Defence Research and Development Canada
Recherche et développement pour la défense Canada

DEFENCE R&D DÉFENSE

Développement d'une interface hommemachine en C-sharp pour PAASoM

Morgane Delahaye, Damien Rialet et Mathieu Caillet

R & D pour la défense Canada – Ottawa

Rapport de contrat
DRDC Ottawa CR 2012-116
septembre 2012

Canada

Développement d'une interface homme-machine en C-sharp pour PAASoM

Morgane Delahaye
Collège Militaire Royal du Canada

Damien Rialet
Collège Militaire Royal du Canada

Mathieu Caillet
Collège Militaire Royal du Canada

Préparé par :
Collège Militaire Royal du Canada
CP 17000, Succursale Forces
Kingston, Ontario, Canada K7K 7B4
Numéro du document de l'entrepreneur :
Administrateur de projet de l'entrepreneur : Yahia M. M. Antar, 613-541-6000, #6403
Numéro de contrat TPSGC : C1410FE121
ASC : Michel Clénet, Scientifique de la Défense, 613-993-7397

R & D pour la défense Canada – Ottawa

Rapport de contrat
DRDC Ottawa CR 2012-116
septembre 2012

Auteur principal

Original signé par Michel Clénet

Michel Clénet

Authorité scientifique

Approuvé par

Original signé par Bill Katsube

Bill Katsube

CS Guerre Électronique (Comm. & Nav.)

Diffusion approuvée par

Original signé par Chris McMillan

Chris McMillan

Scientifique en chef

Abstract

This report presents the development of a graphical user interface (GUI) for PAASoM (Phased Array Antenna Simulation Tool) using the programming language C Sharp (C#). PAASoM was originally implemented using Matlab, and graphic components and event handling of its GUI were limiting further development of the tool. Thus, it was decided to use a highly capable language for the PAASoM GUI, and keep the Matlab code that achieves the simulation itself. This was possible thanks to the Matlab Builder NE toolbox which allows compiling Matlab code as a dynamic-link library (DLL) and invoking those latter in C#. The environment used to implement the C# interface is Visual Basic Express 2010.

Résumé

Ce document présente le développement d'une interface homme-machine (IHM) en langage C sharp (C#) pour PAASoM (outil de simulation de réseau d'antennes à déphasage). PAASoM a initialement été implémenté avec Matlab, et les composants graphiques et la gestion d'événements associée à son IHM limitaient jusqu'à présent son développement. Il a donc été décidé d'utiliser un langage plus performant pour l'IHM de PAASoM, tout en conservant le code Matlab pour la simulation en tant que telle. Ceci est rendu possible grâce au module Matlab Builder NE qui permet de compiler le code Matlab sous forme de bibliothèques dynamiques 'DLL' et d'invoquer ces dernières en C#. L'environnement utilisé pour implémenter l'interface C# est Visual Basic Express 2010.

Page laissée en blanc à dessein.

Executive summary

Développement d'une interface homme-machine en C-sharp pour PAASoM :

Morgane Delahaye; Damien Rialet; Mathieu Caillet; DRDC Ottawa CR 2012-116; Defence R&D Canada – Ottawa; September 2012.

Background: The development of the Phased Array Antenna Software Tool (PAASoM) started early in the first decade of 2000 at Defense Research and Development Canada (DRDC), and a graphical user interface (GUI) was added in 2004 for a better interactivity. The code and the GUI of PAASoM were implemented using Matlab to take advantage of the computation power and the matrix functionalities offered by Matlab. However, the graphical components and the event handling associated with those components were limiting the development of PAASoM. To tackle this, one solution is to keep the current Matlab code used for the computation, and develop the interface in a different environment.

Results: The PAASoM GUI has been implemented using the C Sharp (C#) programming language. To achieve this, the Microsoft Visual Studio Express 2010 environment was used. This environment facilitated the design of the different PAASoM windows since it provides advanced graphical components that are not available in Matlab.

Significance: The present work allowed the design of a more user friendly, intuitive, complete and modular interface for PAASoM compared to the current interface. Thanks to the interface implemented in C#, the future development of PAASoM will be facilitated to add new advanced graphical functionalities, and the re-organization of the way input data are entered will be carried out in a smoother way as PAASOM is further developed.

Future plans: The next step will consist of implementing the “Plotting options” window to complete the PAASoM interface in C#. Then, the Matlab code achieving the computation of the simulated problem should be integrated to the C# code. To realize this, the considered Matlab functions will have to be compiled using the Matlab Builder NE toolbox and invoked in the C# code.

Sommaire

Développement d'une interface homme-machine en C-sharp pour PAASoM :

Morgane Delahaye; Damien Rialet; Mathieu Caillet ; DRDC Ottawa CR 2012-116 ; R & D pour la défense Canada – Ottawa ; septembre 2012.

Contexte : Le développement de l'outil de simulation de réseau d'antennes à déphasage a débuté au centre de Recherche et Développement pour la Défense Canada (RDDC) au début des années 2000, et une interface homme-machine (IHM) a été ajoutée en 2004 pour en faciliter son utilisation. Le code et l'IHM de PAASoM ont été codé en utilisant Matlab pour bénéficier de la puissance de calcul et des outils matriciels qu'offre Matlab. En revanche, les composants graphiques et la gestion d'événements associée à ces composants limitent le développement de PAASoM. Pour remédier à cela, une solution est de conserver le code Matlab, et de développer l'IHM de PAASoM dans un environnement différent.

Résultats : L'IHM de PAASoM a été implémentée en utilisant le langage C sharp (C#). Pour cela, l'environnement Microsoft Visual Studio Express 2010 a été utilisé. Cet environnement a facilité la conception des différentes fenêtres de PAASoM car il possède des composants graphiques avancés qui ne sont pas présent dans Matlab.

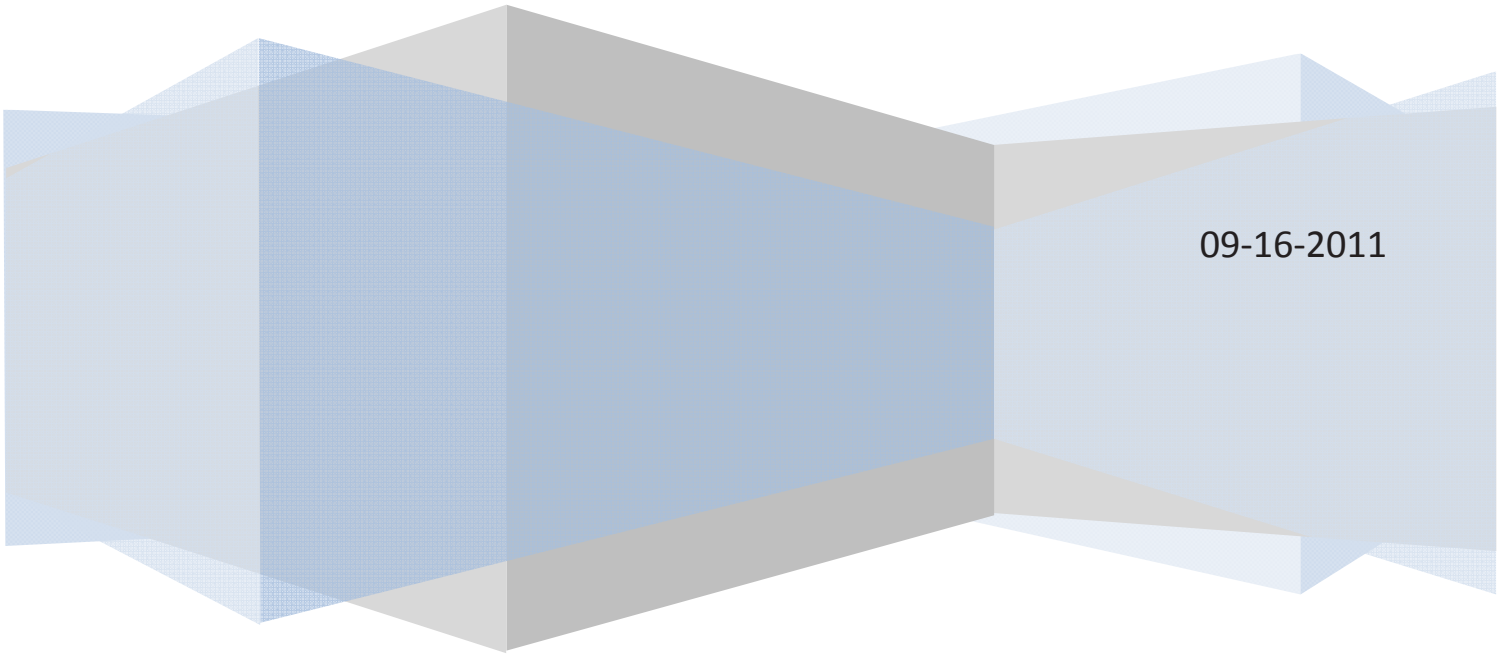
Importance : Les présents travaux ont permis la conception d'une IHM pour PAASoM plus conviviale, plus intuitive, plus aboutie et plus modulaire par rapport à l'interface actuelle. Grâce à l'interface implémentée en C#, les développements futures de PAASoM seront simplifiés pour l'ajout de nouvelles fonctionnalités graphiques avancées, et la réorganisation de la saisie des données se fera de manière plus progressive au fur et à mesure du développement de PAASoM.

Perspectives : La prochaine étape consiste à implémenter la fenêtre "Plotting options" pour terminer l'interface de PAASoM en C#. Ensuite, le code Matlab réalisant les calculs correspondant au problème simulé devra être intégré en C#. Pour cela, les fonctions Matlab concernées devront être compilées à l'aide du module Matlab Builder NE, et invoquée dans le code C#.

Royal Military College of Canada - Collège militaire royal du Canada

Développement d'une interface homme-machine en C# pour PAASoM

Morgane Delahaye



09-16-2011

Sommaire

Sommaire 2
Introduction 5
I- Présentation générale 6
1 .1- Architecture du projet 6
1 .2- Architecture logicielle générale 8
1.3- Conventions pour les noms de variables 9
1.4- Passage des variables 10
1.5- Structure des classes 11
1.6- Fonctions Matlab à convertir au niveau de l'interface 13
1.7 L'environnement Visual Studio Express 2010 14
II- La fenêtre principale 17
2.1- Présentation générale 17
2.2- Pointeurs de la fenêtre « PAASoM » 17
2.3- Méthodes de la classe « Cpassom » 18
III- La fenêtre « Frequency » 21
3.1- Présentation générale 21
3.2- Fonctionnement de la fenêtre 21
3.3- La classe « CfrequencyParameters » 22
3.4- Pointeurs de la fenêtre « Frequency » 22
3.5- Méthodes de la classe « Cfrequency » 23
IV- Fenêtre « Problem Definition » 26
4.1 – Présentation générale 26
4.2 – Fonctionnement de la fenêtre 26
4.3- La classe « CproblemDefParameters » 28
4.4- La classe « Cdirection » 29
4.5- Pointeurs de la fenêtre « Problem definition » 29
4.6– Méthodes de la classe « CproblemDef » 30
V- La fenêtre « Array characteristics » 37
5.1- Présentation générale 37
5.2- Fonctionnement de la fenêtre 37
5.3- La classe « CarrayCharacteristicsParameters » 40
5.4- Pointeurs de la fenêtre « Array characteristics » 41

5.5- Méthodes de la classe « CarrayCharacteristics » 42
VI- Fenêtre « Define ERP » 49
6.1 – Présentation générale 49
6.2 – Fonctionnement de la fenêtre 49
6.3- La classe « CdefineERPParameters » 51
6.4- Pointeurs de la fenêtre « Define ERP » 52
6.5 – Méthodes de la classe « CdefineERP » 53
VII - La fenêtre « Amplitude distribution » 62
7.1- Présentation générale 62
7.2- Fonctionnement de la fenêtre 62
7.3- La classe « CamplitudeDistributionParameters » 63
7.4- Pointeurs de la fenêtre « Amplitude distribution » 63
7.5- Méthodes de la classe « CamplitudeDistribution » 64
VIII - La fenêtre « Element and sub-array components » 68
8.1- Présentation générale 68
8.2- Fonctionnement de la fenêtre 68
8.3- La classe « CelementAndSubArrayComponentsParameters » 70
8.4- La classe « CelementAndSubArrayComponents_Amplitude » 70
8.5- La classe « CelementAndSubArrayComponents_PhaseShifter » 70
8.6- Pointeurs de la fenêtre « Element and sub-array components » 71
8.7- Méthodes de la classe « CelementAndSubArrayComponents » 73
IX- Fenêtre « Simulation method » 79
9.1 – Présentation générale 79
9.2 – Fonctionnement de la fenêtre 79
9.3- La classe « CsimulationMethodParameters » 81
9.4- Pointeurs de la fenêtre « Simulation method » 82
9.5 – Méthodes de la classe « CsimulationMethod » 83
X- Fenêtre « Compute options » 89
10.1 – Présentation générale 89
10.2 – Fonctionnement de la fenêtre 89
10.3- La classe « CcomputeOptionsParameters » 90
10.4- Pointeurs de la fenêtre « Compute options » 91
10.5 – Méthodes de la classe « CcomputeOptions » 92
XI- Intégration du code Matlab en C# 96

Conclusion 98
References 99
[1] PAASoM 99
[2] Visual studio 99
[3] Matlab 99
[4] Microsoft « .NET » 99
[5] Mathworks MWArray assembly 100
Table des illustrations 101

Introduction

PAASoM [1] est un outil permettant d'étudier et d'analyser les réseaux d'antennes à déphasage en prenant en compte leur géométrie et architecture ainsi que les caractéristiques réelles ou approchées des différents éléments d'un réseau (éléments rayonnants, amplificateurs, déphaseurs, unités de retard temporel). Le développement de cet outil a commencé à Recherche et Développement pour la Défense Canada (RDDC) au début des années 2000, et une interface homme-machine a été ajoutée en 2004 pour en faciliter son utilisation. Le code et l'interface de PAASoM ont été codés en utilisant Matlab pour bénéficier de la puissance de calcul et des outils matriciels qu'offre Matlab. En revanche, les composants graphiques et la gestion d'événements associée à ces composants limitent le développement de PAASoM. Pour remédier à cela, une solution est de conserver le code Matlab, et de développer l'interface de PAASoM dans un environnement .NET. Cela est réalisable en compilant le code Matlab sous forme de composants .NET qui sont utilisés dans l'environnement de l'interface. Le langage choisi pour le développement de l'interface .NET est le C Sharp (C#) car c'est un langage haut-niveau orienté objet qui offre à la fois flexibilité, performance et efficacité pour la création d'applications sous Windows.

Ce document est le dossier technique de l'interface graphique créée pour le logiciel PAASoM. Cette interface a été codée en langage C Sharp avec l'environnement Visual Studio express 2010 [2]. Elle reprend l'apparence de l'interface en langage Matlab [3] existante, en étant plus stable et offre une meilleure gestion du passage de variables.
Grâce à l'outil de compilation Matlab Builder NE dédié à l'environnement DOT.NET [4], les fonctions Matlab utilisées dans le cœur de PAASoM pourront être intégrées plus tard à l'interface C#.

Ce document débute par la présentation générale du projet, en particuliers l'organisation et la structure des classes de l'interface. Puis, l'environnement utilisé pour développer l'interface est présenté. Ensuite, les éléments et fonctions de la fenêtre principale sont décrits. Selon le même format, l'implémentation des fenêtres secondaires est détaillée. Enfin, les conclusions et perspectives sont adressées.

I- Présentation générale

1 .1- Architecture du projet

Le rôle d'une interface est de permettre d'entrer des données de manière conviviale et intuitive. De plus, de cette façon les données sont plus robustes face aux incohérences grâce à l'affichage de messages d'erreur pour guider l'utilisateur.

L'interface principale se compose de plusieurs boutons qui permettent chacun d'ouvrir une interface secondaire. Dans notre cas, les variables entrées sous forme de chaînes de caractères dans les interfaces secondaires vont être stockées dans différentes instances de classe.

Pour la clarté du code, il a été décidé de créer une classe par interface secondaire (par exemple, pour la fenêtre 'ProblemDef' une intance de la classe "CproblemDefParameters" sera créée). Les instances de classe sont privées, ce qui permet de les protéger, et entre autre d'empêcher de modifier les attributs d'une instance par d'autres fonctions ou classes. Egalement, cela garantit que les valeurs au niveau de la fenêtre principale sont à jour en permanence et ne contiennent pas d'erreurs.
Le projet PAASoM C Sharp contient les éléments suivants :

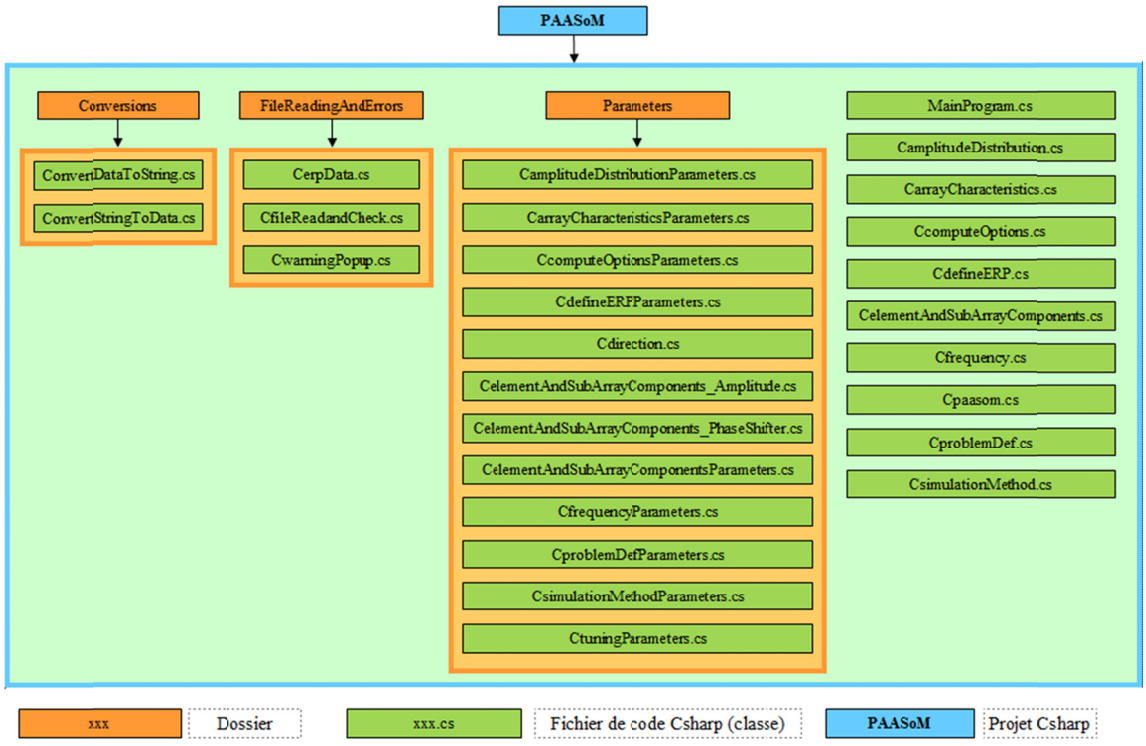


Figure 1 : fichiers contenus dans le projet PAASoM C Sharp

Chaque fichier d'extension « *.cs » est une classe. Chaque classe est détaillée dans la suite du document.
Directement sous la racine du projet, on trouve le programme principal (MainProgram.cs), qui est le point d'entrée de l'interface (c'est cette classe qui va lancer la fenêtre principale).
On y trouve également les 9 fenêtres composant actuellement l'interface graphique (la fenêtre principale, « Cpaasom.cs », et les 8 sous-fenêtres).

Ces 9 classes sont particulières : il s'agit de classes graphiques ou « Windows form ». Chacune de ces classes est en fait composée de trois fichiers (exemple avec la fenêtre « Amplitude distribution ») :

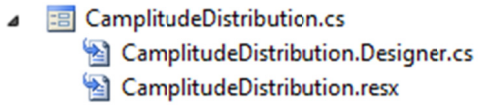


Figure 2 : fichiers composant une classe de type « Windows form »

Le fichier « xx.cs » est la classe elle-même, contenant le code pour la gestion des évènements de la fenêtre. Le fichier « xx.Designer.cs » contient la partie graphique de la fenêtre (coordonnées et caractéristiques de tous les élements tels que les boutons, les champs de saisie, etc.).
Le fichier « xx.resx » comprend les ressources des fenêtres telles que des images. Dans notre cas ce fichier contiendra uniquement l'objet « ToolTip » utilisé pour afficher des bulles d'aide.

1 .2- Architecture logicielle générale

Le schéma suivant présente les liens logiciels (appels) entre les différentes classes du programme :

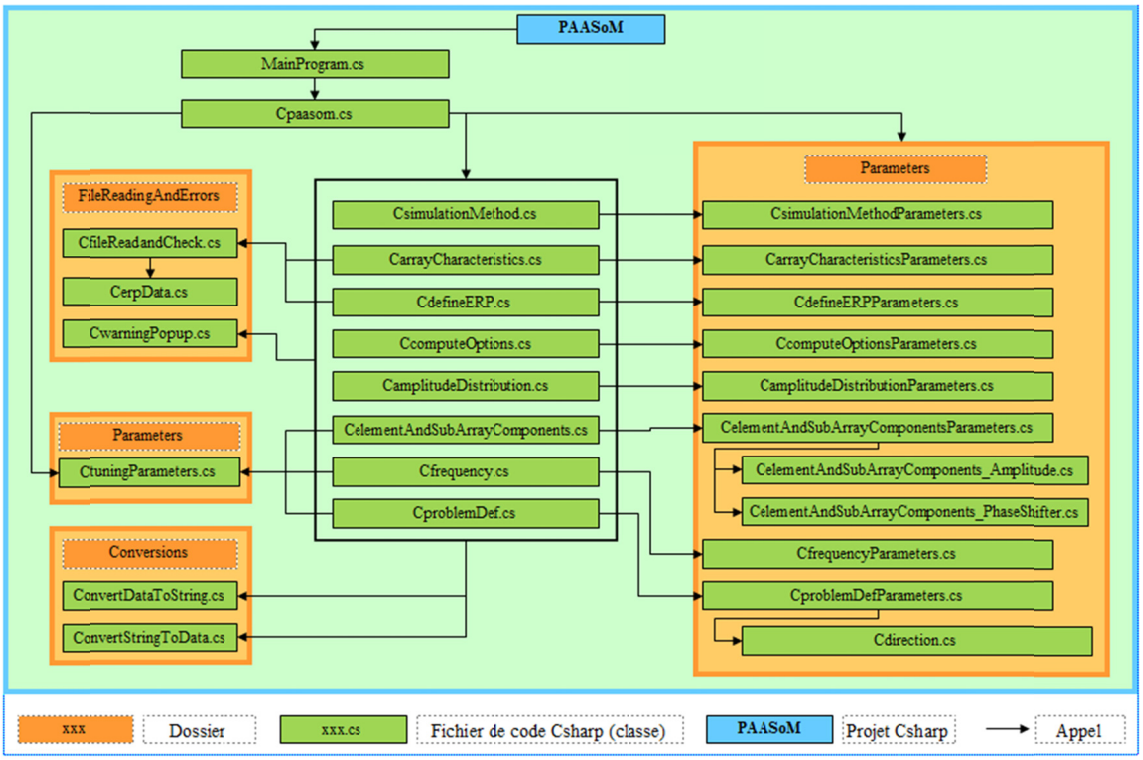


Figure 3 : architecture logicielle générale

Pour plus de lisibilité, Le rectangle noir englobe toutes les classes représentant les sous-fenêtres de l'interface. Ainsi, la classe « Cpaasom.cs » appelle toutes ces classes, et chacune de ces classes appelle les classes « ConvertDataToString.cs », « ConvertStringToData.cs » et « CwarningPopup.cs ».

Chaque sous-fenêtre (classes dans le rectangle noir) appelle un fichier de paramètres permettant de mémoriser le contenu des champs de saisie, les éléments sélectionnés dans les menus déroulants et l'état des boutons radio et des cases à cocher.
Ainsi, lorsque l'utilisateur appelle une nouvelle fois une fenêtre, il la retrouve dans l'état où elle était au moment de sa fermeture la fois précédente.

La classe « Cpaasom.cs » contient également une instance de chaque classe de paramètres, et une instance de la classe CtuningParameters.
Cette classe « CtuningParameters » est également appelée par les trois sous-fenêtres « Element and sub-array components », « Frequency » et « Problem definition », qui sont les trois fenêtres dans lesquelles peut être défini un tuning.

1.3- Conventions pour les noms de variables

1.3.1- Pour les contrôles

Les contrôles sont les éléments graphiques de l'interface (boutons, champs de saisie, etc.). Ils sont accessibles dans le programme par des pointeurs, c'est pourquoi on nommera ces pointeurs par un nom commençant par « p_ » suivi du type de contrôle (« Button », « CheckBox »,…).
Le nom complet pour ces variables sera donc du type « `p_type_nom` ». L'ensemble de ces noms sont précisés en annexe.
Voici les différents types de contrôles utilisés :

 Button	Bouton
 CheckBox	Case à cocher (choix multiples)
 ComboBox	Menu déroulant
 GroupBox	Groupement de contrôles
 Label	Texte simple
 RadioButton	Case à cocher (choix unique)
 TextBox	Champ de saisie
 ToolTip	« Bulle » contextuelle contenant un message

1.3.2- Pour les autres variables

Chaque variable a un nom permettant d'identifier son type : « `type_nom` ».

Type	Syntaxe C#	Nom de la variable (préfixe)
Entier	int	i
Réel	float	f
Caractère	char	c
Chaîne de caractères	string	str
Booléen	bool	b

De plus, un tableau sera considéré comme un vecteur, et un tableau de tableaux comme un vecteur de cellules (lien avec Matlab). Pour un tableau on ajoutera donc « **vec_** » devant le type de la variable, et pour un tableau de tableaux la chaîne « **vec_cell_** ».

1.4- Passage des variables

Lors de l'ouverture d'une interface secondaire, le mécanisme d'échange de modifications des variables et attributs et de mise à jour dans la fenêtre principale est le suivant :

1) L'instance de CuuParameters de la fenêtre principale (instance A) est envoyée vers la fenêtre secondaire. Cela permet d'initialiser l'affichage de la fenêtre.
2) L'instance A est copiée dans une instance B de CuuParameters appartenant à la fenêtre secondaire.
3) A chaque modification dans la fenêtre faite par l'utilisateur, des variables internes à la sous-fenêtre, regroupée sous forme de structure (struct_uuData) sont mises à jour.
4) Si l'utilisateur appuie sur le bouton « Ok », et qu'il n'y a pas d'erreur dans la fenêtre, l'instance B de CuuParameters est mise à jour.
5) La fenêtre secondaire se ferme, et l'instance A de CuuParameters prend les valeurs de l'instance B via une fonction du type « GetClassuuParameters » qui est appelée depuis la fenêtre principale.

Ceci est résumé sous forme schématique à la figure 4.

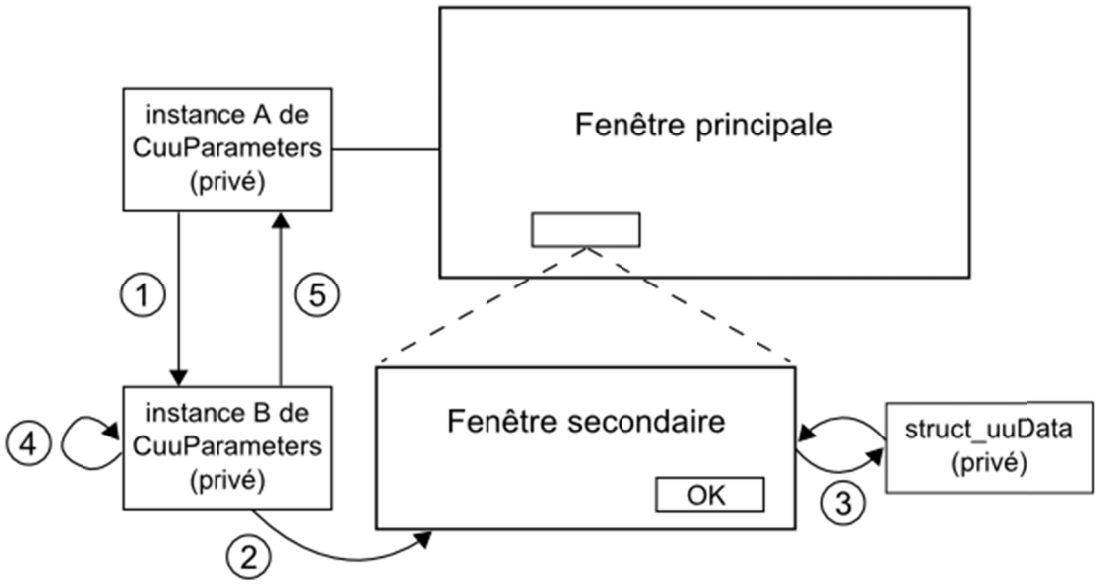


Figure 4 : Passage des variables dans l'interface PAASoM

La variable de type structure 'struct_uuData' contient les valeurs numériques des champs de saisie de la fenêtre secondaire en cours. Elle permet de vérifier la cohérence des données et l'affichage de messages d'erreur. Cette variable n'est pas sauvegardée au niveau de l'interface principale.

En procédant de cette manière, les variables sont protégées et les valeurs enregistrées au niveau de l'interface principales sont toujours correctes.

