



# AUTO-FORMATION BUILDSYSPRO 4. CONSEILS ET ASTUCES

Support de formation de BuildSysPro  
Dernière révision : janvier 2016

EDF R&D  
Département Enerbat (Energie dans les Bâtiments et les Territoires)  
Groupe « Simulation énergétique et bâti »



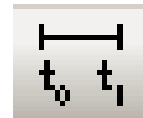


## 4. CONSEILS ET ASTUCES

- [Quelques conseils en vrac](#)
- [Configurer l'IHM](#)
- [Détecter les erreurs de programmation](#)
- [Utiliser des scripts .mos](#)

# QUELQUES CONSEILS EN VRAC

- Pour déplacer un composant A dans l'arborescence
  - Utiliser la fonction rename et non duplicate afin que les modèles utilisant A conservent son chemin dans l'arborescence
- Pointer vers des fichiers externes selon leur chemin en relatif
  - Exemple : « ./Documentation/Meteo/METEONORM/France/trappes.txt »
- Conseils pour le setup de la simulation
  - Décocher *Store variables at events*



# CONFIGURER L'IHM

# PLAN DU TUTORIEL

## ■ IHM ?

- Les interfaces de saisie des paramètres d'un modèle
- La documentation attenante

## ■ Objectifs

- Connaître les pré-requis au niveau du langage Modelica (pour savoir où renseigner les paramètres liés à l'IHM)
- Connaître les principales options d'affichage et savoir les coder

## ■ Plan

- Organisation de l'interface de saisie des paramètres d'un modèle
- Remplir une documentation html

# PARTIE DU CODE À REGARDER

## MODÈLE D'EXEMPLE : « MODELTEMPLATE »

The screenshot shows the Modelica Text editor interface. The Package Browser on the left lists packages including DymolaCommands, Modelica Reference, Modelica, Unnamed, BuildSysPro, ModelTemplate, Building, Systems, Boundary Conditions, and BuildingStrike. The main editor displays the ModelTemplate code, which is divided into sections: a declarative section for parameters and variables, a section for algorithms and equations, and a section for equations and connections between models. Annotations are provided for each section:

- Déclaration des paramètres et variables du modèle**  
*y compris modèles utilisés dans l'assemblage*
- Équations et algorithmes**  
*y compris liaisons entre modèles*
- Annotations cachées par défaut**  
(html, aspects graphiques, IHM, ...)

```
partial model ModelTemplate
  "Model template with recommendations in terms of documentation and code-writing"

  //----- Declarative section - please comment each variable and parameter -----//
  // Do not write any equation in this section for variables. Otherwise, the variable will be considered as a parameter (constant)

  // Declaration of model parameters using associated units and range of variation if necessary
  parameter Real alpha( min=0, max=1)
    "Alpha coefficient - range of variation between 0 and 1";
  parameter Modelica.SIunits.Length L=2 "Length";
  parameter BuildSysPro.Utilities.Types.FileNameIn Data=
    "../Resources/Donnees/Meteos/METEONORM/France/trappes.txt"
    "Path of weather data file";
  parameter Boolean Bool "Parameter showing radio buttons" B;
  parameter Integer Choice "Parameter showing a Pop-down menu" B;
  parameter BuildSysPro.Utilities.Records.ParoiGenerique Wall
    "Choice between wall data records" B;

  // Declaration of variables
  Modelica.SIunits.Length D "Distance";

  // Used models

  // Variables or parameters visible only in the model
  protected
    parameter Real beta=alpha*100 "Alpha in the form of a percentage...";
    Real gamma "Intermediate variable not visible by default in the results";

  //----- Section of algorithms & equations -----//

  //-- Algorithms - for classic or sequential calculations, not taking part of the equational
  // Be careful, an algorithm should not be considered as an equation, contrary to equations
  algorithm
    D:=beta*L; // For instance...

  //-- Equations and connections between models --//
  // The equations form the matrix system solved by the solver. Number of equation = Number
  equation
    gamma = if Bool then D else 0;

end ModelTemplate;
```

# MODIFICATION DES ANNOTATIONS

## *ANNOTATION "DIALOG"*

- Paramètres possibles dans l'annotation "Dialog"

**annotation**(Dialog(enable = true,

    tab = "General",

    group = "Parameters",

    showStartAttribute = false,

    groupImage = "modelica://MyPackage/Resources/Images/image.png",

    connectorSizing = false));

- Explications succinctes...

- enable : si =false alors saisie du paramètre désactivée (cellule grisée)
- tab & group : position dans la fenêtre de dialogue
- showStartAttribute : si =true alors possible de définir une valeur initiale
- groupImage : pour insérer une image dans la fenêtre de dialogue (1 par groupe)
- connectorSizing : utilisé dans la librairie fluids & state machines pour redimensionner la taille d'un vecteur

# MODIFICATION DES ANNOTATIONS

## *ANNOTATION "DIALOG" : ONGLETS ET GROUPES*

- Création d'onglets "tab" et de groupes "group" de paramètres
  - En personnalisant dans le code les annotations qui suivent chacun des paramètres et en utilisant la fonction « Dialog() »

