DART TOOLS USER MANUAL

Technical document

- Prospect
- SAIL
- DEMGenerator
- Vegetation
- Lidar
- Sequencer
- BroadBand
- Inversion

Reference:	Modules_User_Manual.docx
Release date:	May 24, 2016
Author:	CESBIO

CESBIO : Centre d'Etudes Spatiales de la BIOsphère – UMR 5126 (UPS-CNRS-CNES-IRD)
18, avenue Edouard Belin – 31401 Toulouse – France
☎: 05 61 55 85 01 - http://www.cesbio.ups-tlse.fr/index_us.htm

Associated industrial company: **Magellium** - 24, rue Hermès - BP 12113
31521 Ramonville Saint-Agne. – France
☎: 05 62 24 70 00 - Fax: 05 62 24 70 01 - www.magellium.fr

MODULES USER MANUAL (to be seriously up-dated)

1 PROSPECT (to be updated) 5

2 SAIL 6

3 DEMGenerator (to be up-dated) 9

4 Vegetation(to be up-dated) 14

5 Lidar (to be up-dated) 17

6 Sequencer (to be up-dated) 33

7 BroadBand 37

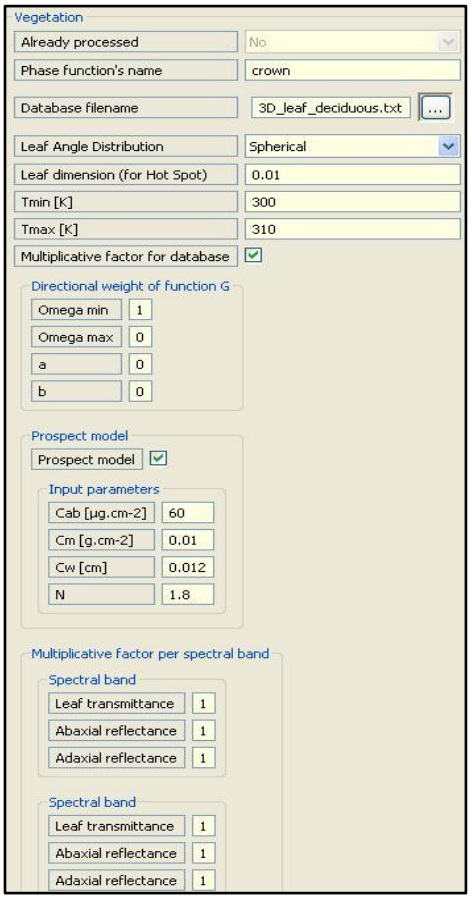
8 Image Inversion (to be totally reformulated) 41

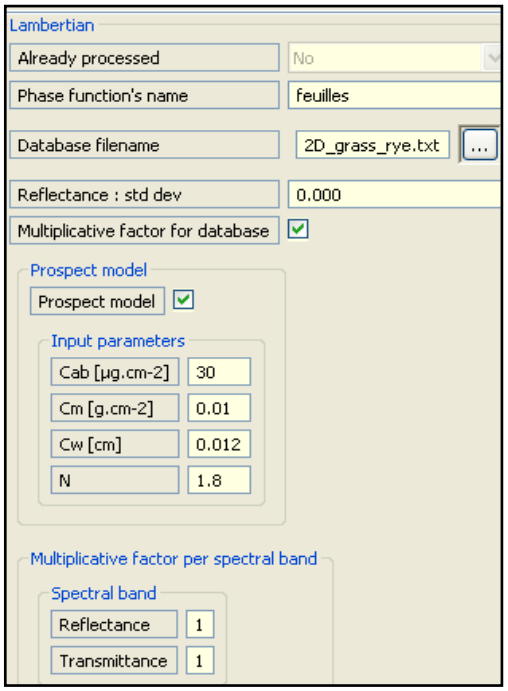
1 *PROSPECT (to be updated; cf. DART User Manual)*

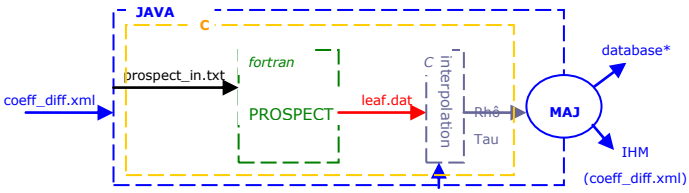
Prospect module computes 2D surface and 3D turbid media reflectance $\rho_{lamb}(\lambda)$ and transmittance $\tau_{lamb}(\lambda)$, from 0.4µm to 2.5µm, with Prospect model (Jacquemoud et Baret, 1990) and an interpolation on leaf refraction index and absorption coefficients ('calcul_prospect_values.txt' file; $\Delta\lambda$=5nm).

Link for downloading the PROSPECT model: http://teledetection.ipgp.jussieu.fr/prosail/

The 6 columns of text file "**calcul_prospect_values.txt**" give all spectral input parameters every 5nm:
[Wavelength] [Refraction index] *Absorption coefficients* [Albino leaf] [Chlorophyll a+b] [Water] [Dry matter]







1st step: specification of the spectral bands (DART GUI: "Spectral intervals") and the use of the PROSPECT module and a spectral data base (DART GUI: "Scattering properties"). The data base is for turbid medium (*e.g.*, 3D_leaf_decideous.txt) or for a surface (*e.g.*, 2D_grass_rye.txt).

2nd step: run of the Prospect module. It stores in the user local database directory the data base specified at step 1, with the spectral values calculated by the Prospect model for the spectral bands specified at step 1. If the data base was already in the local data base, it is updated.

☞ If you use the sequencer, do not forget to specify that the sequence uses the Prospect module. When the Prospect module is used, the number of simulations that can be run in parallel is 1, only.



Menu of the sequencer.

2 *SAIL*

-For comparing DART & SAIL simulations, for pure turbid media -

This module is an adaptation of the very well known SAIL model (Verhoef[1], 1984). It simulates the reflectance of homogeneous vegetation covers (*i.e.*, Ground + Homogeneous vegetation), for any illumination (sun direction (θ_s,ϕ_s), SKYL), vegetation structure (LAI, LAD, number of layers), optical properties (leaf reflectance ρ_f, transmittance τ_f) per layer, ground reflectance ρ_{ground}, and any view direction (*e.g.*, directions of the "directions.txt" DART file. Due to its approximations, 1st order reflectance is not 100% exact (1-2% relative difference). Multiple scattering is less accurate due to more important approximations. The hot spot is simulated only along the exact hot spot direction: reflectance values for directions other than the hot spot do not depend on hot spot.

The SAIL module computes 4 reflectance values (*Figure 1*) and stores them in a single text file (*e.g.*, "rresu") with 6 columns: View zenith angle, View azimuth angle, Canopy total reflectance, Vegetation direct (single scattering) reflectance, Vegetation multiple scattering reflectance and Ground total direct reflectance. These reflectance values are also stored in individual text files:

- *Canopy total reflectance* R_{total} : text file "brf_resu_total",
- *Vegetation single scattering reflectance* R_s: text file "brf_resu_order1",
- *Vegetation multiple scattering reflectance* R_d: text file "brf_resu_diffuse", and
- *Ground total direct reflectance* R_{ground}: text file "brf_resu_ground".

SAIL input parameters are input with an interactive menu (*Figure 3*) or with the command "sail.exe *Batch_SAIL.txt*", where " *Batch_SAIL.txt* " is a text file that stores the input parameters (*Figure 2*).

To run the SAIl model from a folder that is not the "bin" folder of DART, copy the 3 libraries "libstdc++-6.dll, libgcc_s_sjlj-1.dll, libgcc_s_dw2-1.dll"

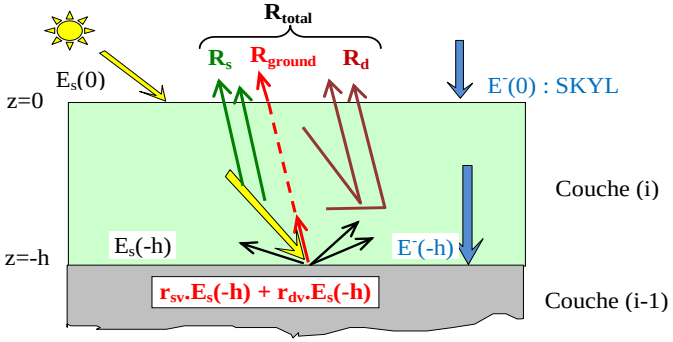


Figure 1: Homogeneous cover above a lambertian ground.

[1] Verhoef W., 1984, Light scattering by leaf layers with application to canopy reflectance modeling: The SAIL model", Remote sensing of Environment, 16:125-141.

1	1: New simulation. 2: Quit
1	Number of vegetation layers
0.01	Ground reflectance
0.49 0.49	Leaf reflectance and transmittance, per vegetation layer
C:\...\Spherical.txt	LAD (Leaf Angle Distribution), per vegetation layer
2	LAI (Leaf Area Index), per vegetation layer
0	SKYL (% diffuse irradiance (*i.e.*, sky))
150 45	Zenith and azimuth angles of sun upward direction
C:\...\directions.txt	Text file that stores the DART view directions
C:\...\rresu	Generic name for the files that store results

Figure 2: Example of Batch_SAIL.txt (text file that stores input parameters).

The LAD can be Uniform (0), Spherical (1), Erectophil (2), Planophil (3), Extremophil (4), Plagiophil (5), Horizontal (6) or Vertical (7). Other LADs (*e.g.*, Ellipsoidal (8) and Elliptical (9) of DART) can be used.

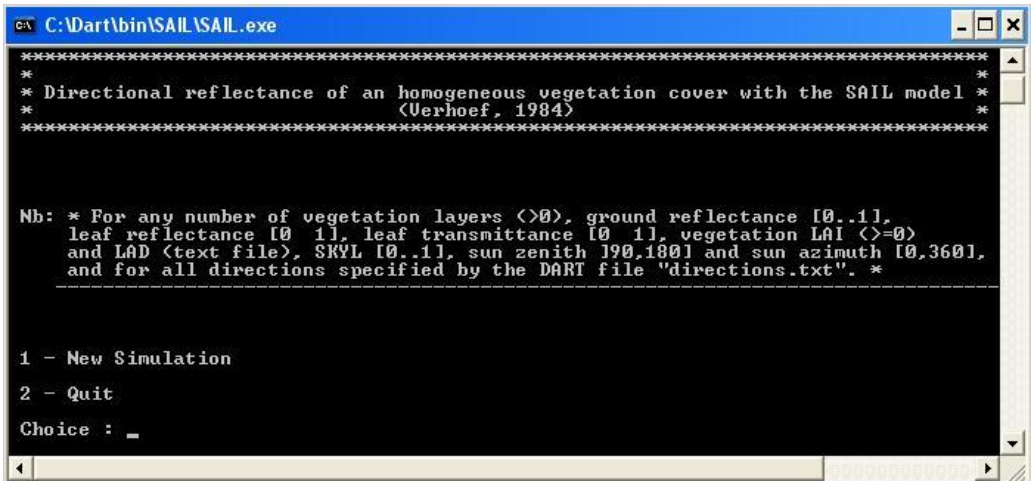


Figure 3: Menu of the SAIL module.
Possible command line: "sail.exe Batch_SAIL.txt"

$$R_s = w \,.\, (1 - SKYL) \,.\, \frac{1-e^{-(k+K).h}}{(k+K)}$$

$$R_d = \frac{\delta_2.(u+v.h_1)}{E_s(0) + E^-(0)} \cdot \frac{1-e^{-(K+g).h}}{(K+g)} + \frac{\delta_3.(u+v.h_2)}{E_s(0) + E^-(0)} \cdot \frac{1-e^{-(K-g).h}}{(K-g)} + (u.C_s+v.D_s).(1 - SKYL) \,.\, \frac{1-e^{-(k+K).h}}{(k+K)}$$

$$R_{ground} = r_{dv}.\frac{\delta_2.h_1}{E_s(0) + E^-(0)}.e^{-(K+g).h} + r_{dv}.\frac{\delta_3.h_2}{E_s(0) + E^-(0)}.e^{-(K-g).h} + (r_{dv}.D_s+r_{sv}).(1-SKYL).e^{-(k+K).h} \,]$$

The SAIL module stores all the below mentioned parameters in the text file "Optional_File":

A : absorption of diffuse fluxes E^+ and E^-, SG : backward scattering of fluxes E^+ and E^-,
sf : forward scattering of direct flux E_s, sb : backward scattering of direct flux E_s,
u : forward scattering of E^+ toward E_v,
v : backward scattering of E^- toward E_v,
w : scattering of E_s toward E_v,
k : extinction of E_s, K : extinction of E_v, g, $h_1 = \frac{a+g}{s}$, $h_2 = \frac{1}{h_1}$, C_s, D_s, δ_2 and δ_3.

r_{dv} and r_{sv} are the "diffuse - direct" and "direct - direct" soil reflectance values.

Figure 4: General expressions of reflectance components that are computed by the SAIL model.

SAIL and DART use the same equations for computing 1st order reflectance if the cover is homogeneous with lambertian leaf elements {reflectance ρ_{leaf}, transmittance τ_{leaf}, LAD: $\frac{g_f(j,\Omega_f)}{2\pi}$}:

- DART 1st scattering order along (Ω_v, $\Delta\Omega_v$) by a turbid volume (type j) that intercepted energy along (Ω_s)

$$\mathbb{W}_{scat,1}(\Delta l_i,\Omega_s\rightarrow\Omega_v) = W_{int}(\Omega_s).T(j,\Omega_s,\Omega_v)$$

$$\text{with } [T(j,\Omega_s,\Omega_v)] = \int_{\Delta\Omega_v} \frac{\int_{2\pi} \frac{g_f(j,\Omega_f)}{2\pi}.|\Omega_s.\Omega_f|.f(j,\Omega_f,\Omega_s\rightarrow\Omega_v).d\Omega_f}{G(j,\Omega_s)}.d\Omega_v \approx \frac{\int_{2\pi} \frac{g_f(j,\Omega_f)}{2\pi}.|\Omega_s.\Omega_f|.f(j,\Omega_f,\Omega_s\rightarrow\Omega_v).d\Omega_f}{G(j,\Omega_s)}.\Delta\Omega_v$$

The scattering function of lambertian leaf elements, is: $f(j,\Omega_s\rightarrow\Omega_v,\Omega_f) = \begin{cases} \frac{1}{\pi}.\rho_{leaf}(j).|\Omega_f.\Omega_v| & \text{if } (\Omega_s.\Omega_f).(\Omega_f.\Omega_v) < 0 \\ \frac{1}{\pi}.\tau_{leaf}(j).|\Omega_f.\Omega_v| & \text{if } (\Omega_s.\Omega_f).(\Omega_f.\Omega_v) > 0 \end{cases}$

A radiation $E_s(z,\Omega_s)$ incident on a cell (height Δz, area S) gives: $W_{int}(\Omega_s) = E_s(z,\Omega_s).S.[1 - \exp(-\frac{u_f.\Delta z}{\mu_s})]$

- SAIL 1st scattering order along (Ω_v, $\Delta\Omega_v$) by a turbid volume that has intercepted energy along (Ω_s)

A vegetation layer dz scatters the radiance:

$$dL_v(\Omega_v) = \frac{w(j,\Omega_s,\Omega_v)}{\pi}.E_s(z,\Omega_s).\mu_s.dz \quad \text{with} \quad w(j,\Omega_s,\Omega_v) = u_f.\int_0^{2\pi} \pi.\frac{|\Omega_s.\Omega_f|.f(j,\Omega_f,\Omega_s\rightarrow\Omega_v)}{|\mu_s.\mu_v|}.\frac{g_f(j,\Omega_f)}{2\pi}.d\Omega_f$$

which leads to: $w(j,\Omega_s,\Omega_v) = \frac{u_f.\pi}{\mu_s.\mu_v}.T(j,\Omega_s,\Omega_v).\frac{G(j,\Omega_s)}{\Delta\Omega_v}$

Actually, the SAIL version at hand does not compute $w(j,\Omega_s,\Omega_v)$ as an integral. It uses:

$$w(j,\Omega_s,\Omega_v) = \sum_{13}^{k=1} \mathbb{P}(\theta_k).w(\theta_k,\Omega_i,\Omega_v) \text{ with } \mathbb{P}(\theta_k) = \int_{\Delta\theta_k} g(\theta_f).\sin\theta_f.d\theta_f \text{ and } w(\theta_k,\Omega_i,\Omega_v) = \frac{u_f}{2\pi}.\int_0^{2\pi} \pi.\frac{|\Omega_i.\Omega_f|.f(\Omega_f,\Omega_i\rightarrow\Omega_v)}{|\mu_i.\mu_v|}.d\phi_f$$

The above approximation explains that DART and SAIL give:

- identical reflectance values for vegetation with schematic LADs. It occurs if leaves are 100% horizontal. In that case, the relationship $w(j,\Omega_s,\Omega_v) = \frac{u_f.\pi}{\mu_s.\mu_v}.T(j,\Omega_s,\Omega_v).\frac{G(j,\Omega_s)}{\Delta\Omega_v}$ is verified.
- different reflectance values for classical LADs (*e.g.*, spherical) where leaf zenith angles span over the whole range [0 π]. Usually, relative differences are around 1-2%.

The file "SAIL_DART.xls" (*cf.* DART documentation) gives DART and SAIL 1st order reflectance values, and their differences, for a scene with LAI = 2 and 5, a spherical LAD, and a sun direction (θ_s=150°, ϕ_s=45°). For large zenith angles, DART accuracy decreases if the LAI of DART cells is too large. The table below shows maximum values of LAI_{cell} for obtaining reflectance with 1% accuracy.

θ_v	50°	62	76	84
$LAI_{cell,max}$	1	0.5	0.25	0.1

$LAI_{cell,max}$ for having 1% accuracy. The reference is {$LAI_{cell,max}$ = 0.025, LAI=5, spherical LAD, ρ_f=τ_f=0.49}

Links for downloading SAIL model:

- Prospect, SAIL and ProSAIL models: http://teledetection.ipgp.jussieu.fr/prosail/
- WinSAIL : http://www.ars.usda.gov/services/software/software.htm

3 *DEMGenerator (to be up-dated)*

This module is used by DART GUI in order to:
- resample an existing DEM (Digital Elevation Models),
- create a theoretical DEM (*e.g.*, bell shape hill), using a limited number of parameters,
- create a DEM, using a number of points (x,y,z) stored in a text file.

A DEM file is actually made of 2 files:
- a file that stores the altitudes, as double float numbers. Its extension is *.mp#*.
- a file that stores the characteristics of the *.mp# file. Its extension is *mpr*.

The use of DEMGenerator module involves several steps:

1) You select DEMGenerator module. It is needed if the DEM does not exist or has not the spatial resolution of current simulation.
2) You select a functionality ('*Theoretical DEM building*', '*Scaling*',…), and possibly a sub-functionality (e.g., '*Gaussian bell*' with '*DEM building*'), an output product name and DEM parameters.
3) You run the module. The new DEM can be used in the "Earth scene". This module can be automatically run in a sequence (*e.g.*, DEM spatial resampling).It can be run also with the menu "Run / DEMGenerator"
4) You specify the DEM of your simulation
5) You can view the DEM

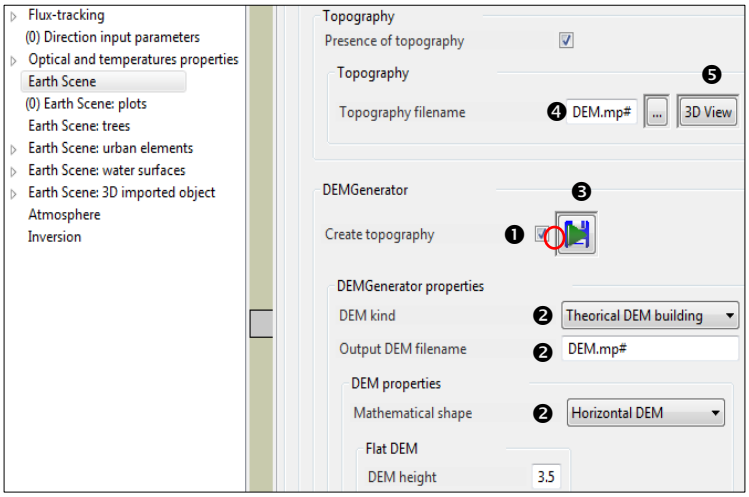


Figure 5. Horizontal DEM creation: 3.5m altitude, same horizontal dimensions as current Earth scene, and 2 symmetric triangles per DART cell. It is useful for having data per triangle in 3D radiative budget.

The 5 functionalities of DEMGenerator are described below.

- Theoretical DEM building: creation of a theoretical DEM. 5 shapes with specific parameters are available:
  0: Horizontal (1 parameter: height) (Figure 5),
  1: Generalized valley (7 parameters),
  2: Concave profile (no parameter),
  3: Gaussian bell (5 parameters) (Figure 7),
  4: Gaussian bell + Concave profile (1 parameter: height)
  5: Sinusoid along x axis (height + number of patterns).

