Descriptif informatique de CALC_ESSAI

1 Introduction

La macro-commande CALC_ESSAI regroupe un ensemble de fonctionalités en corrélation calcul-essai, contrôlables par le biais d'une interface graphique en python Tk :

- expansion de données expérimentales sur une base de déformées numériques (via la macro-comande MACRO_EXPANS),
- modification structurale une structure connue par une base de modes expérimentaux et d'un modèle numérique simplifié,
- identification d'efforts sur une structure à partir de la donnée de l'inter-spectre et d'une base de modes propres décrivant son comportement,
- calcul de spectres filtrés et moyennés (via la macro-commande CALC_SPEC),
- visualisation de déformées modales, de courbes et de matrices de MAC dans Salomé ou dans des outils associés à Aster (GMSH, Xmgrace, Tk).

La macro-commande est un lointain descendant du logiciel MEIDEE, qui était utilisé dans les années 90 pour l'identification d'efforts turbulents sur les structures tubulaires (type tubes de générateur de vapeur ou crayon d'assemblages).

2 Lancement, mode intéractif et non interactif

La macro-commande CALC_ESSAI est lancée avec la routine python calc_essai_ops.py située dans le dossier /bibpyt/Macro. Cette routine effectue les opérations décrites ci-dessous

2.1 Récupération des concepts pré-déclarés

Pour les onglets d'identification des efforts et de modification structurale, il est nécessaire de déclarer à l'appel de la macro-commande le nom des concepts en sortie de celle-ci, par l'utilisation du mot-clé RESU_XXX = CO('NAME'). Les noms des concepts pré-déclarés sont transportés dans toutes les classes d'exécution de calcul de la macro-commande sous la forme d'une liste (RESU_MODIFSTRU) ou d'un dictionnaire (RESU_IDENTIFICATION) contenant les noms, et, dans le cas de l'onglet d'identification, le type de résultat, un compteur (incrémenté de 1 à chaque fois que l'utilisateur déclare un concept sortant) et la routine DeclareOut.
Pour l'onglet d'expansion de données, les résultats peuvent être déclarés intéractivement. Cependant, ils ne peuvent être utilisés dans la suite du calcul que dans le cadre d'une poursuite.
Pour l'onglet de traitement du signal, le nom des concepts sortant est fixé en dur (voir la section 5.5).

2.2 Lancement en mode non interactif

Il n'est pas très pertinent de lancer la macro-commande en mode non-interactif, sauf, éventuellement, dans le cas de l'identification d'efforts, mode qui ne peut être remplacé par l'utilisation d'une autre commande. Ainsi, pour l'expansion de modèles, il vaut mieux utiliser directement la macro-commande MACRO_EXPANS. Pour la modification structurale, on peut se référer à l'enchaînement de commandes décrit dans la documentation U2.07.03. Pour le traitement du signal, on peut utiliser la macro-commande CALC_SPEC (voir U4.32.01).
Le mode non-interactif est donc utilisé pour l'identification d'efforts, et, le plus souvent, pour la validation des cas-tests associés à CALC_ESSAI.
Dans ce mode, on renvoie directement à la routine CalcEssaiTest.

2.3 Lancement en mode interactif

Le lancement se fait dans calc_essai_ops.py dans la routine create_interactive_window. Dans cette routine :

- on importe la classe CalcEssaiObjets (gestionnaire de concepts aster),
- on crée un objet TabbedWindow (routine create_tab_mess_widget) : cet objet est une fenêtre à onglets (voir section 4.1) avec la fenêtre de message en pied,
- on importe toutes les classes graphiques, appelées InterfaceXxxxxx,
- on crée des instances de chaque classe graphique, qu'on affiche dans la fenêtre tabs (instance de TabbedWindow) définie plus haut,
- on crée un fichier de messages qui s'affichera dans la fenêtre mess définie en pied de l'instance tabs.

3 Fichier cata_ce.py : catalogues

Le fichier meidee_cata.py a été défini à l'époque où les méthodes python pour manipuler les concepts Aster n'étaient pas aussi développées. Elles sont néanmoins encore assez utiles, car particulièrement adaptées aux besoins des calculs de CALC_ESSAI.

Ce fichier contient :

- la classe CalcEssaiObjects : classe contenant tous les concepts Aster qui pourront être utiles dans les calculs,
- la classe Resultats : les concepts Aster sd_resultat en sont des instances,
- la classe ModeMeca : sous-classe de la précédente : instancie les concepts Aster mode_meca
- la classe DynaHarmo : sous-classe de Resultats : instancie les concepts Aster dyna_harmo

- les classes CaraElem, ChampMateriau : instancient les concepts Aster cara_elem et cham_mater
- la classe InterSpectre : instancie les concepts Aster table_sdaster de type table_fonction contenant des fonctions fréquentielles,
- la classe Tempo : instancient les concepts Aster table_sdaster de type table fonction contenant des fonctions temporelles (pour le traitement du signal),
- la classe Modele : instancie les concepts Aster modele_sdaster.

Seules les classes les plus utilisées dans les calculs sont décrites ici.

3.1 Description de la classe CalcEssaiObjects

Cette classe est appelée à l'appel de la macro-commande, et après chaque calcul créant un nouveau concept Aster afin de remettre à jour le dictionnaire des concepts disponibles, avec l'utilisation de la méthode recup_objects.
Description des méthodes principales :

3.1.1 recup_objects

Cette méthode récupère tous les concepts disponibles et les range dans un dictionnaire indexé par les noms de ceux-ci, pour chaque type de concepts :

```
self.mode_meca = {'NOM_1':mode_meca_1,'NOM_2':mode_meca_2}
```

Si une classe particulière a été créée pour le concept en question, alors l'objet `mode_meca_X` sera une instance de ce concept. Les classes ModeMeca et InterSpectre sont les plus utilisées dans les opérations courantes. La classe InterSpectre possède notamment une méthode très utile pour transformer le concept inter-spectre Aster en une matrice python indexée par les fréquences de discrétisation et les numéros d'ordre i et j (ou, à la place, les noms de nœuds et des composantes i et j).
Plusieurs méthodes relatives aux classes en question sont appelées pour lier concepts aux nume_ddl, aux modèles, aux matrices, aux maillages ascendants...

3.1.2 update

Permet d'ajouter un concept à l'instance de MeideeObjects sans avoir à la recréer entièrement (ce qui peut être long étant donné le nombre d'initialisations).

3.1.3 Debug

Routine d'affichage des concepts associés à un concept.

3.1.4 Routines get_xxx et get_xxx_name

Les routines get_xxx_name servent à récupérer la liste des noms des concepts xxx existants. La routine get_xxx permet de récupérer le concept à partir de son nom.

3.2 Description de la classe ModeMeca

La notion de mode_meca regroupe indifféremment les modes dynamiques, issus de **CALC_MODES**, et les modes statiques. Les premiers sont indexés par leur numéro d'ordre et leur fréquence propre, les seconds par leur numéro d'ordre et la variable **NOM_CMP**, qui est le degré de liberté associé à la déformée statique.

3.2.1 Classes get_xxx

Ces classes servent à lier le concept mode_meca avec les concepts ascendants :

- get_matrices récupère les matrices de masse, de raideur et d'amortissements associées au concept. Les attributs associés sont .mass, .kass et .cass.
- get_nume récupère le nume_ddl par le même biais (extension .nume),
- get_maillages récupère le maillage associé, soit par le biais du nume_ddl, soit directement dans les informations contenues dans le .REFE de chaque numéro d'ordre du résultat.
- get_modele récupère le modèle associé, soit par la routine dismoi d'Aster, soit en passant par le maillage associé

Attention : certaines routines sont inter-dépendantes ; lors des actualisations, il convient (comme dans la routine update de CalcEssaiObjects) de les exécuter dans un certain ordre.

- get_mode_data : récupère l'ensemble des données généralisées sous la forme d'un dictionnaire ; pour les modes statiques, seuls les champs NUME_MODE, NUME_ORDRE, et NOEUD_CMP sont remplis, les autres valent None ; pour les modes dynamiques, les champs **FREQ**, **AMOR_REDUIT**, AMOR_GENE, RIGI_GENE et MASS_GENE sont remplis, le champ NOEUD_CMP vaut None. La routine show_cara_mod permet d'afficher l'ensemble des caractéristiques modales extraites.