model Annotations

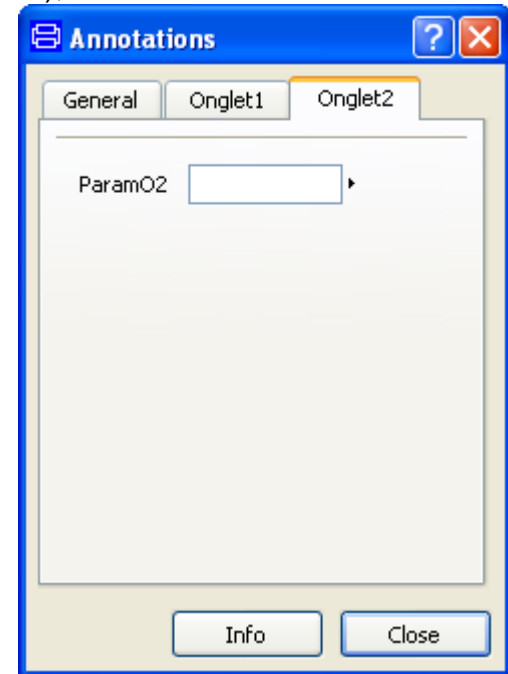
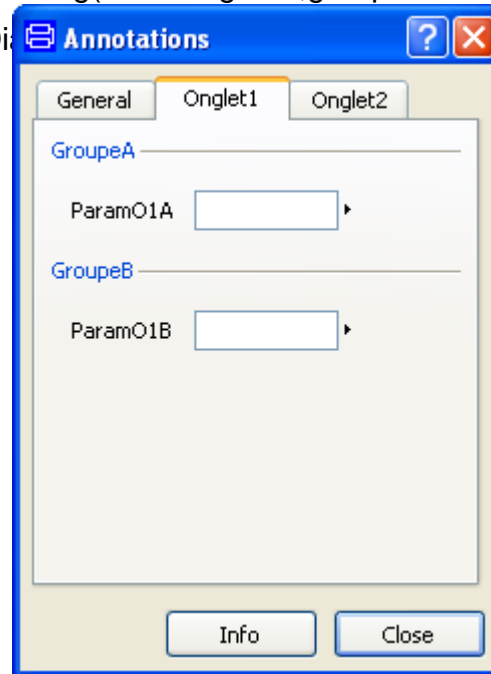
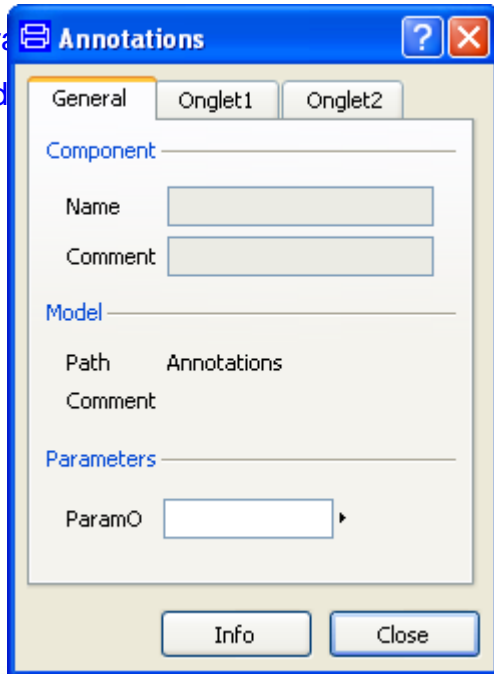
parameter Real ParamO; //Par défaut tab="General" et group="Parameters"

parameter Real ParamO1A annotation(Dialog(tab="Onglet1",group="GroupeA");

parameter Real ParamO1B annotation(Dialog(tab="Onglet1",group="GroupeB");

parameter Real ParamO2 annotation(Dialog(tab="Onglet2",group="GroupeC");

end



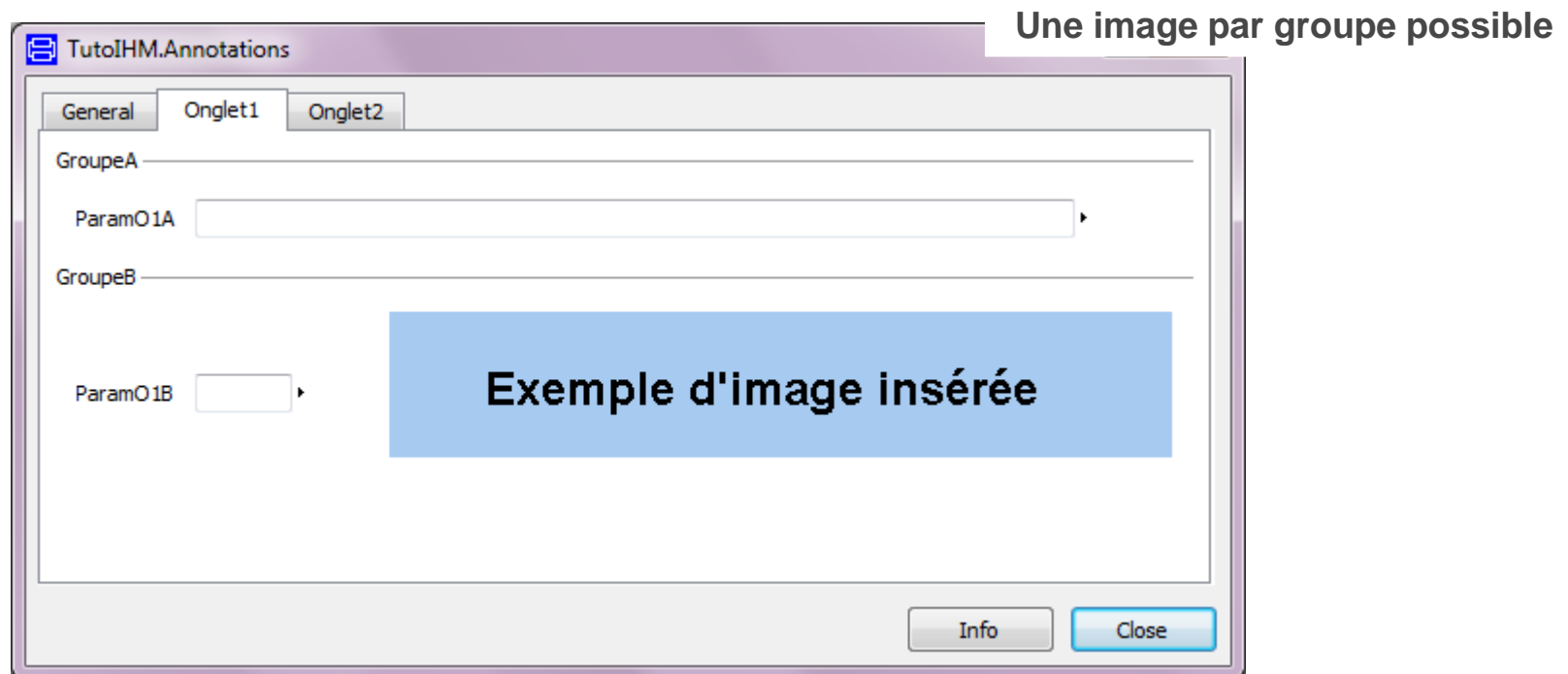


# MODIFICATION DES ANNOTATIONS

## *ANNOTATION "DIALOG" : INSERTION D'IMAGE*

- Reprise de l'exemple précédent en ajoutant dans "Dialog" la notation "groupImage"

```
parameter Real ParamO1B annotation(Dialog(tab="Onglet1",group="GroupeB",groupImage = "modelica://OSMOSYS/Documentation/Images/exemple.bmp"));
```