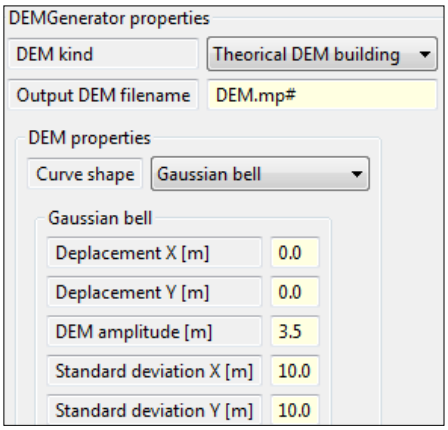
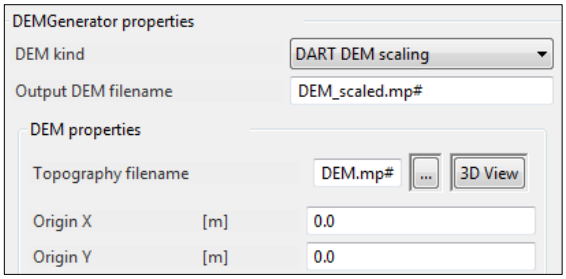
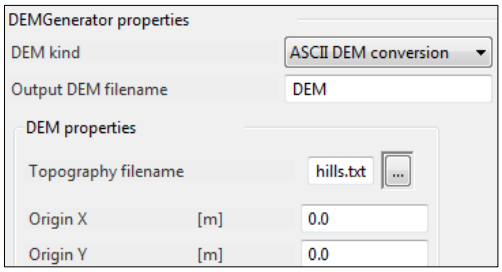


Figure 6. Creation of a Gaussian bell DEM.

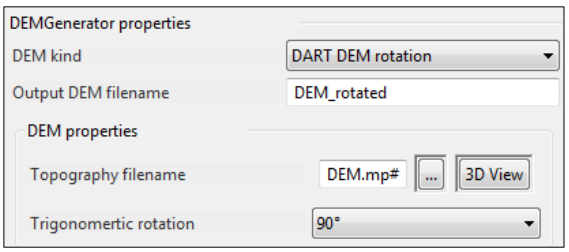
- DART DEM scaling: resampling an already existing DEM.



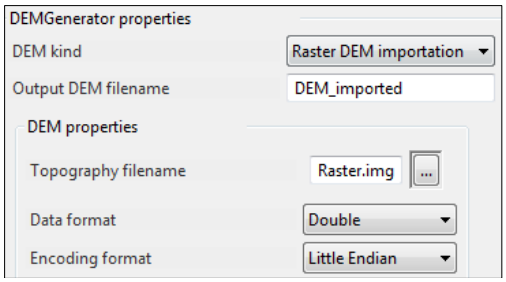
- ASCII DEM conversion: interpolation from an ASCII file (file ".xyz") that stores points (x,y,z).



- DART DEM rotation: 90°, 180° or 270°

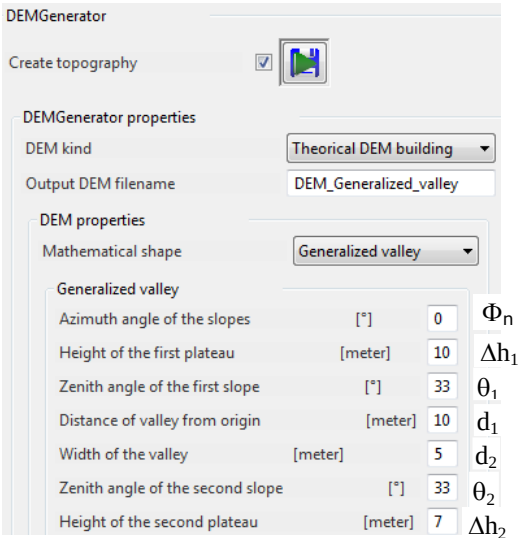


- Rater DEM importation: import of a raw (no header) binary (double or float) raster map that stores pixel altitudes ($h_{00}, h_{01}, h_{02}, \ldots$). Its numbers of lines and columns must be equal to those of the current DART scene.



• **Creation of a generalized valley (slope, etc.)**

It is a rather complex functionality to manage. It will be improved in the future. Examples are described below in order to help the user.



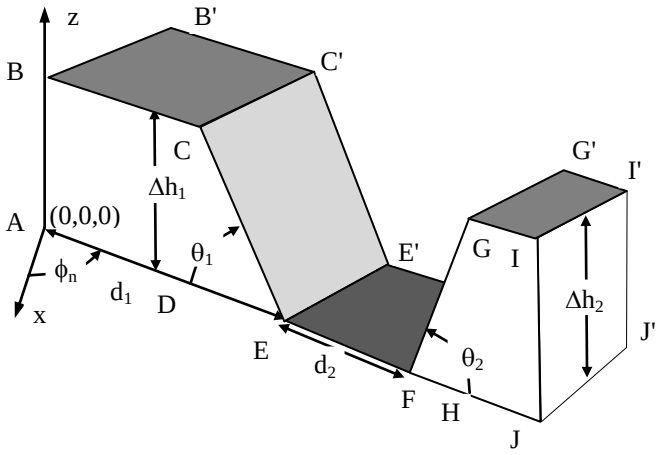


Figure 7. G*eneralized valley*. a) Parameters. Slope angles have priority over plateau heights. b) 3D view.

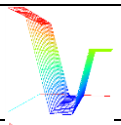
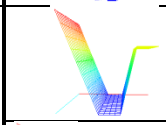
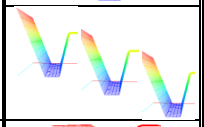
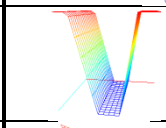
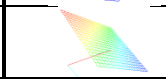
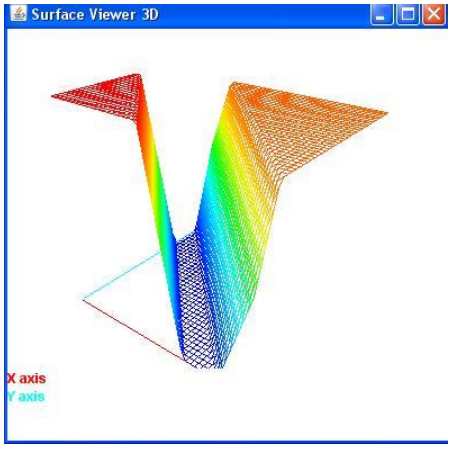
Example	Φ_n	Δh_1	θ_1	d_1	d_2	θ_2	Δh_2	3D view	DART image (θ_v=θ_s=0°; 1st scat. order)
1	0	5	45	8	4	45	3		
2.a (scene: repetitive)	0	5	30	8	4	45	3		
2.b (scene: infinite slope)									
3	0	3	30	8	4	45	3		
4	45	10	20	27	x	x	x		20° slope with a 45° azimuth angle

Figure 8. *Examples of topography simulated with the "generalized valley" option.*

a) 

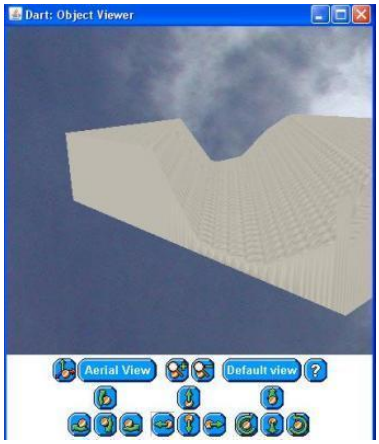
b) 

Figure 9. *a) Theoretical DEM with an azimuth rotation (Φ_n=45°). b) DEM actually built by DART.*

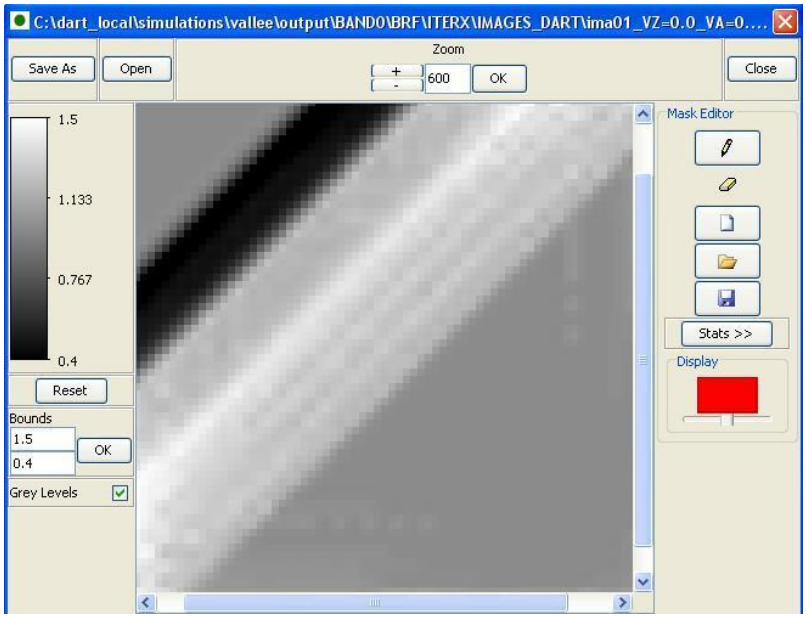
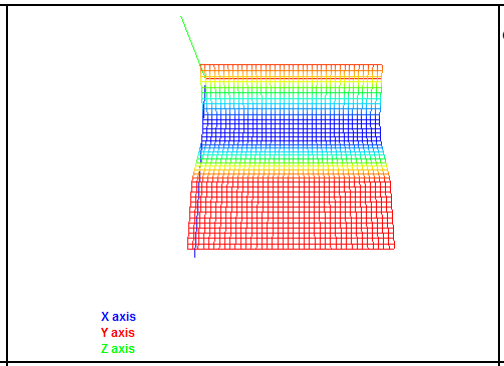
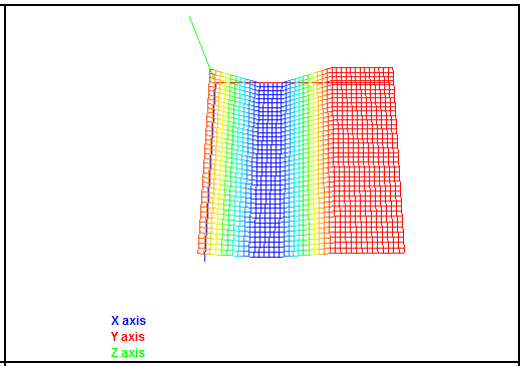
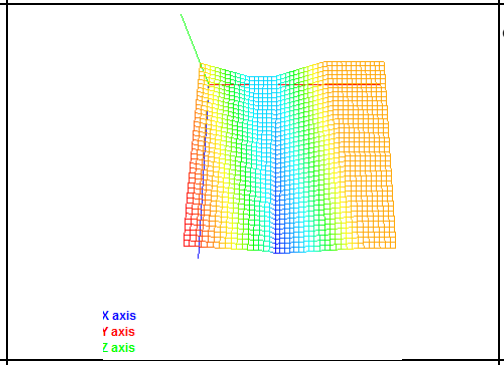
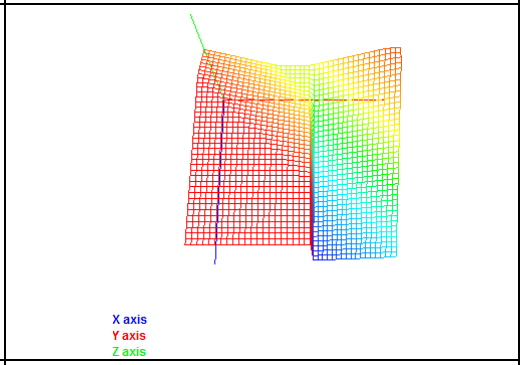
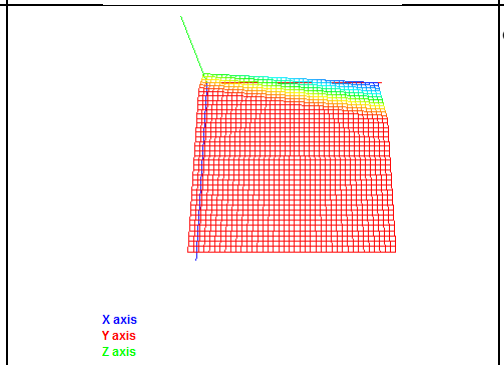
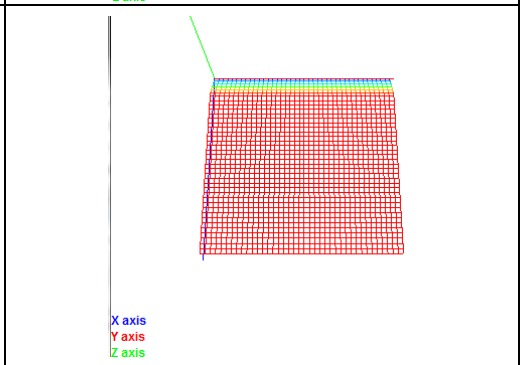
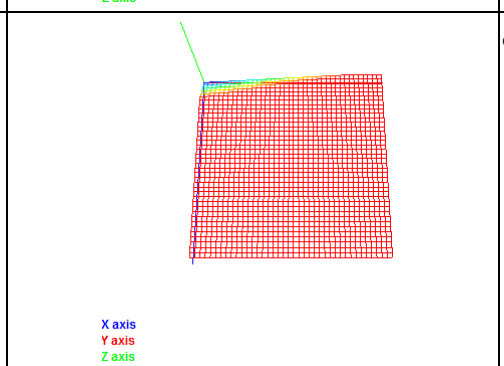
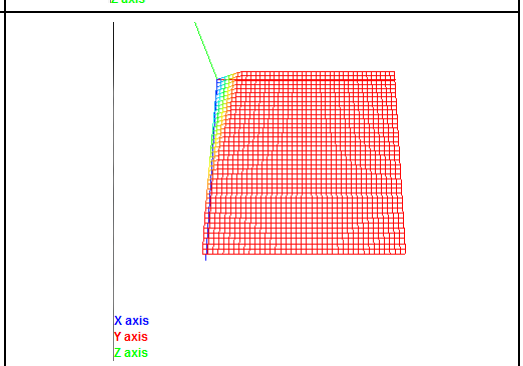
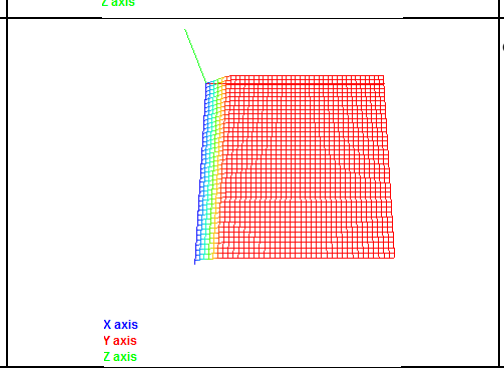
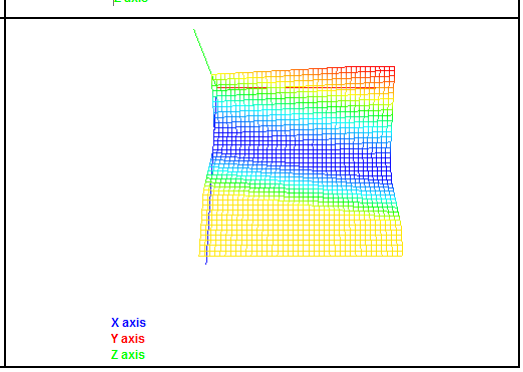


Figure 10. *DART image of a scene with an azimuth rotation. ρ_{ground} = 1. θ_s = 30°. ϕ_s = 45°. SKYL=0. 40m x 40m.*

Because it has the higher altitude (8m), the top left plateau reflectance is 1. It is nearly the same for the lower right plateau. Being shaded, the first slope has a 1st order reflectance equal to 0. Here, it is ≈0.4 because it is illuminated by the opposite slope. Conversely, being very well illuminated, the 2nd slope has a reflectance larger than 1 (≈1.5).

Border (shadow) effects can appear if DART scene is "repetitive" (*i.e.*, repetition of a basic pattern) with different altitudes on its left and right sides. These effects do not occur if the scene is "infinite slope" (example 2.b in above table)

Generalized Valley with default parameters and variable azimuth angle Φ_n

Φ_n = 0°		Φ_n = 90°	
Φ_n = 100°		Φ_n = 135°	
Φ_n = 170°		Φ_n = 180°	
Φ_n = 190°		Φ_n = 260°	
Φ_n = 270°		Φ_n = 350°	

Algorithm

Input parameters

- $\theta_{n1} \,]0\ \frac{\pi}{2}]$, $\theta_{n2} \,]0\ \frac{\pi}{2}]$, $\phi_n \,[0\ \frac{\pi}{2}]$, $\Delta h_1 \geq 0$, $\Delta h_2 \geq 0$, $d_1 \geq 0$, $d_2 \geq 0$
- maket parameters: Nb_lig, Nb_col, Δx, Δy

Pre-computation

- C $(x_C,y_C,0)$: $x_C = d_1.\cos\phi_n$, $y_C = d_1.\sin\phi_n$
- D $(x_D,y_D,0)$: $x_D = (d_1 + d_2).\cos\phi_n$, $y_D = (d_1 + d_2).\sin\phi_n$

Equations

- Line{$\perp$ OC, C) : $x.\cos\phi_n + y.\sin\phi_n = d_1$
- Line{$\perp$ OD, D) : $x.\cos\phi_n + y.\sin\phi_n = d_1 + d_2$
- Plane $\{\Omega_n(\theta_{n1},\phi_n)$, C $(x_C,y_C,0)\}$: $(x - x_C).\sin\theta_{n1}.\cos\phi_n + (y - y_C).\sin\theta_{n1}.\sin\phi_n + z.\cos\theta_{n1} = 0$

  $z = \frac{-1}{\cos\theta_{n1}}.[(x - x_C).\sin\theta_{n1}.\cos\phi_n + (y - y_C).\sin\theta_{n1}.\sin\phi_n]$ if $(\theta_{n1} \neq \frac{\pi}{2})$
- Plane $\{\Omega_n(\theta_n,\phi_n+\pi)$, A $(x_D,y_D,0)\}$: $-(x - x_D).\sin\theta_{n2}.\cos\phi_n - (y - y_D).\sin\theta_{n2}.\sin\phi_n + z.\cos\theta_{n2} = 0$

  $z = \frac{1}{\cos\theta_{n2}}.[(x - x_D).\sin\theta_{n2}.\cos\phi_n + (y - y_D).\sin\theta_{n2}.\sin\phi_n]$ if $(\theta_{n2} \neq \frac{\pi}{2})$

❶ if $(\Delta h_1 \neq 0)$ and $(\theta_{n1} \neq \frac{\pi}{2})$ for(i=0;i++;Nb_lig - 1) for(j=0;j++;Nb_col - 1)

```
if (θn1 = 0)                                          /* Vertical slope ⇒ very oblique slope */
    if (cosφn ≠ 0)                                    /* Slope not parallel to the y axis */
        if (i.Δx.cosφn < d1 - j.Δy sinφn)   z(i,j) = Δh1   /* Horizontal plateau, left of line {⊥ OC, C) */
    else if (j.Δy.sinφn < d1)   z(i,j) = Δh1          /* Horizontal plateau, left of line {⊥ OC, C) */
else  {z(i,j) = -tanθn1.[i.Δx.cosφn + j.Δy.sinφn – d1]   /* Not a vertical slope */
      if (z(i,j) > Δh1)   z(i,j) = Δh1                /* Horizontal plateau, left of line {⊥ OC, C) */
      if (z(i,j) < 0)   z(i,j) = 0                    /* Valley and plateau right of line {⊥ OC, C) */
else z(i,j) = 0                                       /* Horizontal landscape */
```

❷ for(i=0;i++;Nb_lig - 1) for(j=0;j++;Nb_col - 1)

```
if (cosφn ≠ 0)                                        /* Slope not parallel to the y axis */
    if (i.Δx.cosφn > d1 + d2 - j.Δy sinφn)            /* Zone, right of line {⊥ OD, D) */
        if (θn2 = 0)   z(i,j) = Δh2                   /* Vertical slope: horizontal plateau */
        else  {z(i,j) = tanθn2.[i.Δx.cosφn + j.Δy.sinφn - d1 - d2]   /* Slope */
                if (z(i,j) > Δh2)   z(i,j) = Δh2}     /* Horizontal plateau */
else if (j.Δy.sinφn > d1 + d2)   z(i,j) = Δh2         /* Slope parallel to the y axis */
    if (θn2 = 0)   z(i,j) = Δh2                       /* Vertical slope: horizontal plateau */
    else  {z(i,j) = tanθn2.[i.Δx.cosφn + j.Δy.sinφn - d1 - d2]   /* Slope */
            if (z(i,j) > Δh2)   z(i,j) = Δh2}         /* Horizontal plateau */
```

4 *Vegetation(to be up-dated)*

Note: The exercise "Land_Cover" illustrates how to use the Vegetation module.

I. Objective of the Vegetation module

The Vegetation module creates a scene for DART multispectral mode. This scene is a juxtaposition of quadrilaterals, called "plots", made of "ground", "vegetation" or "ground + vegetation". In order to simulate this scene, the Vegetation module reads a raster map (*e.g.*, classification map of a satellite image), called COS map, where each pixel count number indicates a land cover class that is either "ground", "vegetation" or "ground + vegetation" with specific characteristics.

"Ground" plot: opaque surface characterized by their optical properties: type of surface reflectance (*i.e.*, "lambertian", "lambertian + specular" or "Hapke") and data file that contains the spectral reflectance values.

"Vegetation" plot: volume of turbid medium that is characterized by:

- geometric properties: height of the base, depth, depth standard deviation, Leaf Area Index (LAI) or leaf volume density (m^2/m^3),
- optical properties: data file that stores spectral optical values: leaf reflectance and transmittance values, specular coefficients and clumping coefficients.

"Ground + Vegetation" plot: "ground" plot with a "vegetation" plot on top of it.

Vegetation module works with input parameters stored in a "plots.xml" file that is created by the editor of DARTIHMSimulationEditor simulation. It works only in multispectral mode. If necessary, the monospectral case can be simulated using the multispectral mode with one spectral band, only.

