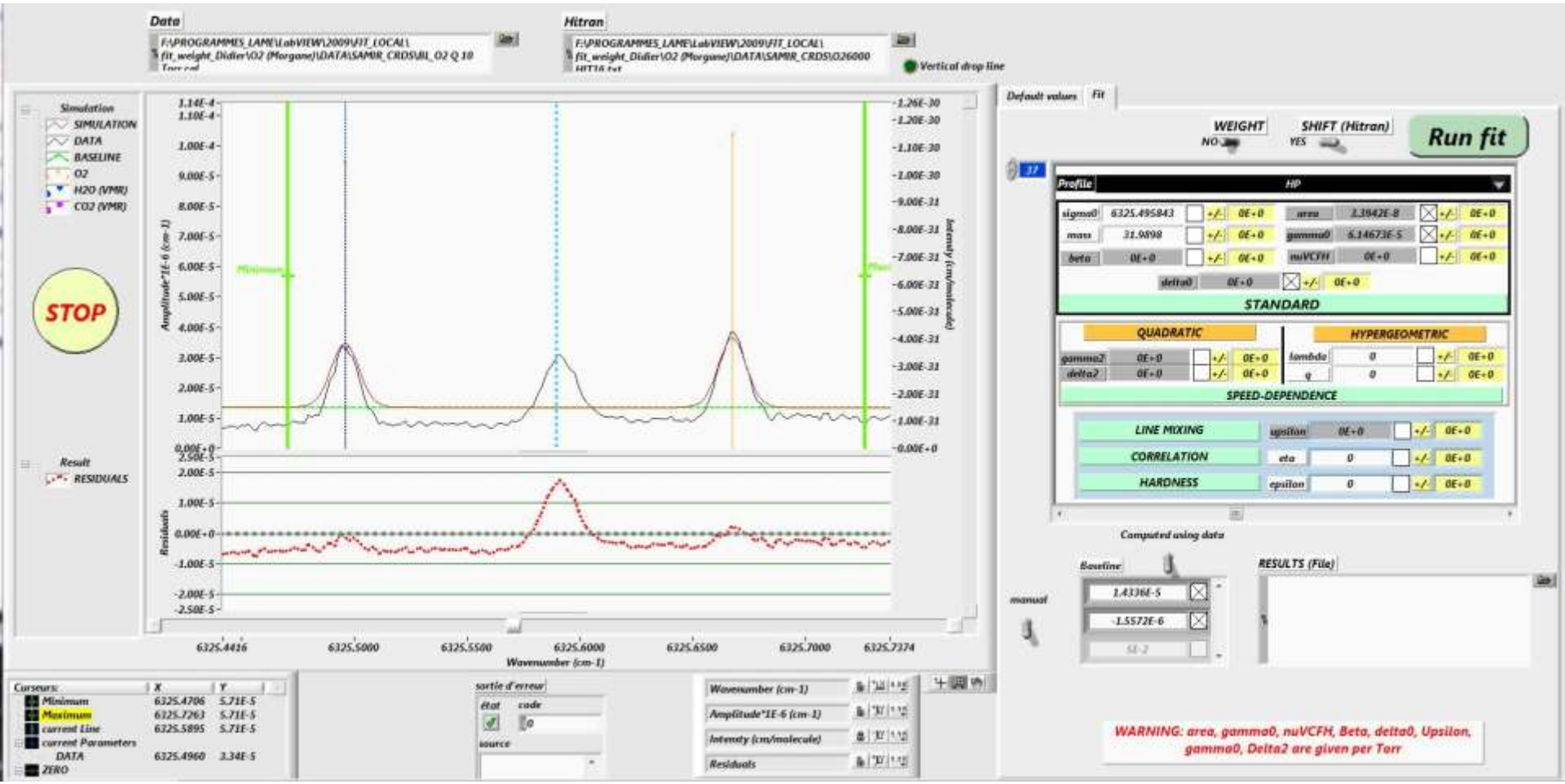
Laboratoire Interdisciplinaire de PHYsique

