ENTERPRISE ARCHITECT

Série de Guides d'Utilisateur

SysML Simulation Paramétrique

Author: Sparx Systems
Date: 7/11/2024
Version: 17.0

Table des Matières

SysML Simulation Paramétrique

Enterprise Architect fournit une intégration avec OpenModelica et MATLAB Simulink pour support une évaluation rapide et robuste du comportement d'un modèle SysML dans différentes circonstances.

Les bibliothèques OpenModelica sont des ressources complètes qui fournissent de nombreux types, fonctions et modèles utiles. Lors de la création de modèles SysML dans Enterprise Architect , vous pouvez référencer les ressources disponibles dans ces bibliothèques.

L'intégration MATLAB d' Enterprise Architect se connecte via l'API MATLAB, permettant à vos simulations Enterprise Architect et autres scripts d'agir en fonction de la valeur de toutes les fonctions et expressions MATLAB disponibles. Vous pouvez appeler MATLAB via une classe Solveur ou exporter votre modèle vers MATLAB Simulink, Simscape et/ou Stateflow.

fonctionnalités de SysML Simulation

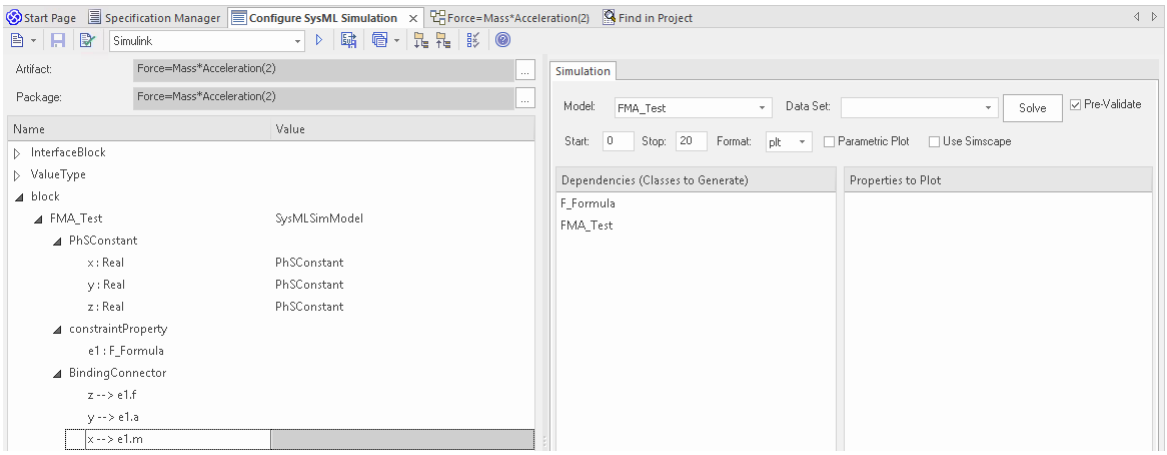
Ces sections décrivent le processus de définition d'un modèle Paramétriques , l'annotation du modèle avec des informations supplémentaires pour piloter une simulation et l'exécution d'une simulation pour générer un graphique des résultats.

Section	Description
Introduction aux modèles Paramétriques SysML	Les modèles SysML Paramétriques support l'analyse technique des paramètres critiques du système, notamment l'évaluation des mesures clés telles que les performances, la fiabilité et d'autres caractéristiques physiques. Ces modèles combinent les modèles Exigences avec les modèles de conception de système, en capturant les contraintes exécutables basées sur des relations mathématiques complexes. diagrammes Paramétriques sont diagrammes Bloc internes spécialisés qui vous aident, en tant que modélisateur, à combiner des modèles de comportement et de structure avec des modèles d'analyse technique tels que les modèles de performances, de fiabilité et de propriétés de masse. Pour plus d'informations sur les concepts des modèles SysML Paramétriques , reportez-vous au site officiel OMG SysML et à ses sources liées.
Création d'un Modèle Paramétrique	Un aperçu du développement d'éléments de modèle SysML pour la simulation, de la configuration de ces éléments dans la fenêtre Configurer Simulation SysML et de l'observation des résultats d'une simulation.
Artefact de configuration SysMLSim	Enterprise Architect vous aide à étendre l'utilité de vos modèles SysML Paramétriques en les annotant avec des informations supplémentaires qui permettent de simuler le modèle. Le modèle résultant est ensuite généré sous forme de modèle pouvant être résolu (simulé) à l'aide de MATLAB Simulink ou d'OpenModelica. Les propriétés de simulation de votre modèle sont stockées dans un artefact Simulation . Cela préserve votre modèle d'origine et supporte plusieurs simulations configurées par rapport à un seul modèle SysML. L'artefact Simulation se trouve sur la page de la boîte à outils « Artefacts ».
Interface Utilisateur	L'interface utilisateur de la simulation SysML est décrite dans la rubrique *Configurer la fenêtre Simulation SysML* .
Analyse Modèle à l'aide d'un ensemble de données	En utilisant la configuration Simulation un Bloc SysML peut avoir plusieurs jeux de données définis par rapport à lui. Cela permet d'exécuter des variations répétables sur une simulation du modèle SysML.
Support de la norme	La *norme SysPhS* est une *extension SysML pour Simulation d'interaction physique*

SysPhS	*et de flux de signaux* . Elle définit une méthode standard de traduction entre un modèle SysML et un modèle Modelica ou un modèle Simulink/Simscape, offrant ainsi une méthode plus simple basée sur un modèle pour le partage de simulations. Consultez la rubrique d'aide *Support de la norme SysPhS* .
Exemples	Pour vous aider à comprendre comment créer et simuler un modèle SysML Paramétriques , trois exemples ont été fournis pour illustrer trois domaines différents. Ces trois exemples utilisent les bibliothèques OpenModelica. Ces exemples et ce que vous pouvez en apprendre sont décrits dans la rubrique *Exemples Simulation SysML* .

Configurer Simulation SysML

La fenêtre Configurer Simulation SysML est l'interface par laquelle vous pouvez fournir des paramètres d'exécution pour exécuter la simulation d'un modèle SysML. La simulation est basée sur une configuration de simulation définie dans un élément d'artefact SysMLSimConfiguration.

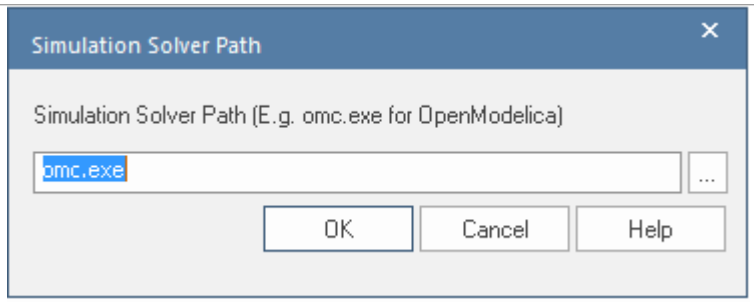


Accéder

Ruban	Simulate > Comportement du Système > Modelica/Simulink > SysMLSim Configuration Manager
Autre	Double-cliquez sur un artefact avec le stéréotype SysMLSimConfiguration.

Options de la barre d'outils

Option	Description
	Cliquez sur la flèche déroulante et sélectionnez l'une de ces options : • Sélectionner un artefact — Sélectionnez et chargez une configuration existante à partir d'un artefact avec le stéréotype SysMLSimConfiguration (si aucun n'a déjà été sélectionné) • Créer un artefact — Créez une nouvelle configuration SysMLSim ou sélectionnez et chargez un artefact de configuration existant • Sélectionner Paquetage — Sélectionnez un Paquetage pour rechercher les éléments SysML à configurer pour la simulation • Recharger — Recharger le gestionnaire de configuration avec les modifications apportées au Paquetage actuel • Configurer Solveur Simulation — Affichez la dialogue « Chemin Solveur Simulation », dans laquelle vous pouvez saisir ou rechercher le chemin d'accès au Solveur à utiliser. Pour MATLAB/Simulink, le chemin sera automatiquement détecté. Il ne doit donc être modifié qu'en cas de problème avec la détection automatique ou si plusieurs versions de MATLAB sont installées.

	
	Cliquez sur ce bouton pour enregistrer la configuration de l'artefact actuel.
	Cliquez sur cette icône pour valider spécifiquement le modèle par rapport à la configuration SysML maintenant . Les résultats de la validation s'affichent dans l'onglet « Simulation SysML » de la fenêtre Sortie système. Vous pouvez également sélectionner une option pour pré-valider automatiquement le modèle avant l'exécution de chaque simulation. Voir l'option « Pré-valider » dans le tableau *de l'onglet Simulation* .
	Cliquez sur cette icône pour développer chaque élément de la hiérarchie dans la colonne « Nom » de la fenêtre.
	Cliquez sur cette icône pour réduire tous les éléments développés dans la hiérarchie du modèle dans la colonne « Nom » de la fenêtre.
	Cliquez sur cette icône pour afficher une liste des types object qui peuvent être supprimés dans la simulation. Cliquez sur la case à cocher en regard de chaque object à supprimer ou cliquez sur le bouton Tous pour sélectionner tous les éléments à supprimer. Vous pouvez également utiliser la Barre de Filtre en haut de la colonne « Option » pour afficher uniquement les éléments ayant la lettre ou string de texte spécifiée dans le nom.
	Cliquez sur la flèche déroulante et sélectionnez l'application sous laquelle la simulation est exécuter - comme OpenModelica ou Simulink.
	Cliquez sur ce bouton pour générer, compiler et exécuter la configuration actuelle, et afficher les résultats.
	Après la simulation, le fichier de résultats est généré au format plt, mat ou csv. C'est-à-dire avec le nom de fichier : • ModelName_res.mat (la valeur par défaut pour OpenModelica) • ModelName_res.plt ou • Nom du modèle_res.csv Cliquez sur ce bouton pour spécifier un répertoire dans lequel Enterprise Architect copiera le fichier de résultats.
	Cliquez sur ce bouton pour sélectionner parmi ces options : • Exécuter le dernier code - Exécuter le code le plus récemment généré • Générer du code — Générer le code sans le compiler ni l'exécuter • Ouvrir le répertoire Simulation — Ouvrir le répertoire dans lequel le code OpenModelica ou Simulink sera généré • Modifier Gabarits — Personnalisez le code généré pour OpenModelica ou

	Simulink, à l'aide de l'éditeur de code Gabarit

Artefact Simulation et sélection Modèle

Champ	Action
Artefact	Cliquez sur l'icône  et recherchez et sélectionnez un artefact SysMLSimConfiguration existant ou créez un nouvel artefact.
Paquetage	Si vous avez spécifié un artefact SysMLSimConfiguration existant, ce champ correspond par défaut au Paquetage contenant le modèle SysML associé à cet artefact. Sinon, cliquez sur l'icône  et recherchez et sélectionnez le Paquetage contenant le modèle SysML à configurer pour la simulation. Vous devez spécifier (ou créer) l'Artefact avant de sélectionner le Paquetage .

Objets Paquetage

Ce tableau décrit les types d' object du modèle SysML qui seront répertoriés sous la colonne « Nom » de la fenêtre Configurer Simulation SysML, pour être traités dans la simulation. Chaque type object se développe pour répertorier les objets nommés de ce type et les propriétés de chaque object qui nécessitent une configuration dans la colonne « Valeur ».