Vegetation module is run from the DART Launcher ("Run / Vegetation"). It loads the current simulation and creates the missing phase functions of phase and the plots, starting from the COS.

The characteristics of all possible plots are given in a text file provided by the user: one line for each pixel count number of the input raster map in order to set the type of plot and its properties.

The input raster map, or a sub-zone of it, is read line per line. This implies that plots are horizontal rectangles, with no dependence between rectangles of successive lines. The raster map is read using information provided by the user:

- Numbers of lines and columns;
- Presence of a sub-zone, and definition of it (line and column of the top left corner, and number of lines and columns);
- Code of pixel values: byte (default) or float or double;
- Size of a pixel (ordinate and abscissa) in meter into the simulation.

This information is stored in the plots.xml file. This input file is modified by the Vegetation module.

A run of the Vegetation module updates 2 output XML files:

- "plots.xml": coordinates of the 4 corners of all plots. All previous plots in the plot.xml file **are crushed** by default, except if the "override existing plots" option is un-checked. This option allows the user to add plots from different land cover maps. It must be used with care! Moreover, the user can input plots directly in the DART scene editor simulation, before (with option " override existing plots") or after the use of the land cover map. The plots.xml contains also a link to the optical properties for each plot and other plot properties (LAI, height …).

- "coeff_diff.xml": optical properties. Optical properties already defined in the "coeff_diff.xml" file **are not crushed**. The optical properties associated to the input raster file are simply added to the "coeff_diff.xml" file. Optical properties that already exist (*i.e.*, same optical text file) are not added to the "coeff_diff.xml" file in order to avoid duplicate definitions of optical properties.

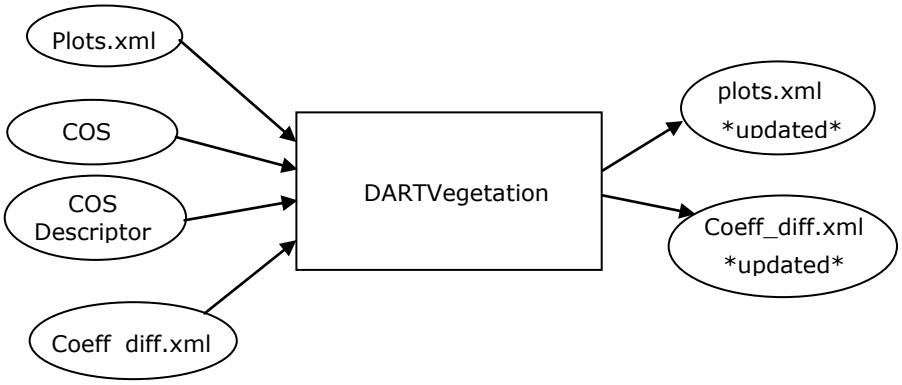


Figure 11. *Dataflow of the vegetation module*

II. Input parameters for the Vegetation module

- **Land cover map**. This raster (binary) file contains pixels coded in byte, float or double. Its characteristics (numbers of lines and columns, file name, number of bytes for coding pixels, pixel dimension) are input in the GUI by the user.
- **Description of the land cover map.** This text informs on each class of the land cover map (plot height, LAI, optical properties...).
- Possibility to work with a **sub-scene** of the vegetation map

Class name	N°	Plot	Plot type	Soil type	Soil file	Veg. file	LAI	HBase	H	σ_H	T_{mean}	ΔT	Ω_{min}	Ω_{max}	a	b	LAD	ALA or Thetam	Eccen-tricity	Leaf size (m)
String	*Integer*				*String*	*String*	*Double*						*Double*				*Integer*	*Double*		
Class1	*1*	*0*	*0*	*2*	*2D_argile.txt*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*
Class2	*2*	*1*	*1*	*0*	*0*	*3D….txt*	*2.5*	*0*	*2*	*0*	*270*	*300*	*1*	*0*	*0*	*0*	*1*	*0*	*0*	*0.05*
Class3	*3*	*1*	*2*	*0*	*2D_asphalt.txt*	*3D_g...txt*	*2.3*	*0*	*3*	*0*	*250*	*280*	*1*	*0*	*0*	*0*	*8*	*1.25*	*0*	*0.05*
Class4	*4*	*0*	*0*	*1*	*2D_snow.txt*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*
Class5	*5*	*1*	*2*	*2*	*2D_snow.txt*	*3D_f...txt*	*3*	*0.25*	*4*	*0*	*265*	*275*	*1*	*0*	*0*	*0*	*1*	*0*	*0*	*0.05*
Class6	*6*	*1*	*2*	*1*	*2D_ice.txt*	*3D_h...txt*	*1.2*	*0.15*	*1*	*0*	*255*	*295*	*1*	*0*	*0*	*0*	*9*	*0.65*	*1.25*	*0.05*
Class7	*7*	*0*	*0*	*1*	*2D_sandy.txt*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*
Class8	*8*	*1*	*1*	*0*	*0*	*3D_....txt*	*1.6*	*0*	*2*	*0*	*300*	*320*	*1*	*0*	*0*	*0*	*2*	*0*	*0*	*0.05*
Class9	*9*	*0*	*0*	*2*	*2D_ice.txt*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*

Figure 12. Land cover map descriptor for the Vegetation module.

There are 21 fields per line separated by blank characters or tabulations.

The parameters listed below appear in the same order as in the text file.

- Name of the class: name of the current class, without blank character! It gives the name of the associated phase function in DART.
- Number of the class: index of the class.
- Presence of plot: 1 if a plot uses the class, 0 if not
- Type of plot: -0: ground type,
  - 1: vegetation type,

- 2: ground type + vegetation

- Type of soil phase function: - 0: lambertien
  - 1: lambertien + specular
  - 2: Hapke
- File that stores soil spectral signature: This file should be in directory *database* for best performance, either in directory $DART_HOME or in directory $DART_LOCAL. A "0" indicates if there is no ground.
- File that stores vegetation spectral signature: This file should be in directory *database* for best performance, either within directory $DART_HOME or within directory $DART_LOCAL. A "0" indicates if there is no vegetation.
- LAI: vegetation LAI.
- HBase: Height of the base of the vegetation above the ground
- H: Average height of the vegetation
- σ_H: Standard deviation height of the vegetation
- T_{min}: Minimal temperature of the vegetation in Kelvin
- T_{max}: Maximum temperature of the vegetation in Kelvin
- Ω_{min}, Ω_{max}, a, b: Directional weights of the function G for the vegetation phase function (default values (1,0, 0, 0).
- LAD: Angular distribution of the vegetation

0: Uniform	2: Erectophile	4: Extremophile	6: Horizontal	8: Ellipsoidal
1: Spherical	3: Planophile	5: Plagiophile	7: Vertical	9: Elliptical

If the LAD is Ellipsoidal, then: - ALA. If the LAD is Elliptical, then: Thetam and Eccentricity.

- ALA or Thetam: depending on the LAD, this column can store the LAD or the Thetam (θ_m) value.
- Eccentricity: The eccentricity (ε) value of the vegetation.
- Leaf dimension (m). Used for simulating hot spot.

III. Output files of the Vegetation module

- **plots.xml.** This file stores the plots.
- **coeff_diff.xml**. This file stores the optical properties (2D and 3D data bases).

Note: Vegetation module is written in JAVA.

IV. Most common know errors

- **"No COS file found!" or "No COS descriptor found!"**
  Be sure these files are correctly located. They are usually put in the simulation input directory.
- **"Phase X was not found in COS descriptor!"**
  Check if the phase is present in the COS descriptor file. If so, check if you are using the right COS descriptor for your simulation.
- **"Missing elements in Phase X!"**
  Check the COS descriptor and correct it according to its format (cf. above).
- **"Sub-zone defined outside COS file!"**
  This is not exactly an error message. It warns you that the sub-zone expands outside the COS limits. However, the system will not crash. The module will try to redefine a sub-zone, the largest possible in the COS file, according to the provided information.

5 *Lidar (to be up-dated)*

The Lidar module

The Lidar module simulates waveforms of Earth scenes (urban / natural landscapes with topography) with atmosphere. Earth scenes are simulated with turbid cells and/or translucent / opaque triangles. Atmosphere is simulated with turbid layers/cells the optical properties of which differ from those used for simulating radiance with DART flux tracking method. Indeed, for simulating radiance, cells/layers are characterized by constant extinction coefficients, and the associated optical depths. On the other hand, for lidar simulation, the extinction coefficients within atmosphere cell/layers are not constant. Indeed, sun induced TOA radiance depends on the atmosphere optical depth, whereas lidar signal depends on the vertical profile of the atmosphere components extinction coefficients. Thus, all atmosphere components extinction coefficients are simulated as continuous functions in any cell/layer, with a continuity for each layer. The formalism of these functions minimizes computer time when determining the location of the points in the atmosphere where scattering processes take place.

Photon propagation in the atmosphere and the Earth scene is modeled with a so-called RayCarlo method that combines Ray Tracing and Monte Carlo methods. The Monte Carlo method is used for determining the direction of the photons that are scattered at each scattering event and for determining the location of scattering points in turbid cells (atmosphere,…). The ray tracing method is used for defining the fraction of interception of 1st order photons that cross an atmosphere layer, the photon path length in a turbid cell, the ray fraction that is directed to the Lidar in a scattering event, and the scattering and absorption fractions for any intercepted photon. Thus, one does not track "conventional" photons: one tracks photons with weights that are float numbers. This allows one to simulate Lidar signals with reasonable computer times, without using the classical reverse Monte Carlo method.

2 factors affect the spatial distribution of photons (*i.e.*, angular distribution) that are launched:

1) <u>*Lidar illumination configuration*</u>

   It is defined s defined by the origin of photons (*i.e.*, Lidar position) and by a grid of points at BOA level. Each point is a cell face subcenter (i,j), within the lidar illumination BOA footprint. The number $N_{subcenters}$ of subcenters within the illumination footprint is defined by the footprint dimension and by the number $N_{sub\ centers\ per\ cell}$ of face sub centers per cell (*i.e.*, squared value of the input parameter "Nb of axial-subdivisions for scene illumination", in the expert mode menu).

   The illumination footprint can have any "shape" (being implemented). Per default, it is Gaussian: Gauss distribution of relative weight $w_{subcenter}(i,j)$ of each subcenter (i,j), with $\sum_{i,j} w_{subcenter}(i,j) = 1$, and $w_{subcenter}(i,j)$ being a float number and not an integer number.

2) <u>*Simulation computer time and computer memory*</u>.

   They depend on the number of DART photons that are launched: $N'_{photons,DART} = \Sigma_{i,j}\ N_{photon,subcenter}(i,j)$ photons that are launched per subcenter (i,j), knowing that the number of DART photons to launch per subcenter is an integer number and must be larger than a specified number $N_{photon,subcenter,min}$ (*i.e.*, DART input parameter "Minimum photons number per subcenter"). The minimum number $N_{photons,DART,input}$ of DART photons to launch is an input parameter.

   Because photon numbers are integer numbers, we have:

   - Minimal number of photons to launch per subcenter: $Round(w_{subcenter}(i,j) \,.\, N_{photons,DART})$
   - Minimal total number of photons to launch: $N'_{photons,DART\ input} = \sum_{i,j} Round(w_{subcenter}(i,j) \,.\, N_{photons,DART})$

Let (i',j') be the subcenter for which $w_{subcenter}(i,j)$ is smaller. We have:

$$N_{photon,subcenter}(i,j)=Round(Max\{Round(w_{subcenter}(i',j').N_{photons,DART,input}),N_{photon,subcenter,min}\}.\frac{w_{subcenter}(i,j)}{w_{subcenter}(i',j')}.N_{photons,DART})$$

Thus, the total number of photons that are actually launched is $N'_{photons,DART}=\sum_{i,j} N_{photon,subcenter}(i,j)$

Actually, the lidar pulse is characterized by its energy and wavelength. It corresponds to $N_{photons,Lidar}$ monochromatic photons. Thus, each DART photon has the weight: $\frac{N_{photons,Lidar}}{N'_{photons,DART}}$. This weight is used for providing DART results in terms of the actual number of photons that a Lidar launches, and not in terms of the photons that are launched by DART.

When a Lidar simulation starts, DART computes $N'_{photons,DART}$. If it is larger than $N_{photons,DART,input}$, the DART menu displays the number $N'_{photons,DART}$ of photons that are actually launched.

The lidar signal (*Figure 14*) is defined by its pulse, geometry and acquisition parameters. These parameters are described below.

- **Pulse energy**

  The pulse energy W_{pulse} (*e.g.*, 1 mJ) gives the number N_{phot} of monochromatic photons (frequency ν) that an actual Lidar system would launch: $N_{phot}=\frac{W_{pulse}}{h.\nu}$. This number is usually very large. So, DART launches a number of photons that is usually much smaller than N_{phot}.

- **Gaussian pulse duration**

  The pulse is a centered Gauss function $A.exp[-\frac{t^2}{2.\sigma_t^2}]$ with an amplitude A and a standard deviation σ_t.

  - *Half pulse duration (effective)*: it is equal to $n_\sigma.\sigma_t$, with σ_t the standard deviation and n_σ a number. It means that the signal duration is $[-n_\sigma.\sigma_t \ +n_\sigma.\sigma_t]$.
  - *Sigma definition (σ_t)*: the Gauss pulse standard deviation is defined by the time Δt_x (ns) when the pulse amplitude A is divided by the pulse relative amplitude x.

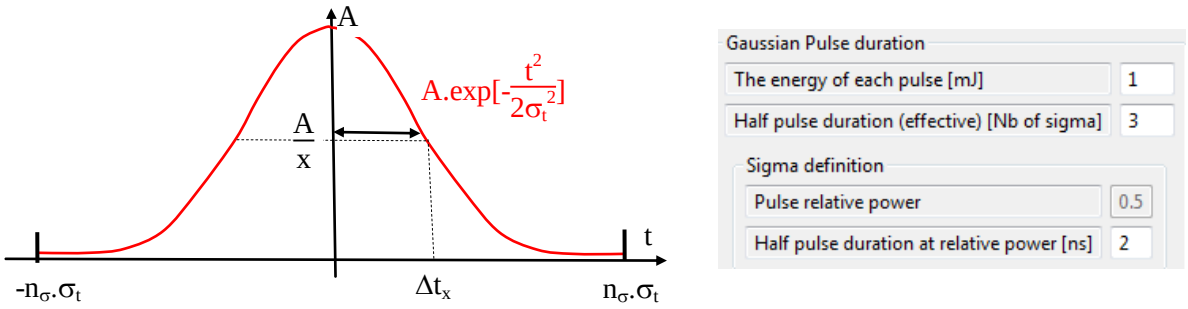


Figure 13: Lidar Gauss pulse.

- **Lidar geometry**

  The Lidar geometry is defined:

  - *Lidar altitude*:
  - *Area of lidar sensor*: this parameter is used to determine the solid angle under which the lidar is seen from any point in the atmosphere and in the earth scene.
  - *Zenith angle and azimuth angle*: they define the lidar view direction.

- *Lidar altitude*:
- *Lidar position (x,y)*: location of the central part of the illuminated footprint over the Earth scene.
- *Center position of illumination*: Location (x, y) of the footprint of illumination at the BOA level. It is also the center of the footprint of measurement.
- *Lidar radius footprint (illumination)*: radius at BOA level. Its projection on the plane perpendicular to the lidar direction is noted $r_{lighted}$.
- *Lidar radius footprint (scattering)*: radius at BOA level.
- *Maximum / Minimum photon path within the scene*: the specified Min and Max values indicate that photon information is stored only for photon paths that belong to the interval

[Min * Earth scene height Max * Earth scene height].

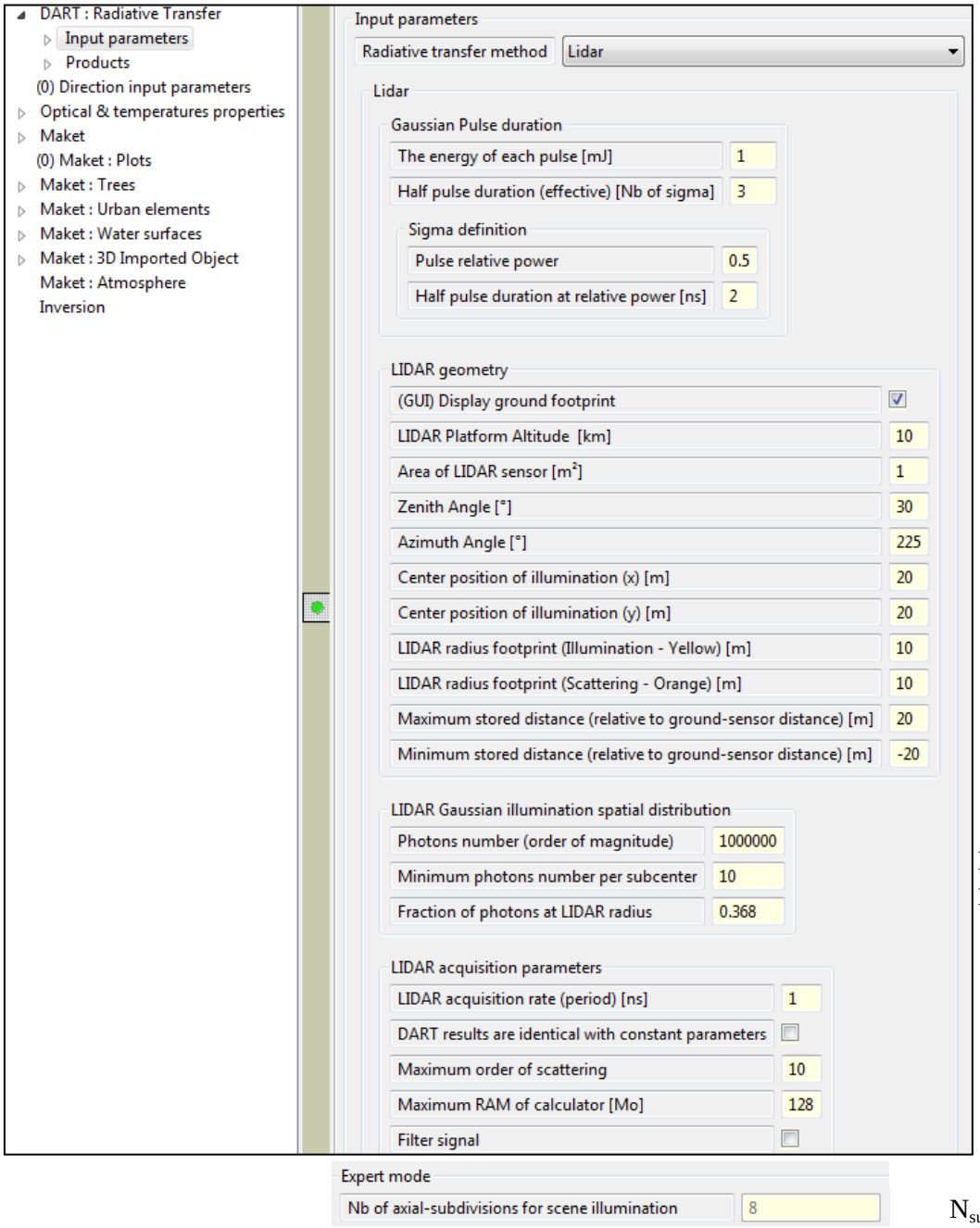


Figure 14: Menu of the lidar module

- **Lidar Gaussian illumination spatial distribution**

  The illumination photons have a Gauss spatial distribution $B.\exp[-\frac{r^2}{2\sigma_r^2}]$ with a standard deviation σ_r.

  - *Photons number*: number of photons that DART should launch is $\int_0^{r_{lighted}} B.\exp[-\frac{r^2}{2\sigma_r^2}].dr$. The
  - *Minimum photons number per subcenter*: $N_{photon,subcenter,min}$ It gives the minimum number of photons that must illuminate each sub center of the top faces of the cells that make up the illumination footprint at BOA level.
  - *Fraction of photons at Lidar radius*: η. It allows one to compute σ_r, using $\eta = \exp[-\frac{r_{lighted}^2}{2\sigma_r^2}]$.

- **Lidar acquisition parameters**

  General parameters that characterize lidar acquisition.

  - *Reception solid angle* $\Delta\Omega_{lidar}^{dart}$: the direction $(\Omega_{lidar}^{dart},\Delta\Omega_{lidar}^{dart})$ is added to the DART discrete directions. Thus, the actual lidar signal is equal to the DART result multiplied by $\Delta\Omega_{lidar} / \Delta\Omega_{lidar}^{dart}$.
  - *Lidar acquisition rate*: rate that is used for binning the waveform. 1ns corresponds to a 30cm path, which corresponds to a 15cm altitude difference for a two ways path.
  - *DART results are identical with constant parameters*: simulations that are conducted with the same parameters (scene dimensions, number of photons, etc.) give the same results. Indeed, in that case, random numbers of successive simulations are the same.
  - *Maximum order of scattering.* Number of scattering events that a photon can undergo.
  - *Maximum RAM of calculator*: information that is used for optimizing the RAM management.
  - *Filter signal*: filtering implies that the first and last bins are not empty, which defines the first and last bins. With "no filtering", the first and last bins are defined by the values of the "Maximum and Minimum stored distance" parameters.

