uses MODTRAN as a primary source for atmospheric optical parameters has been developed at Defense R&D Canada, Valcartier (DRDC Valcartier). The main benefit of the library is its capability to perform fast wide spectral band calculations with an appreciably high accuracy. Coherent calculations on wide bands are made possible by using a modified version of the correlated-k theory. The main features of the library are discussed, and comparisons with conventional spectral MODTRAN 4 calculations are presented. It is shown that the library is capable of producing band results that are usually within 5% of MODTRAN 4 with computation times that are thousands of times faster.

Keywords: radiative transfer, propagation code, wide band, visible, infrared

1. INTRODUCTION

Accurate modeling and performance assessment of imaging systems is usually time consuming, even with the processing power of modern computers. The reasons are numerous. First, even if the sensor integrates the electromagnetic spectrum onto a wide spectral band, accurate modeling of the underlying spectral processes is necessary. Secondly, an imaging sensor is usually composed of multiple pixels corresponding to as many lines of sight through the atmosphere. Each line of sight has its own geometry relative to the inhomogeneous atmosphere and the position of the sun. A large number of atmospheric radiative transmission calculations is thus necessary. In addition, applications may require to take into account refraction and multiple scattering within the atmosphere, since these phenomena often produce non negligible effects in the outcome of a simulation.

Moreover, radiative modeling of objects in a scene requires that the radiative environment of the object be known with sufficient accuracy, since the environment may influence greatly the objects radiance through illumination and heating.

To date, very few computational tools exist to efficiently fulfill the requirements underlined in the previous paragraphs. Most electromagnetic propagation codes do calculations on a single line of sight at a time, and almost all do their calculations spectrally at high to moderate resolutions before integrating the results on the sensor band. An exception is the *k*URT model which employs similar principles as those described in this paper[1].

The SMART (Suite for Multiresolution Atmospheric Radiative Transfer) library, along with a simplified interface library called SMARTI (for SMART Interface), has been developed at DRDC Valcartier to provide a new radiative transfer computational solution using MODTRAN. In this paper, the acronym SMART(I) is used when both SMART and SMARTI apply. Libraries were developed instead of executable programs to offer versatility and to facilitate integration into larger projects. The libraries are written in C++, and have seamless wrappers for integration in Java and Matlab applications while wrappers for other languages can be readily generated.

* vincent.ross.AEREX@drdc-rddc.gc.ca; phone: 1 418 844-4000 ext: 4751

Infrared Imaging Systems: Design, Analysis, Modeling, and Testing XXII, edited by Gerald C. Holst, Keith A. Krapels,
Proc. of SPIE Vol. 8014, 80140S · © 2011 SPIE · CCC code: 0277-786X/11/$18 · doi: 10.1117/12.883548

The main feature of SMART(I) is that it allows one to perform computations on wide spectral bands using a wideband version of the correlated-k (CK) approach (see section 3). In this mode, transmission calculations for the entire band of a sensor can be performed in a fraction of the time required for traditional radiative transfer models while maintaining fairly good accuracy. To optimize multiple line of sight calculations, SMART(I) pre-initializes all atmospheric optical properties at once within a single call to MODTRAN, and then proceeds to do all radiative transfer calculations on its own. SMART(I) also handles spectral computations (in a separate spectral mode) at MODTRAN moderate resolutions.

SMART(I) also includes atmospheric refraction and multiple scattering calculations to introduce greater computational accuracy. Through the wide band mode, these features now become usable in applications that would usually require rudimentary approximations in order to maintain appropriate computing times. Multiple scattering computations can also generate radiative fluxes which can be used in target or background modeling.

2. USE OF MODTRAN

SMART(I) uses MODTRAN[2],[3] as a primary source of atmospheric optical parameters. MODTRAN 4 v3 r1 is incorporated seamlessly into SMART, with no modifications to the MODTRAN source code; the official executable release is used directly. SMART accesses the memory space used by MODTRAN and copies the extinction and scattering coefficients on the fly into its own memory. Phase functions are computed in SMART, making use of the same algorithms as in MODTRAN. It is planned to upgrade the compatibility of SMART(I) to other versions of MODTRAN as they become available, starting with MODTRAN 5.

3. THE WIDE BAND CK APPROACH

3.1. The correlated-k method

Transmittance along an inhomogeneous optical path can be calculated through segmentation into homogenous sections. The transmittance along the entire path is then obtained by multiplying the transmittance of each segment:

$$T_{tot} = \prod_{s=1}^{N} T_s \ . \tag{1}$$

The multiplication of segment transmittances in this way is only valid when the extinction spectrum can be perfectly resolved (the case for monochromatic or line-by-line calculations), or if transmittances are completely uncorrelated (if the gas mixture for each segment is completely different for example). These requirements are known as Beer's law. In a real atmosphere, even in a relatively narrow band (1 cm^{-1} or 15 cm^{-1} as used in MODTRAN for example), many spectral lines will likely be aligned (correlated) between segments. In order to adhere to Beer's law, one potentially needs hundreds of monochromatic transmission calculations. This is obviously non desirable when speed is important.

Band models solve part of the problem, but are still not accurate enough for multiple scattering calculations. Correlated-ks offer a good compromise between accuracy, versatility and speed of the calculation. This method consists in reorganizing the spectral information, which is usually very jagged, into a smooth and monotonic function (Figure 1). In practice, this is done by producing an extinction histogram for the band under consideration, and integrating this histogram into a cumulative probability distribution (which by definition will be smooth and monotonic). A smooth monotonic function can be accurately represented by a more limited number of values. A great benefit is that Beer's law can then be used for calculating transmittances since the few correlated-ks effectively resolve the absorption spectrum within the band. The transmittance then becomes the weighed sum (also called block or Riemann integration) of all path segment correlated-k transmittances,

$$T_{tot} = \sum_{k=1}^{n_k} \left[\prod_{s=1}^{N} T_{s,k} \right]_k \Delta g_k \ . \tag{2}$$

where the *s* subscript represents path segments, and *k* is the index of the correlated-k bin. In this new configuration, the information on the origin of the extinction coefficient on the spectral scale is lost (see Figure 1). Hence, all other spectral quantities that make up a radiative propagation calculation (scattering coefficients, phase functions, air blackbody functions, etc.) must be considered invariant on the entire band, not knowing where to place them on the correlated-k scale. This is usually the approximation that limits the most the width of the considered spectral band when dealing with conventional band approaches.

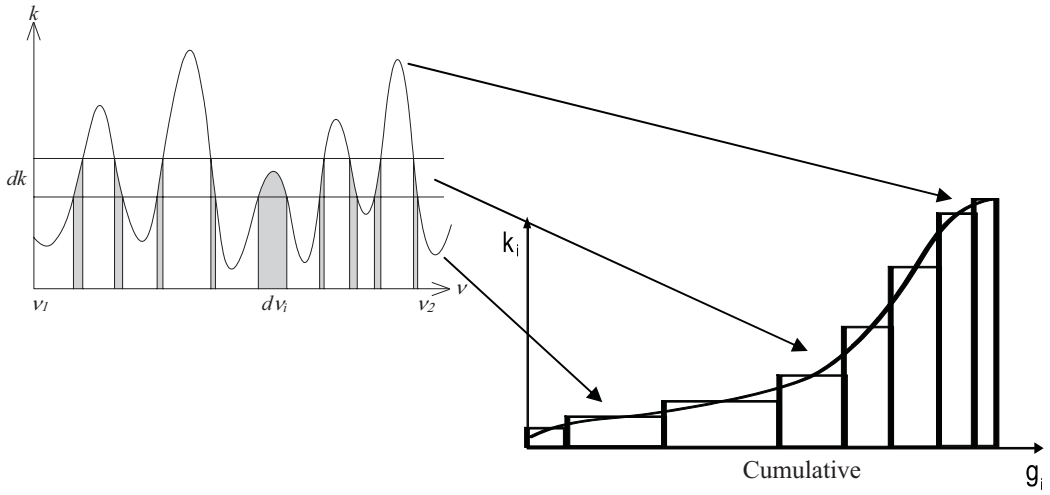

Figure 1: Transformation of line-by-line spectral data into a correlated-k distribution.

3.2. Adapting the method for wide band calculations

When considering wider bands, the use of the correlated-k approach requires some modifications. In the following, wide bands denote any band ranging from about 100 cm^{-1} to many thousand wave numbers (which encompasses usual imaging sensor bands). The theory could be applied to smaller bands, but the speed and accuracy gain would become less attractive.

Wide band correlated-k distributions can be obtained by combining distributions of finer resolution (1 to 15 cm^{-1}) to cover the desired spectral range. In SMART(I), correlated-k distributions of fine resolution are obtained from MODTRAN 4. These individual correlated-k's are then reordered into ascending order along with their summation weight Δg_k in equation (2) (see Figure 2). This produces a distribution for the entire band which is also smooth and monotonic. We can then interpolate these hundreds of smooth correlated-ks on a more limited grid; usually 15 to 35 values suffice. Transmittances can then be obtained using trapeze integration. Equation (2) becomes

$$T_{tot} = \int_{trapeze} \left[\prod_{s=1}^{N} T_s \right] dg \, . \tag{3}$$

The interpolation of correlated-k values is what differentiates the approach used in SMART(I) to other wide band methods which use binning as with the high resolution approach[1]. This pseudo-monochromatic correlated-k approach is more accurate since a trapeze integration scheme better capture the large variations between points than a Riemann sum. For narrow bands (such as the bin widths used in MODTRAN), this is less of an issue since the in-band variability of extinction coefficients is less significant.

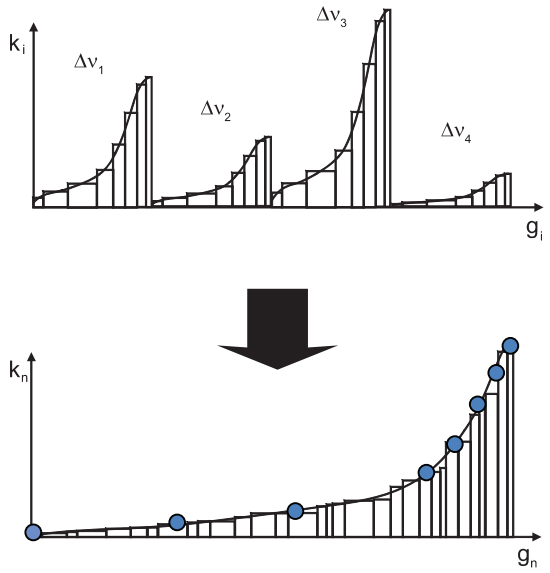

Figure 2: Sorting and interpolating of correlated-k bins into a wide band distribution.

As mentioned in the previous section, all other quantities intervening in the calculation, such as scattering coefficients and phase functions, cannot be represented by a constant in order to maintain some level of accuracy for wide bands. The wide band distribution being made up of higher resolution distributions, the information on the spectral bin origin ($\Delta\nu_i$) of each correlated-k can be retained. The spectral curves describing the other quantities can then be rearranged in exactly the same order as the extinction coefficients (see Figure 3-A). The resulting distributions will likely not be smooth and monotonic, but a good approximation is to use their average value in a bin centered on the wide band cumulative probability points (Figure 3-B). This way, we obtain a much more coherent representation of the many important spectral quantities than by using the traditional correlated-k approach described in the previous section.

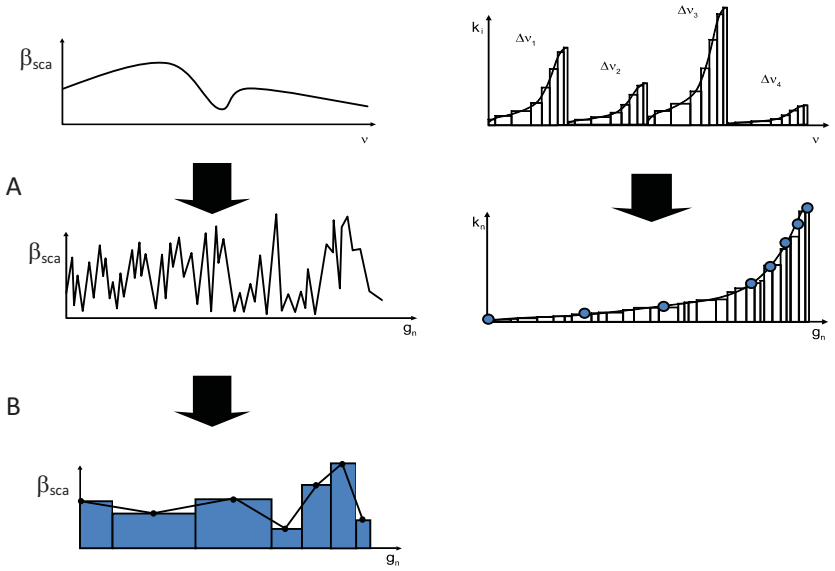

Figure 3: Schematic example of the coherent transformation of a scattering coefficient spectrum (β_{sca}) into the wide band correlated-k space.

4. IMPORTANT CONSIDERATIONS FOR APPLICATION DESIGN

When conceiving algorithms where wide band correlated-ks intervene, many factors must be taken into consideration. For the method to be as efficient as possible, all spectral quantities that might come into play in a given simulation must be converted to wide band correlated-k distributions in an initialization phase. This prevents multiple conversions of the same quantity. Hence, optical parameters of the atmosphere and radiative quantities (scattering coefficients, phase functions, etc. for all layers) and of surfaces (reflectivities, emissivity, blackbodies, etc.) must be converted. This can take as little as a few seconds (or more depending on the number of parameters to transform, and the band width). Once the transformation is done, one can use the wide band correlated-k distributions in the same equations as in spectral calculations, as they keep their physical meaning and units.

Since spectral functions representing the thermal and solar contributions in calculations usually have very different shapes, they each have their own set of correlated-k distributions in SMART(I). This is especially important in the near infrared where both contributions come into play. For instance, different reflectivities have to be considered for any given surface; one for reflecting the sun or atmospheric solar scattering, and one for reflecting atmospheric thermal emissions.

Having all spectral information mapped in wide band correlated-k distributions and kept in the computer's memory, very high calculation speeds (up to 1000 lines of sight per second) can be achieved with relatively low discrepancies with regular MODTRAN computations. Section 10 shows comparisons against pure MODTRAN computations.

It is important to emphasize that transmittances and radiances calculated with this method are scalar quantities, only applicable to the spectral band being considered. It is not possible to infer from the correlated-k distributions any spectral information of higher resolution. Nonetheless, the calculations are done with almost all the spectral coherence that the base MODTRAN resolution (1 to 15 cm^{-1}) would permit.

5. AEROSOL, CLOUDS AND PRECIPITATION MODELS

Since SMART(I) uses MODTRAN as its primary source of optical parameters, all of the built-in MODTRAN aerosol models are available to the SMART(I) user. This includes the rural, navy maritime, maritime, urban, desert and tropospheric models. SMART(I) also includes a more advanced maritime aerosol model developed at DRDC Valcartier[4]. This model produces vertical profiles of extinction, scattering and phase function according to measured environmental quantities, such as temperature gradients, relative humidity, wind speed, wind fetch (distance over which wind has travelled over sea), etc.

Cloud models available in SMART(I) are also those found in MODTRAN. All standard cloud models are available, although MODTRAN cirrus clouds have not been implemented yet. The position, thickness and opacity of clouds can be controlled.

As with clouds, the rain models in SMART(I) are the standard types found in MODTRAN. They are usually associated with one type of cloud, and can be modified according to rain rate.

SMART(I) also adds a falling snow model that is not available in MODTRAN. The model has been developed at DRDC Valcartier, and associates snow flake types along with their optical characteristics with environmental parameters such as temperature. Snow flake type can also be set by the user, and snowfall rate can be specified.

6. SURFACE BRDF MODELS

Radiation impinging at the ground is partially reflected towards the atmosphere where it can be rescattered, and so on; the ground becomes part of the multiple scattering process. Proper characterization of the lower surface bidirectional reflectivity (BRDF) is therefore important, since the ground acts as a lower boundary condition to the multiple scattering algorithm. By providing the lower surface BRDF to the multiple scattering model in the form of Fourier expansion

coefficients (as required by the discreet ordinate method[7]), the atmosphere and ground can be properly coupled. The BRDF is also used to calculate the background radiance when a path intercepts the surface.

SMART(I) incorporates two very different surface BRDF models. The first and simplest one is a Lambert BRDF, and reflects incoming radiation equally in all directions:

$$\rho_d\left(\theta_s,\phi_s,\theta_r,\phi_r\right)=\omega \tag{4}$$

where θ and ϕ are the elevation and azimuth angles, and subscripts s and r denote source and receiver perspectives respectively. The second BRDF in SMART(I) is a sea surface BRDF which contrary to the Lambert BRDF, is a very directional function. We chose an advanced model developed at DRDC Valcartier which relies on sea surface slope statistics and includes wave shadowing and hiding[5],[6].

7. MULTIPLE SCATTERING

The two stream source function is calculated using the Isaacs algorithm[8] as implemented in MODTRAN. Contrary to MODTRAN though, the upward and downward fluxes are calculated using DISORT[7] instead of a dedicated two-flux model. The two-stream, or two-flux multiple scattering source function is obtained from the upward flux $F_{\uparrow}$ and the downward flux $F_{\downarrow}$

$$S_{2str}=\frac{K_{sca}}{\pi K_{ext}}\left[\beta F_{\uparrow}+\left(1-\beta\right)F_{\downarrow}\right]=\frac{\omega_0}{\pi}\left[\beta F_{\uparrow}+\left(1-\beta\right)F_{\downarrow}\right] \tag{5}$$

where β is called the backscattering fraction, K_{sca} and K_{ext} are the scattering and extinction coefficients, and ω_0 is the single scattering albedo. The backscattering fraction is interpolated from a 5th order polynomial in g (the asymmetry parameter) whose coefficients are hard coded in SMART for a series of cosine angles, and linearly interpolated (between polynomials) in cosine of path zenith angle.

The N-stream source function is calculated by using a modified version of the DISORT code. The modified code outputs all the necessary matrices in order to compute the source functions for any scattering angle. This adds one level of accuracy from the DISORT model which is a pure plane parallel code, since the curvature of the path due to refraction and the curvature of the earth is taken into account. The matrices themselves are calculated within DISORT in a plane parallel geometry though. For more information on how the source functions are calculated for a given scattering angle please refer to Stamnes et al. [7]

8. REFRACTION

The calculation of refracted paths through the atmosphere has its basis in the well-known second order vector eikonal differential equation[9],[10],[11]

$$\frac{d}{ds}\left(n\frac{d\vec{r}}{ds}\right)=\nabla n \tag{6}$$

where $\vec{r}$ is the position vector of a point along the path, n is the refraction index of the propagation environment and s is the arc length of the path from start to point $\vec{r}$. If we make the assumption that the atmosphere characteristics vary only along the z axis, differentiation of equation (6) according to θ_0 (the initial look angle corresponding to the direction of $\vec{r}$) leads to a series of coupled differential equations[9]. The solution to the system of differential equations in SMART is

obtained by the Runge-Kutta method of order (4)5 (fifth degree with fourth degree error estimation) according to the Dormand & Prince algorithm with variable step size. The variable transformation

$$n' = n\left(\rho_c + z\right)$$
$$\frac{dn'}{dz} = n + \frac{dn}{dz}\left(\rho_c + z\right) \tag{7}$$

lets us incorporate the curvature of the earth while retaining the simplicity of a 1D differential equation system (see Figure 4). Here ρ_c is the local curvature radius of the earth in the ray's propagation plane. In SMART, the earth is considered ellipsoid, and the local curvature is calculated at each path step.