# MODIFICATION DES ANNOTATIONS

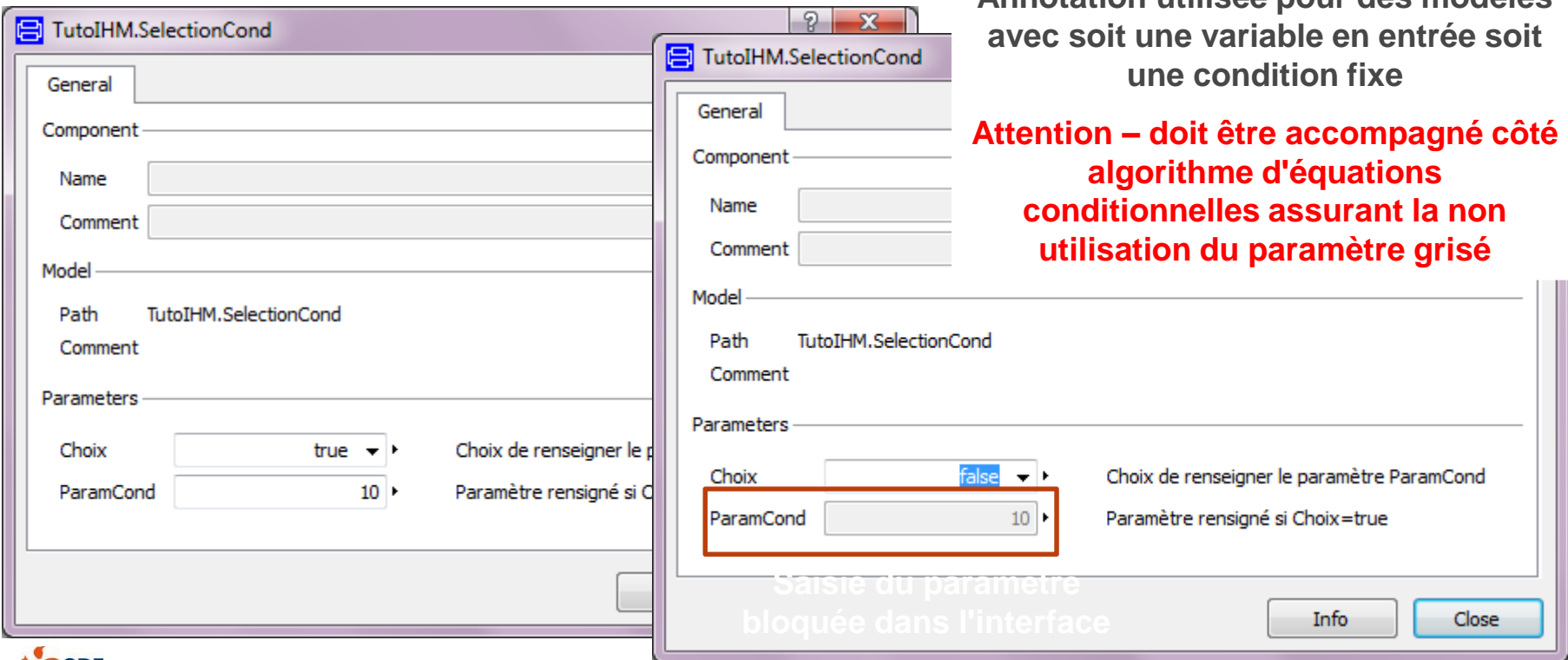
## ANNOTATION "DIALOG" : ESPACE DE SAISIE

### CONDITIONNEL

- Saisie impossible d'un paramètre conditionnée par la valeur d'un autre paramètre : notation "enable"

parameter Boolean Choix=true "Choix de renseigner le paramètre ParamCond";

parameter Integer ParamCond=10 "Paramètre renseigné si Choix=true" annotation(Dialog(enable=Choix));



Annotation utilisée pour des modèles avec soit une variable en entrée soit une condition fixe

**Attention – doit être accompagné côté algorithme d'équations conditionnelles assurant la non utilisation du paramètre grisé**

Saisie du parametre bloquée dans l'interface

# MODIFICATION DES ANNOTATIONS

## *ANNOTATION "CHOICES" : LISTE DE CHOIX*

- Liste déroulante par défaut

parameter Boolean ListeD "Paramètre montrant le choix multiple"

annotation(choices(choice=true "Vrai", choice=false "Faux"));

- Case d'option en remplacement de la liste via l'ajout de l'annotation "radioButtons"

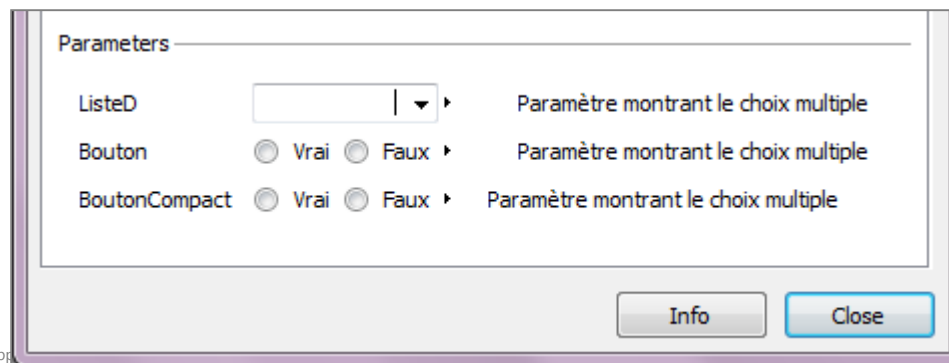
parameter Boolean Bouton "Paramètre montrant le choix multiple"

annotation(choices(choice=true "Vrai", choice=false "Faux", radioButtons=true));

- Rendu "compact" (aligné à gauche) des commentaires

parameter Boolean BoutonCompact "Paramètre montrant le choix multiple"

annotation(Dialog(compact=true),choices(choice=true "Vrai", choice=false "Faux", radioButtons=true));



