Interface d'accès à *Code_Aster* : astk

Résumé :

Ce document présente astk (contraction de *Aster* et Tk), l'interface permettant d'organiser ses calculs *Code_Aster*.
On décrit la logique de fonctionnement d'astk, l'interface graphique elle-même, bsf qui permet d'explorer les fichiers sur les différents serveurs, puis on explique comment l'utilisateur peut configurer astk en fonction de ses besoins.

Pour un accès rapide à *Code_Aster*, on montre à partir d'exemples comment lancer une étude et comment faire une étude paramétrique.

Pour les personnes ayant accès au serveur de calcul *Code_Aster* du réseau EDF, on explique comment émettre une fiche d'anomalie, accéder aux fiches de retour d'expérience (REX).

Enfin, dans une seconde partie, on présente as_run qui offre de nombreuses possibilités en ligne de commande.

Sont décrites ici les fonctionnalités d'astk version 1.13 qui gère toutes les versions supportées de *Code_Aster*.

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

Table des matières

1 Présentation et notions générales....4
1.1 Démarrage....4
1.2 Fonctionnalités....4
1.3 Mode de fonctionnement....4
1.4 Serveurs astk....5
1.5 Profil astk....6
2 Description de l'interface graphique....7
2.1 Barre de menu....7
2.1.1 Menu Fichier....8
2.1.2 Menu Configuration....9
2.1.3 Menu Outils....11
2.1.4 Menu Options....12
2.1.5 Menu Aide....13
2.2 Les onglets....14
2.2.1 Onglet ETUDE....14
2.2.1.1 Liste des types pour ETUDE....15
2.2.2 Bouton REX....16
2.3 Paramètres de soumission....17
2.4 Barre d'état....19
2.5 Suivi des jobs....19
3 Boite de sélection de fichiers : bsf....20
3.1 Navigation....21
3.2 Menu Fichier et menu contextuel....22
3.3 Menu Affichage....22
4 Configuration....23
4.1 Serveurs....24
4.2 Configurations....24
4.2.1 Préférences Générales....24
4.2.2 Préférences Réseau....24
4.3 Outils....25
5 Comment faire une étude ?....26
5.1 Création du profil....26
5.2 Sélection des fichiers....26
5.2.1 Définition d'un chemin de base....26
5.2.2 Ajout de fichiers existants....26
5.2.3 Ajout de fichiers….....27
5.2.3.1 …en insérant une ligne vide....27
5.2.3.2 …avec « Valeur par défaut »....27

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

5.2.4 Supprimer un fichier....28
5.3 Lancement du calcul....28
5.4 Consultation des résultats....29
5.5 Utilisation des outils....29
5.6 Fonctionnalités avancées....30
5.6.1 exectool....30
6 Comment lancer une étude paramétrique....31
7 Lancement de calculs en parallèle....31
7.1 Distribution de calculs....31
7.1.1 Utilisation des ressources disponibles....31
7.1.2 Délai d'expiration....32
7.2 Activation du parallélisme de Code_Aster....32
7.3 Exécutions multiples....32
8 Utilisation de as_run....33
8.1 Pour l'utilisateur....33
8.2 Pour le développeur....34
8.3 Pour maintenir une installation locale....34
8.4 Pour les interfaces lançant des calculs....35
8.5 Pour les tâches d'administration....35
8.6 Superviser le lancement et le retour d'un calcul....36
8.6.1 Hypothèses....36
8.6.2 Commandes....36
8.6.2.1 Récupération des infos du serveur (permet également de tester la connexion)....36
8.6.2.2 Lancement d'un calcul....36
8.6.2.3 Interrogation....37
8.6.2.4 Actualisation de l'état du job....37
8.6.2.5 Récupération des résultats....37
8.6.2.6 Supprimer le job....37
8.6.2.7 Consulter la fin de l'output d'un job en cours....38
8.6.2.8 Éditer le fichier output....38
8.7 Options....38
8.7.1 Précisions supplémentaires....39
9 Questions Fréquentes....40

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

1 Présentation et notions générales

La mise à disposition d'un outil comme *Code_Aster* qui fonctionne sur de multiples plates-formes nécessite une interface graphique pour simplifier le travail de l'utilisateur.
Par ailleurs, l'évolution des moyens de calcul centralisés et la diffusion en libre de *Code_Aster* ont nécessité le développement d'un produit portable dont l'architecture puisse s'adapter aux différentes configurations informatiques.

astk est l'interface graphique qui permet d'organiser les calculs *Aster* : préparer les données, organiser les fichiers, accéder aux outils de pré et post-traitement, lancer et suivre l'évolution des calculs. astk permet également de choisir la version de *Code_Aster* à utiliser parmi celles disponibles (stable, testing…).

Le nom « astk » vient de la contraction de *Aster* et Tk qui est la librairie graphique utilisée.

1.1 Démarrage

Dans Salome-Meca, ouvrir le module Aster, puis lancer astk depuis son icône ou bien depuis le menu *Aster/Tools/Run ASTK* .
En ligne de commande, il faut exécuter : `salome shell -- astk`, `salome` étant un script disponible dans l'installation de Salome-Meca. On peut le trouver en remplaçant `nnn` par l'indice de version dans : `/opt/salome-meca/nnn/appli_Vnnn/salome`.
En dehors de Salome-Meca, la commande `astk` a, en général, été placée dans le chemin par défaut des utilisateurs, l'interface s'ouvre avec une configuration par défaut (voir [*§ 4] Configuration*).
Si la commande n'est pas trouvée, il faut lancer : `[aster_root]/bin/astk` où `[aster_root]` est le répertoire d'installation de *Code_Aster* (contactez éventuellement votre administrateur).

1.2 Fonctionnalités

ETUDE *:* astk permet de lancer un calcul *Aster* sur la machine locale (par exemple dans le cadre d'une utilisation sur ordinateur personnel), sur un serveur de calcul départemental ou bien sur le serveur de calcul *Aster* de EDF-R&D (accès restreint à EDF et ses prestataires d'études).

REX *:* Faire vivre le code passe par la prise en compte des remarques, besoins, et avis des utilisateurs. Des fiches de retour d'expérience peuvent être émises et consultées par les utilisateurs depuis l'interface, les développeurs peuvent y répondre (accès au serveur EDF requis). Cette fonctionnalité est réservée à EDF et à ses prestataires.

MULTI-MACHINES *:* Les fichiers nécessaires à ces différentes actions (fichiers de maillage, source, résultats…) peuvent être répartis sur différentes machines sur le réseau (déclarées dans astk), astk assurant le transfert et la compression/décompression.
Le passage d'un coupe-feu n'est pas proposé.

OUTILS *:* L'utilisateur peut lancer différents outils pré-définis et configurer ceux dont il a besoin (mailleur, outil de post-traitement, éditeur…).

BSF *:* Un navigateur est fourni (appelé `bsf`, boîte de sélection de fichiers), il permet de parcourir les systèmes de fichiers des machines distantes définies et d'effectuer des opérations courantes sur les fichiers : copie, suppression, changement de nom, impression, ou encore d'ouvrir une fenêtre de commandes sur ces machines.

ASJOB *:* On peut suivre les calculs, leur état (notamment dans le cas de lancement en batch) depuis la fenêtre « Suivi des jobs », aussi appelée `asjob`.

1.3 Mode de fonctionnement

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

L'architecture client/serveur autorise une séparation nette entre l'interface (client) et les outils utilisés pour accéder au code (services). Le point d'entrée du serveur est `as_run` : il permet d'utiliser la version de référence en batch et/ou d'accéder à toutes les versions disponibles sur un réseau ou en local.
Le protocole de communication par défaut entre les différentes machines est ssh pour les commandes shell et scp pour la copie de fichiers. On peut toutefois encore utiliser rsh et rcp mais ceci est déconseillé sur la plupart des réseaux.

Exemple : Le client (c'est-à-dire l'interface lancée par la commande `astk` ou `codeaster-gui`) demande le lancement d'un calcul sur un serveur de calcul.
L'interface exécute un service `as_run` qui se charge de copier les fichiers de données sur le serveur (dans un répertoire intermédiaire défini par celui-ci et partagé par tous les noeuds de ce serveur), de demander au serveur de démarrer le calcul. Les résultats du calcul restent sur le serveur d'exécution (dans le répertoire intermédiaire s'ils sont censés être recopiés vers le client ou dans leur répertoire définitif si leur destination est sur le serveur de calcul). Les résultats sont rapatriés le cas échéant sur la machine cliente lorsque de l'état du calcul est ENDED dans le suivi des jobs ; c'est-à-dire après une actualisation manuelle ou automatique.

De ce fait, seule la connexion dans le sens machine client vers serveur est sollicitée (clé ssh valide pour une connexion sans mot de passe). Il n'y a plus de connexions inverses initiées par le serveur vers le client (comme dans les versions antérieures à la 1.9).
De plus, une fois le calcul soumis, la liaison réseau entre le client et le serveur peut être interrompue sans risque de perdre le calcul car les données ont déjà été déposées sur le serveur et les résultats ne seront rapatriés que sur demande du client (interface).

Pour les puristes : `as_run` n'est pas véritablement un serveur dans les faits, il n'y a pas de démon à l'écoute sur un port particulier. Il est démarré à la demande.

1.4 Serveurs astk

Pour astk, un serveur est :

- soit un serveur de calcul *Aster* : une machine sur laquelle on peut trouver `as_run`, c'est-à-dire l'ensemble des services qui permettent d'accéder au code ; on pourra lancer des calculs via ce serveur et utiliser des fichiers sur ce serveur pour un calcul,
- soit un serveur de fichiers : on pourra simplement utiliser des fichiers sur ce serveur lors d'un calcul.

Le serveur appelé "Local" est en fait un serveur de fichiers (seules les informations de connexion sont nécessaires pour un serveur de fichiers, or on les connaît facilement sur la machine locale.
Si l'on souhaite lancer des calculs sur la machine locale (sur laquelle sont installées une version d'*Aster* et la partie serveur d'astk), il faut aussi déclarer cette machine comme un serveur de calcul (Local et Machine dans l'exemple suivant et §2.1.2).

Dans ce document, on appellera « serveur de référence » la machine sur laquelle est gérée la version qualifiée de *Code_Aster* en interne EDF. Certaines fonctionnalités sont accessibles uniquement si l'utilisateur a un accès à cette machine.

Exemple de configuration : (voir [§ 2.1.2])
astk est utilisé sur la machine de nom `mach00`, sur laquelle est installé *Code_Aster*. On a accès à un serveur de fichiers départemental `file01`, les fichiers de ce serveur sont accessibles (montage nfs) depuis deux machines de calcul `comp02`, `comp03`. On a aussi accès à un cluster avec N nœuds de calcul dont la machine frontale est `front04`.

On a alors :

- « Local » (label réservé) : on l'a toujours pour explorer les fichiers qui se trouvent sur `mach00` ;

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

- « Machine » (label quelconque, la procédure d'installation fixe ce nom à la valeur retournée par la commande `` `uname -n` ``) : dont l'adresse IP est celle de `mach00` qui est le serveur de calcul (services as_run installés) de la machine locale ;
- « Depart » (label quelconque) : dont l'adresse IP est celle de `file01`, qui permet d'explorer les fichiers hébergés par le serveur de fichier, configuré comme un serveur de calcul Aster (services as_run installés) ayant `comp02` et `comp03` vus comme des nœuds de calcul (`comp0i` peut être la même machine que `file01`) ;
- « Cluster » (label quelconque) : dont l'adresse IP est celle de `front04` qui est un serveur de calcul (services as_run installés) ayant N nœuds, la seule machine accessible étant `front04`.

La différence entre « Depart » et « Cluster » est qu'en général les stations de calcul sont accessibles directement (on pourrait donc soumettre un calcul en interactif sur l'une ou l'autre), alors que pour un cluster, les utilisateurs ne voient en général que la machine frontale, la répartition étant faite sur les nœuds par un séquenceur de travaux batch.

On peut aussi bien avoir plusieurs serveurs de calcul « Depart » ou « Cluster » que n'avoir uniquement le poste « Local+Machine ».

1.5 Profil astk

Un profil astk est un fichier qui contient toutes les informations relatives à une étude : l'emplacement des fichiers de l'étude, en donnée, en résultat, le type associé à chaque fichier, les paramètres de soumission du calcul (mémoire, temps, machine de calcul, batch/interactif…).

Le profil contient également des paramètres sur l'interface elle-même pour reprendre une étude exactement dans le même état où on l'avait laissée.
Le profil est enregistré sur demande de l'utilisateur (menu *Fichier/Enregistrer* ou *Enregistrer sous…*) et à chaque lancement d'un calcul (il s'agit du fichier dont l'extension est, par convention, `.astk`).

On parlera plus loin du fichier « export » (extension `.export`) qui est une représentation simplifiée du profil astk : il ne contient que les informations sur les fichiers et les paramètres nécessaires au calcul (et aucune information sur l'interface graphique).

