Simulación por Elementos Finitos

Dr. Casimiro Gómez González
Facultad de Electrónica, UPAEP
correo: casimiro.gomez@upaep.mx
Tel: 222 229 9428

Otoño 2011

II

Prólogo

El presente manual es para el diseño y simulación de software por elementos finitos tanto mecánico como fluidos. Utilizando Code Aster y Code Saturne. Tambien se utiliza Salome Meca para el preprocesamiento y post procesamiento.

El autor
Casimiro Gómez González
Doctor en Ingeniería Mecatrónica

Índice general

Capítulo 1

Introducción

Code_Aster puede realizar cálculos estructurales de los fenómenos térmicos, cálculos mecánicos, acústicos termo-mecánica y termo-hidro-mecánico, junto con un comportamiento lineal o no lineal y los cálculos de acústica interior.

Las no linealidades están en el comportamiento de los materiales (plasticidad, viscoplasticidad, daño, los efectos metalúrgicos, hidratación y secado del hormigón, ...), grandes deformaciones y rotaciones grandes y el contacto con la fricción. Consulte el folleto de presentación Code_Aster para la presentación de diferentes características.

Los estudios actuales de la industria requieren la aplicación de herramientas para la creación de redes y gráficos, que no forman parte del código. Sin embargo, hay varias herramientas utilizadas para estas operaciones a través de la interfaz de los procedimientos incorporados en el código.

Para llevar a cabo un estudio, el usuario general deberá preparar dos archivos de datos:

- el archivo de malla: Este archivo define la descripción geométrica y topológica de la malla sin elegir en esta fase, el tipo de formulación de elementos finitos utilizado o el fenómeno físico que está siendo modelado. Algunos estudios pueden llevar al uso de múltiples archivos de malla. Este archivo de malla es generalmente producido por los comandos integrados para Code_Aster a partir de un archivo proveniente de un software de mallado utilizado en el pre-procesador (SALOMÉ Gibi, gmsh, IDEAS ...).

  La información que debe contener este archivo son específicos de Code_Aster. Ellos definen entidades clásicas de los métodos de elementos finitos:

  - nodos: los puntos se define por su nombre y por sus coordenadas cartesianas en 2D o 3D.

- Mallas: Las figuras topológicas llamadas planas o volumen (punto, segmento, triángulo, cuadrángulo, tetraedro, ...) que pueden ser aplicados en diferentes tipos de elementos finitos, las condiciones de contorno o cargas.

Para mejorar la seguridad y comodidad de uso de operaciones de modelado y procesamiento de los resultados, podemos definir en el archivo de malla, los niveles más altos de las entidades,que tienen alguna propiedad en común y que pueden ser utilizados directamente por su nombre:

- Grupos de Nodos: listas llamadas de nombres de los nodos
- Grupos de malla: las listas llamada de nombres de mallas

Tenga en cuenta, ahora que todas las entidades geométricas manipuladas (nodos, mallas, grupos de nodos y grupos de mallas) son nombrados por el usuario y disponibles en todo momento por su nombre (8 carácteres máximo).El usuario puede utilizar esta oportunidad para identificar explícitamente ciertas partes de la estructura estudiada y facilitar así la tabulación de los resultados. La numeración de las entidades no se explica: sólo sirve para señalar a los valores internos de las diferentes variables involucradas.

■

Capítulo 2

Introducción a Code Aster

2.1. Como Empezar Code_Aster, tomado del U0.00.01

Code_Aster es un código general para el estudio del comportamiento mecánico de las estructuras. Para entender Code_Aster, se recomienda:

- Consulta de la documentación
- Uso de la interfaz gráfica de usuario astk Code_Aster un ejemplo sencillo,
- El uso de EFICAS para desarrollar su archivo de control

Se recomiendan los tres puntos como la guía esencial para estudiar.

2.2. Documentación

El sitio web (www.code-aster.org) dedicado a Code_Aster es una fuente de información. Para entender Code_Aster, consulte la página de Uso/Introducción:

- Usted puede consultar la documentación general a:
  - Introducción a Code_Aster [U1.02.00]
  - Los principios básicos de la utilización de Code_Aster [U1.03.00].
- Usted puede confiar en ejemplos sencillos para familiarizarse con el uso de Code_Aster.

Para llevar a cabo su estudio, se puede ver:

- En la documentación [U1.04.00], la interfaz de acceso a Code_Aster: astk.
- Todos los casos de prueba de un modelo básico.

A continuación, puede explorar más a fondo con los siguientes documentos:

- Toda la documentación de uso.
- La página de capacitación que ofrece todas las diferentes presentaciones y sobre todo la capacitación "Introducción a la Mecánica Código: Code_Aster" (para acceder a esta página, primero debe identificarse).

2.2.1. Recomendación

La primero es crear un inicio de sesión en el sitio de intranet Code_Aster. Este se ofrece en todas las páginas del sitio.