C++ et LabVIEW: adaptation des types dans un appel de DLL

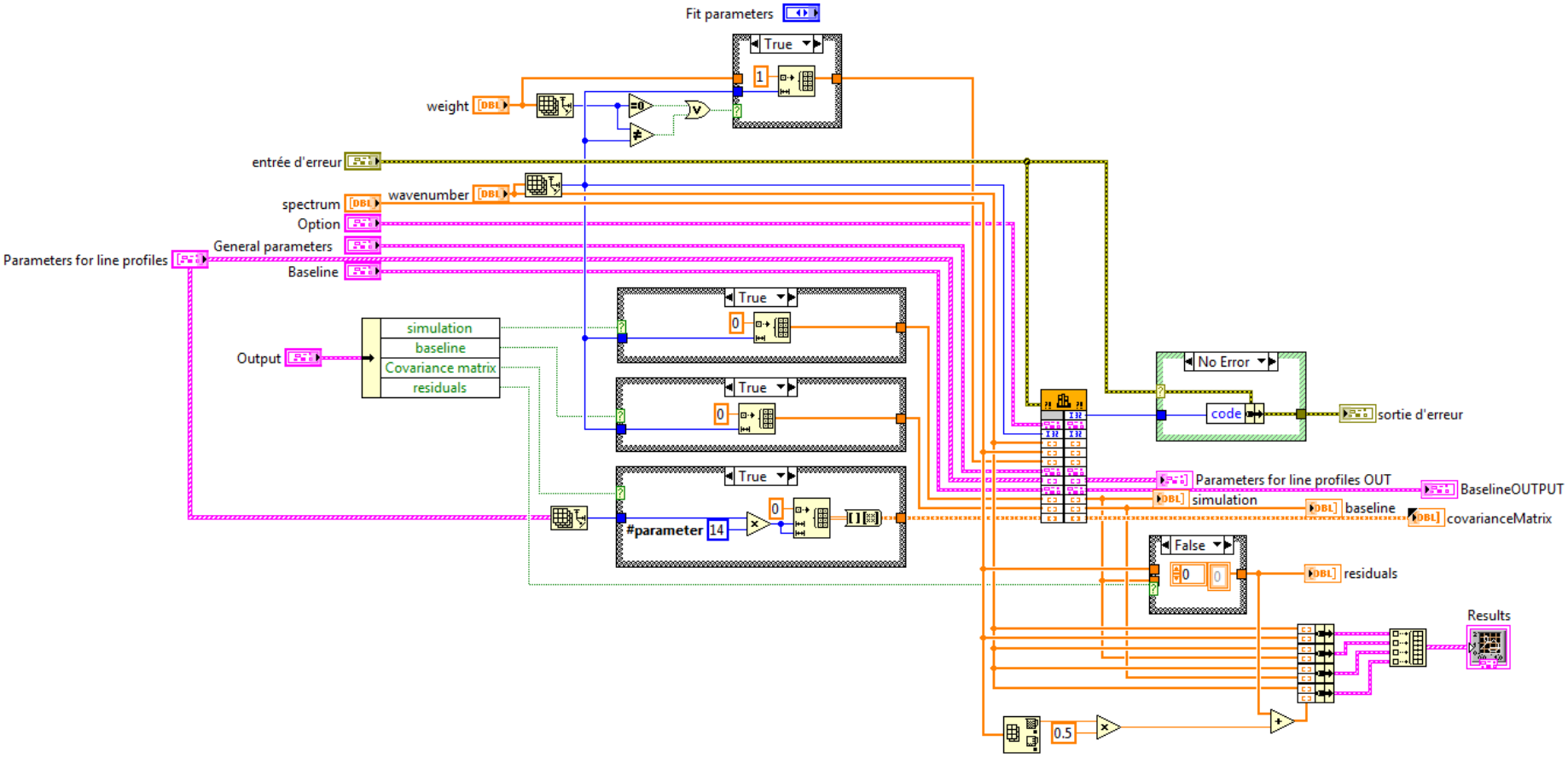
Serge Béguier

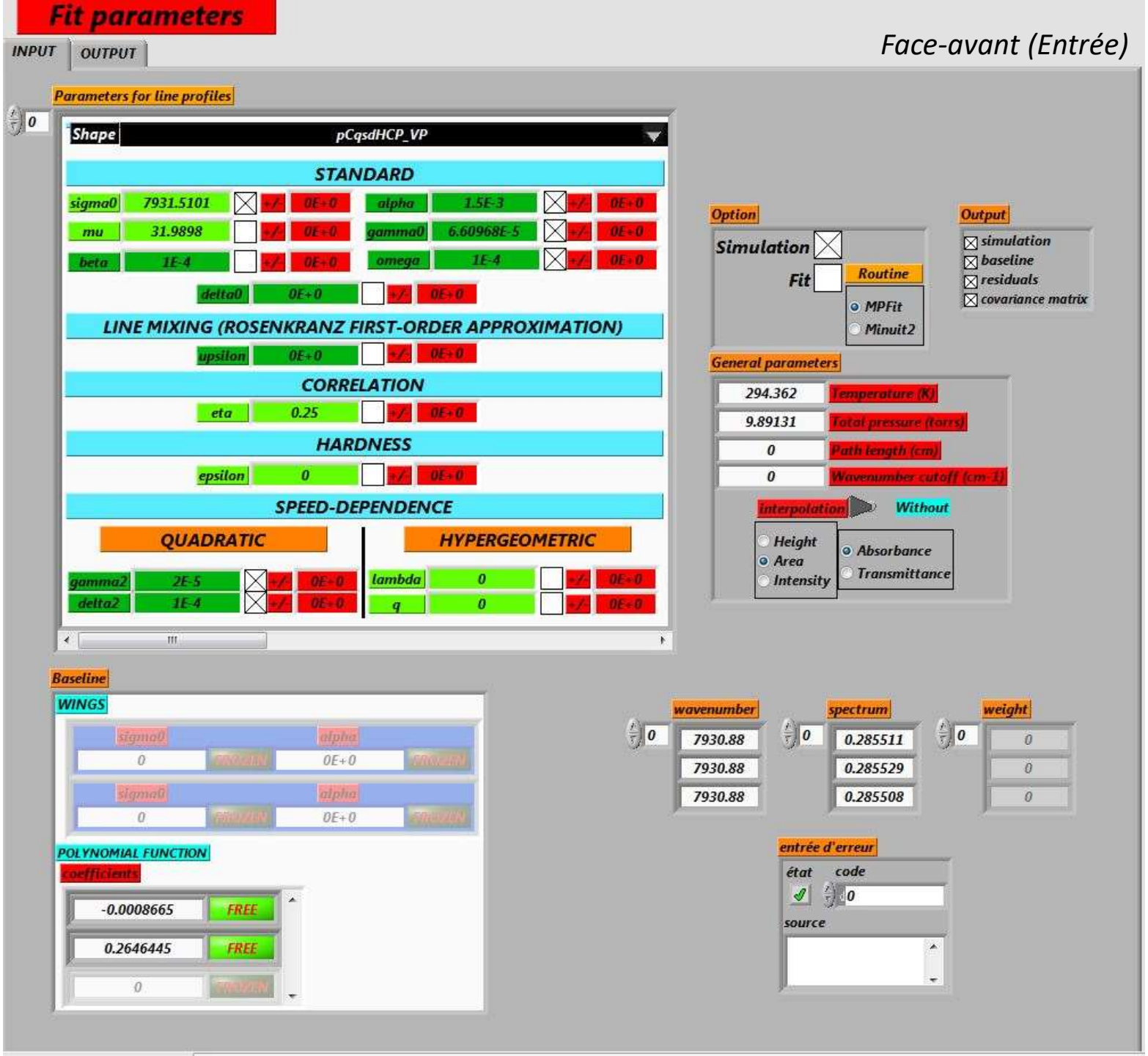
I] Contexte

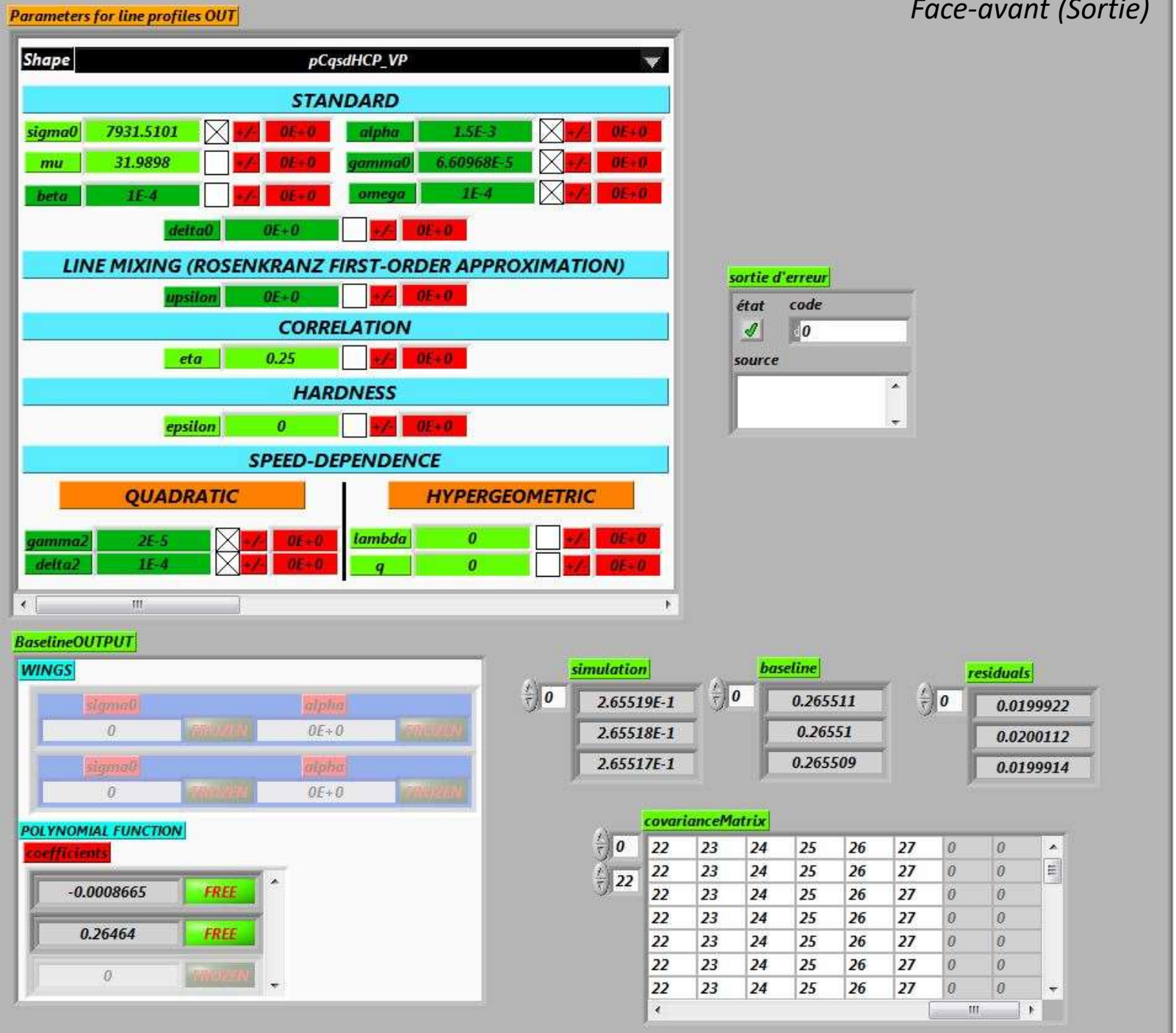

Data
Hitran
Vertical drop line
Simulation
SIMULATION
DATA
BASELINE
O2
H2O (VMR)
CO2 (VMR)
STOP
Result
RESIDUALS
Amplitude*1E-6 (cm-1)
Intensity (cm/molecule)
Residuals
Wavenumber (cm-1)
Default values
Fit
WEIGHT
SHIFT (Hitran)
Run fit
Profile
HP
STANDARD
QUADRATIC
HYPERGEOMETRIC
SPEED-DEPENDENCE
LINE MIXING
CORRELATION
HARDNESS
Computed using data
Baseline
manual
RESULTS (File)
Curseurs:
Minimum
Maximum
current Line
current Parameters
DATA
ZERO
sortie d'erreur
WARNING: area, gamma0, nuVCFH, Beta, delta0, Upsilon, gamma0, Delta2 are given per Torr

II] Sous-programme de simulation

Diagramme




Fit parameters
INPUT
OUTPUT
Face-avant (Entrée)
Parameters for line profiles
0
Shape
pCqsdHCP_VP
STANDARD
sigma0
7931.5101
+/-
0E+0