1.5- Structure des classes

1.5.1- Structure des classes de type « Windows form »

Les classes de fenêtres graphiques (windows form) contiendront toujours les éléments suivants, dans cet ordre :

- Entête de fichier,
- Attributs (variables locales à la classe) et structures de données,
- Initialisation (constructeur de la classe),
- Fonctions de sortie (mise à jour des paramètres dans la fenêtre principale),
- Evènements sur les contrôles de la fenêtre (boutons, champs de saisie,…),
- Fonctions pour l'affichage et la gestion de l'état du contenu de la fenêtre,
- Fonctions de vérification des valeurs entrées par l'utilisateur.

Chaque partie sera séparée par des commentaires. La structure de la classe sera donc :

Début du fichier : appel des librairies

```
//*****************************************************************************//
//                        Nom du fichier de fenêtre graphique                  //
//=============================================================================//
//  Author                 : Prénom Nom                                        //
//  Creation date          : mm/jj/aaaa                                        //
//  Last modification      : mm/jj/aaaa                                        //
//*****************************************************************************//
```

Début de la classe

```
        //=========================================================================//
        //                              ATTRIBUTES                                 //
        //=========================================================================//
```

Attributs

```
        //=========================================================================//
        //                              STRUCTURES                                 //
        //=========================================================================//
```

Structure(s)

```
        //=========================================================================//
        //                            INITIALIZATION                               //
        //=========================================================================//
```

Constructeur de la classe

```
        //=========================================================================//
        //                             OUTPUT METHODS                              //
        //=========================================================================//
```

Fonction(s) appelées dans le programme principal après la fermeture de la fenêtre

```
        //=========================================================================//
        //                            BUTTONS EVENTS                               //
        //=========================================================================//
        //=========================================================================//
        //                         CHECKBOXES EVENTS                               //
        //=========================================================================//
        //=========================================================================//
        //                               ...                                       //
        //=========================================================================//
```

Evènements sur les différents contrôles de la fenêtre, les fonctions étant classées par catégorie de contrôle

```
        //=========================================================================//
        //                      WINDOW STATE'S CONTROL METHODS                     //
        //=========================================================================//
```

Fonctions pour la gestion de l'affichage et de l'état de la fenêtre

```
        //=========================================================================//
        //                         CHECKING VALUES METHODS                         //
        //=========================================================================//
```

Fonctions de vérification des données entrées par l'utilisateur

1.5.2- Structure des classes de paramètres

Les classes de paramètres (classes présentes dans le dossier « Parameters » du projet) contiendront toujours les éléments suivants, dans cet ordre :

- Entête de fichier,
- Attributs (variables locales à la classe),
- Fonctions d'accès aux attributs depuis les autres classes,
- Constructeur de la classe.

Début du fichier : appel des librairies

```
//*****************************************************************************//
//                          Nom du fichier de paramètres                       //
//=============================================================================//
//  Author                : Prénom Nom                                         //
//  Creation date         : mm/jj/aaaa                                         //
//  Last modification     : mm/jj/aaaa                                         //
//*****************************************************************************//
```

Début de la classe

```
// Declare Local Attributes
```

Attributs de la classe

```
// Access to 'xxx' Attribute
// Access to 'xxx' Attribute
// Access to 'xxx' Attribute
// …
```

Fonctions d'accès aux attributs depuis les autres classes

```
//=============//
// Constructor //
//=============//
```

Constructeur de la classe

1.5.3- Structure des autres classes

Pour toutes les classes autres que celles décrites dans les deux parties précédentes, la structure sera la suivante :

- Entête de fichier,
- Attributs de la classe s'il y en a,
- Fonctions de la classes (méthodes).

Début du fichier : appel des librairies

```
//*****************************************************************************//
//                              Nom du fichier                                 //
//=============================================================================//
//  Author                : Prénom Nom                                         //
//  Creation date         : mm/jj/aaaa                                         //
//  Last modification     : mm/jj/aaaa                                         //
//*****************************************************************************//
```

Début de la classe

```
// Declare Local Attributes (optionnal)
```

Attributs

```
// Fonction 1
// Fonction 2
// …
```

Fonctions de la classe

1.6- Fonctions Matlab à convertir au niveau de l'interface

Quelques autres fonctions Matlab sont à intégrer dans l'interface, notamment au niveau de l'affichage de graphes.

Dans la fenêtre « Array characteristics » : l'espace vide en haut à droite de la fenêtre contiendra un graphe représentant la géométrie du réseau d'antennes, et sera mis à jour en fonction des actions de l'utilisateur.

Dans la fenêtre « Define ERP » : les trois boutons « Show ERP », « Show geometry » et « ERP Parameter(s) » n'ont pour l'instant aucune action. Chacun devra ouvrir une sous-fenêtre affichant la fonction Matlab correspondante (affichage des éléments du réseau avec la possibilité de les sélectionner, etc., de la même façon que dans l'interface Matlab actuelle).

Dans la fenêtre « Simulation method » : il s'agit de fonctions similaires à celles de la fenêtre « Define ERP », avec les trois boutons « Amplitude error », « Phase error » et « Show weights ».

1.7 L'environnement Visual Studio Express 2010

Pour le développement de l'interface en C#, l'environnement Visual Studio Express 2010 a été utilisé. Cet environnement permet de créer différents types de projets en C# (interface, classe, bibliothèque, etc.) et offre un puissant éditeur, un compilateur ainsi qu'un environnement d'exécution. Pour le développement de l'interface de PAASoM, les interfaces et les classes seront principalement utilisées. Un aperçu des principales étapes pour la création d'interfaces est donné dans cette partie.

La fenêtre de l'environnement Visual Studio Express 2010 est présentée à la Figure 5. Pour créer un nouveau projet, il suffit de cliquer sur « Nouveau projet … ». Une fenêtre s'ouvre ensuite et offre le choix du type de projet que l'on souhaite créer. Pour créer une interface, le choix doit être « Application Windows Forms », pour une classe « Projet vide ». Il est également possible de créer une application qui s'exécute dans une fenêtre DOS si « Application console » est sélectionné.

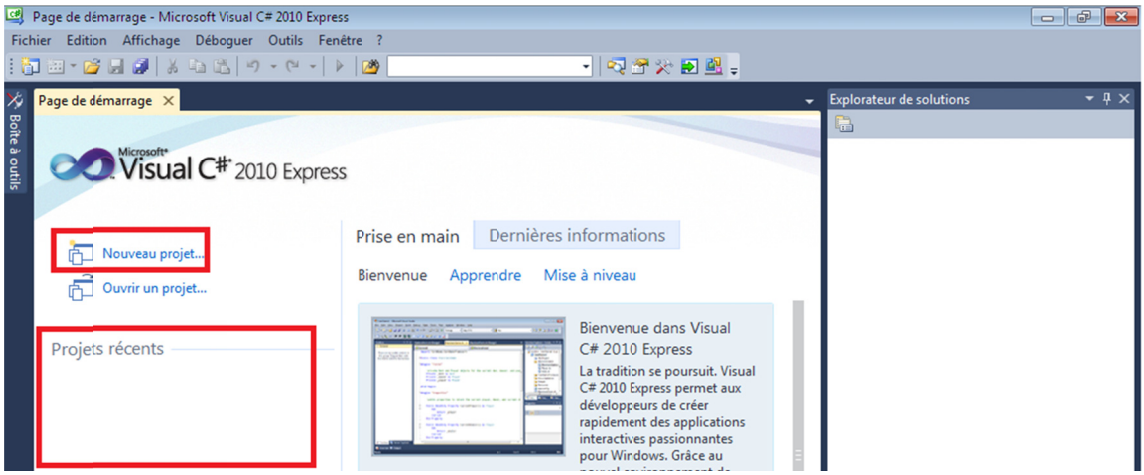


Figure 5 : L'environnement de Visual Studio Express 2010

Dans le cas où « Application Windows Forms » est choisi, Visual Studio crée une interface vide. Il est alors possible d'ajouter des composants graphiques dans cette interface en utilisant la boite à outils dans le panneau de gauche dans la fenêtre Visual Studio (Figure 6). Pour chaque composant graphique, il est possible d'accéder à ces propriétés dans le panneau de droite en cliquant sur l'icône « Propriétés ». C'est là que le nom du composant graphique est défini. C'est dans ce panneau également que sont définis les événements associés aux composants graphiques (par exemple, quelle fonction est exécutée quand on clique ou on double-clique sur un composant). Pour cela, il faut cliquer sur l'icône « Evénements » et définir le nom de la fonction qui sera appelée (Figure 7).

Le code C# associé aux composants graphiques est rassemblé dans une classe (Form1.cs dans l'exemple de la Figure 6). La localisation du code à exécuter lors de l'interaction avec l'interface peut-être trouvée en double-cliquant sur un composant graphique. Cela correspond aux fonctions définies lors de l'ajout d'événements.

Pour tester le projet en cours, il faut appuyer sur F5. Le débogueur permet de placer des points d'arrêt dans le code, et l'exploration des valeurs des variables est alors possible. Pour mettre un point d'arrêt, il faut se positionner sur la ligne où nous souhaitons nous arrêter, et cliquer dans la marge à gauche. Un point rouge s'affiche et indique qu'il y a un point d'arrêt à cet endroit. Pour visualiser les informations d'une variable, il suffit de passer la souris sur la variable considérée, et Visual Basic affiche alors une mini-information comprenant notamment la valeur de cette variable.

Enfin, la touche F10 permet d'exécuter le programme en mode pas à pas, et F11 donne la pile des appels de fonctions.

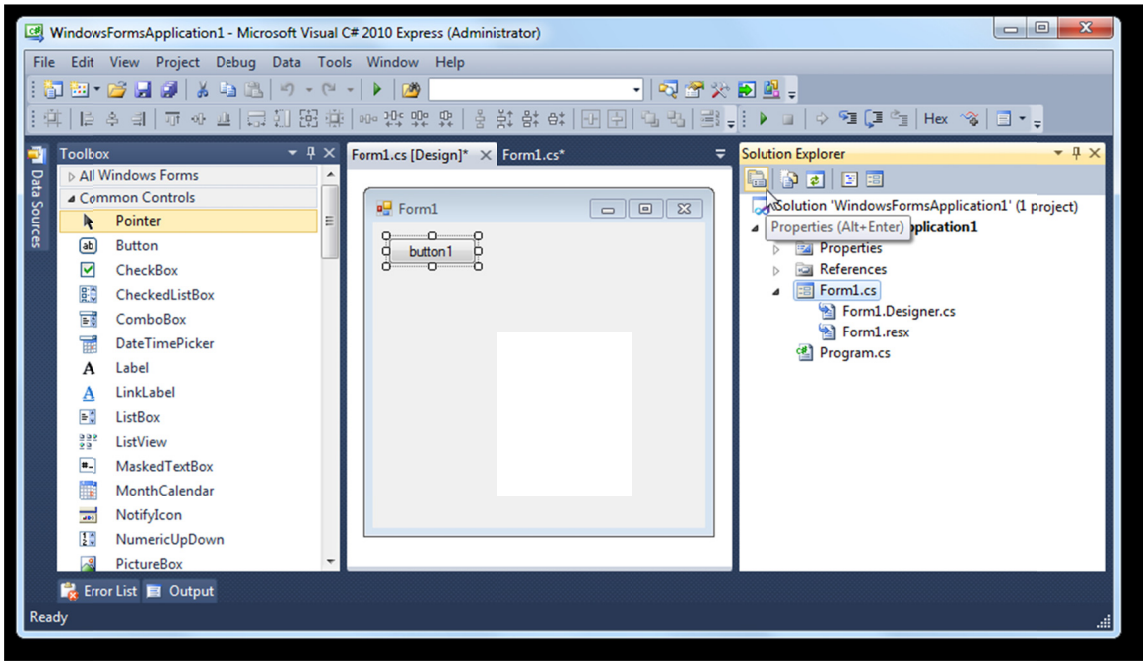


Figure 6 : Création d'interfaces avec Visual Studio Express 2010

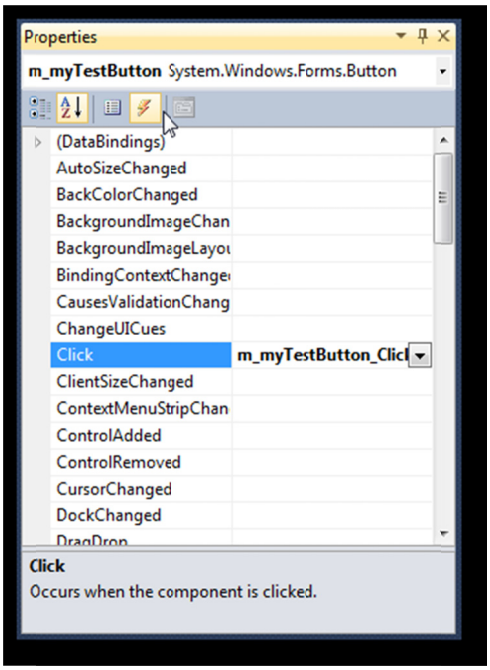


Figure 7 : Ajout d'événement à un composant graphique

Pour plus d'informations sur l'environnement Visual Studio et pour savoir comment développer des applications, rendez-vous à l'adresse suivante : http://msdn.microsoft.com/fr-fr/library/dd831853.aspx

Pour les informations de référence du langage C#, consultez la page internet http://msdn.microsoft.com/fr-ca/library/kx37x362.aspx

Enfin, un tutoriel assez complet sur le développement d'applications en C# est présenté à la page internet http://www.siteduzero.com/tutoriel-3-523498-apprendre-a-developper-en-c.html

II- La fenêtre principale

2.1- Présentation générale

La fenêtre principale se présente sous la forme suivante :

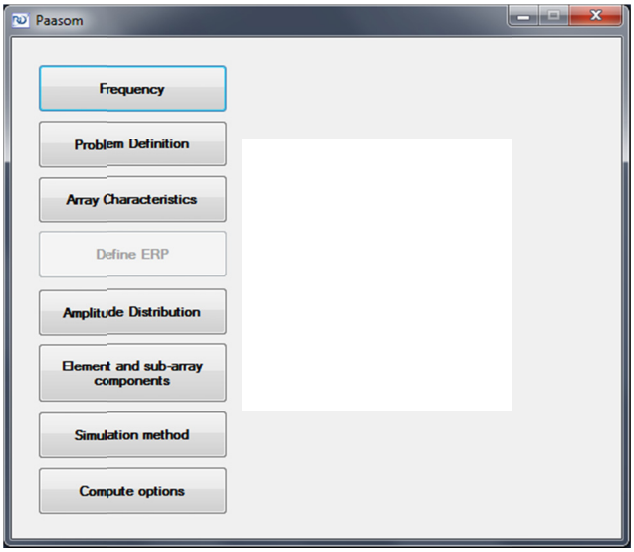


Figure 8 : Fenêtre principale

C'est à partir de cette interface que l'utilisateur pourra accéder à toutes les sous-fenêtres. L'ergonomie reste à améliorer, celle-ci est provisoire.

2.2- Pointeurs de la fenêtre « PAASoM »

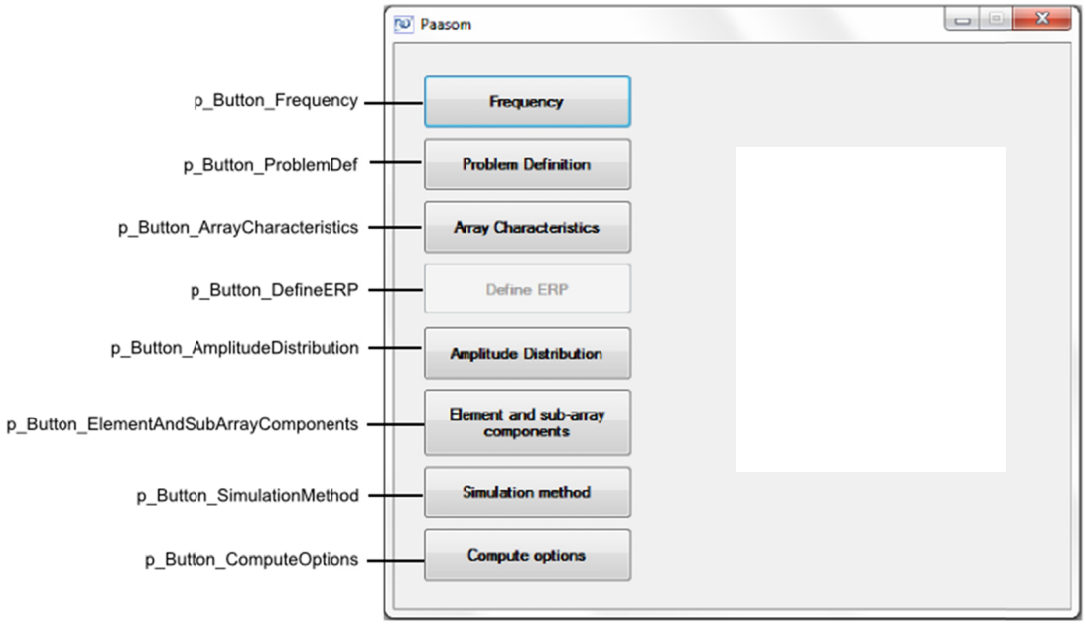


Figure 9 : Pointeurs de la fenêtre « PAASoM »

2.3- Méthodes de la classe « Cpassom »

Cpaasom (Constructeur)

Nom	`public Cpaasom()`
Description	Initialise les boutons de la fenêtre « PAASoM ».
Paramètres	Aucun.
Retour	Aucun.

p_butt_AmplitudeDistribution_Click

Nom	`private void p_butt_AmplitudeDistribution_Click(object sender, EventArgs e)`
Description	Fonction d'écoute de l'évènement : appui sur le bouton « Amplitude distribution ». Cette fonction ouvre la sous-fenêtre « Amplitude distribution » et se met en attente tant que l'utilisateur n'a pas fermé la sous-fenêtre. Si l'utilisateur ferme la fenêtre en appuyant sur « Ok », les variables de la classe « CamplitudeDistributionParameters » sont mises à jour.
Paramètres	Paramètres automatiques de capture d'évènement contenant l'objet (contrôle) concerné et les paramètres de l'évènement.
Retour	Aucun.

p_butt_ArrayCharacteristics_Click

Nom	`private void p_butt_ArrayCharacteristics_Click(object sender, EventArgs e)`
Description	Fonction d'écoute de l'évènement : appui sur le bouton « Array characteristics ». Cette fonction ouvre la sous-fenêtre « Array characteristics » et se met en attente tant que l'utilisateur n'a pas fermé la sous-fenêtre. Si l'utilisateur ferme la fenêtre en appuyant sur « Ok », les variables de la classe « CarrayCharacteristicsParameters » sont mises à jour.
Paramètres	Paramètres automatiques de capture d'évènement contenant l'objet (contrôle) concerné et les paramètres de l'évènement.
Retour	Aucun.

p_butt_ComputeOptions_Click

Nom	`private void p_butt_ComputeOptions_Click(object sender, EventArgs e)`
Description	Fonction d'écoute de l'évènement : appui sur le bouton « Compute options ». Cette fonction ouvre la sous-fenêtre « Compute options » et se met en attente tant que l'utilisateur n'a pas fermé la sous-fenêtre. Si l'utilisateur ferme la fenêtre en appuyant sur « Ok », les variables de la classe « CcomputeOptionsParameters » sont mises à jour.
Paramètres	Paramètres automatiques de capture d'évènement contenant l'objet (contrôle) concerné et les paramètres de l'évènement.
Retour	Aucun.

p_butt_DefineERP_Click

Nom	`private void p_butt_DefineERP_Click(object sender, EventArgs e)`
Description	Fonction d'écoute de l'évènement : appui sur le bouton « Define ERP». Cette fonction ouvre la sous-fenêtre « Define ERP » et se met en attente tant que l'utilisateur n'a pas fermé la sous-fenêtre. Si l'utilisateur ferme la fenêtre en appuyant sur « Ok », les variables de la classe « CdefineERPParameters » sont mises à jour.
Paramètres	Paramètres automatiques de capture d'évènement contenant l'objet (contrôle) concerné et les paramètres de l'évènement.
Retour	Aucun.

p_Button_ElementAndSubArrayComponents_Click

Nom	`private void p_Button_ElementAndSubArrayComponents_Click(object sender, EventArgs e)`
Description	Fonction d'écoute de l'évènement : appui sur le bouton « Elements and sub-array components ». Cette fonction ouvre la sous-fenêtre « Elements and sub-array components » et se met en attente tant que l'utilisateur n'a pas fermé la sous-fenêtre. Si l'utilisateur ferme la fenêtre en appuyant sur « Ok », les variables de la classe « CelementAndSubArrayComponentsParameters » sont mises à jour.
Paramètres	Paramètres automatiques de capture d'évènement contenant l'objet (contrôle) concerné et les paramètres de l'évènement.
Retour	Aucun.

p_butt_frequency_Click

Nom	`private void p_butt_frequency_Click(object sender, EventArgs e)`
Description	Fonction d'écoute de l'évènement : appui sur le bouton « Frequency ». Cette fonction ouvre la sous-fenêtre « Frequency » et se met en attente tant que l'utilisateur n'a pas fermé la sous-fenêtre. Si l'utilisateur ferme la fenêtre en appuyant sur « Ok », les variables de la classe « CfrequencyParameters » sont mises à jour.
Paramètres	Paramètres automatiques de capture d'évènement contenant l'objet (contrôle) concerné et les paramètres de l'évènement.
Retour	Aucun.

p_butt_ProblemDef_Click

Nom	`private void p_butt_Problemdef_Click(object sender, EventArgs e)`
Description	Fonction d'écoute de l'évènement : appui sur le bouton « Problem definition ». Cette fonction ouvre la sous-fenêtre « Problem definition » et se met en attente tant que l'utilisateur n'a pas fermé la sous-fenêtre. Si l'utilisateur ferme la fenêtre en appuyant sur « Ok », les variables de la classe « CproblemDefParameters » sont mises à jour.
Paramètres	Paramètres automatiques de capture d'évènement contenant l'objet (contrôle) concerné et les paramètres de l'évènement.
Retour	Aucun.

p_butt_SimulationMethod_Click

Nom	`private void p_butt_SimulationMethod_Click(object sender, EventArgs e)`
Description	Fonction d'écoute de l'évènement : appui sur le bouton « Simulation method ». Cette fonction ouvre la sous-fenêtre « Simulation method » et se met en attente tant que l'utilisateur n'a pas fermé la sous-fenêtre. Si l'utilisateur ferme la fenêtre en appuyant sur « Ok », les variables de la classe « CsimulationMethodParameters » sont mises à jour.
Paramètres	Paramètres automatiques de capture d'évènement contenant l'objet (contrôle) concerné et les paramètres de l'évènement.
Retour	Aucun.

III- La fenêtre « Frequency »

3.1- Présentation générale

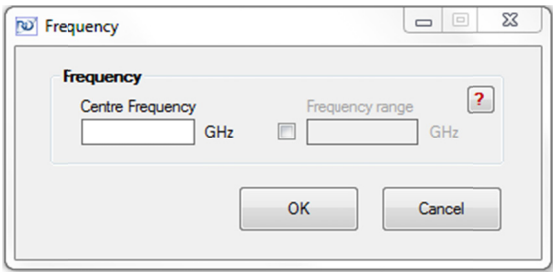


Figure 10 : La fenêtre « Frequency »

Cette fenêtre permet de définir la fréquence pour le projet en cours. On y trouve deux champs : « Centre Frequency » (fréquence centrale) et « Frequency Range » (plage de fréquence).

3.2- Fonctionnement de la fenêtre

3.2.1- Règles générales

Le champ de saisie pour la fréquence centrale sera nécessaire pour la suite du projet, il est donc obligatoire de le remplir.
Le champ « Frequency range » n'est pas obligatoire, c'est pourquoi une case à cocher se trouve devant le champ de saisie.
Si l'utilisateur veut remplir une valeur, il cochera cette case afin de rendre disponible la saisie. Si la case est cochée, le champ doit obligatoirement contenir une valeur.
Si l'utilisateur remplit la plage de fréquence mais pas la fréquence centrale, un message d'avertissement s'affichera lors de la fermeture de la fenêtre.
L'utilisateur aura donc la possibilité de fermer la fenêtre sans remplir la fréquence centrale tout de suite. Cependant il devra y revenir pour pouvoir continuer par la suite.

3.2.2- Champ « Centre frequency »

Type de variable : un seul nombre réel.
Nom de la variable dans la structure de données locale : f_CentreFrequency.
Nom du pointeur pour le contrôle : p_TextBox_CentreFrequency.
Contraintes particulières : la valeur doit être exclusivement positive (fréquence).

3.2.3- Champ « Frequency range »

Type de variable : tableau de nombres réels.
Nom de la variable dans la structure de données locale : vec_f_FrequencyRange.
Nom du pointeur pour le contrôle : p_TextBox_FrequencyRange.
Contraintes particulières : les valeurs doivent être exclusivement positives (fréquence). Elles peuvent être renseignées une à une en les séparant par des virgules (par exemple « 2,5,7,12 ») ou en définissant une plage de valeurs régulières avec un minimum, un maximum et éventuellement un pas (1 par défaut), par exemple « 2:5 » ou « 1:0.5:7 ». Il est possible de combiner les deux : par exemple « 1,2,3:9,20 » fonctionnera.

3.3- La classe « CfrequencyParameters »

Cette classe contient les variables de la fenêtre « Frequency » qui seront mémorisées dans la fenêtre principale.
Ces variables sont les suivantes :

```
private float m_f_CenterFrequency;
private float[] m_vec_f_FrequencyRange;
private string m_str_CenterFrequency;
private string m_str_FrequencyRange;
```

Il s'agit des données entrées par l'utilisateur dans les deux champs de saisie, sous forme de chaine de caractères et sous forme de données.
Dans les autres fenêtres, on ne sauvegardera les données que sous forme de chaînes de caractères.
Une exception est faite pour la fenêtre « Frequency » car ses valeurs sont souvent vérifiées dans les autres fenêtres (cela évite donc de reconvertir les chaînes de caractères en données à chaque fois).
Le constructeur de la classe « CfrequencyParameters » initialise ces différentes variables.

3.4- Pointeurs de la fenêtre « Frequency »

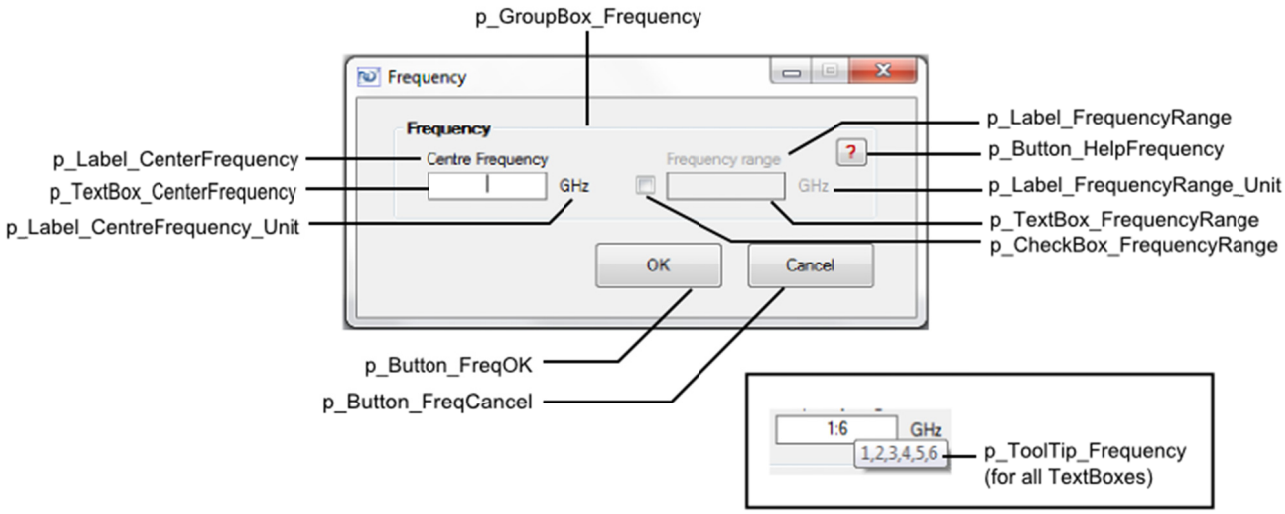


Figure 11 : pointeurs de la fenêtre «Frequency »

3.5- Méthodes de la classe « Cfrequency »

Cfrequency (Constructeur)

Nom	`public Cfrequency(string str_CentreFrequency_GUI, string str_FrequencyRange_GUI, CtuningParameter tunningParameter_GUI, CtuningParameter tunningParameterUseInPlottingOptions_GUI)`
Description	Initialise les contrôles et les variables de la fenêtre « Frequency ».
Paramètres	**`string str_CentreFrequency_GUI`:** chaîne à afficher dans le champ "Center frequency". **`string str_FrequencyRange_GUI`:** chaîne à afficher dans le champ "Frequency range". **`CtuningParameter tunningParameter_GUI`:** voir classe "CtuningParameter". Il s'agit du tuning défini pour le probleme courant. `CtuningParameter` **`tunningParameterUseInPlottingOptions`**`_GUI`: voir classe "CtuningParameter". Il s'agit du tuning en cours d'utilisation dans la fenêtre « Plotting options ».
Retour	Aucun.