2.3. Interfaz gráfica astk

astk es la interfaz gráfica que permite organizar los cálculos de Aster: preparar los datos, organizar los archivos, el acceso a las herramientas de prey postprocesamiento, lanzamiento y seguimiento de los cálculos. Su documentación está disponible en el sitio Code_Aster [U1.04.00].

Proponemos primero para familiarizarse con un ejemplo sencillo astk existente. Esto evitará que las dificultades asociadas a acumular astk y las de un nuevo estudio.

2.4. El estudio del comportamiento mecánico de las estructuras

Code_Aster es un código general para el estudio del comportamiento mecánico de las estructuras.

El principal objetivo es la mecánica de sólidos deformables: lo que justifica el número de características unidos al fenómeno mecánico. Sin embargo, el estudio del comportamiento mecánico de los componentes industriales requiere de modelado ante el estrés que están sometidos, o los fenómenos físicos que modifican los parámetros de este comportamiento (fluido interno o externo, la temperatura, los esfuerzos de cambio de fase metalúrgica de origen electro-magnética ...). Por estas

razones, Code_Aster ofrece muchas posibilidades de "encadenamiento" de los fenómenos mecánicos con los fenómenos térmicos o acústicos, o con software externo, y se juntan en un problema "kit" de construcción termo-hidro-mecánica. Aunque Code_Aster se puede utilizar para muchos problemas de análisis estructural (código general), se ha desarrollado especialmente para permitir el estudio de los componentes de los materiales o maquinaria utilizada en el campo de la generación y transmisión de electricidad. Así se dio prioridad a la modelización de estructuras metálicas geomateriales isotrópico y componentes estructurales de material de hormigón armado o compuesto.

El análisis no lineal, tantos Mecánicos como Térmicos, están en el corazón de Code_Aster: para hacer un tratamiento eficaz se han hecho necesarios el desarrollo de algoritmos eficientes y relativamente sencillo de usar, incluso si el objetivo no es operar en "recuadro negro". Para estudios complejos, es necesario comprender la naturaleza de las operaciones realizadas por el código, para que puedas trabajar óptimamente es necesario referirse a los archivos que detallan los métodos y modelos teóricos, agrupados en el Manual de de referencia.

El aseguramiento de la calidad para el desarrollo de software y la implementación de estudios industriales justifica varias opciones:

Que existan códigos de una versión de referencia fijo y documentado, el suministro de algoritmos completos, pero en principio comandos fijos con parámetros de ortogonalidad (independencia del contexto de uso), la meta de realización de modelos utilizables.

Un análisis estructural se lleva a cabo con Code_Aster en la secuencia de una serie de comandos que se describen en un "archivo de comandos" en formato de texto. El motor y el intérprete del archivo de comandos es el lenguaje de script PYTHON. Por tanto, es posible utilizar todas las características proporcionadas por Python. Véase especialmente la documentación [U1.03.01], [U1.03.02] y ejemplos de uso [U1.05.00] y [U1.05.01]. Cada orden (por ejemplo, la lectura de la malla, los datos del material condición, cálculo estático lineal) produjo un "resultado de concepto", todas las estructuras de datos que el usuario puede manipular y la reutilización de los pedidos posteriores del cálculo (por ejemplo, la creación de redes, el material de datos de campo, el campo de desplazamientos ...).

La sintaxis de todos los comandos se describen y analizan los libros de texto U4 y U7 documentación de uso.

Para simplificar la tarea del usuario, hay comandos globales que incluyen la secuencia de las operaciones para un número de casos de cálculo (por ejemplo, lineal estática - comando MECA_STATIQUE, no lineal estático - Comando STAT_NON_LINE, no térmicos comando THER_NON_LINE , etc). Algunos se han desarrollado direc-

tamente en una forma integrada, los demás son macro-comandos de Python que sólo gestionan las llamadas a la unidad de comandos diferentes (como MACRO_MATR_ASSE para calcular y ensamblar las matrices de masa, amortiguamiento y rigidez una estructura). También hay controles macro, especialmente dedicado a las aplicaciones específicas (ver [§ 4]).

Capítulo 3

Comandos de CODE_ASTER

Code Aster es un "solver" de elementos finitos:

- sin herramientas sofisticadas para crear una geometría y para malla
- no hay imágenes post-procesamiento de colores
- Sin interfaces "click 'n' drop"

Se muestra un diagrama a bloques de la simulación por elementos finitos, la primera etapa esta formada por dos bloques: El preprocesamiento donde se realiza el diseño en cad y la generación del mallado, y el archivo de comandos, el cual describe las características de la simulación así como las propiedades del modelo. En la segunda etapa se crea el modelo por elementos finitos y se realiza la simulaci'on con Code Aster. Y como tercera y última etapa se realizá el post procesamiento.