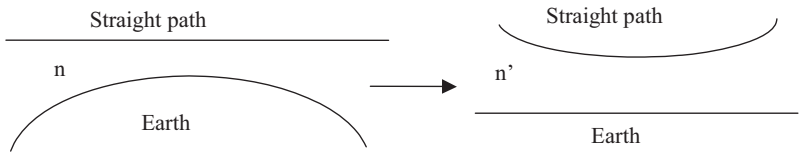

Figure 4: Referential change for a curved earth.

9. COMPUTATIONAL RESULTS

9.1. Spectral mode

Spectral mode SMART(I) computations are compared against conventional MODTRAN 4 calculations with a 1 cm^{-1} resolution. Figure 5 shows both transmittance and path radiance computational outputs, along with the residual differences. The computations are for a 45-degree slant path from ground to space in a standard MODTRAN mid-latitude summer atmosphere with 23-km rural aerosols. Since in SMART(I) optical parameters are taken from MODTRAN, very similar results are expected, and are actually obtained. The small residuals come from slightly different algorithms used for calculating the radiance and transmittance in SMART(I) and MODTRAN.

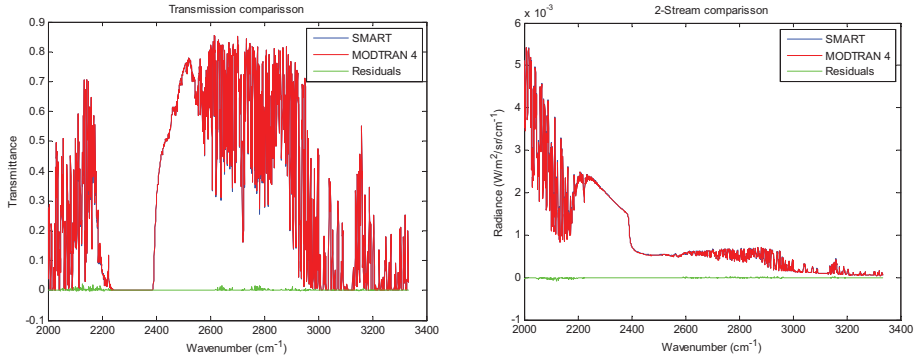

Figure 5: Transmittance (left) and radiance (right) results comparison between SMART(I) and MODTRAN 4.

9.2. Wide band mode

Accuracy and speed of SMART(I) in the wide band mode are evaluated through comparisons with band integrated MODTRAN 4 computations. Parameters are the same as for the spectral computations presented in the previous section. Discrepancies with MODTRAN outputs are reported in Table 1. One notes that in both the visible and the mid-infrared (3 to 5 microns) bands, differences between SMART(I) and MODTRAN do not exceed 3% for both transmittance and radiance. Computation time comparisons are shown in Table 2. SMART(I) wide band CK mode proves to be considerably faster than running a spectral simulation and band-integrating the output. For example, the wide band CK simulation is more than 18 thousand times faster than the band-integrated MODTRAN computation in the visible and with the 16 stream DISORT multiple scattering enabled.

Table 1: Difference between SMART(I) wide band mode results and band integrated MODTRAN 4 results for single, two-stream and DISORT 16 stream scattering modes.

	Radiance			Transmittance
	Single	2 str MS	16 str DISORT	
Visible	0.41%	0.44%	0.18%	0.32%
3-5 μm	2.3%	1.5%	1.8%	3.0%

Table 2: Speed comparison between SMART(I) in wide band CK mode and MODTRAN 4 in single, two-stream and DISORT 16 stream scattering modes.

	Wide band CK			MODTRAN 4		
	Single (17 ck)	2 Str MS	16 Str DISORT	Single	2 Str MS	16 Str DISORT
Visible	0.00078 s	0.00125 s	0.166 s	0.83 s	2.86 s	3061 s
3-5 μm	0.00124 s	0.00234 s	0.19 s	1.05 s	3.08 s	1586 s

10. SMART(I) APPLICATIONS

SMART(I) is suitable to a large array of applications, including the development of electro-optical tactical decision aids (EO-TDA) where rapid computations of detection probabilities is desired (Figure 6) and synthetic 3D scene generation which requires a large number of transmittances and radiances calculations over a wide array of look angles (Figure 7). At DRDC Valcartier, SMART(I) has been recently integrated in the KARMA engagement simulator that requires radiative transfer computations in near real time on a frame by frame basis[12] (Figure 8). This application is especially demanding, since besides transmittance and radiance, radiative fluxes around the target are also required for proper target rendering.

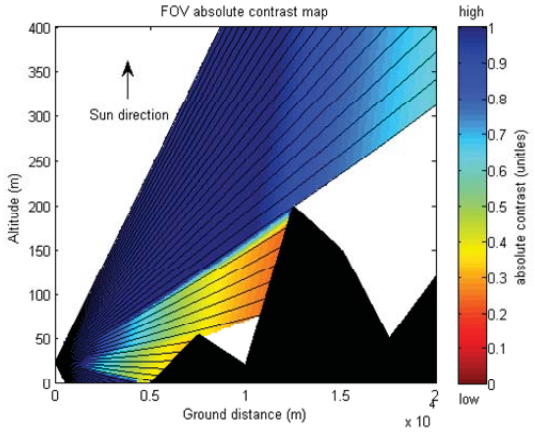

Figure 6: Contrast computations for a target against sea and terrain background.

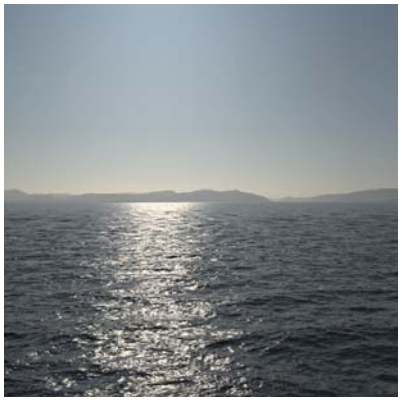

Figure 7: Synthetic 3D rendering of a coastal scene. The realism of the scene is mostly due to the physically modeled sea surface and atmospheric electromagnetic transmission.

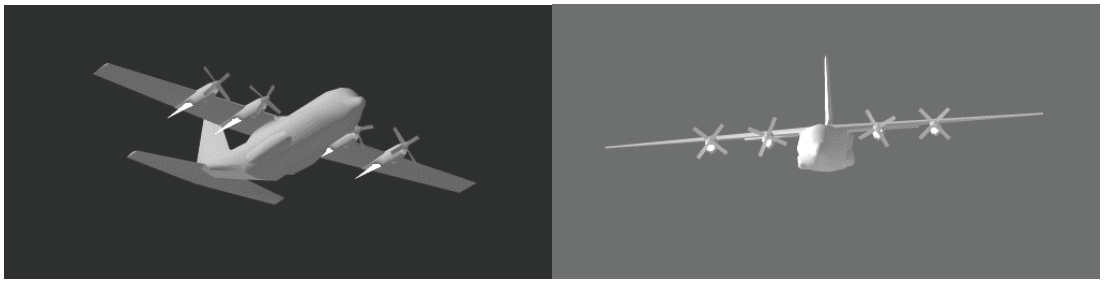

Figure 8: Two views of a target rendered in the threat engagement simulator KARMA (DRDC Valcartier); transmittance, radiance and impinging radiation on the target are calculated using SMART(I).

11. CONCLUSION

SMART(I) developed at DRDC Valcartier provides the capability to calculate radiative transfer quantities such as radiance, transmittance, or radiative fluxes using a wide band adaptation of the correlated-k approach. In this mode of calculation, SMART(I) computations can be thousands of times faster than regular MODTRAN computations (in particular when multi-stream multiple scattering is considered) with output quantities remaining within 5% of full spectral calculations. This opens the door for applications needing small computational times to use features such as multiple scattering normally reserved for cases where accuracy is preferred over speed.

The SMART and SMARTI C++ library structure (with wrappers to other languages like Java) makes it easy to incorporate in larger scale simulations, without the overhead of dealing with writing input files and parsing output files. Getting the key optical parameters and coefficients from MODTRAN, SMART(I) assures a physical validity thoroughly demonstrated over the last decades.

SMART and SMARTI have been developed with evolution in mind, and have a structure that facilitates addition of new models (aerosol models, BRDFs, etc,).

ACKNOWLEDGMENTS

The authors would like to thank Jean-François Lepage at DRDC Valcartier, who engineered the integration of SMART(I) into KARMA. His contribution to the transport plane IR rendering produced with KARMA is noteworthy.

REFERENCES

[1] Acharya, P.K., Panfili, R., Berk, A, Adler-Golden, S. M., Fox, M. and Wetmore, A., "A Correlated-*k* Based Ultra-Fast Radiative Transfer (*k*URT) Method," Proc. SPIE 5982, 598217-598217-9 (2005).
[2] Berk, A., Bernstein, L.S., and Robertson, D. C., "MODTRAN: A Moderate Resolution Model for LOWTRAN7", Rep. GL-TR-89-0122, Air-Force Geophysics Lab., Bedford, MA, (1989).
[3] Acharya, P.K., Berk, A., Anderson, G.P., Larsen, N.F., Tsay, S-Chee, and Stamnes, K.H., "MODTRAN4: Multiple Scattering and Bi-Directional Reflectance Distribution Function (BRDF) Upgrades to MODTRAN", Proc. SPIE 3756, 19-21 (1999).
[4] Dion, D., Gardenal, L., Forand, L., Duffy, M., Potvin, G. and Daigle, S., "IR Boundary Layer Effects Model (IRBLEM) ", IRBLEM software documentation, (2003).
[5] Ross, V., Dion, D., and Potvin, G., "Detailed analytical approach to the Gaussian surface bidirectional reflectance distribution function specular component applied to the sea surface," J. Opt. Soc. Am. A **22**, 2442-2453 (2005)
[6] Ross, V. and Dion, D., "Sea surface slope statistics derived from Sun glint radiance measurements and their apparent dependence on sensor elevation", J. Geophys. Res. 112, C09015 (2007)
[7] Stamnes, K., Tsay, S. C., Wiscombe, W. and Jayaweera, K. "Numerically stable algorithm for discrete-ordinate-method radiative transfer in multiple scattering and emitting layered media", Appl. Opt. 27 (12), 2502–2509 (1988).
[8] Isaacs, R. G., Wang, W.-C., Worsham, R. D., and Goldenberg, S., "Multiple scattering LOWTRAN and FASCODE models," Appl. Opt. 26, 1272-1281 (1987)
[9] Uginčius, P., "Intensity equation in ray acoustics. II", J. Acoust. Soc. Amer. 45 (1), 206-209 (1969)
[10] Blanchard, A., "Phase and intensity ray tracing to study the propagation of coherent radiation in the atmosphere and other media", Defense Research Establishment of Valcartier Canada Report 4699/93, (1993)
[11] Durans, J. C., Granier, "Radar coverage assessment in nonstandard and ducting conditions: a geometrical optics approach", Proc. IEEE 137 (2), 95-101 (1990)
[12] Lepage, J-F., Labrie, M-A., Rouleau, E., Richard, J., Ross, V., Dion, D. and Harrison, N., "DRDC's approach to IR scene generation for IRCM simulation", Proc. SPIE 8015, (2011).

EOSPEC

Description, development environments and installation

LEGAL NOTICE

Martin soucy (LTI)
Vincent Ross (Aerex)
Denis Dion (DRDC Valcartier)

This page is left intentionally blank

Table of contents

1. Introduction

The DRDC Valcartier Environmental Characterization Group has been developing a library of computer models for calculating atmospheric effects on sensors: the Electro-Optical Sensor PErformance Computation Library (EOSPEC-LIB). EOSPEC-LIB contains computational Modules and Utility functions. The Modules included in this version are listed in Table 1. Because of its importance, the SMARTi Utility is added to the list.

The EOSPEC library provides the Integrated Development Environment (IDE) to facilitate the exploitation of the Modules and Utilities. Examples are given for the Modules and the main utility functions. Demos (with GUI) are also given for the Modules.

The EOSPEC-LIB directory structure is as follows:

bin/ ..... Binary files (dll, exe)

build/..... Development solution and projects (.sln and .vcproj)

data/ ..... Data/parameters files used by the modules/functions

doc/ ..... User and technical documentation

examples/..... Source code of examples in C, C++, Java

external/ ……...Contains the external libraries and programs

include/..... Include files for the library API (.h)

lib/..... Compiled objects (.lib files)

license/…….. License documents

src/ ..... Source code of Modules and Utility functions/classes

Upon download of EOSPEC, the *bin* directory contains compilation outputs (.dll files and .exe programs) for a default IDE. One may run demos to get acquainted with the EOSPEC modules. In particular, we recommend running the Demo Manager: *eospec-module-gui-demomanager.exe*.

This document gives instructions for an easy installation of the library in one's development or working environment. The procedure handles 4 development environments, including one for Linux, to adapt to the user compiler suites. The various environments supported by EOSPEC are described in Section 2.

Table 1 - EOSPEC Modules

MODULE	DEPENDENCY	DESCRIPTION
MslMc	none	Based on the Monin-Obukhov Similarity theory, MslMC computes the vertical profiles of air temperature, wind speed, relative humidity and total pressure in the atmospheric surface layer (SL), over land or sea. Besides the atmospheric profiles, MslMC outputs the micrometeorological parameters that characterizes the SL: the Monin-Obukhov length, the scaling parameters and the roughness heights for wind speed, potential temperature and specific humidity.
Metprof	MslMc (called by Metprof)	The MetProfC Module determines the vertical profiles of the dependant meteorological quantities in the atmosphere. In a first step, MetProfC gets the vertical profiles of air temperature, wind speed, relative humidity and total pressure in the surface layer from MslMC. Then, surface profiles are then linked to the upper-air profile specified as input. The height of the inversion layer is also given as output.
AeroProC	MslMc (called by Metprof) Metprof Required files: INTTABLE.REG PHASE.REG External library: Fortran	AeroProC library calculates vertical profiles of aerosol optical properties in the marine boundary layer. Are calculated: Vertical profiles of spectral aerosol extinctions, spectral scattering coefficients and phase functions. (The first 16 Legendre coefficients of the phase functions are calculated). A selection of models is offered for the calculation in the marine surface layer. Profiles are extended in elevation based on NOVAM.
RefPath	MslMc (called by Metprof) Metprof	RefPath calculates the geometrical refracted path of rays through the atmosphere for a vertically varying refraction index. Calculations are done in a 2D propagation plane considering 1D atmospheric profiles, meaning that no lateral or forward gradient in refraction index is considered. Although atmospheric profiles are 1D, the propagation space is 3D along the geographical surface of any given ellipsoidal representing the earth.
RefPlane	MslMc (called by Metprof) Metprof	The RefPlane function is used for analyzing the refractive properties of a propagation plane. Calculations are done in a 2D propagation plane with a 1D atmospheric profile, meaning that no lateral or forward gradient in refraction index is considered. Are calculated: the geometrical and optical horizons, MIVR, the refractance and the distortion transfer function.
Modtran	MslMc (called by Metprof) Metprof AeroProc External: MODTRAN 5.3	The Modtran module provides interfaces for MODTRAN5.3® calculations in marine environments. An option allows one to calculate a complete air-sea background image.

Smarti	SMART utility MslMc (thru Metprof) Metprof AeroProc External: MODTRAN 4 V3 R1 PC executable	The Smarti module provides an interface to facilitate SMART (Suite for Multi-resolution Atmospheric Radiative Transmission) calculations of radiative quantities such as transmitted solar irradiance, atmospheric radiative fluxes, path and background radiances, and transmittance. Calculations can be performed at moderate resolutions (1, 5 or 15 cm-1 wave number bins) or for an entire sensor band using the wideband correlated-k (CK) theory following the DRDC Valcartier approach.

2. Supported working environments

This section shows the various IDE setups that are supported by EOSPEC-LIB. One has to select a setup that is compatible with his/her working/development environment and compiler suite.

The installation procedure is described in section 4 for each setup.

Setup A

Microsoft Visual Studio 2002 2003 2005 2008 2010	C/C++ source code (IDE and compiler) Fortran source code (IDE)
Intel Fortran > 10.1.025	Fortran source code (compiler)

Setup B

Microsoft Visual Studio 2002 2003 2005 2008 2010	C/C++ source code (IDE and compiler)
Code::Blocks with the Fortran IDE (http://darmar.vgtu.lt/)	Fortran source code (IDE)
gfortran > 4.5	Fortran source code (compiler)

Setup C

Code::Blocks with the Fortran IDE (http://darmar.vgtu.lt/)	C/C++ source code (IDE) Fortran source code (IDE)
gfortran > 4.5	Fortran source code (compiler)
gcc > 4.5	C/C++ source code (compiler)

Setup D (for LINUX)

Code::Blocks with the Fortran IDE (http://darmar.vgtu.lt/)	C/C++ source code (IDE) Fortran source code (IDE)
gfortran > 4.5	Fortran source code (compiler)
gcc > 4.5	C/C++ source code (compiler)

3. Required external libraries, programs and tools

EOSPEC-LIB makes use of the external libraries and programs listed here. Users from outside DRDC Valcartier have to proceed by themselves to acquire these elements. (Upon installation from the DRDC Valcartier svn server, all libraries will be stored in the *external* directory.)

Note that since upon download of EOSPEC all *.dll* and *.lib* compatible with the user (default) working environment are resident in the directory structure, the external libraries are not necessary if one does not want to modify the functions that make use of them.

Table 2- External Libraries

Library	Version	Description	Used by
Boost	> 1.4	Boost provides free peer-reviewed portable C++ source libraries.	- The Graphical Demos - SMARTi/SMART for multithreading capability
Swig	> 1.3	to wrap C/C++ functions for use with scripting languages such as Perl, PHP, Python and Java to call EOSPEC library.	Java example of MslMc Module

BOOST can be obtained free from: http://www.boost.org/

Swig can be obtained free from: http://www.swig.org/

Table 3 - External Programs

Program	Version	Description	Used by
MODTRAN	MODTRAN 5.3 executable	Atmospheric band radiation transport model developed by AFRL/VSBT in collaboration with Spectral Sciences, Inc.	MODTRAN module
MODTRAN	MODTRAN 4 V3 R1 PC executable	Atmospheric band radiation transport model developed by AFRL/VSBT in collaboration with Spectral Sciences, Inc.	SMARTi/SMART

4. Setup of the EOSPEC working environment

Premake is a build configuration tool. It is used to generate IDE related files. All files will be created in sub-directory **XXXX** of the **build** directory, where **XXXX** indicates the IDE name. *Premake* does not support Fortran IDE. Nevertheless, facilities have been added to adapt EOSPEC to:

- Visual Studio 2002
- Visual Studio 2003
- Visual Studio 2005
- Visual Studio 2008
- Visual Studio 2010
- Code::Blocks C/C++/Fortran IDE (Windows and Linux)

In the following subsections, installation procedures are described for environment setup options presented in Section 2. For the generation of IDE solutions for the development of Demos (with QT), please see Section 5.1.

4.1 Windows OS – Setup A

4.1.1 Requirements

Microsoft Visual Studio 2002 2003 2005 2008 2010	C/C++ source code (IDE and compiler) Fortran source code (IDE)
Intel Fortran > 10.1.025	Fortran source code (compiler)

4.1.2 Easy build

4.1.2.1 Pre-build process

Launch the script **CreateProject.bat** in the **build** directory and select your version of Visual Studio (Figure 1).