3.2.2 Routine extr_matr

Routine permettant d'extraire les déformées modales sous la forme d'une matrice numpy. L'extraction se fait avec l'opérateur CREA_CHAMP et la routine EXTR_COMP. L'encapsulation de ces deux commandes est faites dans la routine crea_champ à la fin du fichier ce_cata.py.
La routine nume_ddl_phy permet de fabriquer une numérotation des degrés de liberté actifs. Ces DDL sont issus du filtre que l'utilisateur a pu appliquer en utilisant l'opérateur OBSERVATION pour créer la base des modes, ou en utilisant le filtre de DDL dans l'interface graphique.

3.3 Classe DynaHarmo

Les routines permettant d'accéder aux concepts ascendant des dyna_harmo sont relativement similaires à celles de la classe ModeMeca.
Les DynaHarmo ne sont utilisés que pour l'expansion de modèles, classe de calcul qui ne réalise que du calcul Aster. Il n'est donc pas nécessaire d'en extraire les données au format python.

3.4 Classes CaraElem, ChampMateriau

Aucun traitement python de ces concept n'a été réalisé. Ces classes sont donc presque vides. Elles permettent juste d'encapsuler le concept Aster correspondant et de pouvoir pointer sur celui-ci avec son nom.

3.5 Classe InterSpectre

Classe permettant de traiter les concepts de type inter-spectre. Les inter-spectres sont des tables contenant des fonctions de variable fréquentielle indicées par :

- des numéros d'ordre (indices NUME_ORDRE_I et NUME_ORDRE_J),
- ou des noeuds et composantes (NOEUD_I, NOEUD_J, NOM_CMP_I, NOM_CMP_J).

A l'initialisation de l'instance de la classe, on extrait les fréquences de discrétisation des fonctions, ce qui permet de différencier les tables de type-inter-spectre d'autres type de tables (tables contenant des fonctions temporelles, par exemple, qui seront traitées par la classe Tempo).

3.5.1 Routine make_inte_spec

A l'image de l'opérateur LIRE_INTE_SPEC, cette routine fabrique un inter-spectre Aster à partir de fonctions définies sous forme de liste.
Si le concept est relié à un modèle (avec set_model), au travers d'un concept résultat (DynaHarmo ou ModeMeca), alors il est associé à une numérotation des degrés de liberté physiques. L'inter-spectre créé est donc indicé par ses nœuds et composantes. Dans le cas contraire, l'inter-spectre est associé à un concept généralisé. Il est donc indicé par ses numéros d'ordre.

3.5.2 Routine set_model

Permet d'associer un concept mode_meca à l'inter-spectre défini. La cohérence en taille de l'inter-spectre et du concept résultat associé doit être respectée (mais pas forcément la cohérence en DDL, car les inter-spectres ne sont pas tout le temps définis par une numérotation physique.

3.5.3 Routines extr_inte_spec et extr_freq

Opération inverse de la routine make_inte_spec. L'inter-spectre créé est une matrice numpy à 3 dimensions. Etapes de la routine :

- extraction des couples de numéros d'ordre ou des noeuds et composantes i et j, selon que l'inter-spectre soit physique ou modal, avec `extr_nume_ordre`,
  - `coupl_ddl = [('N1_DX','N1_DY')... ('N10_DZ','N10_DZ')]`,
  - ou `coupl_ddl = [(1,2)...(10,10)]`,
- extraction du nombre de fréquences de discrétisation (`nb_freq`),
- la fonction `set` permet de récupérer les numéros d'ordre dans une liste ne comportant qu'une seule occurrence de chacun ; elle peut être utile pour dimensionner les inter-spectres non carrés, par exemple, les fonctions FRF et de cohérences créées dans `CALC_SPEC` ; cette fonctionnalité n'est cependant pas utilisée actuellement par l'onglet de traitement du signal ; dans celui-ci, on extrait les fonctions de l'inter-spectre par des `RECU_FONCTION`, et on ne réalise pas d'opérations matricielles sur les matrices ; on ajoute donc provisoirement une vérification sur l'égalité entre nombre de lignes et de colonnes,
- association à un concept résultat et extraction des DDL actifs (`self.nume_phy`),
- vérification de la cohérence entre la taille du nume_phy et la taille de l'inter-spectre,
- fabrication d'une liste de couple de string pour indicer en python les fonctions de l'inter-spectre : des `NOEUDS_CMP_X` ou des `NUME_ORDRE_X`, selon la manière dont a été fabriqué l'inter-spectre dans Aster (`coupl_ddl`),
- extraction des fonctions au format python ; selon la manière dont a été défini l'inter-spectre, il se peut que les fonctions ne soient pas accessibles directement dans la base à partir de la donnée de son nom ; la solution alternative est d'utiliser `RECU_FONCTION`,
- pour les inter-spectre indicés par les noms des noeuds et les composantes associées, on vérifie la cohérence avec le `self.nume_phy` ; si l'inter-spectre n'est défini que par ses numéros d'ordre, l'utilisateur doit s'assurer lui-même de cette cohérence.

La routine extr_freq fonctionne de manière identique pour ne récupérer que les fréquences de discrétisation.

3.6 Classe Tempo

L'objectif de cette classe est d'encapsuler les concepts contenant des tables de fonctions de variable temporelle. Les méthodes de cette classe sont assez semblables à celles de la classe InterSpectre. Les fonctions temporelles sont destinées à être utilisées par l'opérateur `CALC_SPEC`. Il n'est donc pas nécessaire d'extraire les fonctions au format python.

3.6.1 Méthode extr_tempo

Les échantillons temporels sont indexés par le numéros d'ordre, et le numéro de la mesure, de manière à pouvoir rassembler les échantillons ayant été mesurés simultanément. L'extraction se fait donc par la méthode values, en recherchant les mots-clés `NUME_ORDRE_I`, `NUME_MES` et en rendant les noms `FONCTION` associées.

3.7 Classe Modele

Cette classe permet d'encapsuler les modèles Aster. Les méthodes associées permettent d'extraire le maillage ascendant et l'ensemble des nume_ddl descendants qui dépendent de ce modèle (il peut y en avoir plusieurs. Cette classe a peu d'intérêt en elle-même.

4 Description des outils graphiques génériques

Les routines graphiques sont organisées de la manière suivante :
- routines chapeau et routines génériques : dans le fichier `outils_ihm.py`,
- interfaces graphiques spécifiques à chaque onglet de calcul : dans un fichier python spécifique : par exemple, le fichier `ce_ihm_expansion.py` contient la classe graphique liée à l'onglet d'expansion de données, alors que le fichier ce_calcul_expansion.py contient les calculs associés (et peut être testé en mode non interactif sans faire appel aux modules Tk de python).

On détaille ici le fonctionnement des classes principales TabbedWindow et MessageBoxInteractif. Pour les autres classes, se reporter à leur description en-tête de celles-ci. Ces petites classes permettent d'afficher des menus spécifiques : sélection d'un groupe de noeuds dans un maillage, choix des paramètres pour une opération Aster, paramètres pour un changement de repère.

4.1 Classe TabbedWindow

A l'appel de la macro-commande `CALC_ESSAI` en mode interactif, la routine `create_interactive_window` lance :
- une fenêtre vierge `TabbedWindow` (dans outils.py) dans laquelle sont affichés le contenu spécifique de chaque onglet de travail.,
- une fenêtre de message `MessageBoxInteractif`, dans laquelle seront affichés les message interactif affichés en cours de calcul.

Remarques :
1. L'intérieur de la fenêtre vierge `TabbedWindow` est un Canvas (l'avantage du Canvas sur une Frame classique est la possibilité d'ajouter une barre de défilement verticale si celui-ci est grand),
2. la routine `set_current_tab` sert à afficher l'interface correspondante à chaque changement d'onglet ; la méthode `setup` associée à la classe graphique de l'onglet est appelée pour rafraîchir l'environnement (concepts Aster crées) ; **la méthode `setup` doit donc exister dans les classes graphiques de tous les onglets,**
3. la taille initiale des fenêtres principale et de message sont déterminées respectivement dans les routines `__init__` de définition desdites fenêtres.