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

2 Description de l'interface graphique

L'interface graphique (IHM) se décompose en 4 parties :

1. Une barre de menu
2. Les onglets
3. Les paramètres de soumission
4. La barre d'état
5. Zone des arguments Aster

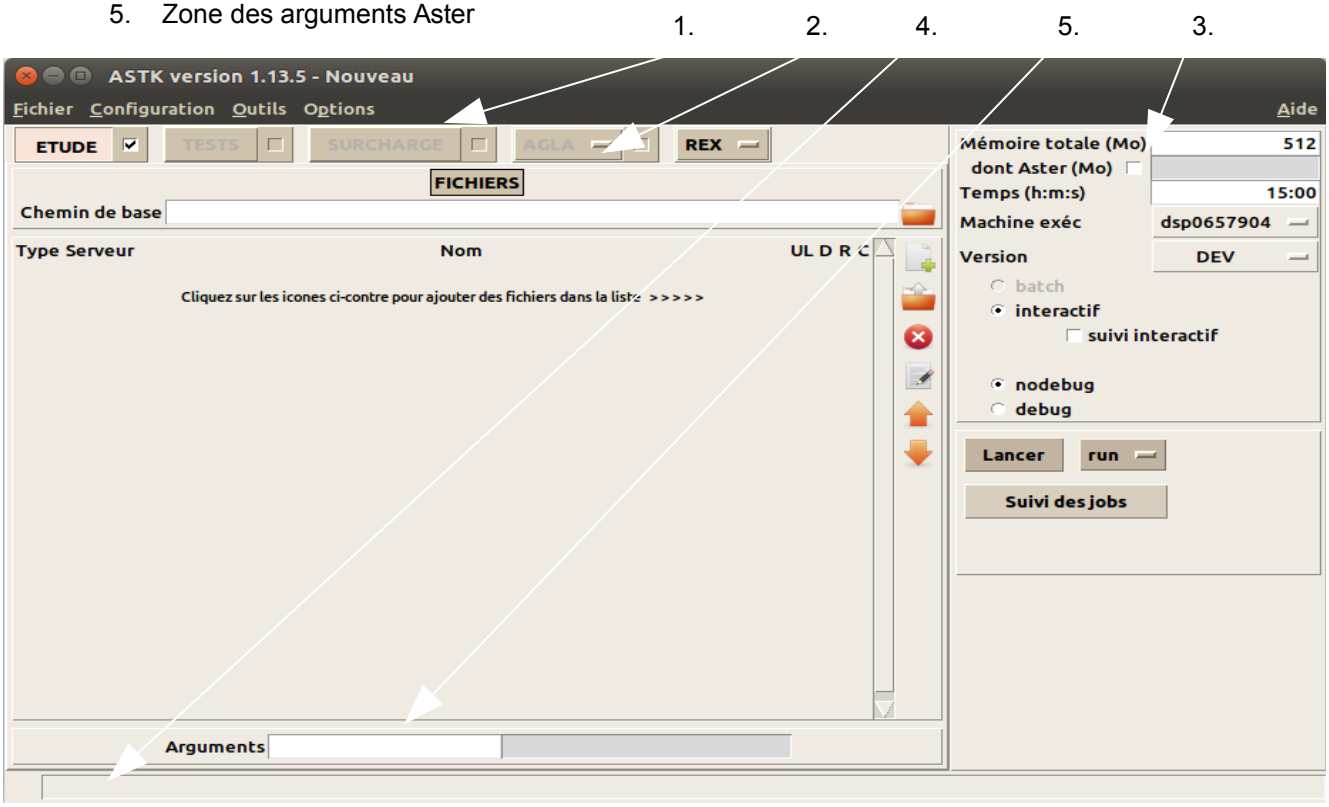


Figure 2-1: Fenêtre principale

Les onglets TESTS, SURCHARGE et AGLA sont maintenant inactifs.

2.1 Barre de menu

Le menu « Aide » permet également d'accéder à la description des menus.

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

2.1.1 Menu Fichier

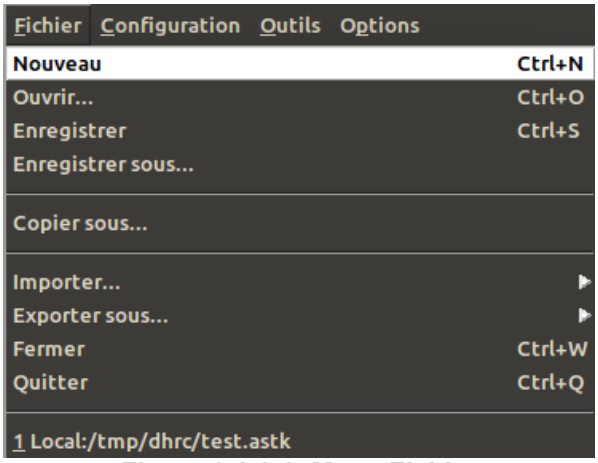


Figure 2.1.1-1: Menu Fichier

- **Nouveau** : Permet de créer un nouveau profil.
- **Ouvrir** : Ouvre un profil créé auparavant par `astk`.
- **Enregistrer** : Sauvegarde le profil courant (ceci est fait automatiquement lors du lancement d'un calcul).
- **Enregistrer sous** : Sauvegarde le profil courant en donnant le choix du nom du profil et de son emplacement.
- **Copier sous** : Copie le profil courant dans un autre répertoire et propose de copier les fichiers (en données et/ou résultats) qu'il référence.
- **Importer** :
  - `.export astk` : Relit un fichier `.export` d'une exécution précédente (cela peut être le fichier `*.pNNN` du répertoire `$HOME/flasheur`).
  - `.export astk (mode ajout)` : contrairement à l'import classique qui part d'un profil vierge, les fichiers et répertoires présents dans le fichier .export sont ajoutés au profil actuel. Les paramètres et arguments sont ignorés. Cela permet par exemple d'importer une étude dans un profil de surcharge.
  - `.export d'une fiche REX` : Importe les fichiers attachés à une fiche du REX dont on donne le numéro.
  - `cas-test` : Importe les fichiers nécessaires au lancement d'un cas-test. Les fichiers sont pris sur la machine d'exécution sélectionnée.
  - `cas-test (mode ajout)` : idem + les fichiers sont ajoutés à ceux déjà présents dans le profil (y compris la surcharge).
- **Exporter sous** : Permet de produire le fichier `.export` du profil courant.
- **Fermer** : Fermer le profil courant
- **Quitter** : Termine l'application
- Les N derniers profils ouverts sont directement accessibles à partir du menu Fichier.

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

2.1.2 Menu Configuration

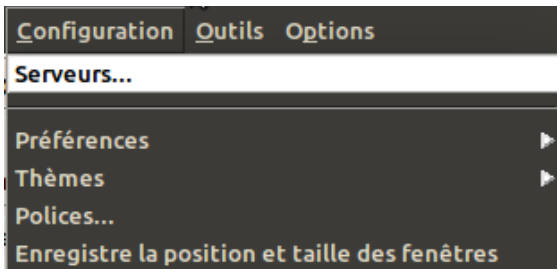


Figure 2.1.2-1: Menu Configuration

- **Serveurs** : Permet de modifier la configuration des serveurs : serveurs de calcul ou serveurs de fichier (voir Figure 2.1.2-2).
  Pour les serveurs de calcul, on choisit le mode de téléchargement de la configuration *Aster* : le serveur retourne la liste des versions disponibles, le mode d'exécution (batch et/ou interactif) et les limites associées.
  Il faut indiquer le 'login' à utiliser pour se connecter au serveur, et le répertoire où est installé *Code_Aster*.
  Pour les serveurs de fichier, choisir "aucun" comme mode de téléchargement de la configuration Aster.

- **Préférences** : Définit les préférences de l'utilisateur :

  - Générales (Voir Figure 2.1.2-3)
    Pour `astk` :
    Nom, prénom, email seront utilisés pour le lien vers l'outil de gestion du retour d'expérience (émission, consultation des fiches d'anomalies, d'évolution...).
    Le chemin d'accès à l'éditeur et au terminal (fenêtre xterm) sur la machine locale, la version à sélectionner par défaut, le nombre de profils rémanents dans le menu Fichier, le niveau de message (debug).
    On peut choisir d'afficher tous les types de fichiers connus, une liste restreinte ou une choix personnel, triés ou non par nom (voir §2.2.1).
    Pour le suivi des jobs :
    Le nombre de lignes affichées lors de la consultation d'un calcul en cours d'exécution (tail), et la fréquence d'actualisation automatique en minutes.
  - Réseaux (Voir Figure 2.1.2-4)
    Le nom de domaine réseau de la machine et si le mode DHCP/VPN (adresse IP dynamique) est actif, possibilités de fixer la variable DISPLAY pour l'affichage des applications externes, les protocoles de communication utilisés pour les commandes shell (rsh ou ssh) et la copie de fichiers (rcp, scp ou rsync).
    Si on travaille uniquement en local ou si toutes les machines sont dans le même domaine réseau, on peut laisser le nom de domaine vide et ignorer l'avertissement au démarrage.
    Attention : utiliser l'option DISPLAY en connaissance de cause et uniquement si la valeur par défaut ne convient pas. Laisser le champ vide pour laisser astk déterminer seul le DISPLAY (en fonction de sa valeur au lancement d'astk, du nom de domaine...).
- **Couleurs** : `Classique`, `KDE3 Crystal`, `Nostalgique`, `Personnalisé`,...
- **Polices**
- **Enregistre** la position et taille des fenêtres

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

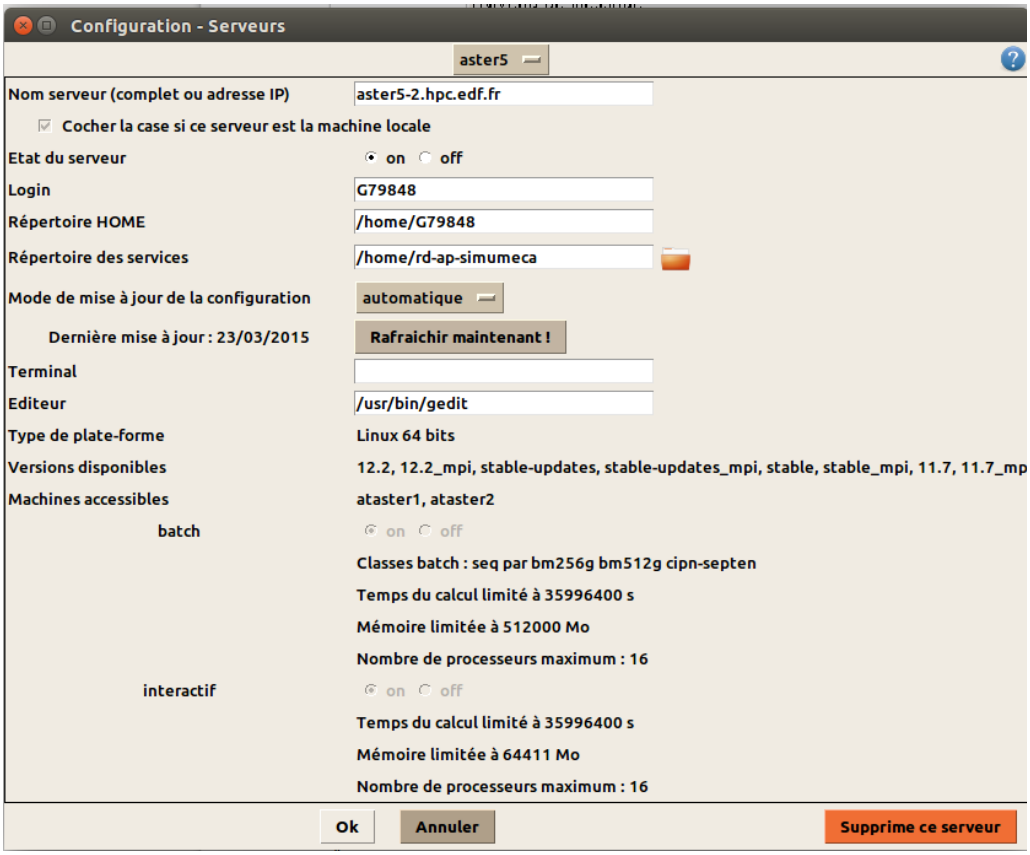


Figure 2.1.2-2: Menu Configuration/Serveurs

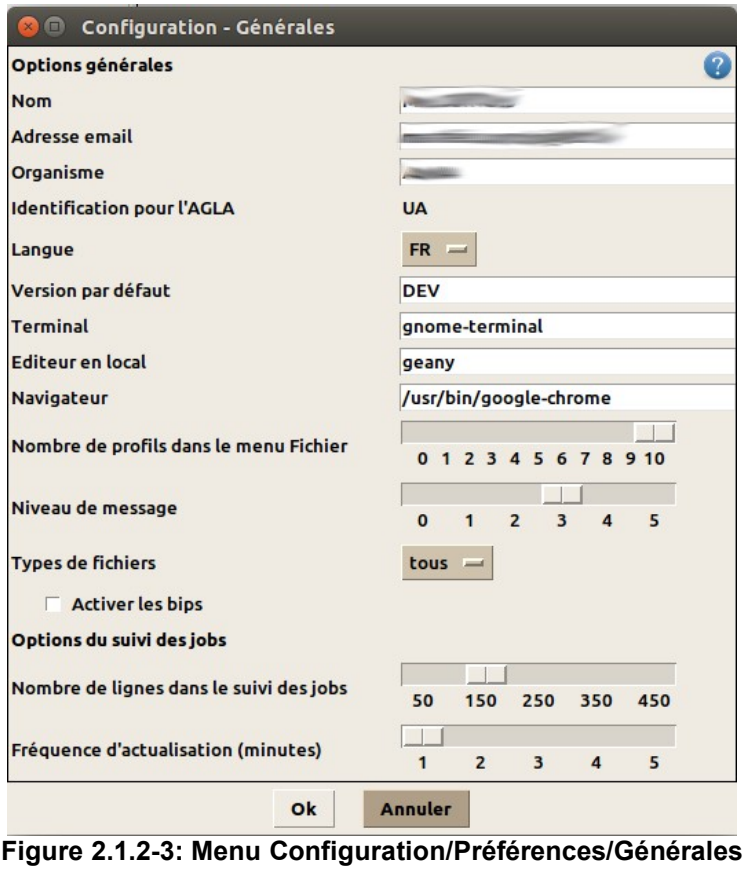


Figure 2.1.2-3: Menu Configuration/Préférences/Générales

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)



Figure 2.1.2-4: Menu Configuration/Préférences/Réseau

Remarques

Que l'on choisisse le protocole rsh/rcp ou ssh/scp, les connexions doivent fonctionner sans mot de passe : fichiers .rhosts correctement remplis ou clés privées/publiques SSH valides.
Selon ce que retourne la commande « uname » sur la machine, il arrive qu'astk pense être sur une machine distante et fasse alors un rsh ou ssh pour exécuter une commande locale... Dans ce cas, la connexion triviale de la machine vers elle-même doit également être configurée.