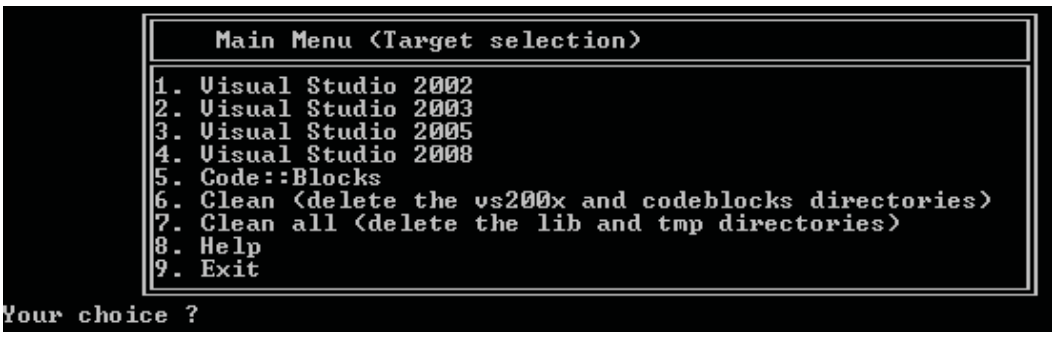

Figure 1: Target IDE selection

Figure 2: Fortran compiler selection

Then, select option **1** to choose the Intel Fortran compiler (Figure 2). The Intel Fortran re-distributable library used by default is saved in the *external/fortran* directory. To use another location or to be able to compile the Fortran source code, define the environment variable **INTEL_FORTRAN_DIR** before running the script **CreateProject.bat**. See section 5 for more details.

4.1.2.2 Compile the Fortran source code [Optional]

In order to compile the Fortran source code, you must install the Intel Fortran Compiler and define the environment variable **INTEL_FORTRAN_DIR**. See the section 5 for more details.

I. Open the solution **module-fortran.sln** in the folder **build\vs20XX** with Microsoft Visual Studio 20XX.

II. Using the menu **File > Add > Existing Project ...** manually add each Fortran project in the folder **build\vs200X\module-fortran** (Figure 3)

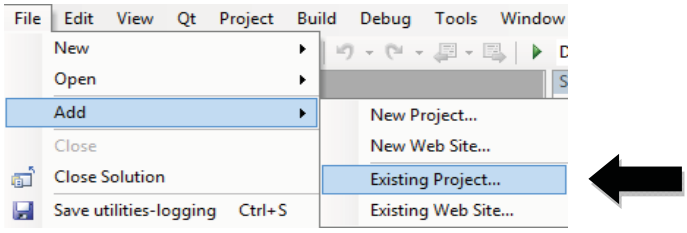

Figure 3: Add an existing project

III. Using the menu **Build > Batch Build ...** (Figure 4), build the entire solution in debug and release configuration. Push the **Select all** button and select **Rebuild**.

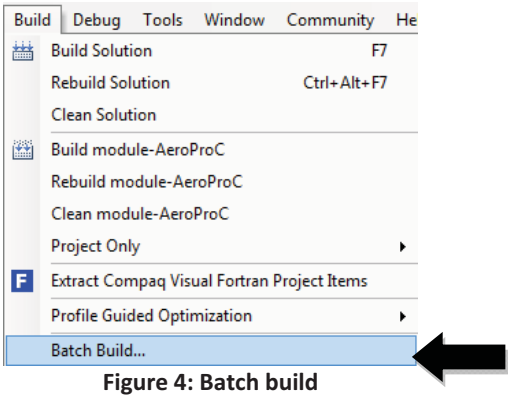

Figure 4: Batch build

4.1.2.3 Compile the C/C++ source codes (to produce dll's)

I. Open the solution **module.sln** in the folder **build\vs20XX** with Microsoft Visual Studio 20XX.

II. Using the menu **Build > Batch Build ...** build the entire solution in debug and release configuration. Push the **Select all** button and select **Rebuild**.

4.1.2.4 Compile the C/C++ examples (to produce .exe programs)

I. Open the solution **module-examples.sln** in the folder **build\vs20XX** with Microsoft Visual Studio 20XX.

II. Using the menu **Build > Batch Build ...** build the entire solution in debug and release configuration. Push the **Select all** button and after select **Rebuild**.

4.2 Windows OS – Setup B

4.2.1 Requirements

Microsoft Visual Studio 2002 2003 2005 2008 2010	C/C++ source code (IDE and compiler)
Code::Blocks with the Fortran IDE (http://darmar.vgtu.lt/)	Fortran source code (IDE)
gfortran > 4.5	Fortran source code (compiler)

4.2.2 Easy build

4.2.2.1 Pre-build process

Launch the script **CreateProject.bat** in the **build** directory and select your version of Visual Studio (Figure 1). After, select the option **2** to choose the gfortran compiler (Figure 2).

4.2.2.2 Compile the Fortran source code

I. Open the workspace **module-fortran.workspace** in the folder **build\codeblocks** with Code::Blocks Fortran IDE.

II. Using the menu **File > Open …** manually add each project in the folder **build\codeblocks\module-fortran** not already in the workspace.

III. Using the menu **Build > Select target > Debug** and **Build > Build workspace** build the entire workspace in debug.

IV. Using the menu **Build > Select target > Release** and **Build > Build workspace** build the entire workspace in release.

4.2.2.3 Compile the C/C++ source code of the core modules

See the section 4.1.2.3 for instructions.

4.2.2.4 Compile the C/C++ examples

See the section 4.1.2.4 for instructions.

4.3 Linux OS

4.3.1 Requirements

Code::Blocks with the Fortran IDE (http://darmar.vgtu.lt/)	C/C++ source code (IDE) Fortran source code (IDE)
gfortran > 4.5	Fortran source code (compiler)
gcc > 4.5	C/C++ source code (compiler)

4.3.2 Easy build

4.3.2.1 Pre-build process

I. Issue the following commands in the shell (don't type $; that represents the shell's prompt):

```
$  cd path_to_eospec
$  cd build
$  ./CreateProject.sh
```

II. Select the option **1** to choose Code::Blocks IDE.

Note: the files CreateProject.sh and premake4 must have “execute” permission. This can be achieved from the shell’s prompt in the eospec/build/ directory with the following commands:

```
$  chmod +x CreateProject.sh
$  chmod +x premake4
```

4.3.2.2 Compile the Fortran source code

I. Open the workspace **module-fortran.workspace** in the folder **build\codeblocks** using Code::Blocks Fortran IDE.

II. Using the menu **File > Open …** manually add each project in the folder **build\codeblocks\module-fortran** not already in the workspace.

III. Using the menu **Build > Select target > Debug** and **Build > Build workspace** build the entire workspace in debug.

IV. Using the menu **Build > Select target > Release** and **Build > Build workspace** build the entire workspace in release.

4.3.2.3 Compile all the C/C++ source code

Using Code::Blocks or Code::Blocks Fortran IDE, open all others workspaces (*.workspace) in the folder **build\codeblocks** and:

I. Using the menu **Build > Select target > Debug** and **Build > Build workspace** build the entire workspace in debug.

II. Using the menu **Build > Select target > Release** and **Build > Build workspace** build the entire workspace in release.

The workspaces must be compiled in the same order than describes in the file **build\codeblocks\ReadMe.txt**.

5. Advanced setup

The **config.lua** file located in the build directory, is used to configure the pre-build process (premake, see section 4). If do not exist, the pre-build process will create one using the **config.tpl** file. You can manually create one by copying the **config.tpl** to **config.lua**.

5.1 Optional Solutions

You can disable or skip solutions of the prebuild process by using the lua function skipSolution(**SOLUTION_DEF**). **SOLUTION_DEF** is the file name of a file located in the **build/solutions_definition** folder. The above example present the content of a file used to disable two solutions, i.e. the modules binding and the modules demos.

```
-- File: config.lua
-- Module binding
skipSolution("3-module-binding.lua")
-- GUI (created with QT)
skipSolution("5-module-demo.lua")
```

You can comment, using the characters "**--**", or remove a line to re-enable a solution in the prebuild process. Furthermore, you can't simply delete the file **config.lua** to add all solutions in the prebuild process.

5.2 Library naming convention

Each library filename is composed of a common sequence of elements that describe how it was built. For the computational module MslMc compiled in debug mode using the Microsoft Visual Studio 2008 IDE, we will have the following file name:

eospec-module-mslmc-vc9-d.dll

eospec	The project name.
module	EOSPEC contains computational modules and utility functions, respectively identified by **module** and **utility**.
msmlc	The computation module name.
vc9	The toolset or the compiler name (see the **Table 4** for more details). This part is optional and can be disabled by adding or un-comment the line **createBuildSuffix = false** in the **config.lua** file. **Note: it is automatically deactivated when we used multiple compilers, i.e. vc9 and gcc (fortran).**
d	Debug version

Table 4: Toolsets or compilers

Toolset name	Description
vc10	Visual Studio 2010
vc9	Visual Studio 2008
vc8	Visual Studio 2005
vc7	Visual Studio 2003
vc6	Visual Studio 2002
gcc	Code::Blocks

5.3 Environment variables

Environment variables are specially named aliases for certain basic system properties that are present for convenience in programming and in system administration. In EOSPEC, environment variables are used to define location of external dependencies (library, tools …).

Table 5- EOSPEC environment variables

	Variables	Core modules	Examples	**Modules binding**	Smart
Optional (see note 1)	INTEL_FORTRAN_LIB_DIR	×	×		
	BOOST_INCLUDE_DIR				×
	BOOST_LIB_DIR				×
Must be manually defined (see note 2)	**JAVA_HOME**			**×**	
	SWIG_DIR			**×**	
	PYTHON_DIR			**×**	
	EOSPEC_DIR			**×**	
	EOPSEC_LIBRARY_SUFFIX			**×**	
See section 5.3.2	**VLD_DIR**	**×**	**×**		**×**

Note 1:

Libraries required to build EOSPEC are available in the external directory. These variables **must be defined** only if you do not want to use the versions provided in the external directory.

Note 2:
The **modules binding** are a particular kind of project used to connect C++ code with scripting languages such as Java, Matlab and Python. Please note that the required tools by the **modules binding** must be manually installed, i.e. the Java Develoment Kit (JDK), the Simplified Wrapper and Interface Generator (SWIG) and Python. The variable **EOSPEC_DIR** is used by the Windows and Linux scripts to prepare and compile the module binding. **EOPSEC_LIBRARY_SUFFIX** is used by the Java code to load the proper library version. By default, if the variable is not configured, the system will load the dll with the **-vc9** suffix (see the section 5.2 for more details).

5.3.1 Details about the environment variables

INTEL_FORTRAN_LIB_DIR
Define the directory of the Intel Fortran compiler containing the **lib**. It's only required with the *Windows OS – Setup A* to build the Fortran source code. [http://software.intel.com]

Ex.: For the version 10.1.025 the INTEL_FORTRAN_LIB_DIR will usually take the following value:

C:\Program Files (x86)\Intel\Compiler\Fortran\10.1.025\IA32\Lib

or

C:\Program Files\Intel\Compiler\Fortran\10.1.025\IA32\Lib

BOOST_INCLUDE_DIR
The root directory containing the BOOST header files. [http://www.boost.org/]

BOOST_LIB_DIR
Define the root directory of the BOOST lib. [http://www.boost.org/]

EOSPEC_DIR
EOSPEC root directory. Required only by the module binding scripts, i.e. build Java, Matlab and Python interfaces.

SWIG_DIR
Define the root directory of Swig. [http://www.swig.org/]

JAVA_HOME
Defines the JDK root directory. Required only by the module binding scripts, i.e. build Java and Matlab interfaces.
[http://www.oracle.com]

5.3.2 Visual Leak Detector

Visual Leak Detector is a free, robust, open-source memory leak detection system for Visual C++ (http://vld.codeplex.com/). First, you must install VLD. Thereafter, update the file **config.lua** to enable VLD to the pre-build process.

```
-------------------------------------------------------
-- Disable Visual Leak Detectector
-------------------------------------------------------
disableVisualLeakDetector = true
```

You can manually create the environment variable **VLD_DIR** to specify the location of VLD. Otherwise, the pre-build process will try the following locations:

1. external/vld
2. C:/Program Files (x86)/Visual Leak Detector
3. C:/Program Files/Visual Leak Detector
4. C:/ProgramFiles/Visual Leak Detector

The prebuild process will create the symbol **USING_VLD** in debug configuration of each C/C++ project.

Then it can be used with any C++ project simply by adding the following line:

```
#ifdef USING_VLD
        #include <vld.h>
#endif
```

By default the file *EnableVisualLeakDetector.cpp* located in the *src* folder has the required statement and is added to each C++ project when the VLD is activated.

When you run a program under the Visual Studio debugger, VLD will output a memory leak report at the end of the debugging session. The leak report includes the full call stack showing how any leaked memory blocks were allocated. Double click on a line in the call stack to jump to that file and line in the editor window.

The SMARTI library

v. 1.1 β

Vincent Ross (AEREX Avionique inc.)

November 2013

Table of Contents

1 Introduction

The SMARTI (for SMART Interface) C++ library was developed to simplify the interfacing of DRDC's SMART (Suite for Multi-resolution Atmospheric Radiative Transmission) C++ library into projects. The main objective of SMARTI is the calculation of atmospheric radiative quantities such as transmitted solar irradiance, atmospheric radiative fluxes, path and background radiances, and transmittance. Although the SMART library is much more versatile, that makes it by design quite complicated and time consuming to learn. SMARTI provides an interface to many of the more useful features of SMART, while taking care of the more complicated aspects of working with SMART internally. SMARTI is capable of calculating radiative quantities at moderate resolutions (1, 5 or 15 cm^{-1} wave number bins) or for an entire band using the wideband correlated-k (CK) theory.

This document is meant to get you started with the SMARTI library. Much more information is also provided in the well documented SMART and SMARTI header files. Knowledge of the SMART library outside of the scope of this document is not necessary in order to use SMARTI. On the other hand, such knowledge is quite beneficial, since as mentioned above, it provides further useful and powerful capabilities to your radiative transmission applications.

1.1 The wide band correlated-k method, briefly

The correlated-k approach originates from the modeling of stellar atmospheres in the 1930's, and has subsequently been adapted to the study of electromagnetic propagation trough the earth's atmosphere in the 1970's and 1980's. The method was initially developed in order to accelerate radiative transmission calculations where line-by-line calculations were too slow, and band models not accurate enough, especially in calculations involving multiple scattering.

Transmittance along an inhomogeneous optical path can be calculated trough the segmentation of the path into homogenous segments. The path transmittance of the entire path is then obtained by multiplying the transmittance of each segment

$$T_{tot} = \prod_{s=1}^{N} T_s \ . \tag{1}$$

The multiplication of segment transmittances in this way is only valid when the extinction spectrum can be perfectly resolved (the case for monochromatic or line-by-line calculations), or if the transmittances are completely uncorrelated (if the gas mixture for each segment is completely different for example). We call this requirement Beer's law. In a real atmosphere, even in a relatively narrow band ($1cm^{-1}$ or 15 cm^{-1} wave numbers for example), many spectral lines will likely be aligned (correlated) between segments. In order to adhere to Beer's law, one potentially needs hundreds of monochromatic transmission calculations. This is obviously non desirable when speed is of the essence.

Band models like those used in the LOWTRAN or MODTRAN [1,2] model scan solve part of the problem. These permit fast computation of atmospheric transmission along an optical path using analytic equations steaming from the extinction coefficient statistics along the path. This requires the analysis of the path as a hole, which must be known in its entirety. This is obviously inapplicable in the case of random paths that are underlying in multiple scattering calculations.

The third method, using correlated-ks, offers a good compromise between accuracy, versatility and speed of the calculation. This method consists in reorganizing the spectral information, which is usually very jagged, in a smooth and monotonic function (Figure 1). In practice this is done by producing an extinction histogram for the band under consideration, and integrating this histogram into a cumulative probability distribution (which by definition will be smooth and monotonic). Having a smooth monotonic function has the benefit of being accurately represented by a limited number of values (the medium resolution bands in MODTRAN 4 use between 17 and 33) instead of the hundreds or thousands required in order to represent a line-by-line spectrum. The correlated-k method also makes the assumption that the spectral content at a given cumulative probability value is the same for all segments, or in other words, that the transformation does not destroy the correlation that existed between the extinction spectra of all atmospheric layers. The benefit of this is that Beer's law can be used for calculating transmittances. The transmittance then becomes the weighed sum (also called block or Riemann integration) of all correlated-k transmittances for the path,

$$T_{tot} = \sum_{i=1}^{n_k} \left[\prod_{s=1}^{N} T_s \right]_k \Delta g_i \,. \tag{2}$$

In this new configuration, we completely lose the information on the origin of the extinction on the spectral scale (see Figure 1). That being, all other spectral quantities that make up a radiative transmission calculation (scattering coefficients, phase functions, blackbody functions, etc.) must be considered invariant on the entire band, not knowing where to place them on the correlated-k scale. This is usually the approximation that limits the most the width of the considered spectral band.

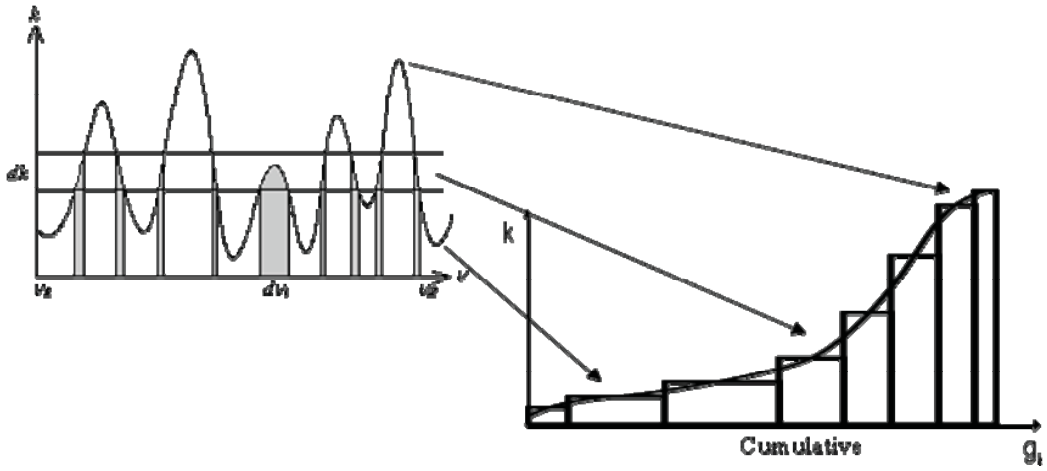

Figure 1

1.1.1 The conversion to wide band

When considering wider bands, the use of the correlated-k approach requires more thought. In the following, we mean by wide bands anything ranging from about 100 cm-1 to many thousand wave numbers (or any reasonable band used in an imaging sensor). The theory could be applied to smaller bands, but the speed and accuracy gain would become less important.

The easiest way to obtain a wide band correlated-k distribution is to start from distributions of finer resolution (1 to 15 cm^{-1}) that encompass the desired spectral range. In practice, these can be calculated using existing models like MODTRAN 4. We then proceed to reorder these individual correlated-ks into ascending order along with their summation weight Δg_i in equation (2) (see Figure 2). This produces a distribution for the entire band which is also smooth and monotonic. We can then interpolate these hundreds of smooth correlated-ks on a more limited grid; usually 15 to 35 values suffice. Transmittances can then be obtained using trapeze integration, instead of a Riemann sum that would no longer capture the large variations between points sufficiently.