Varias herramientas pueden ser usadas para crear una malla y para visualizar resultados, en la medida que existe el módulo de importar/exportar en Code Aster:

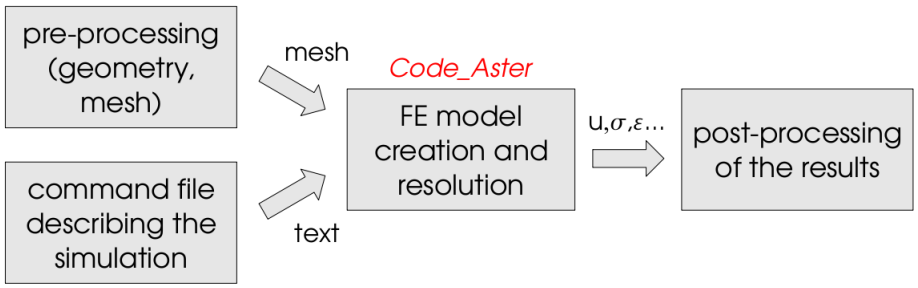

Figura 3.1: Diseño y simulación por elementos finitos

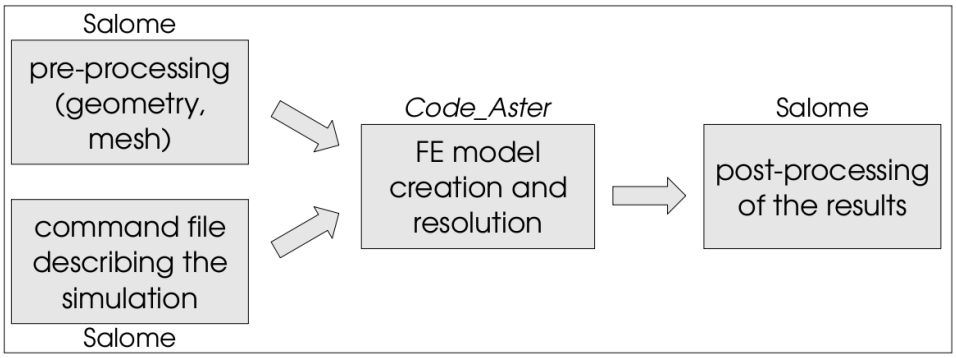

Figura 3.2: Arquitectura Salome Meca

- Gmsh
- I-DEAS
- Gibi
- Cualquier herramienta capaz de importar/exportar mallas en formato MED

MED es un formato de plataforma independiente para el intercambio de datos de mallas (nodos, elementos, grupos de nodos, grupos de elementos) y campos definidos en estos datos (e.g. estreses, tensión, desplazamientos, conjuntos de nivel...). Las librerías para manejar archivos MED estan disponibles libremente.

Aún si Code Aster esta oculto atras de la interfaz gráfica de Salome-Meca, la manera en la cual Salome y Code Aster interectuan determina como el modelo por elementos finitos es creado

Para que los datos introducidos en el "wizard" sean enlazados a la malla sean incluídas en el archivo MED, debes crear grupos de nodos, elementos, bordes y caras en la malla. Puedes definir estos grupos en la geometría del modelo o directamente en la malla. El usar los grupos de la geometría permite la modificación de la malla si necesidad de redefinir los grupos. En el "wizard" puedes usar estos grupos para aplicar las condiciones de frontera, fuerzas, presiones, materiales y demas. Una vez Code aster ha resuelto el modelo FE, los resultados son importados en Salome-Meca a través de dos archivos de texto:

- .mess: Contiene la salida de Code Aster para cada comando. Los errores y mensajes de advertencia son reportados en este archivo.
- .resu contiene la salida de algunos resultados de Code Aster en forma de tabla.

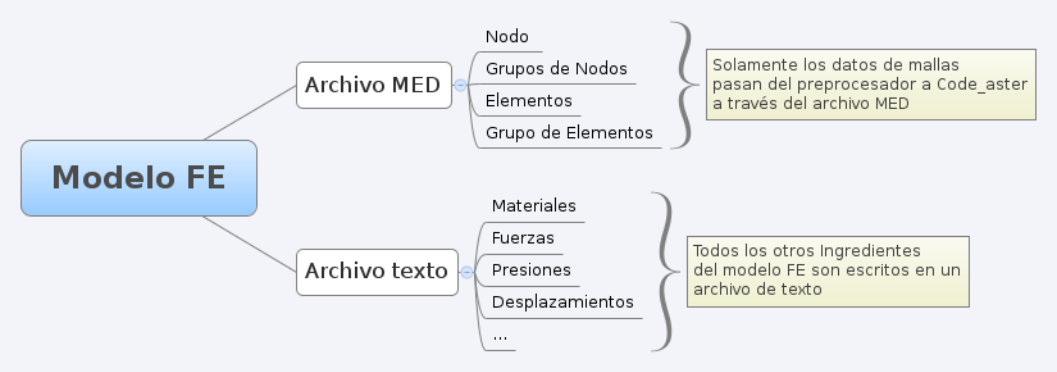

Figura 3.3: Formatos de Salome Meca

Se pueden leer estos archivos directamente dentro de Salome-Meca. La única manera para acceder a todas las características de Code Aster es manejar directamente el archivo de comandos a mano:

- El archivo de comandos es un archivo de texto
- Una interfaz gráfica llamada Eficas puede ser usada para simplificar esta tarea.