2.1.3 Menu Outils

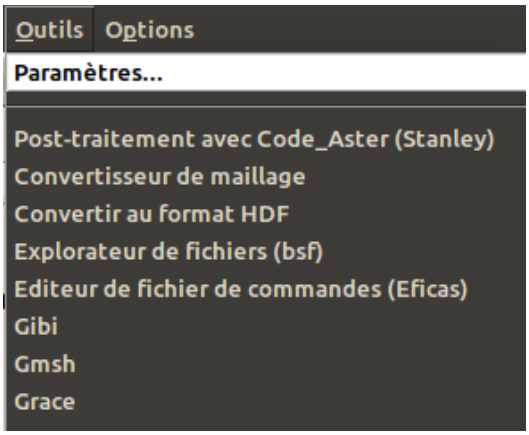


Figure 2.1.3-1: Menu Outils

- **Paramètres** : Permet de configurer le lancement des outils.
  Certains outils sont prévus en standard (sans forcément être disponibles sur la machine locale) :
  - `Post-traitement avec Code_Aster` : Outil de post-traitement, commande STANLEY,
  - `Convertisseur de maillage` : Permet de convertir un maillage d'un format vers un autre, les formats disponibles sont : `aster`, `med`, `gmsh`, `ideas` et `gibi`,
  - `Convertir au format HDF` : Permet de convertir une base classique Aster au format HDF,
  - `bsf` : Explorateur de fichiers multi-machines,
  - `Eficas` : Editeur de fichier de commandes *Aster*,
  - `Gibi` : Mailleur et outil de post-traitement (utilisable gratuitement avec *Aster*),
  - `Gmsh` : Mailleur et outil de post-traitement (libre),
  - `grace` : Traceur de courbes 2D.

Lorsque l'on choisit de lancer Stanley, astk cherche dans le profil les bases disponibles (par ordre de préférence une `base` en résultat, s'il n'y en a pas, une `bhdf` (base HDF) en résultat, sinon une `base` en donnée et finalement, une `bhdf` en donnée), produit un profil temporaire à partir du profil

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

courant avec un fichier de commandes Aster qui commence par `POURSUITE()` et qui lance `STANLEY()`.

L'utilisateur peut ajouter ses propres outils, paramétrer le chemin d'accès aux outils (y compris modifier la commande d'accès aux outils standards), définir à quels types de fichiers associer l'outil et préciser si on peut utiliser l'outil sur un fichier distant.

Les codes suivants peuvent être utilisés dans la ligne de commande :
- @F : chemin absolu du fichier sélectionné,
- @R : répertoire contenant le fichier sélectionné,
- @f : nom du fichier (sans le répertoire),
- @D : adresse du DISPLAY (celui connu au moment du lancement de l'interface).

Les outils sont appelés soit à partir du menu Outils, soit à partir du menu contextuel sur un fichier d'une liste ou dans l'explorateur (bouton droit).
Pour sélectionner un fichier, il suffit de cliquer sur son nom dans une liste (dans l'onglet Etude, Tests ou Surcharge).
En double-cliquant sur un fichier, l'association entre le type du fichier et l'outil à utiliser s'appuie sur l'extension du nom de fichier pour la bsf, alors que le type sélectionné par la liste déroulante (cf. description des onglets) prévaut dans astk.

2.1.4 Menu Options

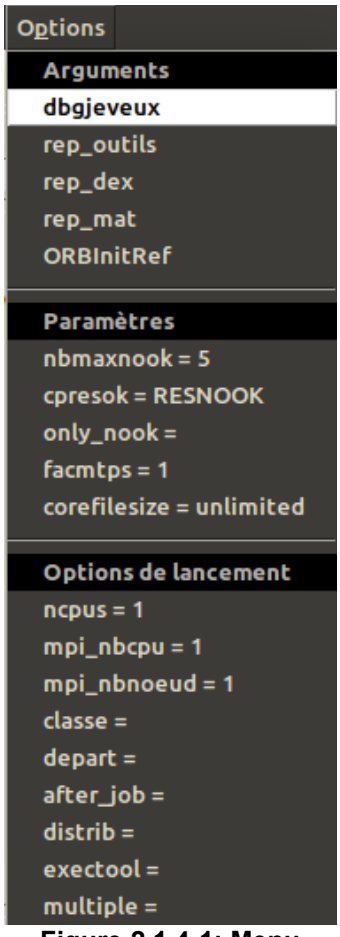


Figure 2.1.4-1: Menu Options

- Arguments :
  Permet de positionner des arguments optionnels qui seront passés sur la ligne de commande de l'exécution de *Code_Aster*.

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

`dbgjeveux` : active un mode différent de la gestion des objets en mémoire pour détecter les écrasements et destruction d'objets,
`rep_outils` : définit le répertoire où sont cherchés les outils appelés depuis *Aster* (onmetis ou gibi par exemple),
`rep_dex` : définit le répertoire où sont cherchées les données externes (données de maillage),
`rep_mat` : définit le répertoire où sont stockées les données du catalogue matériau.
`ORBInitRef` : permet de contacter une session Salomé active (cette valeur est automatiquement positionnée quand astk est lancé depuis Salomé).
Les arguments sélectionnés apparaissent dans la partie grisée de la zone des arguments passés à *Code_Aster*. La partie gauche de cette zone est libre.

- Paramètres
  Il s'agit de définir des paramètres optionnels qui seront écrits dans le fichier .export. Les trois premiers sont utilisés lors du lancement de tests.
  `nbmaxnook` : nombre maximal de cas-tests invalides (NOOK, ARRET_ANORMAL...) avant l'interruption de la liste des tests,
  `cpresok` : permet de choisir quels sont les tests dont on garde les fichiers résultat,
  RESOK : on garde les fichiers des tests OK et NOOK,
  RESNOOK : on ne garde que les fichiers des tests NOOK,
  `facmtps` : facteur multiplicatif du temps des tests (par rapport au temps de référence du para). Utile, par exemple, quand on soumet des tests avec un exécutable construit en mode debug, plus lent.
  `corefilesize` : taille limite du fichier core lors d'un plantage

- Options de lancement
  `ncpus` : définit le nombre de processeurs utilisés par le solveur MULT_FRONT
  `mpi_nbcpu` : définit le nombre de processeurs pour le parallélisme MPI
  `mpi_nbnoeud` : définit le nombre de nœuds pour le parallélisme MPI (Où les `mpi_nbcpu` processeurs seront distribués)
  `classe` : permet de choisir la classe batch (ou le groupe de classe) dans laquelle le calcul sera soumis. Il faut bien évidemment vérifier que la classe existe et que les paramètres temps et mémoire sont compatibles avec cette classe.
  `depart` : permet de différer le départ d'un calcul. L'heure de départ est fournie au format `heure:minute` (pour préciser le jour de départ, la syntaxe dépend du gestionnaire de travaux).
  `distrib` : lance une étude paramétrique (valeur oui/non, voir §6).
  `exectool` : lancement en utilisant un outil particulier (voir §5.6.1)
  `multiple` : exécute le profil sur plusieurs machines (valeur oui/non, voir §7).

2.1.5 Menu Aide

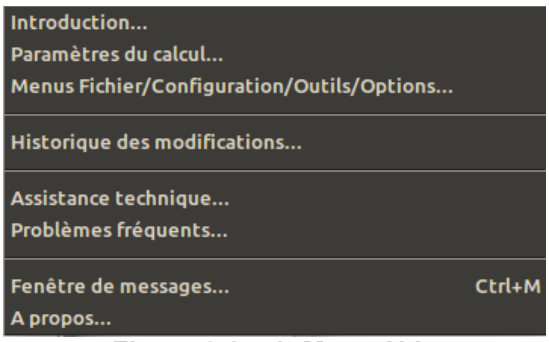


Figure 2.1.5-1: Menu Aide

- Introduction/Paramètres du calcul/Menus : accès au texte d'aide
- Paramètres de calculs : permet de définir la taille mémoire, la version de *Code_Aster* qui sera utilisées, le mode de calcul (Batch ou interractif), …
- Menus Fichier/Configuration/outils/options…

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

- Historique des modifications : Evolution d'astk au fil des versions, nouvelles possibilités, corrections effectuées, anomalies connues...
- Assistance Technique…
- Problèmes fréquents : Quelques questions fréquemment posées avec leurs solutions.
- Fenêtre des messages : Les messages d'informations <INFO>, d'erreurs <ERREUR> sont écrits dans cette fenêtre. Avec un niveau de message supérieur ou égal à 1 (voir [§ 2.1.2]), on obtient plus ou moins d'informations.
- A propos… : l'incontournable fenêtre d'information.

2.2 Les onglets

On trouve cinq boutons dans cette zone. On parle d'onglet quand le contenu que l'on trouve sous le bouton change selon celui qui est pressé. Graphiquement, ce ne sont pas de « vrais » onglets car le widget n'existe pas dans la librairie Tk standard (et on n'a pas voulu ajouter un pré-requis supplémentaire !).

ETUDE est un onglet dans lequel on renseigne la liste des fichiers nécessaires pour lancer une étude. REX est un simple bouton qui affiche une liste de choix. Les autres boutons sont maintenant désactivés.

2.2.1 Onglet ETUDE

Pour accéder plus facilement aux fichiers, parcourir les arborescences plus rapidement (les fichiers étant souvent regroupés dans des répertoires proches), ou encore simplifier l'affichage des noms de fichiers, on peut définir un **chemin de base**.



Figure 2.2.1-1: Chemin de base

On le définit en cliquant sur le bouton Parcourir  et on choisit le répertoire qui sera le répertoire par défaut.

Chaque onglet contient une liste de fichiers (deux listes pour SURCHARGE).

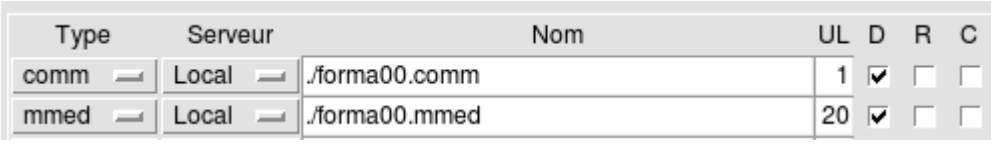


Figure 2.2.1-2: Liste de fichiers et répertoires

On définit un fichier ou répertoire par ligne. On trouve de la gauche vers la droite :

- Type : type du fichier ou du répertoire ;
- Serveur : nom du serveur sur lequel se trouve le fichier/répertoire ;
- Nom : chemin d'accès au fichier : en absolu s'il commence par '/', en relatif par rapport au chemin de base dans le cas contraire ;
- UL : numéro d'unité logique associé à ce fichier ;
- D, R, C : cases à cocher pour indiquer si le fichier/répertoire est respectivement en donnée, résultat, compressé (avec gzip).

Lorsque l'on change le type d'un fichier/répertoire, des valeurs par défaut sont positionnées pour les indicateurs D, R, C et le numéro d'unité logique.

Remarque sur les types :

> *Code_Aster manipule les fichiers via les procédures fortran qui utilisent des numéros d'unité logique (fichier `fort.19` par exemple) ; il affecte donc des numéros d'unité*

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

logique par défaut pour simplifier la vie de l'utilisateur. Le « type » permet ainsi d'accéder facilement aux numéros utilisés par défaut ; il permet aussi à astk de vérifier que l'utilisateur fournit des données cohérentes (par exemple que l'on fournit bien un répertoire pour tel type.

Pour manipuler la liste, on dispose de six boutons :

-  : ajoute une nouvelle entrée vierge à la fin de la liste ;
-  : ajoute un fichier/répertoire à la fin de la liste en parcourant l'arborescence de fichiers ;
-  : supprime la ligne actuellement sélectionnée dans la liste. Un clic-droit permet de supprimer toutes les lignes ;
-  : ouvre le fichier/répertoire actuellement sélectionné dans la liste avec l'éditeur du serveur où se trouve le fichier (cf. [§ 4.1]). S'il s'agit d'un répertoire, tous les fichiers du répertoire sont ouverts avec l'éditeur (attention aux répertoires contenant beaucoup de fichiers ou des fichiers binaires non éditables !) ;
-  : déplace une ligne vers le haut ;
-  : déplace une ligne vers le bas.

Arguments :

Cette zone de texte permet de transmettre des arguments à l'exécutable *Aster*. Voir aussi [§2.1.4].

On peut par exemple lancer le calcul en interactif et demander à ce que l'invite de commandes soit proposée à la fin des instructions pour continuer l'exécution de commandes manuellement avec l'option « `--interact` ».

Menu contextuel :

En cliquant avec le bouton droit sur une entrée de la liste, on accède à un menu contextuel :

- Ouvrir : lance l'outil associé à ce type de fichier (déterminé par le champ « Type », et non l'extension du fichier), si aucun outil n'est associé à ce type, on édite le fichier ;
- Editer : édite le fichier (ou tous les fichiers du répertoire) de la même manière que le bouton  ;
- Ouvrir avec… : on peut choisir d'ouvrir le fichier sélectionné avec un des outils disponibles (le résultat peut être étonnant si l'outil ne connaît pas ce type de fichier !) ;
- Valeurs par défaut : l'interface détermine un nom de fichier par défaut en fonction du « Type » choisi à partir du nom du profil (fichier `.astk`), aucune valeur n'est proposée si le profil n'a pas encore été enregistré (s'utilise en général sur une ligne vierge que l'on vient d'insérer, les indicateurs D/R/C ne sont pas affectés par cette opération) ;
- Terminal : permet d'ouvrir une fenêtre terminale `xterm`
- Propriétés : affiche les permissions, la taille, la date et l'heure du fichier (commande `ls -la`).