alpha
1.5E-3
+/-
0E+0
mu
31.9898
+/-
0E+0
gamma0
6.60968E-5
+/-
0E+0
beta
1E-4
+/-
0E+0
omega
1E-4
+/-
0E+0
delta0
0E+0
+/-
0E+0
LINE MIXING (ROSENKRANZ FIRST-ORDER APPROXIMATION)
upsilon
0E+0
+/-
0E+0
CORRELATION
eta
0.25
+/-
0E+0
HARDNESS
epsilon
0
+/-
0E+0
SPEED-DEPENDENCE
QUADRATIC
HYPERGEOMETRIC
gamma2
2E-5
+/-
0E+0
delta2
1E-4
+/-
0E+0
lambda
0
+/-
0E+0
q
0
+/-
0E+0
Option
Simulation
Fit
Routine
MPFit
Minuit2
Output
simulation
baseline
residuals
covariance matrix
General parameters
294.362
Temperature (K)
9.89131
Total pressure (torrs)
0
Path length (cm)
0
Wavenumber cutoff (cm-1)
interpolation
Without
Height
Area
Intensity
Absorbance
Transmittance
Baseline
WINGS
sigma0
alpha
0
FROZEN
0E+0
FROZEN
sigma0
alpha
0
FROZEN
0E+0
FROZEN
POLYNOMIAL FUNCTION
coefficients
-0.0008665
FREE
0.2646445
FREE
0
FROZEN
wavenumber
0
7930.88
7930.88
7930.88
spectrum
0
0.285511
0.285529
0.285508
weight
0
0
0
0
entrée d'erreur
état
code
0
source