Usando Eficas tu puedes:

- Crea un archivo de comando sintacticamente correcto
- Crea un archivo de comando a mano sin recordar todas las opciones de cada comando

Sin embargo, se debe editar el archivo de comando a mano usando un editor de texto si se quiere obtener mas del codigo de Code aster.

- Usando las últimas características del software
- Usando programación Python

3.1. Programas para Code Aster

Los programas para code aster tienen una estructura definida y comandos definidos, el lenguaje de base es python. Empezaremos definiendo los comandos en el orden en el que se recomienda deben aparecer.

3.1.1. Comando DEBUT

Esta función, **DEBUT()**, es usada para iniciar un nuevo análisis. Si este análisis es la continuación de uno previo, se debe utilizar la función **POURSUITE()**

3.1.2. Comando DEFI_MATERIAU

El comando DEFI_MATERIAU define las propiedades mecánicas de un material. Por ejemplo si a una material llamado MA se le desea asignar sus propiedades elásticas se utiliza la palabra clave ELAS, de la siguiente forma

```
MA=DEFI_MATERIAU(ELAS=_F(E=210000000000.0,NU=0.3,),);
```

Cada comando acepta uno o más parámetros identificados por una palabra clave:

```
concepto = COMANDO(keyword_1=xxx,
                   keyword_2=yyy,
                   ...
                   keyword_n=zzz)
```

En el caso de una palabra clave simple, su valor se proporciona directamente despues del =. Esto puede ser un simple valor (ejemplo un entero, un valor real o otro concepto) o una lista de valores simples:

```
concepto_1 = COMANDO_1(simple_keyword_1=3)

concepto_2 = COMANDO_2(simple_keyword_1=1.56,
                       simple_keyword_2=(1.0,0.0,5.6),
                       simple_keyword_3=concepto_1)
```

En el caso de una palabra clave compuesta, sus valores son una lista de palabras claves simples:

```
concepto_1 = COMANDO_1(simple_keyword_1=3.0)

concepto_2 = COMANDO_2(simple_keyword_1=1.56,
                      composed_keyword_1=_F(
                       simple_keyword_2=3.1415,
                       simple_keyword_3=concepto_1),)
```

La lista de palabras claves simples de una palabra clave compuesta se da dentro de `_F(...)`. La letra significa "facteur" en frances.

En el comando:

```
MA=DEFI_MATERIAU(ELAS=_F(E=210000000000.0,NU=0.3,),);
```

La constante elástica es dada por la palabra clave ELAS, en donde el módulo de YOUNG (E) y la razón de Poisson (NU) estan dentro de una palabra clave compuesta de dos palabras claves simples. La definiciones del material estan dentro del concepto llamado MA, el tamaño máximo de los conceptos es de 8 carácteres. Si el concepto a utilizar existe antes de la ejecución del comando, se dispara un error y el análisis es abortado.

Este concepto es un objeto de Python. Esto significa que tiene un tipo asociado:

- El concepto resultado de MA es automaticamente creado por el comando, quien determina tambien el tipo
- El concepto MA puede ser usado como un valor de una palabra clave simple solamente si sus tipos coinciden con el tipo requerido por la palabra clave.

Por ejemplo, si se desean definir dos materiales elásticos (un acero genérico y un aluminio genérico). Estos dos materiales tienen diferentes valores del módulo de Young pero el mismo (mas o menos) valor de la razón de Poisson. Para forzar esta igualdad, se declara de la siguiente manera:

```
steel=DEFI_MATERIAU(ELAS=_F(E=2.06E11,
                            NU=0.3,),);
alu=DEFI_MATERIAU(ELAS=_F(E=0.76E11,
                            NU=steel,),);
```

Esto es incorrecto debido a que `steel` es un concepto de tipo materiales mientras que la palabra clave `NU` necesita un concepto de tipo real. La forma correcta de hacer esto, es usar una variable (no hay que olvidad que estamos escribiendo en código python).

```
poisson=0.3
steel=DEFI_MATERIAU(ELAS=_F(E=2.06E11,
                            NU=poisson,),);
alu=DEFI_MATERIAU(ELAS=_F(E=0.76E11,
                            NU=poisson,),);
```