2.2.1.1 Liste des types pour ETUDE

comm :	fichiers de commande *Aster* (y compris les fichiers de poursuite)
mail :	fichier maillage au format *Aster*
erre :	fichier d'erreur (fort.9 d'Aster)
mess :	fichier des messages de l'exécution
resu :	fichier de résultat (impression des tests, impression au format *Aster*)
base :	répertoire contenant la base du calcul
bhdf	répertoire contenant la base du calcul au format HDF
cast :	fichier résultat au format CASTEM
mast :	gardé pour raison de compatibilité
mgib :	maillage au format Gibi

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

mmed : maillage au format MED
msh : maillage au format Gmsh
msup : maillage au format IDEAS
datg : fichier de commande Gibi
pos : fichier résultat au format Gmsh
ensi : répertoire résultat au format Ensight
dat : fichier résultat contenant des courbes au format XMGRACE
ps : fichier au format postscript
agraf : fichier résultat contenant les données pour Agraf (les anciennes versions d'Aster écrivait les directives et les données dans un même fichier qu'il fallait découper avec la commande `post_agraf` sur la machine de référence)
digr : fichier résultat contenant les directives pour Agraf
rmed : fichier résultat au format MED
unv : fichier résultat au format « UNiVersel » (IDEAS)
distr : fichier des valeurs utilisées pour une étude paramétrique
hostfile : fichier décrivant les ressources machines à utiliser (étude paramétrique)
nom : pour récupérer à partir de son nom, un fichier présent dans le répertoire temporaire /tmp
para : fichier de paramètres (retranscription des paramètres du calcul pour les tests)
repe : répertoire en données et/ou résultats (permet de transmettre/récupérer le contenu complet d'un répertoire ; comme on n'affecte pas de numéros d'unité logique *Aster* doit accéder aux fichiers par leurs noms).
répertoire des résultats lors d'une étude paramétrique
libr : fichier ou répertoire au choix de l'utilisateur
btc : script de lancement généré par le service (on peut ainsi le récupérer, le modifier…).

Lors d'un astout sur la machine de référence, `resu_test` doit être sur celle-ci.
Lors d'une étude paramétrique, le répertoire des résultats (type `repe`) doit être sur la machine d'exécution.

2.2.2 Bouton REX

Ce bouton permet :

- D'**Emettre une fiche sans joindre le profil** : Cette fonction permet à l'utilisateur d'émettre une fiche de retour d'expérience pour signaler une anomalie dans *Code_Aster* (AL : anomalie logiciel), demander une évolution de *Code_Aster* (EL : évolution logiciel), dans un des outils associés (Metis, Homard, Eficas, astk, bsf…) (AO : anomalie outil/EO : évolution outil), une modification de la documentation (ED : évolution documentation), ou une demande d'expertise en modélisation (AOM : aide à l'optimisation de la modélisation). Les informations relatives à l'émetteur de la fiche sont accessibles par le menu *Configuration/Interface*. Les fichiers contenus dans le profil courant ne sont pas joints à la fiche, ce qui peut rendre très difficile le traitement de la fiche. Il est conseillé d'utiliser "émettre une fiche et y associer les fichiers". La version de *Code_Aster* indiquée est celle sélectionnée dans l'interface (*Paramètres du calcul*).
- D'**Emettre une fiche et y associer les fichiers** : Cette fonction permet d'émettre le même type de fiche que la précédente, cette fois-ci les données incluses dans le profil sont jointes à la fiche. Pour une anomalie, les fichiers pour reproduire l'erreur devraient toujours être fournis. Ce qui n'empêche pas l'utilisateur d'essayer d'isoler autant que possible le problème rencontré, notamment d'essayer de joindre une étude sur un modèle qui nécessite peu de mémoire et de temps de calcul !
- De **Consulter les fiches** : permet d'accéder à l'outil de gestion de retour d'expérience (sur la machine de référence).

Remarque

Les fonctionnalités REX ne sont disponibles que sur le serveur de référence.

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

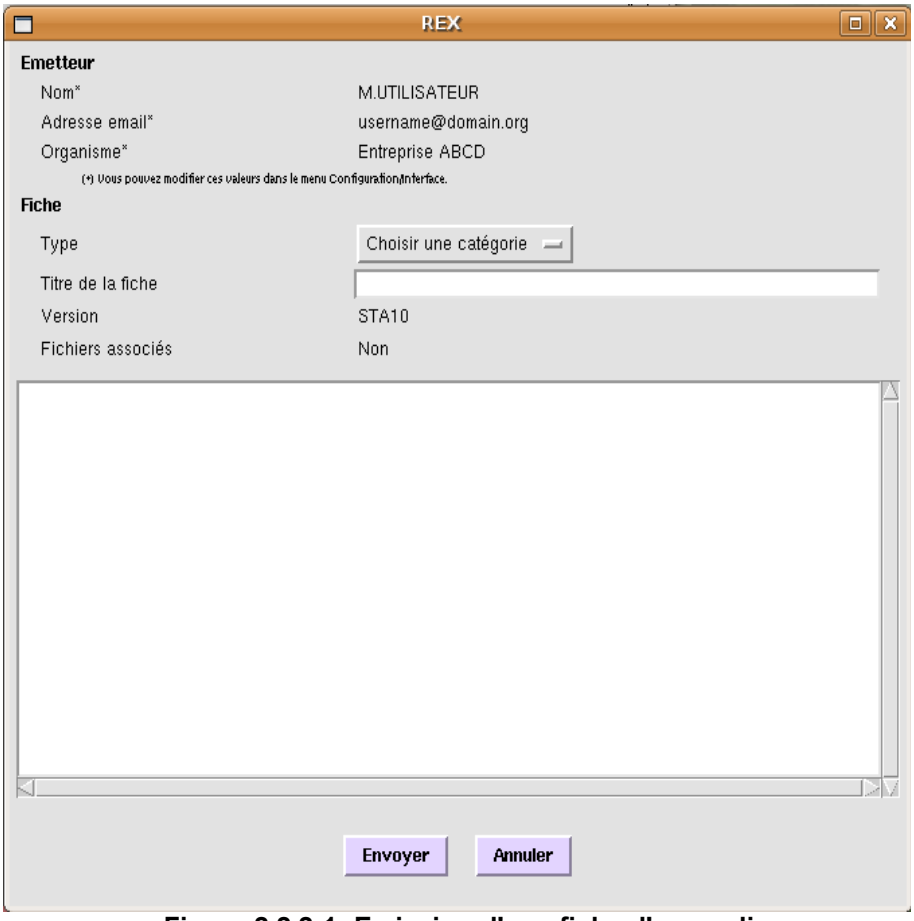


Figure 2.2.2-1: Emission d'une fiche d'anomalie

2.3 Paramètres de soumission

Les paramètres du calcul sont fournis dans la partie droite de la fenêtre principale.

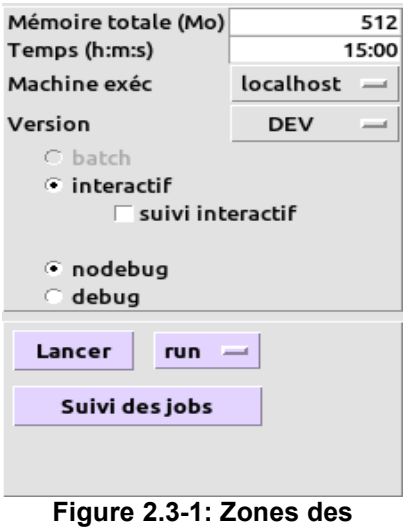


Figure 2.3-1: Zones des paramètres du calcul

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

On définit la quantité de :
- `Mémoire totale` utilisée pour le job (en mégaoctets).
- le temps maximum du calcul (en secondes, minutes:secondes ou heure:minutes:secondes).

On choisit sur quelle machine le calcul est exécuté, la version de *Code_Aster* utilisée, si le calcul est soumis en batch ou en interactif. L'option « suivi interactif » permet d'exécuter le calcul tout en suivant son exécution dans une fenêtre de type terminal. Si le calcul est lancé sur un serveur distant, ce terminal est exécuté à distance. C'est pour cela que l'interface est alors bloquée tant que l'exécution du calcul n'est pas terminée pour conserver la connexion ssh ouverte.
debug/nodebug : pour une étude sans surcharge, on précise quel exécutable on souhaite utiliser (sous réserve que les deux soient disponibles).

Le bouton « Lancer » exécute les actions en fonction des onglets cochés.
Le bouton « Suivi des jobs » ouvre la fenêtre décrite après.

Mode de fonctionnement (pour une ETUDE) :
Lors du lancement d'une étude, un bouton d'option est disponible à côté du bouton « Lancer ». Trois modes de lancement sont disponibles :
- « run » : exécute l'étude (fonctionnement classique),
- « dbg » : lance l'étude en utilisant le debugger,
- « pre » : prépare le répertoire de travail sans exécuter l'étude.

Quand on sélectionne « dbg » ou « pre », le mode « debug » est choisi par défaut.

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

2.4 Barre d'état

La zone de texte située tout en bas de la fenêtre principale fournit de l'aide lorsque l'on navigue dans les menus, ou bien quand le pointeur passe au dessus des boutons de l'interface.
Lors du lancement d'un calcul, les opérations en cours sont affichées ici.

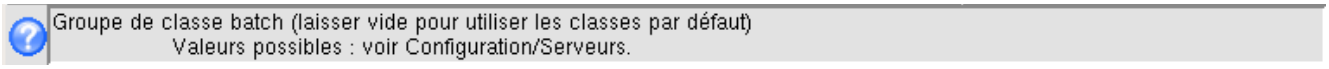


Figure 2.4-1: Barre d'état

2.5 Suivi des jobs

Cette fenêtre fournit des informations sur l'état des calculs lancés, qui apparaissent sous forme d'une liste. En cliquant avec le bouton droit sur un calcul, on a les choix suivants :

- **Editer** se décompose en « Fichier output », « Fichier error » et « Fichiers export » qui donne accès aux messages envoyés par le job sur la sortie standard (`stdout`) et la sortie d'erreur (`stderr`). On peut aussi accéder au fichier .export utilisé lors de l'exécution. Un double-clic sur un job de la liste affiche également le fichier output. Ces fichiers sont d'abord recopiés en local dans un répertoire temporaire puis ouvert avec l'éditeur local.
- **Télécharger les fichiers résultats** force la recopie des fichiers résultats. Cette action est faite automatiquement quand le calcul passe dans l'état ENDED. Si cette étape échouait (problème réseau), cela permettrait de l'exécuter à nouveau.
- **Actualiser** interroge les serveurs sur lesquels des calculs sont en cours. La case à cocher permet de réactualiser automatiquement à une fréquence définie dans *Configuration/Interface*.
- **Actualiser tous** rafraichit l'état de tous les calculs non terminés.
- **Supprimer** efface les jobs sélectionnés de la liste, les fichiers liés à ce job dans le flasheur et interrompt le calcul si celui-ci n'est pas terminé (en envoyant le signal SIGKILL).
- **Arrêter proprement** envoie le signal SIGUSR1 au calcul. *Code_Aster* intercepte ce signal pour interrompre le calcul, puis ferme proprement la base qui sera rapatriée (si une base est en résultat dans le profil).
- **Purger flasheur** parcourt tous les serveurs et y supprime les fichiers du répertoire flasheur non accessibles depuis le suivi des jobs.
- **Rechercher** permet de consulter les dernières lignes du fichier message d'un job *en cours d'exécution* (il ne fait rien sur un job terminé). On peut utiliser la zone de texte « Filtre » pour n'afficher que les lignes contenant la chaîne de caractères indiquée (sous forme d'expression régulière).

La zone de texte permet de suivre l'avancement des requêtes exécutées sur les serveurs distants.

Chaque ligne correspond à un job, on trouve 13 colonnes :

- Le numéro du job (en batch), numéro du processus en interactif
- Le nom du job (nom du profil pour une étude, une surcharge, ou nom de la fonction AGLA)
- Date de soumission
- Heure de soumission
- Etat du job (PEND, RUN, SUSPENDED, ENDED)
- Diagnostic du job (OK, NOOK, <A>_ALARM, <F>_ERROR, <F>_ABNORMAL…)
- Nom de la queue en batch ou « interactif »
- Temps CPU de l'exécution *Aster*
- Login sur le serveur de calcul utilisé
- Adresse du serveur de calcul utilisé
- Machine de calcul (nom du nœud pour un cluster)
- Version d'astk
- Indicateur batch/interactif

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

3 Boite de sélection de fichiers : bsf

bsf est un outil livré avec astk qui peut être lancé seul. Il s'agit d'un explorateur de fichiers qui permet de naviguer sur la machine locale, comme un explorateur de fichiers classique, et aussi sur les différents serveurs distants configurés.

bsf utilise la configuration des serveurs de astk, notamment les champs nécessaires à la connexion (adresse IP, login) et les commandes pour ouvrir un terminal ou un éditeur.
La lecture de la configuration n'est faite qu'au démarrage de bsf, si on modifie la configuration dans astk, il faut donc fermer la bsf puis l'ouvrir de nouveau.