- **Lidar products**

  Products of the Lidar module are images (Figure 15) and text files (Figure 16, DART User manual):

  - *Images*: 2 images are created. The Lidar ground image displays the 2D distribution of photons that are intercepted by the ground. The Lidar BOA image displays the photons that are scattered by the Earth scene within the field of view of the Lidar. The image is in the plane perpendicular to the lidar view direction.
  - *pulse.txt*: time pulse function. It is used for convolving the Lidar waveform.
  - *stat_illumination_MC0.txt*: statistics of illumination per spectral band. For order 0: numbers of photons that are launched, that escaped the scene, that were absorbed, that reached the ground (index 1), that were lost,… For order 1: numbers of photons that are scattered and absorbed, etc.
  - *photonInformations.txt*: It can be very large. It is used for the Lidar 3D view. It gives for each photon that is measured by the Lidar:
    - the photon total path,
    - the number of scattering events,
    - the photon weight,
    - the bin index,

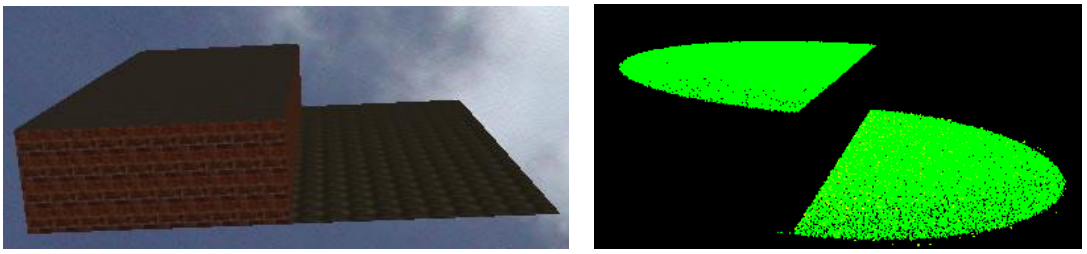
- the type of last scatterer,
- a boolean (O/1) that indicates if the photon interacted with the ground,
- the coordinates (x,y,z) of the last scattering point.

- *LIDAR_DART_wave.txt*: not convolved waveform. 4 columns: bin, time, length, number of photons.

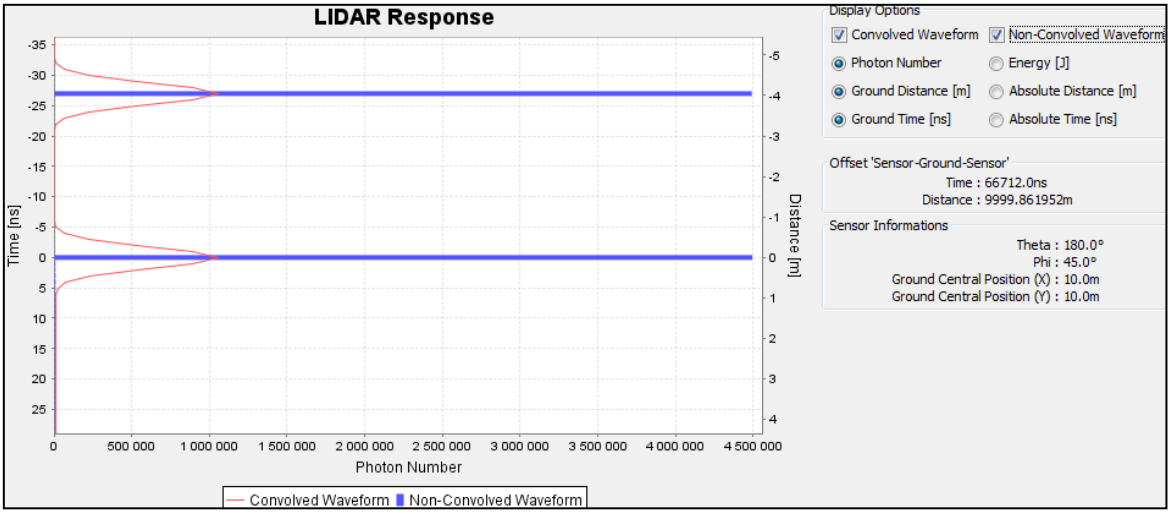
- *LIDAR_CONVOLVED_wave.txt*: the convolved waveform. It has 2 columns (time, number of photons) and can be displayed as a graph, using the "Display menu" of DART.

- *lidarInformations.txt*: lidar position (X, Y), footprints $r_{lighted}$ and r_{view}, θ_{lidar}, ϕ_{lidar}, ground path length ground bin number total number bin duration, σ_t, number of σ_t in the pulse, pulse length, number of bins in the pulse, pulse itself.

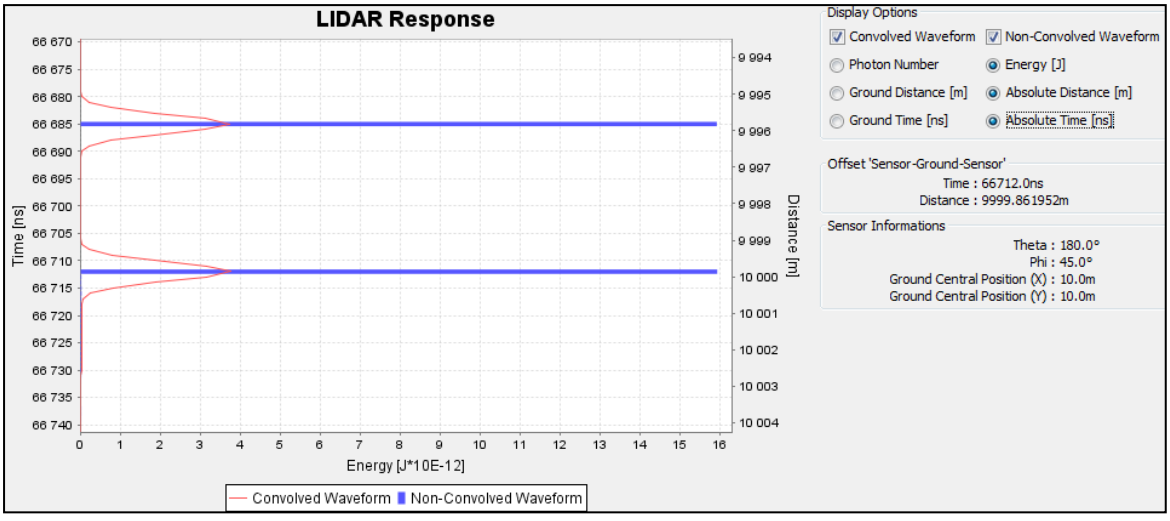
a)

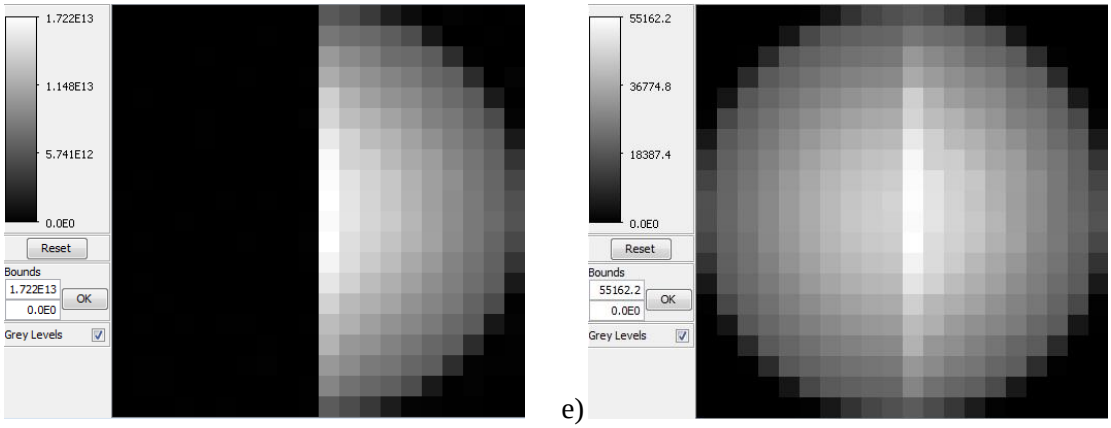


b)



c)





d) e)

Figure 15: a) DART earth scene: ground + house with flat roof. Reflectance = 1. b,c) Raw and convolved waveforms in terms of photon number/energy, and ground/absolute distance and time. d) Lidar photons incident on the ground. e) Upward lidar photons over the scene.

a)

105	-2.8000000000000000e+001	-8.394188824000000e+000	0.0000000000000000e+000	
106	-2.7000000000000000e+001	-8.094396366000000e+000	4.490107920569540e+006	House roof
107	-2.6000000000000000e+001	-7.794603907999999e+000	0.0000000000000000e+000	
131	-2.0000000000000000e+000	-5.995849160000000e-001	0.0000000000000000e+000	
132	-1.0000000000000000e+000	-2.997924580000000e-001	0.0000000000000000e+000	
133	0.0000000000000000e+000	0.0000000000000000e+000	4.492139310518254e+006	Ground
134	1.0000000000000000e+000	2.997924580000000e-001	8.640721853718715e+003	
135	2.0000000000000000e+000	5.995849160000000e-001	9.323761938472160e+003	
136	3.0000000000000000e+000	8.993773740000000e-001	1.069038673172476e+004	Interactions
137	4.0000000000000000e+000	1.199169832000000e+000	1.018127509864334e+004	"ground - house wall"
138	5.0000000000000000e+000	1.498962290000000e+000	1.128041064790022e+004	
139	6.0000000000000000e+000	1.798754748000000e+000	1.078779566544153e+004	
140	7.0000000000000000e+000	2.098547206000000e+000	7.970918407654935e+003	
221	8.8000000000000000e+001	2.638173630400000e+001	1.599519141566407e+002	

b)

0	-5.000000e+000	3.088973e-003
1	-4.000000e+000	1.469372e-002
2	-3.000000e+000	4.942357e-002
3	-2.000000e+000	1.175497e-001
4	-1.000000e+000	1.976943e-001
5	0.000000e+000	2.350995e-001
6	1.000000e+000	1.976943e-001
7	2.000000e+000	1.175497e-001
8	3.000000e+000	4.942357e-002
9	4.000000e+000	1.469372e-002
10	5.000000e+000	3.088973e-003

c)

```
Spectral band number : 0
        launched photons :              49664.9
        Photons out :                   0 (0%)
        Photons absorbed :              0 (0%)
        Photons lost :                  10 (0.0201349%)
        Photons that reached the ground :       28298 (56.9778%)
        Photons under the ground :      0 (0%)
        Maximum order of scattering:    11
        Maximum order of scattering within a cell:      9
 Order 1 :
        Scattered photons : 51684 (104.065%)
        Absorbed photons : 0 (0%)
```

d)

1.99919100e+004	1	1.59283634e-009	106	12	0	8.89760450e+000	3.88982306e-002	4.05000000e+000
1.99919101e+004	1	1.59283633e-009	106	12	0	8.88756278e+000	1.77130297e-002	4.05000000e+000
2.00000101e+004	1	1.59154640e-009	133	2	1	1.10700363e+001	1.99747717e+001	0.00000000e+000

Figure 16: a) Raw waveform (LIDAR_DART_wave.txt). b) Pulse (pulse.txt). c) Statistics (stat_illumination_MC0.txt). d) DART photon information(PhotonInformations.txt): Photon total path, Number of scattering events, Photon weight, Bin index, Type of last scatterer, Boolean (O/1) that indicates if the photon interacted with the ground, Coordinates (x,y,z) of the last scattering point. Presence of multiple scattering events.

Below sections present Lidar modules that process DART and/or LVIS (https://lvis.gsfc.nasa.gov/, NASA) waveforms. These modules can be applied to any actual waveform with correct format.

LVIS waveform format:

- 1st line: longitude, latitude and altitude of 1st bin, longitude, latitude and altitude of last bin, mean noise
- 1 column that stores 1 bin value per line

The Deconvolution module (Deconvolution.bat)

This module deconvolves a Lidar waveform. It is adapted to DART files format.

Theory:

The emitted Lidar signal x(t) is a pulse and not a Dirac function (*i.e.*, null duration). Thus, the Lidar response y(t) of an horizontal surface is an impulse response h(t) and not a Dirac function. It is the convolution "Signal measured with a Dirac initial Lidar - Impulse response h(t)". With a noise N(t):

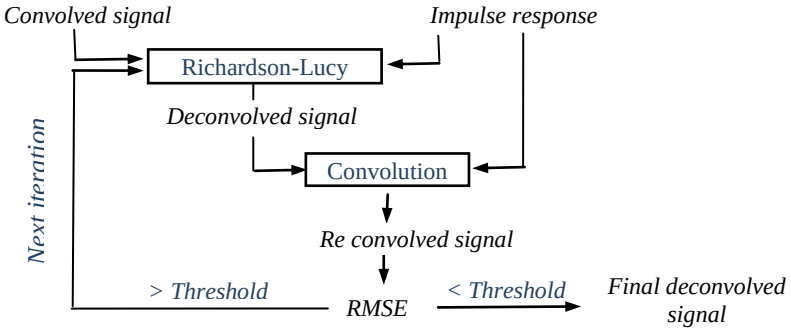
$$y(t) = x(t)*h(t) + N(t) = \int_{+\infty}^{-\infty} x(t).h(t-\tau).d\tau + N(t) = y[n] = \sum_{k=-\infty}^{k=+\infty} (x[k].h[n-k] + N[n]) + N[n]$$

The aim is get an estimate $\hat{x}(t)$ of x(t), starting from y(t) and knowing h(t). Here, the Richardson-Lucy (Richardson, 1972; Lucy, 1974) algorithm is used. It can be applied in presence of noise.

$$\hat{x}_{i+1}[n] = \hat{x}_i[n].\{\frac{y[n]}{h[n] * \hat{x}_i[n]} * h[n]\} = \hat{x}_i[n]. \sum_{k=0} \{\frac{y[n]}{\hat{x}_i[j] * h[n-j]} . h[k-n]\}$$

Algorithm:

The "deconvolution module" uses an iterative approach with a maximum number of loops set to 10^4. It stops at iteration n if results of iterations (n-1) and n are equal, with a "stop 10^{-3} threshold".



Principle of the deconvolution algorithm

User manual:

Input data	Output data (in the path of input data)
• Paths of DART waveform (lidar_convoluted_wave.txt) and impulse response (lidar_dart_desc.txt) • Maximum number of iterations • "Stop threshold"	• Deconvolved signal (Signal_Deconvolue_Calcule.txt) • Convolved signal of the deconvolved signal (Signal_Reconvolue_Calcule.txt)

☞ The module removes the 1st lines of the LIDAR_DART_desc.txt (impulse response) file in order to work with an array with 3 columns only.

Example:

Convoluted		Deconvolved	
Initial	Calculated	Initial	Calculated
36,250	36,250	145	145
89,000	89,000	66	66
69,250	69,249	0	0
16,500	16,500	0	0
49,750	49,751	199	199
99,500	99,499	0	0
49,750	49,751	0	0
……	……	…….	……
8,250	8,250	0	0
0,000	0,000	0	0
0,000	0,000	0	0
837,250	837,250	3349	3349
1674,500	1674,500	/	/
837,250	837,250	/	/

References:

- *Convolution and signal processing:*

Bellanger, M., Agrain, P., [2006], Traitement du signal : Théorie et pratique, Dunod, 8ème édition, ISBN-13 : 978-2100501625, 447p.

- *Deconvolution:*

Durrieu, S., [2007], Le lidar à retour d'onde complet. Nature de l'information enregistrée et perspectives d'utilisations ; Revue française de photogrammétrie et de télédétection, Atelier REGLIS : LiDAR et surfaces continentales : techniques, applications et perspectives, Montpellier, 7-8 décembre 2006, N° 186, pp. 15-35

Nordin, L., [2006], Analysis of waveform data from airborne laser scanner systems; Master thesis, Lulea University of technology, Computer Science and Electrical Engineering - Signal Processing, Sweden, N° 2006:336

Waniak, W., [1997], Image restoration by simple adaptative deconvolution; Astronomy & astrophysics supplement series, Ser. 124, pp. 197-203

The EM module (emDART.bat and emLVIS.bat)

It fits DART (module EMDART) and LVIS (module EMLVIS) waveforms with a sum of Gauss functions. EMLVIS can manage at once a set of waveforms and it gives correlation coefficients r^2 and RMSE (classes "CalculCoefficient" and "Resume") for "sum of Gauss functions - initial waveform"

Theory:

The analysis of the wave lidar uses a model (*e.g.*, log-normal, gauss) and a fit method.

Here, only the gauss model can be used to parameterize the wave Lidar. The latter is assumed to be a sum of k Gauss functions (μ_i, σ_i) with weights p_i:

$$\hat{y} = f(x) = \sum_{j=1}^{k} p_j * f_j(x) \quad \text{with} \quad f_j(x) \approx N(\mu_j, \sigma_j^2)$$

Here, the expectation-maximization (EM) algorithm is the fit method. It has the following steps:

a) *Determination of the number of peaks (i.e., Gauss functions)*

The method calculates the 1st derivative of the waveform and counts the number of sign changes from positive to negative, which gives the number of maxima. To exclude local maxima due to residual noise, waveforms other than DART are smoothed. A restrictive criterion is added if the number of peaks is > 10: the amplitude of any peak must be larger than 55% of the larger peak.

b) *Initialization of gauss functions*

Each Gauss mean is initialized with its peak position. The relative weight is initialized to the inverse of the number of gauss functions. For LVIS waveforms, gauss standard deviations σ are initialized to 8 bins (*i.e.*, 16 ns). Start with a maladjusted σ value can stop the program.

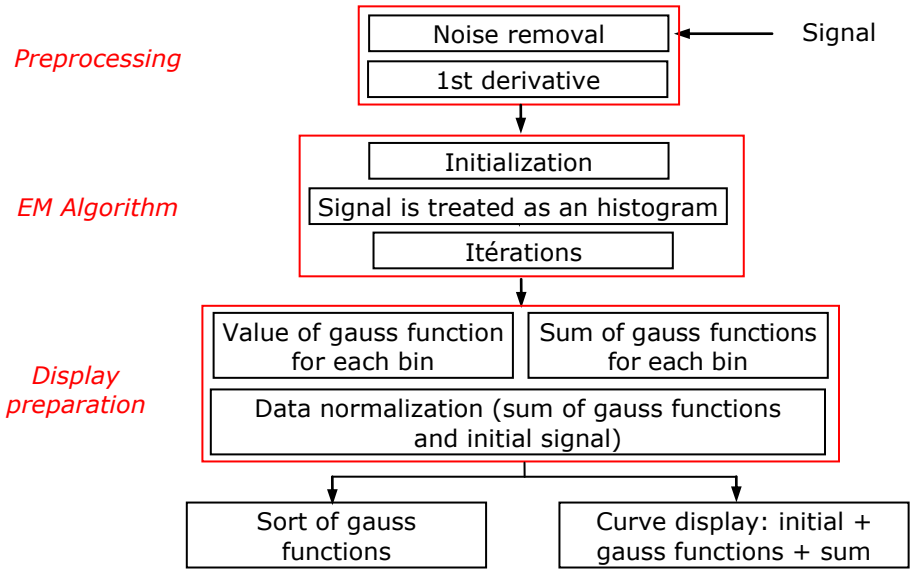
c) *EM algorithm*

The EM algorithm iterates until the sum of gauss functions fits well the initial waveform. It uses:

$$Q_{ij} = \frac{p_j \cdot f_j(x)}{\sum_{j=1}^{k} p_j \cdot f_j(x)} \qquad p_j = \frac{\sum_{i=1}^{n} N_i \cdot Q_{ij}}{\sum_{i=1}^{n} N_i} \qquad \mu_j = \frac{\sum_{i=1}^{n} N_i \cdot Q_{ij} \cdot i}{p_j \cdot \sum_{i=1}^{n} N_i} \qquad \sigma_j = \sqrt{\frac{\sum_{i=1}^{n} N_i \cdot Q_{ij} \cdot (i - \mu_j)^2}{p_j \cdot \sum_{i=1}^{n} N_i}}$$

where Q_{ij} is the probability that an element i belongs to gauss function j, n is the number of elements (*i.e.*, bins) of the waveform and Ni the intensity of the element i.

Algorithm:



Principle of the decomposition of a waveform as a set of gauss functions

Noise removal uses the noise standard deviation σ_{noise} for each complete waveform (Persson, 2005). It is not applied to DART waveforms because they are assumed to be noise free. Here, σ_{noise} is the dispersion of a distribution around the median: MAD (median absolute deviation).

$$\sigma_{noise} = 1.4826 \text{ . median}(|\text{wave-median(wave)}|)$$

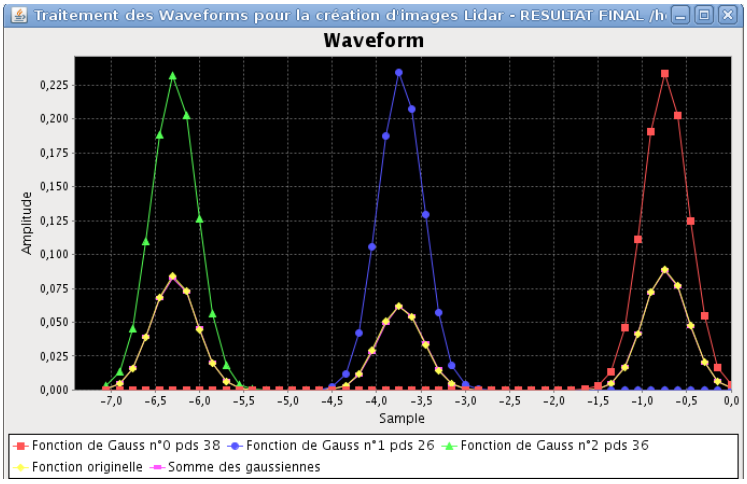
The median value is subtracted from waveform bin values, and then divided by σ_{noise}. Then, all values below a certain value k times this ratio are set to zero. Other values are "initial value - median(wave). The k value depends on the processed data. For LVIS data, a good choice is k = 2.