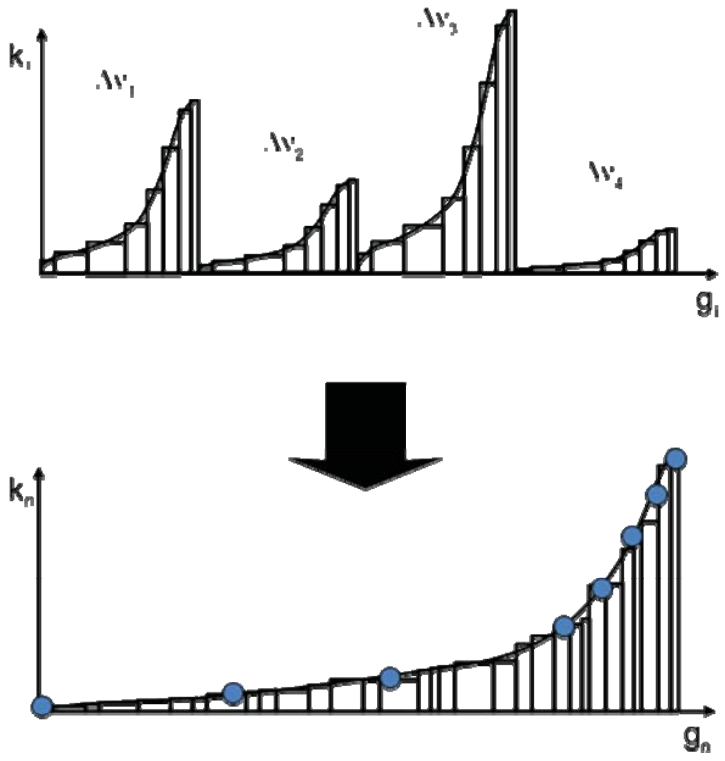

Figure 2

As mentioned in the previous section, all other quantities intervening in the calculation, such as scattering coefficients and phase functions, must be represented by something other than a constant in order to maintain some level of accuracy for wide bands. The wide band distribution being made up of higher resolution distributions, the information on the spectral bin origin ($\Delta\nu_i$) of each correlated-k can be retained. The spectral curves describing the other quantities can then be rearranged in exactly the same order as the extinction coefficients (see Figure 3-A). The resulting distributions will likely not be smooth and monotonic, but a good approximation is to average them around the wide band cumulative probability points (Figure 3-B). This way, we obtain a much more coherent representation of the many important spectral quantities than by using the traditional correlated-k approach described in the previous section. It is this maintained spectral coherence that enables us to maintain good accuracy in most cases, even with wide bands.

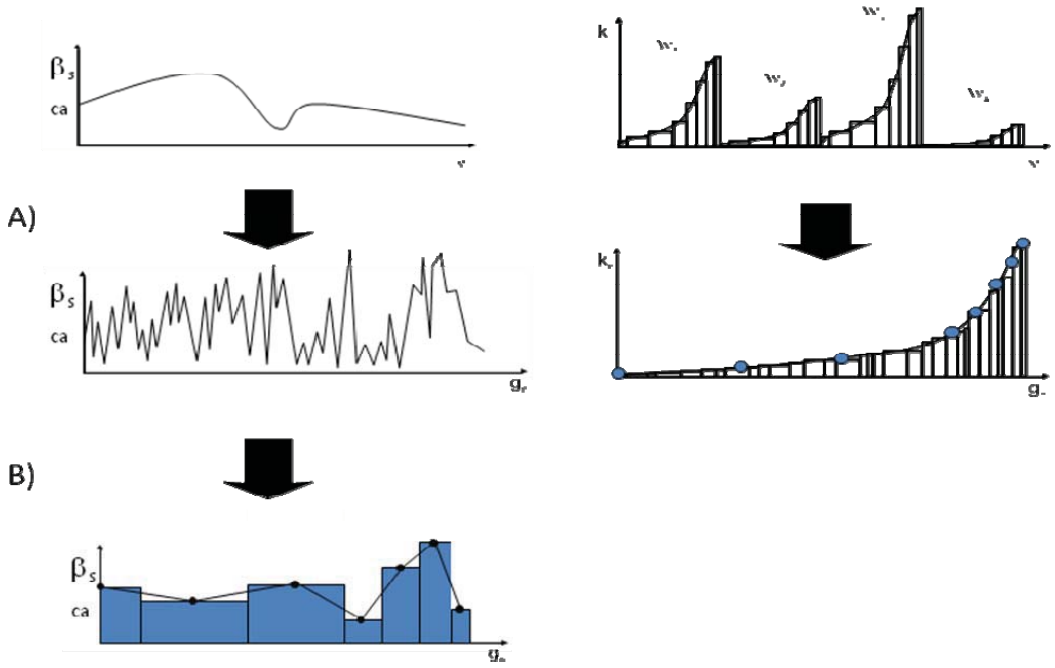

Figure 3: Schematic example of the coherent transformation of a scattering coefficient curve into the wide band correlated-k space.

1.1.2 Important considerations in algorithm conception

When conceiving algorithms where wide band correlated-ks intervene, many factors must be taken into consideration.

For the method to be as efficient as possible, all spectral quantities that might come into play in a given simulation must be converted to wide band correlated-k distributions in an initialization phase. This prevents multiple conversions of the same quantity. Hence, optical parameters of the atmosphere (scattering coefficients, phase functions, blackbodies, etc. for all layers) and of surfaces (reflectivities, emissivity, blackbodies, etc.) must be converted. This can take as little as a few seconds (or more depending on the number of parameters to transform, and the band width). Once the transformation is done, you can use the wide band correlated-k distributions in the same equations as if they were spectral, as they keep their physical meaning and units.

Since spectral functions for the thermal and solar contributions in calculations usually have very different shapes, they each have their own set of correlated-k distributions in SMART and SMARTI. This is especially important in the near infrared where both constituents are present. Because of this, for example, you will end up with two different reflectivities for any given surface; one for reflecting the sun or atmospheric solar scattering, and one for reflecting thermal emissions.

If an algorithm is well conceived, once all spectral information is converted to wide band correlated-k distributions and kept in the computers memory, you can expect to obtain very high calculation speeds

(up to 1000 lines of sight per second) with relatively good accuracy (usually within 5% compared to MODTRAN 4 spectral calculations).

It is important to emphasize that the transmittances and radiances calculated with this method are scalar quantities, only applicable to the spectral band being considered. It is not possible to infer from the correlated-k distributions any spectral information of higher resolution. Nonetheless, the calculations are done with almost all the spectral coherence that the base MODTRAN resolution (1 to 15 cm^{-1}) would permit.

For a concrete example of how to use the wide band correlated-k method in a typical case, please refer to annex A.

2 System requirements

SMARTI is a Windows (PC) .dll library, and has been compiled and tested on a Windows XP 32 bit and Windows 7 32 and 64 bit platforms (but there is no reasons why it should not work on any 32 or 64 bit Windows version, providing that the proper C run time libraries, CRT, are installed). Many of the atmospheres optical properties are obtained by interfacing to either the official MODTRAN® 4 version 3 release 1 executable (Mod4v3r1.exe) as distributed by the ONTAR corporation (www.ontar.com), or a version of MODTRAN® 4 or 5 modified with the provided utility. The correct version of MODTRAN® should thus be properly installed prior to using SMARTI.

Since SMARTI's philosophy is to pre-initialize as much data as possible, depending on the use you make of it, it might also be a good idea to have plenty of RAM available (2 Gb).

3 Exception handling with SMARTI

SMARTI methods and the SMART classes within will throw `smartException` type exceptions when something happens so it is good practice to include all code involving SMART in a try/catch block:

```
try{
   // ... Code ...
}catch( smartException& e){
   // ... Handle exception...
}
```

Since the `smartException` type is derived from the standard `exception` type, using `catch( exception& e)` would also work but would be less specific.

A `smartException`, overloads the standard `exception` class `what()` method to output a formatted message indicating the origin of the exception (file and line) as well as a track trough the code. Each time a function catches the exception, it throws it back (using the `smartReThrow` macro), adding a file and line trace to the message, the calling function can then catch and re-throw the exception adding its own trace and so on. This can help you track the exception through the code, so you should also make use of the `smartReThrow` macro in your own code. For example, an intentional out of bounds `gridVector` class element access introduced somewhere in the code produces this message:

```
Access to element 49 when legal values are 0 to 48.

    File: gridvector.h
    Line: 1540

    File: sourcefunction.h
    Line: 958

    File: pathradiance.h
    Line: 374
```

meaning that the exception was initially thrown in gridVector.h, which was being called by sourcefunction.h (where the error was introduced) which was being used by pathradiance.h.

Note that many exceptions (such as out of bounds errors in SMART classes) are thrown only if SMART_FULL_DEBUG is defined at the compiler lever. Because of this, it is a good idea to define this while developing and testing code involving SMART or SMARTI. When you are certain your code works well without throwing exceptions, then you can remove SMART_FULL_DEBUG from your definitions in order to improve compilation and run-time efficiency.

4 Data types

For the most part, the SMARTI library uses its own set of classes as output in order to organize and make accessible the information. These can then be used as input into other SMARTI methods. The SMARTI data types are often based on elaborate SMART types, so their interface will not be described in detail in the following sections. Instead, the objective here will be to present the most useful features in the context where SMARTI is most likely to be used. Those interested in the complete interface description are invited to consult the appropriate SMART library header files.

4.1 Spectral types

Types describing spectral quantities are divides into two categories: normalized spectral quantities and radiative spectral quantities. Each of these two is then subdivided into two sub-categories: spectral and correlated-k (CK). The four basic types are then:

- `normType` (for normalized spectral data)
- `radType` (for radiative spectral data)
- `normCKType` (for normalized spectral correlated-k data)
- `radCKType` for nradiative spectral correlated-k data)

Normalized data types represent any quantity that needs renormalization after integration on a spectral band. Such quantities include transmittances, reflectances and albedos, extinction and scattering coefficients, etc. In other words, anything that does not include (spectral unit)$^{-1}$ in their own units.

Radiative data types, on the other hand, represent any quantity that does not need renormalization after integration on a spectral band. Such quantities include radiances, irradiances, fluxes, etc. In other words, anything that include (spectral unit)$^{-1}$ in their own units.

All spectral quantities are binned onto the same wave number grid. A spectral data bin is represented by a scalar value and a correlated-k spectral data bin is represented by a vector of correlated-k values (see Figure 4).

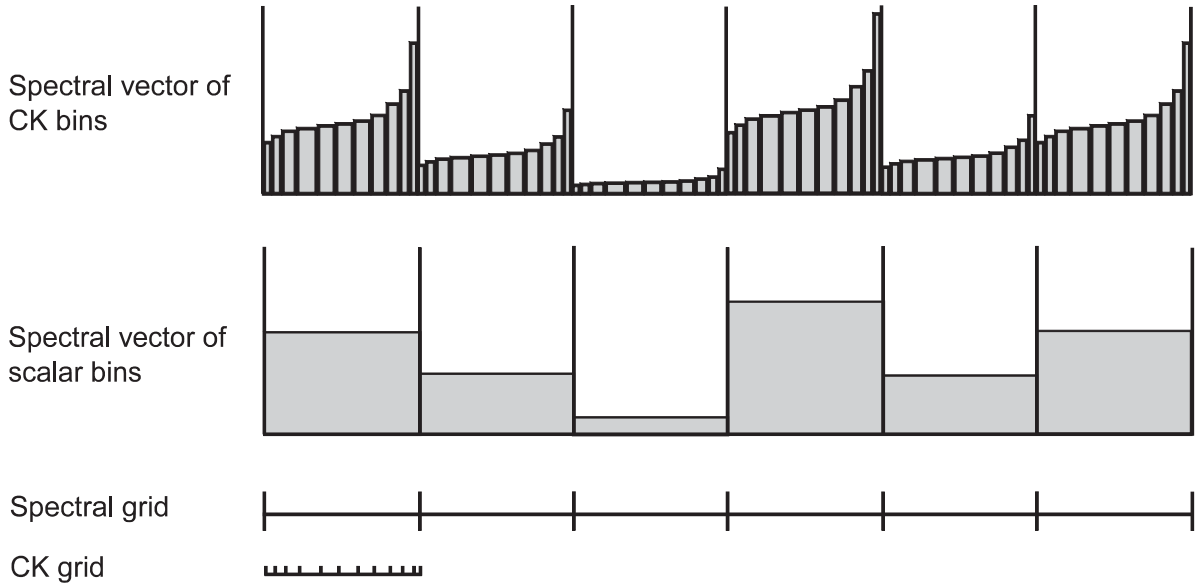

Figure 4: Representation of the different spectral quantities

The integrated value of any spectral vector or value of a spectral bin or CK element is always obtained by using the `value()` function which always returns a floating point value (type `float` in the SMARTI implementation). Furthermore, access to spectral or CK elements is achieved by using the `[]` operator

(with indexes starting at 0). So say you have a spectral CK quantity of type `radCKType` called `radCKType_instance`, with units $W/m^2/cm^{-1}$, the band integrated value of `radCKType_instance` is:

```
float band_value = value(radCKType_instance);
```

with units W/m^2. The value of the 10^{th} spectral bin of `radCKType_instance` is then:

```
float bin_value = value(radCKType_instance[9]);
```

with units $W/m^2/cm^{-1}$. To know the length of the vector, you can use the `getLength()` method. It is not hard to guess that obtaining the value of the 2^{nd} correlated-k value of the 10^{th} spectral bin of `radCKType_instance` is done by doing:

```
float ck_value = value(radCKType_instance[9][1]);
```

Note that CK bins also have a `getLength()` method. In fact, you should always use the `value()` function to access any spectral value at any level, unless you know the SMART library very well and actually expect to get whatever type is returned by the `[]` operator (which is usually not a `float`).

All SMARTI vectorized spectral quantities (spectral vectors or CK bins) have an associated grid. For a binned vector, this grid represents the bin boundaries, and therefore has a length one more than the vector length. You can access the grid values using the `getGrid( index )` method:

```
float spectral_grid_value = radCKType_instance.getGrid(9);
float ck_grid_value = radCKType_instance[9].getGrid(1);
```

SMARTI spectral types also have all arithmetic (unary and binary) operators overloaded (+,–,/,*,+=,–=,*=,/=), as well as all standard math functions (`sin`,`cos`,`log`,`exp`,etc.). You can then apply any operator between two SMARTI spectral type instances (normalized or radiative), or between a SMARTI spectral type instance and a scalar (`float`), in any sequence:

```
radCKType operation_result;
operation_result = - radCKType_instance*exp(normCKType_instance)/2;
```

Note that one quirk of SMARTI arithmetic operations is that you cannot initialize a new instance with the result of an operation using the = operator:

```
radCKType operation_result = 2*radCKType_instance; //won't compile
```

Instead, to initialize a new instance with an operation, you must use a constructor:

```
radCKType operation_result( 2*radCKType_instance); //compiles
```

Also, you can only operate between two SMARTI spectral type instances that have the same spectral grid, which is only true when both have been produced by the same SMARTI instance. Trying to operate between two SMARTI spectral type instances from two different SMARTI instances (even if they have

identical spectral bands and resolution) will throw an exception of type `smartException` (see section 3).

It is possible to down-cast from CK to spectral types, as long as you stay within the same category (radiative and normalized):

```
radType radtype_instance = radType(radcktype_instance);
normType normtype_instance = normType(normcktype_instance);
normType normtype_instance = normType(radcktype_instance);  //won't compile
```

It is not possible to cast from spectral to CK types:

```
radCKType radcktype_instance = radCKType(radtype_instance); //won't compile
```

4.2 Wide band types

SMARTI wideband types are completely analogous to the spectral types. The main difference is the meaning and organization of the data within these types, which are:

- `wideNormType` (for normalized wideband data)
- `wideRadType` (for radiative wideband data)
- `wideNormCKType` (for normalized wideband correlated-k data)
- `wideRadCKType` for nradiative wideband correlated-k data)

Like the spectral types, these are split into normalized and radiative types, as well as band and CK types. In this case the band types simply represent a value that is constant over an entire band, and are mapped onto a grid representing the band extremities. The wideband CK types also contain the information on the band extremities, but instead of containing a single binned value between them, contains a set of 17 values mapped on a cumulative probability scale ranging from 0 to 1. In other words, wideband types all contain a single spectral bin representing the band and 1 or 17 CK values. Contrary to CK spectral types, CKs are in fact values and not bins. Data within these types can be accessed with the `[]` operator just like with spectral types, but the first index is always 0 (since there is only one spectral bin). The `getLength()` methods are also present. In any case, you should not need to access this internal data in most applications, since the `value()` function will allow you to obtain the band integrated value as a `float`.

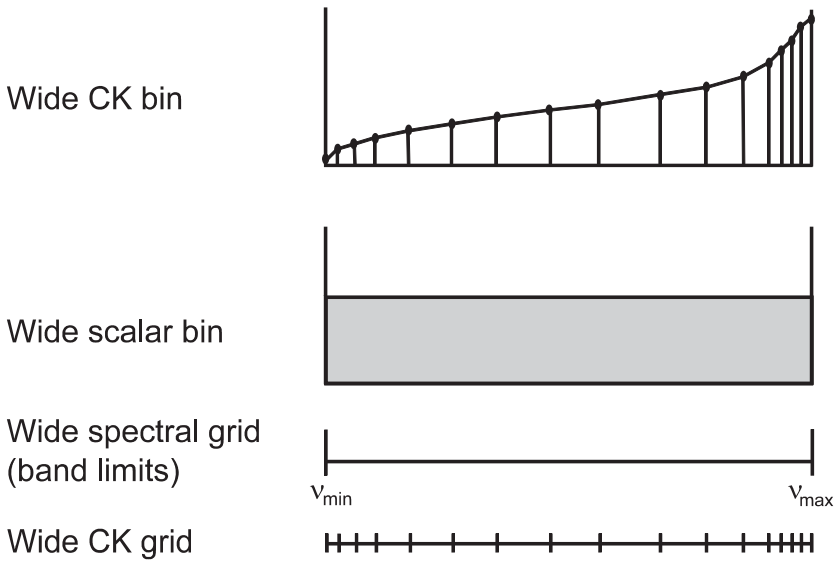

Figure 5: Representation of the different wide band quantities

Arithmetic operations and math functions can also be used between and on wideband types in exactly the same way as with spectral types. Note though that you cannot apply arithmetic operators between wideband and spectral types.

Just like spectral types, you can down-cast wide types from CK to band within the same category (radiative or normalized), but you can't up-cast from band to CK.

4.3 Paths

Some of the SMARTI outputs require that you first define a path trough the atmosphere. This is the case with path radiances, transmittances and backgrounds for example. Different ways of getting these paths exist in SMARTI and will be covered in section 12. For now, let's just say that these paths are stored in instances of type `pathOutputType`.

These can contain more than one path, as some methods that generate them trace multiple paths. On the other hand, a `pathOutputType` instance can be empty if the generating method could not find a path corresponding to your input (obviously these are not useful for radiative calculations!). To know how many paths are contained within a `pathOutputType` instance, you can call the `getLength()` method.

Paths within a `pathOutputType` instance can be accessed in two different formats: original and formatted (also called translated). The original path is as traced by the path tracing algorithm, is at its maximum resolution, and contains a number of methods for obtaining or interpolating quantities along

the path (altitude, slope, surface distance, geodesic coordinates, etc.). The translated path is formatted optimally for radiative quantities calculation and contains a different set of methods. In SMARTI, the translated path is always of type `translatedPath`, and the original path is always output as a `originalPath` type.