4.2 Classe MessageBoxInteractif

Fenêtre vide dans laquelle sont imprimés les messages d'information pendant les calculs. Comme précisé dans la section précédente, la fenêtre est créée à l'appel de la macro CALC_ESSAI (instance `mess`), et est transportée comme argument de toutes les classes graphiques. Exemple :
```
iface = InterfaceCorrelation(main, objects, macro, mess,param_visu)
```
Les arguments sont les suivants :
- `main` : la fenêtre principale de l'IHM, qui contient la TabbedWindow,
- `objects` : instance de CalcEssaiObjects qui contient les concepts Aster existants,
- `macro` : macro-commande, désuet
- `mess` : instance de la fenêtre de messages
- `param_visu` : instance de la classe IHM InterfaceParametres, qui permet de choisir les paramètres de visualisation des déformées et des courbes (Xmgrace + GMSH ou Salomé).

4.2.1 Affichage des messages d'erreur issus d'exceptions

Lorsqu'on lance une commande Aster dans la macro au travers d'une commande interactive, et que le calcul termine en erreur (récupérée sous forme d'une exception par Aster), le message d'erreur est affiche dans la fenêtre de message interactive. On récupère cette erreur grâce à la commande suivante :
```
try:
```

```
        MACRO_EXPANS( MODELE_CALCUL = mcfact_calcul,
                      MODELE_MESURE = mcfact_mesure,
                      NUME_DDL = nume,
                      RESU_ET = res_ET,
                      RESU_RD = res_RD,
                      RESOLUTION = parametres,
                      **args )
    except aster.error,err :
        message = "ERREUR ASTER : " + mess.GetText('I',err.id_message,
                                          err.valk, err.vali, err.valr)
        self.mess.disp_mess( message)
        UTMESS('A','CALCESSAI0_7')
        return
```

Remarque : lors des dernières restitutions réalisées, on a remarqué que ce tracking d'erreurs n'était plus fonctionnel. L'affichage en fenêtre interactive a été supprimé, et remplacé par un simple « Erreur dans le calcul Aster ».

4.3 Classe ScrollList

Frame générique permettant d'afficher un tableau avec barre de défilement. L'attribut values est une liste remplie par `set_values` correspondant à ce qui esst affiché dans le tableau. get_selection rend une liste contenant les indices sélectionnés à la souris et la valeur de values correspondante :

```
[[1,'N1_DX'],[3,'N4_DY']]
```

4.3.1 Classes ModeList et ModeFreqList

Sous-classe de ScrollList, elle possède deux boutons en-dessous permettant de sélectionner ou de dé-sélectionner d'un coup toutes les lignes de la liste. Elle est utilisée pour afficher des numéros d'ordre et une valeur associée (type fréquence propre ou `NOEU_CMP`). `get_selection` retourne uniquement la liste des numéros d'ordre sélectionnés.
La classe ModeFreqList, sous classe de ModeList est à utiliser préférentiellement car elle permet de mettre en forme correctement les valeurs des fréquences propres ou des `NOEU_CMP`.

4.3.2 Classe StudyList

Sous-classe de ScrollList, elle permet d'afficher la liste des études Salomé ouvertes. Elle possède une méthode `valider` qui stocke l'étude Salomé courante pour l'affichage des déformées, MAC et courbes.

4.4 Classe VisuSpectre

Cette classe crée une frame composée de un ou plusieurs tableaux pour l'affichage des coordonnées de l'inter-spectre à visualiser. Cette classe est utilisée pour l'onglet d'identification d'efforts et pour le traitement du signal.
Si `choix != None`, on ajoute un bouton au-dessus permettant de choisir entre plusieurs options à visualiser.

Remarque : il conviendrait d'ajouter dans cette classe des méthodes communes pour l'extraction des fonctions de l'inter-spectre, et leur affichage vers le logiciel de visualisation choisi. Actuellement, ces fonctionnalités sont écrites en double dans les classes d'IHM des onglets correspondantes.

4.5 Classes ParamModelCouple, ParamModeLMME et ParamProjMesuModal

Ces classes permettent d'afficher dans la fenêtre principale ou dans des TopLevel des frame permettant de contrôler les paramètres de calcul pour l'utilisation de CALC_MODES ou PROJ_MESU_MODAL.

4.6 Classe Observation Window

Cette classe génère une Frame pour appliquer à un concept résultat de type mode_meca la macro-commande OBSERVATION. Elle est utilisée dans l'onglet d'identification d'efforts et dans la fenêtre de visualisation des résultats.
Cette Frame est composée de deux cadres de couleurs verte et bleue, dans lesquelles on affiche les groupes de nœuds contenus dans le modèle expérimental choisi, et les DDL disponibles dans chaque groupe. Un bouton permettant de choisir le changement de repère associé est utilisé.

4.6.1 Classe DispObs

Cette classe ouvre une fenêtre externe TopLevel lorsqu'on clique sur le bouton « Observations » dans l'onglet « Paramètres et visualisation ».
On note que dans la méthode set_resu, on vérifie si le concept résultat choisi en amont a été créé par la macro-commande OBSERVATION. Si c'est le cas, alors on pré-sélectionne les DDL sélectionnés et les paramètres de changement de repère (voir la méthode set_selected dans la classe GroupNoWidget ci-dessous).

4.6.2 Classes SelectionNoeuds, SelectionMailles et _SelectionBase

Les classes SelectionNoeuds et SelectionMailles affichent les lignes de groupes de nœuds et de mailles , et sont des sous-classes de _SelectionBase. Cette classe permet de faire le lien entre les DDL sélectionnés par l'utilisateur et la construction du mot-clé dans la macro OBSERVATION. Elles sont basées sur la classe ci-dessous.

4.6.3 Classe GroupNoWidget

Cette classe affiche les listes de groupes de nœuds et de mailles, avec les DDL associés (méthode set_data).
get_selected est une méthode qui retourne sous forme de dictionnaire les options choisies par l'utilisateur.
set_selected est utilisé lorsque le concept mode_meca à visualiser a déjà été créé par OBSERVATION. Dans ce cas, on va chercher dans le contexte les mots-clés qui ont été utilisés lors de la création du concept, et on modifie les cases de l'interface en conséquence. Cela doit permettre de modifier le concept existant sans avoir à reproduire toutes les opérations.

Remarque : la méthode set_selected n'est pas optimale, et il conviendrait de la modifier :
- on a ajouté en fin de routine la commande « return Rien », ce qui génère une erreur ; on a cependant remarqué que si on ne retourne vraiment rien (commande « return » seule), alors lorsqu'on ouvre la fenêtre, les cases à cocher ne le sont pas,
- si on veut modifier un concept déjà créé par OBSERVATION, il faudrait repartir du résultat initial, et non pas du concept déjà projeté, car en partant de ce dernier, on ne peut que retirer encore des points, mais on ne peut pas en modifier.

5 Validation des outils graphiques et arbre d'appel

Dans cette section, on propose, pour chaque onglet :
- une procédure de recettage des outils graphiques,
- la description de l'arbre d'appel pour lesdites procédures

Le test des outils graphiques devra être effectué sur toutes les distributions sur lesquelles on assure le suivi qualité de Code_Aster (en particulier Calibre). On vérifiera que les branchements vers les outils de visualisation sont correctement effectués :
- affichage sur XMGrace et GMSH,
- affichage sur Salomé : ouvrir une cession de Salomé, et vérifier dans l'onglet « Paramètres de visualisation » que l'option « Salome » est bien sélectionnée.

5.1 Fenêtre de visualisation des déformées et des MAC

Lorsqu'on ouvre **CALC_ESSAI** , on arrive en premier sur une fenêtre permettant de paramétrer la visualisation des résultats de calcul. Deux options sont possibles :
- Gmsh/Xmgrace : les déformées sont visualisées dans GMSH, les courbes dans Xmgrace. Les MAC sont visualisés dans une fenêtre Tk spécifique,
- Salomé : tous les résultats de calcul sont visualisés dans Salomé ; pour les MAC, on a créé un script salomé permettant de voir le MAC sous la forme d'un champ constant par éléments aux points de Gauss.

5.1.1 Test de la visualisation : cas-test sdls112a

La validation des fonctionnalités de cet onglet est validée en section 5.3 , en même temps que les fonctionnalités d'expansion de données.