The EM algorithm (*cf.* "PseudoCode" in the Forum Developpez.com) works with a data set that is characterized by an histogram. Thus, a 1st step creates a set of data, the histogram of which is the waveform. The iterative process starts after the parameters (mean μ, standard deviation σ and relative weight p) of all Gauss functions are initialized. It stops if two successive iteration give similar results (± threshold) or if the number of iterations exceeds 10^4. Then, all gauss parameters are transformed in meters and gauss functions with {p < 5%, σ > 10} are removed. Remaining gauss functions are sorted according to their mean values. Very close gauss functions are treated as a single gauss function. Moreover, gauss functions beyond the ground peak are removed. This approach keeps the most relevant gauss functions.

```
0.35785953177206387   6.283878504673759   0.5157678395550706
0.2642140468232537    3.7281645569658055  0.509459075824108
0.3779264214046823    0.7373893805309735  0.5124392428395826
```

Example of Gauss parameters (p, μ, σ) given by the EM algorithm

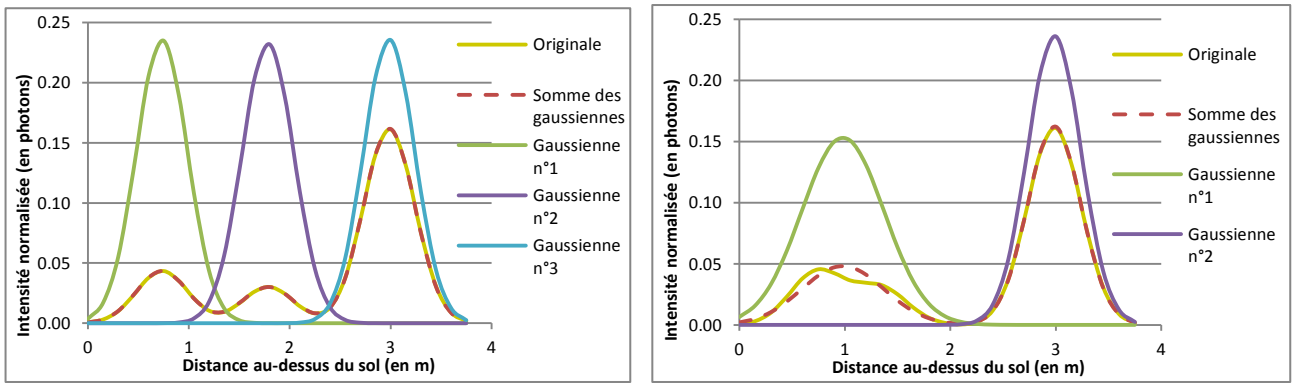
The last step is the preparation of the graphic display. The initial signal and the sum of gauss functions are normalized in order to ease their comparison. Residues can be viewed



User manual:

	Input data	Output data
DART	• "Path of DART waveform"	• Screen: - parameters of Gauss functions - curves: initial waveform, normalized Gauss functions and their sum • Only for LVIS: Text file (r^2 and RMSE for "Sum of Gauss functions *vs.* Initial waveform")
Both	• Number of iterations • "Stop threshold"	
LVIS	• "Path + generic prefix of waveform files" • Number of waveforms • Existing directory where to store correlations	

Note: emDART.bat does not work with the provided example. However, EMLVIS works fine.



Examples of waveforms with 3 and 2 maxima.

References:

Durrieu, S., [2007], Le lidar à retour d'onde complet. Nature de l'information enregistrée et perspectives d'utilisations ; Revue française de photogrammétrie et de télédétection, Atelier REGLIS : LiDAR et surfaces continentales : techniques, applications et perspectives, Montpellier, 7-8 décembre 2006, N° 186, pp. 15-35

Hofton, M.A., Minster, J.B., Blair, J.B., [2000], Decomposition of laser altimeter waveforms; IEEE Transactions on Geoscience and Remote Sensing, Vol. 38, n°4

Li, Q., [2008], Decomposition of airborne laser scanning waveform data based on EM algorithm ; The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Vol. XXXVII, Part B1, pp. 211-217

Nordin, L., [2006], Analysis of waveform data from airborne laser scanner systems; Master thesis, Lulea University of technology, Computer Science and Electrical Engineering - Signal Processing, Sweden, N° 2006:336

Persson, Å. , Söderman, U. , Töpel, J. , Ahlberg, S., [2005], Visualization and analysis of full-waveform airborne laser scanner data ; ISPRS WG III/3, III/4, V/3 Workshop "Laser scanning 2005", Enschede, the Netherlands, September 12-14, 2005, pp. 103-108

Ueberschlag Audrey,

Comparison of DART and LVIS waveforms (comparisonDart_LVIS.bat)

The aim is to compare LVIS waveforms that are measured and simulated by DART. This implies that the DART scene and the measured LVIS waveforms are well geographically referenced.

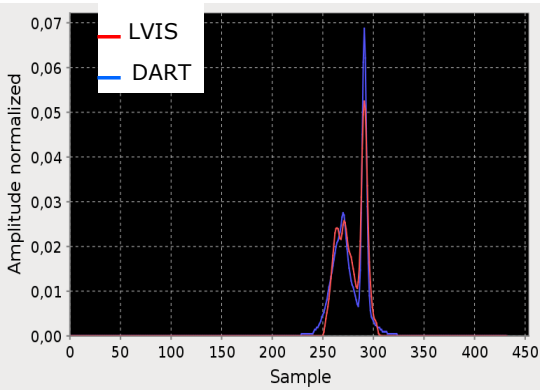
Algorithm:

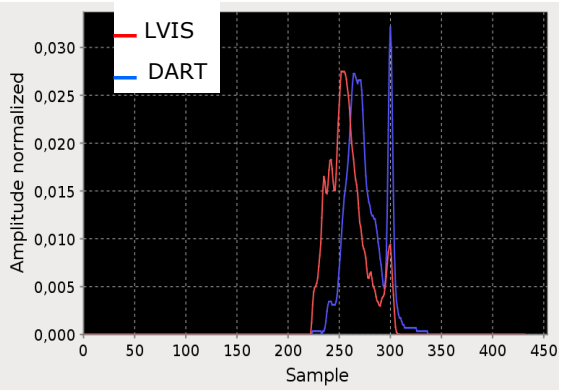
- Input: * Path to the directory that stores the LVIS and DART waveforms
  * LVIS file, and removal of the 7 first elements (geographical location and mean noise)
- Calculation of the noise standard deviation σ_{noise} as the dispersion of a distribution around the median: MAD (median absolute deviation): $\sigma_{noise}=1.4826.\text{median}(|\text{wave-median(wave)}|)$.
- The actual waveform is determined if $\sigma_{noise} < 10$.
- Noise removal: the median value is subtracted from waveform bin values, and then divided by σ_{noise}. Then, all values below k times this ratio are set to zero. Other values are "initial value - median(wave). The k value depends on the processed data. For LVIS data, a good choice is k = 2.
- Waveform processing: smoothing on 3 bins, 1st derivative, search of peaks (1st derivative goes to 0)
- Detection of DART peaks: 1st derivative, search of peaks (1st derivative goes to 0)
- Translation of DART waveform in order that its ground peak corresponds to the LVIS ground peak.
- Computation and registration of R2 and graphs (format svg) for each couple of waveforms.

User manual:

Input data	Output data
• "Path + generic prefix" of LVIS waveforms • "Path + generic prefix" of DART waveforms • Number of waveforms • Existing directory where to store screen displays (.svg format) R^2 coefficients	• Curves stored as svg files • R^2 coefficients : graphic window + Text file

Examples:





Référence:

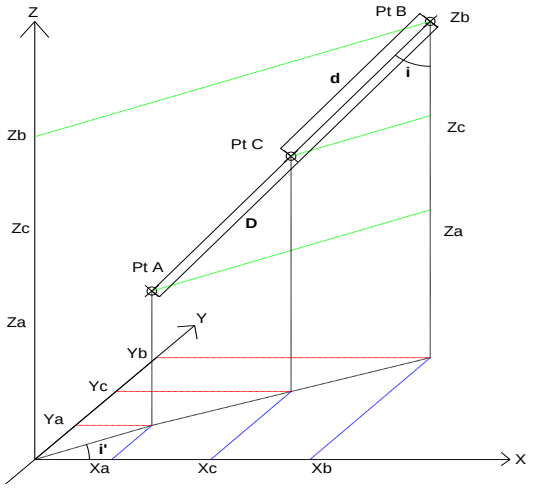
Persson, Å. , Söderman, U. , Töpel, J. , Ahlberg, S., [2005], Visualization and analysis of full-waveform airborne laser scanner data ; ISPRS WG III/3, III/4, V/3 Workshop "Laser scanning 2005", Enschede, the Netherlands, September 12-14, 2005, pp. 103-108

Altitude maps derived from Lidar waveforms (methodeNasa.bat)

Starting from a set of LVIS waveforms, this module determines for each waveform the peaks of the top altitude and ground. These peak coordinates are transformed into geographic coordinates (X, Y, Z).

Algorithm:

- Input: * Number of waveforms to process + View angles for each waveform
  * LVIS file, and removal of the 7 first elements (geographical location and mean noise)
- Calculation of the noise standard deviation σ_{noise} as the dispersion of a distribution around the median: MAD (median absolute deviation): $\sigma_{noise}=1.4826.\text{median}(|\text{wave-median(wave)}|)$
- The actual waveform is determined if $\sigma_{noise} < 10$. It starts when the raw waveform becomes > 15 σ_{noise}, and it stops when the raw waveform becomes < 15 σ_{noise}, and remains below this level.
- Waveform processing: smoothing on 3 bins, 1st derivative, search of peaks (1st derivative goes to 0)
- If the number of peaks differs from 0 (problem in the signal) and 1 (ground or top canopy peak?), the peak coordinates are stored in an array and transformed in the UTM coordinate system with:
- Top canopy and ground altitudes, canopy heights, with a B-spline interpolation (used by DART for DEM interpolation) of top canopy and ground points, after removal of points with zero value.



$$X_c = X_a + (D - d) \cdot \sin(i) \cdot \sin(i')$$

$$Y_c = Y_a + (D - d) \cdot \sin(i) \cdot \sin(i')$$

$$Z_c = Z_b - d \cdot \cos(i)$$

$$i' = \text{atan}\left(\frac{Y_b - Y_a}{X_b - X_a}\right)$$

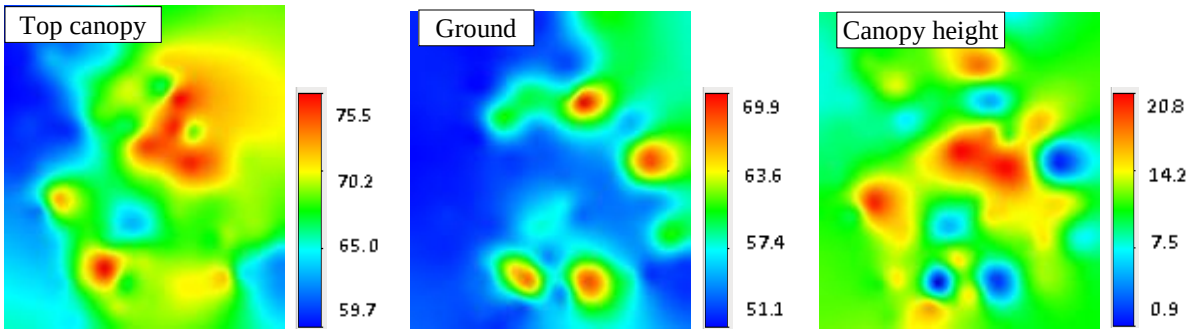
$$i' = \text{atan}\left(\frac{\sqrt{(X_b - X_a)^2 - (Y_b - Y_a)^2}\,Y_b - Y_a}{Z_b - Z_a}\right)$$

Coordinates of any point C between the first point B) and last (point A) bin. Here: ||BC|| =d and ||AB|| =D.

User manual:

Input data	Output data
• "Path + generic prefix" of LVIS files • Number of waveforms • Images dimensions: image (DX,DY), pixels (Δx, Δy) • Path: text file that stores the view zenith angle of each waveform	• 3 raw images: top canopy, ground and canopy height *(double real numbers)*

Example: altitudes images (1m resolution) of Howland forest (USA) derived from LVIS waveforms



LAI within the Lidar footprint in a DART scene (calcul_lai_theorique.bat)

This module computes the LAI that is within the Lidar footprint within a DART scene, for each simulation that belongs to a sequence of simulations.

Algorithm:

- Input: path of the DART root simulation
  ⇒ Phase.xml: DART scene size (line, column, layer) + lidar footprint (disk center (x,y) & radius)
  ⇒ LAI3D: 3D distribution of LAI (from ground to top scene)
- Computation of LAI vertical profile $LAI_{layer}(k)$ and $\sum_{i=0}^{i=k} LAI_{layer}(i)$, using LAI3D. Here, the height of each layer k is 1 DART cell

☞ The Lidar footprint defined in DART as a disk is actually assumed to be a square.

User manual:

Input data	Output data
• Path of the DART root simulation • Number of waveforms (*i.e.*, number of simulations in the sequence).	• Vertical profile of LAI_{cell} and cumulated LAI

Note: does not give correct values in the case of a sequence with variable Lidar positions

Inversion of waveforms (calcul_lai_couche_LVIS.bat)

Sequential inversion of waveforms for obtaining vegetation vertical profiles (LAI / bin. There are 2 modes of any waveform (with LVIS format) with an iterative analytical inversion.

- ***Theory: Lesfky inversion method***

This "classical" method, called "Canopy Height Profile Measurements (CHP)" (Lesfky et al., 1999; Harding et al., 2001), is provided here as reference. It quantifies the LAI vertical distribution of a vegetation canopy that is horizontally homogeneous with a constant leaf angle distribution:

$$LAI(h) = -\ln(1-cover(h)) \qquad \text{with } cover(h) = 1 - \frac{k \,.\, Signal_{ground}}{Signalcanopy(h) + k \,.\, Signal_{ground}}$$

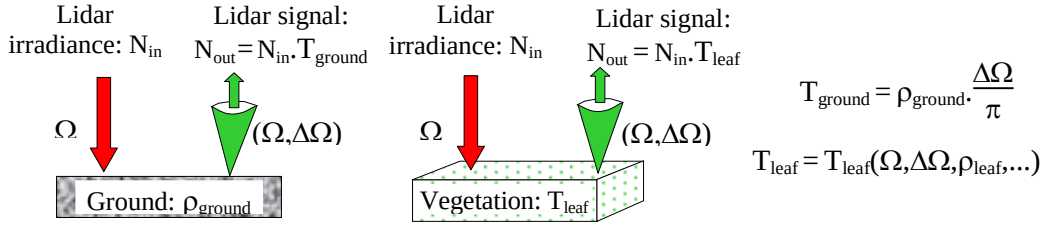
The coefficient k is assumed to represent the reflectance difference between vegetation and ground. In the near infrared, Lefsky *et al.* (1999) uses k=2. With LVIS and DART waveforms, this method was very incorrect for obtaining absolute LAI values. Relative LAI values were much more correct.

- ***Theory: Iterative analytical inversion***

Let an homogeneous turbid scene S (*i.e.*, 100% ground cover) made of k layers. Each layer i has: height Δz, leaf area volume density $u_{f,i}$ ($LAI_{layer} = u_{f,i}.\Delta z$), leaf cross section $G(\Omega_s)$ and phase function $T_{leaf}(\Omega_s,\Omega_v)$, with Ω_s the incident direction and Ω_v the scattering direction. With $N_{in}(\Omega_s)$ photons incident along Ω_s, then $N_{out,i}$ photons are scattered by the i^{th} layer and measured. Thus:

$$\frac{N_{out,i}(S)}{N_{in}(S)} = T_{leaf}(\Omega_s,\Omega_v).\exp\left[-\frac{G(\Omega_s)}{|\mu_s|}.\sum_{u=1}^{i-1} u_{f,u}.\frac{c.\Delta t}{2}\right].\left\{1-\exp\left[-\frac{G(\Omega_s)}{|\mu_s|}.u_{f,i}.\frac{c.\Delta t}{2}\right]\right\}$$

with $\mu_s = \cos\theta_s$, and θ_s the Lidar zenith angle, and
Δt the time needed by a photon for crossing back and forth a layer: $c.\Delta t = 2.\Delta z$



However, an actual scene is horizontally heterogeneous. Thus, a scene layer i is treated as η_i % of "holes" and $(1-\eta_i)$ % of turbid medium with leaf volume density $u_{f,ti} = u_{f,i}(z)/(1-\eta_i)$. The inversion procedure computes $u_{f,i}(z)$ for 2 types of distributions of holes: identical in all layers or random.

- *Random distribution of holes along z*

$$\frac{N_{out,1}(S)}{N_{in}(S)} = T_{leaf}(j,\Omega_s,\Omega_v).\{1-T_1(\Omega_s)\} \text{ with } T_1(\Omega_s)=(1-\eta_1).\exp\left[-\frac{G(\Omega_s)}{|\mu_s|}.u_{f,t1}.\frac{c.\Delta t}{2}\right]+\eta_1 \Rightarrow u_{f,t1} \text{ and } \mathcal{J}(H-\Delta z)$$

$$\frac{N_{out,2}(S)}{N_{in}(S)} = T_{leaf}(j,\Omega_s,\Omega_v).T_1(\Omega_s).\{1-T_2(\Omega_s)\} \text{ with } T_2(\Omega_s)=(1-\eta_2).\exp\left[-\frac{G(\Omega_s)}{|\mu_s|}.u_{f,t2}.\frac{c.\Delta t}{2}\right]+\eta_2 \Rightarrow u_{f,t2} \text{ and } \mathcal{J}(H-2.\Delta z)$$

$$\frac{N_{out,i}(S)}{N_{in}(S)} = T_{leaf}(j,\Omega_s,\Omega_v).\prod_{k=1}^{i-1} T_k(\Omega_s).\{1-T_i(\Omega_s)\} \text{ with } T_i(\Omega_s)=(1-\eta_i).\exp\left[-\frac{G(\Omega_s)}{|\mu_s|}.u_{f,ti}.\frac{c.\Delta t}{2}\right]+\eta_i \Rightarrow u_{f,ti}$$

$$\frac{N_{out,ground}(S)}{N_{in}(S)} = T_{ground}(j,\Omega_s,\Omega_v).\prod_{k=1}^{K} T_k(\Omega_s) \quad \text{with } LAI_{total} = \sum_{u=1}^{I} u_{f,u}.\frac{c.\Delta t}{2}$$

- *Constant distribution of holes along z*

$$\frac{N_{out,1}(S)}{N_{in}(S)} = T_{leaf}(j,\Omega_s,\Omega_v).(1-\eta).\{1-T_{1,turb}(\Omega_s)\} \text{ with } T_{t1}(\Omega_s)=\exp\left[-\frac{G(\Omega_s)}{|\mu_s|}.u_{f,t1}.\frac{c.\Delta t}{2}\right] \Rightarrow u_{f,t1} \text{ and } \mathcal{J}(H-\Delta z)$$

$$\frac{N_{out,2}(S)}{N_{in}(S)} = T_{leaf}(j,\Omega_s,\Omega_v).(1-\eta).T_{1,turb}(\Omega_s).\{1-T_{2,turb}(\Omega_s)\} \text{ with } T_{t2}(\Omega_s)=\exp\left[-\frac{G(\Omega_s)}{|\mu_s|}.u_{f,t2}.\frac{c.\Delta t}{2}\right] \Rightarrow u_{f,t2} \text{ and } \mathcal{J}(H-2.\Delta z)$$

$$\frac{N_{out,i}(S)}{N_{in}(S)} = T_{leaf}(j,\Omega_s,\Omega_v).(1-\eta).\prod_{k=1}^{i-1} T_{k,turb}(\Omega_s).\{1-T_{i,turb}(\Omega_s)\} \text{ with } T_{ti}(\Omega_s)=\exp\left[-\frac{G(\Omega_s)}{|\mu_s|}.u_{f,ti}.\frac{c.\Delta t}{2}\right] \Rightarrow u_{f,ti} \text{ and } \mathcal{J}(H-i.\Delta z)$$

$$\frac{N_{out,ground}(S)}{N_{in}(S)} = T_{ground}(j,\Omega_s,\Omega_v).(1-\eta).\prod_{k=1}^{K} T_{k,turb}(\Omega_s) \quad \text{with } LAI_{total} = \sum_{u=1}^{I} u_{f,u}.\frac{c.\Delta t}{2}$$

Algorithm:

- For DART waveforms:
  - Input: * Path "directory + generic name" that stores the simulations to be treated
    * Number of simulations (waveforms) to process
    * Path where to store results
    * Filename that stores the phase functions (liste_fonction_phase.txt)
  - Computation with DART file RadiativeBudget 3D: number of photons that are incident on the scene and that are scattered upwards by each scene layer.
  - Calculation with DART file LAI_3D.txt: vegetation cover (*i.e.*, 1 - fraction η of holes) and LAI of each scene layer. Vertical profiles of the LAI values and cumulative values are stored
  - Waveform inversion: $LAI_{layer}(k)$ & $\sum_{i=0}^{i=k} LAI_{layer}(i)$. Height of each layer k = 1 DART cell = 1 bin
- For LVIS waveforms:
  - Input: * Path to the directory that stores the LVIS waveforms
    * Vegetation phase function + Incident energy on top of the scene
    * LVIS file. The 7 first elements (location and mean noise) are automatically removed
  - Calculation of the noise standard deviation σ_{noise} as the dispersion of a distribution around the median: MAD (median absolute deviation): $\sigma_{noise} = 1.4826.\text{median}(|\text{wave-median(wave)}|)$
  - Noise removal: the median value is subtracted from waveform bin values, and then divided by σ_{noise}. Then, all values below k times this ratio are set to zero. Other values are "initial value - median(wave). The k value depends on the processed data. For LVIS, a good choice is k = 2.
  - The actual waveform is determined if $\sigma_{noise} < 10$. It starts when the raw waveform becomes > 15 σ_{noise}, and it stops when the raw waveform becomes < 15 σ_{noise}, and remains below this level.
  - Waveform processing: smoothing on 3 bins, 1st derivative, search of peaks (1st derivative goes to 0)
  - Input of vegetation cover as a text file.
  - Waveform inversion: vertical profile $LAI_{layer}(k)$ and $\sum_{i=0}^{i=k} LAI_{layer}(i)$. The height of each layer k = 1 bin.