# MODIFICATION DES ANNOTATIONS

## *ANNOTATIONS SPÉCIFIQUES*

- Annotations utilisées dans des nouveaux types de BuildSysPro

- **FileNameIn**

- Permet la sélection d'un fichier dans l'explorateur windows
- *Exemple : BuildSysPro.BoundaryConditions.Weather.Meteofile*  
`annotation(Dialog(__Dymola_loadSelector(caption="Ouvrir le fichier suivant")))`

- **FileNameOut**

- Permet la création d'un fichier de sortie directement via l'explorateur windows (avec choix du format csv, mat et txt)
- *Exemple : BuildSysPro.Utilities.Analysis.Simulation2CSV()*  
`annotation(Dialog(__Dymola_saveSelector(filter="Fichier CSV (*.csv);;Fichier Matlab (*.mat);;Fichier txt (*.txt)",  
caption="Fichier d'enregistrement")),`

# MODIFICATION DES ANNOTATIONS

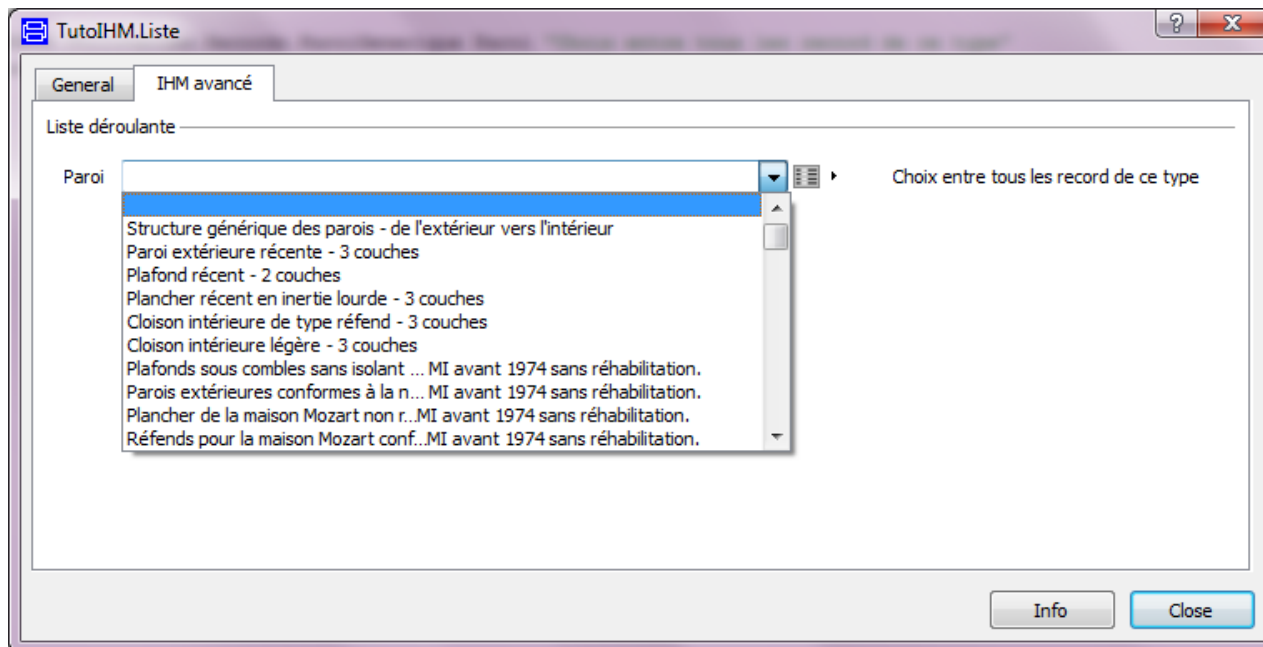
## CRÉATION DE RECORD SPÉCIFIQUES

- Exemple des parois : "record" créé spécifiquement pour faciliter la saisie des paramètres
  - Record "SolideGenerique" qui a des valeurs de (lambda, rho, Cp)
    - Un matériau solide est défini par des propriétés thermiques (conductivité, masse volumique et chaleur spécifique)
    - Un solide sera défini comme ci-dessous (Exemple *BuildSysPro.Utilities.Data.Solids.Beton*)  
`record Beton = BuildSysPro.Utilities.Records.SolideGenerique (lambda=1.5, rho=2500, c=1000) "<html>Béton (lambda=1.5)</html>";`
  - Record "ParoiGenerique" qui a des valeurs de (n, m, e, mat, position)
    - Une paroi est définie par plusieurs couches de matériaux (nombre de couches de matériaux, nombre de maille de la discrétisation spatiale, épaisseur des couches et matériaux de chaque couche)
    - Des parois spécifiques ont besoin en plus d'avoir l'information sur la position de l'isolant
    - Voici une paroi type (Exemple *BuildSysPro.Utilities.Data.WallData.paroExtRecente*)
  - `record` paroiExtRecente = `BuildSysPro.Utilities.Icons.ParoExt` (  
    `n=3, m={4,3,1}, e={0.2,0.15,0.01},`  
    `mat={BuildSysPro.Utilities.Data.Solids.PolystyreneExpanse30(),`  
    `BuildSysPro.Utilities.Data.Solids.PlatrePlaque()},`  
    `positionIsolant={0,1,0}) "Paroi extérieure récente - 3 couches"`