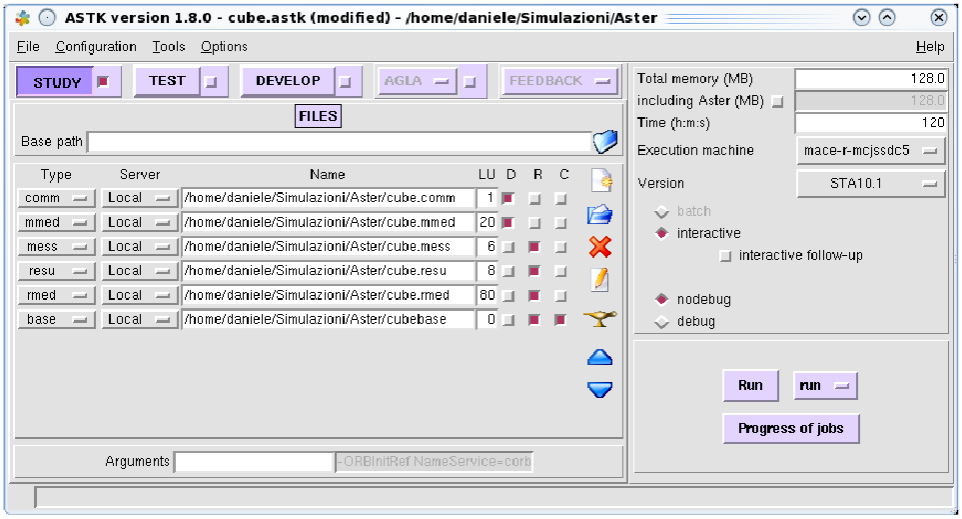

Figura 3.4: Tipos de archivos

3.1.3. Comando LIRE_MAILLAGE

En Code Aster una malla es simplemente una lista de nodos (numeros y coordenadas) y elementos (sus definiciones en términos de formas geométricas y definición de nodos). Todo estará contenido en el concepto MAIL. La malla esta contenida en un archivo MED (`FORMAT='MED'`). Como en el caso de LIRE_MAILLAGE, hay otros comandos que trabajan sobre archivos (de entrada o salida). El nombre de los archivos para leer o escribir por un comando no es especificado en el archivos de comandos pero fuera:

- Esta abstracción facilita la creación de scripts, como en el caso estudios paramétricos donde el mismo análisis esta corriendo usando diferente mallas
- Es realmente práctico en todos los casos en los cuales uno o más de los archivos de entrada/salida cambia (diferentes nombres o paths, diferentes mallas...) nada cambia en el archivo de comando.

Una herramienta gráfica **ASTK**, puede usado para listar todos los archivos necesarios para la simulación. Si se utiliza Salome-Meca, los archivos de entrada/salida son automáticamente enlazadas a el archivo de comandos y no hay que hacer nada.

Esto es cierto en el caso en el cual se usa Salome-Meca para crear un archivo básico de comandos y entonces se puede usar Eficas para modificarlo.

Cada archivo tiene un **tipo** asociado:

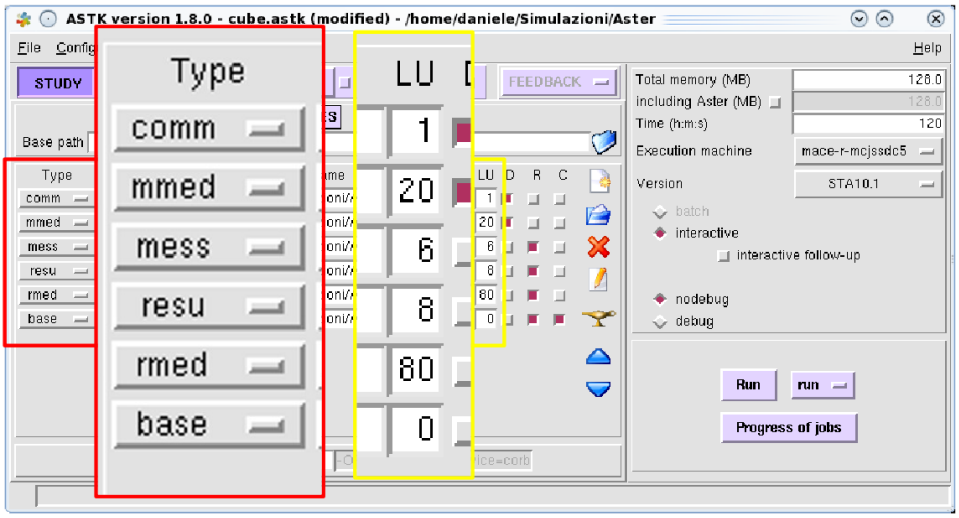

Figura 3.5: Números lógicos asigandos a los tipos de archivos

Cada tipo tiene una **unidad lógica** numérica asociada. Cada comando accede a un archivo (lectura/escritura) a través de su número de unidad lógica y no su nombre.