To obtain the original path, the `pathOutputType` has the `getOriginal(index)` method, while the translated path is obtained with the `getTranslated(index)` method, where `index` is the path number (from 0 to `getLength()-1`). Gladly, in most cases you won't need these, since the [] operator is also overloaded for the `pathOutputType` type, and the proper path type is automatically resolved with it is possible to do so without ambiguity. So say a method or function requires a `translatedPath`:

```
someMethod_trans(translatedPath trans_path);
```

And other method wants an `originalPath`:

```
someMethod_orig(originalPath orig_path);
```

And you want to use the first path in your `pathOutputType` instance `pathOutputType_inst` to call both methods. You can simply do:

```
someMethod_trans(pathOutputType_inst[0]);
someMethod_orig(pathOutputType_inst[0]);
```

Which would be exactly the same as doing:

```
someMethod_trans(pathOutputType_inst.getTranslated(0));
someMethod_orig(pathOutputType_inst.getOriginal(0));
```

Note that some methods in SMARTI require the entire `pathOutputType` directly as input. Some methods require a `pathElement` input. This is simply what is returned from the `pathOutputType[]` operator before it is automatically cast to an `originalPath` or a `translatedPath`. A `pathElement` object also has `getTranslated()` and `getOriginal()` methods (without arguments). Because of this, another equivalent to the above lines of code is:

```
someMethod_trans(pathOutputType_inst[0].getTranslated());
someMethod_orig(pathOutputType_inst[0].getOriginal());
```

4.3.1 Accessing Path data

While the `translatedPath` is formatted for use in the radiative transfer algorithms, and has many methods for these algorithms to use, it is not of much use to the SMARTI library user. The `originalPath` on the other hand has many methods that can give information on the shape and span of the path. Say you have an `originalPath` called `path`.

```
originalPath path = pathOutputType_inst.getOriginal(0);
```

or

```
originalPath path = pathOutputType_inst[0].getOriginal();
```

The number of points along the path that were calculated by the path tracing algorithm is obtained with:

```
size_t length = path.getLength();
```

Now, if `index` is any `size_t` value from 0 to `length`, you can retrieve any of these quantities:

- Travel distance (path arc length from start in meters):

```
float travel_distance = path.getTravelDistance( index );
```

- Surface distance (horizontal distance along the earth surface at sea level, in meters):

```
float surface_distance = path.getSurfaceDistance( index );
```

- Path slope (d(altitude)/d(surface distance)), also sine of path elevation:

```
float path_slope = path.getPathSlope( index );
```

- Path altitude in meters

```
float path_altitude = path.getAltitude( index );
```

- Path refractance (unitless):

```
float path_refractance = path.getRefractance( index );
```

- Path latitude of point along path (rad.):

```
float path_latitude = path.getLatitude( index );
```

- Path longitude of point along path (rad.):

```
float path_longitude = path.getLongitude( index );
```

- Path azimuth (heading) of point along path (east of north, rad.):

```
float path_azimuth = path.getAzimuth( index );
```

Analogous methods are also in place in case you need to interpolate these quantities in between points on the path, so say you want some information at 10 km surface distance (and you know the path extends further than that using the `getSurfaceDistance` method):

```
float surface_distance = 10000.0;
```

You can get the corresponding travel distance in meters with:

```
float travel_distance = path.interpTravelDistance( surface_distance );
```

Or the other way around (surface distance in meters from travel distance):

```
float surface_distance = path.interpSurfaceDistance( travel_distance );
```

All the other interpolation (`interp…`) methods are completely analogous to their `get…` counterpart, except that they take the travel distance (path arc length) in meters as an argument:

```
float path_slope = path.interpPathSlope( travel_distance );
float path_altitude = path.interpAltitude( travel_distance );
float path_refractance = path.interpRefractance( travel_distance );
float path_latitude = path.interpLatitude( travel_distance );
float path_longitude = path.interpLongitude( travel_distance );
float path_azimuth = path.interpAzimuth( travel_distance );
```

It can also be useful to get information on the tangent points along a path. These are the points where the slope is zero (the path is perfectly horizontal in respect to the local ellipsoid earth surface). These points also separate all monotonic segments of the path. The number of these points is retrieved using:

```
size_t getTangentLength() const;
```

while the index along the path corresponding to each tangent point is obtained with:

```
size_t getTangentIndex( size_t index ) const;
```

So obviously,

```
path.getPathSlope(getTangentIndex( index ) )
```

always returns 0.

The last useful method returns a `geodesic` object (see section 4.4.1) representing the geodesic position and referential of the sensor at the origin of the path:

```
const geodesic& getStartCoordinates() const;
```

4.4 Coordinates

Coordinates input into or output from SMARTI methods also have their own types. In cases where geodesic coordinates are required (coordinates describing position on the earth surface), the `geodesic` class is used. For coordinates on the sky dome (azimuth and elevation) from a point on the surface of the earth, the class `horizontalGeodesic` is instead employed.

4.4.1 The geodesic class

The `geodesic` class is used for representing geodesic coordinates (see Figure 6). In other words, it stores and makes available the longitude (ϕ) and geodesic latitude (ξ), as well as the altitude (H) above

the reference ellipsoid (which is also defined in the `geodesic` class with the values of its semi-major (a) and semi-minor (b) axes).

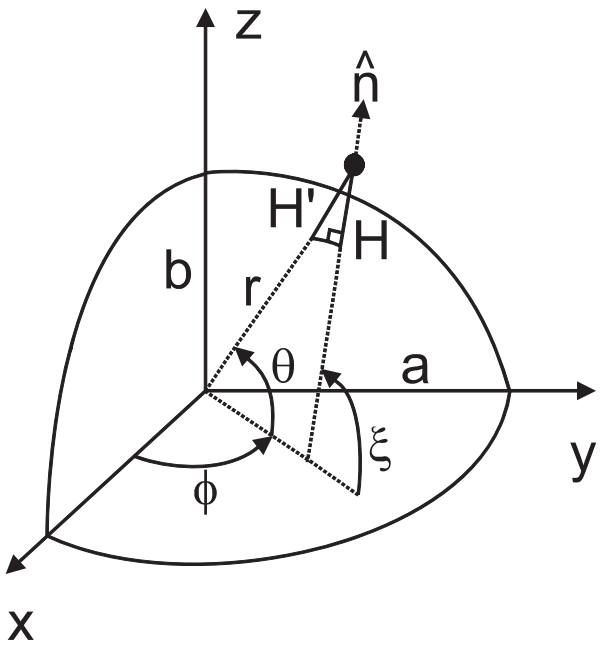

Figure 6: Geodesic coordinates schematic representation

A geodesic instance is constructed by initializing the coordinates and reference ellipsoid:

```
geodesic myCoordinates(
     46.8 * u_const::DEG_TO_RAD,     // Sensor geodesic latitude (in radians)
     -71.38 * u_const::DEG_TO_RAD,   // Sensor longitude (in radians)
     10.0,                           // Sensor geodesic altitude (in meters)
     WGS84::semi_major_axis,         // Earth semi-major axis (in meters)
     WGS84::semi_minor_axis          // Earth semi-minor axis (in meters)
     );
```

Where `u_const` is a SMART namespace containing many useful constants, the degrees to radians conversion in this case and `WGS84` contains the values for the standard World Geodetic System semi-major and semi-minor axes.

You can the query your `geodesic` instance to retrieve the longitude with the `phi()` method, the latitude with the `getXi()` and the altitude with the `getH()` method (all return double precision floating point values, `double`):

```
double myLongitude = myCoordinates.phi();
double myLatitude = myCoordinates.getXi();
double myAltitude = myCoordinates.getH();
```

4.4.2 The horizontalGeodesic class

Some SMARTI methods return `horizontalGeodesic` instances as output, although none require one as input so you will probably never need to construct one yourself from scratch. Because of this construction of a `horizontalGeodesic` is not covered here (see header file \include\utilities\coordinates\Coordinates.h for more detail).

Horizontal geodesic coordinates are a local (observer perspective) coordinate system defining the azimuth (clockwise from north, or "east of north"), elevation (from local horizontal) and distance (straight line from origin) of a point as seen from an observer (the origin) located at given geodesic coordinates.

Having an instance of the `horizontalGeodesic` class (named my`HorizontalGeodesic` in the following example), you can access the azimuth with the `getAzimuth()` method, the elevation with the `getElevation()` method and the distance with the `getDistance()` method (all return double precision floating point values, `double`):

```
double myAzimuth = myHorizontalGeodesic.getAzimuth();     // In radians
double myElevation = myHorizontalGeodesic.getElevation(); // In radians
double myDistance = myHorizontalGeodesic.getDistance();   // In meters
```

4.5 Radiative quantities container classes

When transmittance, radiance and background are all required, or if any are required at multiple points along a line of sight, radiative quantities container classes are returned from appropriate SMARTI methods (see section 13.3). If spectral quantities are required, a `smartiSpectralOutput` object is returned, and for wide band CK output, the `smartiWideBandOutput` class is used for output. Both classes are similar, but have subtle differences which are discussed in the following sections.

4.5.1 The smartiSpectralOutput container class

Just as `pathOutputType` objects can contain more than one path, the `smartiSpectralOutput` class can be used to return results along multiple lines of sight. In order to obtain the number of lines of sight along which radiative quantities have been calculated for a given `smartiSpectralOutput` object, you can use the `getOutputNumber` method. This method has no arguments and returns a `size_t` integer representing the number of path radiative results. Most `smartiSpectralOutput` methods require an `output_number` index argument, which can range from 0 to `getOutputNumber()`-1.

All radiative quantities can be interpolated at a distance along the path. Trying to obtain any radiative quantity at a distance greater than the maximum path length will always produce the end of path result (for path radiance, this does not include the ground radiance). Because of this, it can be useful to now the maximum distance for a given `output_number` index. You can get this information with the `getMaxDistance` method which takes one argument:

`size_t output_number`	Index of the path at which to get the maximum distance.

You can obtain the spectral CK transmittance interpolated anywhere along a path by using the `getCKTransmittance` method with parameters

`size_t output_number`	Index of path to query (0 to getOutputNumber() - 1)
`float travel_distance`	travel distance along path (0 to +Inf). Defaults to the end of the path if larger than `getMaxDistance(output_number)`.

The `getCKTransmittance` method returns a `normCKType` object. To obtain a non-CK path transmittance of type `normType`, one can also use the `getSpectralTransmittance` method, with exactly the same parameters. Calling this method is in fact exactly the same as casting the result of `getCKTransmittance` to a `normType`:

```
// Here, res1 and res2 will be the same
normType res1 = getSpectralTransmittance (output_number, travel_distance);
normType res2 = normType(getCKTransmittance(output_number, travel_distance));
```

but is nonetheless provided for purposes of clarity and simplicity.

Obtaining path radiance anywhere along a path is done with methods `getCKRadiance` and `getSpectralRadiance` which return a `radCKType` and `radType` respectively. They have the same inputs as their transmittance counterparts and work the same way.

Methods for obtaining the background radiance (including the ground contribution if intercepted by the path) are also provided. These are `getCKBackground` and `getSpectralBackground` which return a `radCKType` and `radType` respectively. Both take in a single parameter:

`size_t output_number`	Index of path to query (0 to `getOutputNumber() - 1`)

4.5.2 The smartiWideBandOutput container class

The `smartiWideBandOutput` class is very similar to the `smartiSpectralOutput` class. It has a `getOutputNumber` and a `getMaxDistance` method, as well as `getCKTransmittance` and `getBandTransmittance` (instead of `getSpectralTransmittance`) methods for obtaining transmittances and `getCKRadiance` and `getBandRadiance` (instead of `getSpectralRadiance`) methods for path radiance. The difference is the inputs of the radiance and transmittance methods, which are:

`size_t output_number`	Index of path to query (0 to getOutputNumber() - 1)
`output_modes mode`	Output mode (`solar`, `thermal` or `both`). Option `both` is only available for radiance calculations.
`= solar`	Only the solar contributions are output in the solar CK space.

= thermal	Only the thermal contributions (emitted and scattered) are output in the thermal CK space.
= both	Both solar and thermal contributions are summed into a single CK distribution/value before output. Note that this option is not available for transmittance and would result in an exception being thrown.
float travel_distance	travel distance along path (0 to +Inf). Defaults to the end of the path if larger than getMaxDistance(output_number).

Like with the smartiSpectralOutput class, methods for obtaining the background radiance (including the ground contribution if intercepted by the path) are also provided. These are getCKBackground and getBandBackground which return a wideRadCKType and wideRadType respectively. Both take in two parameters:

size_t output_number	Index of path to query (0 to getOutputNumber() - 1)
output_modes mode	Output mode (solar, thermal or both).
= solar	Only the solar contributions are output in the solar CK space.
= thermal	Only the thermal contributions (emitted and scattered) are output in the thermal CK space.
= both	Both solar and thermal contributions are summed into a single CK distribution/value before output.

Note that for all methods, you should only use mode both for final results, since for repropagation (after say a reflection off a surface, etc.) you need to remain in the solar and thermal CK spaces until the end.

5 Creating a SMARTI instance

Creating a SMARTI (class smarti) instance requires only that you define the global spectral band and wave number resolution. It is constructed with the following parameters (read on for more on the types used) :

float low_spec	Lower spectral boundary (min is 0.2 micron or 250 cm-1). Default if omitted is 0.4.
float high_spec	Upper spectral boundary (max is 40.0 micron or 50000 cm-1). Default if omitted is 20.0.

`spec_units units`	Spectral units, `wavelength` (microns) or `wavenumber` (cm-1). Default if omitted is `wavelength`. See following paragraphs for the definition of `spec_units`.
`resolution res`	Spectral resolution (always cm^{-1}). Default if omitted is `res5` (5 cm^{-1}). See following paragraphs for the definition of `resolution`.

All sensors associated with the `smarti` instance will share the same spectral band, and will differ only by their response function and position. Because of this a `smarti` instance usually regroups similar sensors, while another different group of sensors might be added to a different `smarti` instance:

```
// Initialize a smarti instance to regroup midwave infrared sensors
smarti smart_MW(
            3.0f,                       // Band minimum at 3.0 microns
            5.0f,                       // Band maximum at 5.0 microns
            wavelength,                 // Units for min and max input
            smarti::res15               // Use 15 cm-1 band model resolution
            );

// Initialize a smarti instance to regroup longwave infrared sensors
smarti smart_LW(
            8.0f,                       // Band minimum at 8.0 microns
            12.0f,                      // Band maximum at 12.0 microns
            wavelength,                 // Units for min and max input
            smarti::res5                // Use 5 cm-1 band model resolution
            );
```

Where the input for spectral units can be `wavelength` or `wavenumber`, and only informs the constructor of the units for the band minimum and maximum (SMARTI spectral calculations and output are always on a wave number grid). The resolution argument can be either `smarti::res1`, `smarti::res5` or `smarti::res15` for $1cm^{-1}$, $5cm^{-1}$ or $15cm^{-1}$ moderate band model resolution. This resolution is used for spectral calculations, and is the internal coherence resolution for wide band CK calculation.

6 Configuring the SMARTI instance

6.1 Setting up MODTRAN location and directories (setModtran)

SMARTI uses the officiel MODTRAN® 4 version 3 release 1 or a modified and recompiled version of MODTRAN® 4 or 5 in order to calculate some of the optical coefficients (molecular and aerosols depending on the chosen parameters) describing the atmosphere. Because of this, your `smarti` class instance must know where to find the MODTRAN® PC executable. You can also tell SMARTI what directory MODTRAN® should use for input/output, the case name (name assigned to the input/output

files) and the name of the executable if you changed it for some reason. This is all done with the `setModtran` method with parameters:

`char* modRoot`	MODTRAN root directory path. This is the full path to the directory that contains the MODTRAN executable and all MODTRAN subdirectories.
`char* modIOdir`	MODTRAN input/output directory name. Omitting this parameter default in using the existing MODTRAN "TEST" directory.
`char* modCase`	Case name. This is the name that will be given to the MODTRAN input and output files located in `modIOdir`. For example `"case"` will result in MODTRAN files "case.tp5", "case.tp6", etc. Omitting this parameter results in default value `"driver_output"`.
`char* modExe`	MODTRAN 4v3r1 official executable or modified version. Normally, the official executable should be called `"Mod4V3R1.exe"` so this is the default value if you omit this parameter.

For example:

```
/// Set up MODTRAN case name and location
smarti_inst.setModtran(
        "C:/MOD4V1R3",          // MODTRAN root directory path
        "smartIO",              // MODTRAN input/output directory name
        "driver_output",        // Case name
        "Mod4V3R1.exe"          // MODTRAN 4v3r1 official executable name
        );
```

Note that the input/output directory that you specify (`"smartIO"` here) should already exist on your computer, since it will not be created for you.

6.2 Setting optimization flags

6.2.1 Memory optimization (memOptim)

The `smarti` class is designed to keep as much information as possible into memory to minimize the time required to call the update method (see section 10). In some cases, you might intend on creating your `smarti` instance, and using it for a very large number of calculations before re-updating it with new parameters (if ever). In these instances, it might be beneficial to clear some intermediate quantities from memory to free up resources. In order to do that, you can call the `memOptim` method with argument:

`bool flag`	Set to true to enable memory optimization (but slow down subsequent updates), and false to disable it (default).

In this form, `memOptim` does not return anything. Calling `memOptim` with no arguments will return the current state (`bool` value).

Note that this flag takes effect as soon as it is set, but memory will only be freed/optimized on the next call to `update()`.

6.2.2 Only wideband mode (onlyWide)

Many spectral quantities are pre-calculated by SMARTI to speed up multiple spectral calculations. If you do not intend on doing any spectral calculations with the `smarti` instance (only wideband CK), set your `smarti` instance in "only wide" mode. All unecessary spectral information will either not be calculated or discarded on the next call to the `update()` method (see section 10). All spectral methods (most start with getSpectral...) will throw an exception if mode is "only wide". To do this, call the `onlyWide` method with single argument:

`bool flag`	Set to true to enable only wideband CK calculations mode, and false to disable it (default).

In this form, `onlyWide` does not return anything. Calling `onlyWide` with no arguments will return the current state (`bool` value).

Note that this flag takes effect only on the next call to update, so before calling `update()`, `onlyWide` without an argument will return the previous (still active) state.

7 Modifiers

Most of the parameters needed to launch radiative transmission calculations are set to default values when you create a `smarti` instance. There is a very good chance though, that these default values and models are not suitable for your particular case. On most occasions, you will want to set up your `smarti` instance with the wide range of modifiers, or "set" methods that are available. With these methods you can select and parameterize meteorological profile and aerosol models, chose the position of the sun and moon, add sensors and move them around, etc. The following sections will explain how to do all of that and more.

The last call to any modifier for a given class of model or parameters (meteorological, aerosol, etc.) has precedence on any call made previously in the same modifier group for a given `smarti` instance. For example, the generic example (these are not actual `smarti` methods):

```
smarti_inst.setSomeMeteoModel(2,1);
smarti_inst.setSomeAerosolModel(8);
smarti_inst.setSomeOtherMeteoModel(4);
```

```
smarti_inst.setSomeOtherAerosolModel(5,3);
```

results in the `smarti` instance `smarti_inst` being set with meteorological model "Other Meteo Model" with parameter "4" and aerosol model "Other Aerosol Model" with parameters "5" and "3".

It is very important to note that any modifier called will not take effect in the calculation of radiative quantities until you call the `update()` method (see section 0). Some modifiers have an immediate effect on the behavior of the `smarti` instance; these will be explicitly described below.

7.1 Meteorological models

Meteorological models define the temperature, pressure and water content profile of the atmosphere. The profiles can be determined from databases, of from measurements (radiosonds for example).

7.1.1 The MODTRAN meteorological profiles (setModtranMeteo)

The easiest way to set up a meteorological profile is by selecting one of the MODTRAN profiles by using the `setModtranMeteo` method. This method takes a single integer (`model`) as argument, ranging from 1 to 6:

`int model`	MODTRAN atmosphere model to use from ground to top of atmosphere
= 1	Tropical atmosphere (representative of 15° North latitude)
= 2	Mid-latitude summer (representative of 45° North latitude in summer)
= 3	Mid-latitude winter (representative of 45° North latitude in winter)
= 4	Sub-Arctic summer (representative of 60° North latitude in summer)
= 5	Sub-Arctic winter (representative of 60° North latitude in winter)
= 6	1976 US standard atmosphere

So if we want to set the mid-latitude summer atmosphere for a `smarti` instance called `smarti_inst` we simply do:

```
smarti_inst.setModtranMeteo(2);
```

Any integer outside the 1 to 6 range will cause an exception when you try to update the `smarti` instance.