# MODIFICATION DES ANNOTATIONS

## CRÉATION DE RECORD SPÉCIFIQUES

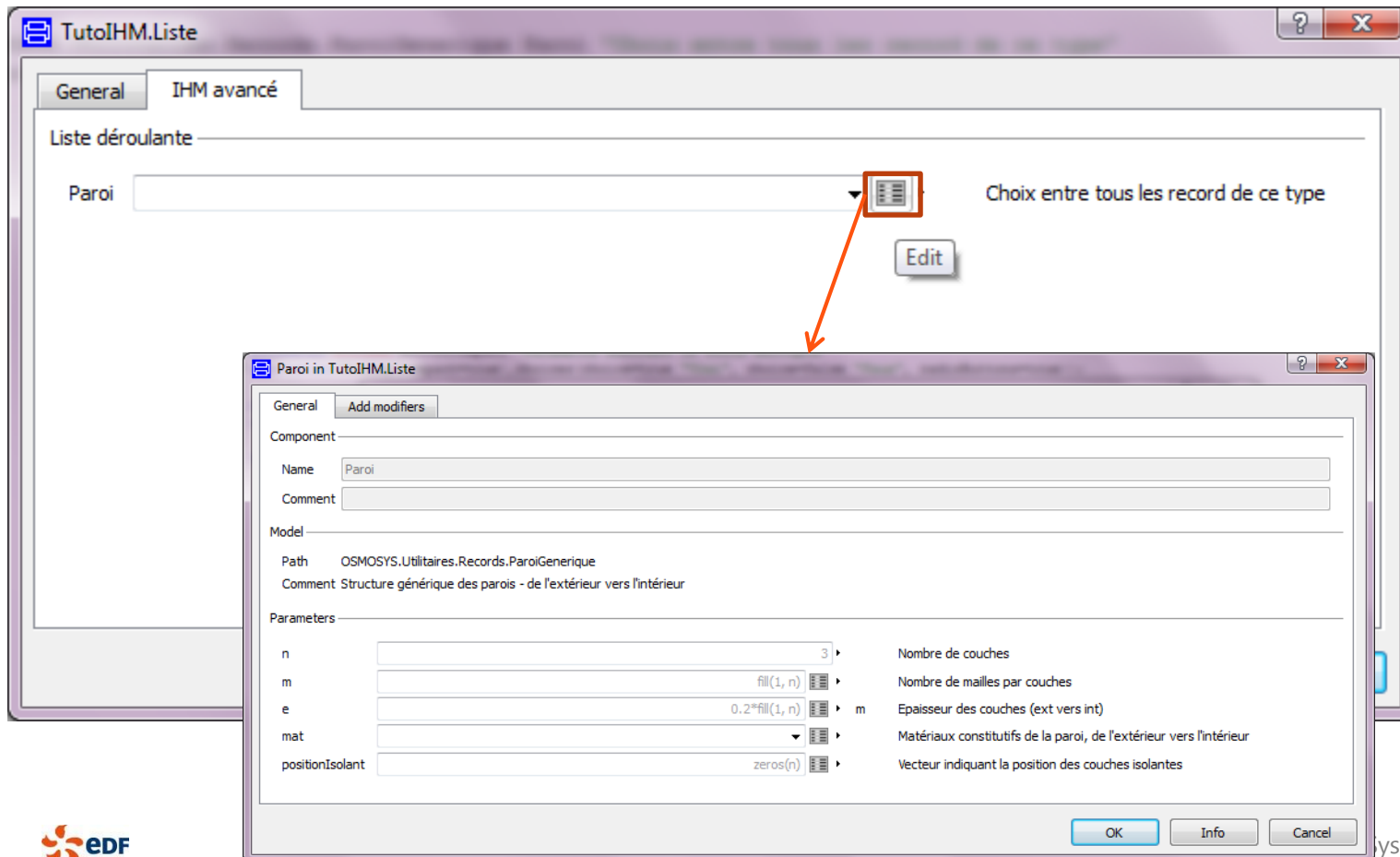
- Exemple des parois : "record" créé spécifiquement pour faciliter la saisie des paramètres
  - Utilisation du record ParoiGenerique  
parameter BuildSysPro.Utilities.Records.ParoiGenerique Paroi "Choix entre tous les record de ce type"  
annotation(Dialog(tab="IHM avancé",group="Liste déroulante"),\_\_Dymola\_choicesAllMatching=true);
  - L'annotation "\_\_Dymola\_choicesAllMatching" permet de proposer en liste déroulante tous les matériaux qui sont construits sur le type "Utilities.Records.ParoiGenerique"



# MODIFICATION DES ANNOTATIONS

## CRÉATION DE RECORD SPÉCIFIQUES

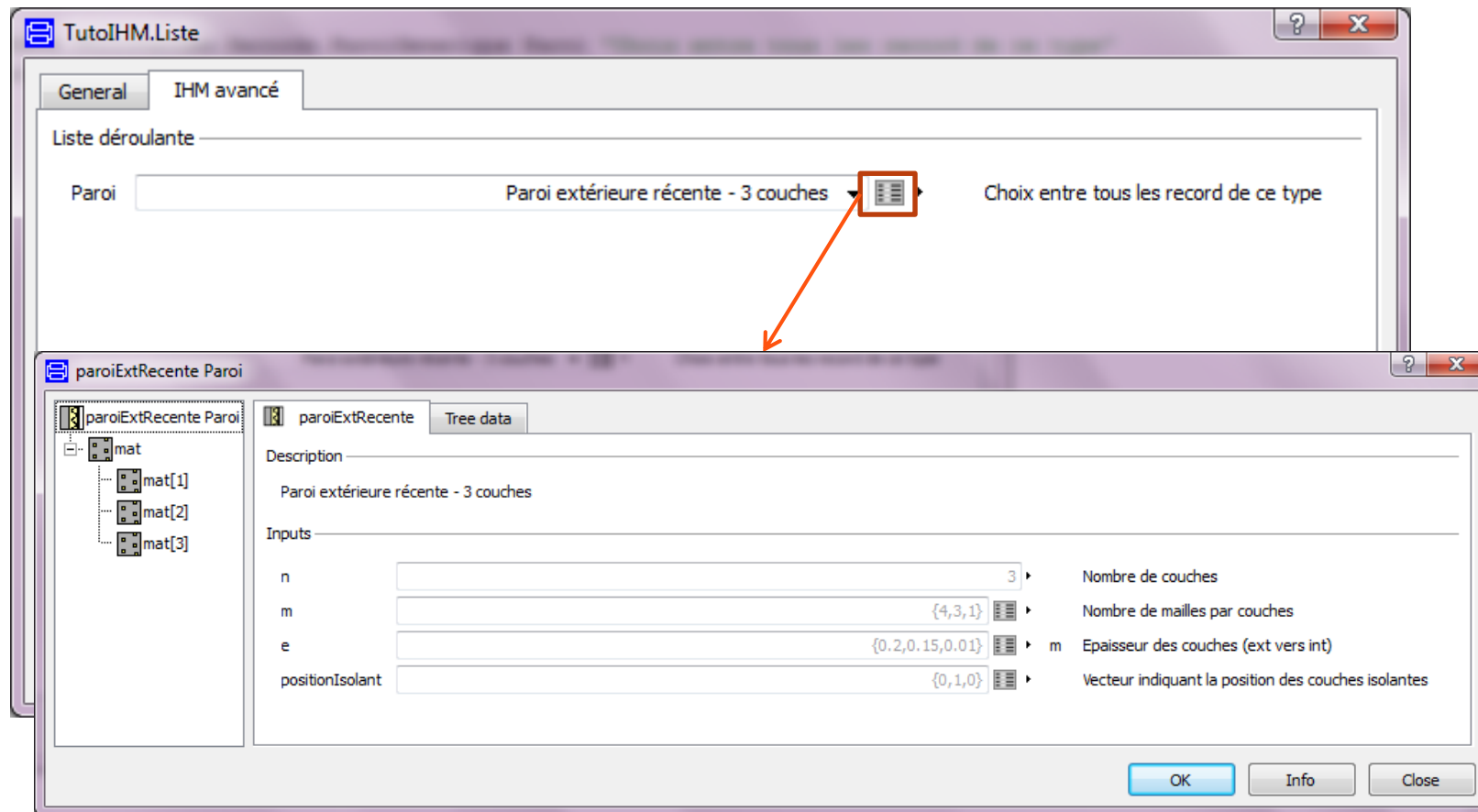
- Exemple des parois : "record" créé spécifiquement pour faciliter la saisie des paramètres