5.1.2 Arbre de calcul

Classe InterfaceParametres (fichier `ce_ihm_parametres.py`) :

C'est une sous-classe de la classe Frame (Tk) dans laquelle est fabriquée l'IHM, appelée au lancement de `CALC_ESSAI`. Elle est composée de deux cadres principaux :
- un cadre permettant de définir le choix de visualisation (GMSH/Xmgrace ou Salomé) ; à noter que la visualisation distante sur Salomé n'est pas disponible ; lorsqu'une étude Salomé est ouverte, son nom apparaît dans la fenêtre ; penser à cliquer sur valider pour choisir l'étude dans laquelle les résultats seront affichés
- un cadre permettant de visualiser les déformées modales, critère de MAC, ou de simuler des FRF « coup de marteau » en donnant une base modale et un point d'excitation (et de les visualiser).

Lorsqu'on sélectionne un résultat, la routine **check_state** vérifie notamment si les modèles des résultats 1 et 2 sont les mêmes. Si c'est le cas, le bouton de MAC est activé. Le calcul de la matrice de MAC est effectué dans un fichier de calcul : **ce_calcul_expansion.py** (routine **calc_mac_mode**).
La routine **view_frf** permet de visualiser les FRF dans une fenêtre annexe (classe **DispFRFDialogue**, dans **outils_ihm.py**), qu'on décrit ci-dessous.

Lorsqu'on choisit un set de logiciels pour la visualisation, on crée un instance associée au choix réalisé :
- classe **CalcEssaiGmsh**, et CalcEssaiXmgrace pour l'affichage des courbes,
- classe **CalcEssaiSalome** ; et **CalcEssaiSalomeCourbes** pour l'affichage des courbes.

La liste des études Salomé ouvertes est actualisée avec le bouton « Actualiser », qui renvoie vers la méthode **visu_study_list**. La méthode appelée (méthode de la classe **CalcEssaiSalome**), utilise un script commun avec **STANLEY** (**SalomeGetStudies.py**).

3 méthodes **visu_resu**, **visu_mac** et **visu_courbe** renvoient respectivement vers les méthodes des deux classes citées ci-dessus.

Classes CalcEssaiLogiciel (fichier ce_ihm_parametres.py) :
Classe chapeau pour le pilotage des logiciels extérieurs. Permet notamment de définir les fichiers où sont imprimés les résultats, et de faire les **IMPR_RESU** nécessaires.

Classes CalcEssaiGmsh (fichier ce_ihm_parametres.py) :
Classe pour l'affichage des déformées dans Gmsh. A noter, pour l'affichage du MAC : celui-ci se fait dans une fenêtre python Tk, qui est définie dans **outils_ihm.py** : on ouvre une fenêtre annexe TopLevel, et on affiche la frame **MacWindowFrame** dedans.

Classes CalcEssaiSalome (fichier ce_ihm_parametres.py) :
Classe pour l'affichage des déformées dans Salomé. Les routines **studylist** et **Show** renvoient vers des scripts Salomé définis pour **STANLEY** , via l'opérateur **EXEC_LOGICIEL** .
Pour la visualisation du MAC, on fabrique dans **make_mac_salome** (dans **ce_calcul_expansion.py**) un maillage carré, et on affecte à chaque maille un champ constant par élément, dont les valeurs valent respectivement :

- la valeur du MAC,
- le numéro d'ordre de la première liste,
- la fréquence de la première liste,
- le numéro d'ordre de la seconde liste,
- la fréquence de la seconde liste.

La routine Show affiche la déformée, en utilisant un script salomé appelé **salomeScriptMac** .

Remarque importante : un bug a été signalé dans Salomé (fiche REX 19375), et l'affiche réalisé n'est pas correct. Pour corriger celui-ci, il faut faire un clic droit sur la vue + edit + OK (sans rien changer aux paramètres).

Classes DispFRFDialogue (fichier outils_ihm.py) :
Classe permettant d'afficher une fenêtre TopLevel (nouvelle fenêtre dépendant de la fenêtre Tk principale), pour afficher des FRF issues d'un concept dyna_harmo, ou de calculer un résultat harmonique à partir d'une base de modes et d'une excitation « marteau ».

- l'ensemble des concepts est initialisé sous la forme d'une liste à deux composantes, pour chacune des deux colonnes,
- quand certains paramètres sont modifiés, on remet à jour tout l'IHM ; ainsi, si le concept choisi est un `mode_meca`, on fait apparaître la Frame pour choisir les hypothèses de calcul du résultat harmonique associé ; si le concept est déjà un résultat harmonique, alors on n'a pas besoin d'afficher cette fenêtre
- quelques attributs de la classe :
  - `self.dyna` : contient les dyna_harmo, à calculer, ou déjà calculés ; le troisième est utilisé par l'onglet de modification structurale, pour calculer le déplacement interne donné en entrée (défini sur une super-maille `self.sumail`),
  - `self.ddls` : pour chacune des colonnes, la liste des ddl associés à un type de champ (`self.champ_choisi`) ; NB : on vérifie pour chaque champ que le DDL existe, mais on ne vérifie pas si la composante est non vide pour le noeud choisi ; si la composante n'existe pas, on renvoie un message d'erreur dans la fenêtre interactive (au moment de `RECU_FONCTION`)
  - `self.var_type_resu` : pour chacune des deux colonnes, permet de dire si le résultat sélectionné est un `dyna_harmo` ou un `mode_meca`, et donc d'afficher, ou pas, la fenêtre de calcul d'un résultat harmonique,
  - `self.param_calc` : permet de rassembler dans un dictionnaire les variables entrées dans l'IHM par l'utilisateur pour le calcul d'un résultat harmonique : le noeud et la direction d'excitation, la bande de fréquence et la résolution fréquentielle,
  - `self.param_disp` : pour chacune des deux colonnes, les paramètres pour afficher les FRF : noeud et direction de visualisation, et le champ à visualiser.

- calc_dyna_line_harm : méthode de calcul du résultat harmonique, sur base physique : le calcul est faux pour les résultats expérimentaux (et n'a, par ailleurs, pas d'intérêt) car les matrices physiques expérimentales ne sont pas connues
- choix_champ : regarde dans le .TACH de la structure de données si les champs correspondant aux déplacements, vitesses et accélérations ne sont pas vide ; s'ils ne le sont pas,
- choix_ddl : utilise CREA_CHAMP et EXTR_COMP pour récupérer le champ pour le premier numéro d'ordre au format python ; les composantes de ce champ disponibles sont stockées dans le .comp dela structure de données résultat,

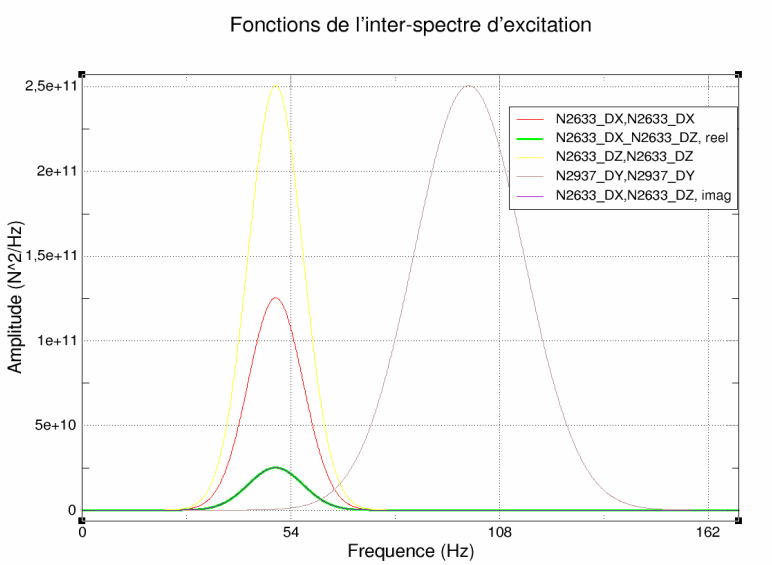
affich_FRF : récupération de la fonction, et récupération du message d'erreur si la composante et/ou le noeud choisis n'existent pas ; l'affichage se fait en passant par la classe param_visu.

5.2 Identification d'efforts

5.2.1 Présentation générale : cas-test sdlv125a

L'objectif de ce cas-test est de simuler une mesure sur un cylindre excité en trois points. L'effort est défini par un inter-spectre de taille 3, dont les fonctions sont des gaussiennes (bruit rose). Les efforts sur les DDL N2633_DX et N2633_DZ sont corrélés, avec un déphasage de $\pi/2$. Les fonctions sont représentées sur le graphe ci-dessous :

Figure 5.2.1-a : inter-spectre d'excitation.