Parameters for line profiles OUT
Shape
pCqsdHCP_VP
STANDARD
sigma0 7931.5101 +/- 0E+0
alpha 1.5E-3 +/- 0E+0
mu 31.9898 +/- 0E+0
gamma0 6.60968E-5 +/- 0E+0
beta 1E-4 +/- 0E+0
omega 1E-4 +/- 0E+0
delta0 0E+0 +/- 0E+0
LINE MIXING (ROSENKRANZ FIRST-ORDER APPROXIMATION)
upsilon 0E+0 +/- 0E+0
CORRELATION
eta 0.25 +/- 0E+0
HARDNESS
epsilon 0 +/- 0E+0
SPEED-DEPENDENCE
QUADRATIC
HYPERGEOMETRIC
gamma2 2E-5 +/- 0E+0
delta2 1E-4 +/- 0E+0
lambda 0 +/- 0E+0
q 0 +/- 0E+0
sortie d'erreur
état
code
0
source
BaselineOUTPUT
WINGS
sigma0
alpha
0
FROZEN
0E+0
FROZEN
sigma0
alpha
0
FROZEN
0E+0
FROZEN
POLYNOMIAL FUNCTION
coefficients
-0.0008665 FREE
0.26464 FREE
0 FROZEN
simulation
0
2.65519E-1
2.65518E-1
2.65517E-1
baseline
0
0.265511
0.26551
0.265509
residuals
0
0.0199922
0.0200112
0.0199914
covarianceMatrix
0
22
22 23 24 25 26 27 0 0

Rappels:

int age = 10;
int * pointeurSurAge = &age;

printf ("%d", pointeurSurAge);

// 177450

---

int age = 10;
int * pointeurSurAge = &age;

printf ("%d", * pointeurSurAge);

// 10

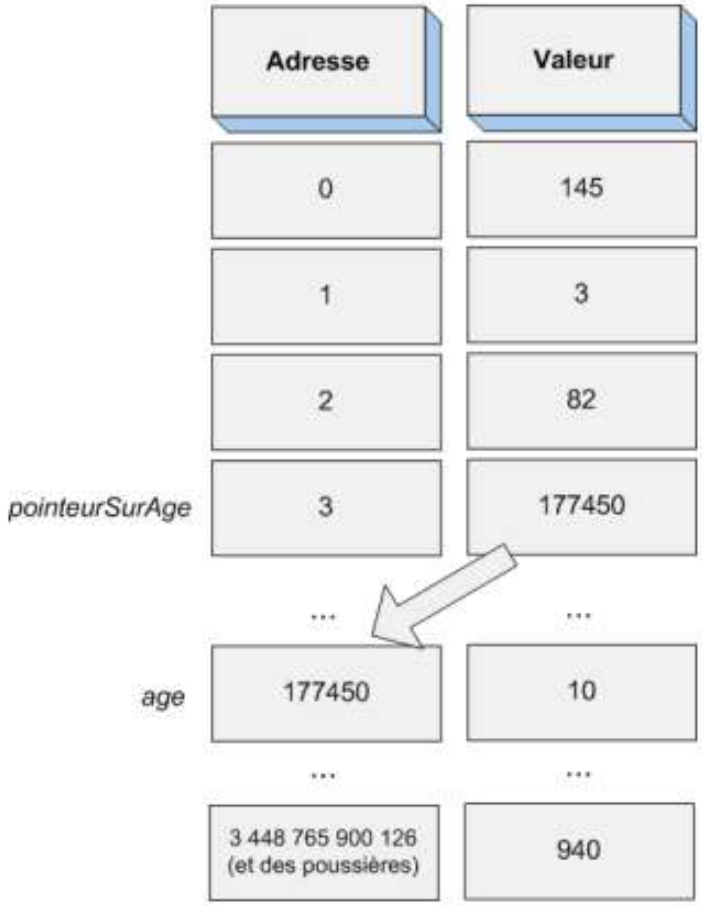
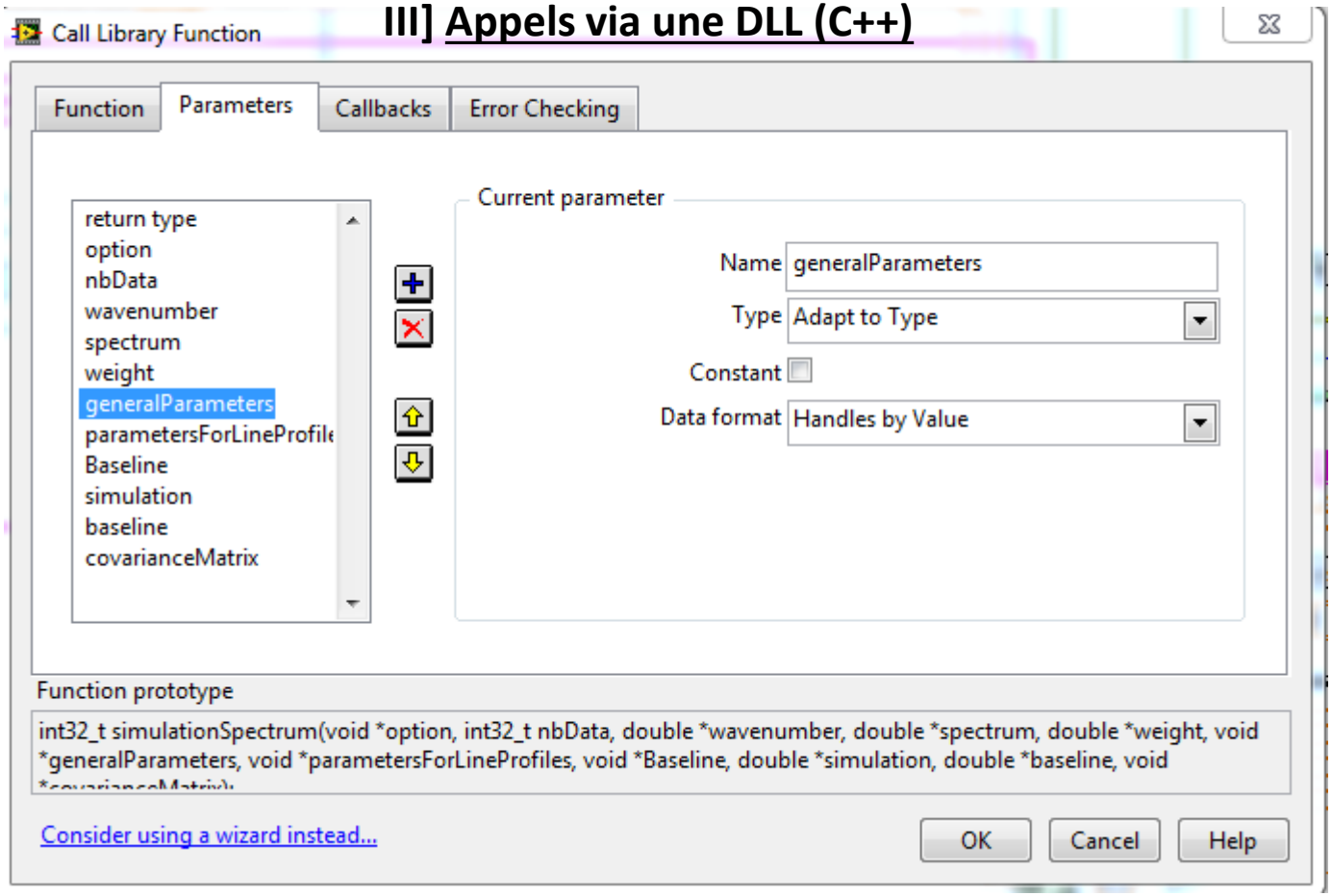