Cfrequency_FormClosing

Nom	`private void Cfrequency_FormClosing(object sender, FormClosingEventArgs e)`
Description	Fonction d'écoute de l'évènement : fermeture de la fenêtre « Frequency » (appui sur le bouton « Ok », le bouton « cancel » ou la croix de fermeture en haut à droite). **Cas du bouton « Ok »** : si une erreur est présente sur la fenêtre, sa fermeture est annulée. Sinon, la fenêtre se ferme sans demander de confirmation à l'utilisateur. **Autres cas** : Une fenêtre de dialogue demande confirmation à l'utilisateur pour fermer la fenêtre ; la fenêtre se ferme donc ou non selon la réponse donnée.
Paramètres	Paramètres automatiques de capture d'évènement contenant l'objet (contrôle) concerné et les paramètres de l'évènement.
Retour	Aucun.

GetClassFrequencyParameters

Nom	`public CFrequencyParameters GetClassFrequencyParameters()`
Description	Retourne au GUI principal les valeurs de la fenêtre « Frequency ».
Paramètres	Aucun.
Retour	Classe `CfrequencyParameters` contenant les valeurs courantes.

GetClassTunningParameters

Nom	`public CtuningParameter GetClassTunningParameters()`
Description	Retourne au GUI principal les valeurs de tunning de la fenêtre « Frequency ».
Paramètres	Aucun.
Retour	Classe `CtuningParameter` contenant les valeurs courantes de tunning.

p_Button_FreqOK_Click

Nom	`private void p_Button_FreqOK_Click(object sender, EventArgs e)`
Description	Fonction d'écoute de l'évènement : appui sur le bouton « Ok ». Vérifie des valeurs de la fenêtre et met à jour la valeur booléenne qui annule la demande de confirmation à l'utilisateur pour fermer la fenêtre. Après l'appel de cette fonction, le programme appelle la fonction « `Cfrequency_FormClosing » (fenêtre modale).`
Paramètres	Paramètres automatiques de capture d'évènement contenant l'objet (contrôle) concerné et les paramètres de l'évènement.
Retour	Aucun.

p_CheckBox_FrequencyRange_CheckedChanged

Nom	`private void p_CheckBox_FrequencyRange_CheckedChanged(object sender, EventArgs e)`
Description	Fonction d'écoute de l'évènement : changement d'état (coché ou non) de la case à cocher devant le champ « Frequency range ». Change l'état (disponible, « enable », ou non disponible, « disable ») du champ de saisie « Frequency range » et des labels associés.
Paramètres	Paramètres automatiques de capture d'évènement contenant l'objet (contrôle) concerné et les paramètres de l'évènement.
Retour	Aucun.

p_TextBox _KeyPress

Nom	`private void p_TextBox _KeyPress(object sender, KeyPressEventArgs e)`
Description	Fonction d'écoute sur l'évènement : appui sur une touche lorsque le focus est sur un champ de saisie. Cette fonction ne fait rien si l'utilisateur a appuyé sur une autre touche que la touche « Entrée ». Vérifie la valeur entrée dans le champ de saisie. Si le contenu ne correspond pas aux contraintes, un message d'erreur apparaît et le focus reste sur le champ de saisie concerné.
Paramètres	Paramètres automatiques de capture d'évènement contenant l'objet (contrôle) concerné et les paramètres de l'évènement.
Retour	Aucun.

p_TextBox _Leave

Nom	`private void p_TextBox_Leave(object sender, EventArgs e)`
Description	Fonction d'écoute sur l'évènement : appui sur la touche « tabulation » ou clic sur un élément de la fenêtre alors que le focus est sur un champ de saisie. Vérifie la valeur entrée dans le champ de saisie. Si le contenu ne correspond pas aux contraintes, un message d'erreur apparaît et le focus reste sur le champ de saisie concerné.
Paramètres	Paramètres automatiques de capture d'évènement contenant l'objet (contrôle) concerné et les paramètres de l'évènement.
Retour	Aucun.

p_TextBox _MouseEnter

Nom	`private void p_TextBox_MouseEnter(object sender, EventArgs e)`
Description	Fonction d'écoute sur l'évènement : survol d'un champ de saisie avec la souris. Cette fonction affiche un contrôle « ToolTip » (bulles d'aide ou bulles de messages). Cette bulle contient la valeur entrée dans le champ de saisie (utile pour avoir une vue d'ensemble des valeurs lorsque la chaîne de caractère dépasse les dimensions du champ de saisie).
Paramètres	Paramètres automatiques de capture d'évènement contenant l'objet (contrôle) concerné et les paramètres de l'évènement.
Retour	Aucun.

UpdateCentreFrequency

Nom	`private string UpdateCentreFrequency()`
Description	Cette fonction vérifie la valeur entrée dans le champ « Centre frequency ». La chaine de caractères est transformée en un nombre réel. S'il ne s'agit pas d'un réel (erreur de syntaxe), si plusieurs valeurs sont entrées ou si la valeur est négative ou nulle, un message d'erreur sera généré.
Paramètres	Aucun.
Retour	Chaîne de caractère vide ou avec un message d'erreur.

UpdateFrequencyRange

Nom	`private string UpdateFrequencyRange()`
Description	Cette fonction vérifie la ou les valeur(s) entrée(s) dans le champ « Frequency range ». La chaine de caractères est transfornée en un tableau de réels. S'il ne s'agit pas de nombres réels (erreur de syntaxe) ou si au moins une des valeurs est négative ou nulle, un message d'erreur sera généré.
Paramètres	Aucun.
Retour	Chaîne de caractère vide ou avec un message d'erreur.

IV- Fenêtre « Problem Definition »

4.1 – Présentation générale

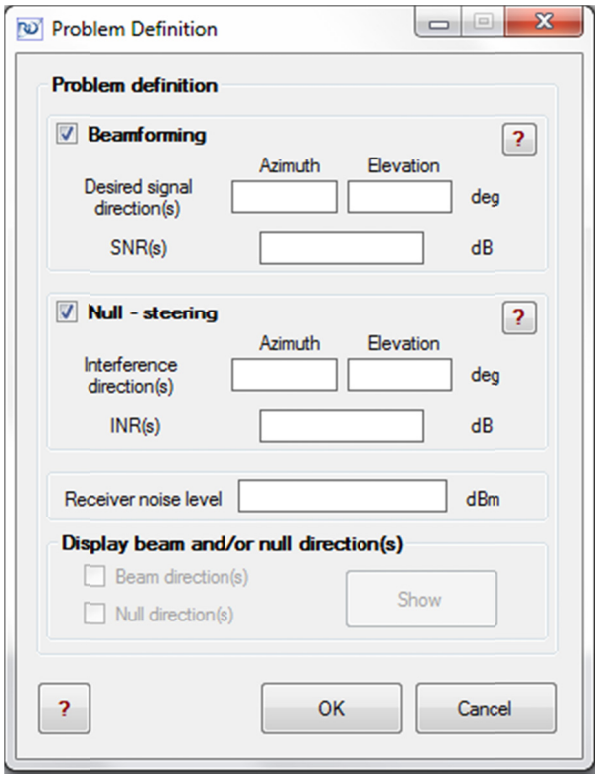


Figure 12 : Fenêtre « Problem definition »

Cette fenêtre permet de définir les directions souhaitées pour la synthèse du facteur de r éseau. On peut y préciser les directions des faisceaux (« Beamforming ») et des nuls (« Null-steering »). Il est possible de visualiser le résultat afin d'éviter les erreurs.

4.2 – Fonctionnement de la fenêtre

4.2.1 - Règles générales

La fenêtre « Problem Definition » contient 4 sous-ensembles : « **Beamforming** », « **Null-Steering** », « **Noise level** » et « **Display directions** ».

Groupes « **Beamforming** » et « **Null-Steering** » :
Ils fonctionnent de façon similaire. Pour activer les champs de saisie et pouvoir entrer des valeurs, il faut cocher la case « Beamforming » ou « Null-Steering ».
Il n'est pas possible de définir un tuning (plage de valeurs) à la fois pour « Beamforming » et « Null-Steering ».
Un champ peut contenir une seule direction, ou plusieurs à condition que le nombre de directions soit le même ou égal à 1 pour les autres champs du groupe.
De même, si un tuning est défini, il doit être défini pour une seule direction être cohérent avec les autres champs du groupe.

Groupe « **Noise level** »:
Le champ de saisie « receiver noise level » sera activé si :
-Le champ « SNRs » dans « Beamforming » contient au moins une valeur,
-La case à cocher « Null-Steering » est cochée.
Une fois le champ « Noise Level » activé il est obligatoire de remplir une valeur.

Dans « **Display directions** », la case à cocher « Beam direction(s) » s'active si au moins une direction est définie dans « Beamforming ». Le principe est le même pour « Null-Steering ».
Si au moins une des deux cases « display directions » est cochée, le bouton « Show » deviendra actif et il sera possible de visualiser les directions.

4.2.2 – Champ « Azimuth » Beamforming

Type de variable : Tableau de tableaux de nombres réels.
Nom de la variable dans la structure de données locale : vec_cell_f_AzimuthBeamForming.
Nom du pointeur pour le contrôle : p_TextBox_AzimuthBeamForming.
Contraintes particulières : il s'agit d'angle(s) en degrés. Les différentes directions doivent être séparées par des barre obliques « / ». Les tunings peuvent être une suite de valeurs séparées par des virgules ou une plage de valeur en utilisant le symbole « : » (ou un mélange des deux).
Exemple : « 10/20:0.5:50/75 » est valide.
Le nombre d'angles entré doit être cohérent avec celui du champ « Elevation » beamforming et avec le nombre de valeurs du champ « SNR(s) » beamforming.

4.2.3 – Champ « Elevation » Beamforming

Type de variable : Tableau de tableaux de nombres réels.
Nom de la variable dans la structure de données locale : vec_cell_f_ElevationBeamForming.
Nom du pointeur pour le contrôle : p_TextBox_ElevationBeamForming.
Contraintes particulières : mêmes contraintes que pour le champ « Azimuth » beamforming.
Le nombre d'angles entré doit être cohérent avec celui du champ « Azimuth » beamforming et avec le nombre de valeurs du champ « SNR(s) » beamforming.

4.2.4 – Champ « SNR(s) » Beamforming

Type de variable : Tableau de tableaux de nombres réels.
Nom de la variable dans la structure de données locale : vec_cell_f_SNRBeamForming.
Nom du pointeur pour le contrôle : p_TextBox_SNRS.
Contraintes particulières : mêmes contraintes que pour le champ « Azimuth » beamforming.
Le nombre de valeurs spécifiées doit être cohérent avec le nombre d'angles des champs « Azimuth » beamforming et « Elevation » beamforming.

4.2.5 – Champ « Azimuth » Null-Steering

Type de variable : Tableau de tableaux de nombres réels.
Nom de la variable dans la structure de données locale : vec_cell_f_AzimuthNullSteering.
Nom du pointeur pour le contrôle : p_TextBox_AzimuthNullSteering.
Contraintes particulières : mêmes contraintes que pour le champ « Azimuth » beamforming.

4.2.6 – Champ « Elevation » Null-Steering

Type de variable : Tableau de tableaux de nombres réels.
Nom de la variable dans la structure de données locale : vec_cell_f_ElevationNullSteering.
Nom du pointeur pour le contrôle : p_TextBox_ElevationNullSteering.
Contraintes particulières : mêmes contraintes que pour le champ « Elevation » beamforming.

4.2.7 – Champ « INR(s) » Null-Steering

Type de variable : Tableau de tableaux de nombres réels.
Nom de la variable dans la structure de données locale : vec_cell_f_INRNullSteering.
Nom du pointeur pour le contrôle : p_TextBox_INRS.
Contraintes particulières : mêmes contraintes que pour le champ « SNR(s) » beamforming.

4.2.8 – Champ « Receiver noise level »

Type de variable : Une seule valeur réelle.
Nom de la variable dans la structure de données locale : f_ NoiseLevel.
Nom du pointeur pour le contrôle : p_TextBox_NoiseLevel.
Contraintes particulières : Aucune.

4.3- La classe « CproblemDefParameters »

Cette classe contient les variables de la fenêtre « Problem definition » qui seront mémorisées dans la fenêtre principale.
Ces variables sont les suivantes :

```
private Cdirection m_direction_BeamFormingValues;
private Cdirection m_direction_NullSteeringValues;
private string m_str_NoiseLevel;
private bool m_b_BeamDisplayCheckBoxState;
private bool m_b_NullDisplayCheckBoxState;
private string[] m_vec_str_SynthesisMethod;
```

Les deux premières variables sont des instances de la classe “Cdirection”, détaillée dans la partie suivante.
« m_str_NoiseLevel » est la chaîne de caractères dans le champ de saisie « receiver noise level ».
Les deux valeurs booléennes correspondent à l’état des deux cases à cocher « Display beam directions » et « Display null directions ».
« `m_vec_str_SynthesisMethod` » contient les éléments visibles dans le menu déroulant de la fenêtre « Simulation method ». En effet la(les) méthode(s) pouvant être utilisées dépendent de la configuration de définition du problème (nombre de directions, valeurs de null-steering ou non, valeurs de beamforming ou non).

Les valeurs des cases à cocher « Beamforming » et « Null-steering » ne sont pas sauvegardées ; lors de l’initialisation de la fenêtre « Problem definition », on vérifie simplement si le champ « Azimuth » est vide ou non. En effet si le champ est vide alors la case à cocher correspondante est forcément décochée.

4.4- La classe « Cdirection »

Cette classe est utilisée dans la classe « CproblemDefParameters ». Elle contient les attributs suivants :

```
private string m_str_Azimuth;
private string m_str_Elevation;
private string m_str_SNR;
```

Il s'agit de chaines de caractères correspondant aux champs de saisie des groupes "Beamforming" et "Null-steering" de la fenêtre « Problem definition ».
C'est pourquoi la classe « CproblemDefParameters » contient deux instances de la classe « Cdirection » : une pour le groupe « Beamforming » et une pour le groupe « Numm-steering ».

4.5- Pointeurs de la fenêtre « Problem definition »

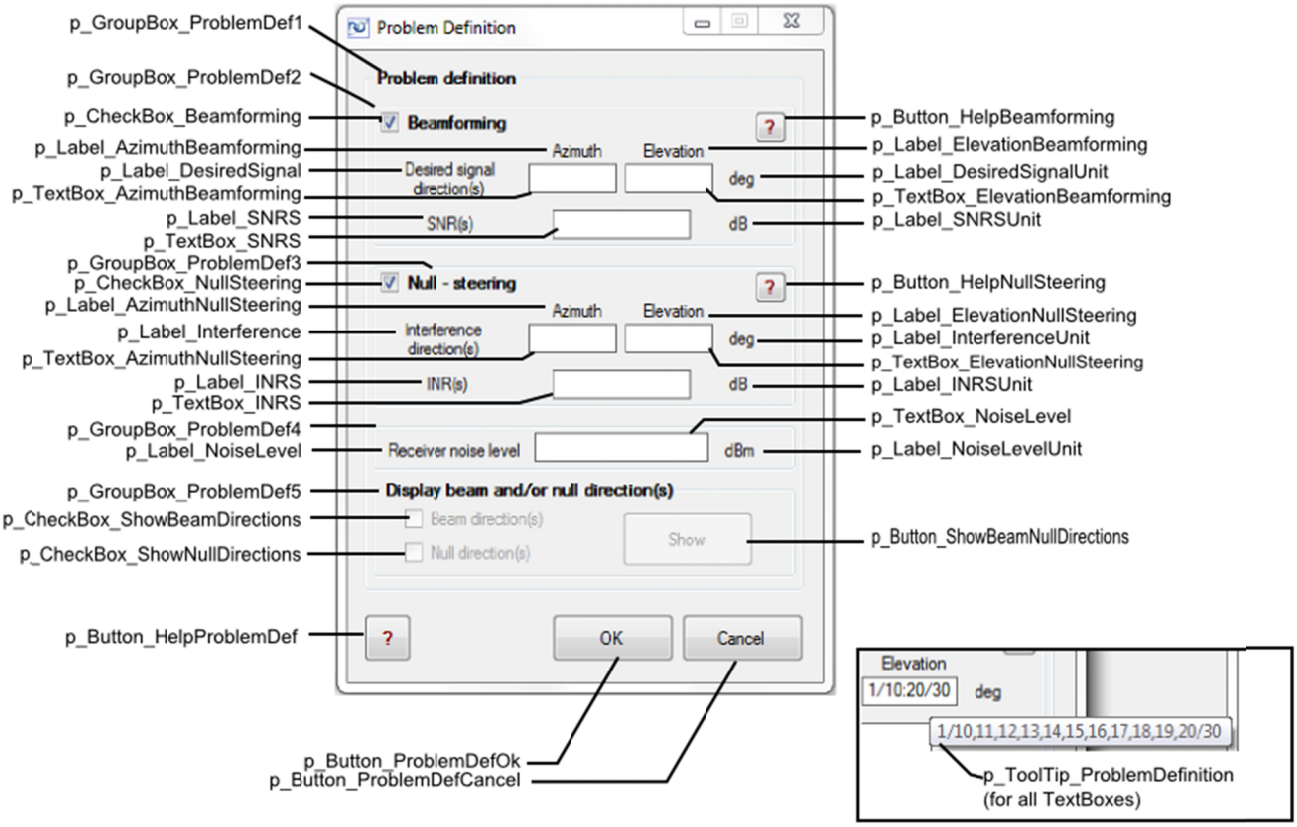


Figure 13 : pointeurs de la fenêtre « Problem definition »

4.6– Méthodes de la classe « CproblemDef »

BeamFormingAzimuth

Nom	`private string BeamFormingAzimuth()`
Description	Vérifie les valeurs entrées dans le champ « Beamforming azimuth » et met à jour les variables associées (ainsi que le tuning si défini).
Paramètres	Aucun.
Retour	Chaîne de caractères vide ou contenant un message d'erreur.

BeamFormingElevation

Nom	`private string BeamFormingElevation()`
Description	Vérifie les valeurs entrées dans le champ « Beamforming elevation » et met à jour les variables associées (ainsi que le tuning si défini).
Paramètres	Aucun.
Retour	Chaîne de caractères vide ou contenant un message d'erreur.

CheckAngleValues

Nom	`private string CheckAngleValues(bool b_IsBeamForming)`
Description	Vérifie que le nombre de directions des groupes « Beamforming » et « Null-steering » est cohérent.
Paramètres	**`bool b_IsBeamForming`**`:indique si l'on se trouve dans le cas "Beamforming" ou "Null-steering".`
Retour	Chaîne de caractère vide ou avec un message d'erreur.

CheckTuningValues

Nom	`private string CheckTuningValues(float[][] vec_cell_f_valuesToCheck, string str_TuningName, TextBox TextBox_Concerned, int i_CurrentIndex)`
Description	Vérifie la validité des valeurs de tuning (longueur, plage de définition des directions, cohérence entre les différents champs de saisie).
Paramètres	**`float[][] vec_cell_f_valuesToCheck:`** tableau de tableaux de réels correspondant aux valeurs entrées dans le champ de saisie à vérifier. **`string str_TuningName:`** Nom du tuning associé au champ de saisie (valeur de la clé pour l'objet « Dictionnary » qui gère les valeurs de tuning dans la classe CtuningParameters). La valeur peut être : «Azimuth (Beamforming)», «Elevation (Beamforming)», «SNR (Beamforming)», «Azimuth (Null-steering)», «Elevation (Null-steering)», «INR (Null-steering)». **`TextBox TextBox_Concerned:`** Référence du contrôle TextBox dont on veut vérifier le contenu (pointeur). **`int i_CurrentIndex:`** Entier compris entre 0 et 5 selon le champ de saisie à vérifier : 0 pour « Azimuth beamforming », 1 pour « Elevation beamforming », 2 pour « SNR(s) beamforming », 3 pour « Azimuth null-steering », 4 pour « Elevation null-steering », 5 pour « INR(s) null-steering ».
Retour	Chaîne de caractère vide ou avec un message d'erreur.

CproblemDef (Constructeur)

Nom	`public CproblemDef(CproblemDefParameters ProblemDefParameters_GUI, CtuningParameter tuningParameter_GUI, CtuningParameter tunningParameterUseInPlottingOptions_GUI)`
Description	Initialise les contrôles et les variables de la fenêtre « Problem definition ».
Paramètres	**`CproblemDefParameters ProblemDefParameters_GUI:`** Valeurs de tous les paramètres de la fenêtre « Problem definition » gardés en mémoire dans la fenêtre principale. **`CtuningParameter tunningParameter_GUI:`** voir classe "CtuningParameter". Il s'agit du tuning défini pour le probleme courant. `CtuningParameter tunningParameterUseInPlottingOptions_GUI:` voir classe "CtuningParameter". Valeurs actuelles du tuning utilisé dans la fenêtre « Plotting options ».
Retour	Aucun.

CproblemDef_FormClosing

Nom	`private void CproblemDef_FormClosing(object sender, FormClosingEventArgs e)`
Description	Fonction d'écoute de l'évènement : fermeture de la fenêtre « Problem definition » (appui sur le bouton « Ok », le bouton « cancel » ou la croix de fermeture en haut à droite). **Cas du bouton « Ok »** : si une erreur est présente sur la fenêtre, sa fermeture est annulée. Sinon, la fenêtre se ferme sans demander de confirmation à l'utilisateur. **Autres cas** : Une fenêtre de dialogue demande confirmation à l'utilisateur pour fermer la fenêtre ; la fenêtre se ferme donc ou non selon la réponse donnée.
Paramètres	Paramètres automatiques de capture d'évènement contenant l'objet (contrôle) concerné et les paramètres de l'évènement.
Retour	Aucun.

ExtractTuningName

Nom	`private void ExtractTuningName()`
Description	Détermine le nom du tuning (par exemple « Azimuth & Elevation (Beamforming) ») et les valeurs qui lui sont associées.
Paramètres	Aucun.
Retour	Aucun.

ExtractTuningString

Nom	`private string ExtractTuningString(string str_FullValues)`
Description	Extrait uniquement les valeurs de tuning dans le texte d'un champ de saisie. Par exemple si le champ de saisie contient «10/20,22,25/30», la fonction renverra «20,22,25».
Paramètres	**`string str_FullValues:`** `chaîne de caractères don't on veut extraire les valeurs de tuning.`
Retour	Chaîne de caractères contenant uniquement les valeurs de tuning.

GetClassProblemDefParameters

Nom	`public CproblemDefParameters GetClassProblemDefParameters()`
Description	Retourne au GUI principal les valeurs de la fenêtre « Problem definition ».
Paramètres	Aucun.
Retour	Classe `CproblemDefParameters contenant les valeurs des paramètres.`

GetClassTuningParameters

Nom	`public CtuningParameter GetClassTuningParameters()`
Description	Retourne au GUI principal les valeurs de tunning de la fenêtre « Problem defintion » (si défini, sinon le tuning n'est pas modifié).
Paramètres	Aucun.
Retour	Classe `CtuningParameters contenant les valeurs des paramètres de tuning.`

INRs

Nom	`private string INRs()`
Description	Vérifie les valeurs entrées dans le champ « Null-steering INR(s) » et met à jour les variables associées (ainsi que le tuning si défini).
Paramètres	Aucun.
Retour	Chaîne de caractères vide ou contenant un message d'erreur.

NoiseLevel

Nom	`private string NoiseLevel()`
Description	Vérifie les valeurs entrées dans le champ « Receiver noise level » et met à jour les variables associées.
Paramètres	Aucun.
Retour	Chaîne de caractères vide ou contenant un message d'erreur.

NullSteeringAzimuth

Nom	`private string NullSteeringAzimuth()`
Description	Vérifie les valeurs entrées dans le champ « Null-steering azimuth » et met à jour les variables associées (ainsi que le tuning si défini).
Paramètres	Aucun.
Retour	Chaîne de caractères vide ou contenant un message d'erreur.

NullSteeringElevation

Nom	`private string NullSteeringelevation()`
Description	Vérifie les valeurs entrées dans le champ « Null-steering elevation » et met à jour les variables associées (ainsi que le tuning si défini). Fonction appelée dans « p_TextBox_Leave » et « p_TextBox_KeyPress ».
Paramètres	Aucun.
Retour	Chaîne de caractères vide ou contenant un message d'erreur.

OkPushButton

Nom	`private string OkPushButton()`
Description	Fonction appelée dans `p_Button_ProblemDefOk_Click`, c'est-à-dire lorsque l'utilisateur a appuyé sur le bouton « Ok ». Vérifie que les champs obligatoires ne sont pas vides. Si le tuning a été changé et qu'il était utilisé dans la fenêtre « Plotting options », un message d'avertissement apparaît pour informer l'utilisateur que la fenêtre « Plotting options » a été réinitialisée. Suivant la configuration du problème, la méthode de synthèse est mise à jour (choix dans le menu déroulant de la fenêtre « Simulation method »).
Paramètres	Aucun.
Retour	Chaîne de caractères vide ou contenant un message d'erreur.

p_Button_ProblemDefOk_Click

Nom	`private void p_Button_ProblemDefOk_Click(object sender, EventArgs e)`
Description	Fonction d'écoute de l'évènement : appui sur le bouton « Ok ». Vérifie les valeurs de la fenêtre et met à jour la valeur booléenne qui annule la demande de confirmation à l'utilisateur pour fermer la fenêtre. Après l'appel de cette fonction, le programme appelle la fonction « `CproblemDef_FormClosing » (fenêtre modale).`
Paramètres	Paramètres automatiques de capture d'évènement contenant l'objet (contrôle) concerné et les paramètres de l'évènement.
Retour	Aucun.

p_CheckBox_CheckedChanged

Nom	`private void p_CheckBox_CheckedChanged(object sender, EventArgs e)`
Description	Fonction d'écoute de l'évènement : changement d'état (coché ou non) des contrôles « CheckBox » de la fenêtre « Problem definition ». **-CheckBox « BeamForming »** : gère l'affichage des contrôles pour le groupe « Beamforming » (désactivé ou activé). **-CheckBox « Null_steering »** : gère l'affichage des contrôles pour le groupe « Null_steering » (désactivé ou activé). **-CheckBox « Display Beam directions » et « Display Null directions »** : gère l'état (actif ou non) du bouton « Show ».
Paramètres	Paramètres automatiques de capture d'évènement contenant l'objet (contrôle) concerné et les paramètres de l'évènement.
Retour	Aucun.

p_TextBox_KeyPress

Nom	`private void p_TextBox_KeyPress(object sender, KeyPressEventArgs e)`
Description	Fonction d'écoute sur l'évènement : appui sur une touche alors que le focus est sur un champ de saisie. Cette fonction ne fait rien si l'utilisateur a appuyé sur une autre touche que la touche « Entrée ». Vérifie le contenu du champ de saisie. Si le contenu ne correspond pas aux contraintes, affichage d'un message d'erreur sous forme de fenêtre « pop-up ».
Paramètres	Paramètres automatiques de capture d'évènement contenant l'objet (contrôle) concerné et les paramètres de l'évènement.
Retour	Aucun.

p_TextBox_Leave

Nom	`private void p_TextBox_Leave(object sender, EventArgs e)`
Description	Fonction d'écoute sur l'évènement : appui sur la touche « tabulation » ou clic sur un élément de la fenêtre alors que le focus est sur un champ de saisie. Vérifie le contenu du champ de saisie. Si le contenu ne correspond pas aux contraintes, un message d'erreur apparaît et le focus reste sur le champ de saisie concerné.
Paramètres	Paramètres automatiques de capture d'évènement contenant l'objet (contrôle) concerné et les paramètres de l'évènement.
Retour	Aucun.

p_TextBox_MouseEnter

Nom	`private void p_TextBox_MouseEnter(object sender, EventArgs e)`
Description	Fonction d'écoute sur l'évènement : passage de la souris au dessus d'un champ de saisie. Si le champ de saisie contient une chaîne valide, une « bulle » apparaîtra avec le détail du contenu de ce champ de saisie. Par exemple, si un champ contient la chaîne « 1:5 », la bulle affichera « 1,2,3,4,5». S'il s'agit d'un fichier, c'est le contenu du fichier qui sera affiché.
Paramètres	Paramètres automatiques de capture d'évènement contenant l'objet (contrôle) concerné et les paramètres de l'évènement.
Retour	Aucun.

setBeamFormGroupBoxState

Nom	`private void setBeamFormGroupBoxState(bool b_BeamFormGroupBoxState)`
Description	Active ou désactive les éléments du groupe « Beamforming ».
Paramètres	`bool b_BeamFormGroupBoxState: état souhaité pour le groupe « Beamforming ».`
Retour	Aucun.

setNoiseLevelGroupBoxState

Nom	`private void setNoiseLevelGroupBoxState(bool b_NoiseLevelGroupBoxState)`
Description	Active ou désactive les éléments du groupe « Noise level ».
Paramètres	`bool b_NoiseLevelGroupBoxState: état souhaité pour le groupe « Noise level».`
Retour	Aucun.

setNullSteeringGroupBoxState

Nom	`private void setNullSteeringGroupBoxState(bool b_NullSteeringGroupBoxState)`
Description	Active ou désactive les éléments du groupe « Null-steering ».
Paramètres	`bool b_NullSteeringGroupBoxState: état souhaité pour le groupe « Null-steering ».`
Retour	Aucun.

ShowButtonDirections

Nom	`private void ShowButtonDirections()`
Description	Active ou désactive le bouton « Show » selon l'état des cases à cocher « Display beam directions » et « Display null directions ».
Paramètres	Aucun.
Retour	Aucun.

SNRs

Nom	`private string SNRs()`
Description	Vérifie les valeurs entrées dans le champ « Beamforming SNR(s) » et met à jour les variables associées (ainsi que le tuning si défini). Fonction appelée dans « p_TextBox_Leave » et « p_TextBox_KeyPress ».
Paramètres	Aucun.
Retour	Chaîne de caractères vide ou contenant un message d'erreur.