Le calcul ayant permis la simulation des données est réalisé avec DYNA_ALEA_MODAL, sur base modale.
Une base de mode restreinte à l'ensemble des capteurs (les « DDL capteurs »)est calculée avec l'opérateur OBSERVATION, sur laquelle on restitue le résultat physique avec REST_SPEC_PHYS, pour obtenir une mesure simulée. L'objectif du cas-test est d'identifier les efforts appliqués à partir de la donnée de l'inter-spectre SPECTPH1 et de la base de modes projetée MODEIDE1. L'identification se fait sur un nombre réduit de DDL choisis a priori et appelés « DDL commande ».

5.2.2 Validation du cas-test en interactif

Pour lancer le cas-test en interactif, modifier le fichier de commande sdlv125a.comm en écrivant interactif=1 avant la commande CALC_ESSAI. L'IHM de CALC_ESSAI se lance. Se placer dans l'onglet « identification de chargement ».

Dans la commande DEBUT, supprimer le mot-clé CODE. De cette manière, si le code plante sur une erreur Aster, celle-ci est rattrapée en Exception, et le message est affiché dans l'IHM de CALC_ESSAI.

1) Choix de la base de modes :
   - choisir MODE_NUM : on n'utilise dans cette base que les caractéristiques modales : masses, fréquences et amortissements modaux

2) Base de déformées observable :
   - modèle expérimental : MODELEXP : ne contient que 31 noeuds,
   - base de modes : MODE_NUM : va être projetée avec OBSERVATION sur MODELEXP,
   - groupe de noeuds GPCYLRED : cocher les DDL DX et DZ, choisir un changement de repère 'NORMALE', avec l'option VECT_Y = (0.0,1.0,0.0),
   - groupe de noeuds GPSUPRED : cocher le DDL DY et choisir un changement de repère 'CYLINDRIQUE' avec AXE_Z = (0.0,1.0,0.0),
   - Cliquer sur « Valider ».

3) Base de commandabilité :
   - modèle expérimental : MODELACT : ne contient que 3 noeuds,
   - base de modes : MODE_NUM : va être projetée avec OBSERVATION sur MODELACT,
   - groupe de noeuds N1 : cocher les DDL DX et DZ,
   - groupe de noeuds N2 : cocher le DDL DY,
   - Cliquer sur « Valider ».

4) Inter-spectre en fonctionnement :
   - choisir SPECTPH1, et laisser Type de champ à 'DEPL'.

5) Paramètres du calcul :
   - Ne pas les modifier (pas de régularisation effectuée),
   - Cliquer sur « Lancer »

   Vérifier, lorsqu'on « oublie » l'une des opérations précédentes, qu'un message apparaît dans la fenêtre de messages. Il faut éviter l'affichage du tracking python dans la fenêtre console, qui n'est pas explicite pour l'utilisateur.
   Le calcul prend moins d'une minute. La plupart du temps de calcul est utilisé pour l'extraction des inter-spectre sous forme de matrice python. Le calcul en lui-même est presque instantané. Lorsque le calcul est terminé, on voit l'ensemble des opérations effectuées, la dernière étant « Calcul des Syy_S : Synthèse modale des déplacements ».

6) Affichage de courbes : allure de l'effort identifié
   - Dans la colonne de gauche de visualisation des résultats, choisir l'option Eff Phy (pour «efforts physiques »), et cliquer sur « Valider » ; sélectionner une ou plusieurs fonctions de l'inter-spectre et visualiser les courbes en modifiant les échelles d'abscisses et d'ordonnées (LIN/LOG),
   - Exporter l'inter-spectre : on peut lui donner un titre dans la case située en-dessous de la liste des fonctions. Le nom de la table est celui pré-déclaré à l'appel de la macro-commande ('EFFORTS').

7) Affichage des courbes : comparaisons déplacements mesurés/synthétisés :
   - Dans la colonne de gauche, choisir Depl Phy (pour déplacements physiques mesurés), et dans celle de droite, Depl synt (pour déplacements synthétisés), et cliquer sur « Valider »
   - Sélectionner dans chacune des colonnes une ou plusieurs fonctions à comparer, et cliquer sur « Afficher courbe »
   - Exporter les deux inter-spectres. Les noms sont ceux pré-déclarés à l'appel de la macro-commande : 'DEPL_PHY', et 'DEPL_SYN'. La création de ces inter-sectres au format Aster est relativement longue.

5.2.3 Arbre de calcul

On mentionne ici les principales routines utilisées pour la réalisation du calcul précédent.

Classe InterfaceIdentification (fichier ce_ihm_identification.py) :

C'est une sous-classe de la classe Frame (Tk) dans laquelle est fabriquée l'IHM d'identification, appelée au lancement de CALC_ESSAI. Remarques sur le comportement de cette classe :

- tous les inter-spectres calculés sont des attributs de cette classe (inititialisation à une taille nulle, la taille est donnée par la classe de calcul) :
  - self.Syy : mesure
  - self.Syy_R : mesure recalculée à partir des déplacements modaux,
  - self. Sqq : déplacements modaux,
  - self.SQQ : efforts modaux,
  - self.SQQ_R : efforts modaux recalculés à partir des efforts physiques,
  - self.Sff : efforts physiques de dimension nb_act (nombre d'actionneurs)
  - self.Syy_S : mesure re-synthétisée à partir des efforts physiques
- Méthode setup : toutes les classes graphiques contiennent une méthode setup, appelée pour rafraîchissement lorsqu'on change d'onglet, ou lorsqu'une série de calculs est terminée ; celle-ci met à jour les menus déroulant avec les nouveaux concepts aster créés (par exemple, une base de modes créée par expansion dans l'onglet correspondant).
- Méthode _create_opt_data : définit certaines caratéristiques des inter-spectres calculés (comme les variables d'accès, nume_ddl ou nume_mode qui leur sont associés), et les fonctions utilisées pour en extraire les fonctions,
- Méthodes _definit_observabilite et _definit_commandabilite : définition des interfaces permettant de paramétriser l'opérateur OBSERVATION qui crée les concepts self.obs_co et self.com_co.
  - ces concepts sont calculés en cliquant sur « Valider » (calculate_xxxx)
  - les fenêtres de couleur sont des widgets de la classe SelectionNoeud (voir ci-dessous) : cette classe permet, entre autres, de récupérer les DDL associés à chaque nœud du modèle, et les groupes de nœud du maillage
- Méthodes _calculate_observabilite et _calculate_commandabilite : lancement de OBSERVATION : les mots-clés facteurs pour les changements de repère et pour le filtrage des DDL sont fabriques dans get_filtres à partir du résultat de la routine get_selected, décrite ci-dessous.
- Méthode calculs : lancements successifs des calculs :
  - calcturb est une instance de la classe CalcEssaiIdentification ,
  - calculs fait les liens entre les concepts de la classe graphique et les concepts équivalents de la classe de calcul,
  - après le calcul, récupère les nouveaux concepts calculés et les associe à des attributs de la classe graphique (exemple : self.Syy = self.calcturb.Syy) ; l'existence des nouveaux concepts est donnée par une variable booléenne (exemple : self.calcturb.is_Syy = 1 , variable mise à 0 au début du calcul)
- Méthodes pour l'affichage des courbes :
  - On choisit le type de courbe à afficher ; exemple, on choisit 'Eff Phy' pour afficher les efforts physiques
  - on clique sur « Valider » : la routine get_list renvoie la liste des fonctions de l'inter-spectre correspondant ; exemple : [['N1_DX','N1_DX'] , ['N1_DX','N2_DX'],['N2_DX','N2_DX']] , listes fabriquées par les fonctions calcul_xxx_xxx , en se basant sur la numérotation associée à l'inter-spectre (physique ou généralisée),
  - plot_curve : affichage des courbes ; utilise la fonction _get_graph_data , qui récupère les abscisses et ordonnées de la fonction dans la matrice inter-spectrale, et la méthode visu_courbe (classe InterfaceParametres) pour l'envoi de l'affichage par Xmgrace ou Salome.