FIGURE 11.2 – Mémoire, adresses et pointeurs

III] Appels via une DLL (C++)

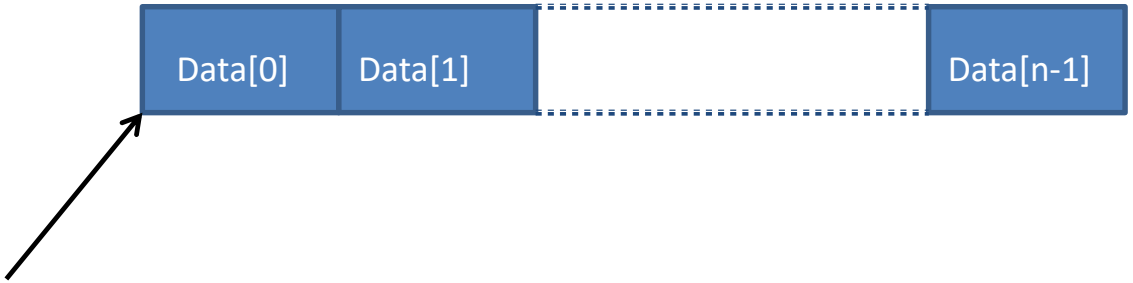


Code.c créé par LabVIEW:

[extern "C" __declspec(dllexport)] *int32_t simulationSpectrum(TD1 *option, int32_t nbData, double wavenumber[], double spectrum[], double weight[], TD2 *generalParameters, TD3Hdl *parametersForLineProfiles, TD5 *Baseline, double simulation[], double baseline[], TD10Hdl *covarianceMatrix) {…}*

A) <u>Tableau (càd un vecteur): Pointeur sur les données du tableau</u>

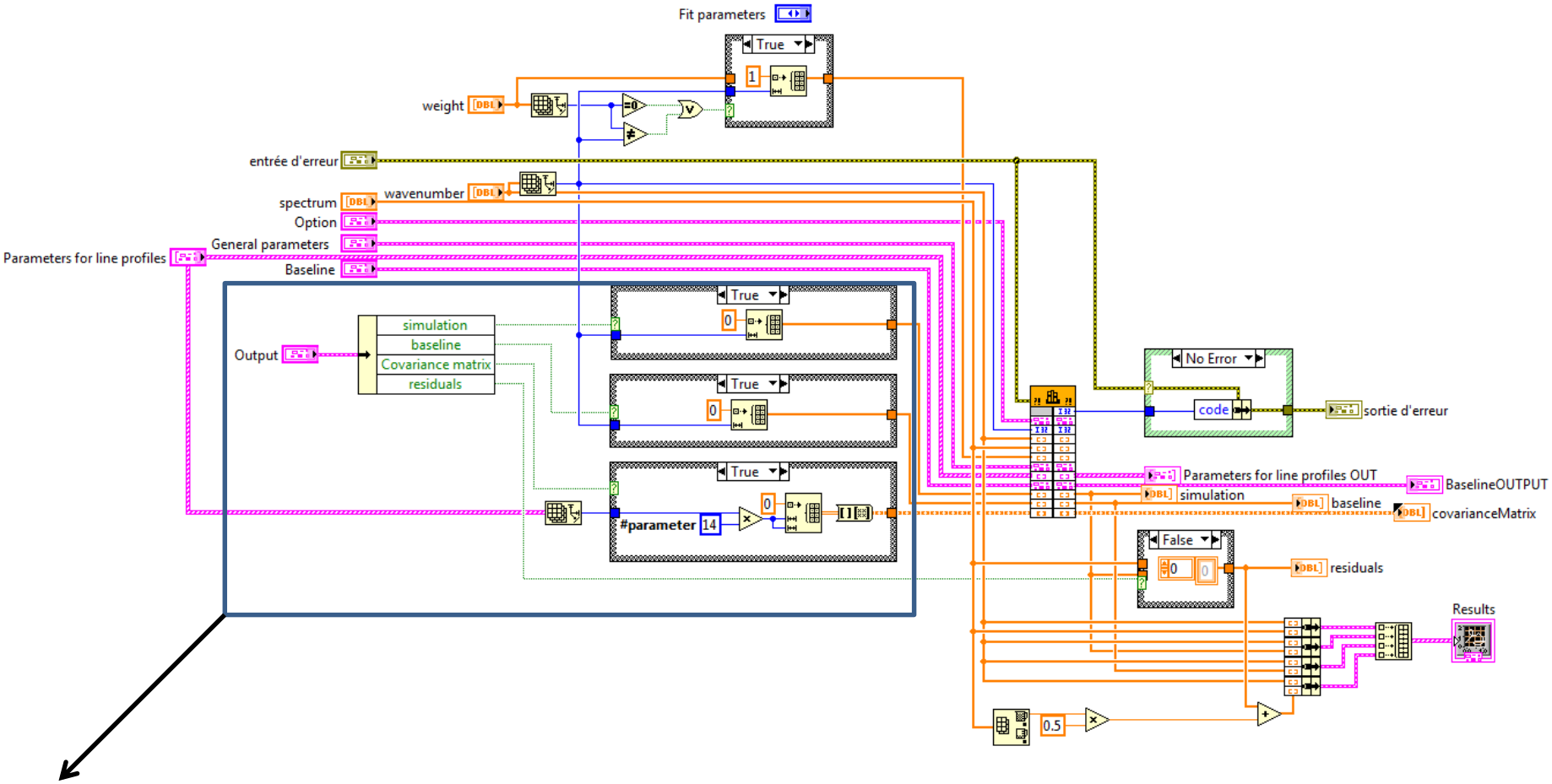
(*Dans l'exemple: double simulation[] ou double baseline[]*)