UpperOrEqualToZero

Nom	`private static bool UpperOrEqualToZero(int i_IntToCheck)`
Description	Fonction utilisée par « `Array.FindIndex` » dans la fonction « `CheckTuningValues` ». « `Array.FindIndex` » permet de repérer les éléments d'un tableau correspondant à un certain critère. Dans le cas présent, on souhaite savoir si un élément du tableau à une valeur supérieure ou égale à 0 (présence d'un tuning). C'est donc cette fonction (`UpperOrEqualToZero`) qui permet de vérifier cette contrainte.
Paramètres	`int i_IntToCheck:` `Nombre entier à vérifier (négatif ou non).`
Retour	Valeur booléenne vraie si le nombre entier est supérieur ou égal à 0, fausse sinon.

V- La fenêtre « Array characteristics »

5.1- Présentation générale

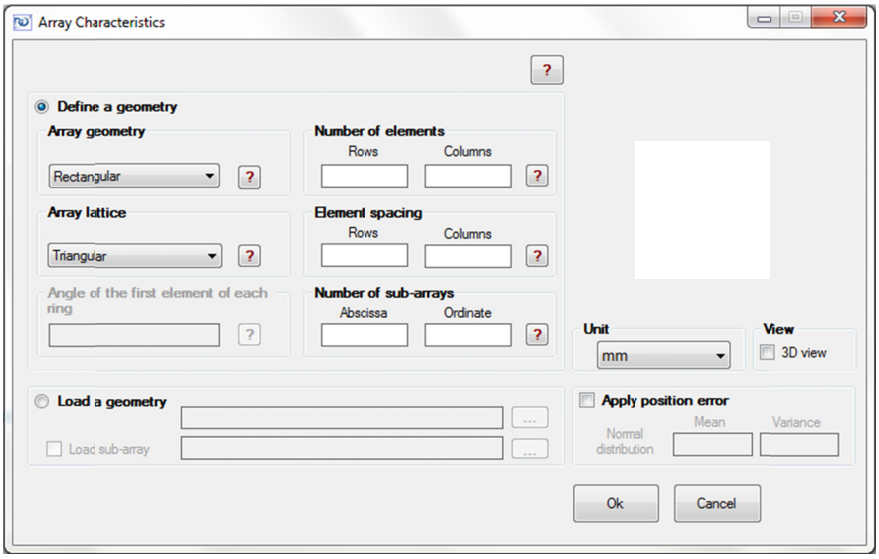


Figure 14 : La fenêtre « Array characteristics »

Cette fenêtre permet de définir la géométrie du réseau d'antennes ainsi que des sous-réseaux. La géométrie et les sous-réseaux peuvent être définis manuellement ou être chargés depuis un fichier.

5.2- Fonctionnement de la fenêtre

5.2.1- Règles générales

La partie vide en haut à droite de la fenêtre contiendra un graphique Matlab permettant de visualiser la géométrie.
La case à cocher « 3D view » permet de choisir entre un affichage 2D ou 3D.
L'unité d'affichage peut également être choisie avec le menu déroulant « Unit ». Si une fréquence centrale a été définie dans la fenetre « Frequency », l'unité « longueur d'onde » (**λ**) s'ajoutera à la liste.

Cadre « Define a geometry » :
Dans cette partie, la géométrie du réseau peut être définie manuellement. Pour l'activer il faut cocher le bouton radio « Define a geometry ». Six formes géométriques sont proposées dans le menu déroulant « Array geometry » : linéaire, rectangulaire, hexagonale, triangulaire, circulaire et octogonale.
Le menu déroulant « Array lattice » permet de choisir le type de « treillis » : rectangulaire ou triangulaire.
Ce choix ne peut se faire que pour les géométries triangulaire, rectangulaire et hexagonale.
En fonction de la géométrie choisie, les titres des champs « Number of elements », « Element spacing » et « Number of sub-arrays » changent. Certains de ces champs peuvent être cachés si il ne s'appliquent pas à la géométrie sélectionnée.

Le champ « Angle of the first element of each ring » ne devient actif que pour la géométrie circulaire. Il s'agit de l'angle de départ du premier point de chaque cercle.

Cadre « Load a geometry » :
Dans cette partie, la géométrie et les sous-réseaux peuvent être importés depuis deux fichiers contenant respectivement les coordonnées et le numéro de sous-réseau pour chaque point du réseau.
On peut importer un fichier en ouvrant un navigateur avec le bouton « … » ou en écrivant directement son nom dans le champ de saisie. Un fichier de géométrie valide doit être importé pour pouvoir importer également un fichier pour les valeurs de sous-réseau.

Cadre « Apply position error » :
Pour activer ce groupe il faut cocher la case « Apply position error ». Si la case est cochée, les deux champs « Error mean » et « Error Variance » doivent contenir une valeur.

5.2.2- Champ « Number of elements – abscissa »

Type de variable : un seul nombre entier.
Nom de la variable dans la structure de données locale : `i_NumberOfElements_Abs`.
Nom du pointeur pour le contrôle : p_TextBox_NumberOfElementsAbscissa.
Contraintes particulières :
- En mode « circulaire » : la valeur doit être supérieure à 0.
- En mode « triangulaire » : si le treillis (« Lattice ») est rectangulaire, la valeur doit être supérieure à 2. Le champ « Number of elements – ordinate » ne peut pas valoir plus que deux fois la valeur du champ « Number of elements – abscissa ». Si le treillis est « triangulaire », le champ « Number of elements – abscissa » doit être inférieur au champ « Number of elements – ordinate ».
- Autres géométries : la valeur doit être supérieure à 1.
- Le nombre de valeurs dans les champs « Angle of the first element of each ring » et « Element spacing – abscissa » change en fonction de la valeur de « Number of elements – abscissa », afin que tout soit cohérent.

5.2.3- Champ « Number of elements – ordinate »

Type de variable : tableau de nombres entiers.
Nom de la variable dans la structure de données locale : `vec_i_NumberOfElements_Ord`.
Nom du pointeur pour le contrôle : p_TextBox_NumberOfElementsOrdinate.
Contraintes particulières :
- En mode « circulaire » : c'est le seul mode ou le champ peut contenir plusieurs valeurs. Il doit contenir soit une seule valeur, soit le nombre indiqué dans le champ « Number of elements – abscissa ». Les valeurs doivent êtres supérieures à 0.
- En mode « triangulaire » : voir les contraintes de « Number of elments – abscissa ». La valeur doit êtres supérieure à 1.
- En mode « rectangulaire » : la valeur doit être supérieure à 1.
- Autres géométries : la valeur doit être supérieure à 0.

5.2.4- Champ « Element spacing – abscissa »

Type de variable : tableau de nombres réels.
Nom de la variable dans la structure de données locale : `vec_f_ElementSpacing_Abs`.
Nom du pointeur pour le contrôle : p_TextBox_ElementSpacingAbscissa.
Contraintes particulières :
- En mode « circulaire » : les valeurs doivent être strictement positives sauf la première valeur qui peut être nulle. Il doit y a avoir soit une seule valeur soit le même nombre que celui indiqué dans le champ « Circles » (Number of elements – abscissa).
- En mode « octogonal » : les valeurs peuvent êtres égales à 0. Il doit y avoir soit une seule valeur soit 3*(« Number of elements – abscissa » - 1) valeurs.

- Autres géométries : les valeurs doivent être strictement positives. Il doit y avoir soit une seule valeur soit « Number of elements – abscissa » - 1 valeurs.

5.2.5- Champ « Element spacing – ordinate »

Type de variable : tableau de nombres réels.
Nom de la variable dans la structure de données locale : vec_f_ElementSpacing_Ord.
Nom du pointeur pour le contrôle : p_TextBox_ElementSpacingOrdinate.
Contraintes particulières :
Dans tous les cas, il ne doit y avoir que des valeurs strictement positives.
- En mode « circulaire » : Il doit y a avoir soit une seule valeur soit le même nombre que celui indiqué dans le champ « Circles » (Number of elements – abscissa).
- En mode « octogonal » : Il doit y avoir soit une seule valeur soit 3*(« Number of elements – abscissa » - 1) valeurs.
- Autres géométries : Il doit y avoir soit une seule valeur soit « Number of elements – abscissa » - 1 valeurs.

5.2.6- Champ « Number of sub-arrays – abscissa »

Type de variable : un seul nombre entier.
Nom de la variable dans la structure de données locale : i_NumberOfSubArrays_Abs.
Nom du pointeur pour le contrôle : p_TextBox_NumberOfSubArraysAbscissa.
Contraintes particulières : La valeur doit être strictement positive.

5.2.7- Champ « Number of sub-arrays – ordinate »

Type de variable : un seul nombre entier.
Nom de la variable dans la structure de données locale : i_NumberOfSubArrays_Ord.
Nom du pointeur pour le contrôle : p_TextBox_NumberOfSubArraysOrdinate.
Contraintes particulières : La valeur doit être strictement positive.

5.2.8- Champ « Angle of the first element of each ring »

Type de variable : tableau de nombres réels.
Nom de la variable dans la structure de données locale : vec_f_AngleFirstelementRing.
Nom du pointeur pour le contrôle : p_TextBox_AngleFirstElementRing.
Contraintes particulières :
Ce champ n'est disponible que dans la configuration « circulaire ». La valeur par défaut est 0.
On ne peut pas mettre d'autre valeur tant que le champ « Number of elements – abscissa » n'est pas renseigné. Il doit alors y avor soit une seule valeur soit le nombre de valeurs indiqué dans le champ « Number of elements – abscissa ».

5.2.9- Champ « Geometry file name »

Type de variable : tableau de tableaux de nombres réels.
Nom de la variable dans la structure de données locale : vec_Position.
Nom du pointeur pour le contrôle : p_TextBox_GeometryFileName.
Contraintes particulières : Le champ en lui-même doit contenir un nom de fichier valide. La variable associé à ce champ de saisie contient les données du fichier. Chaque ligne du fichier doit contenir trois nombres réels (position selon les axes X, Y et Z) séparés par un espace.

5.2.10- Champ « Sub-array file name »

Type de variable : tableau de nombres entiers.
Nom de la variable dans la structure de données locale : vec_i_SubArray.
Nom du pointeur pour le contrôle : p_TextBox_GeometryFileName.
Contraintes particulières : Le champ en lui-même doit contenir un nom de fichier valide. La variable associé à ce champ de saisie contient les données du fichier. Chaque ligne du fichier doit contenir un seul nombre entier.

5.2.11- Champ « Mean »

Type de variable : un seul nombre réel.
Nom de la variable dans la structure de données locale : f_Mean.
Nom du pointeur pour le contrôle : p_TextBox_NormalLawMean.
Contraintes particulières : Aucune.

5.2.12- Champ « Variance »

Type de variable : un seul nombre réel.
Nom de la variable dans la structure de données locale : f_Variance.
Nom du pointeur pour le contrôle : p_TextBox_NormalLawVariance.
Contraintes particulières : Aucune.

5.3- La classe « CarrayCharacteristicsParameters »

Cette classe contient les variables de la fenêtre « Array characteristics » qui seront mémorisées dans la fenêtre principale.
Ces variables sont les suivantes :

```
private string m_str_ArrayGeometry;
private string m_str_ArrayLattice;
private string m_str_AngleOfFirstElementRing;
private string m_str_NumberOfElementsAbscissa;
private string m_str_NumberOfElementsOrdinate;
private string m_str_SpacingElementAbscissa;
private string m_str_SpacingElementOrdinate;
private string m_str_NumberOfSubArraysAbscissa;
private string m_str_NumberOfSubArraysOrdinate;
private string m_str_GeometryFileName;
private string m_str_SubArrayFileName;
private string m_str_ErrorMean;
private string m_str_ErrorVariance;
private string m_str_Unit;
private bool m_b_3DView;
public float[][] m_vec_Position;
```

Il s'agit du contenu de tous les champs de saisie, des éléments sélectionnés dans les menus déroulants et de l'état de la case à cocher « 3D view » (la mémorisation des autres cases à cocher et boutons radio n'est pas nécéssaire, on peut retrouver leur état précédent en regardant quels champs sont vides ou non).
Enfin le tableau de tableaux m_vec_Position contient les coordonnées de chaque élement du réseau.

5.4- Pointeurs de la fenêtre « Array characteristics »

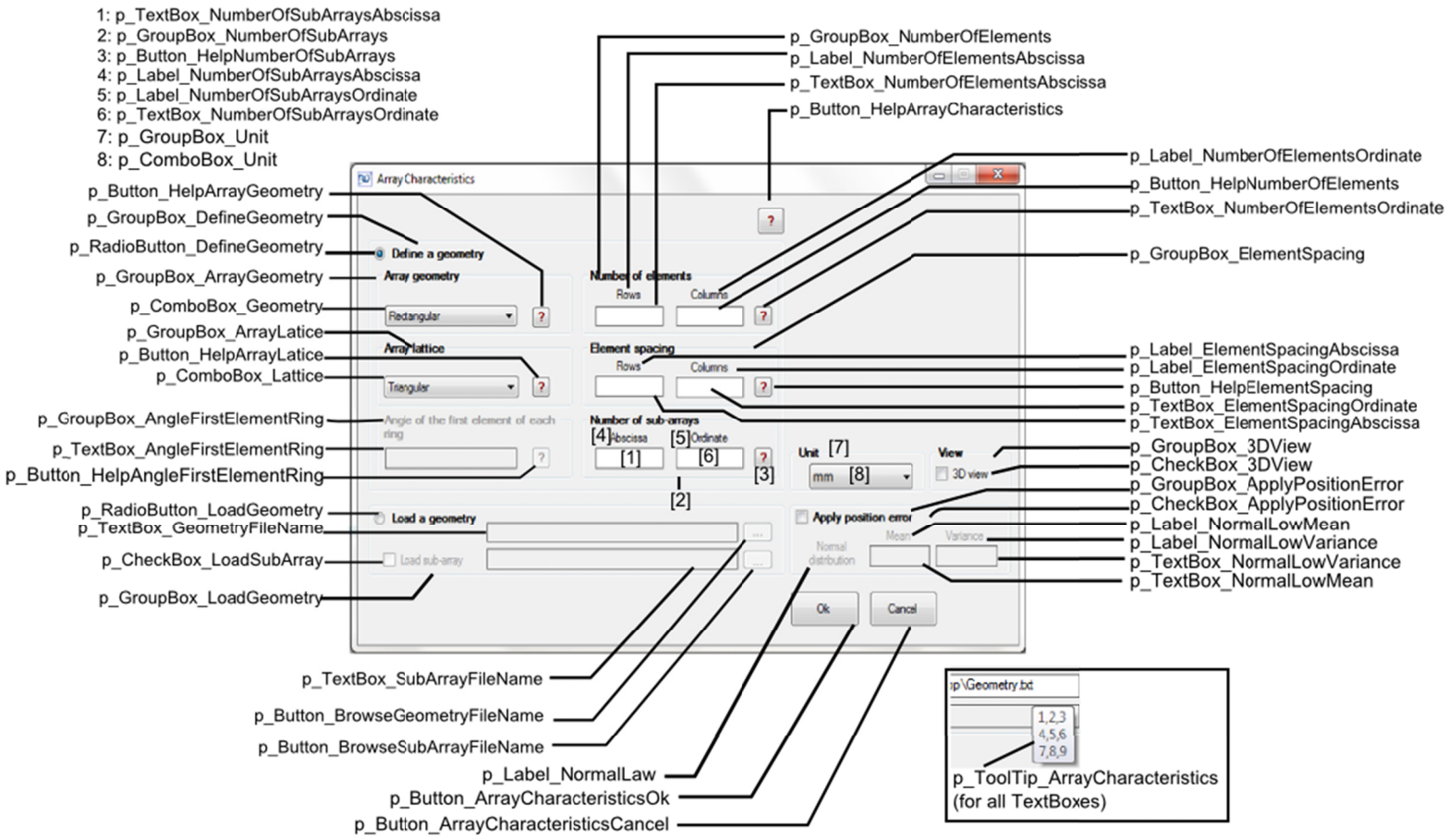


Figure 15 : pointeurs de la fenêtre « Array characteristics »

5.5- Méthodes de la classe « CarrayCharacteristics »

AngleFirstElementRing

Nom	`private string AngleFirstElementRing()`
Description	Vérifie le contenu du champ « Angle of the first element of each ring ».
Paramètres	Aucun.
Retour	Chaîne de caractères vide ou contenant un message d'erreur.

ApplyPositionError

Nom	`private void ApplyPositionError()`
Description	Gère l'affichage du groupe « Apply position error » en fonction de l'état de la case à cocher « Apply position error ».
Paramètres	Aucun.
Retour	Aucun.

CarrayCharacteristics (Constructeur)

Nom	`public CarrayCharacteristics(CarrayCharacteristicsParameters arrayCharacParameters_GUI, float f_CentreFrequency_GUI)`
Description	Initialise les contrôles et les variables de la fenêtre « Array characteristics ».

Paramètres	**`CarrayCharacteristicsParameters arrayCharacParameters_GUI:`**
	`Valeurs de tous les paramètres de la fenêtre « Array characteristics » gardés en mémoire dans la fenêtre principale.`
	
	`float f_CentreFrequency_GUI:`
	`Fréquence centrale définie dans la fenêtre « Frequency ». Cette variable sert ici à savoir si l'unité « lambda » (longueur d'onde) doit être ajoutée au menu déroulant « Unit » ou non.`

Retour	Aucun.

CarrayCharacteristics_FormClosing

Nom	`private void CarrayCharacteristics_FormClosing(object sender, FormClosingEventArgs e)`
Description	Fonction d'écoute de l'évènement : fermeture de la fenêtre « Array characteristics » (appui sur le bouton « Ok », le bouton « cancel » ou la croix de fermeture en haut à droite). **Cas du bouton « Ok »** : si une erreur est présente sur la fenêtre, sa fermeture est annulée. Sinon, la fenêtre se ferme sans demander de confirmation à l'utilisateur. **Autres cas** : Une fenêtre de dialogue demande confirmation à l'utilisateur pour fermer la fenêtre ; la fenêtre se ferme donc ou non selon la réponse donnée.
Paramètres	Paramètres automatiques de capture d'évènement contenant l'objet (contrôle) concerné et les paramètres de l'évènement.
Retour	Aucun.

DisplayDefineGeometry

Nom	`private void DisplayDefineGeometry(bool b_DefineGeometryGroupBoxState)`
Description	Active ou désactive les éléments du groupe « Define geometry ».
Paramètres	**`bool b_DefineGeometryGroupBoxState:`** Valeurs booléenne selon l'état souhaité pour le groupe "Define geometry" (true: actif, false: inactif).
Retour	Aucun.

DisplayLoadGeometry

Nom	`private void DisplayLoadGeometry(bool b_DefineLoadGroupBoxState)`
Description	Active ou désactive les éléments du groupe « Load geometry ».
Paramètres	**`bool b_DefineLoadGroupBoxState:`** Valeurs booléenne selon l'état souhaité pour le groupe "Load geometry" (true: actif, false: inactif).
Retour	Aucun.

DisplayNormalLaw

Nom	`private void DisplayNormalLaw(bool b_DefineErrorGroupBoxState)`
Description	Active ou désactive les éléments du groupe « Apply position error ».
Paramètres	**`bool b_DefineErrorGroupBoxState:`** Valeurs booléenne selon l'état souhaité pour le groupe " Apply position error " (true: actif, false: inactif).
Retour	Aucun.

ElementSpacing_Abs

Nom	`private string ElementSpacing_Abs()`
Description	Vérifie le contenu du champ « Element spacing - abscissa », ainsi que la cohérence avec les valeurs des autres champs.
Paramètres	Aucun.
Retour	Chaîne de caractères vide ou contenant un message d'erreur.

ElementSpacing _Ord

Nom	`private string ElementSpacing_Ord()`
Description	Vérifie le contenu du champ « Element spacing - ordinate », ainsi que la cohérence avec les valeurs des autres champs.
Paramètres	Aucun.
Retour	Chaîne de caractères vide ou contenant un message d'erreur.

Geometry

Nom	`private void Geometry()`
Description	Gère l'affichage des champs de saisie en fonction du type de géométrie choisie dans le menu déroulant « Geometry ».
Paramètres	Aucun.
Retour	Aucun.

GetClassArrayCaracteristicsParameters

Nom	`public CarrayCharacteristicsParameters GetClassArrayCaracteristicsParameters()`
Description	Retourne au GUI principal les valeurs de la fenêtre «Array characteristics ».
Paramètres	Aucun.
Retour	Classe `CarrayCharacteristicsParameters contenant les valeurs des paramètres.`

LoadSubArray

Nom	`private void LoadSubArray()`
Description	Gère l'affichage du champ de saisie « Load sub-array » en fonction de l'état de la case à cocher « Load sub-array ».
Paramètres	Aucun.
Retour	Aucun.

NormalLawMean

Nom	`private string NormalLawMean()`
Description	Vérifie le contenu du champ « Mean ».
Paramètres	Aucun.
Retour	Chaîne de caractères vide ou contenant un message d'erreur.

NormalLawVariance

Nom	`private string NormalLawVariance()`
Description	Vérifie le contenu du champ « Variance ».
Paramètres	Aucun.
Retour	Chaîne de caractères vide ou contenant un message d'erreur.

NumberOfElements_Abs

Nom	`private string NumberOfElements_Abs()`
Description	Vérifie le contenu du champ « Number of elements - abscissa », ainsi que la cohérence avec les valeurs des autres champs.
Paramètres	Aucun.
Retour	Chaîne de caractères vide ou contenant un message d'erreur.

NumberOfElements_Ord

Nom	`private string NumberOfElements_Ord()`
Description	Vérifie le contenu du champ « Number of elements - ordinate », ainsi que la cohérence avec les valeurs des autres champs.
Paramètres	Aucun.
Retour	Chaîne de caractères vide ou contenant un message d'erreur.

NumberOfSubArrays_Abs

Nom	`private string NumberOfSubArrays_Abs()`
Description	Vérifie le contenu du champ « Number of sub-arrays - abscissa », ainsi que la cohérence avec les valeurs des autres champs.
Paramètres	Aucun.
Retour	Chaîne de caractères vide ou contenant un message d'erreur.

NumberOfSubArrays_Ord

Nom	`private string NumberOfArrays_Ord()`
Description	Vérifie le contenu du champ « Number of sub-arrays - ordinate », ainsi que la cohérence avec les valeurs des autres champs.
Paramètres	Aucun.
Retour	Chaîne de caractères vide ou contenant un message d'erreur.

OkPushButton

Nom	`private string OkPushButton()`
Description	Fonction appelée dans `p_Button_ProblemDefOk_Click`, c'est-à-dire lorsque l'utilisateur a appuyé sur le bouton « Ok ». Vérifie que les champs devant être remplis ne sont pas vides.
Paramètres	Aucun.
Retour	Chaîne de caractères vide ou contenant un message d'erreur.

p_Button_ArrayCharacteristicsOk_Click

Nom	`private void p_Button_ArrayCharacteristicsOk_Click(object sender, EventArgs e)`
Description	Fonction d'écoute de l'évènement : appui sur le bouton « Ok ». Vérifie des valeurs de la fenêtre et met à jour la valeur booléenne qui annule la demande de confirmation à l'utilisateur pour fermer la fenêtre. Après l'appel de cette fonction, le programme appelle la fonction « `CarrayCharacteristics_FormClosing » (fenêtre modale).`
Paramètres	Paramètres automatiques de capture d'évènement contenant l'objet (contrôle) concerné et les paramètres de l'évènement.
Retour	Aucun.

p_Button_BrowseFile_Click

Nom	`private void p_Button_BrowseFile_Click(object sender, EventArgs e)`
Description	Fonction d'écoute de l'évènement : appui sur l'un des deux boutons « … » de sélection de fichier. Ouvre un explorateur de fichiers. Lorsque l'utilisateur à sélectionné un fichier, son nom s'affiche dans le champ de saisie correspondant et son contenu est vérifié et chargé en mémoire.
Paramètres	Paramètres automatiques de capture d'évènement contenant l'objet (contrôle) concerné et les paramètres de l'évènement.
Retour	Aucun.

p_CheckBox_CheckedChanged

Nom	`private void p_CheckBox_CheckedChanged(object sender, EventArgs e)`
Description	Fonction d'écoute de l'évènement : changement d'état de l'une des cases à cocher. **- Cas « Load sub-array »** : Active ou désactive le champ de saisie pour le nom du fichier de sous-réseaux. **- Cas « Apply position error »** : Active ou désactive le groupe « Apply position error ».
Paramètres	Paramètres automatiques de capture d'évènement contenant l'objet (contrôle) concerné et les paramètres de l'évènement.
Retour	Aucun.

p_ComboBox_SelectedIndexChanged

Nom	`private void p_ComboBox_SelectedIndexChanged(object sender, EventArgs e)`
Description	Fonction d'écoute de l'évènement : changement dans la sélection de l'un des menus déroulants. **- Cas « Geometry »** : Adapte l'affichage selon la géométrie choisie. Autres menus : pas d'action pour l'instant, par la suite un changement dans ces menus mettra à jour le graphe représentant le réseau.
Paramètres	Paramètres automatiques de capture d'évènement contenant l'objet (contrôle) concerné et les paramètres de l'évènement.
Retour	Aucun.

p_TextBox_KeyPress

Nom	`private void p_TextBox_KeyPress(object sender, KeyPressEventArgs e)`
Description	Fonction d'écoute sur l'évènement : appui sur une touche alors que le focus est sur un champ de saisie. Cette fonction ne fait rien si l'utilisateur a appuyé sur une autre touche que la touche « Entrée ». Vérifie le contenu du champ de saisie. Si le contenu ne correspond pas aux contraintes, affichage d'un message d'erreur sous forme de fenêtre « pop-up ».
Paramètres	Paramètres automatiques de capture d'évènement contenant l'objet (contrôle) concerné et les paramètres de l'évènement.
Retour	Aucun.

p_TextBox_Leave

Nom	`private void p_TextBox_Leave(object sender, EventArgs e)`
Description	Fonction d'écoute sur l'évènement : appui sur la touche « tabulation » ou clic sur un élément de la fenêtre alors que le focus est sur un champ de saisie. Vérifie le contenu du champ de saisie. Si le contenu ne correspond pas aux contraintes, un message d'erreur apparaît et le focus reste sur le champ de saisie concerné.
Paramètres	Paramètres automatiques de capture d'évènement contenant l'objet (contrôle) concerné et les paramètres de l'évènement.
Retour	Aucun.

p_TextBox_MouseEnter

Nom	`private void p_TextBox_MouseEnter(object sender, EventArgs e)`
Description	Fonction d'écoute sur l'évènement : passage de la souris au dessus d'un champ de saisie. Si le champ de saisie contient une chaîne valide, une « bulle » apparaîtra avec le détail du contenu de ce champ de saisie. Par exemple, si un champ contient la chaîne « 1:5 », la bulle affichera « 1,2,3,4,5». S'il s'agit d'un fichier, c'est le contenu du fichier qui sera affiché.
Paramètres	Paramètres automatiques de capture d'évènement contenant l'objet (contrôle) concerné et les paramètres de l'évènement.
Retour	Aucun.

p_RadioButton_CheckedChanged

Nom	`private void p_RadioButton_CheckedChanged(object sender, EventArgs e)`
Description	Fonction d'écoute de l'évènement : changement d'état de l'un des boutons radios. Active ou désactive les parties « Define geometry » et « Load geometry » en fonction de l'état des boutons radio.
Paramètres	Paramètres automatiques de capture d'évènement contenant l'objet (contrôle) concerné et les paramètres de l'évènement.
Retour	Aucun.

VI- Fenêtre « Define ERP »

6.1 – Présentation générale

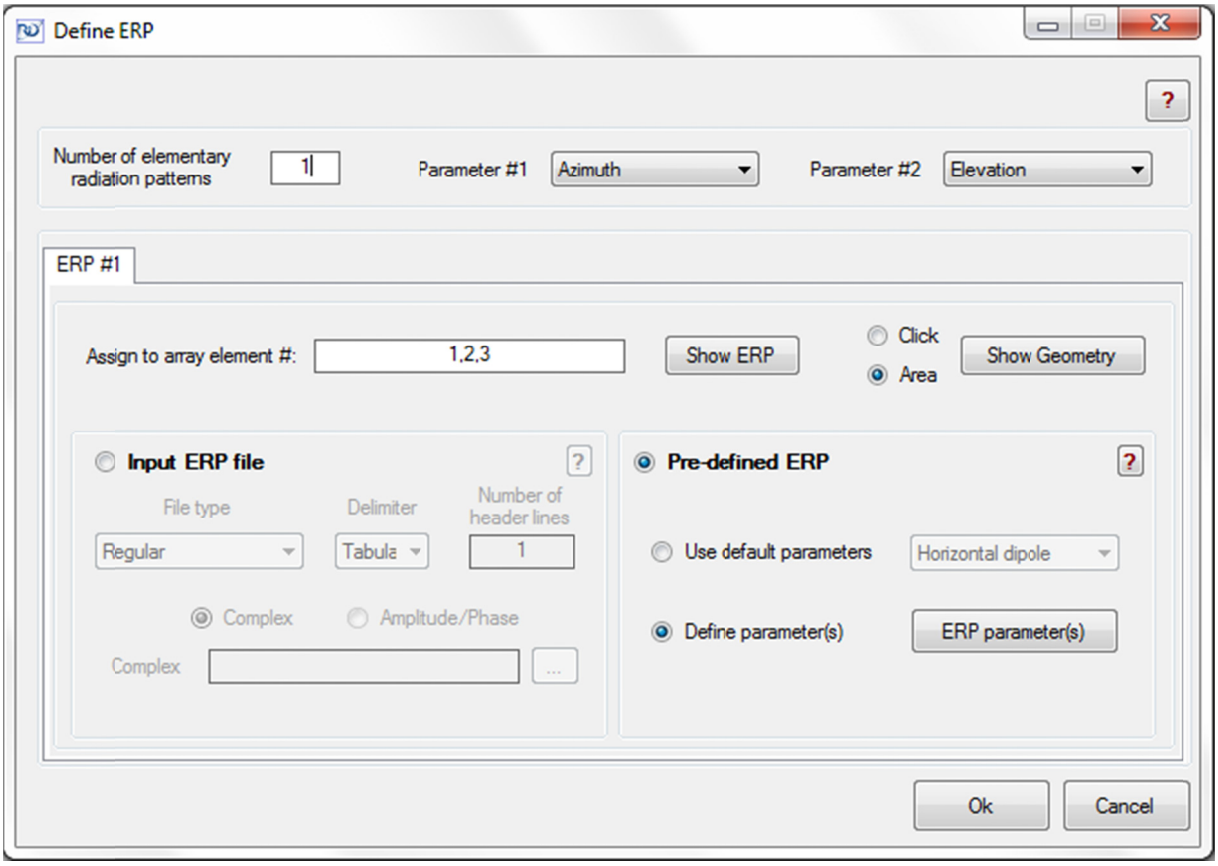


Figure 16 : Fenêtre « Define ERP »

Cette fenêtre permet de définir les caractéristiques des sous-réseaux. Elle ne peut donc apparaître que si des sous-réseaux sont définis dans la fenêtre « Array characteristics ».