De nombreux niveaux de types object , de noms et de propriétés ne nécessitent pas de configuration, de sorte que le champ « Valeur » correspondant n'accepte aucune saisie. Lorsque la saisie est appropriée et acceptée, une flèche déroulante s'affiche à l'extrémité droite du champ ; lorsque vous cliquez sur cette flèche, une courte liste de valeurs possibles s'affiche pour la sélection. Certaines valeurs (telles que « SimVariable » pour une pièce) ajoutent des couches supplémentaires de paramètres et de propriétés, où vous cliquez sur le bouton  pour, à nouveau, sélectionner et définir des valeurs pour les paramètres. Pour les jeux de données, la dialogue de saisie vous permet de saisir ou d'importer des valeurs, telles que des valeurs initiales ou par défaut ; consultez la rubrique d'aide *Analyse de Modèle utilisant Ensemble de Données* .

Type d'élément	Comportement
Type de valeur	Les éléments ValueType sont généralisés à partir d'un type primitif ou sont substitués par SysMLSimReal pour la simulation.
Bloc	Les éléments Bloc mappés aux éléments SysMLSimClass ou SysMLSimModel support la création d'ensembles de données. Si vous avez défini plusieurs ensembles de données dans une SysMLSimClass (qui peuvent être généralisés), vous devez identifier l'un d'entre eux comme étant l'ensemble de données par défaut (à l'aide de l'option de menu contextuel « Définir comme ensemble de données par défaut »). Comme un SysMLSimModel est un élément de niveau supérieur possible pour une simulation et ne sera pas généralisé, si vous avez défini plusieurs ensembles de données, l'ensemble de données à utiliser est choisi pendant la simulation.
Propriétés	La méthode préférée pour spécifier des constantes ou des variables et leurs paramètres consiste à utiliser les stéréotypes SysPhS PhSConstant et PhSVariable sur les Propriétés elles-mêmes. Le stéréotype PhSVariable possède des propriétés

	intégrées pour *isContinuous* , *isConserved* et *changeCycle* . Les Propriétés seront répertoriées sous PhSConstant ou PhSVariable et la valeur ne peut pas être modifiée. Il est également possible de définir les paramètres dans la fenêtre Configurer Simulation SysML. Dans ce cas, ils seront répertoriés sous « Propriétés ». Propriétés d'un Bloc peuvent être configurées comme des SimConstants ou des SimVariables. Pour une SimVariable, vous configurez ces attributs : • isContinuous — détermine si la valeur de la propriété varie en continu (« true », la valeur par défaut) ou discrètement (« false ») • isConserved — détermine si les valeurs de la propriété sont conservées ('true') ou non ('false', la valeur par défaut) ; lors de modélisation d'une interaction physique, les interactions incluent des échanges de substances physiques conservées telles que le courant électrique, la force ou l'écoulement d'un fluide • changeCycle — spécifie l'intervalle de temps auquel une valeur de propriété discrète change ; la valeur par défaut est « 0 » - changeCycle peut être défini sur une valeur autre que 0 uniquement lorsque isContinuous = 'faux' - La valeur de changeCycle doit être positive ou égale à 0
Port	Aucune configuration requise.
Fonction Sim	Les fonctions sont créées sous forme d'opérations dans des blocs ou des blocs de contraintes, stéréotypés comme « SimFunction ». Aucune configuration n'est requise dans la fenêtre Configurer Simulation SysML.
Généralisation	Aucune configuration requise.
Connecteur de liaison	Lie une propriété à un paramètre d'une propriété de contrainte. Aucune configuration n'est requise ; cependant, si les propriétés sont différentes, le système propose une option pour les synchroniser.
Connecteur	Connecte deux ports. Aucune configuration n'est requise dans la fenêtre Configurer Simulation SysML. Cependant, vous devrez peut-être configurer les propriétés du type de port en déterminant si l'attribut isConserved doit être défini sur « False » (pour les propriétés potentielles, afin que le couplage d'égalité soit établi) ou sur « True » (pour les propriétés de flux/conservées, afin que le couplage somme à zéro soit établi).
Bloc de contrainte	Aucune configuration requise.

Onglet Simulation

Ce tableau décrit les champs de l'onglet « Simulation » de la fenêtre Configurer Simulation SysML.

Champ	Action
Modèle	Cliquez sur la flèche déroulante et sélectionnez le nœud de niveau supérieur (un élément SysMLSimModel) pour la simulation. La liste est renseignée avec les noms des blocs définis comme nœuds de modèle de niveau supérieur.

Ensemble de données	Cliquez sur la flèche déroulante et sélectionnez l'ensemble de données pour le modèle sélectionné.
Pré-valider	Cochez cette case pour valider automatiquement le modèle avant l'exécution de chaque simulation du modèle.
Démarrer	Type le temps d'attente initial avant lequel la simulation est démarrée, en secondes (valeur par défaut est 0).
Arrêt	Type le nombre de secondes pendant lesquelles la simulation s'exécutera.
Format	Cliquez sur la flèche déroulante et sélectionnez « plt », « csv » ou « mat » comme format du fichier de résultat, qui pourrait potentiellement être utilisé par d'autres outils.
Graphique Paramétriques	• Cochez cette case pour tracer la légende A sur l'axe des Y par rapport à la légende B sur l'axe des X. • Décochez la case pour tracer les légendes sur l'axe des Y en fonction du temps sur l'axe des X Note : avec la case à cocher sélectionnée, vous devez sélectionner deux propriétés à tracer.
Utiliser Simscape	(si l'outil mathématique sélectionné est Simulink) Cochez la case si vous souhaitez également traiter la simulation dans Simscape.
Dépendances	Répertorie les types qui doivent être générés pour simuler ce modèle.
Propriétés à construire	Fournit une liste des propriétés des variables impliquées dans la simulation. Cochez la case en regard de chaque propriété à tracer.

Création d'un Modèle Paramétrique

Dans cette rubrique, nous abordons la manière dont vous pouvez développer des éléments de modèle SysML pour la simulation (en supposant que vous possédez déjà des connaissances en matière modélisation SysML), configurer ces éléments dans la fenêtre Configurer Simulation SysML et observer les résultats d'une simulation selon certaines des différentes définitions et approches modélisation . Les points sont illustrés par Instantanés de diagrammes et d'écrans issus des exemples Simulation SysML fournis dans ce chapitre.

Lors de la création d'un Modèle Paramétrique , vous pouvez appliquer l'une des trois approches pour définir les équations de contrainte :

- Définition d'équations de contrainte en ligne sur un élément Bloc
- Créer des ConstraintBlocks réutilisables et
- Utilisation des propriétés de contrainte connectées

Vous prendrez également en considération :

- Flux dans les interactions physiques
- Valeurs par défaut et valeurs initiales
- Fonctions Simulation
- Répartition de la valeur et
- Paquetages et importations

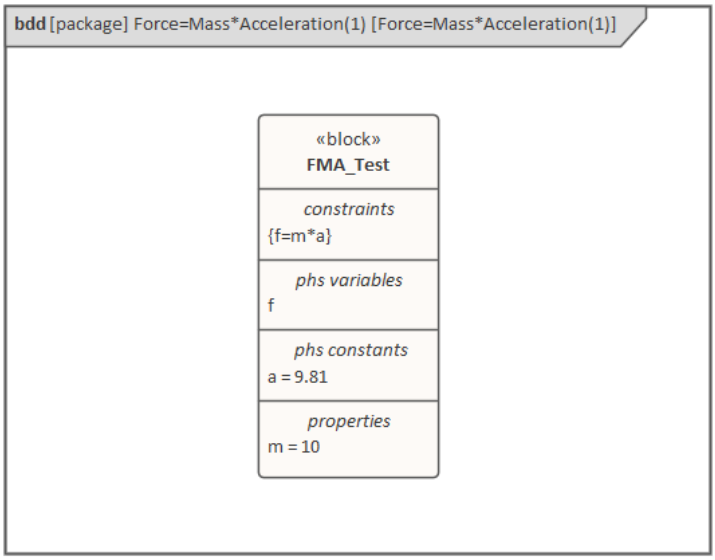
Accéder

Ruban	Simulate > Comportement du Système > Modelica/Simulink > SysMLSim Configuration Manager

Définition d'équations de contraintes en ligne sur un Bloc

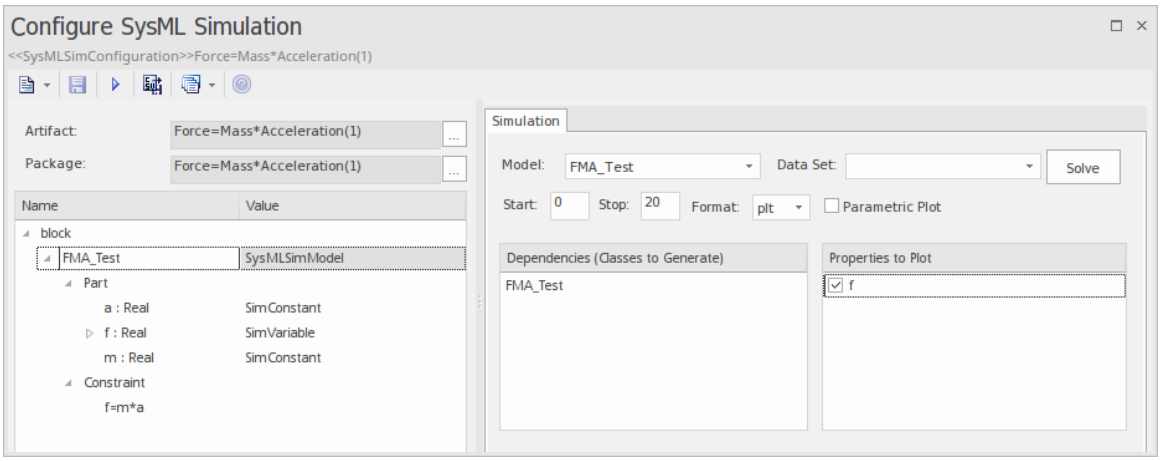
Définir des contraintes directement dans un Bloc est simple et constitue le moyen le plus simple de définir des équations de contraintes.

Dans cette figure, la contrainte 'f = m * a' est définie dans un élément Bloc .



Conseil : Vous pouvez définir plusieurs contraintes dans un même Bloc .

1. Créez un artefact de configuration SysMLSim « Force=Mass*Acceleration(1) » et pointez-le vers le Paquetage « FMA_Test ».
2. Pour « FMA_Test », dans la colonne « Valeur », définissez « SysMLSimModel ».
3. Pour les parties « a », « m » et « f », dans la colonne « Valeur » : définissez « a » et « m » sur « PhSConstant » et (éventuellement) définissez « f » sur « PhSVariable ».
4. Dans l'onglet « Simulation », dans le panneau « Propriétés à tracer », cochez la case en regard de « f ».
5. Cliquez sur le bouton Résoudre pour exécuter la simulation.