Pointeur sur le premier élément: &simulation[0] (en C)

Code C++: double dataDLL0(simulation[0]); // Lecture
simulation[0] = 1984.0; // Écriture

Pour parcourir ce tableau, il faut transmettre à la DLL la taille de celui-ci…
(*Dans l'exemple: int nbData*)



Réservation par LabVIEW de l'espace mémoire nécessaire pour le(s) nouveau(x) vecteur(s)

B) Tableau (càd un vecteur): Handle du tableau

Pointeur sur le premier élément de la structure suivante:

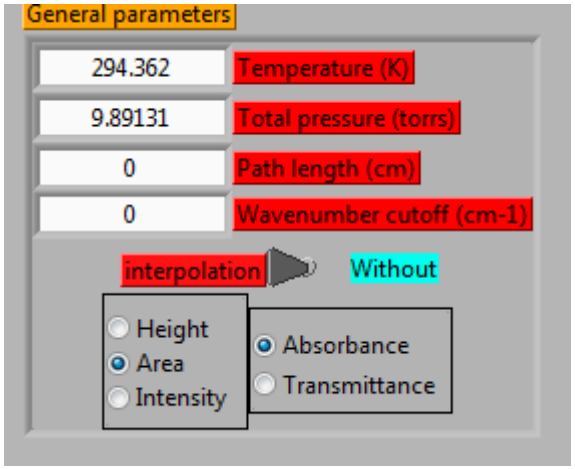
```
typedef struct {
        int32 dimSize;
        long elt[1];
        } TD2;
typedef TD2 **  TD2Hdl;
```

```
Code C++: int Size((*dataInput)->dimSize);          // Lecture
          double dataDLL0((*dataInput)->elt[0]);    // Lecture
           (*dataOutput)->elt[0] = 1984.0;          // Écriture
```

2 tableaux sinon vous écrivez dans le même espace mémoire…

C) Cluster «General Parameters »: Handles par valeur

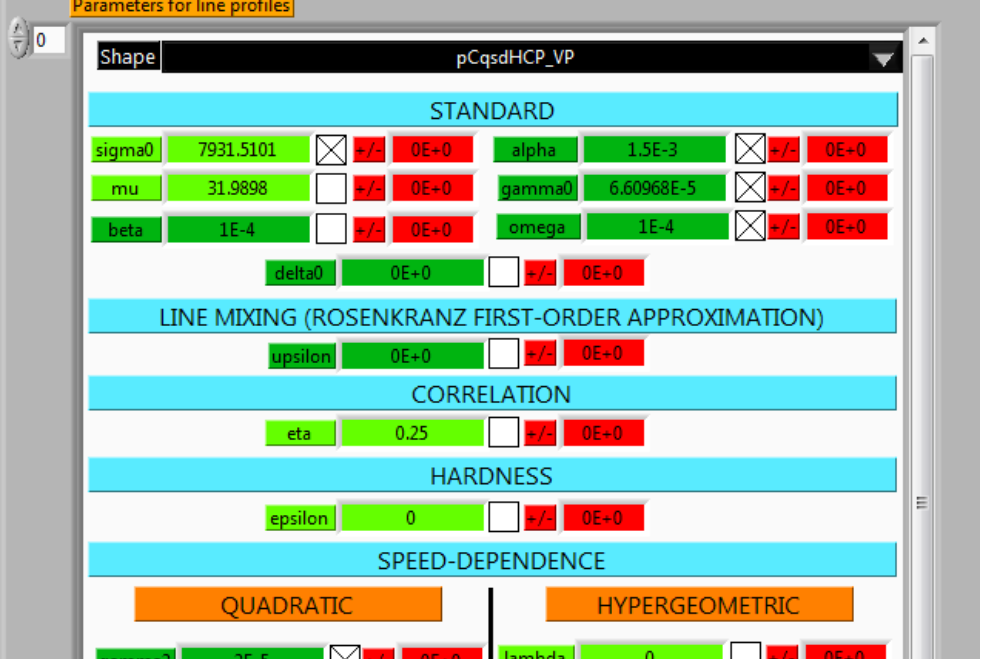
(*Dans l'exemple: TD2 *generalParameters)*