Los archivos básicos manejados para cada simulación son:

- comm: Es el archivo de comandos
- mmed: Es el archivo MED que contiene la malla.
- mess: Un archivo de texto que contiene la salida del "solver" (mensajes, errores, y advertencias)
- resu Un archivo de texto que contiene los resultados de la simulación en formato de tabla
- rmed: Este es un archivo MED que contiene los resultados de la simlación
- base: Este es una carpeta que contiene la base de datos de salida del solver. Esta será usada para continuar la simulación (en el caso de que se utilice el comando POURSUITE) o para post-procesamiento.

Cada uno de este tipo de archivos tiene un número lógico asigando por defecto (ver fig 3.5).

Si en número lógico no es dado como parámetro del comando, el volor por defecto utilizado es uno. En el siguiente ejemplo

```
MAIL = LIRE_MAILLAGE(FORMAT='MED',);
```

En el código anterior el comando LIRE_MAILLAGE leerá el archivo asociado a valor lógico por defecto para los archivos de mallas con formato MED (rmed).. No es necesario conocer cual es el valor por defecto de los números lógicos si estos no han sido modificados manualmente en ASTK. Al contrario, si se cambiaron los valores lógicos por defecto en ASTK, se debe proporcionar estos nuevos valores a el comando a través de la palabra clave **UNITE**. De tal manera que si, por ejemplo, al tipo de archivo mmed se le cambió el número lógico a 21 y se desea leer este archivo, el comando quedaría:

```
MAIL = LIRE_MAILLAGE(FORMAT='MED',UNITE=21,);
```

3.1.4. Comando AFFE_MODELE

Este comando permite asignar un modelo físico particular y tipo de elemento a una malla. Un modelo FE es creado. Primero debe seleccionarse la malla con el comando MAILLAGE.

```
MODE = AFFE_MODELE(MAILLAGE= MAIL, AFFE=_F(TOUT='OUI',
                   PHENOMENE='MECANIQUE', MODELISATION='3D',),);
```

La palabra compuesta AFFE permite especificar los fenomenos físicos y tipos de elementos para ser asignados a una parte o a toda la mesh:

- `TOUT='OUI'` significa `WHOLE_MESH='YES'`
- `PHENOMENE='MECANIQUE'` especifica que se va a modelar fenómenos mecánicos (no térmicos ni acústicos)
- `MODELISATION='3D'` especifica que usaremos elementos sólidos de 3D (y no, por ejemplo, elementos ladrillos 3D)

El grupo de palabras claves dentro de `_F(...)` puede se repetido para definir todos los modelos. Por ejemplo,en el caso en el cual tanto los elementos solidos en 3D y los ladrillos en 3D estan presentes en la malla, el siguiente comando puede se usado para asignar las propiedades de los elementos:

```
mode = AFFE_MODELE(MAILLAGE=MAIL, AFFE=(_F(GROUP_MA='solid',
                   PHENOMENE='MECANIQUE',
                   MODELISATION='3D',),
                  _F(GROUP_MA='beam',
                     PHENOMENE='MECANIQUE',
                     MODELISATION='POU_D_E',),),)
```

Solid y *beam* son dos grupos de elementos (`GROUP_MA`) definido dentro de la malla MAIL (con la ayuda de Salome).

3.1.5. Comando AFFE_MATERIAU

Este comando permite asignar un material a elementos de una malla. La sintaxis es muy simular a las discutidas anteriormente. El material debe ser definido anteriormente usando el comando DEFI_MATERIAU. El concepto creado por el comando es un campo material, una entidad poco usual en los solvers de elementos finitos. Supongamos que se definieron dos materiales por ejemplo *steel* y *alu* de las páginas anteriores. Y supongamos que se han definido dos grupos de elementos en la malla, cada una conteniendo elementos del mismo material:

- **ELsteel** que contiene los elementos para acero
- **ELalu** contiene los elementos para aluminio

La unión de los dos grupos cubre los elementos de la malla, esto es cada elemento de la malla pertenece a alguno de los dos grupos.

La asignación de los materiales puede realizarse de la siguiente manera:

```
mate = AFFE_MATERIAU(MAILLAGE=MAIL,
                     AFFE=(_F(GROUP_MA='ELsteel', MATER=steel,
                           _F(GRUOP_MA='ELalu', MATER=alu,),),)
```

Otra solución puede ser usar la regla superposición:

- Si mas de una propiedad es asignada a la misma entidad, solamente la última es considerada.
- En otras palabras, si cada asignación sobreescribe a la previa.