6.2 – Fonctionnement de la fenêtre

6.2.1 - Règles générales

Au premier démarrage de la fenêtre, seul le champ « Number of elementary radiation pattern » est visible. Il faut alors entrer un nombre entre 1 et le nombre d'éléments dans le réseau d'antennes (en effet, un élément ne peut être associé à plusieurs ERP (Elementary Radiation Pattern), il ne peut donc pas y avoir plus d'ERP que d'éléments dans le réseau).
Pour chaque ERP, un onglet est alors créé. Dans cet onglet, l'utilisateur pourra configurer un sous-réseau et y assigner les éléments de son choix.

La configuration peut se faire par importation de fichier (partie « Input ERP file »). Selon le type de fichier, on pourra spécifier le délimiteur et le nombre de lignes d'entête afin que le programme récupère correctement les données.

Il est également possible de configurer le sous-réseau manuellement ou selon des modèles connus dans la partie « Pre-defined ERP ».

Si un élément est indiqué dans le champ « Assign to array element # » d'un onglet donné, puis dans le même champ d'un autre onglet, il sera automatiquement supprimé du champ dans le premier onglet (comme dit précédemment, un élément ne peut être associé à plusieurs ERP).
Si un ou plusieurs éléments ne sont assignés à aucun ERP lorsque l'utilisateur appuie sur « Ok », un message d'erreur s'affichera.

Il est possible à tout moment de changer le nombre dans le champ « Number of elementary radiation patterns », cependant il doit rester dans la plage de valeurs valide (entre 1 et le nombre d'éléments dans le réseau). Si, par exemple, 5 ERP sont renseignés, puis que l'utilisateur modifie le nombre d'ERP à 3, les onglets 4 et 5 seront réinitialisés. En revanche, si par exemple 3 ERP sont définis puis que l'utilisateur passe à 5 ERP, les 3 ERP définis sont sauvegardés.
Si l'utilisateur entre la valeur 0, tous les ERP sont annulés.

6.2.2 – Champ « Number of elementary radiation patterns »

Type de variable : Un seul nombre entier.
Nom de la variable dans la structure de données locale : i_NumberOfERP.
Nom du pointeur pour le contrôle : p_TextBox_NumberOfERP.
Contraintes particulières : La valeur doit être comprise entre 0 et le nombre d'élements dans le réseau d'entennes (inclus).

6.2.3 – Champ « Assign to array element # »

Type de variable : Tableau de tableaux de nombres entiers (un tableau d'entiers pour chaque onglet).
Nom de la variable dans la structure de données locale : vec_cell_i_AssignToArrayElements.
Nom du pointeur pour le contrôle : p_TextBox_AssignToArrayElements.
Contraintes particulières : Les valeurs sont séparées par des virgules. Chaque valeur doit être un entier compris entre 1 et le nombre d'éléments dans le réseau.

6.2.4 – Champ « Number of header lines »

Type de variable : Tableau de nombre entiers (un entier pour chaque onglet).
Nom de la variable dans la structure de données locale : vec_i_NumberOfHeaderLines.
Nom du pointeur pour le contrôle : p_TextBox_NumberOfHeaderLines.
Contraintes particulières : Ce champ n'est disponible à la saisie que dans le cas où l'élément « Other » est sélectionné dans le menu déroulant « File type ». Il faut alors entrer un nombre entier positif ou nul, correspondant auu nombre de lignes d'entêtes présentes dans le fichier à importer.

6.2.5 – Champ « Amplitude »

Type de variable : Tableau de tableaux de nombres réels.
Nom de la variable dans la structure de données locale : vec_cell_f_ERPDataAmplitude.
Nom du pointeur pour le contrôle : p_TextBox_ERPFileName1.
Contraintes particulières : Ce champ doit contenir un nom de fichier valide, et dont le contenu est valide selon les critères sélectionnés par l'utilisateur (délimiteur de valeurs, nombre de lignes d'entête, nombre de colonnes).

6.2.6 – Champ « Complex »

Type de variable : Tableau de tableaux de nombres réels.
Nom de la variable dans la structure de données locale : vec_cell_f_ERPDataComplex.

Nom du pointeur pour le contrôle : p_TextBox_ERPFileName1.
Contraintes particulières : Ce champ doit contenir un nom de fichier valide, et dont le contenu est valide selon les critères sélectionnés par l'utilisateur (délimiteur de valeurs, nombre de lignes d'entête, nombre de colonnes).

6.2.7 – Champ « Phase »

Type de variable : Tableau de tableaux de nombres réels.
Nom de la variable dans la structure de données locale : vec_cell_f_ERPDataPhase.
Nom du pointeur pour le contrôle : p_TextBox_ERPFileName2.
Contraintes particulières : Ce champ doit contenir un nom de fichier valide, et dont le contenu est valide selon les critères sélectionnés par l'utilisateur (délimiteur de valeurs, nombre de lignes d'entête, nombre de colonnes).

6.3- La classe « CdefineERPParameters »

Cette classe contient les variables de la fenêtre « Define ERP » qui seront mémorisées dans la fenêtre principale.
Ces variables sont les suivantes :

```
// Declare Local Attributes
private string m_str_NumberOfERP;
private string m_str_ParameterName1;
private string m_str_ParameterName2;
private string[] m_vec_str_AssignToArrayElement;
private string[] m_vec_str_FileType;
private string[] m_vec_str_ERPAmplitudeFileName;
private string[] m_vec_str_ERPComplexFileName;
private string[] m_vec_str_ERPPhaseFileName;
private string[] m_vec_str_DefaultParameter;
private string[] m_vec_str_Delimiter;
private string[] m_vec_str_NumberOfHeaderLines;
private bool[] m_vec_b_InputFile;
private bool[] m_vec_b_PreDefinedERP;
private bool[] m_vec_b_Click;
private bool[] m_vec_b_Complex;
private bool[] m_vec_b_DefaultParameters;
```

Il s'agit du contenu des champs de saisie, de la sélection des menus déroulants et de l'état des boutons radios, et ce pour chaque onglet (chaque ERP).

6.4- Pointeurs de la fenêtre « Define ERP »

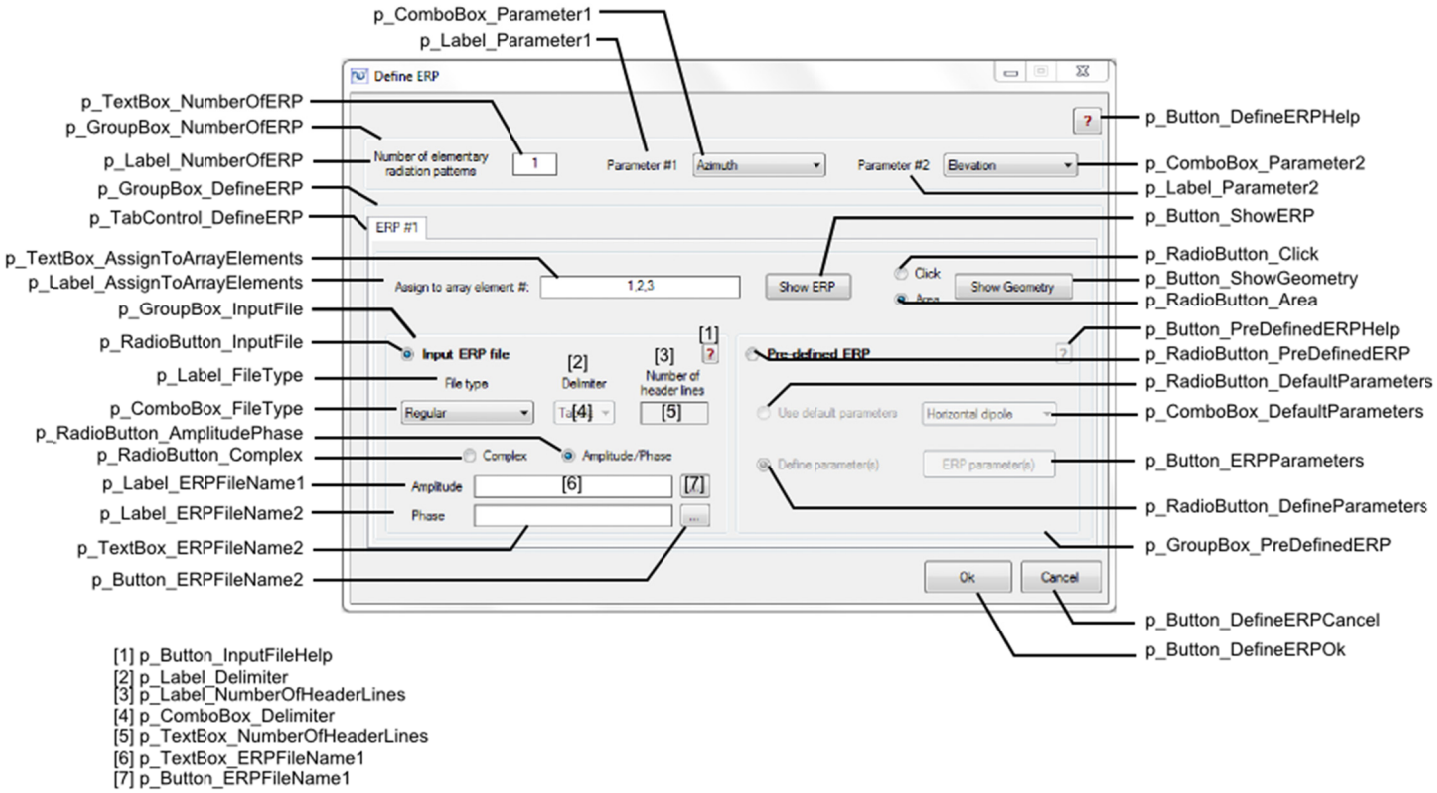


Figure 17 : pointeurs de la fenêtre « Define ERP »

6.5 – Méthodes de la classe « CdefineERP »

AmplPhaseERP

Nom	`private void AmplPhaseERP()`
Description	Gère l'affichage des champs de saisie pour les noms de fichiers (groupe « Input ERP file ») en fonction de l'état du bouton radio « Aplitude/Phase.
Paramètres	Aucun.
Retour	Aucun.

CdefineERP (Constructeur)

Nom	`public CdefineERP(float[][] vec_Position_GUI, CdefineERPParameters ERPParameters_GUI, CcomputeOptionsParameters ComputeOptionsParameters_GUI)`
Description	Initialise les contrôles et les variables de la fenêtre « Define ERP ».
Paramètres	**`float[][] vec_Position_GUI:`** Coordonnées de tous les élements du réseau d'antennes. **`CdefineERPParameters ERPParameters_GUI:`** Valeurs de tous les paramètres de la fenêtre « Define ERP » gardés en mémoire dans la fenêtre principale. **`CcomputeOptionsParameters ComputeOptionsParameters_GUI:`** Valeurs de tous les paramètres de la fenêtre « Compute options » gardés en mémoire dans la fenêtre principale. Ils sont susceptibles d'être réinitialisés si des ERPs définis prcédemment sont supprimés.
Retour	Aucun.

CdefineERP_FormClosing

Nom	`private void CdefineERP_FormClosing(object sender, FormClosingEventArgs e)`
Description	Fonction d'écoute de l'évènement : fermeture de la fenêtre « Define ERP » (appui sur le bouton « Ok », le bouton « cancel » ou la croix de fermeture en haut à droite). **Cas du bouton « Ok »** : si une erreur est présente sur la fenêtre, sa fermeture est annulée. Sinon, la fenêtre se ferme sans demander de confirmation à l'utilisateur. **Autres cas** : Une fenêtre de dialogue demande confirmation à l'utilisateur pour fermer la fenêtre ; la fenêtre se ferme donc ou non selon la réponse donnée.
Paramètres	Paramètres automatiques de capture d'évènement contenant l'objet (contrôle) concerné et les paramètres de l'évènement.
Retour	Aucun.

CheckParameter1

Nom	`private void CheckParameter1()`
Description	Vérifie que la sélection entre les menus « Parameter 1 » et « Parameter 2 » est différente (lorsque la sélection est changée sur le menu 1). Sinon, un message d'erreur est affiché et un autre élément est automatiquement sélectionné.
Paramètres	Aucun.
Retour	Aucun.

CheckParameter2

Nom	`private void CheckParameter2()`
Description	Vérifie que la sélection entre les menus « Parameter 1 » et « Parameter 2 » est différente (lorsque la sélection est changée sur le menu 2). Sinon, un message d'erreur est affiché et un autre élément est automatiquement sélectionné.
Paramètres	Aucun.
Retour	Aucun.

ComplexERP

Nom	`private void ComplexErp()`
Description	Gère l'affichage des champs de saisie pour les noms de fichiers (groupe « Input ERP file ») en fonction de l'état du bouton radio « Complex ».
Paramètres	Aucun.
Retour	Aucun.

DisplayParameterCurrentErp

Nom	`private void DisplayParameterCurrentErp()`
Description	Affiche le contenu des champs de saisie de l'onglet courant (changement d'onglet).
Paramètres	Aucun.
Retour	Aucun.

EltAssignedToCurrErp

Nom	`private string EltAssignedToCurrErp()`
Description	Vérifie le contenu du champ « Assigned to array element # » de l'onglet courant ainsi que la cohérence avec le contenu de ce champ pour les autres onglets. Si un élement était déjà présent dans un autre onglet, il en est effacé.
Paramètres	`Aucun.`
Retour	Chaîne de caractères vide ou contenant un message d'erreur.

EraseAllTabs

Nom	`private void EraseAllTabs()`
Description	Réinitialise toutes les variables et remet l'affichage des onglets à leurs valeurs par défaut.
Paramètres	Aucun.
Retour	Aucun.

ErpFileName1

Nom	`private string ErpFileName1()`
Description	Vérifie que la chaîne de caractères présente dans le champ « Complex » ou « Amplitude » (selon l'état du bouton radio) est un nom de fichier valide et existant.
Paramètres	`Aucun.`
Retour	Chaîne de caractères vide ou contenant un message d'erreur.

ErpFileName2

Nom	`private string ErpFileName1()`
Description	Vérifie que la chaîne de caractères présente dans le champ « Phase » est un nom de fichier valide et existant.
Paramètres	`Aucun.`
Retour	Chaîne de caractères vide ou contenant un message d'erreur.

ErpFileType

Nom	`private void ErpFileType()`
Description	Gère l'affichage du menu « Delimiter » et du champ de saisie « Number of header lines » en fonction de la sélection dans le menu « File type ».
Paramètres	Aucun.
Retour	Aucun.

FillNumberOfERP

Nom	`private string FillNumberOfERP()`
Description	Vérifie le contenu du champ de saisie « Number of ERP » et gère l'affichage du nombre d'onglets et de leur contenu.
Paramètres	`Aucun.`
Retour	Chaîne de caractères vide ou contenant un message d'erreur.

GetClassDefineERPParameters

Nom	`public CdefineERPParameters GetClassDefineERPParameters()`
Description	Retourne au GUI principal les valeurs de la fenêtre « Define ERP».
Paramètres	Aucun.
Retour	Classe `CdefineERPParameters` contenant les valeurs des paramètres.

GetComputeOptionsParameters

Nom	`public CcomputeOptionsParameters GetComputeOptionsParameters()`
Description	Retourne au GUI principal les valeurs de la fenêtre « Compute options» (éventuellement réinitialisés).
Paramètres	Aucun.
Retour	Classe `CcomputeOptionsParameters` contenant les valeurs des paramètres.

InputFile

Nom	`private void InputFile()`
Description	Gère l'état (actif ou non) du groupe « Input ERP file » selon l'état du bouton radio associé.
Paramètres	Aucun.
Retour	Aucun.

NbHeaderLines

Nom	`private string NbHeaderLines()`
Description	Vérifie le contenu du champ de saisie « Number of header lines ».
Paramètres	`Aucun.`
Retour	Chaîne de caractères vide ou contenant un message d'erreur.

OkPushButton

Nom	`private string OkPushButton()`
Description	Fonction appelée dans `p_Button_DefineERPOk_Click`, c'est-à-dire lorsque l'utilisateur a appuyé sur le bouton « Ok ». Vérifie que les champs obligatoires ne sont pas vides, et qu'il n'y a pas d'erreur au niveau de l'assignation des élements aux ERPs (chaque élément doit être assigné, et ce à un unique ERP).
Paramètres	Aucun.
Retour	Chaîne de caractères vide ou contenant un message d'erreur.

p_Button_BrowseFile_Click

Nom	`private void p_Button_BrowseFile_Click(object sender, EventArgs e)`
Description	Fonction d'écoute de l'évènement : appui sur l'un des deux boutons « … » servant à sélectionner un fichier. Ouvre un explorateur de fichiers, et vérifie le contenu du fichier sélectionné par l'utilisateur.
Paramètres	Paramètres automatiques de capture d'évènement contenant l'objet (contrôle) concerné et les paramètres de l'évènement.
Retour	Aucun.

p_Button_DefineERPOk_Click

Nom	`private void p_Button_DefineERPOk_Click(object sender, EventArgs e)`
Description	Fonction d'écoute de l'évènement : appui sur le bouton « Ok ». Vérifie des valeurs de la fenêtre et met à jour la valeur booléenne qui annule la demande de confirmation à l'utilisateur pour fermer la fenêtre. Après l'appel de cette fonction, le programme appelle la fonction « `CdefineERP_FormClosing » (fenêtre modale).`
Paramètres	Paramètres automatiques de capture d'évènement contenant l'objet (contrôle) concerné et les paramètres de l'évènement.
Retour	Aucun.

p_ComboBox_SelectedIndexChanged

Nom	`private void p_ComboBox_SelectedIndexChanged(object sender, EventArgs e)`
Description	Fonction d'écoute sur l'évènement : changement dans la sélection de l'un des menus déroulants. Dans le cas des menus « Parameter 1 » et « Parameter 2 », vérifie la cohérence (les deux menus ne peuvent pas avoir la même sélection) et réinitialise le contenu de tous les onglets. Pour le menu « File type », gère l'affichage du groupe « Input ERP file ». Pour les autres menus, il s'agit d'une simple mise à jour de la variable associée.
Paramètres	Paramètres automatiques de capture d'évènement contenant l'objet (contrôle) concerné et les paramètres de l'évènement.
Retour	Aucun.

p_Onglets_DefineERP_Selected

Nom	`private void p_Onglets_DefineERP_Selected(object sender, TabControlEventArgs e)`
Description	Fonction d'écoute de l'évènement : selection d'un onglet (sélection terminée). Met à jour le contenu de l'onglet à afficher et la variable indiquant le numéro de l'onglet sélectionné.
Paramètres	Paramètres automatiques de capture d'évènement contenant l'objet (contrôle) concerné et les paramètres de l'évènement.
Retour	Aucun.

p_TabControl_DefineERP_Selecting

Nom	`private void p_TabControl_DefineERP_Selecting(object sender, TabControlCancelEventArgs e)`
Description	Fonction d'écoute de l'évènement : selection d'un onglet (sélection en cours). Annule la sélection de l'onglet (fonction « `p_Onglets_DefineERP_Selected` ») si l'un des champs de noms de fichiers n'est pas rempli alors que le bouton radio « Input ERP file » est sélectionné.
Paramètres	Paramètres automatiques de capture d'évènement contenant l'objet (contrôle) concerné et les paramètres de l'évènement.
Retour	Aucun.

p_RadioButton_CheckedChanged

Nom	`private void p_RadioButton_CheckedChanged(object sender, EventArgs e)`
Description	Fonction d'écoute sur l'évènement : changement d'état d'un bouton radio. Gère l'affichage de l'interface selon l'état du bouton radio.
Paramètres	Paramètres automatiques de capture d'évènement contenant l'objet (contrôle) concerné et les paramètres de l'évènement.
Retour	Aucun.

p_TextBox_KeyPress

Nom	`private void p_TextBox_KeyPress(object sender, KeyPressEventArgs e)`
Description	Fonction d'écoute sur l'évènement : appui sur une touche alors que le focus est sur un champ de saisie. Cette fonction ne fait rien si l'utilisateur a appuyé sur une autre touche que la touche « Entrée ». Vérifie le contenu du champ de saisie. Si le contenu ne correspond pas aux contraintes, affichage d'un message d'erreur sous forme de fenêtre « pop-up ».
Paramètres	Paramètres automatiques de capture d'évènement contenant l'objet (contrôle) concerné et les paramètres de l'évènement.
Retour	Aucun.

p_TextBox_Leave

Nom	`private void p_TextBox_Leave(object sender, EventArgs e)`
Description	Fonction d'écoute sur l'évènement : appui sur la touche « tabulation » ou clic sur un élément de la fenêtre alors que le focus est sur un champ de saisie. Vérifie le contenu du champ de saisie. Si le contenu ne correspond pas aux contraintes, un message d'erreur apparaît et le focus reste sur le champ de saisie concerné.
Paramètres	Paramètres automatiques de capture d'évènement contenant l'objet (contrôle) concerné et les paramètres de l'évènement.
Retour	Aucun.

p_TextBox_MouseEnter

Nom	`private void p_TextBox_MouseEnter(object sender, EventArgs e)`
Description	Fonction d'écoute sur l'évènement : passage de la souris au dessus d'un champ de saisie. Si le champ de saisie contient une chaîne valide, une « bulle » apparaîtra avec le détail du contenu de ce champ de saisie. Par exemple, si un champ contient la chaîne « 1:5 », la bulle affichera « 1,2,3,4,5». S'il s'agit d'un fichier, c'est le contenu du fichier qui sera affiché.
Paramètres	Paramètres automatiques de capture d'évènement contenant l'objet (contrôle) concerné et les paramètres de l'évènement.
Retour	Aucun.

p_TextBox_TextChanged

Nom	`private void p_TextBox_TextChanged(object sender, EventArgs e)`
Description	Fonction d'écoute sur l'évènement : changement du texte de l'un des champs de saisie. Seuls les champs de saisie pour les noms de fichiers sont concernés par cette fonction. Cette fonction gère l'état des menus déroulants et du champ de saisie « Number of header lines » du groupe « Input ERP file ».
Paramètres	Paramètres automatiques de capture d'évènement contenant l'objet (contrôle) concerné et les paramètres de l'évènement.
Retour	Aucun.

PreDefErpDefault

Nom	`private void PreDefErpDefault()`
Description	Gère l'affichage du groupe « Pre-defined ERP » en fonction du bouton radio « Use default parameters ».
Paramètres	Aucun.
Retour	Aucun.

PreDefErpDefine

Nom	`private void PreDefErpDefine()`
Description	Gère l'affichage du groupe « Pre-defined ERP » en fonction du bouton radio « Define parameter(s) ».
Paramètres	Aucun.
Retour	Aucun.

PreDefined

Nom	`private void PreDefined()`
Description	Gère l'état (actif ou non) du groupe « Pre-defined ERP » selon l'état du bouton radio associé.
Paramètres	Aucun.
Retour	Aucun.

ReadErpFile

Nom	`private void ReadErpFile(string str_ERPFileName, bool b_isPhase)`
Description	Vérifie le contenu du fichier sélectionné par l'utilisateur.

Paramètres	**`string str_ERPFileName:`**
	`Nom du fichier à vérifier.`
	
	`bool b_isPhase:`
	`Valeur booléenne indiquant si il s'agit d'un fichier pour les valeurs de phase ou non (amplitude ou complexe).`

Retour	Aucun.

SetNewDataValues

Nom	`private void SetNewDataValues()`
Description	Redéfinit les variables selon le nombre d'onglets (donc le nombre d'ERP). La taille des tableaux est recalculée, tout en conservant les valeurs définies auparavant.
Paramètres	Aucun.
Retour	Aucun.

UpdateCurrErp

Nom	`private void UpdateCurrErp()`
Description	Gère l'affichage de l'onglet courant (changement d'onglet).
Paramètres	Aucun.
Retour	Aucun.

ViewAll

Nom	`private void ViewAll(bool b_ViewState)`
Description	Affiche les onglets ou efface tout l'affichage (cas où l'utilisateur entre la valeurs 0 dans le champ « Number of elementary radiation patterns).
Paramètres	**`bool b_ViewState:`** `Valeur booléenne indiquant si l'on doit afficher ou effacer les onglets.`
Retour	Aucun.

VII – La fenêtre « Amplitude distribution »

7.1- Présentation générale

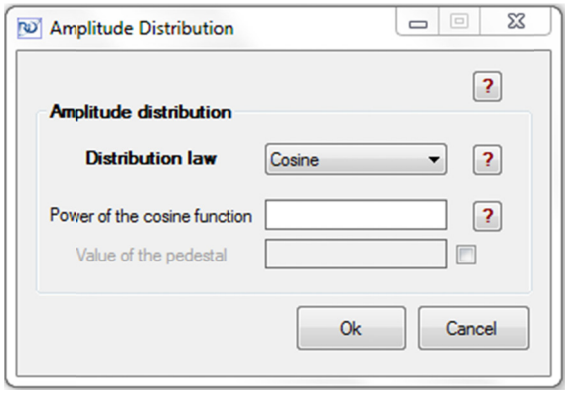


Figure 18 : La fenêtre « Amplitude distribution »

Cette fenêtre permet de définir la distribution d'amplitude appliquée aux éléments du réseau.

7.2- Fonctionnement de la fenêtre

7.2.1- Règles générales

Quatre lois sont possibles : « uniform », « Dolph-Chebyshev », « Taylor » et « Cosine ».
Il faut donc sélectionner l'une des quatre et remplir si nécéssaire les paramètres correspondants.
Il y a au maximum deux paramètres à remplir.

7.2.2- Champ « Paramètre 1 »

Type de variable : un seul nombre réel.
Nom de la variable dans la structure de données locale : f_Parameter1.
Nom du pointeur pour le contrôle : p_TextBox_AmplitudeParameter1.
Contraintes particulières : la valeur doit être positive ou nulle.

7.2.3- Champ « Paramètre 2 »

Type de variable : un seul nombre réel pour la loi « Taylor », un seul nombre entier pour la loi « Cosine ».
Nom de la variable dans la structure de données locale : i_Parameter2_Taylor ou f_Parameter2_Cosine.
Nom du pointeur pour le contrôle : p_TextBox_AmplitudeParameter2.
Contraintes particulières : dans le cas « Taylor », la valeur doit être exclusivement positive.

7.3- La classe « CamplitudeDistributionParameters »

Cette classe contient les variables de la fenêtre « Amplitude distribution » qui seront mémorisées dans la fenêtre principale.
Ces variables sont les suivantes :

```
private string m_str_AmplitudeDistribution;
private string m_str_AmplitudeParameter1;
private string m_str_AmplitudeParameter2;
private bool m_b_Parameter2Enable;
```

Il s'agit du contenu des deux champs de saisie, du type de distribution choisi dans le menu déroulant et de l'état de la case à cocher.