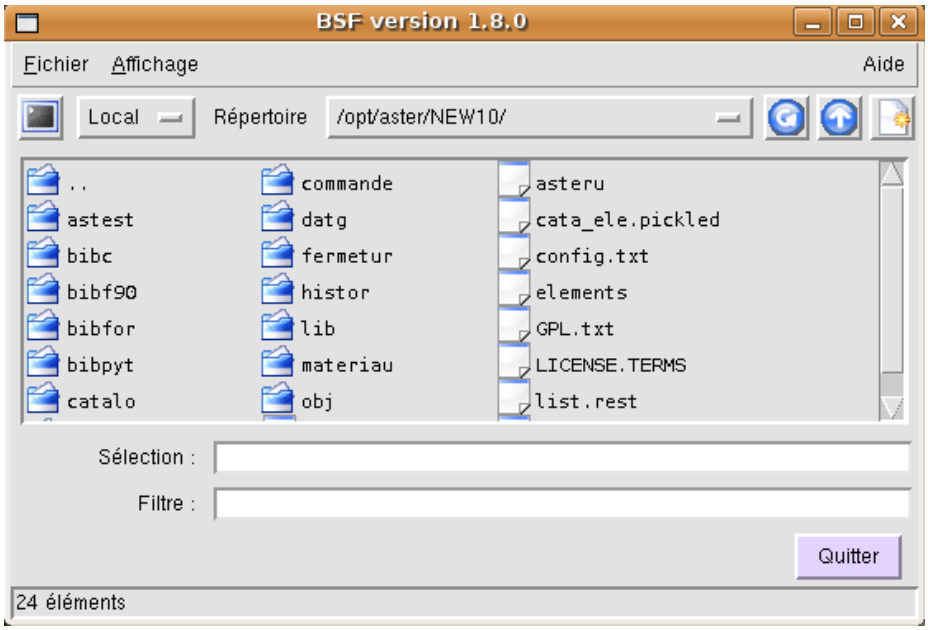


Figure 3-1: bsf

Remarque :

bsf ne traite pas les noms de fichiers et répertoires contenant les espaces (il en résulte un décalage dans les noms et les types des objets suivants).
bsf n'a plus beaucoup d'intérêt, les gestionnaires de fichiers standards sous Gnome ou KDE la remplacent avantageusement.

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

3.1 Navigation

On trouve deux listes déroulantes dans la fenêtre de la bsf.
La première permet de passer d'un serveur à un autre, la deuxième mémorise la liste des douze derniers répertoires où l'utilisateur a effectué une action (édition, copie…). La première entrée de cette liste est '----- Direct -----', qui permet d'aller directement dans un répertoire dont on saisit le nom.
D'une manière générale, lorsque le répertoire demandé n'existe pas, on retourne dans le HOME défini pour le serveur courant.
On peut configurer cette liste et fixer certains répertoires (pour qu'ils restent présents dans la liste) en cliquant avec le bouton droit sur celle-ci (cf. Figure 3.1-a).

-  : rafraîchit le contenu du répertoire courant ;
-  : remonte au répertoire parent ;
-  : propose de créer un nouveau répertoire dans le répertoire courant (et se place dans ce nouveau répertoire) ;
-  : ouvre un terminal sur le serveur actuel.

La barre d'état donne des indications sur la signification de ces boutons lorsque le pointeur de la souris passe au dessus.

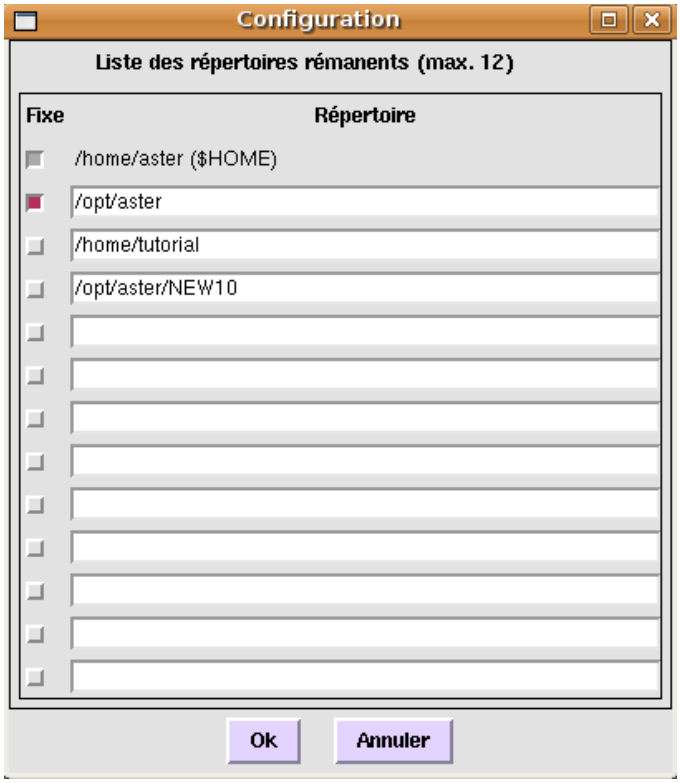


Figure 3.1-1: Fenêtre de configuration des répertoires rémanents

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

3.2 Menu Fichier et menu contextuel

On retrouve certaines entrées semblables au menu contextuel des fichiers dans les onglets (voir [§2.2.1]) : Ouvrir, Editer, Ouvrir avec…, Propriétés.

- **Copier** : permet de copier un ou plusieurs fichiers/répertoires (raccourci par CTRL+C).
- **Couper** : idem Copier, sauf que les fichiers originaux seront supprimés (raccourci par CTRL-X).
- **Coller** : réalise effectivement la copie ou le déplacement des fichiers (raccourci par CTRL+V).
- **Supprimer** : efface les fichiers/répertoires sélectionnés.
- **Renommer** : donne un nouveau nom à un fichier
- **Nouveau fichier/Nouveau répertoire** : permettent respectivement de créer un fichier ou un répertoire dans le répertoire courant.
- **Exécuter une commande** : donne la possibilité d'exécuter une commande dans le répertoire courant, la sélection courante est proposée sur la ligne de commande, il faut choisir un des shells disponibles sur le serveur.
- **Terminal** : ouvre un terminal sur le serveur actuel.

3.3 Menu Affichage

On peut choisir d'afficher ou non les fichiers dits cachés (commençant par '.') en cochant la case prévue.
bsf affiche les répertoires puis les fichiers, on peut les trier par ordre alphabétique ou en fonction de la date dans l'ordre normal ou inverse en modifiant les options du menu.

Les préférences d'affichage sont conservées si l'on sélectionne **Enregistrer les préférences**.

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

4 Configuration

La première fois que l'utilisateur lance l'interface, il dispose d'une configuration par défaut qui a été définie lors de l'installation. La configuration est ensuite stockée dans le répertoire `$HOME/.astkrc`.
Si l'utilisateur veut revenir à la configuration d'origine, il doit effacer ce répertoire et relancer astk.

Remarque n°1

> *A chaque fois que l'on définit une commande à exécuter (terminal, éditeur…), il est conseillé d'indiquer le chemin absolu (depuis la racine) pour éviter que la commande ne soit pas trouvée si la variable `$PATH` est incorrecte.*

Remarque n°2

> *Lorsque astk exécute une commande qui tente d'afficher des fenêtres :si la commande est exécutée en local, pas de problème. Si elle est exécutée sur une machine distante, elle l'est en utilisant `ssh -X` , donc l'affichage doit bien se passer si le serveur ssh l'autorise.*

La partie as_run contient des outils simplifiant certaines tâches des développeurs comme la consultation du code source ou la mise à jour d'une version locale de développement.
Le fichier `$HOME/.astkrc/prefs` contient donc deux informations nécessaires pour contacter le serveur de référence : le nom complet de ce serveur et le login de l'utilisateur. Ce dernier doit être défini dans ce fichier pour éviter l'alarme suivante :

```
<A>_ALARM          remote connection may fail :
devel_server_user not defined in /home/xxxxxx/.astkrc/prefs
```


Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

4.1 Serveurs

On accède à la fenêtre de configuration par le menu *Configuration/Serveurs* (voir Figure 2.1.2-2).

Le premier bouton permet de passer d'un serveur à un autre et d'ajouter un « Nouveau serveur ».
Les champs sont :

- Nom complet ou adresse IP : il s'agit du nom du serveur sur le réseau ; on peut indiquer son nom complet avec le nom de domaine (par exemple : linux.labo.univ.fr) ou son adresse IP (par exemple : 156.98.254.36).
- La case indique que astk ne doit pas utiliser rsh ou ssh pour contacter cette machine.
- Etat du serveur : on peut mettre sur « off » un serveur temporairement inaccessible.
- Login : identifiant avec lequel on se connecte au serveur.
- Répertoire HOME : répertoire par défaut lorsque l'on navigue sur cette machine.
- Répertoire des services : répertoire où sont installés les services sur ce serveur (indiquer le chemin d'installation, par exemple : `/opt/aster`), laisser vide pour un serveur de fichiers.
- Mode de téléchargement de la configuration : aucun (pour un serveur de fichiers), manuel (il faut cliquer sur le bouton « Télécharger maintenant » pour récupérer la configuration Aster du serveur), automatique (astk interroge le serveur au démarrage tous les 30 jours).
- Dernier téléchargement : date de la dernière mise à jour des informations de configuration.
- Terminal : commande pour ouvrir un terminal sur le serveur. Ceci permet d'ouvrir une fenêtre de commandes sur le serveur quand on utilise la bsf.
- Editeur : éditeur texte (par exemple, `nedit`). La procédure d'installation choisit un éditeur parmi (et dans cet ordre) : `nedit`, `gedit`, `xemacs`, `emacs`, `xedit`, `vi`.

Les valeurs suivantes sont retournées par `as_run --info` (rien pour un serveur de fichiers) et dépendent donc de la configuration de la partie as_run :
- Type de plate-forme.
- Versions disponibles.
- Machines accessibles : liste des nœuds de calcul accessibles depuis ce serveur.
- Batch/interactif : précise si le serveur accepte le lancement en batch, en interactif et fournit les limites en mémoire, temps CPU, nombre de processeurs fixés sur le serveur.

Les logiciels de gestion de batch supportés sont LSF, Sun Grid Engine et PBS.

4.2 Configurations

4.2.1 Préférences Générales

On accède à la fenêtre de configuration des préférences générales par le menu *Configuration/ Préférences/générales* (voir Figure 2.1.2-3).

Cette fenêtre permet de renseigner les informations personnelles de l'utilisateur, de choisir la langue utilisée par l'interface.

Ensuite, on trouve la version qui sera sélectionnée par défaut, les commandes pour accéder à un terminal et un éditeur (comme pour les serveurs).
Nombre de profils dans le menu Fichier permet de conserver le nom des N derniers profils ouverts de manière à les rappeler rapidement.

Niveau de message indique le niveau de détails des messages écrits dans la fenêtre des messages du menu *Aide*. Niveau=0 : seuls les messages `<INFO>` et `<ERREUR>` sont écrits ; les niveaux supérieurs permettent de déboguer le comportement de l'interface. Le niveau 1 est conseillé, il permet de voir les messages d'erreur pouvant apparaître lors des problèmes de communication avec les serveurs distants.

Pour le suivi des jobs, on peut choisir le nombre de lignes affichées lorsque l'on visualise le fichier output en cours de job (bouton **Rechercher**), et la fréquence d'actualisation de la liste.

4.2.2 Préférences Réseau

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

On accède à la fenêtre de configuration des préférences concernant le réseau par le menu *Configuration/ Préférences/Réseau* (voir Figure 2.1.2-4).

Cette fenêtre permet de renseigner les paramètres réseau. On doit ensuite préciser le nom de domaine réseau de la machine. Par exemple, `domain.org` si le nom complet de la machine est `mach00.domain.org`. Si le nom de domaine est laissé vide, un message d'alarme est affiché au démarrage car les noms de machine renseignés sans nom de domaine ne seront pas complétés. Cela peut poser des problèmes, mais c'est aussi parfois nécessaire de laisser le nom de domaine à vide dans certaines configurations.
Si le mode DHCP/VPN est actif, c'est dans ce champ que l'on peut indiquer l'adresse IP de la machine locale. En cliquant sur OK, l'interface propose les adresses IP des interfaces réseaux détectées sur la machine.

4.3 Outils

Voir Figure 2.1.3-1.

On sélectionne l'outil à configurer avec la liste déroulante, ou bien on ajoute un nouvel outil.
Des outils standards sont pré-définis (les minuscules/majuscules sont prises en compte dans les noms d'outils). *Les outils sont nécessairement exécutés sur la machine « Local »* (où est lancée l'interface).

En général, les outils sont lancés soit sur un fichier dans l'onglet ETUDE, soit sur un fichier lorsque l'on parcourt le système de fichiers avec la bsf.

On définit simplement la ligne de commande nécessaire au lancement d'un outil (chemin absolu conseillé), on peut placer les codes @F, @R, @f, @D dans la ligne de commande (voir [§ 2.1.3]) pour passer correctement un fichier à l'outil. On peut mettre ces codes entre parenthèses pour pouvoir lancer l'outil seul, sans fichier en argument.

Des types de fichiers peuvent être associés à l'outil. Le type pris en compte pour lancer l'outil est l'extension lorsque l'on parcourt les fichiers avec la bsf, le type de la liste déroulante quand il s'agit d'un onglet.

On peut choisir si l'outil est utilisable sur un fichier distant. Dans ce cas, astk se charge de ramener le fichier en question sur la machine locale dans un répertoire temporaire, de lancer l'outil, puis de redéposer le fichier sur le serveur distant (même s'il n'a pas été modifié par l'outil).

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

5 Comment faire une étude ?

Dans ce paragraphe, on décrit étape par étape comment utiliser astk pour réaliser une étude.

L'étude consiste à calculer la réponse d'un cylindre mince sous pression hydrostatique. On dispose des éléments suivants :

- Le fichier de commande *Aster* : `forma00.comm`
- La description de la géométrie réalisée avec Salomé : `forma00.datg`
- Le maillage du réservoir construit par Salomé : `forma00.mmed`

On produit les résultats suivants :

- Un fichier résultat au format med (champs de déplacements, ...) : `forma00.resu.med`
- Les fichiers classiques de message et résultat Aster : `forma00.mess` et `forma00.resu`

Dans l'exemple, on place tous les fichiers dans le répertoire `/home/tutorial`.
Les fichiers de cet exemple sont disponibles dans le répertoire `astest` de votre version de *Code_Aster*.