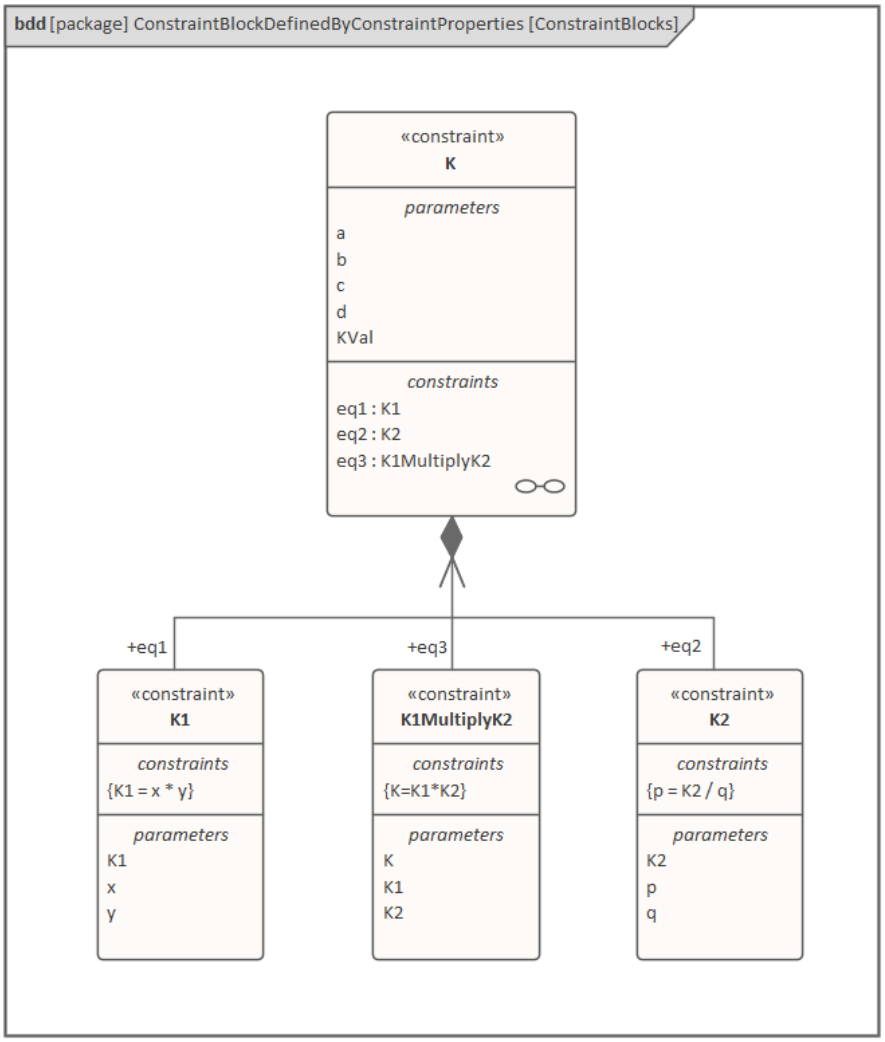


Un graphique doit être tracé avec f = 98,1 (qui vient de 10 * 9,81).

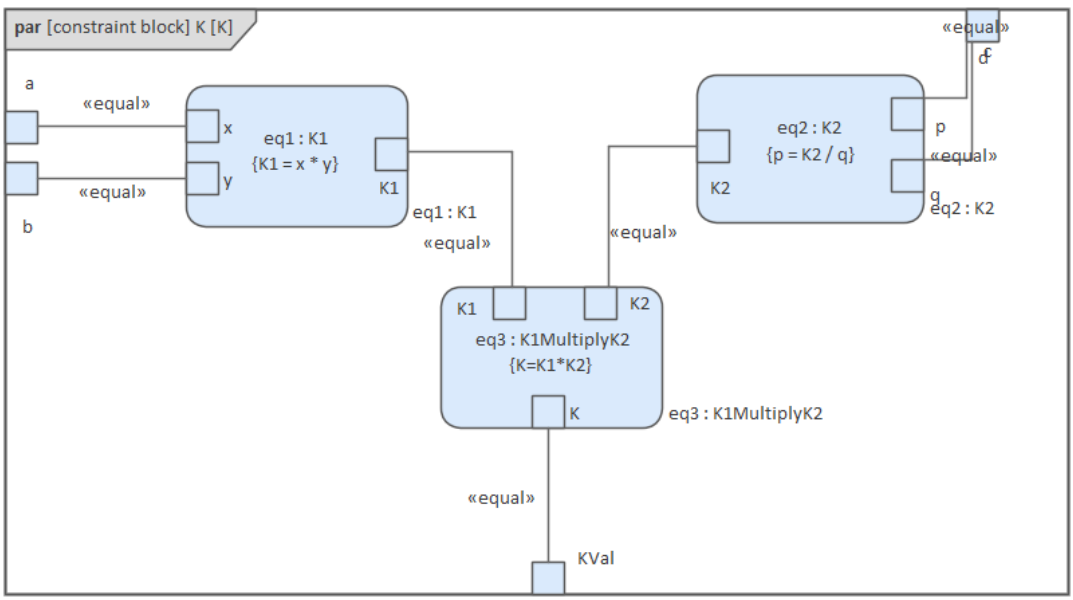
Propriétés de contrainte connectée

Dans SysML, les propriétés de contrainte existant dans ConstraintBlocks peuvent être utilisées pour offrir une plus grande flexibilité dans la définition des contraintes.

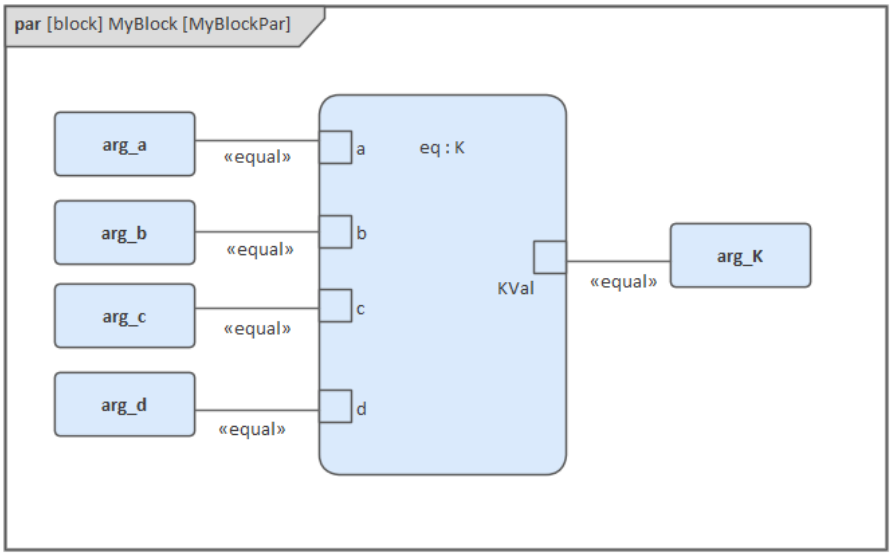
Dans cette figure, ConstraintBlock 'K' définit les paramètres 'a', 'b', 'c', 'd' et 'KVal', ainsi que trois propriétés de contrainte 'eq1', 'eq2' et 'eq3', typées respectivement 'K1', 'K2' et 'K1MultiplyK2'.



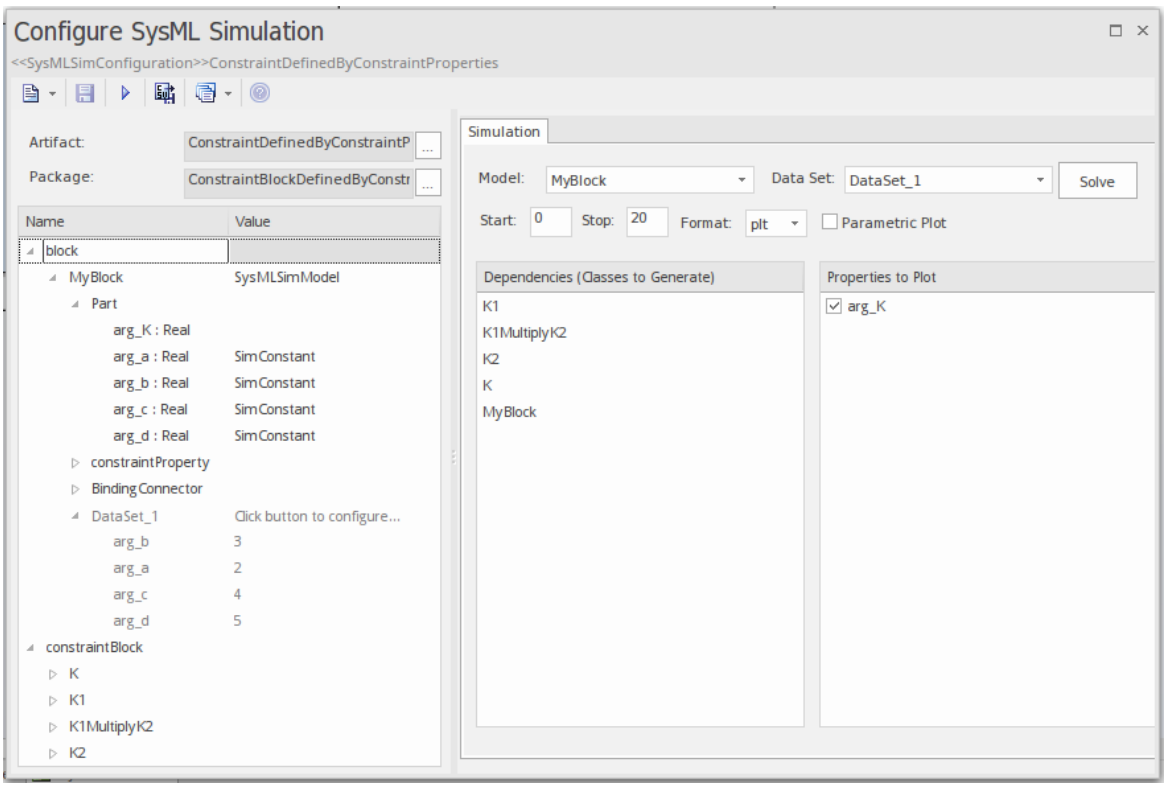
Créez un diagramme Paramétriques dans ConstraintBlock 'K' et connectez les paramètres aux propriétés de contrainte avec des connecteurs de liaison, comme indiqué :



- Créer un modèle MyBlock avec cinq Propriétés (Parties)
- Créez une propriété de contrainte « eq » pour MyBlock et affichez les paramètres
- Lier les propriétés aux paramètres



- Fournir des valeurs (arg_a = 2, arg_b = 3, arg_c = 4, arg_d = 5) dans un ensemble de données
- Dans la dialogue « Configurer Simulation SysML », définissez « Modèle » sur « MyBlock » et « Ensemble de données » sur « DataSet_1 »
- Dans le panneau « Propriétés à tracer », cochez la case en regard de « arg_K »
- Cliquez sur le bouton Résoudre pour exécuter la simulation



Le résultat 120 (calculé comme 2 * 3 * 4 * 5) sera calculé et représenté graphiquement. C'est la même chose que lorsque

nous faisons un développement avec un stylo et du papier : K = K1 * K2 = (x*y) * (p*q), puis lions avec les valeurs (2 * 3) * (4 * 5) ; nous obtenons 120.

Ce qui est intéressant ici, c'est que nous définissons intentionnellement l'équation de K2 comme étant « p = K2 / q » et cet exemple fonctionne toujours.

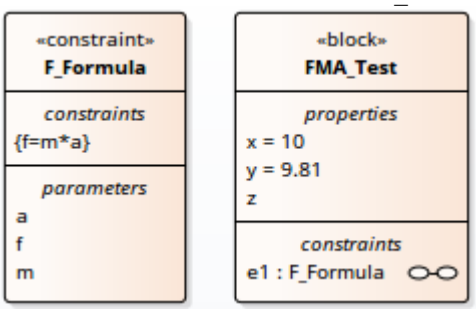
Nous pouvons facilement résoudre K2 comme étant p * q dans cet exemple, mais dans certains exemples complexes, il est extrêmement difficile de résoudre une variable à partir d'une équation ; cependant, Enterprise Architect SysMLSim peut toujours y parvenir.

En résumé, l'exemple vous montre comment définir un ConstraintBlock avec une plus grande flexibilité en construisant les propriétés de contrainte. Bien que nous n'ayons démontré qu'une seule couche dans le ConstraintBlock, ce mécanisme fonctionnera sur des modèles complexes pour un niveau d'utilisation arbitraire.