# MODIFICATION DES ANNOTATIONS

## *CRÉATION DE RECORD SPÉCIFIQUES*

- Exemple des parois : "record" créé spécifiquement pour faciliter la saisie des paramètres





# DOCUMENTATION HTML

## *INSERTION D'IMAGE*

- Pour insérer une image

1/ Dans le menu Info-Editor de la documentation, cliquer sur l'icône Insert Image



2/ Il faut alors modifier le chemin vers l'image en prenant garde au nom du package parent

```

```

doit être remplacé par :

```

```

*Attention, la notation en relatif suivante pose quelques problèmes et n'est pas toujours stable donc toujours préférer la syntaxe avec modelica://*

```
 Notation en relatif
```

Remarque : on peut se passer de l'étape 1 et entrer directement le chemin vers l'image si on est habitué au code html ...

# DOCUMENTATION HTML

## INSERTION D'ÉQUATION

- Pour insérer une équation ou une expression

1/ Dans le menu Info-Editor de la documentation, cliquer sur l'icône Insert Equation



2/ Saisir l'équation (fenêtre intuitive pour écrire des équations pouvant être complexes)

3/ Dymola enregistre alors les équations dans un dossier créé dans le même répertoire, il faut penser à le copier dans le dossier Documentation et modifier le chemin comme pour une image

```

```

doit être remplacé par

```

```

**Remarque :** la notation « alt= » permet de ré-éditer l'équation donc il faut la conserver pour pouvoir modifier celle-ci plus tard, sinon, cela revient à juste coller l'image de l'équation éditée sous Dymola

L'équation affichée serait alors la suivante

$$\alpha = \beta + \sqrt{\pi}$$

# DÉTECTER LES ERREURS DE PROGRAMMATION

# PRÉVENTION DES ERREURS DE MODÉLISATION

## ■ Erreurs couramment rencontrées

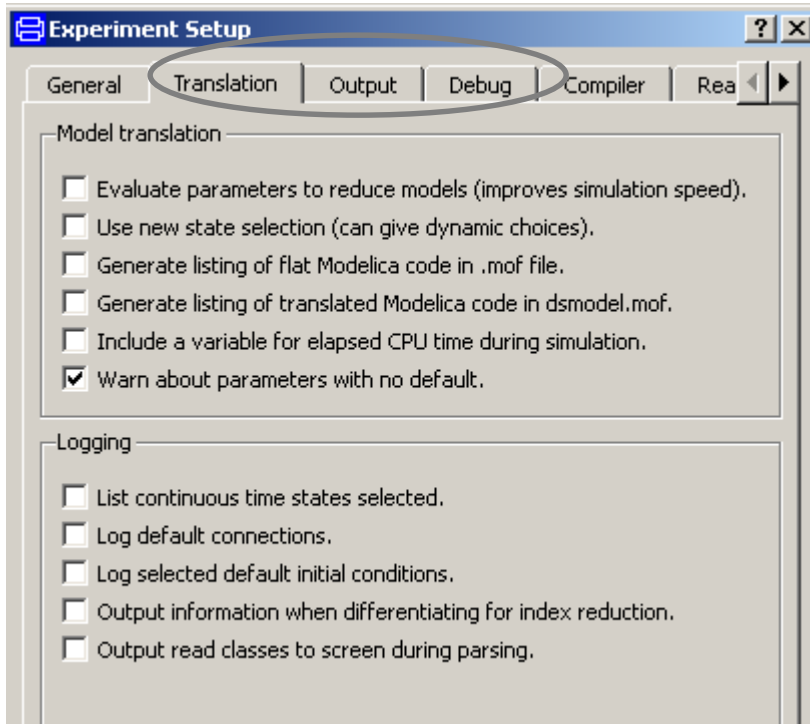
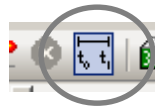
- Erreurs de syntaxe et d'unités de mesure
- Composants non-connectés
- Conditions initiales non précisées
- Sous ou sur-dimensionnement du nombre d'équations par rapport au nombre d'états

## ■ Astuces de modélisation

- Augmenter la complexité du modèle pas à pas (et le simuler pas à pas)
- Vérifier la syntaxe des modèles après toute modification du code
- Faire attention aux alertes (warnings) à la vérification (check) et à la translation (translate)
- Utiliser les connecteurs déjà définis ou définir des nouveaux par duplicata
- Tester les modèles (validations unitaires et d'assemblage)
  - Tester une difficulté à la fois, « diviser pour mieux régner »
  - Développer des scripts pour les tests unitaires
  - Envisager un répertoire « Exemples » et des scripts ou fonctions associés par classe de modèles
- Adapter les modèles aux besoins
  - Les modèles simples peuvent donner souvent satisfaction ⇒ amélioration
- Utiliser des conventions de nom pour les variables et ajouter des commentaires
  - Par exemple : `temperature_entree`