Remarque :

> *Dans le cas d'une étude avec plusieurs fichiers de commandes, tous les fichiers doivent être de type « comm », associés à l'unité logique 1 et c'est l'ordre d'apparition dans le profil qui détermine l'ordre d'exécution.*

5.1 Création du profil

On lance l'interface qui s'ouvre sur un profil vierge, ou bien si astk est déjà lancé, on choisit *Fichier/Nouveau* dans le menu pour créer un nouveau profil vide.
On se place dans l'onglet ETUDE.

5.2 Sélection des fichiers

5.2.1 Définition d'un chemin de base

Dans l'onglet ETUDE, on choisit un chemin de base pour simplifier l'accès aux fichiers.
On clique sur l'icône , on choisit le répertoire `/home/tutorial`.

5.2.2 Ajout de fichiers existants

On ajoute le fichier de commandes en cliquant sur , la sélection de fichier s'ouvre directement dans le chemin de base que l'on vient de définir. Il ne reste qu'à sélectionner le fichier `forma00.comm` (double-clic ou simple clic + ok), et le fichier apparaît dans la liste. Notons qu'astk identifie le type de ce fichier à partir de son extension « comm », le numéro d'unité logique est positionné à 1, la case « D » (donnée) est cochée.

On fait de même pour le fichier de maillage au format med (`forma00.mmed`). astk reconnaît l'extension « mmed », le numéro d'unité logique est positionné à 20, la case « D » est cochée.

On peut également ajouter le fichier `forma00.datg`. On décoche la case « D », il ne sera pas utilisé dans l'étude mais on peut visualiser ce maillage en l'ouvrant avec Salomé (voir §5.5).

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

5.2.3 Ajout de fichiers…

Sauf si une exécution a déjà eu lieu, les fichiers résultats n'existent pas encore, on ne peut donc pas les ajouter en parcourant l'arborescence.

5.2.3.1 …en insérant une ligne vide

On clique sur , une ligne est ajoutée dans la liste. On choisit le type « mess » dans la liste (ce qui a pour effet de positionner le numéro d'unité logique à 6). On indique le nom `/home/tutorial/forma00.mess` ou `forma00.mess` ou `./forma00.mess` (puisque l'on peut indiquer le nom du fichier en relatif par rapport au chemin de base). Le fichier est produit par l'exécution, on coche donc la case « R » (résultat) et on décoche « D ».

5.2.3.2 …avec « Valeur par défaut »

On pourrait continuer ainsi pour ajouter les autres fichiers, mais on va utiliser la fonction « Valeur par défaut » pour les fichiers suivants. Cette fonction utilise le nom du profil astk pour construire les valeurs par défaut (voir [*§ 2.2.1]/Menu contextuel*), on va donc enregistrer le profil.
On choisit *Enregistrer sous…* dans le menu *Fichier*, on va avec le navigateur dans le répertoire `/home/tutorial`, et dans la ligne *Sélection*, on tape `forma00` (l'extension `.astk` est automatiquement ajoutée).
Notons que le titre de la fenêtre principale d'astk donne le nom du profil courant. Le titre est maintenant : `ASTK version 1.13.6 - forma00.astk - /home/tutorial`

On insère une ligne vide en cliquant sur , on choisit le type de fichier « rmed », puis on clique avec le bouton droit dans la case du nom de fichier et on choisit « Valeur par défaut » : astk construit un nom de fichier à partir du chemin de base (voir [§5.2.1]), du nom de profil (en retirant l'extension) et du type « rmed », soit `/home/tutorial/forma00.resu.med`. On voit ainsi : `./forma00.resu.med` (nom relatif au chemin de base).
La case « R » a été cochée, et le numéro d'unité logique fixé à 80. Dans le fichier de commande, on a indiqué :

```
IMPR_RESU( FORMAT='MED', UNITE= 80 , ...)
```

Si on a besoin d'ajouter un nouveau fichier en sortie `forma00.rmed`. On procède de la même manière : sélectionner le type « libr », modifier l'unité logique en 81 et le nom du fichier. La case « R » a été cochée, et le numéro d'unité logique fixé à 80. Dans le fichier de commande, on a indiqué :

```
IMPR_RESU( FORMAT='MED', UNITE= 8 1 , …)
```

On modifie donc le numéro d'unité logique en conséquence, il suffit de cliquer sur l'ancienne valeur, de l'effacer et de taper 81. Seuls deux chiffres sont affichés dans cette case, pour éviter les erreurs, astk vérifie que les numéros d'unité logique sont compris entre 1 et 99.

De même, on ajoute un fichier de type « mess » et un de type « resu » de cette manière (laisser les numéros d'unité logique par défaut).

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

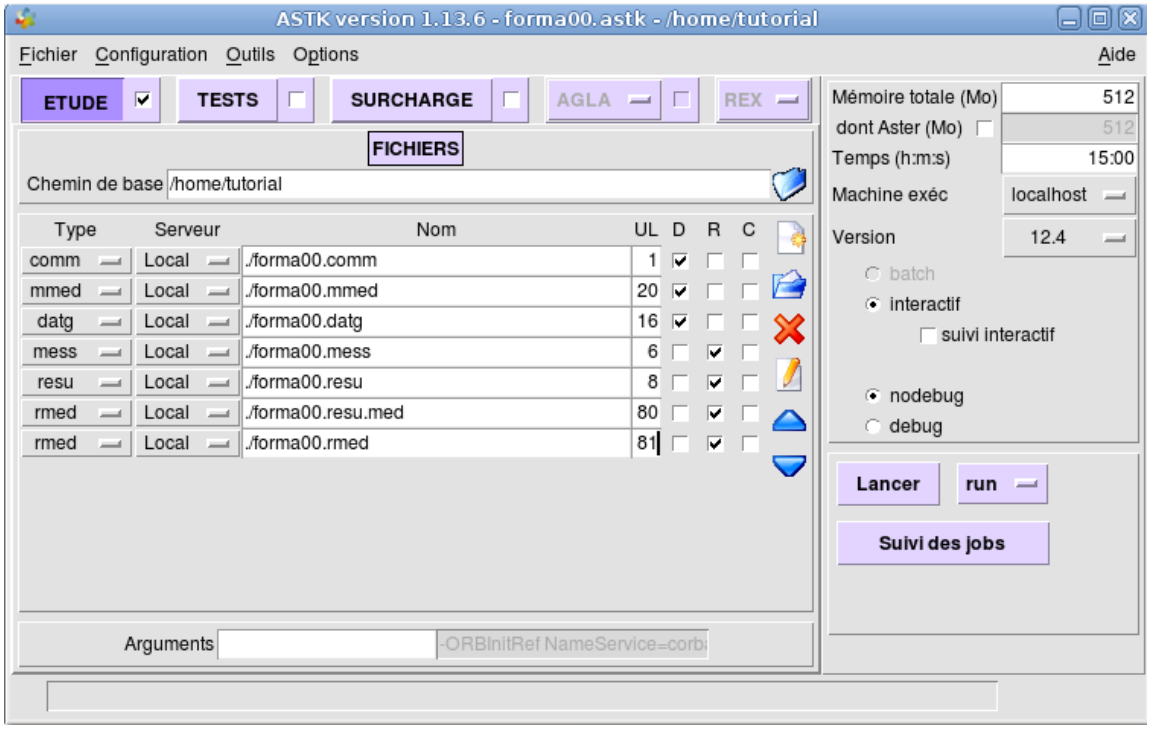


Figure 5.2.3.2-1: Fenêtre du profil d'étude

5.2.4 Supprimer un fichier

Pour supprimer une ligne de la liste des fichiers, il suffit de la sélectionner en cliquant dans la zone où l'on indique le nom du fichier et de cliquer sur l'icône .

Remarque :

> *Seule la référence à ce fichier dans le profil astk est oubliée, le fichier lui-même n'est pas effacé !*

5.3 Lancement du calcul

Les fichiers données et résultats sont sélectionnés, on ajuste les paramètres du calcul (voir [§ 2.3]), et on clique sur le bouton « Lancer ».
On prend soin de cocher la case qui se trouve juste à côté de ETUDE pour signaler que l'on souhaite utiliser le contenu de cet onglet… sinon l'interface nous répond « Rien à lancer ! ».
Si le profil n'a pas encore été enregistré, l'interface demande de choisir un endroit et un nom pour ce profil (voir [§ 5.2.3.2]).

astk appelle as_run pour exécuter le calcul, et transmet au Suivi des jobs (asjob) le numéro du job (numéro du processus en interactif) et d'autres informations qui vont permettre de suivre l'avancement du calcul. L'état initial du calcul est `PEND` (en attente), quand le calcul commence, il devient `RUN`, puis `ENDED` quand il est terminé (d'autres états sont possibles en batch). Le bouton « Actualiser » appelle le service qui rafraîchit l'état des calculs en cours.

Quand le calcul est terminé, on peut consulter l'output du job en double-cliquant sur le job, ou par *Editer/Fichier output*.

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

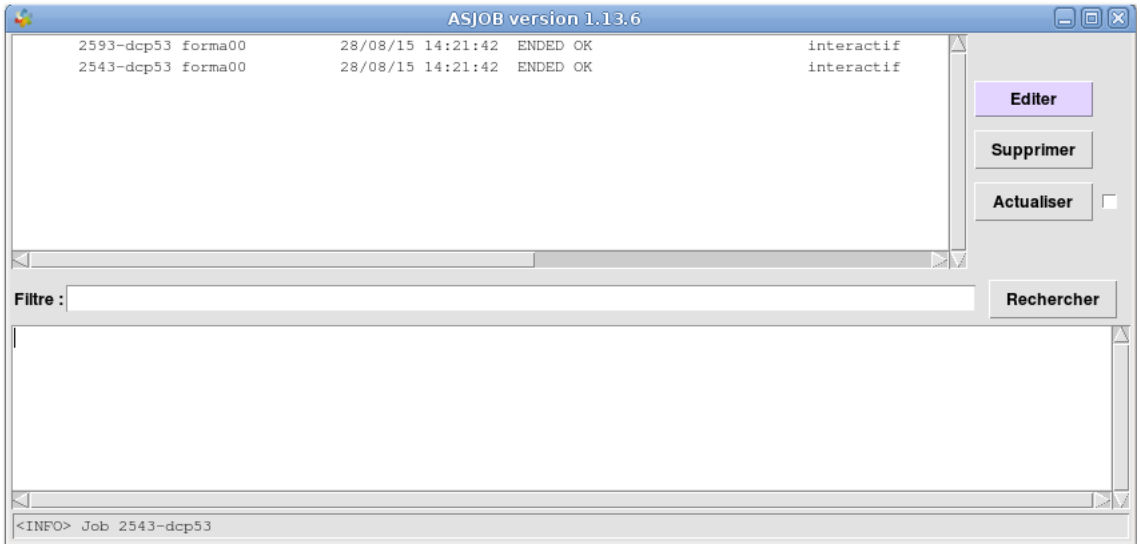


Figure 5.3-1: Fenêtre de suivi des jobs

5.4 Consultation des résultats

On peut consulter les fichiers résultats simplement en double-cliquant sur leur nom, ce qui ouvre un éditeur de texte pour les fichiers « mess » et « resu » . Si on possède un résultat au format grace, « dat », cela a pour effet d'ouvrir directement ce fichier dans le traceur de courbe grace. On visualise ainsi les courbes (déplacement fonction du temps, contrainte-déformation...) (sous réserve que grace ait été installé, et que « dat » soit dans les types de fichiers associés à *grace*).

NB :

- Le répertoire devant accueillir un fichier résultat n'existe pas, il est automatiquement créé si les permissions sont suffisantes.
- Si la copie de fichiers résultats échoue (problème de permissions, de quota, de connexion distante...), ils sont copiés dans un répertoire temporaire sur la machine d'exécution. Une alarme `<A>_COPY_RESULTS` indique le chemin où il faut aller chercher les résultats.

5.5 Utilisation des outils

On peut aussi utiliser astk et le fait que l'on puisse y définir librement des outils pour rassembler dans un profil tous les fichiers nécessaires à une étude même si ceux-ci ne sont pas directement utilisés par *Code_Aster*.

Exemple : Tracer une courbe avec grace

On indique le nom `/home/tutorial/forma00.dat`. Le fichier est produit par l'exécution, on coche donc la case « R » (résultat) et on décoche « D » et le numéro d'unité logique fixé à 21 . Si dans le fichier de commande on a indiqué :

```
IMPR_COURBE(FORMAT='XMGRACE',UNITE=21,COURBE=_F(…)
```

On peut ouvrir directement la courbe en faisant *Ouvrir avec…/grace* (clic droit sur le nom du fichier), modifier la géométrie ou les paramètres du maillage.

Bien évidemment, ceci n'est pas limité à grace ; on peut utiliser d'autres outils (mailleurs, outil de post-traitement, …) directement depuis astk et accéder ainsi à tous les fichiers d'une étude depuis un profil avec l'outil adéquat.

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

5.6 Fonctionnalités avancées

5.6.1 exectool

On choisit dans astk la version de *Code_Aster*, le mode debug ou nodebug, une éventuelle surcharge et ceci conduira à utiliser tel ou tel exécutable de *Code_Aster*.
On peut aller encore plus loin en précisant la manière exacte de lancer cet exécutable : c'est le rôle de l'option de lancement **exectool** (menu Options).