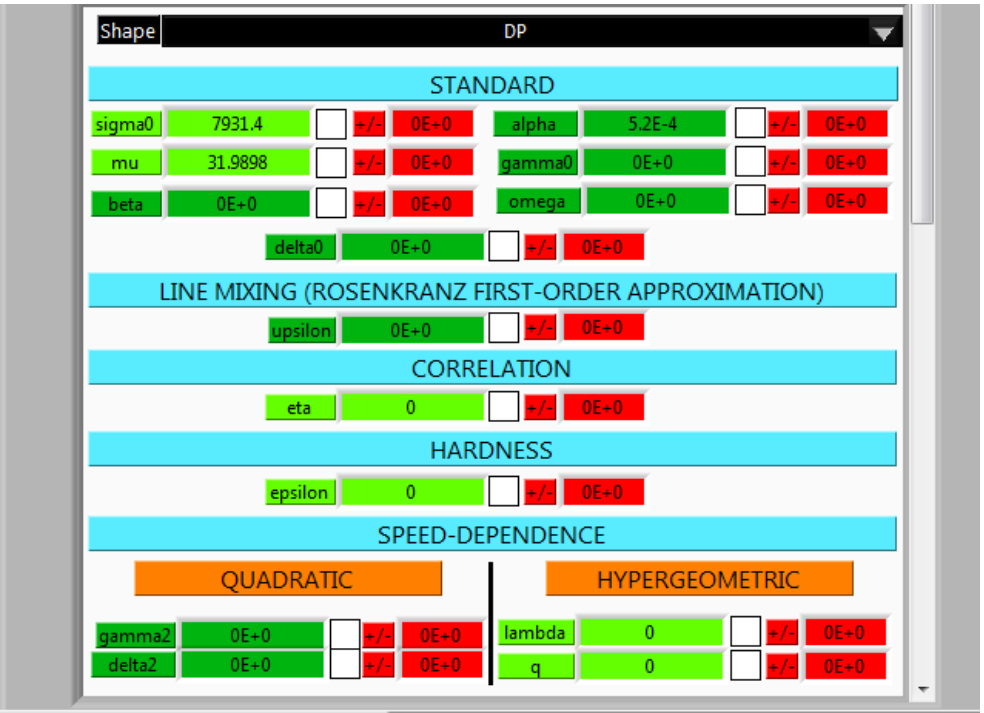

```
typedef struct {
        double TemperatureK;
        double PathLengthCm;
        double TotalPressureTorrs;
        double WavenumberCutoffCm1;
        uint32_t type;
        uint32_t alpha0;
        LVBoolean interpolation;
        } TD2;
```

Code C++:

double temperature(generalParameters-> TemperatureK); ***// Lecture***



(*Dans l'exemple: TD3Hdl *parametersForLineProfiles*)



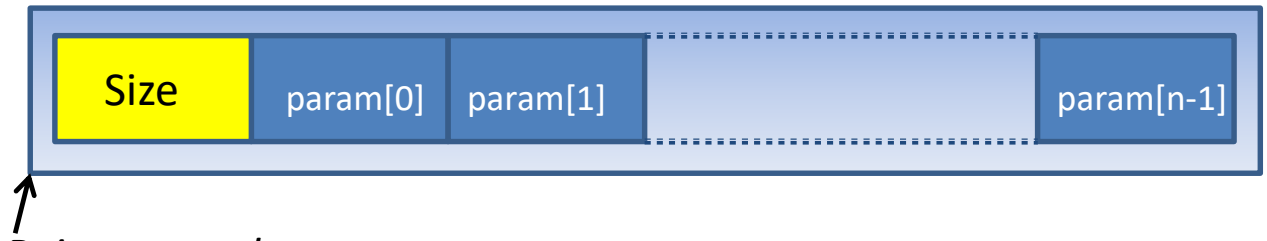
D) Cluster « parametersForLineProfiles »: Pointeurs sur les handles

(i) →

```
typedef struct {
        double sigma0; double alpha;
        double mu; double gamma0;
        double beta; double omega;
        double delta0; double upsilon;
        double eta; double epsilon;
        double gamma2; double delta2;
        double lambda; double q;
        double sigma0Error; double alphaError;
        double muError; double gamma0Error;
        double betaError; double omegaError;
        double delta0Error; double upsilonError;
        double etaError; double epsilonError;
        double gamma2Error; double delta2Error;
        double lambdaError; double qError;
        LVBoolean sigma0Fixed; LVBoolean alphaFixed;
        LVBoolean muFixed; LVBoolean gamma0Fixed;
        LVBoolean betaFixed; LVBoolean omegaFixed;
        LVBoolean delta0Fixed; LVBoolean upsilonFixed;
        LVBoolean etaFixed; LVBoolean epsilonFixed;
        LVBoolean gamma2Fixed; LVBoolean delta2Fixed;
        LVBoolean lambdaFixed; LVBoolean qFixed;
        uint16_t Shape;
        } TD4;
typedef struct {
        int32_t dimSize;
        TD4 param[1];
        } TD3;
typedef TD3**  TD3Hdl;
```

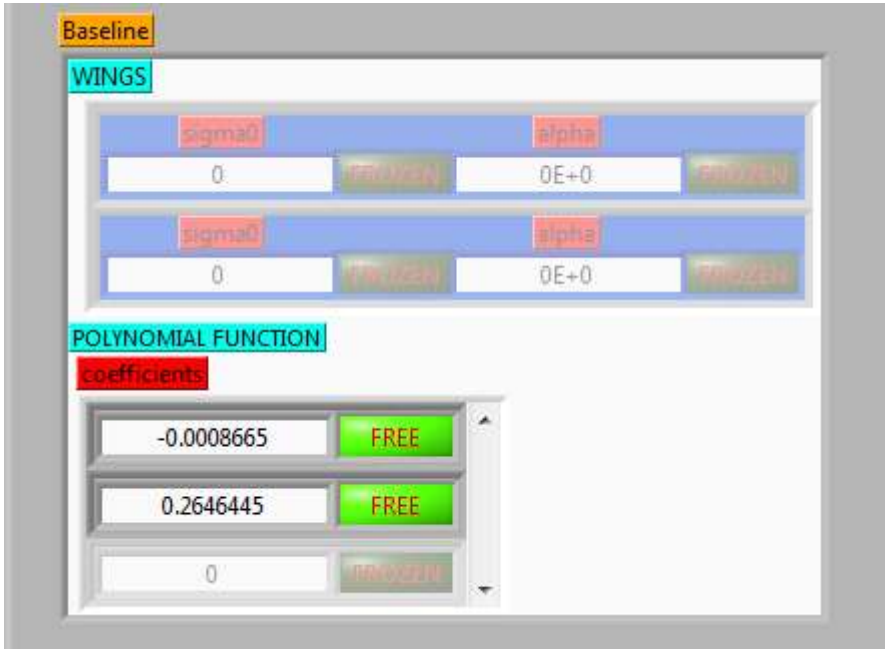
Code C++:

Pour les clusters, LabVIEW passe, par convention, des handles du tableau.




```
// Lecture
int nbShape = (**parametersForLineProfiles)->dimSize;
double center0 = (**parametersForLineProfiles)->param[0].sigma0;

// Ecriture
(**parametersForLineProfiles)->parameters[0].sigma0 = centerFit;
```