Créer des blocs de contraintes réutilisables

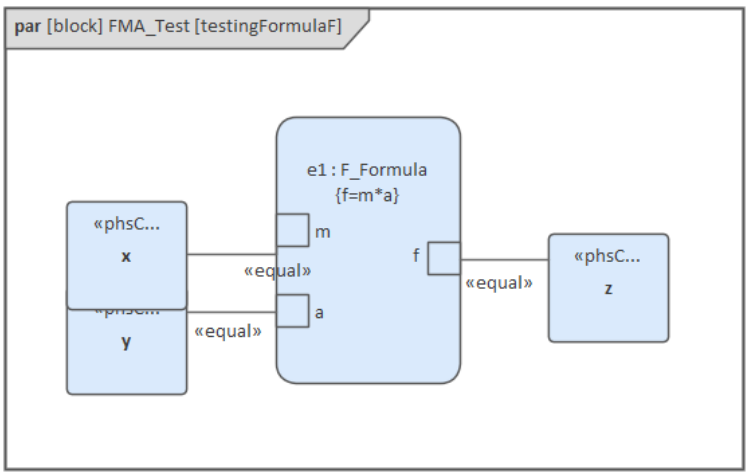
Si une équation est couramment utilisée dans plusieurs blocs, un bloc de contrainte peut être créé pour être utilisé comme propriété de contrainte dans chaque Bloc . Voici les modifications que nous apportons, sur la base de l'exemple précédent :

- Créez un élément ConstraintBlock 'F_Formula' avec trois paramètres 'a', 'm' et 'f', et une contrainte 'f = m * a'



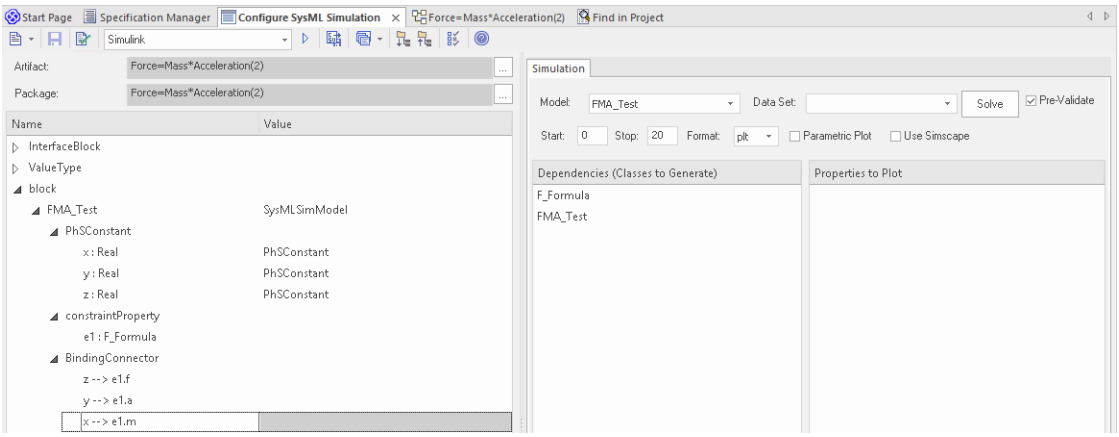
Conseil : Le type primitif 'Real' sera appliqué si les types de propriété sont vides

- Créez un Bloc « FMA_Test » avec trois propriétés « x », « y » et « z », et donnez à « x » et « y » les valeurs par défaut « 10 » et « 9,81 » respectivement
- Créer un diagramme Paramétriques dans 'FMA_Test', montrant les propriétés 'x', 'y' et 'z'
- Créez une ConstraintProperty « e1 » typée « F_Formula » et affichez les paramètres
- Dessinez les connecteurs de liaison entre « x—m », « y—a » et « f—z » comme indiqué :



- Créez un élément d'artefact SysMLSimConfiguration et configurez-le comme indiqué dans le dialogue :
  - Dans la colonne « Valeur », définissez « FMA_Test » sur « SysMLSimModel »
  - Dans la colonne « Valeur », définissez « x » et « y » sur « PhSConstant »
  - Dans le panneau « Propriétés à tracer », cochez la case en regard de « Z »

- Cliquez sur le bouton Résoudre pour exécuter la simulation



Un graphique doit être tracé avec f = 98,1 (qui vient de 10 * 9,81).

Flux dans les interactions physiques

Lors de modélisation de l'interaction physique, les échanges de substances physiques conservées telles que le courant électrique, la force, le couple et le débit doivent être modélisés comme des flux, et les variables de flux doivent être définies sur l'attribut « isConserved ».

Deux types différents de couplage sont établis par les connexions, selon que les propriétés d'écoulement sont potentielles (par défaut) ou d'écoulement (conservées) :

- Couplage d'égalité, pour les propriétés potentielles (également appelées efforts)
- Couplage somme à zéro, pour les propriétés de flux (conservées) ; par exemple, selon la loi de courant de Kirchoff dans le domaine électrique, la conservation de la charge fait que tous les flux de charge en un point sont somme à zéro

Dans le code OpenModelica généré de l'exemple « ElectricalCircuit » :

connecteur ChargePort

flux actuel i ; // le mot-clé flow sera généré si 'isConserved' = true

Tension v;

fin ChargePort;

modèle de circuit

Source source;

Résistance résistance;

Sol sol;

équation

connect(source.p, résistance.n);

connect(terre.p, source.n);

connect(résistance.p, source.n);

fin du circuit;

Chaque équation de connexion est en fait étendue à deux équations (il existe deux propriétés définies dans ChargePort), l'une pour le couplage d'égalité, l'autre pour le couplage somme à zéro :

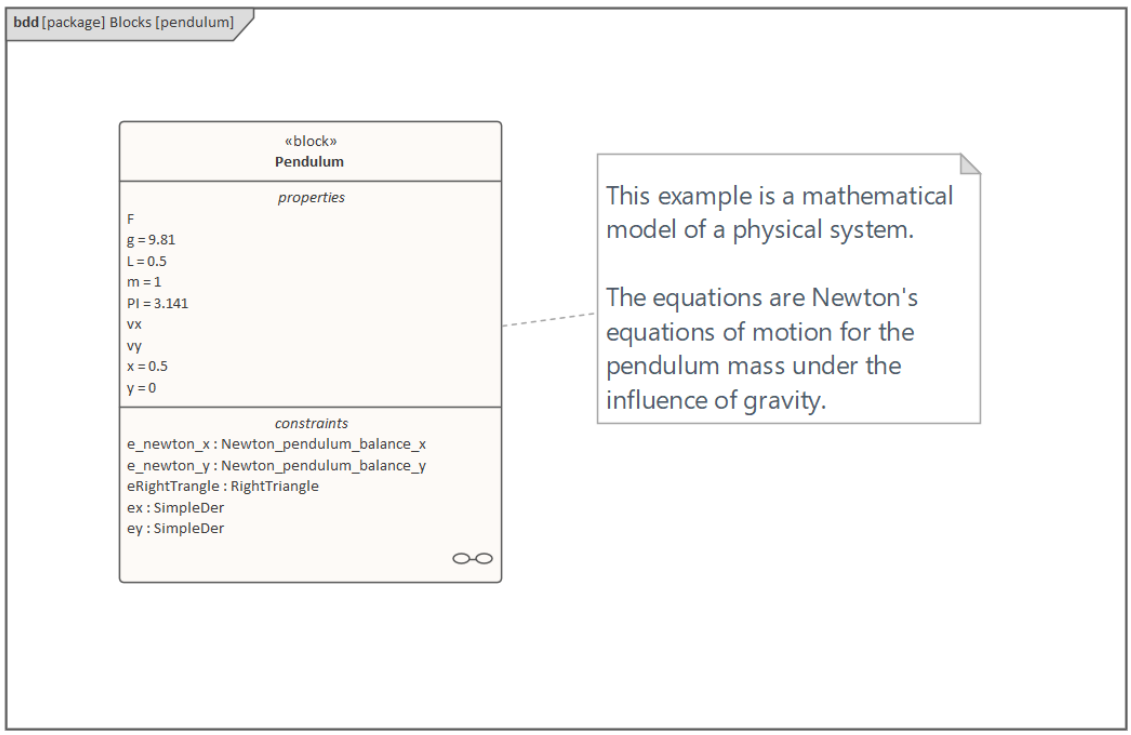
source.pv = résistance.nv;

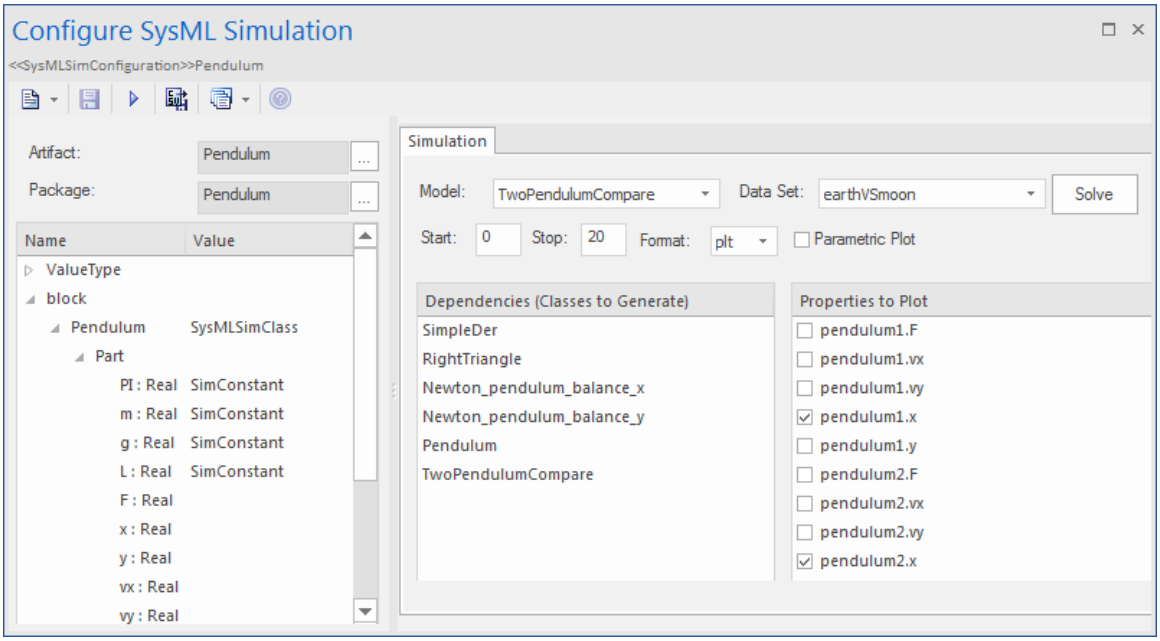
source.pi + résistance.ni = 0;

Valeur par défaut et valeurs initiales

Si des valeurs initiales sont définies dans les éléments de propriété SysML (dialogue « Propriétés » > page « Propriété » > champ « Initiale »), elles peuvent être chargées comme valeur par défaut pour un PhSConstant ou comme valeur initiale pour un PhSVariable.

Dans cet exemple de pendule, nous avons fourni des valeurs initiales pour les propriétés « g », « L », « m », « PI », « x » et « y », comme indiqué sur le côté gauche de la figure. Étant donné que « PI » (la constante mathématique), « m » (la masse du pendule), « g » (le facteur de gravité) et « L » (la longueur du pendule) ne changent pas pendant la simulation, définissez-les comme « PhSConstant ».