7.1.2 The DRDC meteorological models (setDrdcMeteo)

The DRDC meteorological profile [3] model is designed to model the lower atmosphere (close to the ground or sea surface) more accurately in situations where you have measurements. This might be important for many reasons since it will affect some aerosol models, the accuracy of refracted path calculations and the results of radiative calculations over all (especially if your sensor or target is located within the atmosphere's boundary layer). You can set the DRDC meteorological model with the `setDrdcMeteo` method with the following parameters:

`float measHeight_TPR`	Measurement height (m) for temperature, pressure and relative humidity. Validity is from 2 to 40 m.
`float Temperature`	Temperature at measurement height (C), from -40 to 40 °C.
`float Pressure`	Pressure at measurement height (mbar), from 800 to 1200 mbar.
`float RelativeHumidity`	Relative humidity at measurement height (%), from 0 to 100%.
`int ulRsType`	Type of radiosonde file. Types 1 to 6 are the MODTRAN atmospheres defined to the top of the atmosphere and require no additional input. Types 7 to 11 require the input of a radiosonde file in the appropriate format in parameter `ulRsFile`.
= 1	Tropical atmosphere (representative of 15° North latitude)
= 2	Mid-latitude summer (representative of 45° North latitude in summer)
= 3	Mid-latitude winter (representative of 45° North latitude in winter)
= 4	Sub-Arctic summer (representative of 60° North latitude in summer)
= 5	Sub-Arctic winter (representative of 60° North latitude in winter)
= 6	1976 US standard atmosphere
= 7	TEMP radiosonde format (requires `ulRsFile` input)
= 8	US radiosonde format (requires `ulRsFile` input)
= 9	Standard radiosonde format (requires `ulRsFile` input)
= 10	CA radiosonde format (requires `ulRsFile` input)

= 11	HPAC radiosonde format (requires `ulRsFile` input)
`char* ulRsFile`	Path to upper layer radiosonde file (for ulRsType == 7 to 11)
`int default_atm`	Atmosphere to use above radiosonde (MODTRAN atmospheres, 1 to 6)
`float sTemperature`	Surface (altitude = 0) air temperature (C). Defaults to same value as `Temperature` if omitted. Validity from -10 to 40°C. Note that absolute difference between parameters `Temperature` and `sTemperature` cannot exceed 10°C.
`float sRelativeHumidity`	Surface (altitude = 0) air relative humidity. If omitted the value defaults to 98.2% which is typical for marine atmosphere. Valid from 0 to 100%.

7.1.3 Wind speed (setWind)

Surface wind speed and direction can be important factors in some aerosol and surface BRDF models. To set the surface wind speed, use the `setWind` method with parameters:

`float measHeight_W`	Measurement height for wind speed (m). Validity is from 2 to 40 m.
`float windSpeed`	Wind speed at measurement height (m/s). Validity is from 0.1 to 30 m/s.
`float windSpeed24`	Wind speed average in last 24 hours at measurement height (m/s). Validity is from 0.1 to 30 m/s.
`float windDir`	Current wind direction (East of North) (rad).

7.2 Aerosol models

The `smarti` class contains a public enumerator to help you select the proper aerosol model:

```
// Available boundary aerosol models
enum aeroModels{
    rural,      // MODTRAN rural aerosol model (23 km visibility default)
    urban,      // MODTRAN urban aerosol model
    maritime,   // MODTRAN maritime aerosol model
    modnam,     // MODTRAN nam aerosol model
    tropo,      // MODTRAN tropospheric aerosol model
    advfog,     // MODTRAN advective fog aerosol model
    radfog,     // MODTRAN radiative fog aerosol model
    desert,     // MODTRAN desert aerosol model
    drdcmarine  // DRDC maritime aerosol models
```

```
};
```

So if an aerosol modifier calls for a input parameter of type `aeroModels`, and you want to input the rural MODTRAN model for example, you can just use `smarti::rural`.

7.2.1 MODTRAN aerosol models (setModtranAerosols)

With SMARTI you have access to all the MODTRAN boundary aerosol models. To use and parameterize one of these models, use the `setModtranAerosols` method with parameters:

`aeroModels model`	Model (any `aeroModels` except `drdcmarine`, to use `drdcmarine` use the setDrdcMarineAerosols method described in section 7.2.2)
`float visibility`	Koschmieder visibility (km), 0 for model default. If omitted, defaults to 0. The MODTRAN Rural aerosol model uses 23 km visibility by default, to use the 5 km visibility model you must set the visibility yourself.
`int amp`	Air mass parameter, ICSTL in MODTRAN (1=open ocean, 10=strong continental). If omitted, defaults to 3.

Note that wind speed affects the NAM and desert aerosols, 24h average wind speed affects the NAM aerosols, and relative humidity affects all aerosols except the fog models.

7.2.2 DRDC maritime aerosol models (setDrdcMarineAerosols)

The DRDC maritime aerosol models [4] are based on a modified version of the NOVAM algorithm (Navy Oceanic Vertical Aerosol Model) in order to calculate the vertical distribution of aerosols in the atmosphere boundary layer. The optical properties of the aerosols can be based on four different models, KEL (developed by KEL Research), NAM (Navy Aerosol Model), MEDEX (Mediterranean Experiment) and ANAM (Advanced Navy Aerosol Model). The original NOVAM is only based on NAM. To use the DRDC models, use the `setDrdcMarineAerosols` method with parameters:

`float fetch`	Wind fetch (km). This is the distance wind has traveled over the sea surface since leaving the coastline. If omitted, defaults to 120 km. Validity is from 3 to 120 km.
`float visibility`	Visibility (km) set to 0 for model default (calculated from wind speed and relative humidity). If omitted defaults to 0. Validity is from 0 to 200 km.
`int visType`	Visibility type. If omitted defaults to Koschmieder visibility (3).
= 1	Observer visibility

= 2	Instrument visibility
= 3	Koschmieder visibility
`int amp`	Air mass parameter (1=open ocean to 10=strong costal). If omitted defaults to 3.
`int model`	Marine aerosol model. If omitted defaults to 2 (NAM based).
= 1	KEL model based
= 2	NAM model based
= 3	MEDEX model based
= 4	ANAM model based
`int fogType`	Fog type. If omitted defaults to 0 (no fog).
= 0	No fog
= 1	Advective fog
= 2	Radiative fog
`float fogLWC = 0.0`	Fog liquid water content (g/m^3). Only used if fog model is chosen (`fogType` = 1 or 2). If omitted defaults to 0 g/m^3. Validity is from 0 to 10 g/m^3

This model is affected by meteorological profile and surface wind speed (current and 24h averaged).

7.3 Cloud, rain and snow models (setCloudAndPrecipitation)

The cloud and rain models used in SMARTI are those also found in MODTRAN (single layer cloud models), while snow is modeled using the DRDC-Valcartier snow model implementation. As in MODTRAN rain profiles decrease linearly from the ground to the top of the cloud when cloud model is 5 to 10 and rain rate is not specified. When rain rate is specified, it is used to the top of the cloud. Snow rate is constant up to the bottom of the cloud, and decreases linearly, reaching 0 at the top. To add clouds and optionally rain and snow, use the `setCloudAndPrecipitation` method with parameters:

`int cldModel`	MODTRAN cloud/rain model
= -1	Auto determine from total precipitation rate
= 0	No cloud or rain

= 1	Cumulus cloud layer
= 2	Altostratus cloud layer
= 3	Stratus cloud layer
= 4	Stratus/stratocumulus layer
= 5	Nimbostratus layer
= 6	2.0 mm/h ground drizzle with stratus cloud
= 7	5.0 mm/h light rain with nimbostratus cloud
= 8	12.5 mm/h moderate rain with nimbostratus cloud
= 9	25.0 mm/h ground heavy rain with cumulus cloud
= 10	75.0 mm/h ground extreme rain with cumulus cloud
`float rainrate`	Optional rain rate (mm/hr), RAINRT in MODTRAN, set to >0 to modify or add to `cldModel`, set to 0 to keep `cldModel` default. Default is 0 if omitted. Validity is from 0 to 100 mm/hr.
`float snowfallrate`	Snowfall rate, 0 for none (cm/hr). Default is 0 if omitted. Validity is from 0 to 100 cm/hr.
`int snowtype`	Snowflake type (0 to 6). Default is 0 if omitted.
= 0	Auto determine from temperature and relative humidity.
= 1	Thin plate/plate snowflake morphology.
= 2	Hollow column snowflake morphology.
= 3	Needle/sheath snowflake morphology.
= 4	Thick plate/very thick plate/column snowflake morphology.
= 5	Sector plate snowflake morphology.
= 6	Dendrite snowflake type.
`float baseAltitude`	Base altitude (km, -9 for model default). If omitted, defaults to -9.
`float thickness`	Cloud thickness (km, -9 for model default) . If omitted, defaults to -9. Value should be larger or equal to 0.001.

`float extinction` — Extinction coefficient of cloud (km-1). If omitted, defaults to -9. Value should be smaller than 9999.

If `cldModel` contains rain, and snow is also used (`snowfallrate > 0`), snow and rain will be mixed in final optical profile. If `cldModel` is -1, and snowfall and/or rain rate is > 0, model is automatically determined from the total precipitation rate, else it is set to 0 (no clouds).

Default values (when -9 is used for `baseAltitude`, `thickness` or `extinction`) for the different clouds are:

Model	Base Altitude (km)	Thickness (km)	Extinction (km^{-1})
Cumulus (1)	0.66	2.34	92.6
Altostratus (2)	2.40	0.60	128.1
Stratus (3)	0.33	0.67	56.9
Stratus/stratocumulus (4)	0.66	1.34	38.7
nimbostratus (5)	0.16	0.50	92.0

The altitude where the cloud extinction falls to zero is usually higher than `baseAltitude + thickness` because of a falloff region. If the final altitude of the cloud profile as calculated by MODTRAN is higher than the top of the atmosphere, an exception will be thrown. This is normally not an issue with values of `baseAltitude` and `thickness` representing real clouds found in the earth's atmosphere.

7.4 Sun and moon

The sun and moon play important roles in the calculation of radiances, irradiances and fluxes, especially in the UV, visible and mid-infrared bands (from 0.2 to 5 microns). SMARTI possessed many different ways to parameterize the sun and moon.

7.4.1 Sun and moon position

There are different ways to determine the location of the sun and moon. Each of these modifiers positions the sun and moon around the earth (geocentric referential), so each point in the scene can potentially have a different sun and moon apparent coordinate in the sky.

7.4.1.1 From time (setSkyFromTime)

The first method for positioning the sun and moon around the earth is by using a user defined date and time. To do so use the `setSkyFromTime` method with the following parameters:

`int year` — Year (ex. 2008)

`int month` — Month (ex. 11 for November)

`double day`	Day of the month (decimal or integer, if hour is not integer, don't specify hour, minutes and seconds)
`double hour`	Hour (decimal or integer, if hour is not integer, don't specify minutes and seconds)
`double minutes`	Minutes (decimal or integer, if hour is not integer, don't specify seconds)
`double seconds`	Seconds (decimal)
`int timeZone`	Offset from Universal Time (watch out for daylight savings!). If omitted, defaults to 0.

7.4.1.2 From Julian date (setSkyFromJD)

Instead of using ordinary date and time values, you can also set the position of the sun and moon from the Julian date using the setSkyFromJD method and its parameter:

`double julian_date`	Julian date

7.4.1.3 From system time (setSkyFromNow)

When you want to model the current state of the atmosphere, and your system has the proper date and time set, you can simply use the `setSkyFromNow`, which has no input parameters.

7.4.1.4 From viewer perspective (setSkyFromView)

Even if all you have are observer accounts of the position of the sun and/or moon in the sky, without knowledge of the date and time, you can use the `setSkyFromView` method with parameters:

`const geodesic& obs_position`	Geodesic position of observer. See section 4.4 for more on the `geodesic` class.
`float sun_azimuth`	Sun Azimuth, east of north (rad)
`float sun_elevation`	Sun elevation above local horizontal (rad), from $-\pi/2$ to $\pi/2$.
`float moon_azimuth`	Moon Azimuth, east of north (rad)
`float moon_elevation`	Moon elevation above local horizontal (rad) , from $-\pi/2$ to $\pi/2$.
`float sun_distance`	Optional viewer to sun distance (m). If omitted, defaults to 1 astronomical unit (1 AU = 149.597870691e9 meters).
`float moon_distance`	Optional viewer to moon distance (m). If omitted, defaults to 384400000 meters.

Tip: if you only want to set the sun (or moon) position without worrying about the moon (or sun), because you know it is the dominant source, or have no knowledge of the other's position, simply set the elevation of the other to $-\pi/2$.

7.4.2 Obtaining the sun or moon geometry (getSunCoordinates, getMoonCoordinates and getMoonPhase)

As soon as you have called one of the `setSky` methods described above, it is possible to retrieve the *unrefracted* position of the sun and moon from the perspective of a viewer at some given coordinates, as well as the phase of the moon. In other words, the `update()` method (see section 0) does not need to be called in order for you to get to that information.

To get the sun *unrefracted* elevation and azimuth in the sky, as seen at a given coordinate on the surface of the earth, you can use the `getSunCoordinates` method while the moon coordinates are obtained with the `getMoonCoordinates` method. The moon phase can also be obtained with the `getMoonPhase` method. All three methods have the following parameters:

`const geodesic& position`	Geodesic coordinates of viewer where the elevation and azimuth are calculated. This does not need to correspond to any sensor location in the scene (see section 4.4.1 for information on the `geodesic` class).
`bool current`	Flag to get the current (as set by the last `setSky` method) or last updated position. Immediately after calling the `update()` method, parameter `current` has no effect until you call a `setSky` method again. Omitting this parameter defaults to `true`.

Both `getSunCoordinates` and `getMoonCoordinates` return an object of type `horizontalGeodesic` containing the elevation and azimuth of the sun or moon. For more information on the `horizontalGeodesic` class see section 4.4.2. The `getMoonPhase` method simply returns a floating point value (`float`) corresponding to the moon phase, where $-\pi$ is new, 0 is full, π - epsilon is the last crescent.

7.4.3 Illuminator (setIlluminator)

You are responsible for telling SMARTI which of the sun or moon is the illuminator for your scene, as SMARTI won't determine for you which one is the dominant source. You can, on the other hand use the `getSunCoordinates` and `getMoonCoordinates` methods described previously to set your own criteria. To chose the illuminating source in your scene use the `setIlluminator` method the argument

`bool sun`	Set flag to true for solar illumination, and to false for lunar illumination.

In the current version, all sensors in the scene share the same illuminator option, so your choice affects all calculations for a given `smarti` instance.

7.5 Scattering and irradiance mode control

The SMARTI provides the user ways to control the way scattering source functions are calculated. The combination of irradiance mode, scene scattering mode and scattering approximation parameters will affect the update (initialization) time, the computation time and even the accuracy of the results. The modes, and how the affect the `smarti` instance are explained in the following sections.

7.5.1 Irradiance mode (setIrradianceMode)

The irradiance mode determines how the solar/lunar irradiance is calculated. There are two options to choose from: 1D pre-calculation or 2D on-the-fly calculation.

In the 1D pre-calculation approach (see Figure 7), the top of atmosphere solar/lunar irradiance is transmitted to each atmospheric layer boundary directly above given coordinates. The transmitted irradiances are then use as a look-up table in atmospheric radiance calculations, no mater the coordinate of the scattering point; only the altitude is considered. This requires a bit more time when updating the `smarti` instance, but is much faster when it comes to doing path radiance calculations.

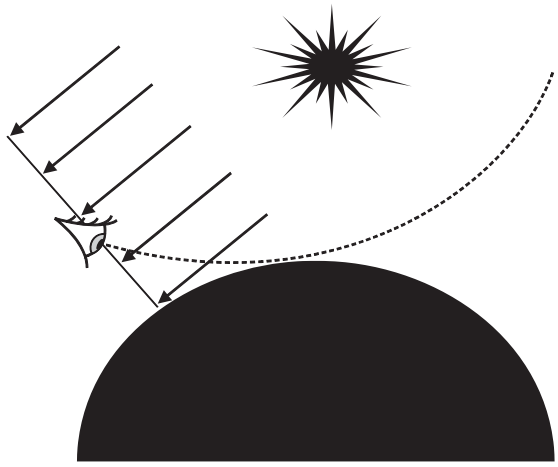

Figure 7: Schematic representation of the 1D solar source mode

The 2D on-the-fly calculation mode on the other hand (see Figure 8), does not do any irradiance pre-calculations. Instead it recalculates the irradiance at each scattering point along each path. This obviously requires more time at computation, but can be more accurate. The increase in accuracy is mostly noticeable for near horizontal lines of sight and/or when the sun/moon is low on the horizon or has set from the perspective of the sensor.

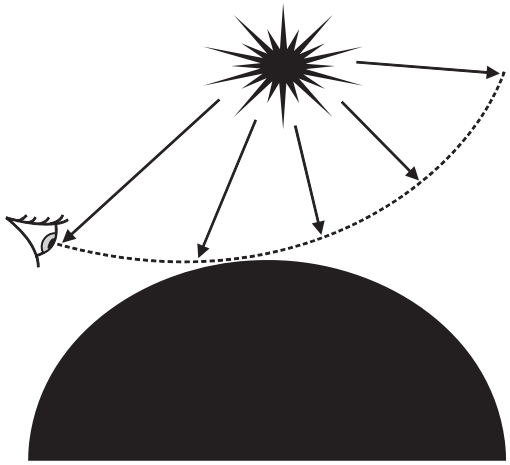

Figure 8: Schematic representation of the 2D solar source mode

You can set the irradiance mode using the `setIrradianceMode` method with parameter:

`int irrtype`		Solar/lunar irradiance calculation mode
=	1	1D pre-calculation mode
=	2	2D on-the-fly calculation mode

7.5.2 Scene scattering mode (setSceneScattering and setSensorScattering)

The scene scattering mode affects how 1D source functions (single and multiple scattering) are calculated. The 1D source functions can be at the scene level (global) or sensor level.

In scene level mode calculations all 1D source functions like multiple scattering sources or single scattering sources in the 1D pre-calculation mode (see previous section) are pre-calculated only once at a user specified location. This location is usually the center of the scene. Moving the sensors has no effect on the source function calculations, and nothing needs to be recalculated on calling the `update` method. You should use this mode when all sensors are located in a relatively constrained space (sun/moon coordinates don't change much from one sensor to the other)

In sensor level mode calculations, 1D source functions are re-calculated at each of the sensor's coordinates (with appropriate solar/lunar geometry). Calculations using each sensor in the scene then use a different source function calculator. Moving sensors will trigger re-calculation of the source functions at the next call to the `update` method. This mode thus requires more memory and time to update the `smarti` instance, but is more accurate when sensors re very far apart.

Obviously, scene scattering mode will not affect 2D on-the-fly irradiance calculation mode, nor will it affect thermal calculations since the later are always the same for all sensors.

To set the scattering to scene level mode, use the `setSceneScattering` method with argument:

`const geodesic& location`	Location at which sun position is calculated for all sensors (see section 4.4.1 for information on the `geodesic` class).

While setting the sensor scattering mode requires using the `setSensorScattering` method, which has no parameters.