La versión de sobreescritura de la asignación es la siguiente:

```
mate = AFFE_MATERIAU(MAILLAGE=MAIL,
                     AFFE=(_F(TOUT='OUI', MATER=steel,
                           _F(GRUOP_MA='ELalu', MATER=alu,),),)
```

En el comando anterior sucede los siguiente:

- Primero, el material *steel* es asignado a todos los elementos de la malla MAIL.
- Entonces, el material *alu* es asignado solamente a los elementos en el grupo **ELalu**
- El material *steel* asignado previamente a estos elementos (**ELalu**) es sobre escrito debido a la segunda asignación

El uso de la regla de sobreescritura puede ser realmente efctivo en muchas situaciones. Aún es este ejemplo simple, la ventaja será:

- Solamente un grupo de elementos (**ELalu**) puede ser definido y mantenido en la malla.
- Una material es asigando a todos los elementos de la malla. No hay riesgo que uno o mas elementos no tengan material asignado porque esten fuera de los dos grupos definido (**ELsteel** y **ELalu**) consecuencia de un error en la creación de los grupos.

3.1.6. Comando AFFE_CHAR_MECA

Este comando permite asignar cargas mecánicas, condiciones de frontera y restricciones cinemáticas a el modelo por elementos finitos. Las opciones:

- DDL_IMPO asiga un valor a uno o mas grados de libertad de un nodo.
- PRES_REP aplica una presión un dominio 2D/3D.

Un ejemplo de código es:

```
CHAR = AFFE_CHAR_MECA(MODELE = MODE,
                      DDL_IMPO = _F(GROUP_MA='Down',
                                    DX=0.0,
                                    DY=0.0,
                                    DZ=0.0,),
                      PRES_REP = _F(GROUP_MA='UP',
                                    PRES=100.0,),);
```

3.1.7. Comando MECA_STATIQUE

Permite en ensamblaje del conjunto discreto de ecuacione por elementos finitos y resolverlas usando un modelo lineal estático Mecánico. Al contrario de otros solvers por elementos finitos, en este punto el usuario debe especificar el modelo, los materiales y las cargas que será aplicadas y usadas en el ensamble.

Esto significa que no todos los materiales definidos, cargas, condiciones de frontera y modelos serán usados durante el ensamblado del modelo por elementos finitos que será resuelto.

El comando MECA_STATIQUE no puede ser usado si hay presente no linealidades (ya sean geométricas y/o en el comportamiento del material). Si es el caso debe usarse el comando STAT_NON_LINE. Similarmente, otros comandos deben usarse en el caso de una simulación dinámica (DYNA_NON_LINE, DYNA_LINE_TRANDYNA_LINE_MODAL, DYNA_LINE_HARM...). Otros comandos pueden ser usados en el caso de simulación térmica (THER_LINEAIRE, THER_NON_LINE, THERM_NON_LINE_MO)

```
RESU=MECA_STATIQUE(MODELE=MODE,
                   CHAM_MATER=MATE,
                   EXCIT=_F(CHARGE=CHAR,),);
```

El concepto creado por el solver (RESU) contiene los resultados del análisis: El campo de desplazamiento del modelo. Esto es un campo nodal.

3.1.8. Comando CALC_ELEM

Este comando permite calcular uno o mas campos asociados a un elemento, como por ejemplo estres, compresión etc. Los campos a ser calculados son especificados con la palabra clave OPTION. Los calculos son realizados iniciando con el concepto RESU (RESULTAT=RESU) el cual contiene los resultados de la simulación (solamente desplazamientos nodales).

```
RESU=CALC_ELEM(reuse=RESU,
               MODELE=MODE,
               CHAM_MATER=MATE,
               RESULTAT=RESU,
               OPTION=('SIGM_ELNO_DEPL', ' EQUI_ELNO_SIGM',),
               EXCIT=_F(CHARGE=CHAR,),);
```

El concepto creado por el comando es RESU. Este ya existia porque fue creado por el comando MECA_STATIQUE.

El objetivo del comando es enriqueser el concepto: Algunos campos nuevos son adicionados (estos son específicamente usados por la palabra clave OPTION):

- El concepto producido por el comando no debe
- Debemos declarar nuestro objetivo usando la palabra **reuse**
- De lo contrario, debe cambiar el nombre del concepto a la izquierda del signo igual

```
RESU=CALC_ELEM(reuse=RESU,
               RESULTAT=RESU,
               OPTION=('SIGM_ELNO_DEPL', ' EQUI_ELNO_SIGM',),
               ,);
```

Los calculos son realizados empezando desde un concepto de tipo "result" (RESULTAT=RESU). Este tipo de concepto contiene toda la información del modelo ademas de los campos de desplazamiento nodales (el cual es efectivamente el resultado):

- Las palabras claves MODELE, CHAM_MATER y EXCIT son por lo tanto inútiles en este caso y deben ser ignoradas.
- Estas palabras claves deben ser usadas solamente si la información pasada a través de ellos no esta disponible en el concepto proporcionado por RESULTAT

En la figura 3.6 se muestra el significado de las tres partes que forman las opciones del comando OPTION.