Le code Modelica généré ressemble à ceci :

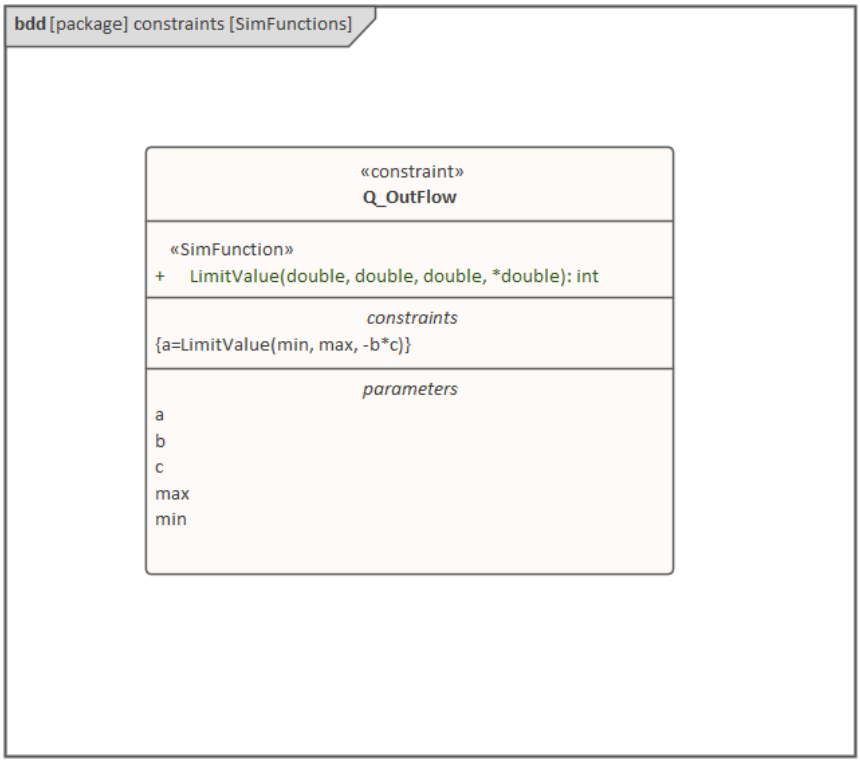
classe Pendule
paramètre PI réel = **3,141 ;**
paramètre Réel m = **1** ;
paramètre réel g = **9,81** ;
paramètre Réel L = **0,5** ;
Réel F;
Réel x **(début=0,5)** ;
Réel y **(début=0)** ;
VX réel;
Vrai vy;
.....
équation
.....
fin du pendule;

- Propriétés 'PI', 'm', 'g' et 'L' sont constantes et sont générées sous forme d'équation de déclaration
- Propriétés 'x' et 'y' sont variables ; leurs valeurs de départ sont respectivement 0,5 et 0, et les valeurs initiales sont générées sous forme de modifications

Fonctions Simulation

Une fonction Simulation est un outil utile pour écrire une logique complexe et est facile à utiliser pour les contraintes. Cette section décrit une fonction de l'exemple TankPI.

Dans le ConstraintBlock 'Q_OutFlow', une fonction 'LimitValue' est définie et utilisée dans la contrainte.



- Sur un Bloc ou un ConstraintBlock, créez une opération ('LimitValue' dans cet exemple) et ouvrez l'onglet 'Opérations' de la fenêtre Fonctionnalités
- Donnez à l'opération le stéréotype « SimFunction »
- Définissez les paramètres et définissez la direction sur « entrée/sortie »

Conseils : Plusieurs paramètres peuvent être définis comme 'out', et l'appelant récupère la valeur au format :

(out1, out2, out3) = nom_fonction(in1, in2, in3, in4, ...); //Forme d'équation

(out1, out2, out3) := function_name(in1, in2, in3, in4, ...); //Formulaire de déclaration

- Définissez le corps de la fonction dans le champ de texte de l'onglet « Code » de la fenêtre Propriétés , comme indiqué :

```
pLim :=
si p > pMax alors
pMax
sinon si p < pMin alors
pMin
autre
p;
```

Lors de la génération de code, Enterprise Architect collecte toutes les opérations stéréotypées comme « SimFunction » définies dans ConstraintBlocks et Blocks, puis génère un code ressemblant à ceci :

fonction LimitValue

entrée Réel pMin;

entrée pMax réel ;

entrée Réel p;

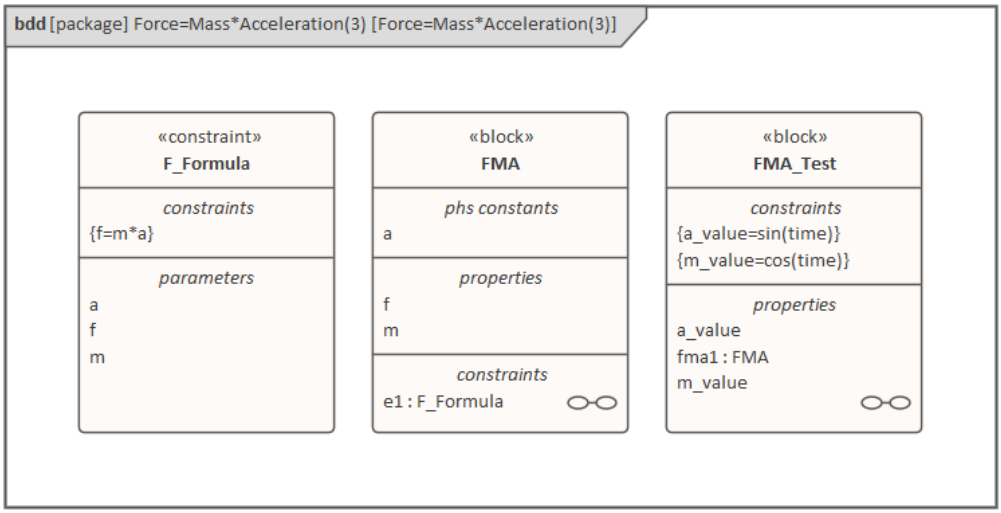
sortie Real pLim;

algorithme

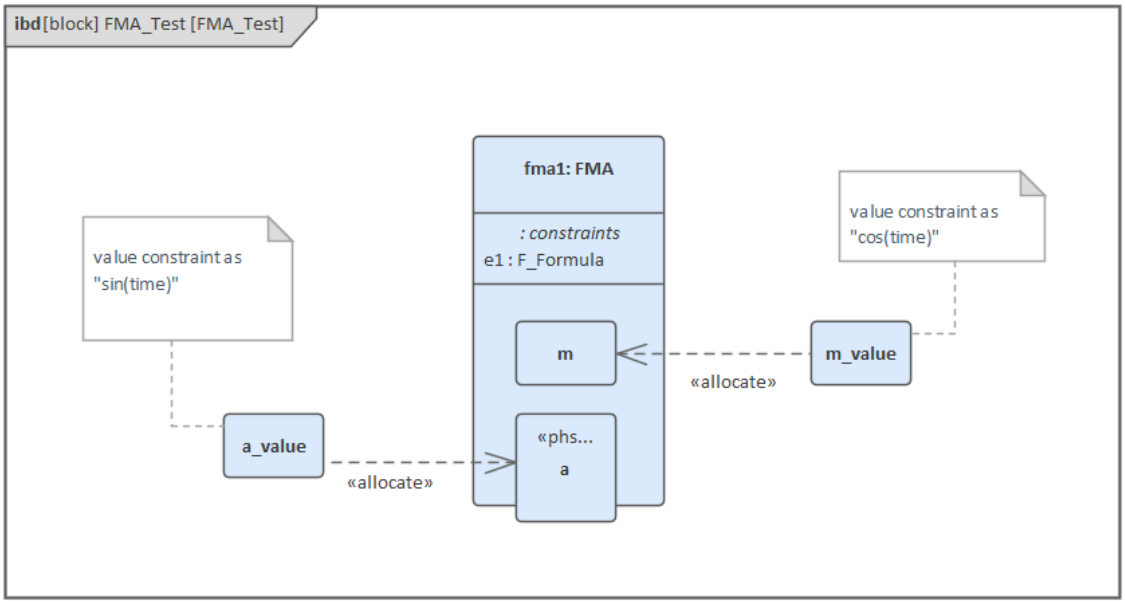
pLim :=
si p > pMax alors
pMax
sinon si p < pMin alors
pMin
autre
p;
fin de LimitValue;

Répartition de la valeur

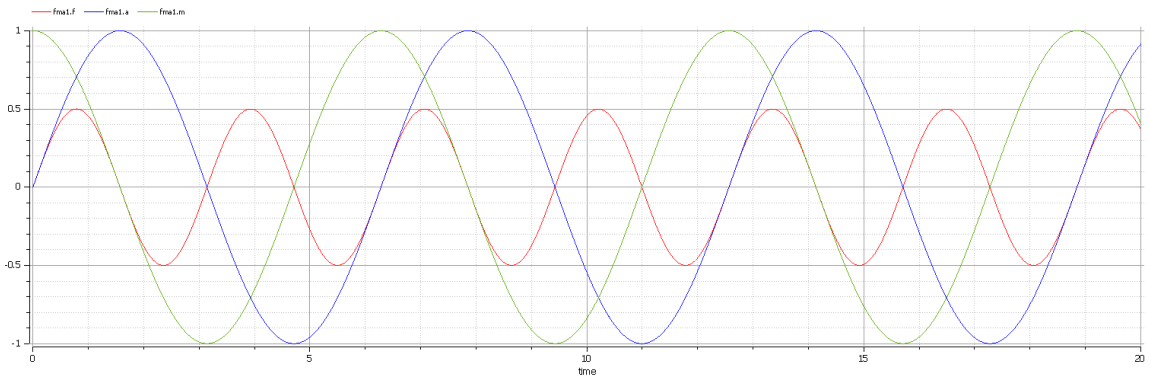
Cette figure montre un modèle simple appelé « Force = Masse * Accélération ».



- Un Bloc « FMA » est modélisé avec les propriétés « a », « f » et « m » et une constraintProperty « e1 », typée sur Bloc de contrainte « F_Formula »
- Le Bloc 'FMA' n'a aucune valeur initiale définie sur ses propriétés, et les propriétés 'a', 'f' et 'm' sont toutes variables, donc leur changement valeur dépend de l'environnement dans lequel elles sont simulées
- Créez un Bloc 'FMA_Test' en tant que SysMLSimModel et ajoutez la propriété 'fma1' pour tester le comportement du Bloc 'FMA'
- Contrainte 'a_value' à être ' sin (time)'
- Contrainte 'm_value' à être ' cos (temps)'
- Dessinez des connecteurs d'allocation pour allouer des valeurs de l'environnement au modèle « FMA »



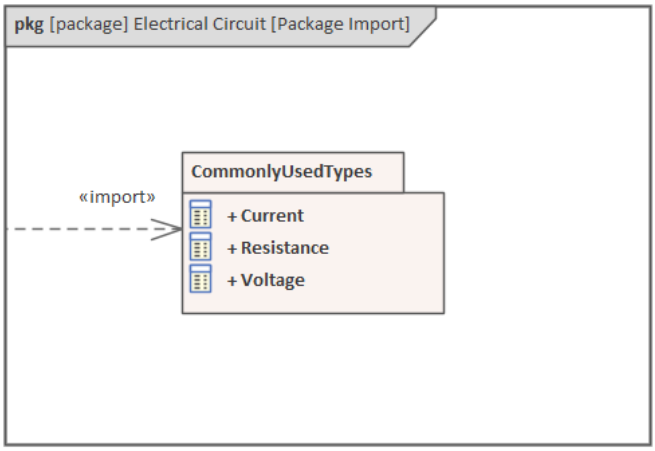
- Cochez les cases « Propriétés à tracer » pour « fma1.a », « fma1.m » et « fma1.f »
- Cliquez sur le bouton Résoudre pour simuler le modèle



Paquetages et importations

L'artefact SysMLSimConfiguration collecte les éléments (tels que les blocs, les blocs de contrainte et les types de valeur) d'un Paquetage . Si la simulation dépend d'éléments qui ne sont pas détenus par ce Paquetage , tels que des bibliothèques réutilisables, Enterprise Architect fournit un connecteur d'importation entre les éléments Paquetage pour répondre à cette exigence.