# IDENTIFICATION DES ERREURS DE MODÉLISATION

- Outil de détection d'erreur (debug) de Dymola



# EXPERIMENTAL SETUP

## *TRANSLATION*

- **Translation du modèle** ⇒ **Améliorer la convergence et la vitesse**
  - Évaluer la qualité du modèle en vue d'une réduction d'indice
  - Changement automatique de variables d'état ⇒ diminuer les non-linéarités
  - Générer dans un fichier la liste de toutes les déclarations de variables ainsi que les équations du code Modelica
    - Investiguer les boucles algébriques
    - Faciliter la compréhension des problèmes numériques du modèle
  - Quantifier l'effort de calcul par variable et par pas de temps de simulation
  - Signaler les paramètres des modèles sans valeurs par défaut
  
- **Messages et avertissements**
  - Identifier les états continus
  - Avertissements liés aux connexions des modèles
  - Avertissements liés à l'initialisation des états
  - Identifier les équations qui seront différenciées (numériquement ou analytiquement) tout au long de la simulation

# EXPERIMENTAL SETUP

## *OUTPUT*

- **Format des résultats**
  - Binaire ou texte (ASCII)
  - Simple ou double précision
  
- **Stockage de variables au cours et en fin de la simulation**
  - Variables d'état
  - Dérivées
  - Entrées et sorties
  - Variables auxiliaires, intermédiaires de calcul
  - Variables protégées, locales
  
- **Restitution au cours et en fin de la simulation**
  - Stockage des variables à chaque événement discret
  - Restitution équidistante dans le temps (en fonction de l'intervalle choisi : Experimental Setup > General > Output interval)

# EXPERIMENTAL SETUP

## *DEBUG*

- Afficher ou non les alertes (paramètres, états, ...)
- Inclure ou non les alertes dans le code traduit (utile pour des applications indépendantes de Dymola)
- Alerte une discontinuité (événement, ...) à l'initialisation ou pendant la simulation
- Diagnostic/détection, itérations internes d'un problème non-linéaire
- Informations concernant la convergence/divergence de chaque variable
- Informations concernant le changement automatique d'états



# IDENTIFICATION DES ERREURS ET SOLUTIONS PROPOSÉES

## *PB = LA COMPILATION DU MODÈLE ÉCHOUE*

### ■ Symptômes

- Erreurs de syntaxe identifiées à la vérification du code (fenêtre de messages - Syntax, Translation tab)

### ■ Remèdes

- Corriger les erreurs de syntaxe
- Problème mal posé (nb états vs nb équations)



### ■ Tests

- Vérification du modèle

# IDENTIFICATION DES ERREURS ET SOLUTIONS PROPOSÉES

## *PB = DIVERGENCE DU SOLVEUR AU PREMIER PAS DE TEMPS*

### ■ Symptômes

- Message d'erreur : Integration terminated before reaching « StopTime » at T=0

### ■ Causes

- Problème initial mal posé, variables sans conditions initiales

### ■ Tests

- Experimental Setup > Translation > Log selected default initial conditions
  - Identifier les variables d'état non-initialisées
- Experimental Setup > Debug : min / max assertions – All variables
  - Imposer des contraintes (physiques, ...) sur la dynamique des états

### ■ Remèdes

- Ajouter des valeurs raisonnables à l'initialisation des variables d'états
  - start=[...], fixed=true, initial equation
- Borner l'évolution des variables d'état et leur associer des unités SI
  - min= [...], max=[...], Modelica.SIunits

# IDENTIFICATION DES ERREURS ET SOLUTIONS PROPOSÉES

## *PB = CONVERGENCE LENTE OU DIVERGENCE À $T > 0$*

### ■ Symptômes

- Message d'erreur : Warning: You have many state events. It might be due to chattering
- Message d'erreur : Probably the communication interval is too large or the systems is stiff
- Message d'erreur : Integration terminated before reaching « StopTime » at  $T = xxx > 0$

### ■ Causes

- Divergence à cause du traitement numérique des évènements, discontinuité
- Phénomènes avec des constantes de temps faibles
- Système d'équations fortement non-linéaire et de taille importante

### ■ Tests

- Experimental Setup > Translation > Use new state selection
- Experimental Setup > Debug > Nonlinear solver diagnostics
- Experimental Setup > General ⇒ changement de solveur, tolérance
- Fenêtre de DOS du solveur (pendant la simulation)
  - Appuyer ctrl-C ⇒ mise à jour du solveur
  - Appuyer ctrl-C pour la deuxième fois ⇒ mode debug (pas à pas)
    - c : continuer
    - q : quitter
    - l : enregistrement dans un fichier log

### ■ Remèdes

- Reformuler les équations du modèle
- Éliminer les constantes de temps trop faible
- Rendre statiques certaines variables d'état (Real x(stateSelect=StateSelect.prefer)
- Vérifier les problèmes mathématiques mal posés (division par zéro, log de zéro, ....)
- Borner l'évolution des variables d'état et leur associer des unités SI
- Assert( $P > 0$ , « La pression P est toujours positive » )