User manual:

	Input data	Output data
DART	• Path of the DART simulation • Path + name of the phase function	• Vertical profile $LAI_{layer}(k)$ & $\sum_{i=0}^{i=k} LAI_{layer}(i)$ Height layer = 1 DART cell. 1 text file / waveform
LVIS & DART	• *Lesfky method*: coefficient k • Constant (0) or random (1) distribution of holes	
LVIS	• Path of LVIS waveforms • Vegetation cover: text file (1 value) / waveform • Number of LVIS waveforms • Number of incident photons • Directory of output results • Parameter F: random (1) or constant (≠1) distribution of holes in the canopy layers • Text file: values of the phase functions (Ω_s,Ω_s)	• Vertical profile $LAI_{layer}(k)$ & $\sum_{i=0}^{i=k} LAI_{layer}(i)$ Height of each layer k = 1 bin (1 text file / waveform)

Harding, D.J., Lefsky, M.A., Parker, G.G., Blair, J.B., 2001. Laser altimetry canopy height profiles methods and validation for closed-canopy, broadleaf forests. Remote Sensing of Environment, 76: 283-297

Lefsky, M.A., Cohen, W.B., Acker, S A., Parker, G.G., Spies, T.A., Harding, D., 1999. Lidar remote sensing of the canopy structure and biophysical properties of Douglas-Fir Western Hemlock Forests ; Remote Sensing of Environment, Elsevier Science Inc., Vol. 70, pp. 339-361

6 *Sequencer (to be up-dated)*

The Sequencer module is designed to run a sequence of N simulations. For that, it copies N times a "root simulation", and in each copy it affects specific values to a number of parameters that are specified by the user. This is useful for sensitivity studies and Look-Up based inversion procedures. The usual approach involves the following steps:

1) Creation of a "root" simulation. The values of all input parameters that are constant in the N simulations are set to the right values.

2) Creation of a "Sequencer.xml" file that specifies all possible values of all variable parameters. This file leads to the automatic creation of a "Launcher.xml" file that stores all the specified parameter combinations (*i.e.*, collection of all successive DART simulations to be run).

   ☞ The sequencer cannot modify parameters (*e.g.*, type of tree crown or type of atmosphere), which depending on their values are linked to a variable number of parameters.

   ☞ When set to 1, some parameters indicate the presence of objects (*e.g.*, trees, atmosphere) or processes. They can be set to "0" in the sequence, but they cannot be changed from "0" to "1.

3) The Sequencer module runs all DART simulations specified in the "Launcher.xml" file.

I. Input parameters for the DART.Sequencer module

The Sequencer module works with a "Sequencer.xml" file (*Figure 17*) that contains:

- *Name of the "root" simulation* (**baseSimulationName**),
- *Name of the sequence* (**sequenceName**),
- *Description of each variable parameter*:
  - The Group to which it belongs (**groupName**) and its n*ame* (**propertyName**).
  - Its mode of variation (**type**): *linear* (1st value, step, number of values) or *enumerate* (individual values)
- Options intended to decrease computation time:
  - Names of the processes (**Process_NameLaunched**) that the sequence will run.
  - Number of parallel threads (**numberParallelThreads**).

If a sequence has G groups with S_g steps in each group, the total number of simulations is $\prod_{g=1}^{G} S_g$.

There can be several parameters in the same group. Parameters of the same group vary together.

```
<DartFile version="1.0">
  <DartSequencerDescriptor baseSimulationName="V5\sequence" sequenceName="V5\sequence\sequence\seq_plot">
    <DartSequencerDescriptorEntries>
      <DartSequencerDescriptorGroup groupName="Lambertian_Reflectance">
        <DartSequencerDescriptorEntry args="0.0;0.5;3"
        propertyName="Coeff_diff.LambertianMultiFunctions.LambertianMulti.nodeMultiplicativeFactorForLUT.multiplicativeFactorForLUT.reflectanceFactor" type="linear"/>
      </DartSequencerDescriptorGroup>
      <DartSequencerDescriptorGroup groupName="LAI">
        <DartSequencerDescriptorEntry args="0;2;3" propertyName="Plots.Plot[0].PlotVegetationProperties.lai" type="linear"/>
      </DartSequencerDescriptorGroup>
    </DartSequencerDescriptorEntries>
    <DartSequencerPreferences dartLaunched="true" demGeneratorLaunched="false" directionLaunched="true" displayEnabled="true" hapkeLaunched="false" maketLaunched="true"
    numberParallelThreads="16" phaseLaunched="true" prospectLaunched="false" triangleFileProcessorLaunched="false" vegetationLaunched="false"/>
  </DartSequencerDescriptor>
</DartFile>
```

Figure 17: Example of "Sequencer.xml" file that defines 9 simulations.

Finally, all the simulations to be run are stored in a file called "Launcher.xml" file (*Figure 18*). This is created by the phase that is called "sequence preparation", using the *"Sequencer.xml*" file. In addition

to all necessary parameters, it gives also the status of each simulation: "TO_PROCESS: simulation to be run", "PROCESSING: running simulation", "PROCESSED: correctly terminated simulation", "ERROR: simulation terminated with an error", "ERROR_XXX: an error occurred in the **xxx** simulation.

```
−<DartFile version="1.0">
  −<DartSequence baseSimulationName="V5\sequence" sequenceName="V5\sequence\sequence\seq_plot">
    −<Simulations>
      −<Simulation name="V5\sequence\sequence\seq_plot_0" status="PROCESSED">
        −<Properties>
            <Property name="Coeff_diff.LambertianMultiFunctions.LambertianMulti.nodeMultiplicativeFactorForLUT.multiplicativeFactorForLUT.reflectanceFactor" value="0"/>
            <Property name="Plots.Plot[0].PlotVegetationProperties.lai" value="0"/>
          </Properties>
        </Simulation>
      −<Simulation name="V5\sequence\sequence\seq_plot_1" status="PROCESSED">
        −<Properties>
            <Property name="Coeff_diff.LambertianMultiFunctions.LambertianMulti.nodeMultiplicativeFactorForLUT.multiplicativeFactorForLUT.reflectanceFactor" value="0"/>
            <Property name="Plots.Plot[0].PlotVegetationProperties.lai" value="2"/>
          </Properties>
        </Simulation>
      −<Simulation name="V5\sequence\sequence\seq_plot_8" status="PROCESSED">
        −<Properties>
            <Property name="Coeff_diff.LambertianMultiFunctions.LambertianMulti.nodeMultiplicativeFactorForLUT.multiplicativeFactorForLUT.reflectanceFactor" value="1"/>
            <Property name="Plots.Plot[0].PlotVegetationProperties.lai" value="4"/>
          </Properties>
        </Simulation>
    </Simulations>
    <DartSequencerPreferences dartLaunched="true" demGeneratorLaunched="false" directionLaunched="true" displayEnabled="true" hapkeLaunched="false" maketLaunched="true"
    numberParallelThreads="16" phaseLaunched="true" prospectLaunched="false" triangleFileProcessorLaunched="false" vegetationLaunched="false"/>
  </DartSequence>
</DartFile>
```

Figure 18: Example of "Launcher.xml" file that defines 9 simulations.

II. Run of the Sequence module

The GUI (*Figure 19*) allows one to run the sequence module. It comprises 4 usual steps (*Figure 20*):

* *Create sequence*: the parameters that vary in the sequence are specified. Their characteristics are stored in an xml file, that is called *"Sequencer.xml"* hereafter.

* *Edit sequence*: edition of a previously created sequence of simulations (*i.e.*, *"Sequencer.xml"*).

* *Prepare sequence*: it creates a *"Launcher.xml"* file that stores all successive simulations and it creates the input directories of all simulations. It runs the "*Run sequence*" automatically if an option is set (*Figure 20*). Moreover, folders that were created in a previous run of the sequence can be automatically deleted by a new run of the sequence. This increases computer time and should be used only if the number of cases of a given sequence is decreased.

* *Run sequence*: runs the sequence of simulations that are listed in the *"Launcher.xml"* file.

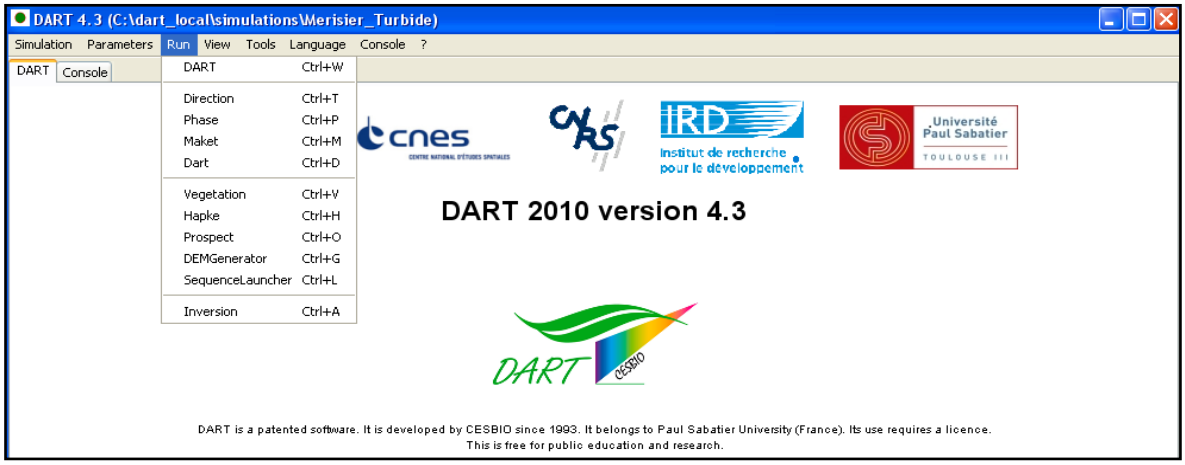


Figure 19: Access to the "SequenceLauncher" module. The root simulation is "Merisier_Turbide".

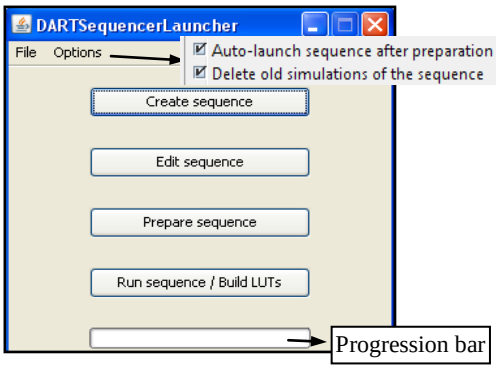


Figure 20: "SequenceLauncher" menu.

The sequence is automatically run after the step "Prepare sequence" if the option "Auto-launch sequence after preparation" is set.

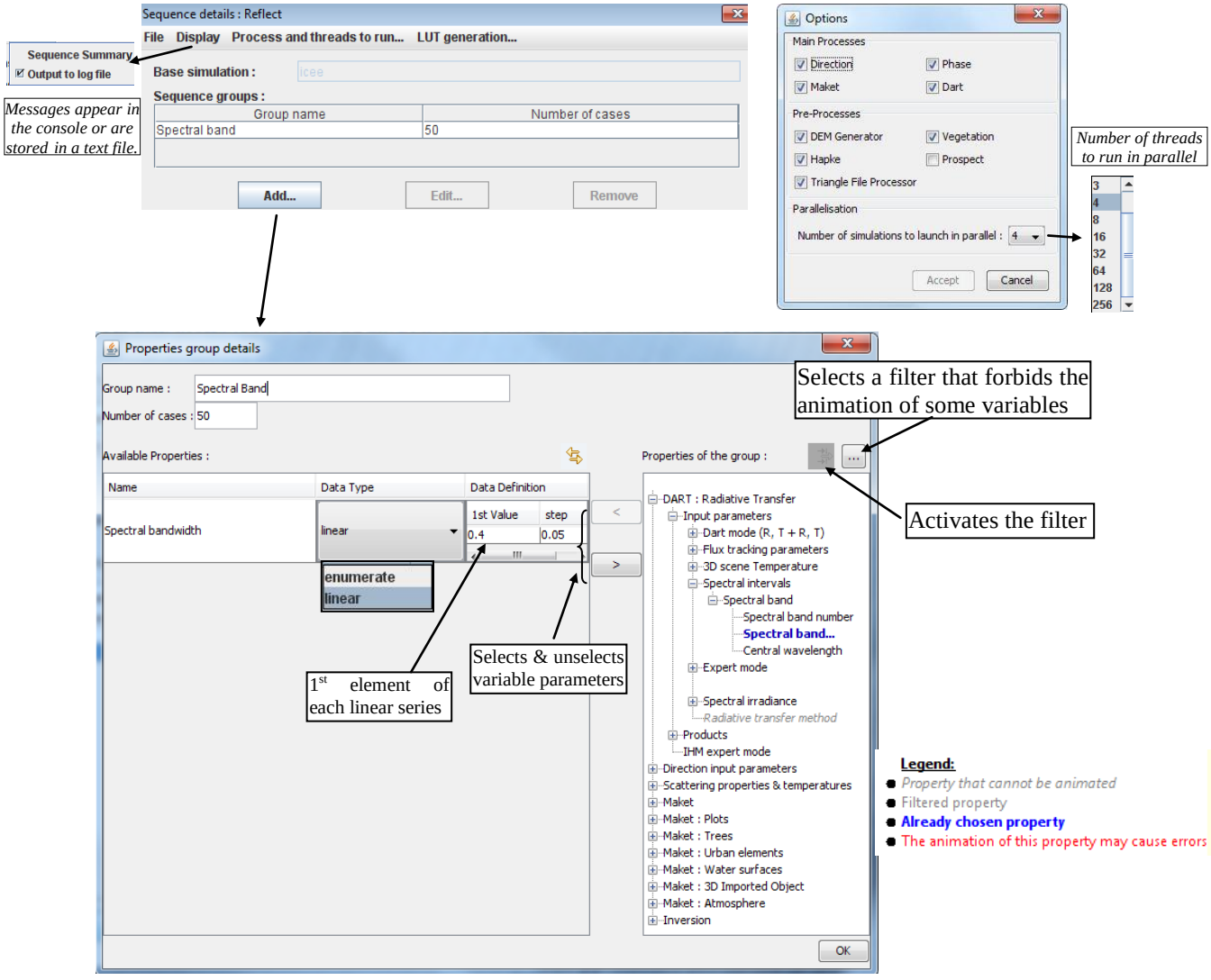


Figure 21: Creation / Edition of a sequence.

"Add": it adds groups where DART parameters vary. Here, the sequence has 1 group with 50 cases: parameter "Spectral bandwidth" takes 50 values. Thus, this sequence creates 50 simulations

"Options": - Displays/skips messages in the console.

- The user can select the processes to run in the sequence, and the number (1 to 256) of processes to run in parallel (this number must be 1 if the "Prospect" module is used).

It is possible to use a filter that allows one to display the parameters that cannot be animated:

```
# This is a comment before an empty line
# node after empty line
Maket.Scene.CellDimensions.x
# node with space
  Maket.Scene.CellDimensions.y
# set of nodes
Maket.*
```

Example of filter.

This is an ASCII file. With this filter, the node Maket.Scene.CellDimensions.x cannot be animated

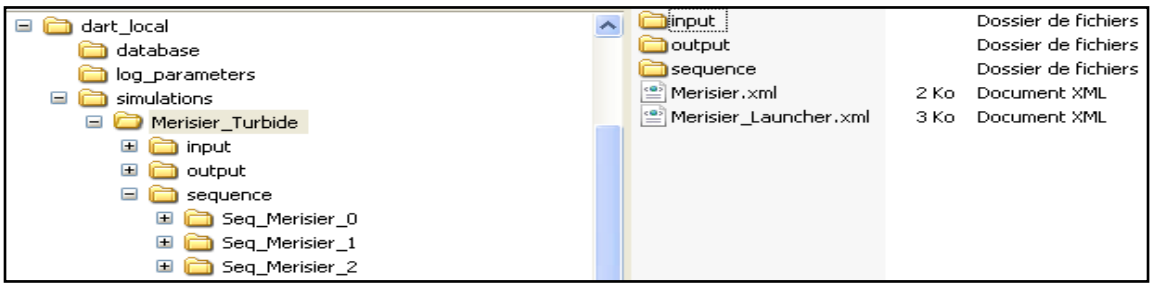


Figure 22: Architecture of a sequence. Here, this is the "Merisier_Turbide" sequence.

The *"Sequence.xml" and "Launcher.xml"* files are stored at the level of the directories "input" and "output" of the root simulation. Simulation names of the sequence are Seq_Merisier_x, where x is the simulation number.

7 *BroadBand*

The Broadband module computes broadband products (radiance $L_{\Delta\lambda}$, exitance $M_{\Delta\lambda}$, reflectance $R_{\Delta\lambda}$, brightness temperature $T_{B,\Delta\lambda},\ldots$), with n DART simulated narrow bands $[\lambda_i^{min}\ \lambda_i^{max}]$, each one with a central wavelength $\bar{\lambda}_i$. These bands can be weighted by the sensor spectral sensitivity s_i. Results are for 3 altitude levels *alt* (BOA, Sensor, TOA) if narrow bands were simulated with atmosphere RT.

☠ This module was designed when atmosphere and scene elements optical properties were computed with no weighting by sun TOA and / or BOA irradiance. So, results differ with this option.

I. Principle

A single DART simulation of a spectral band $\Delta\lambda$ is not an integral over the spectral band $\Delta\lambda$. The interaction of a ray with a scene element is modeled according to the mean optical property of that element over the band $\Delta\lambda$. So, the wavelength that corresponds to the mean spectral property of the interacting objects changes with these objects over $\Delta\lambda$. It is not necessarily the central wavelength $\bar{\lambda}$ of the band. This is especially true as sun irradiance and Planck's law are not spectrally linear. So a single spectral interval is usually inaccurate for simulating a wide spectral band. For that, the BroadBand module simulates a broad spectral interval as an integral of n DART narrow bands, possibly weighted by sensor spectral sensitivity $s(\lambda)$ that is in the range [0 1]. It works with all DART products (images, bidirectional reflectance and temperatures distributions,…), possibly for all DART iterations.

Most products other than temperature are computed as the numerical integral (*Figure 23*).

$$X_{\Delta\lambda,alt}=\frac{1}{2}.\{s_o.X_{\lambda,o,alt}.(\lambda_{max,o}-\lambda_{min,o})+s_{n-1}.X_{\lambda,n-1,alt}.(\lambda_{max,n-1}-\lambda_{min,n-1})+\sum_{i=o}^{n-2}[s_i.X_{\lambda,i,alt}+s_{i+1}.X_{\lambda,,i+1,alt}].(\bar{\lambda}_{i+1}-\bar{\lambda}_i)\}\ \text{ and } s_i=\frac{1}{\Delta\lambda_i}.\int_{\Delta\lambda_i}s(\lambda).d\lambda$$

with $X_{\lambda,i,alt}$ the i^{th} narrow band product (irradiance, radiance, reflectance) of the n DART spectral bands for the altitude (*i.e.*, BOA, Sensor or TOA) and $X_{\Delta\lambda}$ the broadband product. $X_{\lambda,i,alt}$ can be weighted by the spectral sensor sensitivity s_i for each narrow band $\Delta\lambda_i$. s_i values smaller than a threshold that is defined in the BroadBand GUI are assumed to be null.

If there is a single spectral interval (n=1): $X_{\Delta\lambda,alt}=s_o.X_{\lambda,o,alt}.(\lambda_{max,o}-\lambda_{min,o})$.

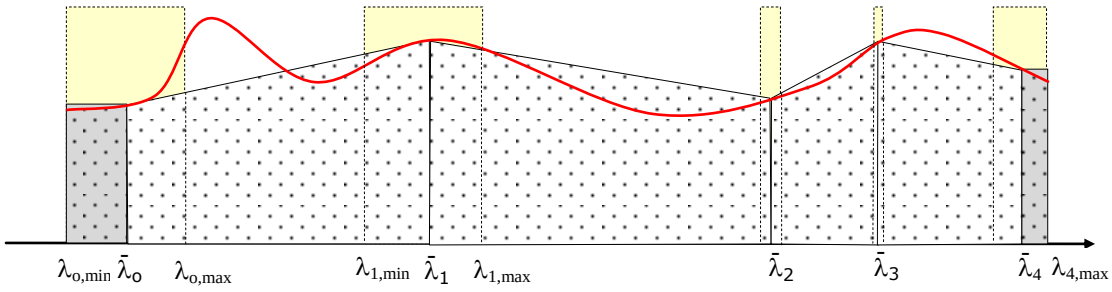


Figure 23: Computation of a broadband product with 5 narrow bands. Rectangle + trapezoid method.