Dans l'exemple de circuit électrique, l'artefact est configuré sur le Paquetage « ElectricalCircuit », qui contient presque tous les éléments nécessaires à la simulation. Cependant, certaines propriétés sont typées sur des types valeur tels que « Tension », « Courant » et « Résistance », qui sont couramment utilisés dans plusieurs modèles SysML et sont donc placés dans un Paquetage appelé « CommonlyUsedTypes » en dehors des modèles SysML individuels. Si vous importez ce Paquetage à l'aide d'un connecteur d'importation, tous les éléments du Paquetage importé apparaîtront dans le gestionnaire de configuration SysMLSim.


pkg [package] Electrical Circuit [Package Import]
«import»
CommonlyUsedTypes
+ Current
+ Resistance
+ Voltage

Analyse de Modèle utilisant Ensemble de Données

Chaque Bloc SysML utilisé dans un modèle Paramétriques peut, dans la configuration Simulation , avoir plusieurs jeux de données définis par rapport à lui. Cela permet des variations de simulation reproductibles en utilisant le même modèle SysML.

Un Bloc peut être typé en tant que SysMLSimModel (un nœud de niveau supérieur qui ne peut pas être généralisé ou faire partie d'une composition) ou en tant que SysMLSimClass (un élément de niveau inférieur qui peut être généralisé ou faire partie d'une composition). Lorsque vous exécutez une simulation sur un élément SysMLSimModel, si vous avez défini plusieurs jeux de données, vous pouvez spécifier le jeu de données à utiliser. Cependant, si une SysMLSimClass dans la simulation comporte plusieurs jeux de données, vous ne pouvez pas sélectionner celui à utiliser pendant la simulation et devez donc identifier un jeu de données comme jeu de données par défaut pour cette classe.

Accéder

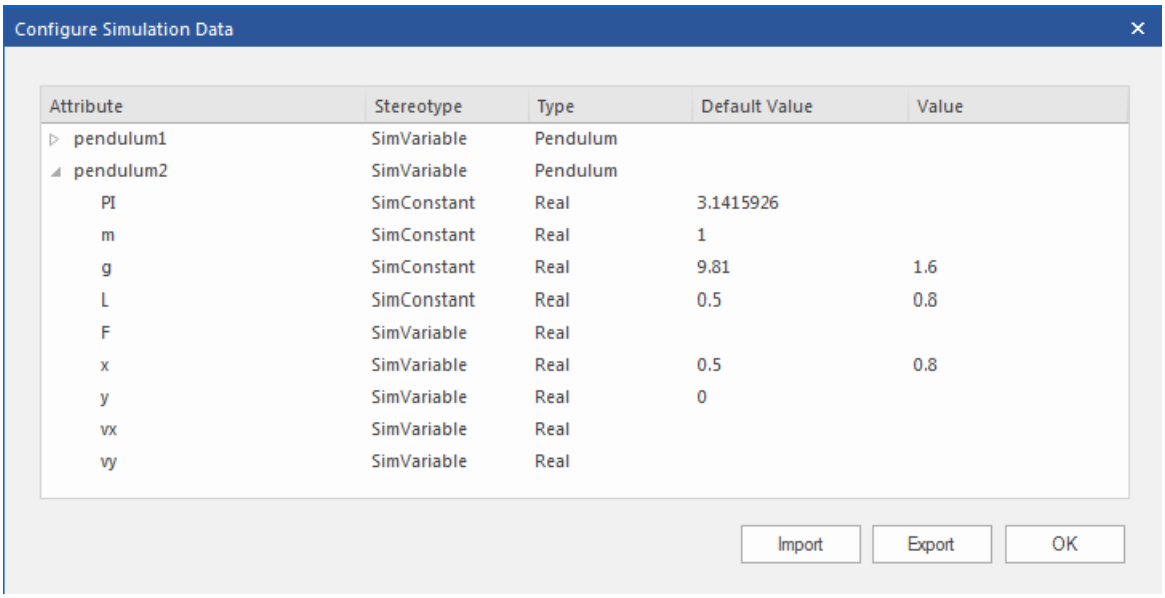
Ruban	Simuler > Comportement du Système > Modelica/Simulink > Gestionnaire de configuration SysMLSim > dans le groupe « bloc » > Colonne Nom > Menu contextuel sur l'élément bloc > Créer un jeu de données Simulation

Gestion des jeux de données

Tâche	Action
Créer	Pour créer un nouveau jeu de données, cliquez-droit sur le nom d'un Bloc et sélectionnez l'option « Créer un jeu de données Simulation ». Le jeu de données est ajouté à la fin de la liste des composants sous le nom Bloc . Cliquez sur le bouton  pour configurer le jeu de données dans la dialogue « Configurer les données Simulation » (voir le tableau *Configurer les données Simulation*).
Double	Pour dupliquer un jeu de données existant comme base pour la création d'un nouveau jeu de données, cliquez-droit sur le nom du jeu de données et sélectionnez l'option « Dupliquer ». Le jeu de données dupliqué est ajouté à la fin de la liste des composants sous le nom Bloc . Cliquez sur le bouton  pour modifier le jeu de données dans la dialogue « Configurer les données Simulation » (voir le tableau *Configurer les données Simulation*).
Supprimer	Pour supprimer un jeu de données qui n'est plus nécessaire, cliquez-droit sur le jeu de données et sélectionnez l'option « Supprimer le jeu de données ».
Définir par défaut	Pour définir le jeu de données par défaut utilisé par une SysMLSimClass lorsqu'elle est utilisée comme type de propriété ou héritée (et lorsqu'il existe plusieurs jeux de données), cliquez-droit sur le jeu de données et sélectionnez l'option « Définir par défaut ». Le nom du jeu de données par défaut est mis en surbrillance en gras. Les propriétés utilisées par un modèle utiliseront cette configuration par défaut, sauf si le modèle les remplace explicitement.

Configurer les données Simulation

Cette dialogue est principalement destinée à l'information. La seule colonne dans laquelle vous pouvez directement ajouter ou modifier des données est la colonne « Valeur ».



Colonne	Description
Attribut	La colonne « Attribut » fournit une arborescence de toutes les propriétés du Bloc en cours d'édition.
Stéréotype	La colonne « Stéréotype » identifie, pour chaque propriété, si elle a été configurée pour être une constante pendant toute la durée de la simulation ou une variable dont la valeur est censée changer au fil du temps.
Type	La colonne « Type » décrit le type utilisé pour la simulation de cette propriété. Il peut s'agir soit d'un type primitif (tel que « Réel »), soit d'une référence à un Bloc contenu dans le modèle. Propriétés référençant des Blocs afficheront les propriétés enfants spécifiées par le Bloc référencé en dessous d'elles.
Valeur par défaut	La colonne « Valeur par défaut » indique la valeur qui sera utilisée dans la simulation si aucune substitution n'est fournie. Cela peut provenir du champ « Valeur initiale » dans le modèle SysML ou du jeu de données par défaut du type parent.
Valeur	La colonne « Valeur » vous permet de remplacer la valeur par défaut pour chaque valeur primitive.
Exporter / Importer	Cliquez sur ces boutons pour modifier les valeurs de l'ensemble de données actuel à l'aide d'une application externe telle qu'une feuille de calcul, puis les réimporter dans la liste.

Exemples Simulation SysML

Cette section fournit un exemple concret pour chacune de ces étapes : création d'un modèle SysML pour un domaine, simulation de celui-ci et évaluation des résultats de la simulation. Les exemples appliquent les informations abordées dans les rubriques précédentes.

Exemples

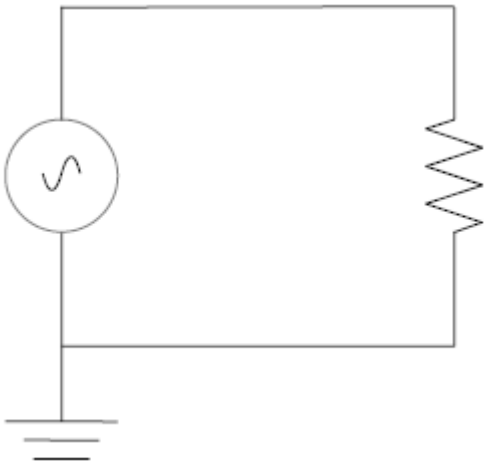
Modèle	Description
Exemple Simulation de circuit électrique	Le premier exemple concerne la simulation du chargement d'un circuit électrique. L'exemple part d'un diagramme de circuit électrique et le convertit en un modèle paramétrique. Le modèle est ensuite simulé et la tension aux bornes source et cible d'une résistance est évaluée et comparée aux valeurs attendues.
Exemple Simulation d'oscillateur masse-ressort-amortisseur	Le deuxième exemple utilise un modèle physique simple pour démontrer le comportement oscillatoire d’un système masse-ressort-amortisseur.
Régulateur de pression du réservoir d'eau	Le dernier exemple montre les niveaux d'eau de deux réservoirs où l'eau est distribuée entre eux. Nous simulons d'abord un système bien équilibré, puis nous simulons un système où l'eau débordera du deuxième réservoir.

Exemple Simulation de circuit électrique

Pour cet exemple, nous parcourons la création d'un modèle SysML Paramétriques pour un circuit électrique simple, puis utilisons une simulation paramétrique pour prédire et cartographier le comportement de ce circuit.

Diagramme de circuit

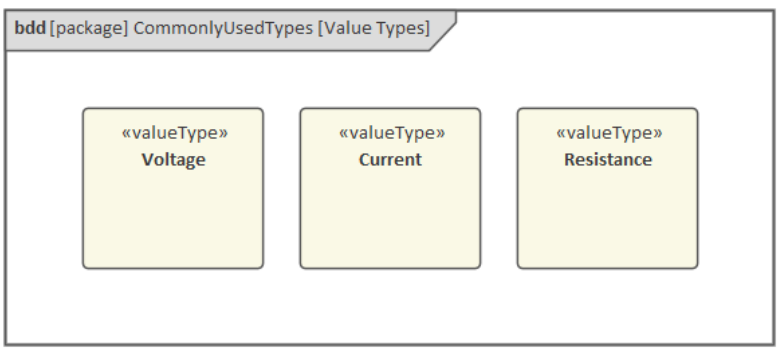
Le circuit électrique que nous allons modéliser, montré ici, utilise une notation de circuit électrique standard.