En la figura 3.7 se muestran las opciones para el comando OPTION sobre las cuales se aplicará el análisis:

- ELGA (**ELement GAuss**): Los calculos deben hacerse en los puntos Gauss de los elementos.
- ELNO (**ELement NOde**): Los calculos deben hacerse en los nodos de cada elemento, considerado separado de los elementos vecinos. Mas de un valor es calculado para cada nodo, un valor para cada elemento compartiendo ese nodo.
- NOEU (**node**): Los calculos son hechos en cada nodo. Un valor principal de los valores que vienen de cada elemento que comparte en nodo es calculado.

La diferencia entre los valores calculados en ELNO y los calculados en NOEU son debido a la discretización FE.

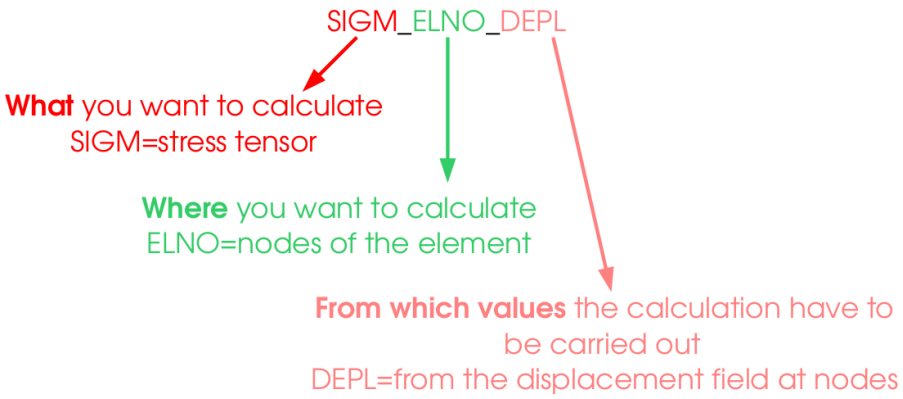

Figura 3.6: Significado de los campos en la palabra OPTION

Figura 3.7: Opciones para la aplicación de los análisis

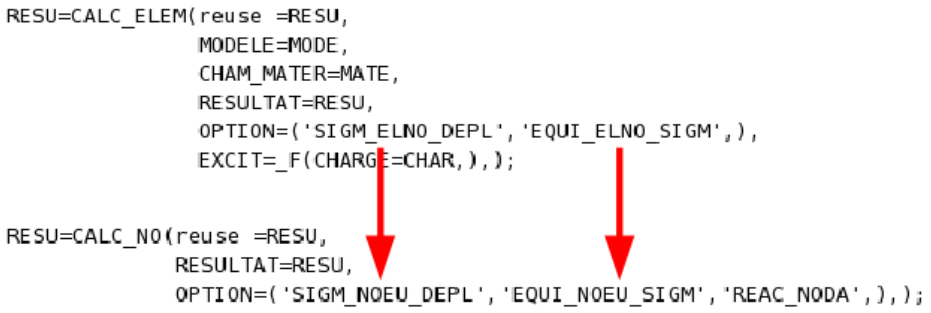

Figura 3.8: Como usar el comando CALC_NO

3.1.9. Comando CALC_NO

Este comando permite calcular uno o mas campos asociados a un nodo, este comando trabaja como CALC_ELEM. Se pueden calcular campos nodales: Fuerzas de reacción, fuerzas nodales y todos los campos de tipo xxxx_NOEU_xxxx. El campo xxxx_NOEU_xxxx pueden ser calculado solamente si el campo correspondiente ELNO existe en el concepto que nos da RESULTAT.

En la figura 3.8 se muestran la correspondencia que debe haber entre el comando SIGM_NOEU_DEPL y EQUI_NOEU_SIGM con los comandos xxxx_ELNO_xxxx, es decir, si se intenta calcular xxxx_NOEU_xxxx sin calcular primero xxxx_ELNO_xxxx CODE ASTER indicará una advertencia y NO realizará los cálculos.

3.1.10. Comando IMPR_RESU

Este comando permite imprimir los resultados en uno o más archivos de salida. El formato es escogido por la palabra clave FORMAT. Si no se especifica ningun campo (NOM_CHAM), todos los campos dentro del resultado (`RESULTAT=RESU`) serán escritos en el archivo de salida. Es importante mencionar que ningún concepto es creado con este comando. Muchos formatos son soportados para salida:I-DEAS, MED, CASTEM, GMSH, RESULTAT. Si `FORMAT='RESULTAT'` es usado, la salida se escribira en un archivo de texto en una o más tablas. Por ejemplo

```
IMPR_RESU(FORMAT='RESULTAT',
          RESU=_F(MAILLAGE=MAIL,
          RESULTAT=RESU,
          NOM_CHAM=('SIGM_NOEU_DEPL', 'EQUI_NOEU_SIGM', 'DEPL',),),);
```

3.1.11. Comando FIN

Todos los archivos de comandos deben terminar con este comando e indida en fin del archivo de comandos.