I.1 Reflectance mode

Reflectance products (mean value, image pixel) **on Δλ:**

$$R_{\Delta\lambda,alt}=\frac{s_o.R_{\lambda,o,alt}.E_{\lambda,o,alt}.(\lambda_{max,o}-\lambda_{min,o})+s_{n-1}.R_{\lambda,n-1,alt}.E_{n-1,alt}.(\lambda_{max,n-1}-\lambda_{min,n-1})+\sum_{i=o}^{n-2}[s_i.R_{\lambda,i,alt}.E_{\lambda,i,alt}+s_{i+1}.R_{\lambda,,i+1,alt}.E_{\lambda,i+1,alt}].(\bar{\lambda}_{i+1}-\bar{\lambda}_i)}{E_{\lambda,o,alt}.(\lambda_{max,o}-\lambda_{min,o})+E_{n-1,alt}.(\lambda_{max,n-1}-\lambda_{min,n-1})+\sum_{i=o}^{n-2}[E_{\lambda,i,alt}+E_{\lambda,i+1,alt}].(\bar{\lambda}_{i+1}-\bar{\lambda}_i)}$$

with $R_{DART,i}$: DART reflectance at wavelength i, with $i \in [0\ \ n-1]$. We have: $R_{\Delta\lambda,alt}(\Omega) = R_o(\Omega)$ if n=1

$E_{i,alt}$ ($W.m^{-2}.\mu m^{-1}$): downward spectral irradiance (TOA irradiance database multiplied by $\cos\theta_i$).

Radiance on Δλ ($W.m^{-2}.sr^{-1}$):

$$L_{\Delta\lambda,alt} = \frac{s_o.R_{\lambda,o,alt}.E_{\lambda,o,alt}.(\lambda_{max,o} - \lambda_{min,o}) + s_{n-1}.R_{\lambda,n-1,alt}.E_{n-1,alt}.(\lambda_{max,n-1} - \lambda_{min,n-1}) + \sum_{i=o}^{n-2} [s_i.R_{\lambda,i,alt}.E_{\lambda,i,alt} + s_{i+1}.R_{\lambda,,i+1,alt}.E_{\lambda,i+1,alt}].(\bar{\lambda}_{i+1} - \bar{\lambda}_i)}{2.\pi}$$

for each DART discrete direction (Ω_j). If n=1: $L_{\Delta\lambda,alt}(\Omega) = L_o(\Omega).(\lambda_{max,o} - \lambda_{min,o})$.

Spectral radiance on Δλ ($W/m^2/sr/\mu m$): $\bar{L}_{\lambda,alt} = \frac{L_{\Delta\lambda,alt}}{\frac{(\lambda_{max,o} - \lambda_{min,o})}{2} + \frac{(\lambda_{max,n} - \lambda_{min,n})}{2} + \sum_{i=o}^{n-2} (\bar{\lambda}_{i+1} - \bar{\lambda}_i)} = \frac{L_{\Delta\lambda,alt}}{\lambda_{max,n-1} - \lambda_{min,o}}$ (3)

Exitance on Δλ ($W.m^{-2}$): $M_{\Delta\lambda,alt} = \sum_{j=1}^{J} L_{\Delta\lambda,alt}(\Omega_j).\cos(\theta_j).\Delta\Omega_j$

where J is the number of upward directions {direction $\Omega_j(\theta_j,\phi_j)$, solid angle ($\Delta\Omega_j$)}

Spectral exitance on Δλ ($W.m^{-2}.\mu m^{-1}$): $M_{\lambda,alt} = \frac{M_{\Delta\lambda,alt}}{\lambda_{max,n-1} - \lambda_{min,o}}$

Irradiance on Δλ ($W.m^{-2}$): $E_{\Delta\lambda,alt} = \frac{1}{2}.\{E_{\lambda,o,alt}.(\lambda_{max,o} - \lambda_{min,o}) + E_{n-1,alt}.(\lambda_{max,n-1} - \lambda_{min,n-1}) + \sum_{i=o}^{n-2} [E_{\lambda,i,alt} + E_{\lambda,i+1,alt}].(\bar{\lambda}_{i+1} - \bar{\lambda}_i)\}$

Spectral irradiance on Δλ ($W.m^{-2}.\mu m^{-1}$): $E_{\lambda,alt} = \frac{E_{\Delta\lambda,alt}}{\lambda_{max,n-1} - \lambda_{min,o}}$

I.2 Thermal mode (modes T and T+R)

In a 1st step, Planck's law $L_B(\lambda,T)$ is used to convert brightness temperature products (images or mean values) $T_{DART,i,alt}(\Omega)$ for direction (Ω) and narrow band i into spectral radiance ($W.m^{-2}.sr^{-1}.m^{-1}$):

$$L_{\lambda,i,alt}(\Omega) = L_B(\lambda_{eq,i,alt}, T_{DART,i,alt}(\Omega)) = \frac{2h.c^2}{\lambda_{eq,i,alt}^5}.\frac{1}{\exp[\frac{h.c}{k.T_{DART,i,alt}(\Omega).\lambda_{eq,i,alt}}] - 1}$$

where the equivalent wavelength $\lambda_{eq,i,alt}$ for narrow band *i* is computed with $L_B(\lambda_{eq,i},T_{mean}) = \frac{1}{\Delta\lambda_i}.\int_{\Delta\lambda_i} L_B(\lambda,T_{mean}).d\lambda$

h = Planck constant ($6.6218\ 10^{-34}$ J.s), k = Boltzmann constant ($1.38066\ 10^{-23}$ $J.K^{-1}$), c = light speed in free space ($2.997925\ 10^8$ $m.s^{-1}$) and T_{mean} = scene elements mean thermodynamic temperature.

Radiance on Δλ: $L_{\Delta\lambda,alt} = \frac{1}{2}.\{s_o.L_{\lambda,o,alt}.(\lambda_{max,o} - \lambda_{min,o}) + s_n.L_{\lambda,n,alt}.(\lambda_{max,n} - \lambda_{min,n}) + \sum_{i=o}^{n-1} [s_i.L_{\lambda,i,alt} + s_{i+1}.L_{\lambda,,i+1,alt}].(\bar{\lambda}_{i+1} - \bar{\lambda}_i)\}$

Spectral radiance on Δλ: $\bar{L}_{\lambda,alt} = \frac{L_{\Delta\lambda,alt}}{\lambda_{max,n-1} - \lambda_{min,o}}$

Brightness temperature on Δλ: $T_{\Delta\lambda,alt}(\Omega) = \frac{h.c}{k.\lambda_{eq,\Delta\lambda}}.\frac{1}{\ln[1 + \frac{2h.c^2}{\bar{L}_{\lambda,alt}.\lambda_{eq,\Delta\lambda}^5}]}$ with $L_B(\lambda_{eq,\Delta\lambda},T_{mean}) = \frac{1}{\Delta\lambda}.\int_{\Delta\lambda} L_B(\lambda,T_{mean}).d\lambda$

Note: $\lambda_{eq,\Delta\lambda}$ is not very coherent if the n spectral intervals $\Delta\lambda_i$ are not adjacent.

Exitance on Δλ ($W.m^{-2}$): $M_{\Delta\lambda,alt} = \sum_{j=1}^{J} L_{\Delta\lambda,alt}(\Omega_j).\cos(\theta_j).\Delta\Omega_j$

where J is the number of upward directions {direction $\Omega_j(\theta_j,\phi_j)$, solid angle ($\Delta\Omega_j$)}

Spectral exitance on Δλ ($W.m^{-2}.\mu m^{-1}$): $M_{\lambda,alt} = \frac{M_{\Delta\lambda,alt}}{\lambda_{max,n-1} - \lambda_{min,o}}$

Irradiance on Δλ ($W.m^{-2}$): $E_{\Delta\lambda,alt} = \frac{1}{2}.\{E_{\lambda,o,alt}.(\lambda_{max,o} - \lambda_{min,o}) + E_{n-1,alt}.(\lambda_{max,n-1} - \lambda_{min,n-1}) + \sum_{i=o}^{n-2} [E_{\lambda,i,alt} + E_{\lambda,i+1,alt}].(\bar{\lambda}_{i+1} - \bar{\lambda}_i)\}$

with E_λ ($W.m^{-2}.\mu m^{-1}$) = sky thermal irradiance + direct sun irradiance + sky scattered sun irradiance.

Spectral irradiance on Δλ ($W.m^{-2}.\mu m^{-1}$): $E_{\lambda,alt} = \frac{E_{\Delta\lambda,alt}}{\lambda_{max,n-1} - \lambda_{min,o}}$

I.3 Radiative budget products

DART computes 5 scene radiative budget terms (fIR↓, fIR↑, fAR, fSR, fIntR) per iteration and stores them in 2 text files: RadiativeBudget_3D and RadiativeBudget_profile (*cf.*, DART_User_Manual). Here, they are noted $fx_{\lambda,i}$, for each narrow band i. The BroadBand module uses them for computing the associated 5 scene broadband radiative budget terms $fx_{\Delta\lambda}$, with the same units as the $fx_{\lambda,i}$ terms:

- *Mode R*

  - unit %: $fx_{\Delta\lambda} = \frac{fx_{\lambda,o}.E_{\lambda,o,top}.(\lambda_{max,o} - \lambda_{min,o}) + fx_{\lambda,n-1}.E_{\lambda,n-1,top}.(\lambda_{max,n-1} - \lambda_{min,n-1}) + \sum_{i=o}^{n-2} [fx_i.E_{\lambda,i,top} + fx_{\lambda,i+1}.E_{\lambda,i+1,top}].(\bar{\lambda}_{i+1} - \bar{\lambda}_i)}{E_{\lambda,o,top}.(\lambda_{max,o} - \lambda_{min,o}) + E_{\lambda,n-1,top}.(\lambda_{max,n-1} - \lambda_{min,n-1}) + \sum_{i=o}^{n-2} [E_{\lambda,i,top} + E_{\lambda,i+1,top}].(\bar{\lambda}_{i+1} - \bar{\lambda}_i)}$

  - Unit $W/m^2/\mu m$: $fx_{\Delta\lambda} = \frac{1}{2}.\frac{fx_{\lambda,o,top}.(\lambda_{max,o} - \lambda_{min,o}) + fx_{\lambda,n-1,top}.(\lambda_{max,n-1} - \lambda_{min,n-1}) + \sum_{i=o}^{n-2} [fx_{\lambda,i,top} + fx_{\lambda,i+1,top}].(\bar{\lambda}_{i+1} - \bar{\lambda}_i)}{\lambda_{max,n-1} - \lambda_{min,o}}$

  - Unit W/m^2: $fx_{\Delta\lambda} = \frac{1}{2}.\{fx_{\lambda,o,top}.(\lambda_{max,o} - \lambda_{min,o}) + fx_{\lambda,n-1,top}.(\lambda_{max,n-1} - \lambda_{min,n-1}) + \sum_{i=o}^{n-2} [fx_{\lambda,i,top} + fx_{\lambda,i+1,top}].(\bar{\lambda}_{i+1} - \bar{\lambda}_i)\}$

- *Mode T*: same expressions as for mode R, except that there is no unit %.

II. Steps

- *Creation of the DART simulations*

  The set of narrow spectral bands can be created in 2 different ways: with or without sequencer. In both cases, the spectral intervals can be contiguous or not and must be sorted in ascending order.

  <u>*With sequencer:*</u> N monospectral simulations, with 'Central wavelength' and 'Bandwidth' the only variable parameters.

  <u>*Without sequencer:*</u> The simulation is realized with N spectral bands.

- *Format of the file that contains the spectral response function*

  There is no restriction for the file name except that the "." must be only for the extension (*e.g.*: band01.txt or ETMpan.rsp ...). Figure 2.a shows that it has 2 columns separated by a single space. The 1st column is the central wavelength in μm, and the 2nd column is the gain (spectral sensitivity).

  To avoid inaccurate interpolation of sensitivity s(λ) for DART narrow bands, make sure that:

  - 1st line: 0.00 0.00.
  - 2nd line: wavelength slightly smaller than the 1st wavelength with a non zero gain.
  - Line after the last line with a non zero gain: gain equal to 0 and a wavelength close to the wavelength of the last line with a non zero gain.
  - Last line: a gain equal to 0 and a wavelength large enough to include all spectral intervals calculated in the DART simulations (e.g., 3.00 for reflectance mode).

  The file that contains the sensor spectral response can be located anywhere. Examples of such files are in directory SensorSpectralFunction, in the global database directory (See Fig. 2.b).

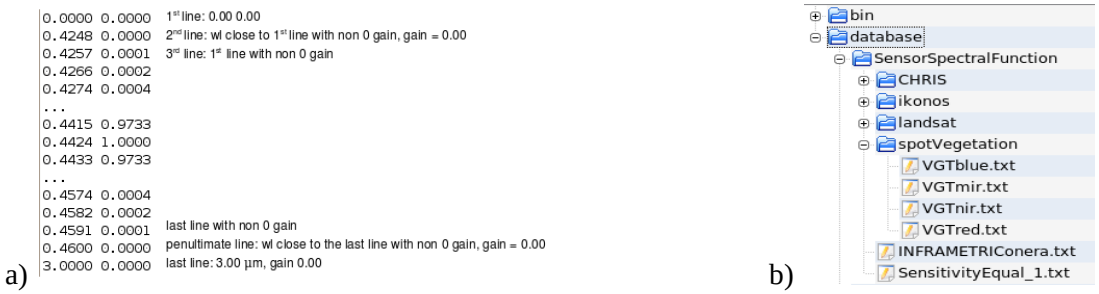


Figure 24: (a) Format of the sensor file: central wavelength (μm) gain. (b) Tree view of examples of sensor spectral response text files provided with DART software.

- *Run BroadBand (cf DART_User_Manual, Chapter "Tools menu").*

BroadBand GUI is run from the DART GUI. 3 parameters must be set:

- Sequence name or multispectral simulation name. At least 1 bandwidth must be non null. Moreover, all DART narrow bands must be for the same DART mode: (R), (T) or (R+T).
- Sensor filter: Choose the text file that contains the sensor spectral response. Its values must belong to [0 1]. Maximal spectral sensitivity can be lower than 1.
- Sensitivity threshold: sensitivity is set to zero if it is less than the specified threshold.

BroadBand processes all products of the n narrow bands (images and/or mean values), possibly at different altitudes if simulations were run with atmosphere radiative transfer.

For BRF and Tapp, the number of iterations is the maximal iteration present in the narrow bands. If this number is smaller for a given narrow band, the maximal iteration of this band is used.

"BroadBand" directory is at the same level as the "output" directory of the root simulation. It stores all broadband products (*Figure 25*). The menu "View \ BroadBand" displays most of them.

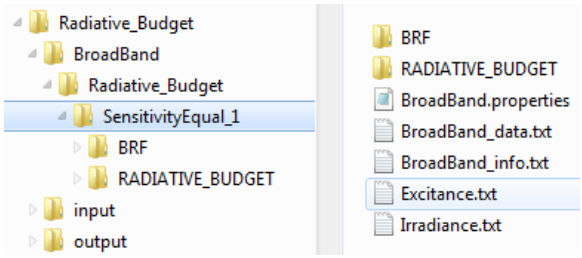


Figure 25: Broadband architecture. The file Broadband_info.txt informs on the format of the BroadBand_dat.txt, Exitance.txt and Irradiance.txt files.

8 *Image Inversion (to be totally reformulated)*

I. Functionality

The Inversion module inverts a set of multi-spectral and/or multi-angle satellite images (format "byte", "float" or "double"), possibly with a mask for limiting the zone to process. It relies on a parametric model that is defined by a set of 6 or 8 coefficients for representing the DART reflectance for each set of DART input parameter values (*e.g.*, LAI_5, $\rho_{sol,7}$, etc.). If satellite data to invert are nadir data, the number of coefficients is automatically set to 1. The parametric model with 6 and 8 parameters are:

$$R(\theta_v, \Delta\varphi_{sv}) = \left(c_1 + c_2\theta_v \cos(\Delta\varphi_{sv}) + c_3\theta_v^2 \cos^2(\Delta\varphi_{sv}) + c_4\theta_v^2 \sin^2(\Delta\varphi_{sv})\right)\times\left(1 + c_5 \exp\left(\frac{-\psi_{sv}^{\,2}}{c_6^2}\right)\right)$$

$$R(\theta_v, \Delta\varphi_{sv}) = \left(c_1 + c_2\theta_v \cos(\Delta\varphi_{sv}) + c_3\theta_v^2 \cos^2(\Delta\varphi_{sv}) + c_4\theta_v^2 \sin^2(\Delta\varphi_{sv}) + c_5\theta_v^4 \sin^4(\Delta\varphi_{sv}) + c_6\theta_v^6 \sin^6(\Delta\varphi_{sv})\right)\times\left(1 + c_7 \exp\left(\frac{-\psi_{sv}^{\,2}}{c_8^2}\right)\right)$$

The sets of coefficients are stored in a LUT of coefficients that is pre-computed with a LUT of reflectance that stores the reflectance values that are computed by a sequence of simulations. These reflectance values can be BOA, Sensor or TOA. Any parameter* can be inverted. For example:

- Any lambertian reflectance,
- Parameters of the Prospect model: - Structure index N,
  - Chlorophyll concentration Cab ($\mu g.cm^{-2}$),
  - Equivalent water depth Cw ($g.cm^{-2}$),
  - Dry matter content Cm ($g.cm^{-2}$)
- The scene LAI

Note: the possibility to invert secondary parameters such as the tree ground cover and the scene LAI in presence of plots is being finalized.

The information that is necessary for the inversion is stored in the "inversion.xml" file (*Figure 26*):

- *Set of images to invert*: numbers of columns and rows, code (double, float, byte) and the possible use of a mask for inverting only a spatial region of the image.
- *For each image*: name, multiplicative factor, spectral index, sun and view zenith / azimuth angles.
- *Inversion process*: type of error, interpolation method, LUT.xml file name, number of simplexes, output inversion file name (useless!!!), mode (inversion or robustness test), stop threshold.
- *Fixed parameter that varies in the LUT*. It can be defined by: * a spectra,
  * a value per spectral band,
  * same value for all bands.
- *Free parameter*. It can be: * free for each spectral band (*i.e.*, N free parameters for N spectral bands),
  * 1 free parameter (*i.e.*, 1 free parameter for N spectral bands),
  * 2 free parameters (*i.e.*, 2 free parameter for N spectral bands),
  * 3 free parameters (*i.e.*, 3 free parameter for N spectral bands).

Necessary conditions for the DART inversion procedure:

- Each free parameter must be within the range of its values used for the building up the LUT.

* The list of parameters that can be inverted can be easily changed through the modification of the GUI.

- In the LUT, parameter values must increase, without identical values
- 2 free parameters cannot be in the same Group (there must be as many Groups as free parameters).

```
−<DartFile version="4.3.6">
  −<DartInversion runInversion="1">
    −<InversionImages columnNumber="10" dataKind="2" isMask="0" rowNumber="10">
      −<ImageProperties ImageNumber="0" dataFileName="bleu.mp#" multiplicativeFactor="1" spectralBandKey="0">
          <SunViewAngleProperties sunAzimuthalAngle="45" sunZenithalAngle="150" viewAzimuthalAngle="0" viewZenithalAngle="0"/>
        </ImageProperties>
      −<ImageProperties ImageNumber="1" dataFileName="vert.mp#" multiplicativeFactor="1" spectralBandKey="1">
          <SunViewAngleProperties sunAzimuthalAngle="45" sunZenithalAngle="150" viewAzimuthalAngle="0" viewZenithalAngle="0"/>
        </ImageProperties>
      −<ImageProperties ImageNumber="2" dataFileName="rouge.mp#" multiplicativeFactor="1" spectralBandKey="2">
          <SunViewAngleProperties sunAzimuthalAngle="45" sunZenithalAngle="150" viewAzimuthalAngle="0" viewZenithalAngle="0"/>
        </ImageProperties>
      −<ImageProperties ImageNumber="3" dataFileName="PIR.mp#" multiplicativeFactor="1" spectralBandKey="3">
          <SunViewAngleProperties sunAzimuthalAngle="45" sunZenithalAngle="150" viewAzimuthalAngle="0" viewZenithalAngle="0"/>
        </ImageProperties>
      </InversionImages>
    −<Inversion errorKind="0" interpolationMethod="1" lutXMLFileName="seq_LAI_lambda_lut.xml" maxSimplexNumber="5" outputInversionFileName="inversion.out"
     processMode="1" stopThresholdRelativeError="0.01">
      −<InversionParameters>
          <freeParameters ParameterType="0" indexLUTEquivalent="0"/>
        </InversionParameters>
      </Inversion>
    </DartInversion>
  </DartFile>
```

Figure 26: Inversion.xml file.