7.4- Pointeurs de la fenêtre « Amplitude distribution »

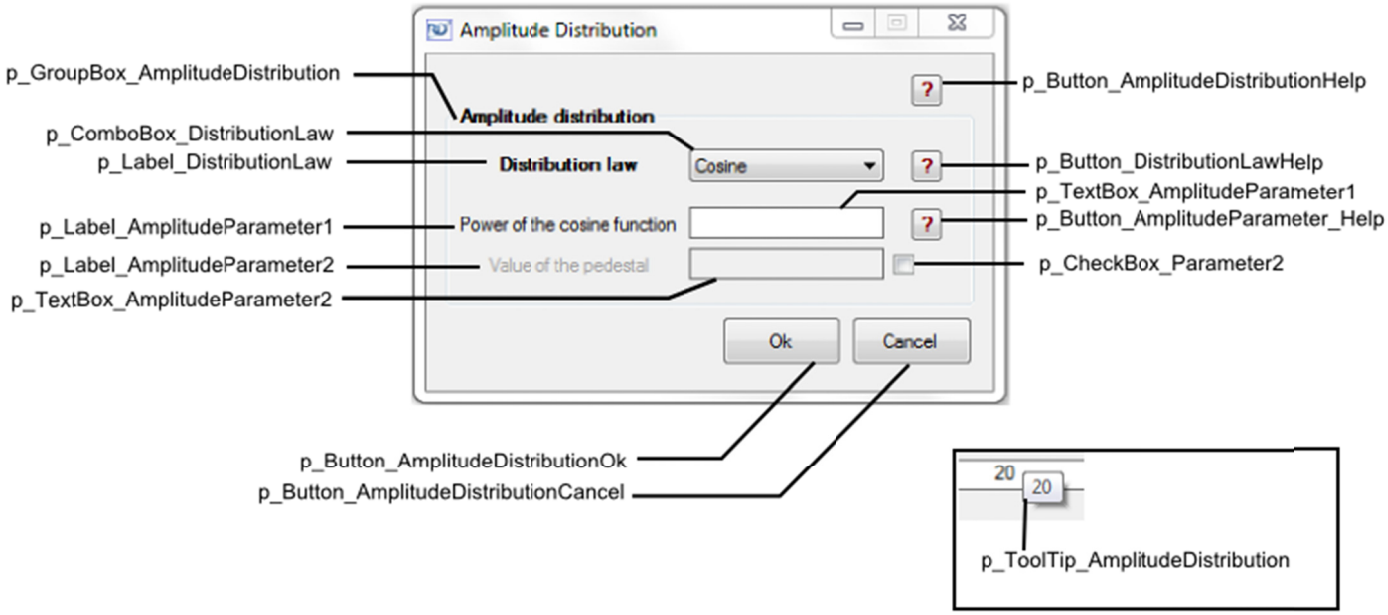


Figure 19 : pointeurs de la fenêtre « Amplitude distribution »

7.5- Méthodes de la classe « CamplitudeDistribution »

AmplitudeParameter1

Nom	`private string AmplitudeParameter1()`
Description	Vérifie le contenu du paramètre 1.
Paramètres	Aucun.
Retour	Chaîne de caractères vide ou contenant un message d'erreur.

AmplitudeParameter2

Nom	`private string AmplitudeParameter2()`
Description	Vérifie le contenu du paramètre 2.
Paramètres	Aucun.
Retour	Chaîne de caractères vide ou contenant un message d'erreur.

CamplitudeDistribution (Constructeur)

Nom	`public CamplitudeDistribution(CamplitudeDistributionParameters amplitudeDistributionParameters_GUI)`
Description	Initialise les contrôles et les variables de la fenêtre « Amplitude distribution ».
Paramètres	**`CamplitudeDistributionParameters amplitudeDistributionParameters_GUI:`** `Valeurs de tous les paramètres de la fenêtre « Array characteristics » gardés en mémoire dans la fenêtre principale.`
Retour	Aucun.

CamplitudeDistribution_FormClosing

Nom	`private void CamplitudeDistribution_FormClosing(object sender, FormClosingEventArgs e)`
Description	Fonction d'écoute de l'évènement : fermeture de la fenêtre « Amplitude distribution » (appui sur le bouton « Ok », le bouton « cancel » ou la croix de fermeture en haut à droite). **Cas du bouton « Ok »** : si une erreur est présente dans la fenêtre, sa fermeture est annulée. Sinon, la fenêtre se ferme sans demander de confirmation à l'utilisateur. **Autres cas** : Une fenêtre de dialogue demande confirmation à l'utilisateur pour fermer la fenêtre ; la fenêtre se ferme donc ou non selon la réponse donnée.
Paramètres	Paramètres automatiques de capture d'évènement contenant l'objet (contrôle) concerné et les paramètres de l'évènement.
Retour	Aucun.

GetClassAmplitudeDistributionParameters

Nom	`public CamplitudeDistributionParameters GetClassAmplitudeDistributionParameters()`
Description	Retourne au GUI principal les valeurs de la fenêtre «Amplitude distribution ».
Paramètres	Aucun.
Retour	Classe `CamplitudeDistributionParameters` contenant les valeurs des paramètres.

OkPushButton

Nom	`private string OkPushButton()`
Description	Fonction appelée dans `p_Button_AmplitudeDistributionOk_Click`, c'est-à-dire lorsque l'utilisateur a appuyé sur le bouton « Ok ». Vérifie que les champs devant être remplis ne sont pas vides.
Paramètres	Aucun.
Retour	Chaîne de caractères vide ou contenant un message d'erreur.

p_Button_AmplitudeDistributionOk _Click

Nom	`private void p_Button_AmplitudeDistributionOk_Click(object sender, EventArgs e)`
Description	Fonction d'écoute de l'évènement : appui sur le bouton « Ok ». Vérifie des valeurs de la fenêtre et met à jour la valeur booléenne qui annule la demande de confirmation à l'utilisateur pour fermer la fenêtre. Après l'appel de cette fonction, le programme appelle la fonction « `CamplitudeDistribution_FormClosing » (fenêtre modale)`.
Paramètres	Paramètres automatiques de capture d'évènement contenant l'objet (contrôle) concerné et les paramètres de l'évènement.
Retour	Aucun.

p_CheckBox_Parameter2_CheckedChanged

Nom	`private void p_CheckBox_Parameter2_CheckedChanged(object sender, EventArgs e)`
Description	Fonction d'écoute de l'évènement : changement d'état de la case à cocher du paramètre 2. Active ou désactive le champ de saisie « paramètre 2 » (mode « Cosine ») en fonction de l'état de la case à cocher.
Paramètres	Paramètres automatiques de capture d'évènement contenant l'objet (contrôle) concerné et les paramètres de l'évènement.
Retour	Aucun.

p_ComboBox_DistributionLaw_SelectedIndexChanged

Nom	`private void p_ComboBox_DistributionLaw_SelectedIndexChanged(object sender, EventArgs e)`
Description	Fonction d'écoute de l'évènement : changement dans la sélection du menu déroulant. Affiche les différents paramètres (ou aucun) selon le type de distribution choisi.
Paramètres	Paramètres automatiques de capture d'évènement contenant l'objet (contrôle) concerné et les paramètres de l'évènement.
Retour	Aucun.

p_TextBox_KeyPress

Nom	`private void p_TextBox_KeyPress(object sender, KeyPressEventArgs e)`
Description	Fonction d'écoute sur l'évènement : appui sur une touche alors que le focus est sur un champ de saisie. Cette fonction ne fait rien si l'utilisateur a appuyé sur une autre touche que la touche « Entrée ». Vérifie le contenu du champ de saisie. Si le contenu ne correspond pas aux contraintes, affichage d'un message d'erreur sous forme de fenêtre « pop-up ».
Paramètres	Paramètres automatiques de capture d'évènement contenant l'objet (contrôle) concerné et les paramètres de l'évènement.
Retour	Aucun.

p_TextBox_Leave

Nom	`private void p_TextBox_Leave(object sender, EventArgs e)`
Description	Fonction d'écoute sur l'évènement : appui sur la touche « tabulation » ou clic sur un élément de la fenêtre alors que le focus est sur un champ de saisie. Vérifie le contenu du champ de saisie. Si le contenu ne correspond pas aux contraintes, un message d'erreur apparaît et le focus reste sur le champ de saisie concerné.
Paramètres	Paramètres automatiques de capture d'évènement contenant l'objet (contrôle) concerné et les paramètres de l'évènement.
Retour	Aucun.

p_TextBox_MouseEnter

Nom	`private void p_TextBox_MouseEnter(object sender, EventArgs e)`
Description	Fonction d'écoute sur l'évènement : passage de la souris au dessus d'un champ de saisie. Si le champ de saisie contient une chaîne valide, une « bulle » apparaîtra avec le détail du contenu de ce champ de saisie.
Paramètres	Paramètres automatiques de capture d'évènement contenant l'objet (contrôle) concerné et les paramètres de l'évènement.
Retour	Aucun.

Parameter2State

Nom	`private void Parameter2State()`
Description	Gère l'état (actif ou inactif) du champ de saisie pour le paramètre 2 en fonction de l'état de la case à cocher et du type de distribution choisi.
Paramètres	Aucun.
Retour	Aucun.

VIII – La fenêtre « Element and sub-array components »

8.1- Présentation générale

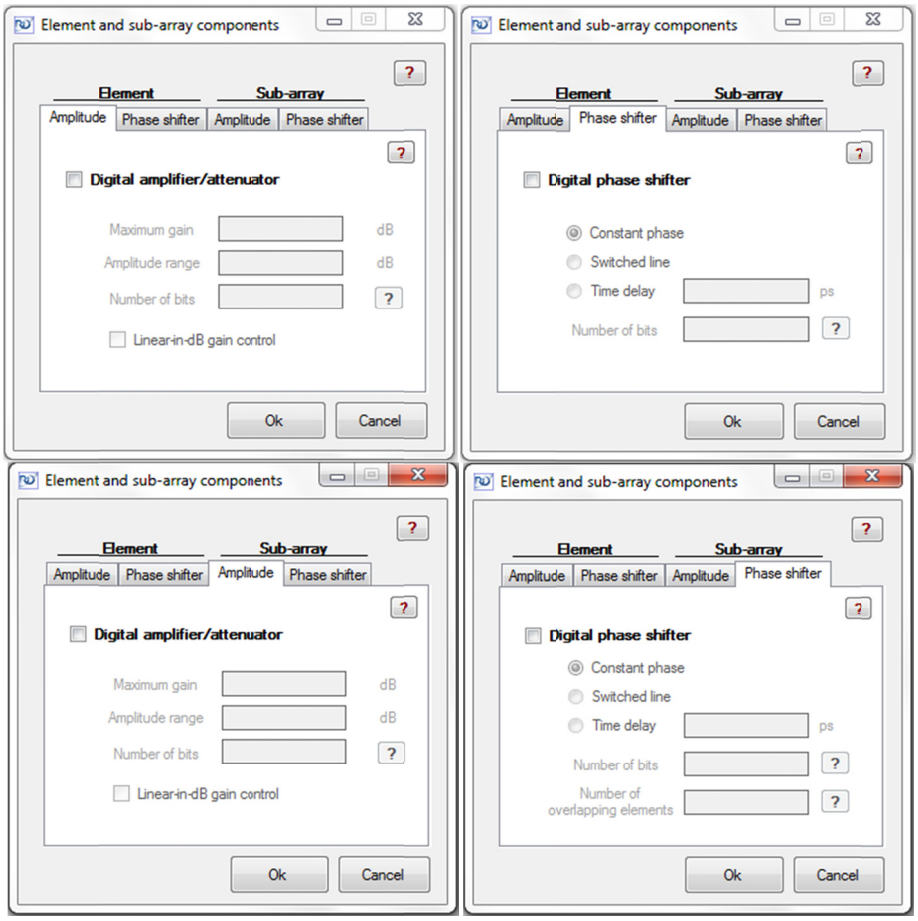


Figure 20 : La fenêtre « Element and sub-array components »

Cette fenêtre permet de définir l'amplitude et la phase des éléments du réseau et/ou des sous-réseaux s'ils sont définis.

8.2- Fonctionnement de la fenêtre

8.2.1- Règles générales

Cette fenêtre contient deux onglets si aucun sous-réseau n'est défini, quatre sinon.
Chaque onglet contient des choix ou des champs de saisie pour paramétrer l'amplitude et la phase des éléments ou des sous-réseaux.

8.2.2- Champ « Maximum gain » (onglets « Amplitude »)

Type de variable : un seul nombre réel.
Nom de la variable dans la structure de données locale : f_MaximumGain (structure « m_struct_LocalElementAmplitudeParameters » ou « m_struct_LocalSubArrayAmplitudeParameters »).
Nom du pointeur pour le contrôle : p_TextBox_ElementAmplMaximumGain ou p_TextBox_SubArrayAmplMaximumGain.
Contraintes particulières : aucune.

8.2.3- Champ « Amplitude range » (onglets « Amplitude »)

Type de variable : un seul nombre réel.
Nom de la variable dans la structure de données locale : f_AmplitudeRange (structure « m_struct_LocalElementAmplitudeParameters » ou « m_struct_LocalSubArrayAmplitudeParameters »).
Nom du pointeur pour le contrôle : p_TextBox_ElementAmplAmplitudeRange ou p_TextBox_SubArrayAmplAmplitudeRange.
Contraintes particulières : la valeur doit être strictement positive.

8.2.4- Champ « Number of bits » (tous les onglets)

Type de variable : tableau de nombres entiers.
Nom de la variable dans la structure de données locale : vec_i_NumberOfBits (structure « m_struct_LocalElementAmplitudeParameters », « m_struct_LocalSubArrayAmplitudeParameters », « m_struct_LocalElementPhaseParameters » ou « m_struct_LocalSubArrayPhaseParameters »,).
Nom du pointeur pour le contrôle : p_TextBox_ElementAmplNumberOfBits, p_TextBox_ElementPhaseNumberOfBits, p_TextBox_SubArrayAmplNumberOfBits ou p_TextBox_SubArrayPhaseNumberOfBits.
Contraintes particulières : Il peut y avoir une seule valeur, ou plusieurs séparées par des virgules ou définies par le signe « : ». Les valeurs doivent être positives ou nulles. Si plusieurs valeurs sont entrées, il s'agit d'un tuning. Il ne peut y avoir qu'un seul tuning par problème (il y aura donc une erreur si un tuning est déjà défini dans un autre onglet, ou dans les fenêtre « Frequency » ou « Problem definition ».

8.2.5- Champ « Time delay » (onglets « Phase »)

Type de variable : un seul nombre réel.
Nom de la variable dans la structure de données locale : f_TimeDelay (structure « m_struct_LocalElementPhaseParameters » ou « m_struct_LocalSubArrayPhaseParameters »).
Nom du pointeur pour le contrôle : p_TextBox_ElementPhase TimeDelay ou p_TextBox_SubArrayPhase TimeDelay.
Contraintes particulières : la valeur doit être strictement positive.

8.2.6- Champ « Number of overlapping elements » (onglet « Sub-array - Phase »)

Type de variable : un seul nombre entier.
Nom de la variable dans la structure de données locale : i_NumberOfOverlappingElements.
Nom du pointeur pour le contrôle : p_TextBox_SubArrayPhaseNumberOfOverLapping.
Contraintes particulières : la valeur doit être comprise entre 1 et le nombre d'éléments dans le réseau d'antennes (inclus).

8.3- La classe « CelementAndSubArrayComponentsParameters »

Cette classe contient les variables de la fenêtre « Element and sub-array component » qui seront mémorisées dans la fenêtre principale.
Ces variables sont les suivantes :

```
private CelementAndSubArrayComponents_Amplitude m_ElementAmplitude;
private CelementAndSubArrayComponents_Amplitude m_SubArrayAmplitude;
private CelementAndSubArrayComponents_PhaseShifter m_ElementPhaseShifter;
private CelementAndSubArrayComponents_PhaseShifter m_SubArrayPhaseShifter;
private string m_str_NumberOfOverlappingElements;
```

Il s'agit du contenu du champ « Number of overlapping elements », ainsi que d'instances des classes « CelementAndSubArrayComponents_Amplitude » et « CelementAndSubArrayComponents_PhaseShifter »

8.4- La classe « CelementAndSubArrayComponents_Amplitude »

Cette classe contient les attributs suivants :

```
private bool m_b_DigitalAmplifierAttenuator;
private bool m_b_LinearIndBGainControl;
private string m_str_MaximiumGain;
private string m_str_AmplitudeRange;
private string m_str_NumberOfBits;
```

Il s'agit de l'état des deux cases à cocher « Digital amplifier/attenuator » et « Linear in-dB gain control », ainsi que le contenu des champs présents dans les onglets « Amplitude ».
Pour résumer, il s'agit de tous les paramètres d'un onglet « Amplitude », c'est pourquoi deux instances de cette classe sont créées dans la classe « CelementAndSubArrayComponentsParameters » : une pour l'onglet « Element – amplitude » et une pour l'onglet «Sub-array – amplitude ».

8.5- La classe « CelementAndSubArrayComponents_PhaseShifter »

Cette classe contient les attributs suivants :

```
private bool m_b_DigitalPhaseShifter;
private bool m_b_ConstantPhase;
private bool m_b_SwitchedLine;
private bool m_b_TimeDelay;
private string m_str_TimeDelay;
private string m_str_NumberOfBits;
```

Selon le même principe que pour la classe « CelementAndSubArrayComponents_Amplitude », il s'agit de tous les paramètres d'un onglet « Phase » (état des trois boutons radio, valeurs des champs de saisie). Deux instances de cette classe sont créées dans la classe « CelementAndSubArrayComponentsParameters » : une pour l'onglet « Element – phase » et une pour l'onglet «Sub-array – phase ».

8.6- Pointeurs de la fenêtre « Element and sub-array components »

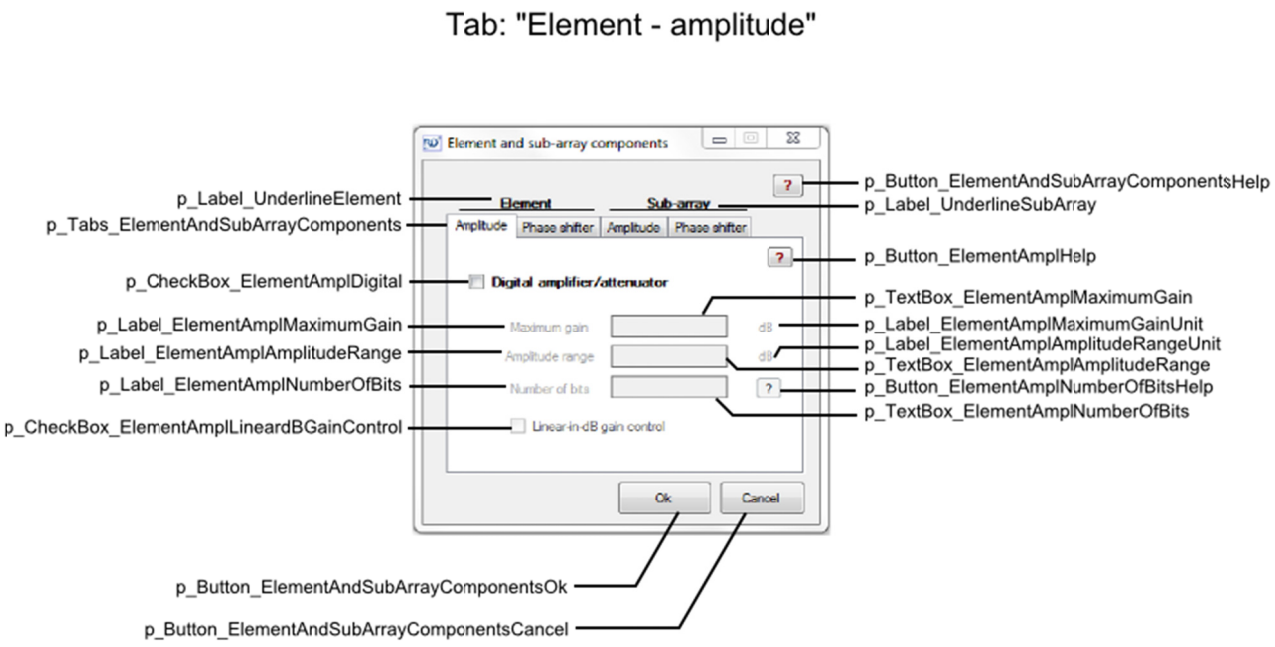


Figure 21 : pointeurs de la fenêtre « Element and sub-array components », onglet « Element – amplitude »

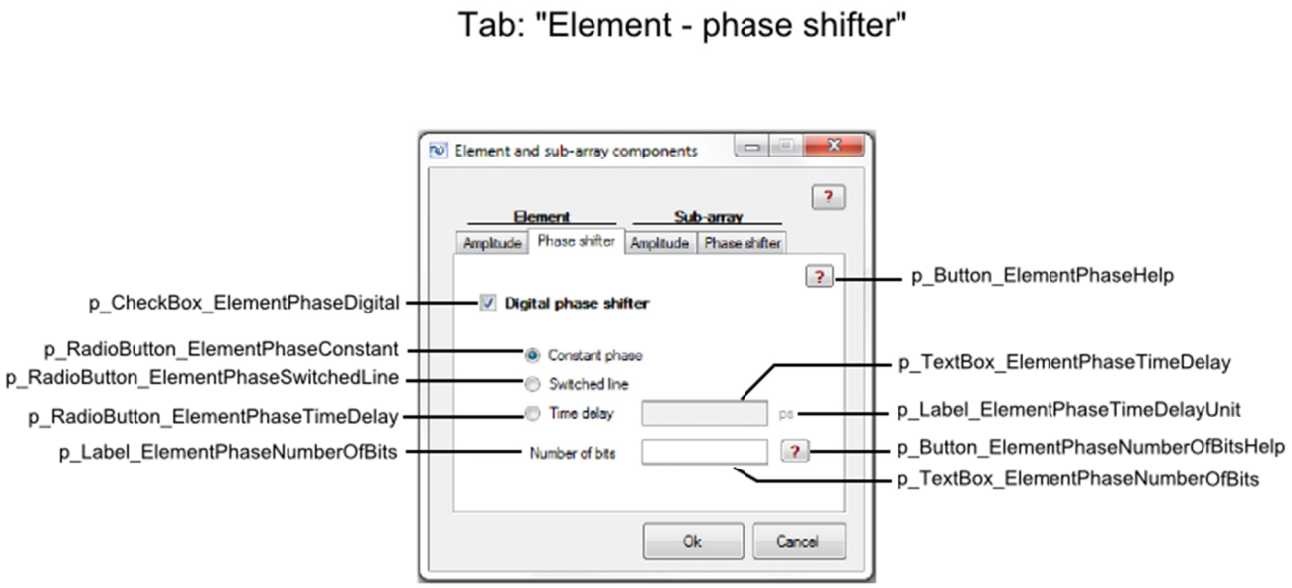


Figure 22 : pointeurs de la fenêtre « Element and sub-array components », onglet « Element – phase shifter »

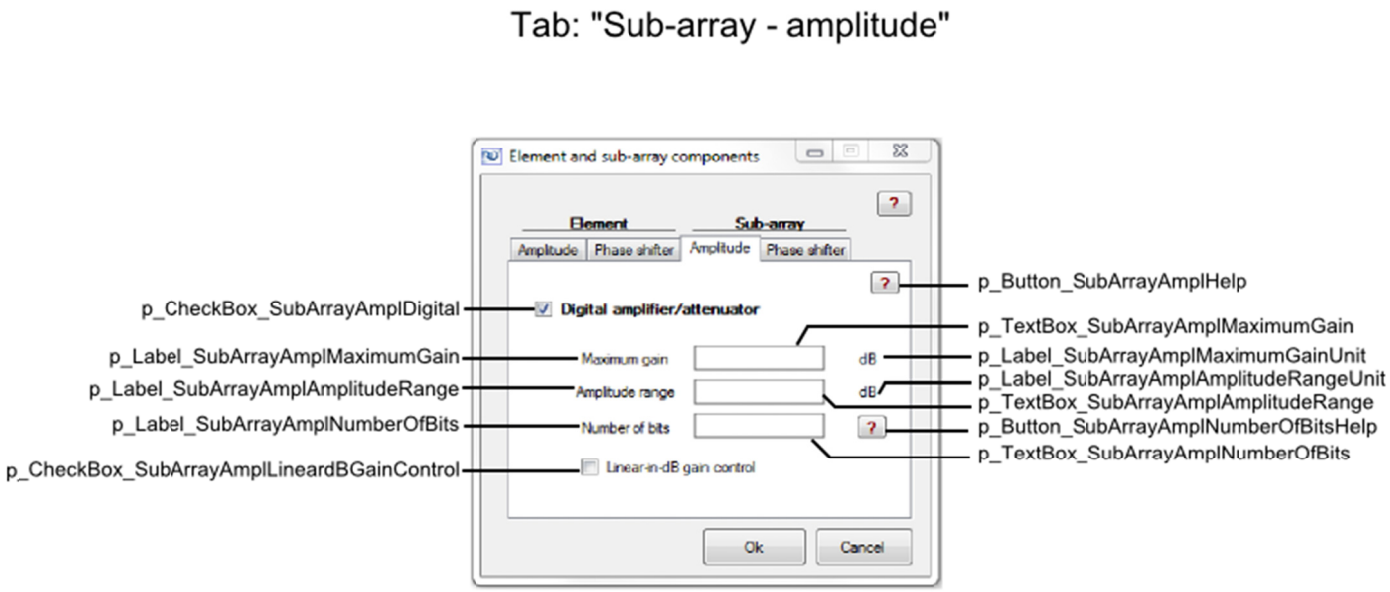


Figure 23 : pointeurs de la fenêtre « Element and sub-array components », onglet « Sub-array – amplitude »

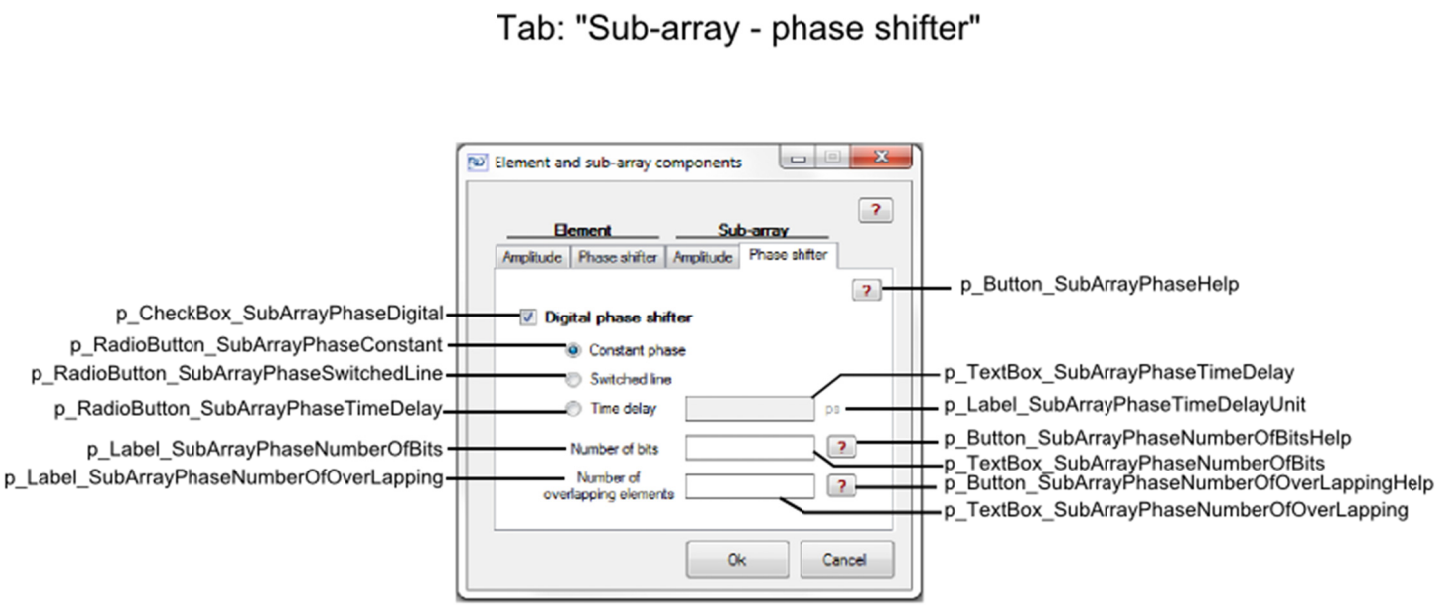


Figure 24 : pointeurs de la fenêtre « Element and sub-array components », onglet « Sub-array – phase shifter »

8.7- Méthodes de la classe « CelementAndSubArrayComponents »

AmplRange

Nom	`private string AmplRange()`
Description	Vérifie le contenu du champ de saisie « Amplitude range » dans l'onglet « Amplitude » sélectionné.
Paramètres	Aucun.
Retour	Chaîne de caractères vide ou contenant un message d'erreur.

CelementAndSubArrayComponents (Constructeur)

Nom	`public CelementAndSubArrayComponents(CelementAndSubArrayComponentsParameters ElementAndSubArrayComponentsParameters_GUI, CtuningParameter tunningParameter_GUI, CtuningParameter tuningParameterUseInPlottingOptions_GUI, int i_NumberOfElements_GUI, bool b_SubarrayDefined_GUI)`
Description	Initialise les contrôles et les variables de la fenêtre « Element and sub-array components ».
Paramètres	**CelementAndSubArrayComponentsParameters ElementAndSubArrayComponentsParameters_GUI:** Valeurs de tous les paramètres de la fenêtre « Element and sub-array components » gardés en mémoire dans la fenêtre principale. **CtuningParameter tunningParameter_GUI:** voir classe "CtuningParameter". Il s'agit du tuning défini pour le probleme courant. CtuningParameter **tunningParameterUseInPlottingOptions_GUI:** voir classe "CtuningParameter". Il s'agit du tuning en cours d'utilisation dans la fenêtre « Plotting options ». **int i_NumberOfElements_GUI:** Nombre d'éléments dans le réseau d'antennes. bool b_SubarrayDefined_GUI: Valeur booléenne indiquant si des sous-réseaux ont été définis (vrai) ou non (faux).
Retour	Aucun.

CelementAndSubArrayComponents_FormClosing

Nom	`private void CelementAndSubArrayComponents_FormClosing(object sender, FormClosingEventArgs e)`
Description	Fonction d'écoute de l'évènement : fermeture de la fenêtre « Element and sub-array components » (appui sur le bouton « Ok », le bouton « cancel » ou la croix de fermeture en haut à droite). **Cas du bouton « Ok »** : si une erreur est présente sur la fenêtre, sa fermeture est annulée. Sinon, la fenêtre se ferme sans demander de confirmation à l'utilisateur. **Autres cas** : Une fenêtre de dialogue demande confirmation à l'utilisateur pour fermer la fenêtre ; la fenêtre se ferme donc ou non selon la réponse donnée.
Paramètres	Paramètres automatiques de capture d'évènement contenant l'objet (contrôle) concerné et les paramètres de l'évènement.
Retour	Aucun.