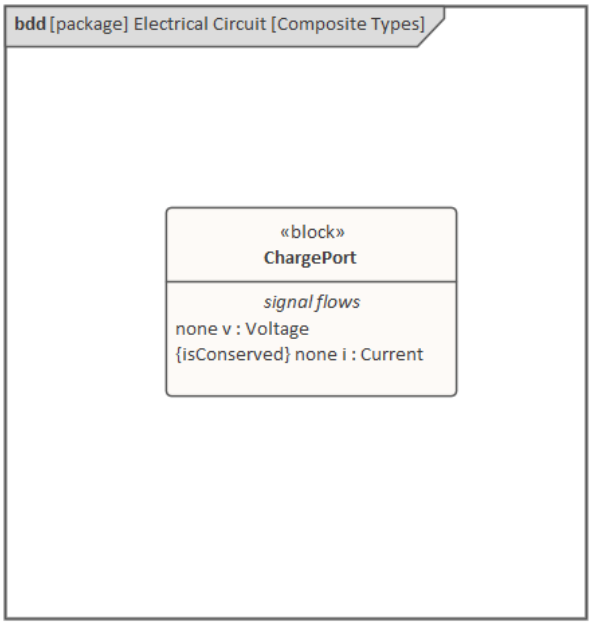


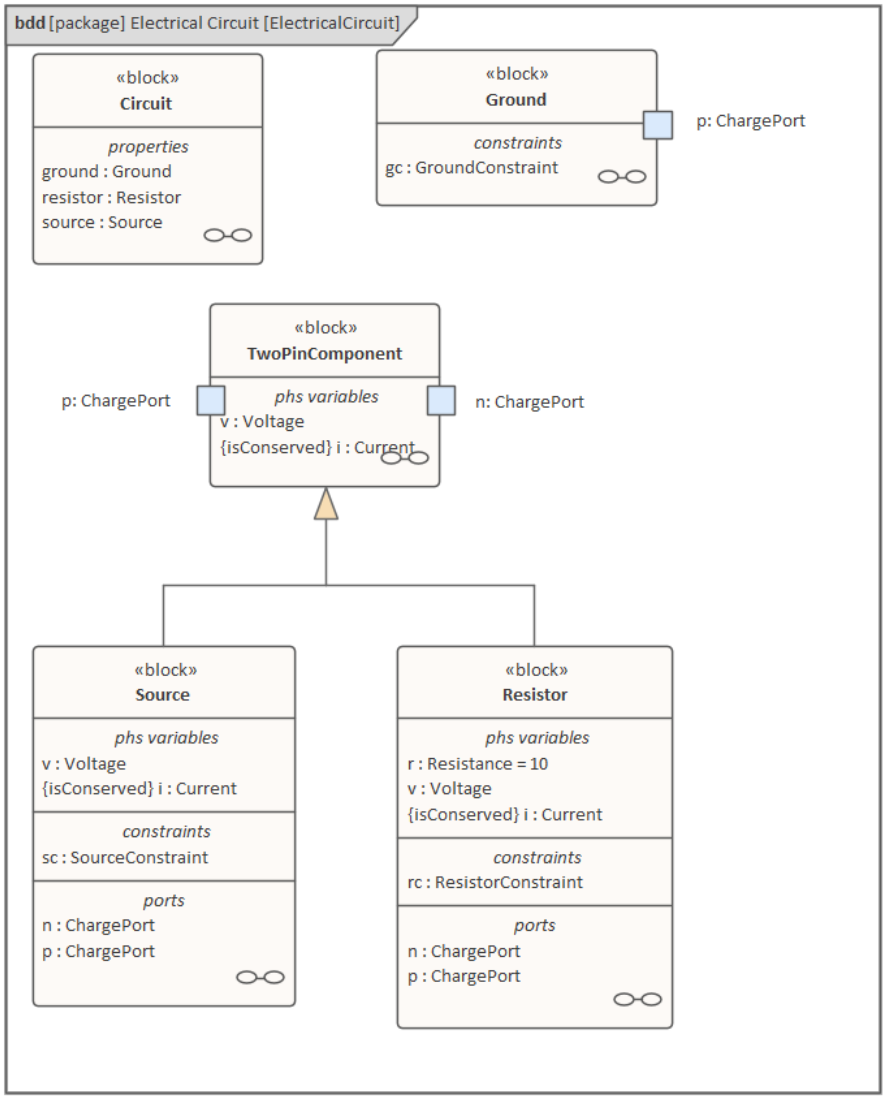
Le circuit comprend une source d'alimentation CA, une terre et une résistance, reliées entre elles par un fil électrique.

Créer Modèle SysML

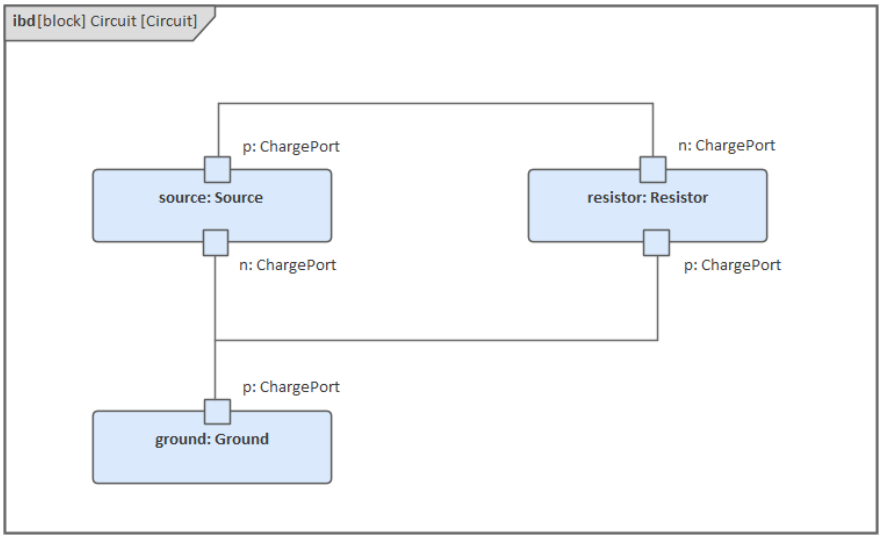
Ce tableau montre comment nous pouvons créer un modèle SysML complet pour représenter le circuit, en commençant par les types de niveau le plus bas et en construisant le modèle une étape à la fois.

Composant	Action
Types	Définissez les types de valeur pour la tension, le courant et la résistance. Le type d'unité et de quantité n'est pas important pour les besoins de la simulation, mais il serait défini lors de la définition d'un modèle SysML complet. Ces types seront généralisés à partir du type primitif « Réel ». Dans d'autres modèles, vous pouvez choisir de mapper un Type de valeur à un type de simulation correspondant distinct du modèle.  De plus, définissez un type composite (Bloc) appelé ChargePort, qui inclut des propriétés pour le courant et la tension. Ce type nous permet de représenter l'énergie

	électrique au niveau des connecteurs entre les composants. 
Blocs	Dans SysML, le circuit et chacun des composants seront représentés sous forme de blocs. Dans un Interrompre lorsqu'une Variable Change de Valeur (BDD), créez un Circuit Bloc . Le circuit comporte trois parties : une source, une masse et une résistance. Ces parties sont de types différents, avec des comportements différents. Créez un Bloc pour chacun des types de composants. Les trois composants du Bloc de circuit sont connectés via des ports, qui représentent pins électriques. La source et la résistance ont une broche positive et une broche négative. La terre n'a qu'une seule broche, qui est positive. L'électricité (charge électrique) est transmise via les pins . Créez un bloc abstrait « TwoPinComponent » avec deux ports (pins). Les deux ports sont nommés « p » (positif) et « n » (négatif), et ils sont de type ChargePort. Cette figure montre le BDD, avec les blocs Circuit, Ground, TwoPinComponent, Source et Resistance.

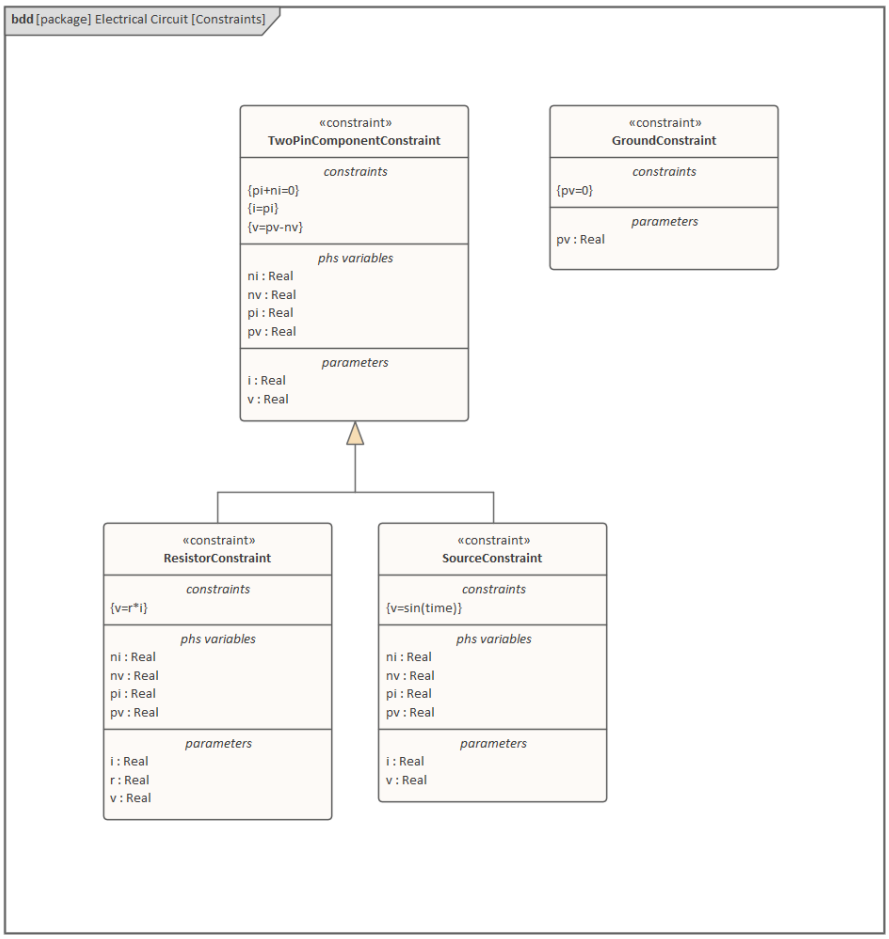


Structure interne	Créez un Diagramme Interne de Bloc (IBD) pour Circuit. Ajoutez les propriétés de la Source, de la Résistance et de la Terre, typées par les Blocs correspondants. Connectez les Ports avec des connecteurs. La broche positive de la Source est connectée à la broche négative de la Résistance. La broche positive de la Résistance est connectée à la broche négative de la Source. La Terre est également connectée à la broche négative de la Source.



Notez que cela suit la même structure que le diagramme de circuit d'origine, mais les symboles de chaque composant ont été remplacés par des propriétés typées par les blocs que nous avons définis.

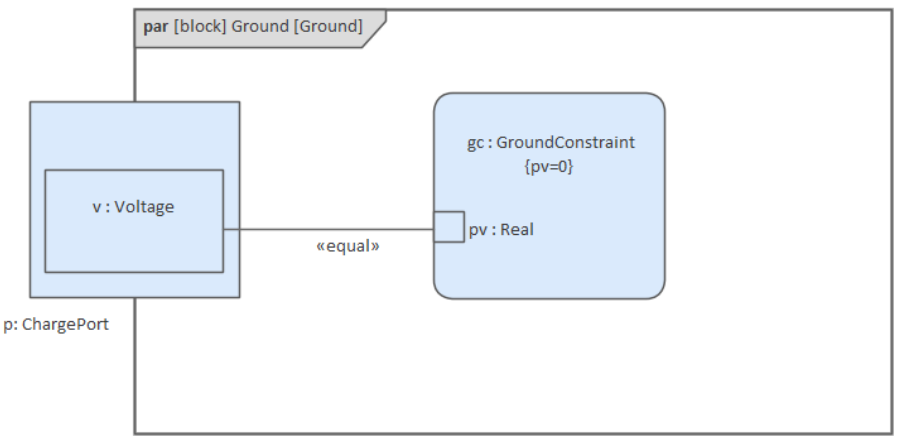
Contraintes	Les équations définissent des relations mathématiques entre des propriétés numériques. Dans SysML, les équations sont représentées sous forme de contraintes dans des ConstraintBlocks. Les paramètres de ConstraintBlocks correspondent aux PhSVariables et PhSConstants des Blocks ('i', 'v', 'r' dans cet exemple), ainsi qu'aux PhSVariables présents dans le type des Ports ('pv', 'pi', 'nv', 'ni' dans cet exemple). Créez un bloc de contraintes « TwoPinComponentConstraint » pour définir les paramètres et les équations communs aux sources et aux résistances. Les équations doivent indiquer que la tension du composant est égale à la différence entre les tensions aux pins positive et négative. Le courant du composant est égal au courant traversant la broche positive. La somme des courants traversant les deux pins doit être égale à zéro (l'une est la négative de l'autre). La contrainte Ground indique que la tension à la broche Ground est nulle. La contrainte Source définit la tension comme une onde sine avec le temps de simulation du courant comme paramètre. Cette figure montre comment ces contraintes sont rendues dans un BDD.