Classes SelectionNoeud et SelectionMaille (fichier cata_ce.py) :

Permet de créer un widget affichant les groupes de noeuds associés à un maillage, et les DDL portés par les noeuds de ce groupe (widgets verts et violet dans l'interface). Utilisation :

- choix d'un modèle expérimental : appel à _observability_changed ,
- récupération du modèle associé au nom (get_modele),
- méthode set_resultat dans SelectionNoeud : appel à set_modele_maillage ,

- méthode `set_modele_maillage` : récupération du PRNM du modèle qui porte les DDL portés par chaque noeud, et des groupes de noeuds,
- méthode `set_modele_maillage` : pour chaque groupe, récupération des « vraies composantes » associées au PRNM (voir wiki Aster pour la description de cette structure de données) avec la fonction `find_composantes` .
- Méthode `get_selected` : retourne les données de changment de repère et de filtrage des DDL pour l'utilisation de `OBSERVATION` .

Classes CalcEssaiIdentification et CalculInverse (fichier `ce_calcul_identification.py`) :
Classes de calcul pour réaliser le calcul inverse d'identification d'efforts.

Remarques :
- dans les autres onglets, la plupart des calculs sont réalisés par des commandes Aster. Ici, une fois que les données d'entrée sont extraites, on utilise uniquement des fonctions python (opérations sur les matrices, décomposition en valeurs singulières...),
- le nom de variable xxxm1 désigne l'inverse ou le pseudo-inverse (par SVD) de xxx.

Description des méthodes principales :
- Méthode `calculate_force` : méthode principale de la classe gérant le calcul des efforts ;
  - les erreurs dans la classe `CalculInverse` sont récupérées et un message d'erreur générique est affiché ; dans chaque méthode de calcul élémentaire, un message plus précis (précisant les dimensions des matrices en jeu) est donnée,
  - chaque résultat est une instance de la classe `InterSpectre` (voir `cata_ce.py`) ; on peut lui associer une numérotation de DDL physiques ou généralisés, selon le type d'inter-spectre calculé ; pour cela, on associe l'inter-spectre à une base existante ; nume_phy renvoie une liste de caractères de la forme `['N1_DX','N1_DY','N2_DX'...]` , et nume_ddl_gene renvoie une liste de la forme `['MO1','MO2','MO4'...]` (numérotation basée sur la variable d'accès `NUME_MODE`).
- Méthode `Calc_Z` : calcul de la matrice d'impédence $Z = diag(-\omega^2 + \omega_j^2 + 2\xi\omega\omega_j)$ et de son inverse $Zm1$; selon le type de données (accélérations, vitesses ou déplacements, on peut être amené à intégrer en divisant par $\omega^{\exp}$, où exp vaut respectivement 2, 1 ou 0.
- méthodes `Calc_SQQ` , `calc_Sff` : calculent respectivement les pseudo-inverses des matrices $C\Phi$ et $\Phi^T B$ (voir U4.90.01, section 6.2.1 pour la signification de ces matrices) ; les valeurs singulières trop faibles (en-dessous d'un critère ϵ choisi par l'utilisateur) sont mises à zéro ; après régularisation de Tikhonov, l'inverse de la valeur singulière s_j vaut $\frac{s_j}{s_j^2+\alpha^2}$, le paramètre α pouvant être différent selon la fréquence où l'on se trouve (ajusté par la puissance m) : ainsi, la régularisation est faible lorsqu'on se trouve avant ou près de la fréquence du mode, et elle devient plus forte lorsqu'on s'en éloigne. Les matrices ainsi calculées sont mises en mémoire dans des instances de la classe `InterSpectre` .
- Méthode `verif_Syy` : reclacule les déplacements physiques `Syy_R` à partir des déplacements modaux pour les comparer avec `Syy` (pour estimer la qualité de l'estimation inverse),
- Méthode `verif_SQQ` : recalcule les efforts modaux `SQQ_R` à partir de `Sff` pour les comparer avec `SQQ` ,
- Méthode `synthes_Syy` : recalcule les déplacements `Syy_S` à partir des efforts identifiés `Sff` pour estimer la qualité générale de l'inversion,
- Méthode `choix_alpha` : le paramètre α de la régularisation de Tikhonov peut être modifié selon la fréquence à laquelle on se trouve par rapport aux fréquences propres de la base.

5.3 Expansion de données

5.3.1 Présentation générale du cas-test sdls112a

L'objectif de ce cas-test, basé sur un benchmark international (benchmark du garteur), a pour objectif d'étendre les FRF et modes expérimentaux identifiés sur la maquette, c'est-à-dire de trouver la meilleure combinaison linéaire d'une base de modes (dynamiques, statiques) collant à l'expérience. Pour plus de précisions, voir U4.90.1 (doc de CALC_ESSAI), section 4.2.1, et V3.03.112 (documentation de référence du cas-test).

Remarque :

> En non-interactif, la commande CALC_ESSAI est ici complètement inutile, il suffit d'utiliser la macro-commande MACRO_EXPANS, ou d'enchaîner les commandes PROJ_MESU_MODAL et REST_GENE_PHYS. Pour la validation en interactif de CALC_ESSAI, on propose de reproduire les expansions réalisées dans ce cas-test.

5.3.2 Validation du cas-test en interactif

Pour lancer le cas-test en interactif, modifier le fichier de commande sdls112a.comm en écrivant interactif=1 avant la commande CALC_ESSAI. Dans la commande DEBUT, supprimer le mot-clé CODE. De cette manière, si le code plante sur une erreur Aster, celle-ci est rattrapée en Exception, et le message est affiché dans l'IHM de CALC_ESSAI.
Augmenter la taille de mémoire utilisée à 1024 Mo. Pour gagner en rapidité de calcul, on peut également effacer ou commenter les opérations suivantes : TEST_RESU, TEST_TABLE, MACRO_EXPANS et les commandes MAC_MODES, que l'on va exécuter en interactif.
L'IHM de CALC_ESSAI se lance. Se placer dans l'onglet « identification de chargement ».

1) Premier test : expansion d'un résultat harmonique sur une base de modes :
   - choix de la base de modes d'expansion : choisir MODES et ne sélectionner que les modes dont la fréquence est comprise entre 1 et 100 Hz,
   - base des FRF à étendre : choisir le concept DYNA ; ne pas sélectionner de numéros d'ordre (il n'est posible avec MACRO_EXPANS , qui utilise EXTR_MODE , d'extraire une partie des numéros d'ordre d'un concept dyna_harmo .
   - choisir le nom du concept résultat : ce nom doit avoir moins de 5 lettres ; les concepts XXX_ET (résultat étendu), XXX_NX (base d'expansion réduite aux numéros d'ordre sélectionnés) et XXX_RD (résultat étendu réduit) seront créés,
   - comparaison des résultats étendus au concept harmonique initial : cliquer sur la case FRF dans l'onglet « Paramètres et visualisation » de l'IHM :
     - choisir les deux concepts résultats à comparer : d'un côté DYNA et de l'autre XXX_RD ,
     - choisir le champ à comparer ('ACCE'),
     - choisir le noeud et la composante : pour DYNA , choisir le noeud 'N1011' , composante 'D3' , et pour XXX_RD , choisir le même noeud avec la composante 'DZ' ,
     - cliquer sur « Afficher » : vérifier que les courbes affichées sont à peu près confondues.

2) Test 2 : simulation d'une FRF à partir d'une base de modes : l'objectif est de comparer une base de FRF expérimentales à la base de FRF simulée numériquement sur une expérience « coup de marteau » :
   - cliquer sur la case FRF dans l'onglet « Paramètres et visualisation »,
   - choisir dans une des colonnes le concept MODES ,
   - choisir un noeud d'excitation (exemple, 'N1' , excitation 'FX'), une bande de fréquence de calcul (1.0 , 50.0) et une résolution fréquentielle (exemple : 1.0 Hz)
   - cliquer sur « Calculer »
   - afficher une FRF (exemple, selon le noeud 'N1, comopsante 'DX') ; vérifier que la FRF s'affiche correctement.