As mentioned above, the inversion of TOA reflectance values is possible. For that, one creates a sequence with the atmosphere. This gives BOA and TOA reflectance values. Then, one creates a LUT of coefficients associated to the TOA reflectance values. The suffix of its name is "TOA_lut_coefficient". Then, the LUT.xml file must be manually edited (it will be automatic soon): in the section <DataFile>, the line that contains the name of LUT of coefficients associated to TOA reflectance must be moved to the second line (cf. *Figure 30*), because the name of the LUT of coefficients that is used by the inversion is necessarily in the second line of the section <DataFile>.

a)

```
−<DartFile>
  −<DartLUT simulationNumber="20" sequenceName="TOSCA_Foto\Inversion_FOTO\sequence\seq_LAI_lambda" dataFormat="TEXT"
   baseSimulationName="TOSCA_Foto\Inversion_FOTO" modelBRDFNumberCoefficient="6">
    −<DataFile>
        <Property name="seq_LAI_lambda_lut_reflectance.txt"/>
        <Property name="seq_LAI_lambda_lut_coefficient.txt"/>
        <Property name="seq_LAI_lambda_TOA_lut_reflectance.txt"/>
        <Property name="seq_LAI_lambda_TOA_lut_coefficient.txt"/>
      </DataFile>
    −<SpectralIntervals>
        <SpectralIntervalsProperties spectralBandKey="0" lambdaMin="0.39" lambdaMax="0.41"/>
        <SpectralIntervalsProperties spectralBandKey="1" lambdaMin="0.54" lambdaMax="0.56"/>
        <SpectralIntervalsProperties spectralBandKey="2" lambdaMin="0.69" lambdaMax="0.71"/>
        <SpectralIntervalsProperties spectralBandKey="3" lambdaMin="0.84" lambdaMax="0.86"/>
      </SpectralIntervals>
```

b)

```
−<DartFile>
  −<DartLUT simulationNumber="20" sequenceName="TOSCA_Foto\Inversion_FOTO\sequence\seq_LAI_lambda" dataFormat="TEXT"
   baseSimulationName="TOSCA_Foto\Inversion_FOTO" modelBRDFNumberCoefficient="6">
    −<DataFile>
        <Property name="seq_LAI_lambda_TOA_lut_reflectance.txt"/>
        <Property name="seq_LAI_lambda_TOA_lut_coefficient.txt"/>
        <Property name="seq_LAI_lambda_lut_reflectance.txt"/>
        <Property name="seq_LAI_lambda_lut_coefficient.txt"/>
      </DataFile>
    −<SpectralIntervals>
        <SpectralIntervalsProperties spectralBandKey="0" lambdaMin="0.39" lambdaMax="0.41"/>
        <SpectralIntervalsProperties spectralBandKey="1" lambdaMin="0.54" lambdaMax="0.56"/>
        <SpectralIntervalsProperties spectralBandKey="2" lambdaMin="0.69" lambdaMax="0.71"/>
        <SpectralIntervalsProperties spectralBandKey="3" lambdaMin="0.84" lambdaMax="0.86"/>
      </SpectralIntervals>
```

Figure 27: LUT.xml file before (a) and after modification (b).

Figure 28 shows the general structure of processes that are used for creating the LUT of reflectance for inverting satellite images.

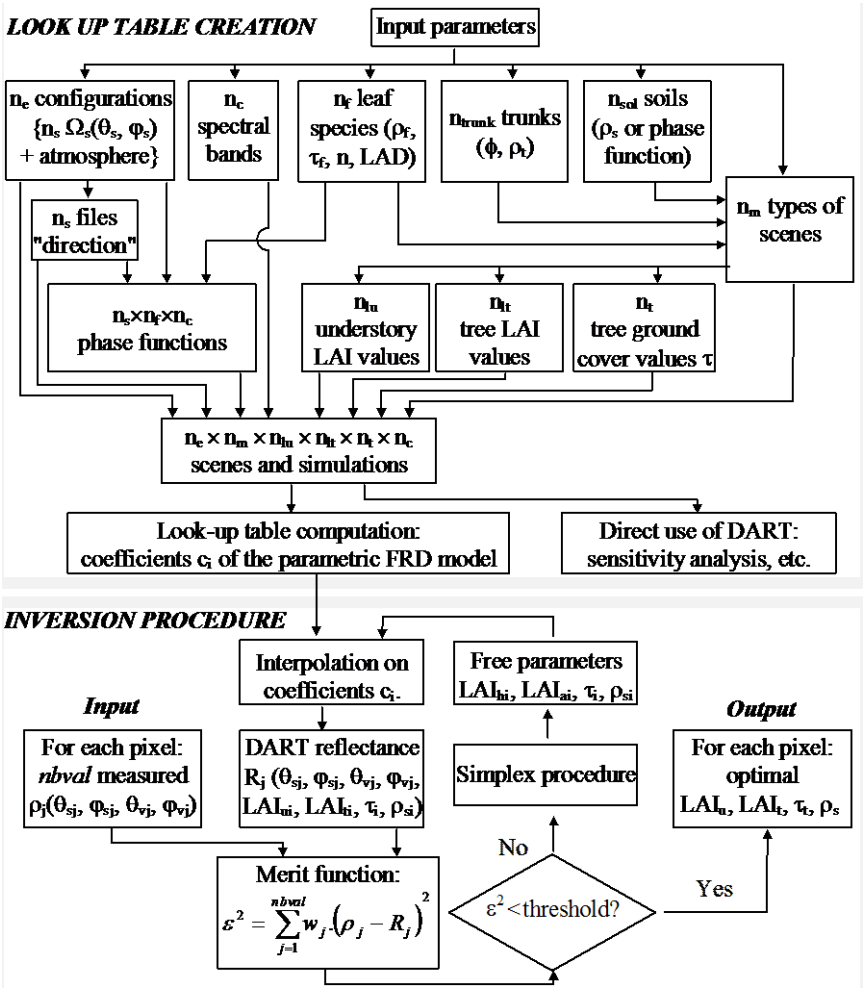


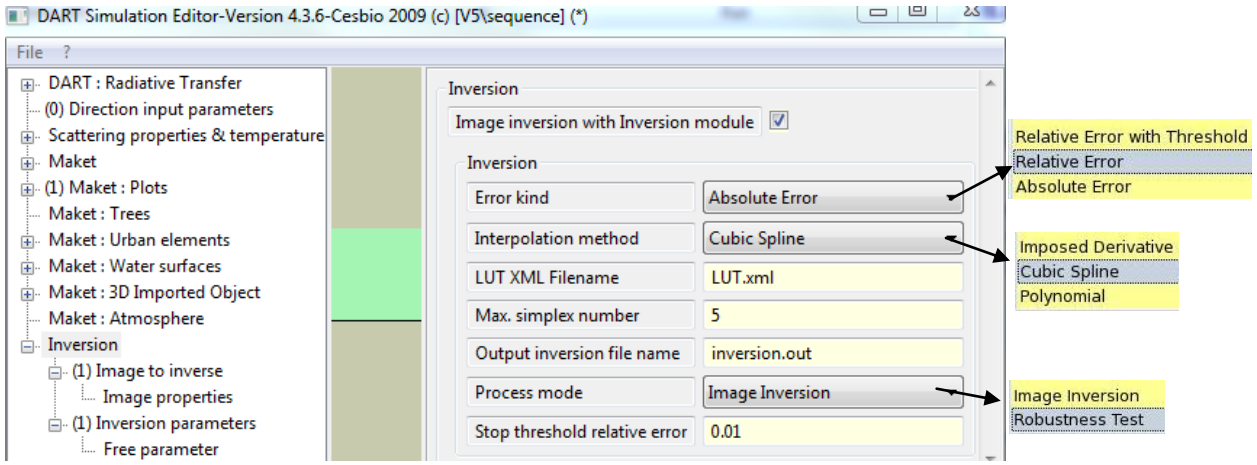
Figure 28: General structure of the look-up (top) and inversion (bottom) procedures.

II. Specification of the parameters in the GUI

- ***General characteristics of the inversion procedure*** (*Figure 29*)

  - type of error: relative, absolute or relative with threshold,
  - interpolation method: imposed derivative, cubic spline or polynomial,
  - number of simplexes to use in the inversion procedure,
  - the task to complete: inversion procedure or robustness test.

a)



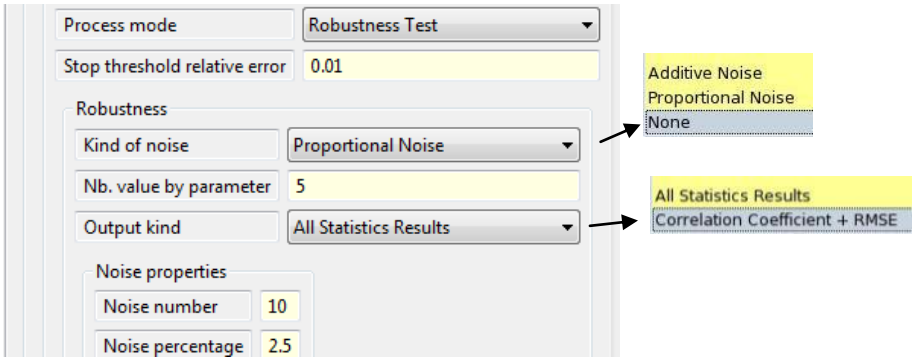


Figure 29: Characteristics of the inversion and test robustness methods.

The images that must be inverted must be "manually" copied in the "Input" directory of the reference simulation that stores the LUT used for the inversion.

• ***Image characteristics*** (*Figure 32*.a, b)

* *Common characteristics to all images*

- Numbers of columns and rows
- Type of data: byte, float or double,
- Presence of a mask (and its file name) for processing the inversion on a part of the image(s),

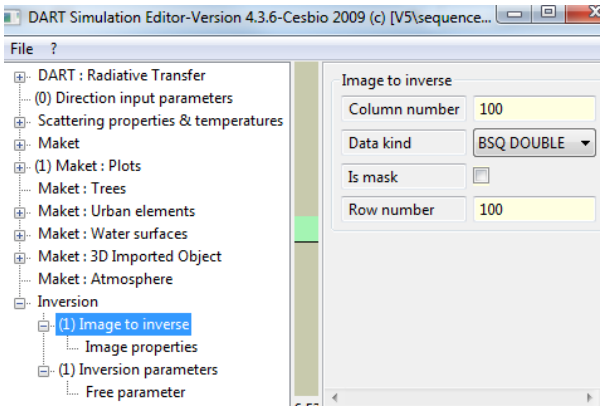
* *Characteristics of each image*

- Multiplicative factor. It allows one to take into the fact that images to invert can have different radiometric ranges. This factor is preset to 1.
- Spectral band number: this number is the one of the DART simulation
- Sun-View angle properties: sun and view azimuth and zenith angles.

In addition, for the inversion approach, one must specify (*Figure 32*.d):

- the parameters to invert and the parameters that are fixed.
- an index (called "EquivalentLUT index") for each free parameter. Its value is found in the inversion.xml file in the input directory of the reference simulation. The index "-1" means that the parameter is not free. If the parameter is fixed, then it is not necessary to set this index.

a)



b)

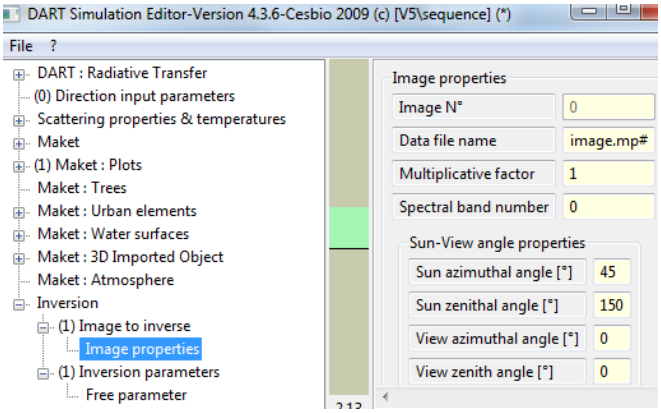


Figure 30: Image characteristics and sun-view configuration

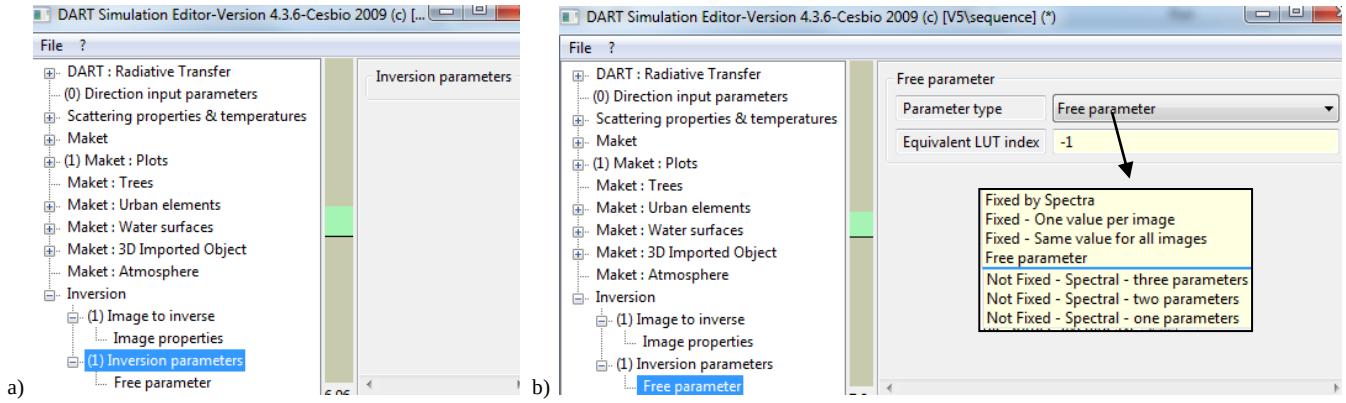


Figure 31: Specification of the type or free and fixed parameters.

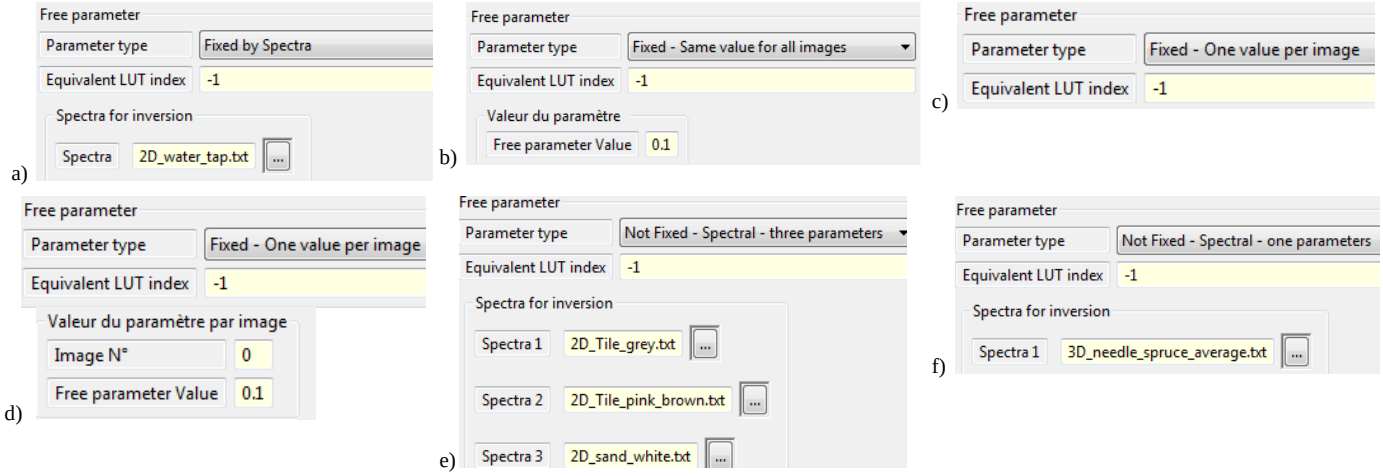


Figure 32: Specification of the free parameters.

Once all inversion parameters are specified, the inversion can be run (Figure 33).

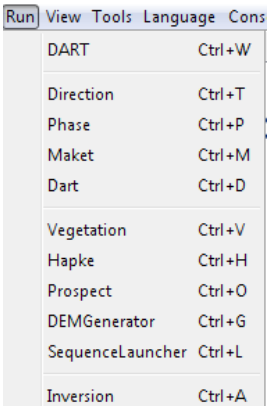


Figure 33: Run of the inversion procedure.

III. Command lines

IV. Example of validation

Here, the inversion is illustrated with the inversion of 3 DART spectral nadir images (0.42 - 0.45 µm, 0.58 - 0.60 µm, 0.8 - 0.9 µm) (*Figure 34*), with $\theta_{s,DART} = 180°$, $\theta_v = 0°$ and a scene (10m x 10m; $\Delta r = 1m$)

that is flat, lambertian (ρ_{ground}: 2D_Agdal_path.txt) with 4 vegetation plots (spherical LAD, ρ_{leaf}: 3D_grass_dry.txt), each with a specific LAI (top left: 0.1, top right: 1, lower right: 2, lower left: 3)

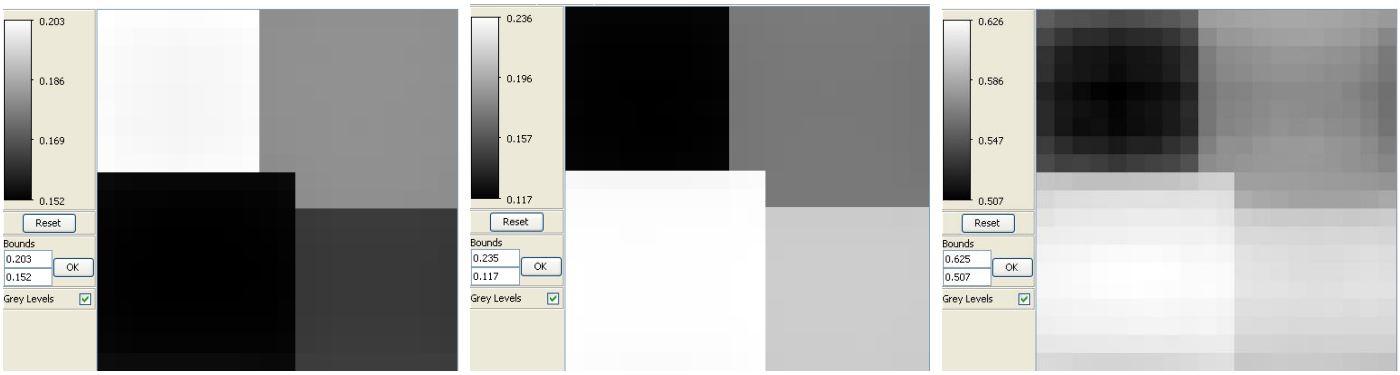
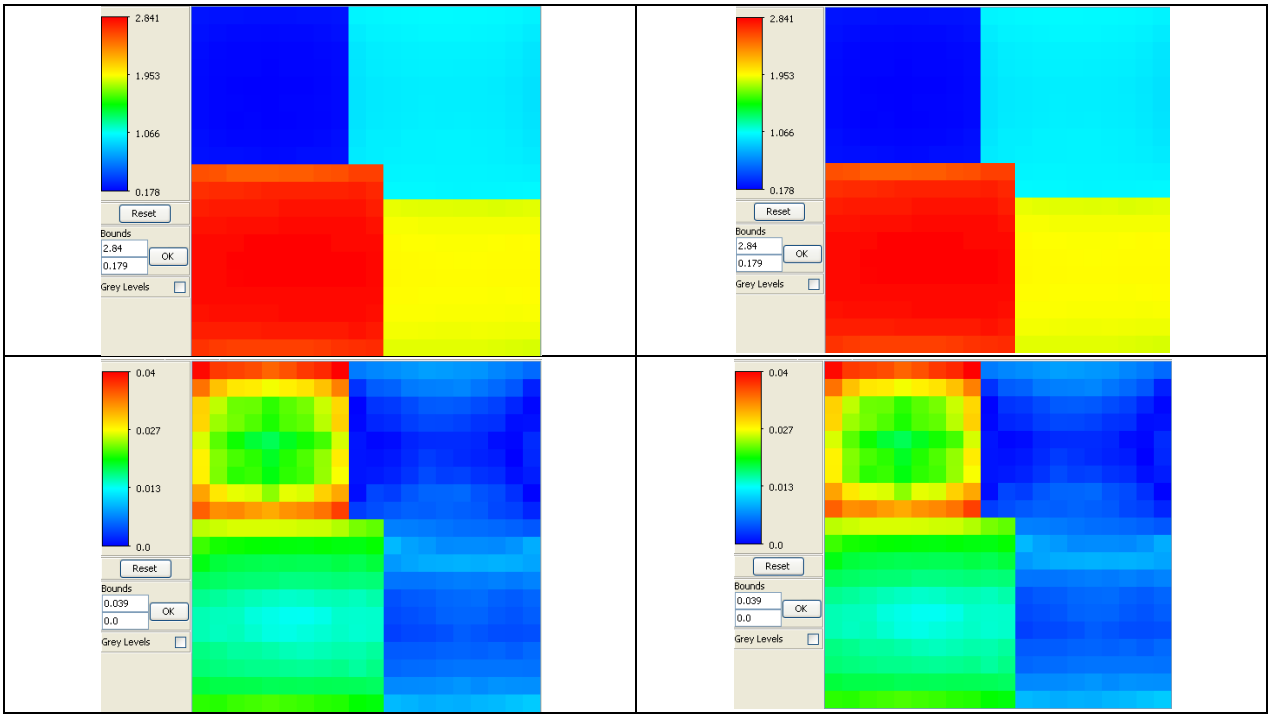


Figure 34: DART simulated images: 100 x 100m in 3 bands: 0.42 – 0.45 µm, 0.58 – 0.60 µm and 0.80 - 0.90 µm

The LUT is for constant soil and vegetation parameters and variable plot LAI ($m^2.m^{-2}$): 0.1; 1; 2; 3, with the fixed derivatives method of interpolation, 5 initial simplexes, a 0.01 relative error stop threshold and a relative error cost function.

The inversion works with 1 coefficient (coefficient = DART nadir reflectance) or 6 coefficients. The LAI is the only free parameter. As expected, LAI is well retrieved on the 4 zones with identical results for the cases "1 coefficient" and "6 coefficients".

Zone	Exact LAI ($m^2.m^{-2}$)	Assessed LAI ($m^2.m^{-2}$)	Absolute error ($m^2.m^{-2}$)	Relative LAI error (%)
1	0.1	0.205	0.105	105
2	1.0	1.016	0.016	1.6
3	2.0	1.938	0.062	3.1
4	3.0	2.757	0.243	8.1



Maps of LAI (top) and RMSE (bottom) obtained by inversion with 1 (left) and 6 (right).coefficients.