7.5.3 Scattering approximation mode (setScatteringMode)

The scattering approximation mode affects the accuracy (and speed) of the scattering calculations. Setting the scattering mode to single scattering is faster but less accurate, while enabling multiple scattering will be more accurate but will inevitably slow down the calculations (more or less depending on the number of "streams" you specify). You can specify the scattering approximation mode with the method `setScatteringMode` with single parameter:

`int nbStreams`	Number of streams to consider in the scattering approximation. Set to 1 for single scattering and to 2 for two-flux multiple scattering. In wideband CK only mode (see section 6.2.2), you can also set this parameter to 4, 8 or 16, which will result in more accurate DISORT (discreet ordinate) calculations.

Note that in single scattering mode, background radiance when intercepting the ground only includes thermal emissions and direct solar/lunar reflections. Sky reflected component is only included in multiple scattering mode.

7.6 Surface properties

The nature of the ground surface will affect background radiances as well as atmospheric fluxes and multiple scattering calculations as it acts as a lower boundary condition. There are currently two types of surface reflectance models implemented in SMART: Lambertian (isotropic) and wind ruffled water.

7.6.1 Setting the lower surface as Lambertian (setLambertSurface)

A Lambertian surface is defined by its spectral albedo. To set the lower surface as Lambertian, use the `setLambertSurface` method with parameters:

`int nb_pts`	Number of points in albedo data (at least two)
`float* spectral_grid`	Spectral grid of albedo data (microns)
`float* albedo`	Albedo data (unitless). Albedo values are normally bounded from 0 to 1.

Note that if the input spectral grid does not cover the entire `smarti` instance spectral band, the end values (first and last) of the `albedo` vector are used for the rest of the band.

7.6.2 Setting the lower surface as wind ruffled water (setSeaSurface)

In SMARTI, water surfaces are described by a statistical micro-facet model developed at DRDC-Valcartier [5,6]. It depends mainly in the slope variances used to describe the roughness of the surface. The variances can be obtained from any of three models that all depend on the wind speed at 10 meters altitude. To set the lower surface as wind ruffled water, use the `setSeaSurface` method with the following parameter:

`int seatype`	Model to use to calculate the surface slope statistics. If parameter is omitted, default is 1 (Cox and Munk model).
= 1	Use Cox and Munk sea surface slope statistics model. Only depends on wind speed.
= 2	Use Shaw and Churnside (Cox and Munk with atmospheric stability dependence) sea surface slope statistics model. Depends on the wind speed and the air-water temperature difference.
= 3	Use surface slope statistics derived from the Elfouhaily et. al power spectrum. Depends on the wind speed, wind fetch, and atmospheric stability condition (air-water temperature difference).
`float wave_fetch`	Travel distance of wave generating wind over sea in (km). If omitted, 120 km is used. This parameter is only used for the Elfouhaily model (`seatype` = 3). Valid from 0 to infinity.

7.6.3 Setting the lower surface temperature (setGroundTemperature)

The lower surface temperature affects the thermal emissions from the sea or ground. If your path intercepts the ground, the background radiance might include a blackbody function contribution emanating from the surface. Even if the path does not hit the ground, this might be important if multiple scattering is enabled (see `setScatteringMode`, section 7.5.3) since ground thermal emissions might be scattered towards the sensor. To change the lower surface temperature, use the `setGroundTemperature` method with the following parameter

`float Temperature`	Lower surface (ground or sea) temperature (in ^{o}C).

Note that if you don't call this method, or set temperature to a value smaller than -273.15 °C, ground/sea temperature defaults to the temperature of the air just above the ground.

7.6.4 The soft ground boundary offset (setSoftGroundAltitude and getSoftGroundAltitude)

In SMARTI, you can set a soft boundary altitude at which optical paths will be clipped before reaching 0m (ground level). The lower surface BRDF is then considered to be at this soft boundary altitude. Note that the atmosphere is always defined starting from 0, and will be truncated at the soft altitude in path tracing and ratiative transfer methods. Atmospheric fluxes also start at 0m.

You can use this altitude in order to simulate the top of three canopy, the tops of sea waves or even a cutoff altitude within a thick cloud layer seen from above for example.

To set this altitude, use the `getSoftGroundAltitude` method, with single parameter

`float altitude`	Altitude of the ground offset above reference ellipsoid or 0 (m).

Input altitude must be a positive number (>= 0). You can also retreive this altitude with the `getSoftGroundAltitude` taking no arguments and returninf a `float` value.

Note that all sensors must be above this altitude. Trying to use a sensor that is below this altitude in a path tracing method will result in an exception being thrown.

8 Grids

8.1 Getting spectral grid points from the `smarti` instance (getSpectralLength and getSpectralValue)

As soon as you create a `smarti` instance you have access to the spectral grid on which all will be calculated. To get the length of the spectral grid (number of spectral bin boundaries, see section 4.1), you can call the `getSpectralLength` method which takes no arguments and returns a `size_t` type value. Remember, the number of spectral data points in vectors returned by your `smarti` instance will be is `getSpectralLengtht() - 1`.

To get the actual grid values, you then use the `getSpectralValue` method, taking as an argument:

`size_t index`	Index of the spectral grid bin boundary to return. Can take values from 0 to `getSpectralLengtht() - 1`.

The `getSpectralValue` method returns a `float` value representing the spectral grid bin boundary (with cm^{-1} units) at `index`.

8.2 Getting information on the altitude grid (getAltitudeLength and getAltitude)

After you update your `smarti` instance at least once you can have access to information on the altitude grid that will be used in the radiative transmission calculations. The number of altitude levels is obtained with the `getAltitudeLength` method which takes no arguments and returns a `size_t` type value.

The actual altitude levels automatically calculated by SMARTI are obtained using the `getAltitude` method with argiment:

`size_t index`	Index of the altitude level to return. Can take values from 0 to `getAltitudeLength() - 1`.

The `getAltitude` method returns a `float` value representing the altitude level (in meters) at `index`.

8.3 Setting and getting the number of wide CK grid points (setWideCKNumber and getWideCKNumber)

You can control the accuracy and speed of the wide band CK approximation to some extent by controlling the number of wide band CK grid points (the default is 17). To do this, you can use the `setWideCKNumber` method with parameter:

`int n`	Number of wide band CK grid points. Must be >= 2.

Note that this modification only takes effect after an update of the `smarti` instance (see section 10). You can also retrieve the number of wide band CK grid points currently in use for the wide band data (or currently set if no update has been done) with the `getWideCKNumber` method which takes no argument and returns an `int`.

9 Sensors

Sensors in SMARTI have many functions. They are the starting point for path tracing and they define response (or filter) functions. Defining filter functions is especially important in wide band CK calculations, since for maximum accuracy they should by incorporated in the wideband CK spaces (each sensor has its own wideband CK space).

Sensors can be added, moved and removed from a `smarti` instance. The following sections explain how this is done.

9.1 Adding sensors (addSensor)

By adding a sensor, you define its starting position (it can be moved later, see next section) and its response function. As mentioned the response function is more important when it comes to wide band calculations. In both spectral and wide band calculations the response function can be converted into a `normType` or `wideNormType` and applied to the results after your calculations are done, but in the wide band case this method is less accurate.

Adding a sensor to a `smarti` instance is done by calling the `addSensor` method with the following parameters:

`const geodesic& position`	Geodesic coordinates of sensor (see section 4.4.1 for information on the `geodesic` class).
`int nb_pts`	Number of points in sensor response function (>=2 for user data or <=0 for default). Omitting this parameter defaults to 0.
`float* spectral_grid`	Spectral grid of sensor response function (microns). Omitting this parameter defaults to NULL.
`float* response`	Sensor response function (unitless), set to NULL for default flat 1.0 response. Omitting this parameter defaults to NULL. Values should normally be bounded from 0 to 1.

If `nb_pts == 0` or `spectral_grid == NULL` or `response == NULL` the sensor response function is set to a flat 1.0 value on the entire band defined by the `smarti` instance. If you do specify the sensor response function, you should make sure it is non zero on at least part of the spectral band of the `smarti` instance, as any part of the function outside the band is not considered and truncated (a zero response function on the entire band is not very useful!). Note that if the input spectral grid does not cover the entire `smarti` instance spectral band, the end values (first and last) of the `response` vector are used for the rest of the band. Because of this, if you want your sensor response to fall to 0, the 0 values must explicitly be defined at the extremities of the `response` vector.

The `addSensor` method returns a `size_t` type sensor handle. Many methods, including path tracing and most wide band CK functionalities, require that you specify the sensor to use. The sensor handle returned by `addSensor` is a unique index that refers to one and only one sensor. For a given `smarti` instance, a sensor handle will never be repeated to insure uniqueness.

9.2 Moving sensors (moveSensor)

Once you've added at least one sensor to your `smarti` instance you can move it around using one of the two `moveSensor` methods. With the first version of the method you simply specify which sensor to move and the new set of geodesic coordinates:

`size_t handle`	Handle of sensor (returned by the `addSensor` method) to move.
`const geodesic& position`	New geodesic coordinates of sensor (see section 4.4.1 for information on the `geodesic` class).

The second way to move a sensor comes in handy when you now its initial heading and the distance traveled. The trajectory calculated is a geodesic path (shortest path between two points on the surface of the reference ellipsoïd representing earth). The parameters are:

`size_t handle`	Handle of sensor (returned by the `addSensor` method) to move.
`float heading`	Heading clockwise from north, or east of north (rad)
`float distance`	Surface (at sea level) distance of travel (m)
`float alt`	New geodesic altitude of sensor (m)

Note that in sensor level scattering mode, source function calculation will not reflect the new position of the sensor until you cal the `update` method. On the other hand, path tracing methods will reflect the new position immediately after calling `moveSensor`.

9.3 Removing sensors (removeSensor)

A sensor can also be removed from a `smarti` instance. The only real advantage of doing so is to free up a bit of memory, but it is still good practice to remove sensors that will not be used anymore. This is especially true for very long simulations where many sensors might come in and out of play during the lifetime of the `smarti` instance. To remove a sensor, simply use the `removeSensor` method by specifying the handle of the sensor to remove.

`size_t handle`	Handle of sensor (returned by the `addSensor` method) to move.

Obviously, the handle of a sensor you have removed will become unusable for the duration of the life of the smarti instance.

9.4 Retrieve sensor information (getSpectralSensorResponse, getWideBandSensorResponse and getSensorPosition)

A few methods in the `smarti` class let you query some of the sensor properties. In order to obtain the spectral sensor response (filter) function, you can use the `getSpectralSensorResponse` method which returns a `normType`, while the wide band counterpart (returning a `wideNormType`) is

obtained with `getWideBandSensorResponse`. The coordinates of the sensor as a `geodesic` object can also be retrieved with the `getSensorPosition` method. All three methods take as single argument:

`size_t handle`	Handle of sensor (returned by the `addSensor` method) to move.

10 Updating the SMARTI instance (update)

Updating your `smarti` instance insures that all pre-calculations necessary for radiative (fluxes, transmittances, radiances, etc.) calculations reflect the parameters set by calls to modifiers. It is in fact mandatory to call the update method at least once before attempting any radiative calculation. Any radiative calculation will reflect the state of the `smarti` instance at the time of the last update.

It is good practice to call the `update` method after a series of calls to modifier methods, before any radiative calculations, but it is not mandatory (except for the first call to `update`). By design, you might even want to postpone the call to `update` until a specific condition is met. For example, say you are in sensor level scattering mode (see section 7.5.2), and sensor locations are updated very often with the `moveSensor` method. You might want to wait for at least one sensor to have moved at least 10 kilometers before calling the `update` method, since calling it too often might slow your application down considerably.

The `update` method takes no parameters.

11 Retrieving meteorological profile information

Once the `smarti` instance has been updated, you have access to the meteorological properties of the atmosphere (temperature, pressure, relative humidity) trough a series of methods. These are described in the following sections.

11.1 Obtaining meteorological profiles at altitude grid levels (getPressure, getTemperature and getRelativeHumidity)

The pressure (mbar), temperature (K) and relative humidity (%) can be obtained at the altitude grid levels (see section 8.2) with methods `getPressure`, `getTemperature` and `getRelativeHumidity` respectively, which take as single argument

`size_t index` — Index of the altitude level at which to retrieve the meteorological data. Valid values are from 0 to `getAltitudeLength() - 1`.

All three methods return a `float` value.

11.2 Interpolating meteorological profiles at arbitrary altitude levels (interpPressure, interpTemperature and interpRelativeHumidity)

Just like their analogous "get" methods, the pressure (mbar), temperature (K) and relative humidity (%) can be interpolated at arbitrary altitude levels with methods `interpPressure`, `interpTemperature` and `interpRelativeHumidity` respectively, which take as single argument

`float alt` — Altitude level (meters) at which to interpolate the meteorological data. Outside the limits of the atmosphere, these values are extrapolated which may result in unexpected behavior; make sure that is really what you want.

All three methods return a `float` value.

12 Tracing paths (getPath_LOS, getPath_FOV, getPath_Target)

Many radiative transmission quantities output by SMARTI are computed along a path trough the atmosphere (radiances and transmittances) or at the end of a path (background radiances). In order to get a single refracted path along a given line of sight defined by view elevation and azimuth, you can call the `getPath_LOS` method, with following parameters.

`size_t sens_handle` — Sensor handle for starting position and effective wavelength for refraction index calculation.

`float elevation` — Path start elevation (rad) above local (sensor position) horizontal (tangent to reference ellipsoïd). From $-\pi/2$ to $\pi/2$.

`float azimuth` — Path start azimuth (rad), clockwize from North (East of North).

`float max_dist` — Maximum distance (meters) of travel. This is the curve length of the refracted path, not the linear distance between the start and end points (although both are usually very close). If the path intercepts the ground or the top of the atmosphere before reaching `max_dist`, it will be truncated. Note that in order to calculate a background radiance that includes the ground

	reflected and/or emmitted radiance, path must intercept the ground, so max_dist should be set accordingly. If parameter is omitted, it defaults to the maximum value for a floating point (type `float`) on your system. This should insure ground or top of atmosphere interception, but can lead to excessive path lengths or even errors in some rare atmospheric conditions (refractive ducting).

The `getPath_LOS` method returns a `pathOutputType` object (see section 4.3) that always contains a single path. In other words, you always access the original and translated paths using index 0.

In order to obtain multiple paths spanning a vertical field of view at equal angle increments, you can use the `getPath_FOV` method, which takes in the following arguments:

`size_t sens_handle`	Sensor handle for starting position and effective wavelength for refraction index calculation.
`float center_elevation`	Center of field of view elevation (rad) above local (sensor position) horizontal (tangent to reference ellipsoïd).
`float azimuth`	Field of view start azimuth (rad), clockwize from North (East of North).
`float FOV`	Vertical field of view (rad). Field of view must be entirely contained within a horizontal half plain (all paths must be within $-\pi/2$ and $\pi/2$ in elevation).
`size_t nb_pth`	Number of trajectories in field of view (equally spaced).
`float max_dist`	Maximum distance (meters) of travel. This is the curve length of the refracted path, not the linear distance between the start and end points (although both are usually very close). If the path intercepts the ground or the top of the atmosphere before reaching `max_dist`, it will be truncated. Note that in order to calculate a background radiance that includes the ground reflected and/or emmitted radiance, path must intercept the ground, so max_dist should be set accordingly. If parameter is omitted, it defaults to the maximum value for a floating point (type `float`) on your system. This should insure ground or top of atmosphere interception, but can lead to excessive path lengths or even errors in some rare atmospheric conditions (refractive ducting).

The `getPath_FOV` method returns a `pathOutputType` object (see section 4.3) that always contains `nb_pth` paths.

To obtain a refracted path that leads from a sensor to a target at a given set of coordinates, you must use the `getPath_Target` method. The method has the following parameters:

`size_t sens_handle`	Sensor handle for starting position and effective wavelength for refraction index calculation.
`geodesic target_coordinates`	Target geodesic coordinates.
`bool stop_at_target = false`	flag to stop path(s) at target (if true) or continue to background (false, default).

Internal calculation of the atmospheric index of refraction takes into account the meteorological profile (temperature, pressure and relative humidity). More methods for obtaining paths will be made available in future versions.

13 Radiative calculation

If you intend on using SMARTI in your project, it is probably to do some kind of radiative transmission calculation (if not, you might find SMARTI a bit bloated just to track sensor positions or calculate the sun or moon position). Once you have set up your `smarti` instance with all the necessary modifier methods, you are ready to call on of the radiative transmission methods (spectral or wide band CK) listed and explained in the following sections.

Most methods have a spectral and a wide band CK alternative (wide band versions usually have the word “Wide” in their name). For wide band CK output, you can often chose weather you require the solar or thermal part. The reason is that the two have different wide band CK spaces to improve accuracy of calculations. For radiances and fluxes, these are physically the solar and thermal parts of the total energy balance. In the case of the transmittance, you should use the solar part to transmit anything that originates from the sun/moon (*sun* glint, reflection off a surface of the *solar* scattered sky radiance, etc.). The thermal part of the wide band CK transmittance is used for anything else (including thermal, or blackbody signatures, obviously), providing that the solar spectrum is not directly involved (visible) in the signature you are transmitting.

13.1 Solar irradiance (getSolarIrradiance and getSolarIrradianceWide)

You can get the solar/lunar irradiance transmitted trough the atmosphere at any geodesic coordinates by using the `getSolarIrradiance` (for spectral calculations) and `getSolarIrradianceWide` (for wide band CK results) methods. Both have identical parameters, but with subtle differences:

`const geodesic& position`	Geodesic coordinates
`size_t sens_handle`	Sensor handle to use, as returned by the `addSensor` method. For spectral calculations (`getSolarIrradiance`), specifying the sensor handle only has an effect in 1D scattering (`setIrradianceMode`) sensor level (`setSensorScattering`) modes. Because of this it is optional for spectral calculations (defaults to 0 if omitted), but it is mandatory for wide band CK calculations (`getSolarIrradianceWide`) as in this case it also specifies the CK space to use.

The `getSolarIrradiance` method returns a `radCKType` object while `getSolarIrradiance` returns a `wideRadCKType`. Both contain the solar/lunar irradiance in $W/m^2/cm^{-1}$ impinging on a surface whose normal points towards the sun/moon. Calling the `value` function on either will give you the band irradiance in W/m^2. The irradiance is solar in origin by default, or if you have called the `setIlluminator` with argument `true`, and is lunar if you called `setIlluminator` with argument `false` (see section 7.4.3).

13.2 Radiative fluxes (getUpFlux, getDownFlux, getUpFluxWide and getDownFluxWide)

In SMARTI, radiative fluxes are defined as the diffuse components of the irradiance impinging on a horizontal surface from the top (down flux) or from the bottom (up flux). Fluxes can be composed of thermal emissions, solar/lunar single scattering/reflection or multiple scattering of solar and thermal components from the atmosphere or ground. Fluxes are only calculated when the number of streams is set to 2 (or larger when it becomes supported) with the `setScatteringApprox` method (see section 7.5.3)

You can retrieve radiative fluxes only after at least one `update` of the `smarti` instance with methods:

- `getUpFlux` (retrieve upward flux spectrally)
- `getDownFlux` (retrieve downward flux spectrally)
- `getUpFluxWide` (retrieve upward flux in wide band CK formalism)
- `getDownFluxWide` (retrieve upward flux in wide band CK formalism)

All four methods share the same first two parameters:

`float alt`	Altitude

`size_t sens_handle`	Sensor handle to use, as returned by the `addSensor` method. For spectral calculations (`getUpFlux` and `getDownFlux`), specifying the sensor handle only has an effect in sensor level (`setSensorScattering`) scattering mode. Because of this it is optional for spectral calculations (defaults to 0 if omitted), but it is mandatory for wide band CK calculations (`getUpFluxWide` and `getDownFluxWide`) as in this case it also specifies the CK space to use.