3) Test 3 : expansion d'une base de modes expérimentaux sur une base numérique :
   - relancer le calcul (pour éviter d'encombrer inutilement la mémoire),
   - dans l'onglet expansion de données, choisir la base d'expansion : MODES ,

- choisir la base de modes expérimentaux à étendre : MODMES ; les fréquences sont les mêmes, car le cas-test est trivial : les modes à étendre sont issus de la projection sur le modèle expérimental des modes numériques,
- Choisir une liste de modes dans les deux colonnes,
- Choisir un nom de concept résultat (moins de 5 lettres) et cliquer sur « Calculer »
- Comparer les déformées résultats dans l'onglet « Paramètres et visualisation » : comparer, par exemple, les modes expérimentaux (MODMES) et les modes étendus réduits (XXX_RD), en les choisissant dans le double menu en bas de l'onglet, et en cliquant sur « Déformées » ; utiliser les modes de visualisation dans GMSH et Salomé,
- Afficher le MAC résultat : choisir deux déformées définies sur le même modèle, et cliquer sur le bouton MAC ; choisir par exmple :
    - les modes étendus (XXX_ET) et la base d'expansion,
    - les modes étendus réduits (XXX_RD) et la base expérimentale (MODMES).

5.3.3 Arbre de calcul

Classe InterfaceCorrelation (fichier ce_ihm_expansion.py) :
C'est une sous-classe de la classe Frame (Tk) dans laquelle est fabriquée l'IHM d'expansion, appelée au lancement de CALC_ESSAI. Remarques sur les attributs créés à l'initialisation :

- self.calculs : c'est une instance de la classe CalcEssaiExpansion, qui effectue tous les calculs Aster inhérents à l'expansion de données (lancement de MACRO_EXPANS principalement, et de MAC_MODE pour le post-traitement),
- self.resu_num et self.resu_exp : désignent respectivement la base d'expansion (un concept mode_meca) et le résultat expérimental (un mode_meca ou un dyna_harmo) ; à ces variables sont associées les StringVar self.var_resu_num et self.var_resu_exp, qui désignent les string contenant le concept sélectionné par l'utilisateur,
- self.var_resu1 et self.var_resu2 : StringVar contenant les noms des concepts sélectionnés par l'utilisateur dans le menu de post-traitement (en bas de l'IHM)
- Méthode setup : toutes les classes graphiques contiennent une méthode setup, appelée pour rafraîchissement lorsqu'on change d'onglet, ou lorsqu'une série de calculs est terminée ; celle-ci met à jour les menus déroulant avec les nouveaux concepts aster créés (par exemple, une base de modes créée par expansion dans l'onglet correspondant),
- Méthode prepare_calculs : permet de lancer le calcul d'expansion proprement dit :
    - vérification que les données nécessaires au calcul ont bien été sélectionnées par l'utilsateur,
    - récupération de la liste des indices des lignes sélectionnées dans la liste des modes (on utilise pour cela la méthode selection de la classe ModeList),
    - affectation des attributs de la classe de calcul en appelant la méthode self.calculs.setup,
    - lancement des calculs : self.calculs.calc_proj_resu.

Classe CalcEssaiExpansion (fichier ce_calcul_expansion.py) :
Lancement des calculs d'expansion, principalement la macro-commande MACRO_EXPANS. On détaille ici le fonctionnement de la méthode principale, à savoir calc_proj_resu :

- on crée deux à quatre concepts résultats :
    - XXX_ET est le résultat principal de l'expansion : la base de modes ou le résultat harmonique étendu
    - XXX_RD est la reprojection du résultat étendu sur le modèle expérimental,
    - si on sélectionne tout ou partie de la base de modes numériques, on crée une base extraite appelée XXX_NX (condition « if self.mode_num_list »),
    - si on sélectionne tout ou partie de la base de modes expérimentaux, on crée une base extraite appelée XXX_EX (condition « if self.mode_exp_list »),
- XXX est le nom du concept choisi, il ne doit pas excéder 5 caractères,
- après le calcul, l'ensemble des concepts créés est ajouté à la liste des objects Aster existants (self.ce_objects , ou mdo) avec la commande update ; en parallèle les menus MyMenu sont eux-mêmes actualisés dans la classe Interface Correlation.

5.4 Modification structurale

5.4.1 Présentation générale du cas-test sdll137

Ce cas test a pour but de valider la procédure et le calcul de modification structurale à partir des informations mesurées. On se propose de calculer les variations des deux premières fréquences propres d'une poutre encastrée-libre, suite à la modification apportée sur une portion de la poutre.
Les modélisations A et D du cas test détaillent le déroulement de la procédure de calcul en se servant uniquement des commandes standards de Aster (sans passer par la commande CALC_ESSAI).
Les modélisations B et C font intervenir la commande CALC_ESSAI.

5.4.2 Validation du cas-test en interactif

Pour lancer les cas-tests sdll137b et sdll137c en interactif, basculer la variable interactif à une valeur non nulle (1 par exemple). Cette variable est localisée avant l'appel de la commande CALC_ESSAI. Dans la commande DEBUT, supprimer le mot-clé CODE. De cette manière, si le code plante sur une erreur Aster, celle-ci est rattrapée en Exception, et le message est affiché dans l'IHM de CALC_ESSAI.
L'IHM de CALC_ESSAI se lance. Se placer dans l'onglet « Modification structurale ».

1) Premier test : modification structurale en utilisant la méthode ES pour le choix de la base d'expansion (sdll137b.comm)
   - Dans le panneau « Choix de la base d'expansion », choisir les données d'entrée suivantes :
     Modes expérimentaux : MODEIDE
     Modèle support : MODLSUP
     Matrice raideur (support) : KASSUP
     Méthode (base expansion) : ES
     Modèle modification : MODLCPL
     Pour le choix des noeuds et ddl capteur, sélectionner les ddl DY et DZ du groupe de noeuds CAPTEUR
     Pour le choix des noeuds et ddl interface, sélectionner tous les ddl du groupe de noeuds EXTERNE (l'utilisation de l'ascenseur est peut-être nécessaire afin d'atteindre ce groupe de noeuds)
     Nom de la super-maille : SUMAIL

     On peut ensuite cliquer sur **Valider** pour la validation des données saisies. La liste des modes propres identifiés et la liste des vecteurs disponibles pour la base d'expansion sont alors affichées dans les deux colonnes de gauche. On sélectionne, avec la souris, tous les vecteurs de la liste des « Modes du modèle expérimental » et les huit vecteurs de la « Base d'expansion ».

   - Dans le panneau « Couplage modification / modèle condensé » :
     On garde les paramètres par défaut pour PROJ_MESU_MODAL et le calcul modal.
     On clique ensuite sur **Calculer** afin d'obtenir les fréquences propres de la structure modifiée qui sont affichées dans la colonne Fréquences structure modifiée.
     Vérifier que les deux premières fréquences propres calculées sont proches de : 7.79 Hz et 32.84 Hz
     On peut ensuite vérifier la qualité de la base d'expansion. On choisit MAC pour la rubrique critère. En cliquant sur **Valider**, une matrice présentant les résultats du critère est affichée. La base d'expansion est supposée correcte si les termes diagonaux de cette matrice sont proches de l'unité.

   - Pour visualiser l'effet de la modification sur les déformées, on clique sur **Déformées** dans le panneau « Comparaison structure initiale / structure modifiée ».
     On voit ensuite apparaître une fenêtre GMSH ou SALOME qui permet de comparer les déformées modales initiales et après modification.

2) Deuxième test : simulation d'une modification structurale en utilisant la méthode LMME pour le choix de la base d'expansion (sdll137c.comm) :

- Dans le panneau « Choix de la base d'expansion », on choisit les mêmes données d'entrée que pour le premier test, sauf pour le choix de la méthode pour le calcul de la base d'expansion et le nom du modèle modification.
  Méthode (base expansion) : LMME
  Modèle modification : MODLX

  En cliquant sur **Valider**, la liste des modes propres identifiés et la liste des vecteurs disponibles pour la base d'expansion sont affichées dans les deux colonnes de gauche. On sélectionne, avec la souris, tous les vecteurs de la liste des « Modes du modèle expérimental » et les huit premiers modes de la « Base d'expansion ».

- Dans le panneau « Couplage modification / modèle condensé », on effectue la même procédure que pour le premier test.
  Pour la vérification de la qualité de la base d'expansion, on va utiliser le critère IERI (au lieu du critère de MAC). Une matrice de pondération est nécessaire pour ce critère. On choisit de pondérer avec la matrice de rigidité en cliquant sur **Rigidité**. En cliquant sur **Valider**, une matrice présentant les résultats du critère est affichée. Avec le critère IERI, la base d'expansion est supposée correcte si les termes diagonaux de cette matrice sont proches de zéro.