En temps ordinaire, *Code_Aster* est lancé avec une commande du type :
```
./aster.exe argument1 argument2 ...
```

Utiliser l'option exectool revient à lancer :
```
cmde_exec ./aster.exe argument1 argument2 ...
```

Dans le menu Options, on peut préciser directement `cmde_exec` ou bien un nom d'outil défini dans le fichier de configuration d'as_run : `[ASTER_ROOT]/etc/codeaster/asrun`

Exemple 1 :
Dans le menu Options, on saisit dans la case exectool : `time`
La commande de lancement sera donc `time aster.exe arguments...`
La commande `time` accepte exactement ce type d'argument (un exécutable et ses arguments), on aura donc le temps d'exécution du calcul. Cela n'a pas grand intérêt, *Code_Aster* donne déjà ce type d'information.

Exemple 2 :
Dans le fichier de configuration `[ASTER_ROOT]/etc/codeaster/asrun`, on définit (sur une seule ligne) :
```
memcheck : valgrind --tool=memcheck --error-limit=no --leak-check=full
--suppressions=/opt/aster/valgrind-python.supp
```

Il suffit ensuite d'indiquer dans le menu Options, exectool : `memcheck`
`memcheck` est défini dans le fichier de configuration (à gauche des « : »), donc c'est la commande complète `valgrind...` qui sera utilisée lors du lancement.

On peut définir autant d'outils que l'on souhaite à condition de ne pas entrer en conflit avec les autres paramètres définis dans ce fichier. Pour cette raison, la définition des « exectools » devrait évoluée à l'avenir.

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

6 Comment lancer une étude paramétrique

On entend par étude paramétrique une étude standard (définie dans l'onglet ETUDE) dans laquelle on souhaite faire varier un ou plusieurs paramètres.

L'étude est aussi générale que n'importe quelle étude définie dans astk.

Remarque

L'étude doit être valide avant d'être déclinée sur le jeu de paramètres. Elle doit donc tourner sans erreur. Il est aussi important d'optimiser l'étude avant de la décliner sur un grand nombre de valeurs des paramètres.

La définition et le lancement d'une étude paramétrique sont décrits dans [U2.08.07].

Le lancement est strictement identique au lancement de l'étude nominale seule. Seule l'option « distrib » doit être mise à **oui** dans le menu Options.
Les calculs sont lancés en parallèle : voir §7.1 concernant les options spécifiques.

7 Lancement de calculs en parallèle

7.1 Distribution de calculs

La gestion des calculs distribués est activée lors du lancement d'une étude paramétrique. En effet, chaque calcul est indépendant des autres. On peut ainsi les soumettre en parallèle pour réduire le temps de retour.

Chaque calcul unitaire est soumis avec les limites de temps et de mémoire renseignées dans l'interface. Par défaut, le job maître utilisera les mêmes limites. Sur les serveurs de calculs, les paramètres peuvent être fixés (par exemple, 200 heures sur Aster5) en utilisant un *plugin* (voir la documentation de développement dans `[ASTER_ROOT]/share/codeaster/asrun/doc/`).

7.1.1 Utilisation des ressources disponibles

On peut insérer dans le profil (onglet ETUDE), un fichier de type « hostfile ». On y définit la liste des machines disponibles et pour chacune le nombre de processeurs et la quantité de mémoire (en Mo) utilisables.

Exemple :

```
[compute01]
cpu=4
mem=8192

[compute02]
cpu=1
mem=1024
```

Cela signifie que jusqu'à 4 calculs pourront être soumis sur `compute01` (dans la mesure où ils ne demandent pas plus de 8192 Mo à eux 4) et 1 calcul sur `compute02` utilisant moins de 1024 Mo.
En batch, on peut soumettre beaucoup plus de calculs que de processeurs disponibles et laisser le gestionnaire de batch répartir les calculs sur un cluster par exemple. Dans ce cas, on peut fixer `cpu=50` pour laisser au maximum 50 calculs dans le gestionnaire de batch.

S'il n'y a pas de fichier « hostfile » dans le profil, on prend celui dont le nom est fixé dans le fichier de configuration `[ASTER_ROOT]/etc/codeaster/asrun` sous le label `interactif_distrib_hostfile` ou `batch_distrib_hostfile` selon le mode de lancement.

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

Si aucun fichier « hostfile » est spécifié, le nombre de processeurs (coeurs en fait) et la mémoire totale sont automatiquement déterminés.

Remarques

– *On peut facilement écrouler une machine en lançant trop de calculs vis à vis des ressources disponibles. Il est conseillé de se renseigner sur les possibilités d'utilisation de moyens de calculs partagés (classe batch dédiée par exemple).*
– *Les calculs parallèles comptent bien pour le nombre de processeurs qu'ils utilisent et non pas pour 1.*
– *Avant le lancement des calculs, la connexion aux noeuds de calcul est testée. La liste sera limitée aux machines ayant été contactées avec succès.*

7.1.2 Délai d'expiration

Lorsque le nombre de calculs à lancer est très supérieur au nombre de processeurs globalement disponibles (et c'est souvent le cas), des calculs sont en attente de soumission.
Si un calcul demande plus de mémoire qu'aucune machine ne peut en offrir, il resterait indéfiniment en attente.
Pour éviter cela, un délai d'expiration (timeout) est défini égal au temps du calcul maître, c'est-à-dire le temps choisi dans astk lors de la soumission globale.

Si aucun calcul n'a été soumis durant ce délai, le calcul est rejeté.

7.2 Activation du parallélisme de *Code_Aster*

Le parallélisme interne à *Code_Aster* est présent sous deux formes :

- Le parallélisme OpenMP fonctionne en mémoire partagée et est disponible dans les solveurs `MULT_FRONT` et `MUMPS`. Il faut bien sûr que la version de *Code_Aster* ait été compilée avec les options adéquates.
- Le parallélisme MPI (par envoi de message, Message Passing Interface) est disponible dans les solveurs `MUMPS` et `PETSC` et dans les calculs élémentaires. La compilation est beaucoup plus compliquée et n'est pas automatique lors de l'installation de *Code_Aster* (il faut choisir une implémentation MPI, compiler les pré-requis et notamment `MUMPS` en MPI, puis *Code_Aster*).

On choisit le nombre de processeurs utilisés en OpenMP et le nombre de processeurs utilisés en MPI (répartis sur un certain nombre de noeuds de calculs) dans le menu Options (voir §2.1.4).

7.3 Exécutions multiples

Il s'agit d'un mode de lancement bien particulier à destination des développeurs.
L'objectif est d'exécuter un profil (une étude ou une liste de cas-tests) sur plusieurs machines simultanément.

On l'active en cochant `multiple=oui` dans le menu `Options`. Puis, au moment de l'exécution une fenêtre s'ouvre afin de sélectionner (en cochant) les machines sur lesquelles l'étude ou les tests seront lancés.
Les résultats, y compris les fichiers de sortie *output* et *error* habituellement copiés dans le répertoire `flasheur`, sont copiés dans le répertoire `$HOME/MULTI` (`$HOME` étant en général égal à `/home/username`). On peut choisir de laisser les résultats sur chaque machine, ce qui est conseillé si les fichiers sont globalement volumineux, ou bien de rapatrier tous les fichiers sur la machine locale.

Bien évidemment, il y a quelques précautions à prendre pour que cela fonctionne : la version sélectionnée doit être disponible sur toutes les machines, les paramètres de calcul compatibles avec les ressources de chaque machine, etc.

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

8 Utilisation de as_run

Lorsque l'on utilise l'interface astk pour lancer des calculs, celle-ci en tant que client fait appel à des services proposés par un serveur qui peut se trouver sur la même machine ou une machine distante (dans le cas où le serveur est distant, il y a des échanges de fichiers et une commande shell à travers le réseau (protocole rsh ou ssh) que nous ne détaillons pas ici).

Les fonctionnalités de `as_run` peuvent être classées en plusieurs catégories :

- **pour l'utilisateur** : fonctionnalités qui peuvent être appelées par les **utilisateurs**, comme par exemple lancer *Code_Aster* « à la main »,
- **pour le développeur** : fonctionnalités utilisées par les développeurs pour visualiser un fichier, copier un fichier, construire une liste de tests, vérifier le catalogue de messages...
- **réservées aux clients** : fonctionnalités sans intérêt direct hors de astk (ou un autre client),
- **pour les tâches d'administration** : lien vers l'outil de suivi des anomalies, la base de données d'études...

Les différentes fonctionnalités de `as_run` sont données en tapant : `as_run --help`.

Dans les fichiers de configuration de `as_run` et dans les fichiers `.export`, on peut utiliser des variables d'environnement (uniquement pour les noms de fichiers).
Par exemple, dans `etc/codeaster/asrun` :

```
mpirun_cmd : $ASTER_ROOT/public/mpi/bin/mpirun ...
```

ou dans un fichier `.export` :

```
R repe $HOME/results_repe R 0
```

8.1 Pour l'utilisateur

Si la commande `as_run` n'est pas disponible directement, il faut exécuter :

```
salome shell -- as_ run [options] [arguments]
```

`salome` étant un script disponible dans l'installation de Salome-Meca. On peut le trouver en remplaçant `nnn` par l'indice de version dans : `/opt/salome-meca/nnn/appli_Vnnn/salome` .

- Lance l'exécution décrite par le profil (action par défaut) :

```
as_run --run [options] nom_profil.export
```

Le fichier export peut être en local ou sur une machine distante.
Chaque ligne du fichier commence par :

- P : définition d'un paramètre,
- A : définition d'un argument de la ligne de commandes de *Code_Aster,*
- F : définition d'un fichier,
- R : définition d'un répertoire,
- N : utilisé uniquement pour l'atelier de génie logiciel (AGLA).

Le format pour les paramètres et arguments est : `P nom_parametre valeur` ou `A nom_argument valeur`.
Pour les fichiers et répertoires, le format est :

```
F/R type chemin DRC unite_logique
```

où `DRC` précise si le fichier ou répertoire est en `D`onnée, `R`ésultat (les deux sont possibles ensemble), et si le contenu est `C`ompressé.

- Exécute rapidement un calcul à partir des fichiers en arguments :

```
as_run --quick [options] file1 [file2 [...]]
```

Les développeurs peuvent indiquer une surcharge de sources fortran (resp. python) avec les options `--surch_fort` (resp. `--surch_pyt`).

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

- Exécute un cas-test en interactif :
```
as_run --test [options] testcase [results_directory]
```

- Produit une bibliothèque dynamique nommée `FILE` en compilant les fichiers source `src1`... `srcN`. Utilisé typiquement pour construire une bibliothèque UMAT. :
```
as_run --make_shared --output=FILE [src1 [...]] srcN
```

C'est le compilateur et les options de compilations définies dans le fichier `config.txt`, de la version par défaut ou choisie avec l'option `--vers`, qui sont utilisés pour construire la bibliothèque. `FILE` est le nom de la bibliothèque produite qui sera indiqué dans les mots-clés de *Code_Aster*.

8.2 Pour le développeur

Remarque

Ces fonctions ne sont plus maintenues et pourraient disparaître.

- Affiche un fichier source : fortran, c, python, capy, cata, histor ou test :
```
as_run --show [options] obj1 [obj2...]
```

- Copie un fichier source dans le répertoire courant :
```
as_run --get [options] obj1 [obj2...]
```

- Affiche le diff d'un fichier source : fortran, c, python, capy, cata, histor ou test :
```
as_run --diff [options] obj1 [obj2...]
```

- Affiche la subroutine principale d'une commande *Code_Aster* :
```
as_run --showop [options] commande[.capy]
```

- Construit un fichier export pour lancer un cas-test et l'imprime sur stdout :
```
as_run --get_export testcase_name
```

- Construit une liste de cas-tests à partir de commandes/mots-clés et/ou vérifiant des critères de temps cpu ou mémoire :
```
as_run --list [--all] [--test_list=FILE] [--filter=...]
   [--command=...] [--user_filter=...] [test1 [test2 ..]]
```

- Construit le diagnostic des cas-tests *Code_Aster* (depuis DIRi ou le répertoire astest par défaut) et écrit un fichier pickled du résultat :
```
as_run --diag [--astest_dir=DIR1,[DIR2]] [--test_list=LIST]
   [--only_nook] [diag_result.pick]
```

- Opération sur les catalogues de messages de *Code_Aster*. subroutine = retourne les messages appelés par "subroutine". message_number = retourne les routines utilisant ce message. check = vérifie les catalogues et affiche quelques statistiques. move = déplace un message d'un catalogue à un autre et produit les catalogues et fichiers source modifiés. :
```
as_run --messages subroutine | message_number
as_run --messages check [--fort=...] [--python=...] [--unigest=...]
as_run --messages move old_msgid new_msgid
```

- Retourne les informations sur les processeurs et la mémoire des machines données :
```
as_run --get_infos [--output=FILE] host1 [host2 [...]]
```

8.3 Pour maintenir une installation locale

- Retourne le numéro de la version de développement :
```
as_run --getversion [options]
```


Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

- Retourne le chemin d'installation de la version de développement :
```
as_run --getversion_path [options]
```

- Affiche des informations à propos de l'installation et de la configuration :
```
as_run --showme [options] bin|lib|etc|data|locale|rcdir
as_run --showme [options] param parameter_name
```

8.4 Pour les interfaces lançant des calculs

- Lance une exécution (en appelant as_run dans un processus séparé) :
```
as_run --serv user@mach:/nom_profil.export
```

- Envoie le contenu de "filename" (éventuellement sur une machine distante) aux adresses email1, email2,... :
```
as_run --sendmail [--report_to=EMAIL1,EMAIL2] filename
```

- Retourne les informations sur la configuration d'un serveur : batch, interactif (oui/non, limites), noeuds de calcul, versions :
```
as_run --info
```

- Retourne l'état, le diagnostic, le noeud de calcul, le temps cpu et le répertoire de travail d'un calcul :
```
as_run --actu job_number job_name mode
```

- Ouvre le fichier output ou error sur l'écran fourni :
```
as_run --edit job_number job_name mode output|error DISPLAY
```

- Affiche la fin du fichier fort.6 ou les lines correspondant au filtre :
```
as_run --tail job_number job_name mode fdest nb_lines [regexp]
```

- Arrête un calcul et détruit les fichiers associés :
```
as_run --del job_number job_name mode [node] [--signal=...]
```

- Supprime les fichiers des calculs qui ne sont pas dans la liste :
```
as_run --purge_flash job_number1 [job_number2 [...]]]
```

- Convertit un fichier (éventuellement distant) au format html et écrit le résultat dans FILE :
en version bêta
```
as_run --convert_to_html [user@machine:]file --output=FILE
```

8.5 Pour les tâches d'administration

- Insère une nouvelle entrée dans le système de suivi d'anomalies et copie les fichiers joints si un fichier export est fourni :
```
as_run --create_issue issue_file [export_file]
```

- Remplit les champs "corrvdev" ou "corrvexpl" (selon vers) dans les fiches trouvées dans `histor` et éventuellement les ferme :
```
as_run --close_issue --vers=VERS histor
```

- Extrait le contenu des fiches listées dans `input_file` dans `histor` :
```
as_run --extract_histor [--status=STAT] [--format=FORM]
   [--all_msg] input_file histor
```

- Prépare le profil pour insérer une exécution dans la base de données :
```
as_run --insert_in_db [export_file]
```


Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

8.6 Superviser le lancement et le retour d'un calcul

On fournit ici la méthodologie pour lancer des calculs depuis une machine locale vers un serveur de calcul, hors ihm graphique, en appelant les différents services de as_run.