The wide band CK methods also have a third parameter:

`bool solar`	Flag to chose solar (true) or thermal (false) fluxes to output. Remember, in wide band CK mode, solar and thermal contributions each have their CK space.

13.3 Path radiative quantities

The radiative quantities that SMARTI can calculate along a given path are the transmittance, the atmospheric radiance (scattered and emitted) as well as the radiance of the background (at the end of the path including ground or sea radiance). All these quantities are closely related; the path transmittance is required in order to calculate the path radiance, and both transmittance and path radiance come into play when calculating the background. Furthermore, many applications require all three quantities (contrast and detection assessment for example). Since SMARTI is designed to optimise radiative transfer calculations as much as possible, it makes sense when needed to produce all three quantities at once. If this was not possible, producing all three quantities separately would require the transmittance to be calculated three times.

Note that the underlying SMART library uses a different approach, using a pre calculated transmittance as input to a path radiance calculation and so on. SMARTI is meant to simplify the use of SMART, so it was decided to put aside as much of these complex input/output interactions as possible.

13.3.1 Obtaining path radiative quantities (getSpectralRadiativeQuantities and getWideBandRadiativeQuantities)

Spectral radiative quantities (transmittance, radiance and background) are obtained using the `getSpectralRadiativeQuantities` while wide band radiative quantities are retrieved by calling the `getWideBandRadiativeQuantities`. These methods return `smartiSpectralOutput` and `smartiWideBandOutput` objects respectively (see section 4.5) and take parameters:

`const pathOutputType& path`	Path(s) obtained by one of the getPath() methods.
`size_t sens_handle`	Sensor handle for source function. Can be omitted (defaults to 0) in the `getSpectralRadiativeQuantities` method

	when not in sensor level mode (see section 7.5.2), but mandatory for the `getWideBandRadiativeQuantities` method (since each sensor has its own wide CK space).

13.4 End of path transmittance (getSpectralEndTransmittance and getWideBandEndTransmittance)

There are times when all you really need is the spectral (CK) transmittance at the end of your path. In this case there is no use in calculating the path radiance and background. For that purpose, the `getSpectralEndTransmittance` returning a `normCKType` object is available and takes in a single parameter:

`const pathElement& path`	One path obtained by one of the getPath() methods (see section 12). Use the [] operator of the `pathOutputType` class to obtain a single path as required.

To get the wide band CK transmittance at the end of a path, the analogous method is `getWideBandEndTransmittance`, returning a `wideNormCKType` and taking parameters:

`const pathElement& path`	One path obtained by one of the getPath() methods (see section 12). Use the [] operator of the `pathOutputType` class to obtain a single path as required.
`size_t sens_handle`	Sensor handle to use (for the wide band CK space).
`output_modes mode`	Output mode to use (`solar` or `thermal`), mode `both` will throw an exception. If omitted, default is `thermal`.

13.5 End of path atmospheric radiance (getSpectralEndRadiance and getWideBandEndRadiance)

For calculating end of path spectral (CK) radiances a method similar to the `getSpectralEndTransmittance` method is provided, the `getSpectralEndRadiance` method returning a `radCKType` object and taking in a single parameter:

`const pathElement& path`	One path obtained by one of the getPath() methods (see section 12). Use the [] operator of the `pathOutputType` class to obtain a single path as required.

To get the wide band CK radiance at the end of a path, the analogous method is `getWideBandEndRadiance`, returning a `wideRadCKType` and taking parameters:

`const pathElement& path`	One path obtained by one of the getPath() methods (see section 12). Use the [] operator of the `pathOutputType` class to obtain a single path as required.
`size_t sens_handle`	Sensor handle to use (for the wide band CK space).
`output_modes mode`	Output mode to use (`solar`, `thermal` or `both`). If omitted, default is `both`.

14 Translator methods (dataToNorm, dataToRad, normToWide and radToWide)

In many cases you will surely have your own spectral data, originating from other programs, or databases, which you will want to convert to SMART/SMARTI compatible data types. A `smarti` object provides methods for doing just that. To convert vectors of floating point spectral data (type `float*` or `float[]`) to `radType` or `normType` SMARTI spectral types, you can use the `dataToRad` and `dataToNorm` methods with parameters:

`int nb_pts`	Number of spectral grid and data points.
`float* specGrid`	Spectral grid, monotonic ascending, `nb_pts` elements.
`float* specData`	Spectral data to be converted to `radType` or `normType`, `nb_pts` elements.
`spec_units units`	Units of input spectral grid: `wavelength` (micron) or `wavenumber` (cm-1). Defaults to `wavelength` if omitted.

Obviously, these methods return a `radType` and a `normType` respectively. Converting your data to wideband CK types `wideRadCKType` and `wideNormCKtype` is just as easy with methods `radToWide` and `normToWide` taking these parameters:

`size_t sens_handle`	Sensor handle for wide band CK space selection.
`int nb_pts`	Number of spectral grid and data points.
`float* specGrid`	Spectral grid, monotonic ascending, `nb_pts` elements.
`float* specData`	Spectral data to be converted to wide band CK, `nb_pts` elements.
`bool solar`	Is it for solar calculations? Defaults to false if omitted.

`spec_units units`	Units of input spectral grid: `wavelength` (micron) or `wavenumber` (cm-1). Defaults to `wavelength` if omitted.

Note that for all the translator methods, if the input spectral grid (`specGrid`) does not cover the entire `smarti` instance spectral band, the end values (first and last) of the `specData` vector are used for the rest of the band.

Annexes

A Typical wide band CK chain of calculation

Working with wide band correlated-ks might seem daunting and unintuitive at first. To help the user of the library grasp the main concepts, we present here an example of a wide band CK calculation that is simple but representative of a typical use of the library. In this example, the signature a flat Lambert reflector/emitter is calculated using incident radiation obtained from SMARTI and a classic blackbody function. The signature is then propagated towards a sensor to finally obtain the apparent target signature as measured by the sensor.

The example presented here is entire conceptual; no code is actually given. We present a few equations to help the reader follow the concepts. We also give, *italicized* and offset to the right, some practical information about what SMARTI method or function might be involved. Readers interested in a coded example can look at the examples directory for SMARTI (\eospec\examples\module\smarti\cplusplus). A similar example (widebandExample.cpp) is coded but the facet has a vertical normal and no thermal emissions.

> *What follows assumes that the sensor and its spectral response is preinitialized in SMARTI with the* `addSensor` *method.*

In the SMARTI wide band CK formalism, all calculations are done in either one of the CK spaces associated with a given sensor. These are the *thermal* and *solar* CK spaces and each sensor has its own, different set. As their name imply, all calculations involving solar contributions are done in the *solar* CK space, while the rest (usually calculations involving thermal contributions), are done in the *thermal* CK space.

To do this, all spectral quantities that intervene in a given calculation must first be converted into both the *thermal* and *solar* CK spaces. This must be done for each sensor since as mentioned each sensor has its own set of CK spaces. In the case that concerns us (calculating the reflected and emitted contributions from a Lambert surface) we need to convert the surfaces spectral emissivity as well as the spectral blackbody function corresponding to the surface temperature. So for a given sensor:

$$\begin{aligned} \varepsilon(\lambda) &\rightarrow \varepsilon_s(ck) \\ &\rightarrow \varepsilon_t(ck) \\ B(\lambda) &\rightarrow B_t(ck) \end{aligned}$$

where the indices s and t denote the *solar* and *thermal* CK spaces respectively. Note that the blackbody function is only converted into the *thermal* CK space (since a blackbody is by definition thermal).

> *In practice, the conversion of a blackbody function to wide band CK with SMARTI is achieved with the* `radToNorm` *method, with the solar argument set to* `false`*. The conversion of the emissivity is done with the* `normToWide` *method with the solar argument to* `false` *for the thermal space and to* `true` *for the solar space.*

The thermal radiance emitted from the surface, being a thermal contribution only, is then calculated in the thermal CK space:

$$E_t = \varepsilon_t B_t(ck)$$

One way of approximating the reflected contributions off the surface is to use the direct solar irradiance as well as the upwelling and downward radiative fluxes. This must be done in both solar and thermal CK spaces according to these equations (dropping the ck dependence for conciseness):

$$R_s = \left(1-\varepsilon_s\right)\left[\frac{\left(I_s\right)\cos\left(\theta\right)}{\pi} + \gamma F_s^{\downarrow} + \left(1-\gamma\right)F_s^{\uparrow}\right]$$

$$R_t = \left(1-\varepsilon_t\right)\left[\gamma F_t^{\downarrow} + \left(1-\gamma\right)F_t^{\uparrow}\right]$$

$$\gamma = \frac{\pi-\theta_n}{\pi}$$

with :

- θ : angle between the surface normal and the sun direction.
- θ_n : angle between the surface normal and the vertical (zenith).
- I_s : Solar irradiance (only in the *solar* CK space by definition)
- $F_s^{\downarrow}$: Solar downward diffuse radiative flux (sky scattered sun light)
- $F_s^{\uparrow}$: Solar upwelling diffuse radiative flux (mostly ground scattered/reflected sun light)
- $F_t^{\downarrow}$: Thermal downward diffuse radiative flux (sky thermal emissions and scattered light)
- $F_t^{\uparrow}$: Thermal upwelling diffuse radiative flux (mostly ground scattered/emitted light)

In practice, the solar irradiance (I_s) is obtained with the getSolarIrradianceWide method. The downward and upwelling fluxes ($F^{\downarrow}$ and $F^{\uparrow}$) are obtained with getDownFluxWide and getUpFluxWide respectively, with parameter `solar` set to `false` for thermal space fluxes and to `true` for solar space fluxes.

The thermal and solar components of the surface radiance are then simply the sum of the emitted and reflected components:

$$L_t = E_t + R_t$$
$$L_s = R_s$$

As mentioned, the thermal and solar CK space distinction should be conserved until the end result (a rule of thumb is that you should maintain distinct spaces as long as there are multiplications of CK quantities in your problem). The transmitted surface radiance (denoted L') is then obtained by applying the thermal and solar transmittances :

$$L'_t = L_t T_t$$
$$L'_s = L_s T_s .$$

The total radiance arriving at the sensor is then simply the sum of the solar and thermal parts (we have no CK multiplications left in our problem):

$$L' = L'_t + L'_s$$

If we also want to include the atmospheric path radiance (denoted P here) to obtain the actual apparent surface radiance, the equation becomes (again, no CK multiplications, so the solar and thermal spaces can be mixed):

$$L' = \left(L'_t + R_t\right) + \left(L'_s + R_s\right)$$

Finally, wide band ck spaces are always associated with a sensor response function (even if it is unitary on the entire band). So in wide band CK calculations only the measured radiance is a proper physical quantity. Because of this, the wide band filter function always needs to be applied to the result (there is no distinction between solar and thermal space for this quantity, so it can be applied to the global result):

$$L'_{measured} = L'F$$

In practice, the wide band CK sensor response function is obtained with the getWideBandSensorResponse method.

Note that the result here is still described by correlated-k values. To obtain a scalar band result, the CK values should be integrated.

In practice, the CK integration into a scalar result is done by using the value function.

B Default parameters

The following table lists the parameters representing the state of a `smarti` class instance when it is constructed. Note that these parameters do not take effect until a call to the `update` method is made. These are the parameters that are used if no modifier method is called to change them.

Parameter name	Value	Associated modifier	Note
`modRoot`	`"C:/Mod4v3r1"`	`setModtran`	
`modIOdir`	`"Test"`	`setModtran`	
`modCase`	`"driver_output"`	`setModtran`	
`modExe`	`"Mod4V3R1.exe"`	`setModtran`	
flag	`false`	`memOptim`	
flag	`false`	`onlyWide`	
model	6	`setModtranMeteo`	MODTRAN meteorological model used for entire atmosphere by default
`measHeight_W`	10.0	`setWind`	
`windSpeed`	5.0	`setWind`	
`windSpeed24`	5.0	`setWind`	
`model`	rural	`setModtranAerosols`	MODTRAN aerosol used by default
`visibility`	0	`setModtranAerosols`	Rural default is 23 km
`model`	0	`setCloudAndPrecipitation`	
`snowfallrate`	0.0	`setCloudAndPrecipitation`	
`sun_azimuth`	Calculated from current computer time	`setSkyFromView`	Constructor actually uses `setSkyFromNow`
`sun_elevation`	Calculated from current computer time	`setSkyFromView`	Constructor actually uses `setSkyFromNow`
`sun_distance`	Calculated from current computer time	`setSkyFromView`	Constructor actually uses `setSkyFromNow`
`moon_azimuth`	Calculated from current computer time	`setSkyFromView`	Constructor actually uses `setSkyFromNow`
`moon_elevation`	Calculated from current computer time	`setSkyFromView`	Constructor actually uses `setSkyFromNow`
`moon_distance`	Calculated from current computer	`setSkyFromView`	Constructor actually uses

	time		setSkyFromNow
sun	true	setIlluminator	
irrtype	1	setIrradianceMode	
scene_level	false	setSensorScattering	Set to sensor level scattering by default
nbStreams	2	setScatteringMode	
nb_pts	2	setLambertSurface	Lambert surface used by default
spectral_grid	{0.2,40.0}	setLambertSurface	Lambert surface used by default
albedo	{0.5,0.5}	setLambertSurface	Lambert surface used by default
Temperature	-999	setGroundTemperature	First atmospheric layer in US standard atmosphere is 288.2 K
altitude	0.0	setSoftGroundAltitude	
n	17	setWideCKNumber	

Note that on construction a smarti class instance does not contain any sensor definition which is essential for many of the SMARTI library functionalities to work.

C Modifying MODTRAN® 4 or 5 for use in SMARTI

> WARNING: Do not create, copy or modify files in the generated /src_mod directory, these may be deleted when executing MODTRAN patching script!

C.1 Introduction

There is a script for patching most versions of MODTRAN® 4 or 5 to output Correlated-k files for use in SMART and SMARTI. In the EOSPEC directory structure, this script is located in

\eospec\external\MODTRAN\smart_MODTRAN_patch

If an Intel Fortran compiler is installed on your computer, this tool will also attempt to compile the patched code into a binary executable.

The resulting compiled MODTRAN® executable is called by SMART and SMARTI.

C.2 Requirements

This script is meant to be executed on a Windows computer. Older versions of Windows XP may require the Microsoft Visual Studio 2008 run time libraries to be installed. It can be downloaded here:

http://www.microsoft.com/en-ca/download/details.aspx?id=29

C.3 Installation

No installation is required to use this script. Simply copy the entire content of the archive to any writable location on your computer.

C.4 Simple usage

To patch MODTRAN® 4 or 5 source code for use with SMART and SMARTI, simply run smart_MODTRAN_patch.bat by double clicking on the file. You are then prompted to enter the MODTRAN® source code directory:

*Please enter MODTRAN *ORIGINAL* source code directory:*

Type the full or relative path of the original MODTRAN® source code and press <Enter>. If a path to modified source code is entered, patching might fail.

You are then prompted to enter the resulting MODTRAN® executable name:

Please enter existing output directory for MODTRAN executable:

Type the full or relative path of the modtran executable output directory. You are then prompted to enter the resulting MODTRAN executable name:

Please enter name for modtran executable (without extension):

If any directories of files to be overwritten exist, you will be prompted to confirm that this is ok by typing *y* (yes) or *n* (no).

In the absence of an Intel Fortran installation on your computer, a message is simply printed asking you to compile the code manually.

C.5 Command line usage

The script can be called from the command line with the following syntax:

smart_MODTRAN_patch *MOD_SRC_DIR MOD_EXE_DIR MOD_EXE_NAME* [-s] [-nc] [-?]

MOD_SRC_DIR :	Original MODTRAN source file directory (in quotes if spaces)
MOD_EXE_DIR :	Output directory for output MODTRAN executable (in quotes if spaces)
MOD_EXE_NAME :	Name of MODTRAN executable (without extension)
-s :	Silent flag: if provided, no overwrite confirmations are required.
-nc :	No compilation flag. If provided, source is patched but not compiled.
-? :	Prints this section to the screen.

Note that if the -nc flag is provided, *MODTRAN_EXE_DIR* and *MODTRAN_EXE_NAME* are not required. If the -? flag is provided, no other input is required.

In the absence of an Intel Fortran installation on your computer, a message is simply printed asking you to compile the code manually.

C.6 Compilation

While compiling the patched source code, warning and even errors might be printed. Warnings are usually harmless, and errors usually occur because some distributions of MODTRAN® contain erroneous and superfluous source files. As long as the link step (at the very end) is successful, errors and warnings can be ignored.

If manual compilation of the source code is required, preprocessor definition SMART_OUTPUT must be defined in order for the additional code required by SMART/SMARTI to take effect. Compiling without the SMART_OUTPUT produces an unmodified MODTRAN® executable.

C.7 User privileges

The compilation script may require administrative privileges to function. The reason is that it requires read access to some registry keys trough the "reg query" command. Some group policies forbid the use of this command. If you do not have the required privileges a message is printed to that effect asking you to compile the patched code in /src_mod manually.

C.8 Compiled executable

Note that the resulting executable IS NO LONGER A VALID MODTRAN® EXECUTABLE. Results from this executable will not be valid if used outside of the SMART/SMARTI integration.

C.9 Atmospheric layering problems

With some early versions of MODTRAN® 4, notably 4.0, the number of internal layers might be insufficient for use with SMARTI, and limiting with SMART.

If MODTRAN® errors occur with messages related to atmospheric layering, you can modify the PARAMS.LST (PARAMS.h in later versions) file to set the LAYDIM parameter to a higher value (minimum of 114, 502 optimally) and recompile the source code.

Contact information

Author:

Vincent Ross
Aerex Avionique inc.
vross@aerex.ca

References

1. Berk, A., L.S. Bernstein, and D. C. Robertson, “MODTRAN: A Moderate Resolution Model for LOWTRAN7”, Rep. GL-TR-89-0122, *Air-Force Geophysics Lab.*, Bedford, MA, 1989.
2. P.K. Acharya, A. Berk, G.P. Anderson, N.F. Larsen, S-Chee Tsay, and K.H. Stamnes, “MODTRAN4: Multiple Scattering and Bi-Directional Reflectance Distribution Function (BRDF) Upgrades to MODTRAN”, *Proc. of SPIE, Optical Spectroscopy Techniques and Instrumentation for Atmospheric and Space Research,* **3756,** 19-21, SPIE, July 1999.
3. J. L. Forand, “The L(W)WKD Marine Boundary Layer Model - Version 7.09,” *Technical Report 1999-099*, Defence Research Establishment of Valcartier (DREV), Valcartier, Quebec, Canada (1999).
4. D. Dion, L. Gardenal, J. L. Forand, M. Duffy, G. Potvin, and S. Daigle, “IR Boundary Layer Effects Model (IRBLEM), IRBLEM5.1 documentation, DRDC-RDDC Valcartier, Quebec, Canada (2004). (Available via e-mail by submitting request to the authors.)
5. Vincent Ross, Denis Dion, and Guy Potvin, "Detailed analytical approach to the Gaussian surface bidirectional reflectance distribution function specular component applied to the sea surface," *J. Opt. Soc. Am. A* 22, 2442-2453 (2005)
6. Ross, V., and D. Dion (2007), “Sea surface slope statistics derived from Sun glint radiance measurements and their apparent dependence on sensor elevation,” *J. Geophys. Res*., 112, C09015, doi:10.1029/2007JC004137.

Note: this bibliography does not encompass the entire set of models used within SMART and SMARTI, and is only provided to lead the reader to clarifications on some of the models and methods presented in this document.

Index

H

I

L

M

N

O

P

R

S

T

U

V

W