5.4.3 Arbre de calcul

Les deux fichiers python les plus importants pour le calcul de modification structurale sont : ce_ihm_modifstruct.py et ce_calcul_modifstruct.py.
Le module ce_ihm_modifstruct gère les classes relatives à la partie interface graphique et le module ce_calcul_modifstruct gère les classes qui lancent la procédure de calcul de modification structurale en appelant les opérateurs aster.

Classe InterfaceModifStruct
Cette classe est une classe du module ce_ihm_modifstruct. Elle est la classe chapeau pour l'onglet « Modification structurale ». Elle possède plusieurs attributs dont expansion et couplage.
self.expansion est une instance de la classe InterfaceExpansion. Cette classe permet de saisir et d'afficher les données nécessaires pour le choix de la base d'expansion nécessaire pour l'obtention du champ de déplacement à l'interface.
self.couplage est une instance de la classe InterfaceCouplage. Cette classe permet de saisir et d'afficher les différents paramètres de calcul pour l'estimation des modes propres de la structure modifiée. Cette structure modifiée est issue du couplage entre le modèle de la structure initiale condensée et le modèle de la modification.

Classe CalcEssaiModifStruct
Cette classe est une classe du module ce_calcul_modifstruct. Elle permet d'enchaîner les opérateurs aster nécessaires pour le calcul des modes propres de la structure modifiée. Elle comprend plusieurs méthodes dont les plus importantes sont :

- calc_base_es : calcule la base d'expansion en utilisant la méthode ES (expansion statique). Cette méthode lance l'opérateur MODE_STATIQUE. Les vecteurs de base obtenus sont des déformées statiques.
- calc_base_lmme : calcule la base d'expansion en utilisant la méthode LMME (Local Model Modeshapes Expansion). Cette méthode effectue un calcul modal avec l'opérateur CALC_MODES avec l'option PLUS_PETITE. Les vecteurs de base obtenus sont des modes propres.
- condensation : effectue la condensation du modèle mesuré et crée la super-maille représentant la structure initiale. Cette méthode enchaîne les opérateurs PROJ_MESU_MODAL, MACR_ELEM_STAT et DEFI_MAILLAGE.
- modes_modele_couple : calcule les modes propres du système couple (structure initiale + modification) avec les paramètres de calcul fournis par l'utilisateur. Cette méthode effectue

également la rétro-projection des résultats sur le maillage mesure en utilisant l'opérateur DEPL_INTERNE.

- indicateur_choix_base_expansion : compare le champ de déplacement à l'interface obtenus avec le modèle couplé et ceux obtenus par expansion statique. On relève le champ de déplacement de la structure modifiée au point capteur, puis on effectue une expansion statique afin d'obtenir le champ à l'interface. Cette tâche est réalisée à l'aide les opérateurs PROJ_MESU_MODAL, REST_GENE_PHYS, PROJ_CHAMP et MAC_MODES.

5.5 Calculs d'inter spectres, d'auto spectres et de fonctions de transfert

5.5.1 Présentation générale du cas-test zzzz241a

L'objectif de ce cas-test est de valider les fonctions de base de traitement du signal intégrées dans l'opérateur CALC_SPEC. On simule la réponse temporelle d'un oscillateur à 4 degrés de libertés, et on calcule différentes fonctions de transferts. Pour plus de précisions, voir la documentation U4.32.21 (doc de CALC_SPEC)

Remarque :

En non-interactif, la commande CALC_ESSAI est ici complètement inutile, il suffit d'utiliser la commande CALC_SPEC. Pour la validation en interactif de CALC_ESSAI, on propose de reproduire les expansions réalisées dans ce cas-test.

5.5.2 Validation du cas-test en interactif

Pour lancer le cas-test en interactif, modifier le fichier de commande zzzz241a.comm en remplaçant toutes les lignes situées sous la commande tab_rep=CREA_TABLE(...) (lignes 196 à 307) par
TEST_CE=CALC_ESSAI(INTERACTIF='OUI',),
et ne pas utiliser le fichier de poursuite zzzz241a.com1.
Dans la commande DEBUT, supprimer le mot-clé CODE. De cette manière, si le code plante sur une erreur Aster, celle-ci est rattrapée en Exception, et le message est affiché dans l'IHM de CALC_ESSAI.
L'IHM de CALC_ESSAI se lance. Se placer dans l'onglet « Traitement du signal ».

1) Premier test : Calcul des estimateurs H1 et H2 des fonctions de transfert, avec la mesure 1 en référence : En partant de la fenêtre initiale de CALC_ESSAI, il faut :
   - Dans la liste « Tables disponibles », sélectionner « tab_rep » pour mettre le nom en surbrillance,
   - Cliquer sur le bouton « => » pour remplir les listes intitulées « Points de mesures » et « Points de référence ». On doit obtenir 5 points de mesures, numérotés de 1 à 5, et 5 points de référence, également numérotés de 1 à 5, correspondant aux mêmes mesures.
   - Choisir les mesures associées aux points 2 à 5 dans la liste « Points de mesures », et la mesure associée au point 1 comme référence,
   - Choisir la fenêtre « Hanning »,
   - Renseigner une longueur de « 12 », et cliquer sur « Durée »,
   - Renseigner un recouvrement de « 50 », et cliquer sur «Pourcent»,
   - Sélectionner « H1 » sous le bouton « Transfert », et cliquer sur « Transfert » pour calculer les estimateurs H1.
   - Visualiser les résultats en sélectionnant dans la liste « Transfert (H1) ». Choisir le format des données à afficher, ainsi que la nature des axes, et cliquer sur « Visualiser ».
   - Sélectionner « H2 » sous le bouton « Transfert », et cliquer sur « Transfert » pour calculer les estimateurs H2.
   - Visualiser les résultats en sélectionnant dans la liste « Transfert (H2) ». Choisir le format des données à afficher, ainsi que la nature des axes, et cliquer sur « Visualiser ».

2) Second test : Calcul des inter spectres et auto spectres : En repartant de la fenêtre initiale de CALC_ESSAI , il faut :
   - Dans la liste « Tables disponibles », sélectionner « tab_rep » pour mettre le nom en surbrillance,

- Cliquer sur le bouton « => » pour remplir les listes intitulées « Points de mesures » et « Points de référence ». On doit obtenir 5 points de mesures, numérotés de 1 à 5, et 5 points de référence, également numérotés de 1 à 5, correspondant aux mêmes mesures.
- Choisir les mesures associées aux points 1 à 5 dans la liste « Points de mesures ». La sélection ou non de mesures de référence ne change pas le résultat, cette sélection n'étant, pour le calcul des spectres, pas prise en compte.
- Choisir la fenêtre « Hanning »,
- Renseigner une longueur de « 12 », et cliquer sur «Points»,
- Renseigner un recouvrement de « 50 », et cliquer sur «Pourcent»,
- Cliquer sur «Interspectres» pour calculer les inter spectres.
- Visualiser les résultats en sélectionnant dans la liste «Interspectres ». Choisir le format des données à afficher, ainsi que la nature des axes, et cliquer sur « Visualiser ».

5.5.3 Arbre de calcul

L'onglet « Traitement du signal » fait appel à une classe unique.

Classe InterfaceCalcSpec (fichier ce_calc_spec.py) :
Classe permettant de gérer l'onglet « Traitement du signal ». On y défini à la fois les objets graphiques et les appels à l'opérateur CALC_SPEC, qui réalise les calculs. Les principales méthodes sont :

- InterfaceCalcSpec : méthode pour la génération de l'interface graphique (listes, cadres, boutons, etc.)
- frame_visu : méthode pour le rafraîchissement / initialisation de la partie visualisation de l'onglet
- visu_tempo : mise à jour de la liste de visualisation en ajoutant les fonctions importées des tables.
- display_mes : génération des courbes avant affichage
- calc_intespec : appel à CALC_SPEC pour le calcul des inter spectres
- calc_coherence : appel à CALC_SPEC pour le calcul des fonctions de cohérences
- calc_transfert : appel à CALC_SPEC pour le calcul des fonctions de transfert
- crea_tab_fonc: création d'un concept TABLE_FONCTION à partir d'une sélection de fonctions