Delay

Nom	`private string Delay()`
Description	Vérifie le contenu du champ de saisie « Time delay » dans l'onglet « Phase » sélectionné.
Paramètres	Aucun.
Retour	Chaîne de caractères vide ou contenant un message d'erreur.

DigitalAmplAtt

Nom	`private string DigitalAmplAtt(string str_TuningNameLocal)`
Description	Affiche ou non le contenu dans les onglets « Amplitude » selon l'état de la case à cocher correspondante.

Paramètres	**`string str_TuningNameLocal:`**
	`Chaîne permettant de différencier l'onglet « Element – amplitude » et « Sub-array – amplitude ». La valeur peut être :`
	`-«NumberOfBitsAmplitude»`
	`-«NumberOfBitsAmplitude_Subarray»`

Retour	Chaîne de caractères vide ou contenant un message d'erreur (notamment en présence de plusieurs tunings).

DigitalPhaser

Nom	`private string DigitalPhaser(string str_TuningNameLocal)`
Description	Affiche ou non le contenu dans les onglets « Phase » selon l'état de la case à cocher correspondante.

Paramètres	**`string str_TuningNameLocal:`**
	`Chaîne permettant de différencier l'onglet « Element – phase » et « Sub-array – phase ». La valeur peut être :`
	`-«NumberOfBitsPhase»`
	`-«NumberOfBitsPhase_Subarray»`

Retour	Chaîne de caractères vide ou contenant un message d'erreur (notamment en présence de plusieurs tunings).

GetClassElementAndSubArrayComponentsParameters

Nom	`public CelementAndSubArrayComponentsParameters GetClassElementAndSubArrayComponentsParameters()`
Description	Retourne au GUI principal les valeurs de la fenêtre «Element and sub-array components ».
Paramètres	Aucun.
Retour	Classe `CelementAndSubArrayComponentsParameters contenant les valeurs des paramètres.`

GetClassTuningParameters

Nom	`public CtuningParameter GetClassTuningParameters()`
Description	Retourne au GUI principal les valeurs de tunning de la fenêtre « Element and sub-array components » (si défini, sinon le tuning n'est pas modifié).
Paramètres	Aucun.
Retour	Classe `CtuningParameters contenant les valeurs des paramètres de tuning.`

MaxGain

Nom	`private string MaxGain()`
Description	Vérifie le contenu du champp de saisie « Maximum gain » dans l'onglet « Amplitude » sélectionné.
Paramètres	Aucun.
Retour	Chaîne de caractères vide ou contenant un message d'erreur.

NumberOfBits

Nom	`private string NumberOfBits(string str_TuningNameLocal)`
Description	Vérifie le contenu du champ de saisie « Number of bits » dans l'onglet sélectionné.

Paramètres	`string str_TuningNameLocal:`
	`Cette chaîne de caractères peut prendre 4 valeurs :`
	`«NumberOfBitAmplitude», «NumberOfBitPhase»,`
	`«NumberOfBitAmplitude_SubArray» ou «NumberOfBitPhase_SubArray».`
	`Cela permet de savoir quel onglet est concerné.`

Retour	Chaîne de caractères vide ou contenant un message d'erreur.

OkPushButton

Nom	`private string OkPushButton()`
Description	Fonction appelée dans `p_Button_ElementAndSubArrayComponentOk_Click`, c'est-à-dire lorsque l'utilisateur a appuyé sur le bouton « Ok ». Vérifie que les champs obligatoires ne sont pas vides. Si le tuning a été changé et qu'il était utilisé dans la fenêtre « Plotting options », un message d'avertissement apparaît pour informer l'utilisateur que la fenêtre « Plotting options » a été réinitialisée.
Paramètres	Aucun.
Retour	Chaîne de caractères vide ou contenant un message d'erreur.

OverlappingElements

Nom	`private string OverlappingElements()`
Description	Vérifie le contenu du champp de saisie « Number of overlapping elements » dans l'onglet « Sub-array – phase ».
Paramètres	Aucun.
Retour	Chaîne de caractères vide ou contenant un message d'erreur.

p_Button_ElementAndSubArrayComponentsOk _Click

Nom	`private void p_Button_ElementAndSubArrayComponentsOk_Click(object sender, EventArgs e)`
Description	Fonction d'écoute de l'évènement : appui sur le bouton « Ok ». Vérifie des valeurs de la fenêtre et met à jour la valeur booléenne qui annule la demande de confirmation à l'utilisateur pour fermer la fenêtre. Après l'appel de cette fonction, le programme appelle la fonction « `CelementAndSubarrayComonents_FormClosing » (fenêtre modale).`
Paramètres	Paramètres automatiques de capture d'évènement contenant l'objet (contrôle) concerné et les paramètres de l'évènement.
Retour	Aucun.

p_CheckBox_CheckedChanged

Nom	`private void p_CheckBox_CheckedChanged(object sender, EventArgs e)`
Description	Fonction d'écoute de l'évènement : changement d'état (coché ou non) des contrôles « CheckBox » de la fenêtre « Element and sub-array components ». Affiche ou non le contenu de l'onglet courant selon l'état de la case à cocher.
Paramètres	Paramètres automatiques de capture d'évènement contenant l'objet (contrôle) concerné et les paramètres de l'évènement.
Retour	Aucun.

p_RadioButton_CheckedChanged

Nom	`private void p_RadioButton_CheckedChanged(object sender, EventArgs e)`
Description	Fonction d'écoute de l'évènement : changement d'état (coché ou non) des contrôles « RadioButton » de la fenêtre « Element and sub-array components ». Affiche ou non le champ « Time delay » dans les onglets « Phase » selon l'état des boutons radios.
Paramètres	Paramètres automatiques de capture d'évènement contenant l'objet (contrôle) concerné et les paramètres de l'évènement.
Retour	Aucun.

p_TextBox_KeyPress

Nom	`private void p_TextBox_KeyPress(object sender, KeyPressEventArgs e)`
Description	Fonction d'écoute sur l'évènement : appui sur une touche alors que le focus est sur un champ de saisie. Cette fonction ne fait rien si l'utilisateur a appuyé sur une autre touche que la touche « Entrée ». Vérifie le contenu du champ de saisie. Si le contenu ne correspond pas aux contraintes, affichage d'un message d'erreur sous forme de fenêtre « pop-up ».
Paramètres	Paramètres automatiques de capture d'évènement contenant l'objet (contrôle) concerné et les paramètres de l'évènement.
Retour	Aucun.

p_TextBox_Leave

Nom	`private void p_TextBox_Leave(object sender, EventArgs e)`
Description	Fonction d'écoute sur l'évènement : appui sur la touche « tabulation » ou clic sur un élément de la fenêtre alors que le focus est sur un champ de saisie. Vérifie le contenu du champ de saisie. Si le contenu ne correspond pas aux contraintes, un message d'erreur apparaît et le focus reste sur le champ de saisie concerné.
Paramètres	Paramètres automatiques de capture d'évènement contenant l'objet (contrôle) concerné et les paramètres de l'évènement.
Retour	Aucun.

p_TextBox_MouseEnter

Nom	`private void p_TextBox_MouseEnter(object sender, EventArgs e)`
Description	Fonction d'écoute sur l'évènement : passage de la souris au dessus d'un champ de saisie. Si le champ de saisie contient une chaîne valide, une « bulle » apparaîtra avec le détail du contenu de ce champ de saisie. Par exemple, si un champ contient la chaîne « 1:5 », la bulle affichera « 1,2,3,4,5». S'il s'agit d'un fichier, c'est le contenu du fichier qui sera affiché.
Paramètres	Paramètres automatiques de capture d'évènement contenant l'objet (contrôle) concerné et les paramètres de l'évènement.
Retour	Aucun.

PhaseChoice

Nom	`private void PhaseChoice()`
Description	Affiche ou non le champ de saisie « Time delay » dans les onglets « Phase » selon le bouton radio sélectionné.
Paramètres	Aucun.
Retour	Aucun.

IX- Fenêtre « Simulation method »

9.1 – Présentation générale

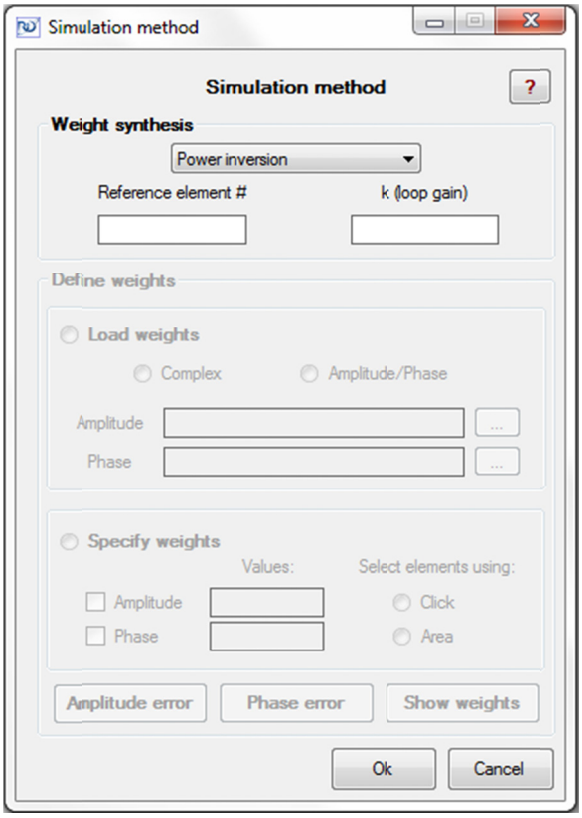


Figure 25 : Fenêtre « Simulation method »

Cette fenêtre permet de définir la méthode de simulation, en fonction du problème défini.

9.2 – Fonctionnement de la fenêtre

9.2.1 - Règles générales

Si aucun problème n'est défini (la fenêtre « Problem definition » n'est pas renseignée), l'utilisateur aura uniquement accès à la partie « Define weights ».
Il aura alors le choix entre importer les données depuis des fichiers (partie « Load weights ») ou les spécifier manuement (partie « Specify weights »). Pour ce dernier cas, l'utilisateur pourra sélectionner les éléments du réseau graphiquement en cliquant sur le bouton « show weights » (fonctionnalité à implémenter en langage Matlab).

A l'inverse, si l'utilisateur a défini un problème, il n'aura accès qu'à la partie « weights synthesis ». Il y différents cas possibles :

- Seuls le beamforming est défini, et ce pour une seule direction, sans SNR : la seule méthode de simulation possible sera « Applebaum ». Les champs « Reference element # » et « k (loop gain) » n'apparaîtront pas.

- Plusieurs directions de beamforming et éventuellement des valeurs de null-steering sont définies : la seule méthode de simulation possible sera « Conventionna BF ». Les champs « Reference element # » et « k (loop gain) » n'apparaîtront pas.
- Seules des valeurs de null-steering sont définies : deux méthodes seront possibles, « Aux. Elt. Power min. » et « power inversion ». Dans le premier cas, seul le champ « Reference element # » apparaîtra. Dans le second cas les deux champs sont visibles.

9.2.2 – Champ « Reference element # »

Type de variable : Un seul nombre entier.
Nom de la variable dans la structure de données locale : i_ReferenceElementNumber.
Nom du pointeur pour le contrôle : p_TextBox_ReferenceElement.
Contraintes particulières : Le nombre doit être compris entre 1 et le nombre d'élements dans le réseau d'antennes.

9.2.3 – Champ « k (loop gain) »

Type de variable : Un seul nombre réel.
Nom de la variable dans la structure de données locale : f_KloopGain.
Nom du pointeur pour le contrôle : p_TextBox_KloopGain.
Contraintes particulières : Le nombre doit être strictement positif.

9.2.4 – Champ « Amplitude » (Load weights)

Type de variable : Tableau de nombres réels.
Nom de la variable dans la structure de données locale : vec_f_AmplitudeWeights.
Nom du pointeur pour le contrôle : p_TextBox_FileName1.
Contraintes particulières : Ce champ doit contenir un nom de fichier valide, et dont le contenu est valide (colonne de valeurs réelles).

9.2.5 – Champ « Complex » (Load weights)

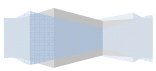
Type de variable : Tableau de tableaux de nombres réels.
Nom de la variable dans la structure de données locale : vec_cell_f_ComplexWeights.
Nom du pointeur pour le contrôle : p_TextBox_FileName1.
Contraintes particulières : Ce champ doit contenir un nom de fichier valide, et dont le contenu est valide (deux colonnes de valeurs réelles).

9.2.6 – Champ « Phase » (Load weights)

Type de variable : Tableau de nombres réels.
Nom de la variable dans la structure de données locale : vec_f_PhaseWeights.
Nom du pointeur pour le contrôle : p_TextBox_FileName2.
Contraintes particulières : Ce champ doit contenir un nom de fichier valide, et dont le contenu est valide (colonne de valeurs réelles).

9.2.7 – Champ « Amplitude » (Specify weights)

Type de variable : Un seul nombre réel.
Nom de la variable dans la structure de données locale : `f_AmplitudeSpecifiedWeight`.
Nom du pointeur pour le contrôle : p_TextBox_ValuesAmplitude.
Contraintes particulières : La valeur doit être comprise entre 0 et 1 (inclus).



9.2.8 – Champ « Phase » (Specify weights)

Type de variable : Un seul nombre réel.

Nom de la variable dans la structure de données locale : `f_PhaseSpecifiedWeight`.
Nom du pointeur pour le contrôle : p_TextBox_ValuesPhase.
Contraintes particulières : La valeur doit être positive ou nulle.

9.3- La classe « CsimulationMethodParameters »

Cette classe contient les variables de la fenêtre « Simulation method » qui seront mémorisées dans la fenêtre principale.
Ces variables sont les suivantes :

```
private string m_str_Kloopgain;
private string m_str_LoadValuesAmplitude;
private string m_str_LoadValuesPhase;
private string m_str_LoadValuesComplex;
private string m_str_AmplitudeValues;
private string m_str_PhaseValues;
private bool m_b_WeightSynthesis;
private bool m_b_LoadWeights;
private bool m_b_Complex;
private bool m_b_AmplitudePhase;
private bool m_b_SpecifyWeights;
private bool m_b_SpecifyAmplitude;
private bool m_b_SpecifyPhase;
private bool m_b_Click;
private bool m_b_Area;
private string m_str_WeightSynthesisMethod;
private string m_str_ReferenceElement;
```

Il s'agit des valeurs entrées dans les champs de saisie, des élements sélectionnés dans les menus déroulants et de l'état des boutons radio de l'interface.

9.4- Pointeurs de la fenêtre « Simulation method »

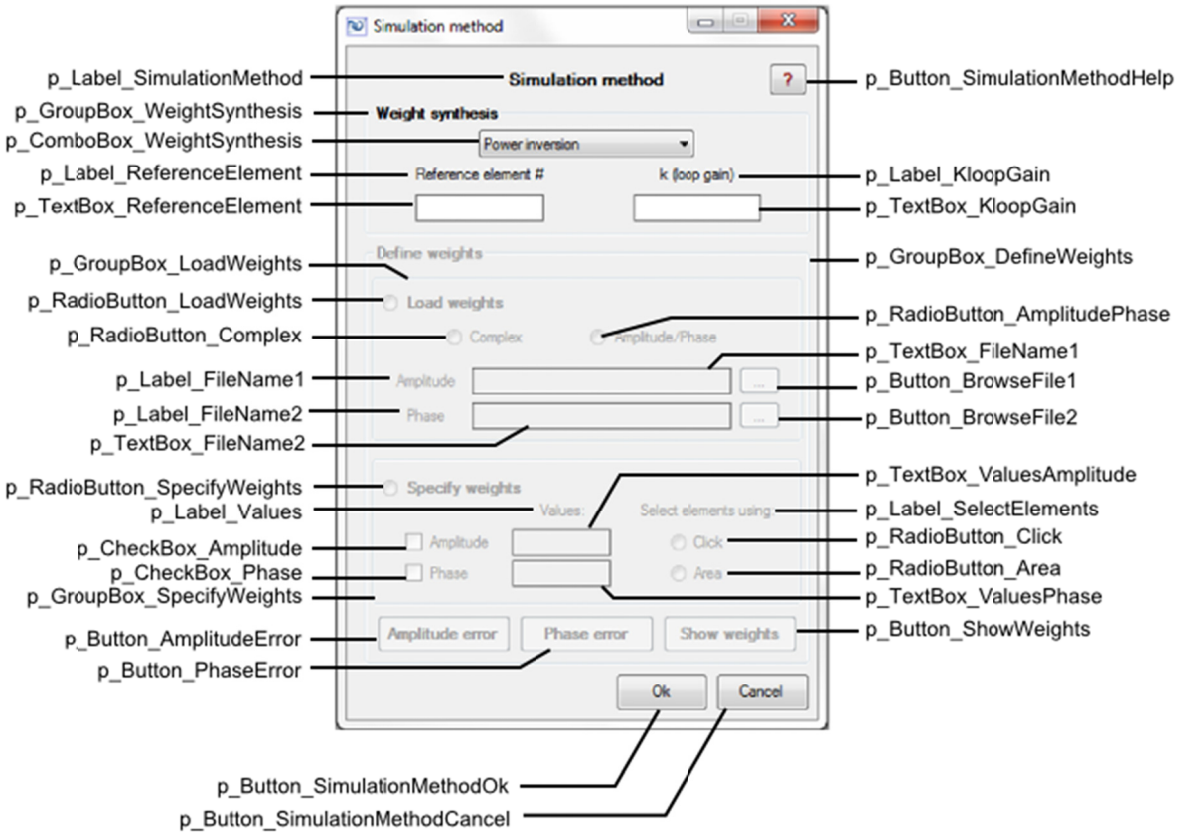


Figure 26 : pointeurs de la fenêtre « Simulation method »

9.5 – Méthodes de la classe « CsimulationMethod »

AmplitudeValue

Nom	`private string AmplitudeValue()`
Description	Vérifie les valeurs entrées dans le champ « Amplitude » du groupe « Specify weights » et met à jour les variables associées.
Paramètres	Aucun.
Retour	Chaîne de caractères vide ou contenant un message d'erreur.

CsimulationMethod (Constructeur)

Nom	`public CsimulationMethod(CsimulationMethodParameters SimulationMethodParameters_GUI, float[][] vec_Position_GUI, string[] vec_str_SynthesisMethod)`
Description	Initialise les contrôles et les variables de la fenêtre « Simulation method ».
Paramètres	**`CsimulationMethodParameters SimulationMethodParameters_GUI:`** Valeurs de tous les paramètres de la fenêtre « Simulation method » gardés en mémoire dans la fenêtre principale. **`float[][] vec_Position_GUI:`** Coordonnées de tous les élements du réseau d'antennes. **`string[] vec_str_SynthesisMethod:`** Liste des méthodes de synthèse possibles selon le problème défini dans la fenêtre « Problem definition ».
Retour	Aucun.

CsimulationMethod_FormClosing

Nom	`private void CsimulationMethod_FormClosing(object sender, FormClosingEventArgs e)`
Description	Fonction d'écoute de l'évènement : fermeture de la fenêtre « Simulation method » (appui sur le bouton « Ok », le bouton « cancel » ou la croix de fermeture en haut à droite). **Cas du bouton « Ok »** : si une erreur est présente sur la fenêtre, sa fermeture est annulée. Sinon, la fenêtre se ferme sans demander de confirmation à l'utilisateur. **Autres cas** : Une fenêtre de dialogue demande confirmation à l'utilisateur pour fermer la fenêtre ; la fenêtre se ferme donc ou non selon la réponse donnée.
Paramètres	Paramètres automatiques de capture d'évènement contenant l'objet (contrôle) concerné et les paramètres de l'évènement.
Retour	Aucun.

GetClassSimulationMethodParameters

Nom	`public CsimulationMethodParameters GetClassSimulationMethodParameters()`
Description	Retourne au GUI principal les valeurs de la fenêtre « Simulation method ».
Paramètres	Aucun.
Retour	Classe `CsimulationMethodParameters` contenant les valeurs des paramètres.

LoadWeights

Nom	`private void LoadWeights()`
Description	Gère l'affichage du groupe « Load weights » en fonction de l'état du bouton radio correspondant.
Paramètres	Aucun.
Retour	Aucun.

LoadWeightsChoice

Nom	`private void LoadWeightsChoice()`
Description	Gère l'affichage des champs de saisie « Amplitude » et « Phase » du groupe « Load weights » en fonction de l'état du bouton radio correspondant (« Complex » ou « Amplitude/Phase »).
Paramètres	Aucun.
Retour	Aucun.

LoadWeightsFile

Nom	`private string LoadWeightsFile(string str_FileType)`
Description	Vérifie les valeurs contenues dans un fichier importé par l'utilisateur (groupe « Load weights »).
Paramètres	**`string str_FileType:`** `Cette chaîne peut prendre deux valeurs : «Complex» ou «Phase». Cela permet de différencier le type de fichier à vérifier et le champ de saisie concerné.`
Retour	Chaîne de caractères vide ou contenant un message d'erreur.

LoopGain

Nom	`private string LoopGain()`
Description	Vérifie les valeurs entrées dans le champ « k (loop gain) » du groupe « Specify weights » et met à jour les variables associées.
Paramètres	Aucun.
Retour	Chaîne de caractères vide ou contenant un message d'erreur.

OkPushButton

Nom	`private string OkPushButton()`
Description	Fonction appelée dans `p_Button_SimulationMethodOk_Click`, c'est-à-dire lorsque l'utilisateur a appuyé sur le bouton « Ok ». Vérifie que les champs obligatoires ne sont pas vides.
Paramètres	Aucun.
Retour	Chaîne de caractères vide ou contenant un message d'erreur.

p_Button_Click

Nom	`private void p_Button_Click(object sender, EventArgs e)`
Description	Fonction d'écoute de l'évènement : appui sur l'un des boutons de la fenêtre « Problem definition ». **Boutons « Browse file 1 » et « Browse file 2 » :** Ouvre un navigateur de fichier et vérifie le contenu du fichier choisi par l'utilisateur. **Boutons « Amplitude error », « phase error » et « show weights » :** pour le moment, aucun code n'est effectué. Il faudra ajouter les fonctions Matlab nécessaires.
Paramètres	Paramètres automatiques de capture d'évènement contenant l'objet (contrôle) concerné et les paramètres de l'évènement.
Retour	Aucun.

p_Button_SimulationMethodOk_Click

Nom	`private void p_Button_SimulationMethodOk_Click(object sender, EventArgs e)`
Description	Fonction d'écoute de l'évènement : appui sur le bouton « Ok ». Vérifie des valeurs de la fenêtre et met à jour la valeur booléenne qui annule la demande de confirmation à l'utilisateur pour fermer la fenêtre. Après l'appel de cette fonction, le programme apelle la fonction « `CsimulationMethod_FormClosing » (fenêtre modale).`
Paramètres	Paramètres automatiques de capture d'évènement contenant l'objet (contrôle) concerné et les paramètres de l'évènement.
Retour	Aucun.

p_CheckBox_CheckedChanged

Nom	`private void p_CheckBox_CheckedChanged(object sender, EventArgs e)`
Description	Fonction d'écoute de l'évènement : changement d'état (coché ou non) des contrôles « CheckBox » de la fenêtre « Simulation method ». Active ou non le champ de saisie « Amplitude » ou « Phase » en fonction de l'état de la case à cocher correspondante.
Paramètres	Paramètres automatiques de capture d'évènement contenant l'objet (contrôle) concerné et les paramètres de l'évènement.
Retour	Aucun.

p_ComboBox_WeightSynthesis_SelectedIndexChanged

Nom	`private void p_ComboBox_WeightSynthesis_SelectedIndexChanged(object sender, EventArgs e)`
Description	Fonction d'écoute de l'évènement : changement de sélection dans le menu déroulant « Weight synthesis ». Gère l'affichage des champs de saisie « Reference element # » et « k (loop gain) » en fonction du type de méthode choisi.
Paramètres	Paramètres automatiques de capture d'évènement contenant l'objet (contrôle) concerné et les paramètres de l'évènement.
Retour	Aucun.

p_RadioButton_CheckedChanged

Nom	`private void p_RadioButton_CheckedChanged(object sender, EventArgs e)`
Description	Fonction d'écoute de l'évènement : changement d'état (coché ou non) des contrôles « RadioButton » de la fenêtre « Simulation method ». Gère l'affichage de la fenêtre selon l'état des boutons radio.
Paramètres	Paramètres automatiques de capture d'évènement contenant l'objet (contrôle) concerné et les paramètres de l'évènement.
Retour	Aucun.

p_TextBox_KeyPress

Nom	`private void p_TextBox_KeyPress(object sender, KeyPressEventArgs e)`
Description	Fonction d'écoute sur l'évènement : appui sur une touche alors que le focus est sur un champ de saisie. Cette fonction ne fait rien si l'utilisateur a appuyé sur une autre touche que la touche « Entrée ». Vérifie le contenu du champ de saisie. Si le contenu ne correspond pas aux contraintes, affichage d'un message d'erreur sous forme de fenêtre « pop-up ».
Paramètres	Paramètres automatiques de capture d'évènement contenant l'objet (contrôle) concerné et les paramètres de l'évènement.
Retour	Aucun.

p_TextBox_Leave

Nom	`private void p_TextBox_Leave(object sender, EventArgs e)`
Description	Fonction d'écoute sur l'évènement : appui sur la touche « tabulation » ou clic sur un élément de la fenêtre alors que le focus est sur un champ de saisie. Vérifie le contenu du champ de saisie. Si le contenu ne correspond pas aux contraintes, un message d'erreur apparaît et le focus reste sur le champ de saisie concerné.
Paramètres	Paramètres automatiques de capture d'évènement contenant l'objet (contrôle) concerné et les paramètres de l'évènement.
Retour	Aucun.

p_TextBox_MouseEnter

Nom	`private void p_TextBox_MouseEnter(object sender, EventArgs e)`
Description	Fonction d'écoute sur l'évènement : passage de la souris au dessus d'un champ de saisie. Si le champ de saisie contient une chaîne valide, une « bulle » apparaîtra avec le détail du contenu de ce champ de saisie. Par exemple, si un champ contient la chaîne « 1:5 », la bulle affichera « 1,2,3,4,5». S'il s'agit d'un fichier, c'est le contenu du fichier qui sera affiché.
Paramètres	Paramètres automatiques de capture d'évènement contenant l'objet (contrôle) concerné et les paramètres de l'évènement.
Retour	Aucun.

PhaseValue

Nom	`private string PhaseValue()`
Description	Vérifie les valeurs entrées dans le champ « Phase » du groupe « Specify weights » et met à jour les variables associées.
Paramètres	Aucun.
Retour	Chaîne de caractères vide ou contenant un message d'erreur.

ReferenceElement

Nom	`private string ReferenceElement()`
Description	Vérifie les valeurs entrées dans le champ « Reference element # » du groupe « Weight synthesis » et met à jour les variables associées.
Paramètres	Aucun.
Retour	Chaîne de caractères vide ou contenant un message d'erreur.

SpecifyWeights

Nom	`private void SpecifyWeights()`
Description	Gère l'affichage du groupe « Specify weights » en fonction de l'état du bouton radio correspondant.
Paramètres	Aucun.
Retour	Aucun.

SpecifyWeightsChoice

Nom	`private void SpecifyWeightsChoice()`
Description	Gère l'affichage des champs de saisie « Amplitude » et « Phase » du groupe « Specify weights» en fonction de l'état des cases à cocher correspondantes.
Paramètres	Aucun.
Retour	Aucun.

SynthesisMethod

Nom	`private void SynthesisMethod()`
Description	Gère l'affichage du groupe « Synthesis method » en fonction de l'élément choisi dans le menu déroulant.
Paramètres	Aucun.
Retour	Aucun.

UpdateAmplitudeAndPhaseError

Nom	`private void UpdateAmplitudeAndPhaseError()`
Description	Active ou non les boutons « Amplitude error » et « Phase error » selon les valeurs dans les champs de saisie « amplitude » et « phase ».
Paramètres	Aucun.
Retour	Aucun.

UpdateShowWeightsButtonState

Nom	`private void UpdateShowWeightsButtonState()`
Description	Active ou non le bouton « Show weights » selon les valeurs dans les champs de saisie « amplitude » et « phase ».
Paramètres	Aucun.
Retour	Aucun.

X- Fenêtre « Compute options »

10.1 – Présentation générale

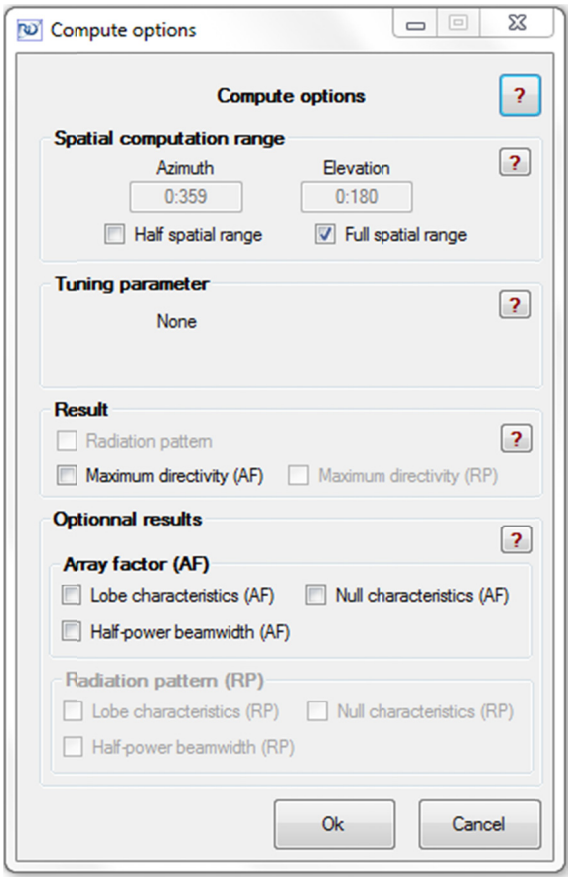


Figure 27 : Fenêtre « Compute options »

Cette fenêtre permet de définir les options de simulation pour le problème défini.