# UTILISER DES SCRIPTS .MOS

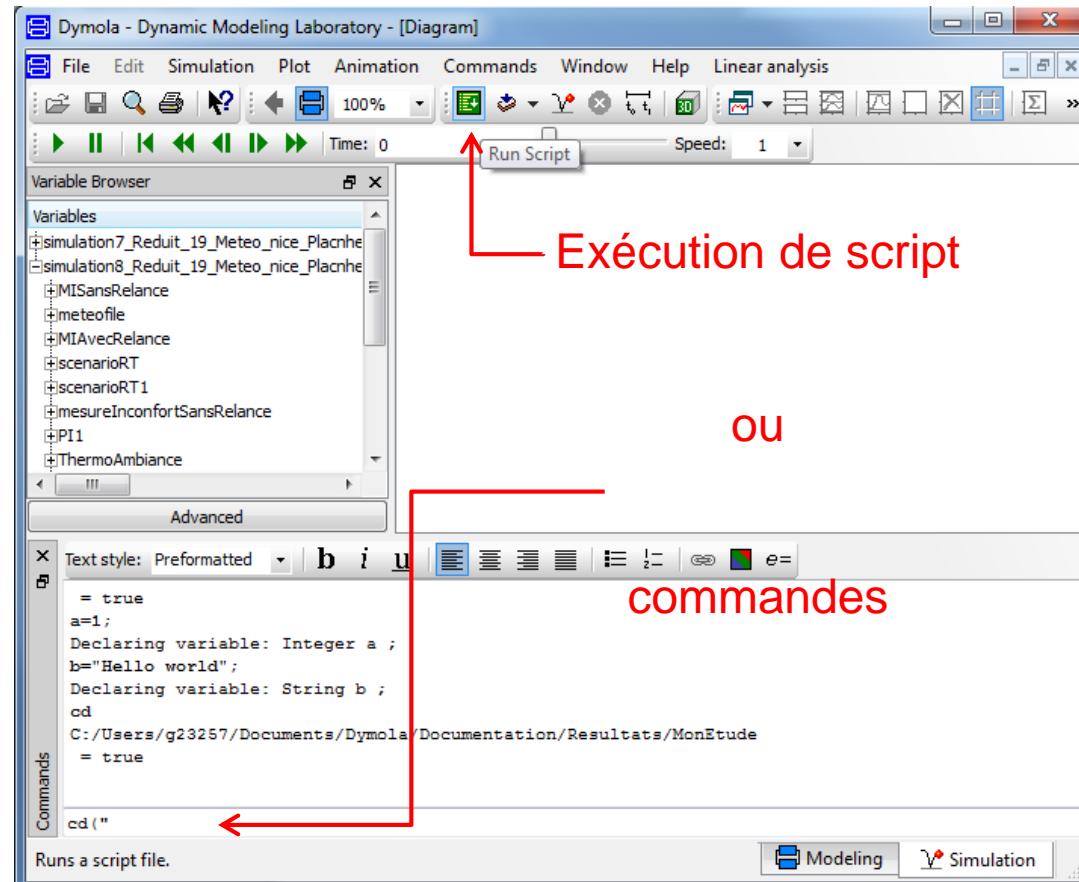
# SCRIPTS MODELICA (.MOS)

## ■ But

- Lancement automatique de simulations
- Pré/post traitement (analyse de sensibilité, optimisation...) préférer Excel, Python, Matlab...

## ■ Premier pas

- Commande par commande
- Enregistrement commandes dans un script via un éditeur de texte (bloc note, notepad++...)



# SCRIPTS MODELICA (.MOS)

## ■ Fonctions principales

- *listfunctions()*, *variables()*, *list()* : lister les fonctions / variables accessibles
- *document(« maFonction »)* : affiche la doc. de maFonction.
- *cd(répertoire)* : change le répertoire de travail.
- *Modelica.Utilities.System.command(« commande Windows »)* : permet l'exécution de commandes windows

## ■ Fonctions de simulation

- *openModel* : Ouverture du modèle ou package
- *translateModel* : Traduction en code C
- *simulateModel* : Lancement de la simulation
- *readTrajectorySize* : Lecture du nombre de pas de temps
- *readTrajectory* : Lecture des variables

## ■ Fonctions graphiques

- *plot({« var1», « var2»}, colors={{0,0,255}, {255,0,0}})* : trace les courbes

# SCRIPTS MODELICA (.MOS)

## ■ Astuces

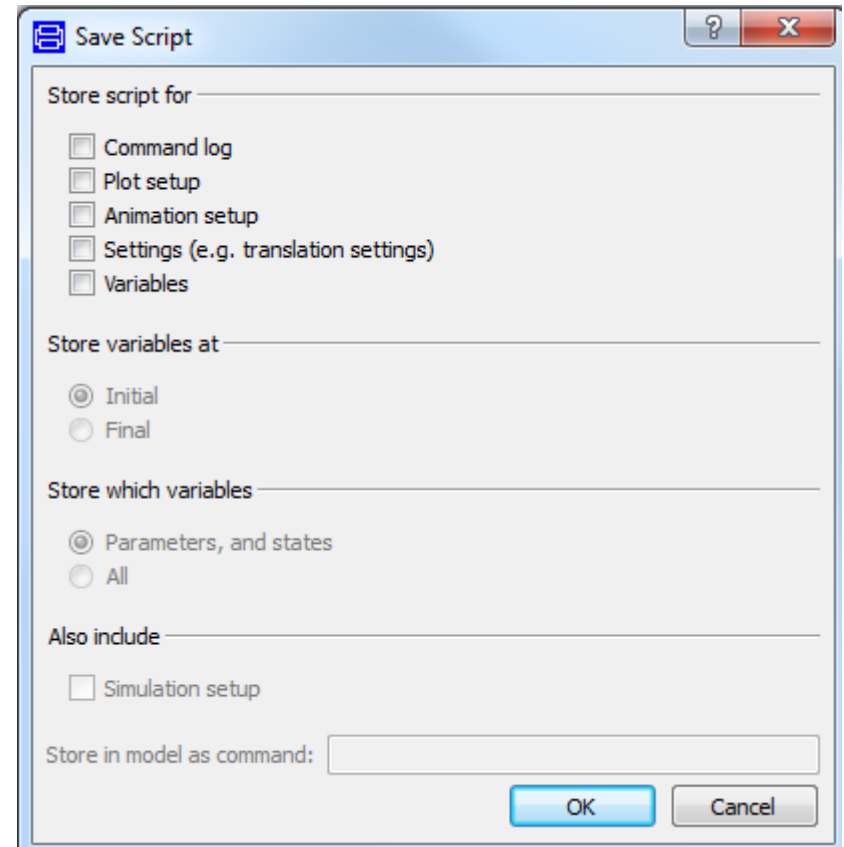
- Choisir un pas de temps équidistant et ne pas sauvegarder les événements :

*experimentSetupOutput  
(events=false,equidistant=true);*

- Voir exemples dans le répertoire  
.\Documentation\scripts\  
□ Voir Dymola User Guide Volume 1  
Scripting : p477 ⇨ p517

- Utiliser la génération automatique de script (graphs...) dans :

*File\save script...*



Ajout de d'un modèle de script pour la prochaine MàJ :

*.\Documentation\scripts\scriptTemplate.mos*