Les explications fournies ci-après permettent de :
- lancer un calcul,
- surveiller un calcul (connaître son état),
- récupérer les résultats,
- éditer les fichiers output/error.

8.6.1 Hypothèses

On explique les différentes étapes en prenant l'exemple de l'exécution d'un calcul sur le serveur Aster5 depuis une machine Calibre7. Les fichiers de données doivent nécessairement être sur le poste Calibre, celui qui soumet le calcul, ou sur le serveur d'exécution (.export, fichiers de données, résultats, base, …).

Dans les exemples, on utilise les noms de machines suivants (et donc à adapter à chaque situation) :
- Calibre7 (client) : hostname=`dspXXXX`
- Aster5 (serveur) : hostname=`aster5.hpc.edf.fr`, user=NNI

L'entête utilisé dans les fichiers `.export` pour identifier le serveur est (il faut modifier NNI et `dspXXXX`) :

```
P mclient dspXXXX
P serveur aster5.hpc.edf.fr
P username NNI
P protocol_exec asrun.plugins.server.SSHServer
P protocol_copyto asrun.plugins.server.SCPServer
P protocol_copyfrom asrun.plugins.server.SCPServer
P aster_root /home/rd-ap-simumeca
P proxy_dir /scratch/NNI
P platform LINUX64
```

Vérification préliminaire : La connexion en ligne de commande `ssh aster5.hpc.edf.fr pwd` doit fonctionner sans mot de passe.

8.6.2 Commandes

L'ajout de l'option `--nodebug_stderr` supprime l'impression de la sortie `stderr` de as_run.

8.6.2.1 Récupération des infos du serveur (permet également de tester la connexion)

```
as_run --nodebug_stderr --proxy --info info.export
```

avec `info.export` qui ne contient que l'entête.
En particulier, on peut voir la liste des versions disponibles. Ensuite, on utilisera la version `testing`.

8.6.2.2 Lancement d'un calcul

```
as_run --nodebug_stderr --proxy --serv study.export
```

avec `study.export`::

```
<<< entete >>> +

P actions make_etude
```


Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

```
P nomjob jobname
P origine ASTK 1.13.4
P mode batch
P consbtc oui
P soumbtc oui
P version testing          <<< fournie par --info
P tpsjob 10
A memjeveux 512
A tpmax 600
P memjob 4194304
F comm $HOME/dev/codeaster/src/astest/zzzz100f.comm D 1
F mess /tmp/zzzz100f.mess R 6
F resu zzzz100f.resu R 8
R base NNI@aster5.hpc.edf.fr:/home/NNI/study1/base R 0
```

`/tmp/zzzz100f.mess` sera recopié sur la machine locale.

`zzzz100f.resu` sera recopié sur la machine locale dans le répertoire où a été exécuté `--serv`.

Il faut ajouter `NNI@aster5.hpc.edf.fr:` devant le nom pour préciser que ces fichiers sont déjà sur le serveur (ou devront y rester en résultat, cas de la base dans l'exemple).

8.6.2.3 Interrogation

Ensuite, on peut utiliser le même fichier `.export` pour les différentes actions, les derniers paramètres seront ignorés la plupart du temps, `post.export` :

```
<<< entete >>> +

P jobid 12399454           <<< fourni dans l'output de --serv
P mode batch
P nomjob jobname

P tail_nbline 50           <<< uniquement pour --tail

P edit_type output         <<< uniquement pour --edit
```

8.6.2.4 Actualisation de l'état du job

```
as_run --nodebug_stderr --proxy --actu \
       --schema=asrun.plugins.default.actu_simple post.export
```

Le schéma par défaut fait *actu + get_results*. Si on ne veut faire que l'actualisation, il faut indiquer --schema=asrun.plugins.default.actu_simple. Si on veut faire l'actualisation et la récupération des résultats, on fait simplement :

```
as_run --nodebug_stderr --proxy --actu post.export
```

8.6.2.5 Récupération des résultats

```
as_run --nodebug_stderr --proxy --get_results post.export
```

8.6.2.6 Supprimer le job

Interrompt le calcul s'il est en cours, supprime les fichiers du flasher :

```
as_run --nodebug_stderr --proxy --del post.export
```


Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

8.6.2.7 Consulter la fin de l'output d'un job en cours

```
as_run --nodebug_stderr --proxy --tail post.export
```

8.6.2.8 Éditer le fichier output

```
as_run --nodebug_stderr --proxy --edit post.export
```

Valeurs possibles pour `edit_type` : `output`, `error`, `export`, `script`, `diag`.

8.7 Options

```
--version             show program's version number and exit
-h, --help            show this help message and exit
-v, --verbose         rend le programme plus bavard
--silent              fonctionne aussi silencieusement que possible
-g, --debug           affiche les informations de debug
--stdout=FILE         permet de rediriger les messages habituellement écrits
                      sur sys.stdout
--stderr=FILE         permet de rediriger les messages habituellement écrits
                      sur sys.stderr (uniquement les messages d'asrun)
--log_progress=FILE   redirige les informations d'avancement vers un fichier
                      au lieu de sys.stderr
--nodebug_stderr      supprime l'impression des informations de debogage sur
                      stderr
-f, --force           force les opérations qui peuvent utiliser un cache
                      (téléchargement, compilation...)
--display=DISPLAY     valeur de la variable DISPLAY (NB : certaines
                      fonctions lisent cette valeur dans un fichier)
--rcdir=DIR           utilise le répertoire de ressources $HOME/'DIR' (le
                      défaut est .astkrc). Il faut éviter de d'indiquer un
                      chemin absolu car il sera passé aux serveurs distants.
--remote_shell_protocol=REMOTE_SHELL_PROTOCOL
                      protocole utilisé pour le lancement de commandes
--remote_copy_protocol=REMOTE_COPY_PROTOCOL
                      protocole utilisé pour la copie de fichiers
--copy_all_results    copy all results in the current directory (for --quick
                      action)
--proxy               demande à un serveur de lancer l'action spécifiée (par
                      exemple, en appelant as_run --serv sur le serveur)
--schema=SCHEMA       permet de modifier le comportement d'as_run en
                      utilisant un schéma alternatif

Options pour les opérations de maintenance:
  --filter=FILTER     filtres appliqués aux paramètres des tests : 'nom_para
                      < valeur' (comparaisons supportées <, >, =).
  --vers=VERS         Version de Code_Aster à utiliser (pour get, show,
                      showop)
  --force_upgrade     Force la mise à jour vers la prochaine version (par
                      exemple de 10.1.xx vers 10.2.0)
  -o FILE, --output=FILE
                      redirige le résultat dans FILE au lieu de stdout
  --surch_pyt=REP     un ou plusieurs répertoires (séparés par une virgule)
                      contenant les fichiers python ajoutés
  --config=FILE       utilise un autre fichier "config.txt" (seulement pour
                      make, update and auto_update).
  --only_nook         rapporte seulement les erreurs (le temps passé dans
                      les tests ok est compté)
  --surch_fort=REP    un ou plusieurs répertoires (séparés par une virgule)
```


Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

```
                      contenant les fichiers fortran ajoutés
  -a, --all           copie de tous les fichiers du cas-test
  --destdir=DIR       répertoire racine (fake-root) où les fichiers seront
                      copiés
  --keep_increment    met à jour une version incrément par incrément en
                      conservant les fichiers exécutables intermédiaire
  --search=REGEXP     conserve les tests vérifiant l'expression régulière
                      donnée (ou une simple chaine de caractères).
  --unigest=FILE      un fichier unigest (pour la suppression)
  --command=COMMANDE[/MOTCLEFACT[/MOTCLE[=VALEUR]]]
                      conserve les tests utilisant la commande  et les mots-
                      clés donnés
  --test_list=FILE    liste des cas-tests
  --report_to=EMAIL   adresse de messagerie où envoyer le rapport d'une
                      exécution (seulement pour --auto_update)
  --user_filter=FILE  fichier contenant les classes testlist.FILTRE. Voir
                      [...]/share/codeaster/asrun/examples/user_filter.py
                      comme exemple.
  --astest_dir=DIR    répertoire des cas-tests à analyser
  -l, --local         les fichiers ne sont pas cherchés sur le serveur
                      distant mais sur la machine locale
  --nolocal           force la recherche des fichiers sur le serveur distant
                      (inverse de --local)

Options for operations on jobs:
  --signal=SIGNAL     envoie un signal au calcul (KILL|USR1)
  --result_to_output  Redirige le résultat dans FILE au lieu de stdout

Options pour l'interface au REX:
  --status=STAT       une erreur se produit si les fiches ne sont pas dans
                      cet état
  --all_msg           recupère tous les messages des fiches
  --format=FORM       format de l'histor généré (texte ou html)
```

8.7.1 Précisions supplémentaires

- Quand on récupère un fichier source avec `get`, `show` ou `showop`, celui-ci est mis dans `/tmp/astk_`login`/cache`. Si on demande de nouveau le même fichier, celui-ci est pris directement dans ce répertoire sauf si l'option `--force` est activée ; les fichiers qui ont plus d'une journée sont automatiquement supprimés du cache.

- `--local`/`--nolocal` : ces options sont utilisées quand le comportement par défaut ne convient pas. Dans l'utilisation de `show`/`get`, il est préférable d'accéder aux sources locaux, c'est donc le défaut. Dans ce cas, on utilisera `--nolocal` pour accéder à une version sur le serveur de référence. Lors de la maintenance d'une installation locale, le défaut est de télécharger les mises à jour sur un serveur distant. Néanmoins, si on récupérer les paquets en local, on peut utiliser `--local` pour les installer.

- Le fichier de configuration principal est `[ASTER_ROOT]/etc/codeaster/asrun`, l'utilisateur peut définir certaines valeurs dans `$HOME/.astkrc/prefs` (où `$HOME` est le répertoire par défaut de l'utilisateur), en particulier le nom d'utilisateur à utiliser sur le serveur distant (`devel_server_user`) pour éviter une alarme à chaque lancement de `as_run`.

- Des compléments sur le fonctionnement interne de `as_run` (mode client/serveur, « schémas/plugins ») ou sur certaines options sont disponibles dans la documentation développeur embarquée avec `as_run` dans `[ASTER_ROOT]/share/codeaster/asrun/doc/`.

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

9 Questions Fréquentes

« Rien ne se passe quand on essaie de lancer un calcul, d'éditer un fichier ou d'ouvrir un terminal sur un serveur distant » *ou bien* ***« Code retour = 2, Profile copy failed dans la fenêtre des messages au lancement d'un calcul »***
Il est probable que la machine locale ne puisse pas communiquer correctement avec le serveur distant. Les communications utilisant le protocole rcp/rsh, les fichiers `.rhosts` sur la machine locale et sur les serveurs distants doivent être correctement renseignés. Si on utilise le protocole scp/ssh, il faut que les clés privées et publiques soient cohérentes.
On peut le vérifier en quittant astk, et en le relançant de cette manière :

`astk --debug 0 --check` (`--debug 0` sert à limiter les impressions)

Lire attentivement les informations qui sont imprimées. astk fournit des informations sur la variable d'environnement DISPLAY (vérifier qu'elle est correctement définie), et teste la connexion entre la machine où est lancée astk et les serveurs distants. En cas de problème de connexion avec un serveur, astk suggère une modification du fichier `.rhosts` de la machine cible.

« Quand on soumet un calcul en batch, on a le message : Le numéro du job et la classe n'ont pas pu être récupérés »

Le temps ou la mémoire demandé est probablement au delà des limites des classes de travaux. Voir dans la fenêtre des messages, il y a probablement un message du gestionnaire de batch du style "Cannot exceed queue's hard limit(s)".
Diminuer le temps ou la mémoire, ou choisir explicitement une classe batch qui convient dans les options supplémentaires.

« En cliquant sur Valeur par défaut le nom de fichier reste vide »
Cette fonction utilise le nom du profil pour déterminer une valeur par défaut, il faut enregistrer le profil avant de pouvoir utiliser cette fonction.

Faire un rapport de bug :
Si une erreur reste inexpliquée, voici la méthode pour faire un rapport de bug en fournissant les traces d'exécution.

1. Fermer astk s'il est en exécution et démarrer une nouvelle session (cela permet de limiter les traces au minimum pertinent).
2. Reproduire le problème en faisant le minimum de chose.
3. Quitter astk et joindre au support le fichier `$HOME/.astkrc/log.astk` (ou `.astkrc_salomeXXXX` dans Salome-Meca).

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)