10.2 – Fonctionnement de la fenêtre

10.2.1 - Règles générales

L'activation ou non des différentes cases à cocher dépend des autres fenêtres de l'interface. Toutes les case à cocher relatives au modèle de radiation (« Radiation pattern ») sont désactivées si aucun ERP n'a été défini dans la fenêtre « Define ERP ».
Si la case « Half spatial range » est cochée, les champs de saisie « azimuth » et « elevation » prendront respectivements les valeurs « 0:359 » et « 0:90 », ce qui représente un demi-espace. Si la case « Full spatial range » est cochée (il n'est pas possible de cocher les deux cases en même temps), les valeurs seront « 0:359 » et « 0:180 », c'est-à-dire l'espace complet.
Si aucune de ces deux cases n'est cochée, les champs deviennent disponibles à la saisie et l'utilisateur peut entrer les valeurs de son choix.

10.2.2 – Champ « Azimuth »

Type de variable : Tableau de nombres réels.
Nom de la variable dans la structure de données locale : vec_f_SpatialRangeAzimuth.
Nom du pointeur pour le contrôle : p_TextBox_Azimuth.
Contraintes particulières : Aucune contrainte particulière hormis la syntaxe pour indiquer plusieurs valeurs (spération par des virgules ou utilisation des deux points « : » pour signifier des plages de valeurs).

10.2.3 – Champ « Elevation»

Type de variable : Tableau de nombres réels.
Nom de la variable dans la structure de données locale : vec_f_SpatialRangeElevation.
Nom du pointeur pour le contrôle : p_TextBox_Elevation.
Contraintes particulières : Aucune contrainte particulière hormis la syntaxe pour indiquer plusieurs valeurs (spération par des virgules ou utilisation des deux points « : » pour signifier des plages de valeurs).

10.3- La classe « CcomputeOptionsParameters »

Cette classe contient les variables de la fenêtre « Compute options » qui seront mémorisées dans la fenêtre principale.
Ces variables sont les suivantes :

```
private string m_str_SpatialRangeAzimuth;
private string m_str_SpatialRangeElevation;
private bool m_b_HalfSpatialRange;
private bool m_b_FullSpatialRange;
private bool m_b_RadiationPattern;
private bool m_b_MaximumDirectivityAF;
private bool m_b_LobeCharacteristicsAF;
private bool m_b_NullCharacteristicsAF;
private bool m_b_HalfPowerBeamWidthAF;
private bool m_b_MaximumDirectivityRP;
private bool m_b_LobeCharacteristicsRP;
private bool m_b_NullCharacteristicsRP;
private bool m_b_HalfPowerBeamWidthRP;
```

Les deux chaînes de caractères ont en mémoire le contenu des deux champs de saisie “Azimuth” et “Elevation”. Les variables booléennes contiennent l’état de toutes les cases à cocher de la fenêtre.

10.4- Pointeurs de la fenêtre « Compute options »

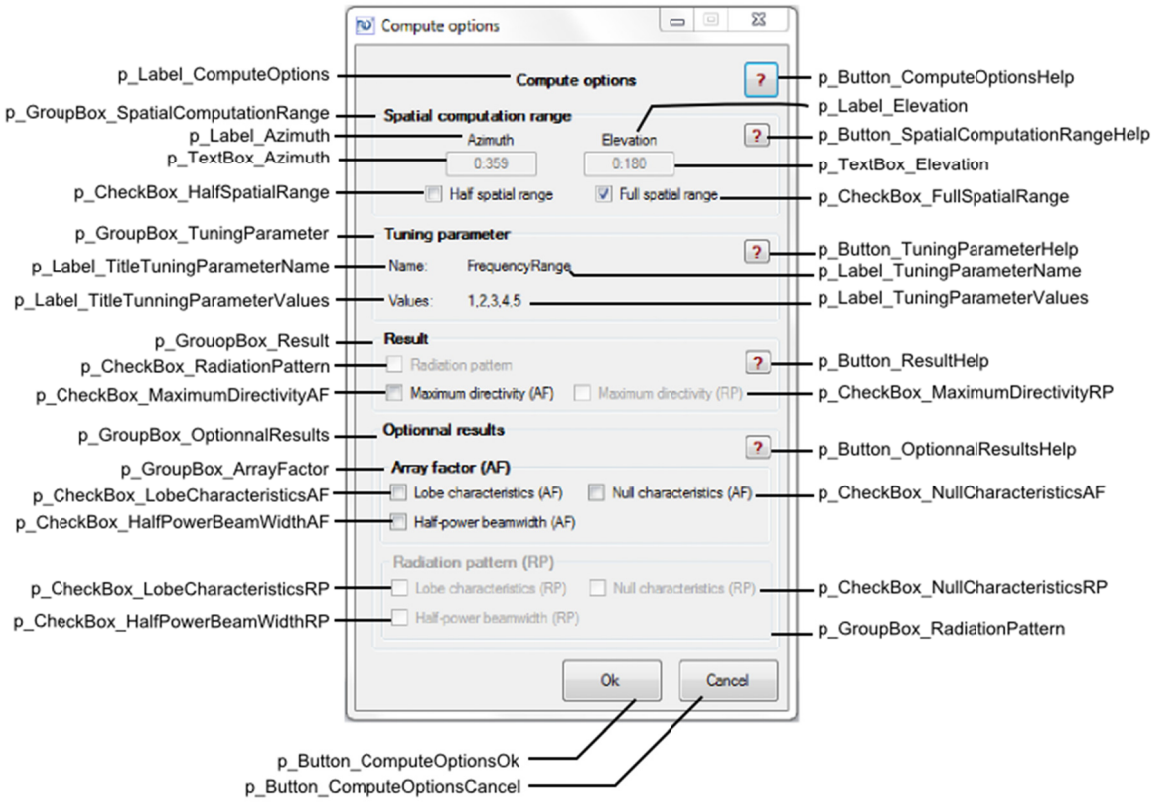


Figure 28 : pointeurs de la fenêtre « Compute options »

10.5 – Méthodes de la classe « CcomputeOptions »

CcomputeOptions (Constructeur)

Nom	`public CcomputeOptions(CcomputeOptionsParameters ComputeOptionsParameters_GUI, CdefineERPParameters DefineErpParameters_GUI, CtuningParameter tuningParameter_GUI)`
Description	Initialise les contrôles et les variables de la fenêtre « Compute options ».
Paramètres	**`CcomputeOptionsParameters ComputeOptionsParameters_GUI:`** Valeurs de tous les paramètres de la fenêtre « Compute options » gardés en mémoire dans la fenêtre principale. **`CdefineERPParameters DefineErpParameters_GUI:`** Paramètres de la fenêtre « Define ERP », utilisés pour connaître l'état des cases à cocher de la fenêtre. **`CtuningParameter tuningParameter_GUI:`** Paramètres de tuning, utilisés pour l'affichage dans la partie « Tuning parameter » de la fenêtre.
Retour	Aucun.

CcomputeOptions_FormClosing

Nom	`private void CcomputeOptions_FormClosing(object sender, FormClosingEventArgs e)`
Description	Fonction d'écoute de l'évènement : fermeture de la fenêtre « Compute options » (appui sur le bouton « Ok », le bouton « cancel » ou la croix de fermeture en haut à droite). **Cas du bouton « Ok »** : si une erreur est présente sur la fenêtre, sa fermeture est annulée. Sinon, la fenêtre se ferme sans demander de confirmation à l'utilisateur. **Autres cas** : Une fenêtre de dialogue demande confirmation à l'utilisateur pour fermer la fenêtre ; la fenêtre se ferme donc ou non selon la réponse donnée.
Paramètres	Paramètres automatiques de capture d'évènement contenant l'objet (contrôle) concerné et les paramètres de l'évènement.
Retour	Aucun.

CheckBoxFullSpatialRange

Nom	`private void CheckBoxFullSpatialRange()`
Description	Gère l'état des champs de saisie « Azimuth » et « Elevation » lorque la case « Full spatial range » est cochée.
Paramètres	Aucun.
Retour	Aucun.

CheckBoxHalfSpatialRange

Nom	`private void CheckBoxHalfSpatialRange()`
Description	Gère l'état des champs de saisie « Azimuth » et « Elevation » lorque la case « Half spatial range » est cochée.
Paramètres	Aucun.
Retour	Aucun.

ComputationRange_Az

Nom	`private string ComputationRange_Az()`
Description	Vérifie les valeurs entrées dans le champ « Azimuth » du groupe « Spatial computation range» et met à jour les variables associées.
Paramètres	Aucun.
Retour	Chaîne de caractères vide ou contenant un message d'erreur.

ComputationRange_El

Nom	`private string ComputationRange_El()`
Description	Vérifie les valeurs entrées dans le champ « Elevation » du groupe « Spatial computation range» et met à jour les variables associées.
Paramètres	Aucun.
Retour	Chaîne de caractères vide ou contenant un message d'erreur.

GetClassComputeOptionsParameters

Nom	`public CcomputeOptionsParameters GetClassComputeOptionsParameters()`
Description	Retourne au GUI principal les valeurs de la fenêtre « Compute options ».
Paramètres	Aucun.
Retour	Classe `CcomputeOptionsParameters contenant les valeurs des paramètres.`

HalfPowerWidthIsEnable

Nom	`private void HalfPowerWidthIsEnable()`
Description	Gère l'état des cases à cocher « Full spatial range » et « Half spatial range » afin qu'elle ne puissent pas être cochées simultanément.
Paramètres	Aucun.
Retour	Aucun.

OkPushButton

Nom	`private string OkPushButton()`
Description	Fonction appelée dans `p_Button_ComputeOptionsOk_Click`, c'est-à-dire lorsque l'utilisateur a appuyé sur le bouton « Ok ». Vérifie que les champs obligatoires ne sont pas vides.
Paramètres	Aucun.
Retour	Chaîne de caractères vide ou contenant un message d'erreur.

p_Button_ComputeOptionsOk_Click

Nom	`private void p_Button_ComputeOptionsOk_Click(object sender, EventArgs e)`
Description	Fonction d'écoute de l'évènement : appui sur le bouton « Ok ». Vérifie des valeurs de la fenêtre et met à jour la valeur booléenne qui annule la demande de confirmation à l'utilisateur pour fermer la fenêtre. Après l'appel de cette fonction, le programme apelle la fonction « `CcomputeOptions_FormClosing » (fenêtre modale).`
Paramètres	Paramètres automatiques de capture d'évènement contenant l'objet (contrôle) concerné et les paramètres de l'évènement.
Retour	Aucun.

p_CheckBox_CheckedChanged

Nom	`private void p_CheckBox_CheckedChanged(object sender, EventArgs e)`
Description	Fonction d'écoute de l'évènement : changement d'état (coché ou non) des contrôles « CheckBox » de la fenêtre « Compute options ». Gère l'état des deux champs de saisie « Azimuth » et « Elevation » en fonction de l'état des cases à cocher.
Paramètres	Paramètres automatiques de capture d'évènement contenant l'objet (contrôle) concerné et les paramètres de l'évènement.
Retour	Aucun.

p_TextBox_KeyPress

Nom	`private void p_TextBox_KeyPress(object sender, KeyPressEventArgs e)`
Description	Fonction d'écoute sur l'évènement : appui sur une touche alors que le focus est sur un champ de saisie. Cette fonction ne fait rien si l'utilisateur a appuyé sur une autre touche que la touche « Entrée ». Vérifie le contenu du champ de saisie. Si le contenu ne correspond pas aux contraintes, affichage d'un message d'erreur sous forme de fenêtre « pop-up ».
Paramètres	Paramètres automatiques de capture d'évènement contenant l'objet (contrôle) concerné et les paramètres de l'évènement.
Retour	Aucun.

p_TextBox_Leave

Nom	`private void p_TextBox_Leave(object sender, EventArgs e)`
Description	Fonction d'écoute sur l'évènement : appui sur la touche « tabulation » ou clic sur un élément de la fenêtre alors que le focus est sur un champ de saisie. Vérifie le contenu du champ de saisie. Si le contenu ne correspond pas aux contraintes, un message d'erreur apparaît et le focus reste sur le champ de saisie concerné.
Paramètres	Paramètres automatiques de capture d'évènement contenant l'objet (contrôle) concerné et les paramètres de l'évènement.
Retour	Aucun.

p_TextBox_MouseEnter

Nom	`private void p_TextBox_MouseEnter(object sender, EventArgs e)`
Description	Fonction d'écoute sur l'évènement : passage de la souris au dessus d'un champ de saisie. Si le champ de saisie contient une chaîne valide, une « bulle » apparaîtra avec le détail du contenu de ce champ de saisie. Par exemple, si un champ contient la chaîne « 1:5 », la bulle affichera « 1,2,3,4,5». S'il s'agit d'un fichier, c'est le contenu du fichier qui sera affiché.
Paramètres	Paramètres automatiques de capture d'évènement contenant l'objet (contrôle) concerné et les paramètres de l'évènement.
Retour	Aucun.

XI- Intégration du code Matlab en C#

Le code Matlab peut être compilé sous forme de 'bibliothèque' à l'aide du module Builder NE pour être utilisé dans une application .NET. Les fonctions Matlab sont habituellement compilées au sein d'une ou plusieurs classes et le fichier obtenu est de type « Dynamic-link library » (DLL). Pour utiliser une fonction compilée, il faut tout d'abord ajouter le fichier « .dll » correspond à la classe compilée dans le dossier « référence » du projet C#. Ensuite, il faut spécifier que ce fichier « .dll » sera utilisées dans une classe C#. Ceci est fait au début de la classe à l'aide de la commande « using ». Par exemple, une fonction Matlab « MaFonctionMatlab.m » a été compilée dans la classe CmaFonctionMatlab et le fichier « .dll » résultant est « MaFonctionMatlabCompilee.dll ». Le fichier « MaFonctionMatlabCompilee.dll » doit être ajouté comme référence du projet, et la commande 'using maFonctionMatlabCompilee' doit figurer au début de la classe C# utilisée.

Pour utiliser la fonction Matlab compilée dans le code C#, il est également nécessaire d'ajouter le fichier « MWArray.dll » fourni avec le module Builder NE, ainsi qu'ajouter les commandes 'using Mathworks.MATLAB.NET.Utility' et 'using Mathworks.MATLAB.NET.Arrays' au début de la classe C#. Ceci permet de charger les 'bibliothèques' permettant de convertir les variables du code C# au format Matlab et vice-versa [5].

Pour appeler la fonction Matlab en tant que telle dans le code C#, les règles générales qui s'appliquent sont les suivantes :

- Les variables d'entrée de la fonction appelée sont celles de la fonction Matlab, précédé du nombre de variables de sortie,
- Les variables d'entrée doivent être converties au format Matlab en utilisant le « casting » propre au language C#,
- Les variables de retour sont stockées dans un tableau de type MWArray[]. Ceci est due au fait qu'un seul argument de retour est possible pour une fonction en C#. Pour retrouver les paramètres de retour sous forme de variables individuelles, il est nécessaire de lire les données dans le tableau de type MWArray[].

Tous les détails sur la compilation du code Matlab et son intégration en C# se trouvent à la page internet suivante :

http://www.mathworks.com/products/netbuilder/

Exemple : Calcul de la transformée de Fourier

L'intitulé de la function Matlab est le suivant :

```
function [fftData, freq, powerSpect] = ComputeFFT(data, interval)
```

La variable d'entrée 'data' est un vecteur de réels, et 'interval' est un réel positif représentant l'intervalle temporel entre deux échantillons. La variable de sortie 'fftData' est un vecteur de complexes, 'freq' est un vecteur de réels positifs, et 'powerSpect' est un vecteur de réels positifs.

Une fois compilée dans la classe CsignalAnalyzer, cette fonction s'utilise de la manière suivante en C# :

```
float[] data = new float[];

const double interval= 0.01;

// Remplissage du tableau 'data' (non détaillé ici)

CsignalAnalyzer signalAnalyzer= new CsignalAnalyzer();

MWArray[] argsOut= signalAnalyzer.computeFFT(3, (MWArray)data,
interval);
```

La première ligne de code déclare la variable 'data' de type tableau de réels (1 dimension), la deuxième déclare la variable 'interval' de type double. Le remplissage du tableau 'data' n'est pas détaillé ici. La troisième ligne de code crée une instance de la classe CsignalAnalyzer, la quatrième ligne appelle la fonction computeFFT. La première variable d'entrée '3' spécifie qu'il y a trois variables de sortie. La deuxième variable d'entrée correspond aux données utilisées pour calculer la transformée de Fourier. '(MWArray)' devant 'data' permet de convertir la variable C# au format Matlab. La troisième variable d'entrée 'interval' est convertie de façon implicite. La variable argOut est de longueur 3, et elle contient les variables de sortie au format Matlab.

Conclusion

Le développement de l'interface C# pour l'outil PAASoM a été présenté dans ce document, avec pour objectif principal de permettre à un programmeur de reprendre facilement le code et de pouvoir le modifier et le compléter. L'implémentation en C# permettra à terme d'augmenter sensiblement les possibilités de développement de PAASoM.

Tout d'abord, les différents concepts utilisés pour l'implémentation de l'interface ont été décrits. En particulier, les mécanismes utilisés pour mémoriser les données saisies dans les interfaces sont les suivants :

- Le format des valeurs entrées dans les interfaces est validé pour chaque fenêtre,
- Une classe contenant des champs privés est définie pour chaque interface. De cette manière, les variables sont à jour en permanence, et les modifications des champs sont faites de façon contrôlées,
- Pour limiter les dépendances entre les fenêtres, une seule sous-fenêtre peut être ouverte à la fois.

Ensuite, les classes correspondant aux différentes interfaces de PAASoM ont été détaillées. Pour chacune d'entre elles, les fonctions communes que l'on retrouve sont l'initialisation de la fenêtre, l'écoute de l'appuie sur la touche entrée du clavier, la sortie d'un champ de saisie, le passage de la souris sur un champ de saisie et la fermeture de la fenêtre.

Enfin, la procédure à suivre pour intégrer le code Matlab en C# a été présenter. Les liens avec le code Matlab doivent être mis en place grâce au module Builder NE pour compiler le code Matlab sous forme de 'bibliothèques' utilisables en C#.

Concernant les perspectives de ce projet, une sous-fenêtre reste à implémenter (« Plotting options »), l'ergonomie de l'interface principale doit être améliorée, et les fonctions Matlab doivent être incorporées pour permettre d'effectuer l'analyse des réseaux d'antennes à déphasage en fonction des données saisies dans l'interface.

References

[1] PAASoM

Acronyme pour “Phased Array Antenna Simulation tool on Matlab”.
Une rapide description de ce logiciel est disponible sur le site internet de RDDC Ottawa: http://www.ottawa.drdc-rddc.gc.ca/html/gecn_238-fra.html. Pour plus de détails, se référer aux documents suivants (en anglais) :

- M. Clénet, M. Caillet, and Y.M.M. Antar, “A phased array simulation tool for synthesized array factors having multiple beams and/or multiple nulls”, 2010 IEEE International Symposium on Phased Array Systems and Technology, p. 819-826.
- S. Rogojan and M. Clénet, “PAASoM User Guide”, DRDC Ottawa SL 2006-213, October 2006.

[2] Visual studio

Microsoft Visual Studio est une suite de logiciels de développement pour Windows conçue par Microsoft. La dernière version s'appelle **Visual Studio 2010**.
Visual Studio est un ensemble complet d'outils de développement permettant de générer des applications Web ASP.NET, des Services Web XML, des applications bureautiques et des applications mobiles. Visual Basic, Visual C++, Visual C# et Visual J# utilisent tous le même environnement de développement intégré (IDE, Integrated Development Environment), qui leur permet de partager des outils et facilite la création de solutions faisant appel à plusieurs langages. Par ailleurs, ces langages permettent de mieux tirer parti des fonctionnalités duFramework .NET, qui fournit un accès à des technologies clés simplifiant le développement d'applications Web ASP et de Services Web XML grâce à Visual Web Developer. »
Source: http://en.wikipedia.org/wiki/Microsoft_Visual_Studio
http://www.microsoftstore.ca

[3] Matlab

MATrix LABoratory.
Matlab est à la fois un environnement de développement et un langage de programmation spécialisé dans les calculs numériques.
http://www.mathworks.com/products/matlab/

[4] Microsoft « .NET »

Microsoft .NET est le nom donné à un ensemble de produits et de technologies informatiques de l'entreprise Microsoft pour rendre des applications facilement portables sur Internet. Le but est de fournir un serveur web local permettant de gérer des services et évitant d'externaliser des données privées sur un service web de stockage ou un hébergement web tiers.
Source: http://en.wikipedia.org/wiki/.NET_Framework

[5] Mathworks MWArray assembly

http://www.mathworks.com/help/toolbox/dotnetbuilder/MWArrayAPI/HTML/index.html

Table des illustrations

Figure 1 : fichiers contenus dans le projet PAASoM C Sharp 6
Figure 2 : fichiers composant une classe de type « Windows form » 7
Figure 3: architecture logicielle générale 8
Figure 4 : Passage des variables dans l'interface PAASoM 10
Figure 5 : L'environnement de Visual Studio Express 2010 14
Figure 6 : Création d'interfaces avec Visual Studio Express 2010 15
Figure 7 : Ajout d'événement à un composant graphique 15
Figure 8 : Fenêtre principale 17
Figure 9 : Pointeurs de la fenêtre « PAASoM » 17
Figure 10 : La fenêtre « Frequency » 21
Figure 11 : pointeurs de la fenêtre «Frequency » 22
Figure 12 : Fenêtre « Problem definition » 26
Figure 13 : pointeurs de la fenêtre « Problem definition » 29
Figure 14 : La fenêtre « Array characteristics » 37
Figure 15 : pointeurs de la fenêtre « Array characteristics » 41
Figure 16 : Fenêtre « Define ERP » 49
Figure 17 : pointeurs de la fenêtre « Define ERP » 52
Figure 18 : La fenêtre « Amplitude distribution » 62
Figure 19 : pointeurs de la fenêtre « Amplitude distribution » 63
Figure 20 : La fenêtre « Element and sub-array components » 68
Figure 21 : pointeurs de la fenêtre « Element and sub-array components », onglet « Element – amplitude » 71
Figure 22 : pointeurs de la fenêtre « Element and sub-array components », onglet « Element – phase shifter » 71
Figure 23 : pointeurs de la fenêtre « Element and sub-array components », onglet « Sub-array – amplitude » 72
Figure 24 : pointeurs de la fenêtre « Element and sub-array components », onglet « Sub-array – phase shifter » 72
Figure 25 : Fenêtre « Simulation method » 79
Figure 26 : pointeurs de la fenêtre « Simulation method » 82
Figure 27 : Fenêtre « Compute options » 89
Figure 28 : pointeurs de la fenêtre « Compute options » 91

FICHE DE CONTRÔLE DU DOCUMENT

(Indiquer la classification de sécurité du titre, du résumé et des renseignements d'indexation si tout le document est classifié.)

1. DEMANDEUR (Le nom et l'adresse de l'organisation qui a préparé le document. Les organisations pour lesquelles le document a été préparé, p. ex., le Centre qui commande un rapport à un entrepreneur ou l'organisme à l'origine du document doivent figurer à la section 8.)

Collège Militaire Royal du Canada
CP 17000, Succursale Forces
Kingston, Ontario, Canada K7K 7B4

2. CLASSIFICATION DE SÉCURITÉ (Classification de sécurité globale du document, y compris les notices d'avertissement spéciales, s'il y a lieu.)

SANS CLASSIFICATION
(MARCHANDISES NON CONTRÔLÉES)
CDM A
RÉVISION : GCEC JUIN 2010

3. TITRE (Titre au long du document qui figure sur la page titre. La classification du titre devrait être indiquée à l'aide de l'abréviation voulue [S, C, DR ou SC], entre parenthèses, après le titre.)

Développement d'une interface homme-machine en C-sharp pour PAASoM :

4. AUTEURS (Nom de famille, puis initiales – Ne pas mettre le grade, le titre, etc.)

Delahaye, M.; Rialet, D.; Caillet, M.

5. DATE DE PUBLICATION (Mois et année de publication du document.)	6a. NOMBRE DE PAGES (Nombre total de pages contenant des renseignements, y compris les annexes, les appendices, etc.)	6b. NOMBRE DE RÉFÉRENCES (Nombre total de références citées dans le document.)
septembre 2012	110	5

7. NOTES DESCRIPTIVES (Catégorie du document, p. ex., rapport, note ou mémorandum technique. Indiquer s'il y a lieu s'il s'agit d'un rapport provisoire, p. ex., d'un rapport d'étape, d'un rapport sommaire, d'un rapport annuel ou d'un rapport final. Si le document porte sur une période précise, indiquer les dates correspondantes.)

Rapport de contrat

8. RESPONSABLE (Nom et adresse du bureau de projet ou du laboratoire du Ministère qui est responsable du travail de recherche et de développement.)

R & D pour la défense Canada – Ottawa
3701, avenue Carling
Ottawa (Ontario) K1A 0Z4
Canada

9a. No DU PROJET OU DE LA SUBVENTION (Indiquer s'il y a lieu le numéro du projet ou de la subvention de recherche et de développement dans le cadre duquel le document a été rédigé. Préciser s'il s'agit d'un projet ou d'une subvention.)	9b. No DU CONTRAT (Indiquer s'il y a lieu le numéro du contrat dans le cadre duquel le document a été rédigé.)
15en01	C1410FE121

10a. No DE DOCUMENT DU DEMANDEUR (Numéro de document officiel par lequel le demandeur désigne le document. Ce numéro doit être propre au document.)	10b. AUTRES Nos DE DOCUMENT (Autres numéros qui pourraient avoir été attribués au document par le demandeur ou le responsable.)
	DRDC Ottawa CR 2012-116

11. DISPONIBILITÉ DU DOCUMENT (Limites à la diffusion du document autres que celles qu'impose la classification de sécurité.)

Diffusion illimitée

12. ANNONCE DU DOCUMENT (Restrictions imposées à l'annonce du document. Elles correspondent normalement à la disponibilité du document (11). Toutefois, si une diffusion plus large que celle qui a été prévue au par. 11 peut être envisagée, un plus large auditoire d'annonce peut être retenu.)

Diffusion illimitée

13. RÉSUMÉ (Résumé succinct du document. Le résumé peut paraître ailleurs dans le corps du document. Il est éminemment souhaitable que le résumé d'un document classifié soit sans classification. Chaque paragraphe du résumé doit commencer par une indication de la classification de sécurité des renseignements qu'il contient [sauf si tout le document est sans classification]; utiliser les lettres voulues : S, C, DR ou SC. Il n'est pas nécessaire de mettre ici le résumé dans les deux langues officielles, sauf si le document est bilingue.)

Ce document présente le développement d'une interface homme-machine (IHM) en langage C sharp (C#) pour PAASoM (outil de simulation de réseau d'antennes à déphasage). PAASoM a initialement été implémenté avec Matlab, et les composants graphiques et la gestion d'événements associée à son IHM limitaient jusqu'à présent son développement. Il a donc été décidé d'utiliser un langage plus performant pour l'IHM de PAASoM, tout en conservant le code Matlab pour la simulation en tant que telle. Ceci est rendu possible grâce au module Matlab Builder NE qui permet de compiler le code Matlab sous forme de bibliothèques dynamiques 'DLL' et d'invoquer ces dernières en C#. L'environnement utilisé pour implémenter l'interface C# est Visual Basic Express 2010.

This report presents the development of a graphical user interface (GUI) for PAASoM (Phased Array Antenna Simulation Tool) using the programming language C Sharp (C#). PAASoM was originally implemented using Matlab, and graphic components and event handling of its GUI were limiting further development of the tool. Thus, it was decided to use a highly capable language for the PAASoM GUI, and keep the Matlab code that achieves the simulation itself. This was possible thanks to the Matlab Builder NE toolbox which allows compiling Matlab code as a dynamic-link library (DLL) and invoking those latter in C#. The environment used to implement the C# interface is Visual Basic Express 2010.

14. MOTS-CLÉS, DESCRIPTEURS ou IDENTIFICATEURS (Termes ou courtes phrases techniquement significatifs qui décrivent le document et qui pourraient en faciliter le catalogage. Choisir des termes qui ne nécessitent pas une classification de sécurité. Des identificateurs comme le modèle, la désignation, la marque de commerce, le nom de code d'un projet militaire et l'endroit peuvent aussi être donnés. Si cela est possible, on tirera les termes choisis d'un thésaurus publié comme le Thesaurus of Engineering and Scientific Terms (TEST) et on indiquera le thésaurus utilisé. S'il n'est pas possible d'utiliser des termes d'indexation sans classification, la classification de chacun devrait être indiquée comme celle du titre.

simulation de réseau d'antennes, interface .NET, C-Sharp

Defence R&D Canada

Canada's leader in Defence
and National Security
Science and Technology

R & D pour la défense Canada

Chef de file au Canada en matière
de science et de technologie pour
la défense et la sécurité nationale

www.drdc-rddc.gc.ca