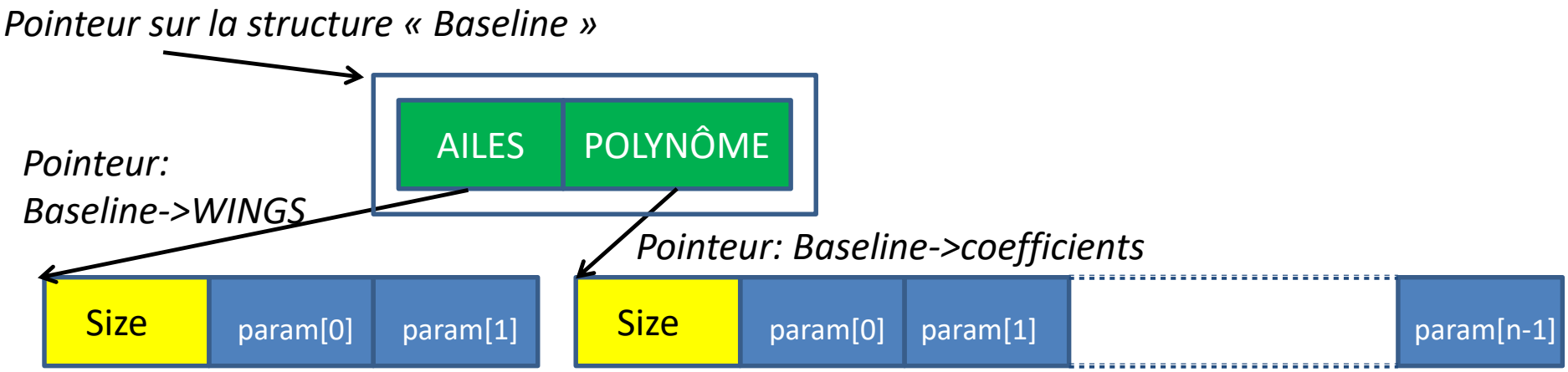
(*Dans l'exemple: TD5 *Baseline)*



(ii)

```
typedef struct {
        double ai;
        LVBoolean aiFree;
        } TD7;

typedef struct {
        int32_t dimSize;
        TD7 ai[1];
        } TD6;
typedef TD6**  TD6Hdl;

typedef struct {
        double sigma0;
        double alpha;
        LVBoolean sigma0Free;
        LVBoolean alpha0Free;
        } TD9;

typedef struct {
        int32_t dimSize;
        TD9 parWings[1];
        } TD8;
typedef TD8**  TD8Hdl;

typedef struct {
        TD6Hdl coefficients;
        TD8Hdl WINGS;
        } TD5;
```

Code C++:




```
if (Baseline->WINGS != nullptr)
        int nbD_Aile = (*(Baseline->WINGS))->dimSize;  // Lecture

[…]

if (Baseline->coefficients != nullptr)
        for (long i = 0; i < (*(Baseline->coefficients))->dimSize; ++i)
         // Lecture
                shapeParameters.push_back((*(Baseline->coefficients))->ai[i].ai);
```

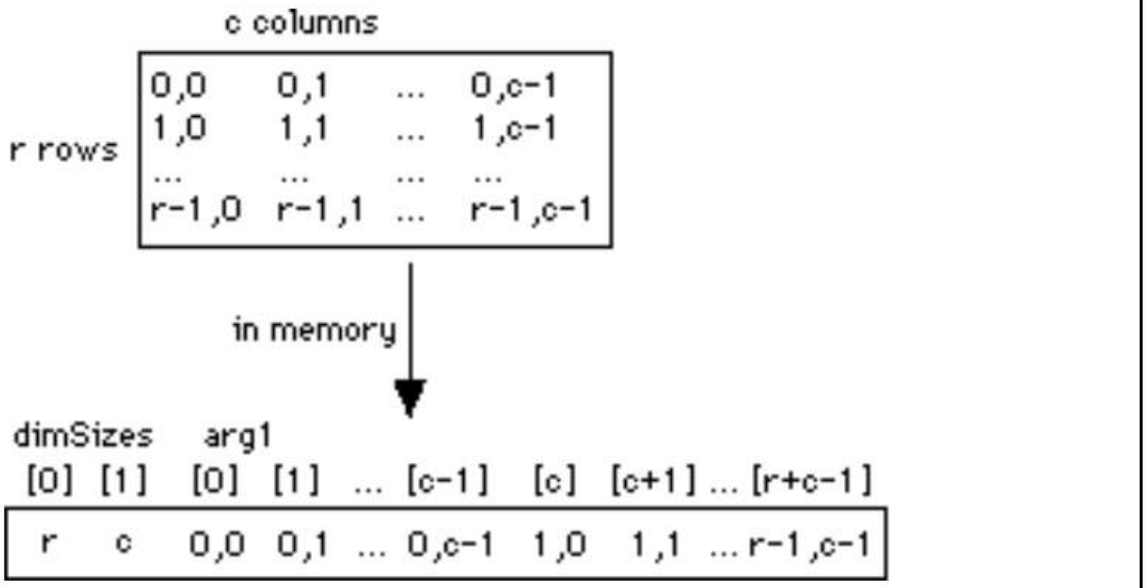
*(Dans l'exemple: TD10Hdl *covarianceMatrix)*

(iii) →

```
typedef struct {
        int32_t dimSizes[2];
        double RealMatrixElement[1];
        } TD10;
typedef TD10**  TD10Hdl;
```

Code C++:

```
if (covarianceMatrix != nullptr)
{
    int matrixLengthRow = (**covarianceMatrix)->dimSizes[0];  // Lecture
    int matrixLengthCol   = (**covarianceMatrix)->dimSizes[1];
    for (int i = 0; i < matrixLengthRow; i++)
    {
            for (int j = 0; j < matrixLengthCol; j++)
            {
               // Écriture
              (**covarianceMatrix)->RealMatrixElement[(i * matrixLengthCol) + j] = (double)j;
            }
    }
}
```



[dans le manuel LabVIEW: « Using external code in LabVIEW » (p.4-14)]

IV] Conclusion

Pour tous renseignements supplémentaires voir les différents cas dans:
Call DLL.vi *ou* **External Code (DLL) Execution.vi** *de « trouver un exemple.. »*