Fixations

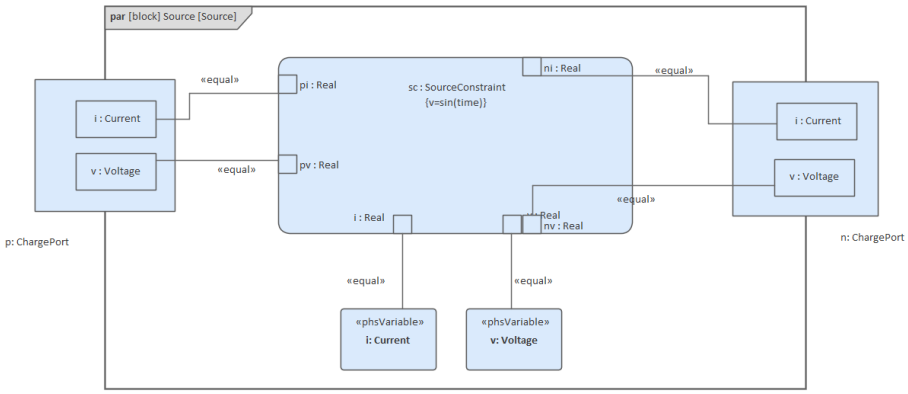
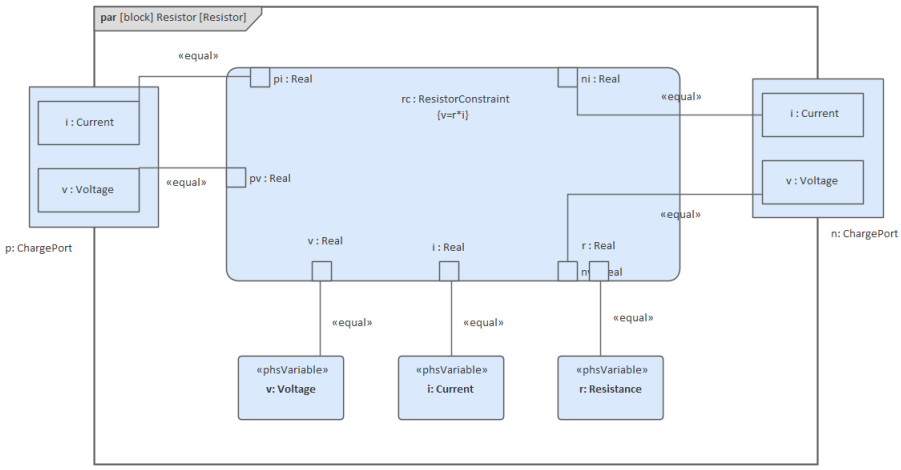
Les valeurs des paramètres de contrainte sont assimilées à des valeurs variables et constantes avec des connecteurs de liaison. Créez des propriétés de contrainte sur chaque Bloc (propriétés typées par ConstraintBlocks) et liez les variables et constantes Bloc aux paramètres de contrainte pour appliquer la contrainte au Bloc . Ces figures montrent les liaisons pour la terre, la source et la résistance respectivement.

Pour la contrainte Ground, liez gc.pv à pv



Pour la contrainte Source, liez :

- sc.pi vers pi
- sc.pv vers pv

	• sc.v à v • sc.i à i • sc.ni à ni et • sc.nv à nv  Pour la contrainte de résistance, liez : • rc.pi vers pi • rc.pv vers pv • rc.v à v • rc.i à i • rc.ni à ni • rc.nv à nv et • rc.r à r 

Configurer le comportement Simulation

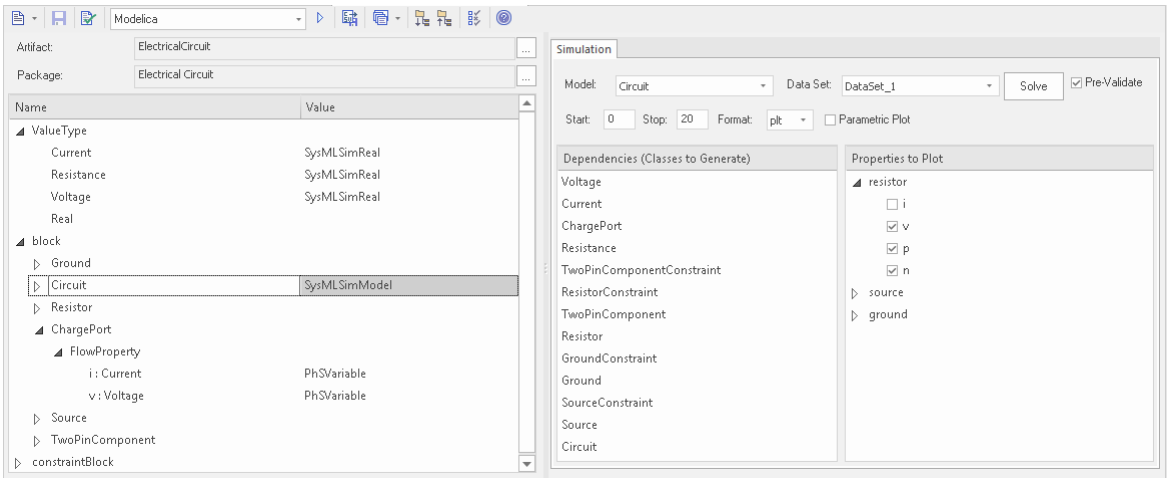
Ce tableau montre les étapes détaillées de la configuration de SysMLSim.

Étape	Action
Artefact de configuration SysMLSim	• Sélectionnez 'Simulate > Comportement du Système > Modelica/Simulink > SysMLSim Configuration Manager'.

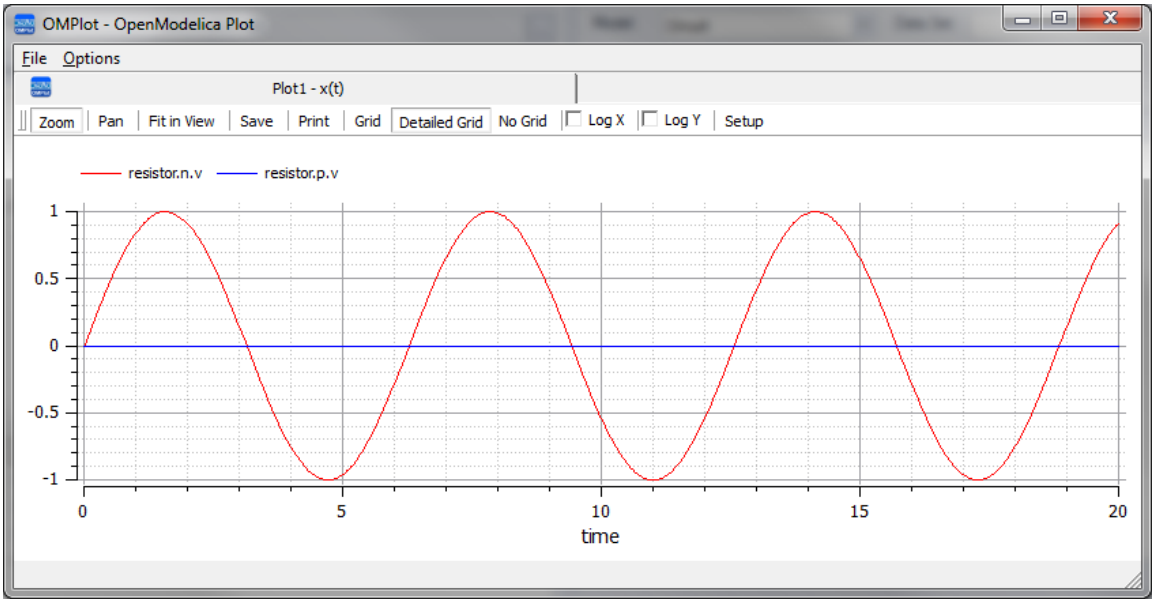
	• Dans la liste déroulante de la première icône de la barre d'outils, sélectionnez « Créer un artefact » et créez l'élément artefact • Sélectionnez le Paquetage qui possède ce modèle SysML
Créer des éléments racines dans Configuration Manager	• Type de valeur • Bloc • Bloc de contrainte
Substitution de type de valeur	Développez ValueType et pour chacun des paramètres Courant, Résistance et Tension, sélectionnez « SysMLSimReal » dans la zone de liste déroulante « Valeur ».
Définir la propriété comme flux	• Développez « block » dans ChargePort | FlowProperty | i : Current et sélectionnez « SimVariable » dans la zone de liste déroulante « Value » • Pour « SysMLSimConfiguration », cliquez sur le bouton  pour ouvrir la dialogue « Configurations des éléments » • Réglez « isConserved » sur « True »
Modèle SysMLSim	C'est le modèle que nous voulons simuler : définissez le Bloc 'Circuit' sur 'SysMLSimModel'.

Exécuter Simulation

Dans la page « Simulation », cochez les cases « résistance.n » et « résistance.p » pour le traçage et cliquez sur le bouton Résoudre.



Les deux légendes « resistor.n » et « resistor.p » sont tracées, comme indiqué.

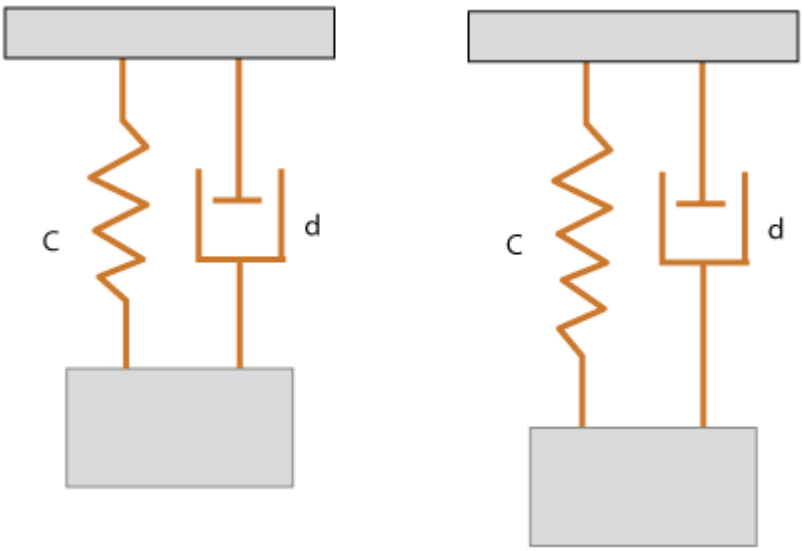

OMPlot - OpenModelica Plot
File Options
Plot1 - x(t)
Zoom Pan Fit in View Save Print Grid Detailed Grid No Grid Log X Log Y Setup
resistor.n.v
resistor.p.v
1
0.5
0
-0.5
-1
0
5
10
15
20
time

Exemple Simulation d'oscillateur masse-ressort-amortisseur

Dans cette section, nous allons parcourir la création d'un modèle paramétrique SysML pour un oscillateur simple composé d'une masse, d'un ressort et d'un amortisseur, puis utiliser une simulation paramétrique pour prédire et tracer le comportement de ce système mécanique. Enfin, nous effectuons une analyse hypothétique en comparant deux oscillateurs fournis avec des valeurs de paramètres différentes via des ensembles de données.

Système en cours de modélisation

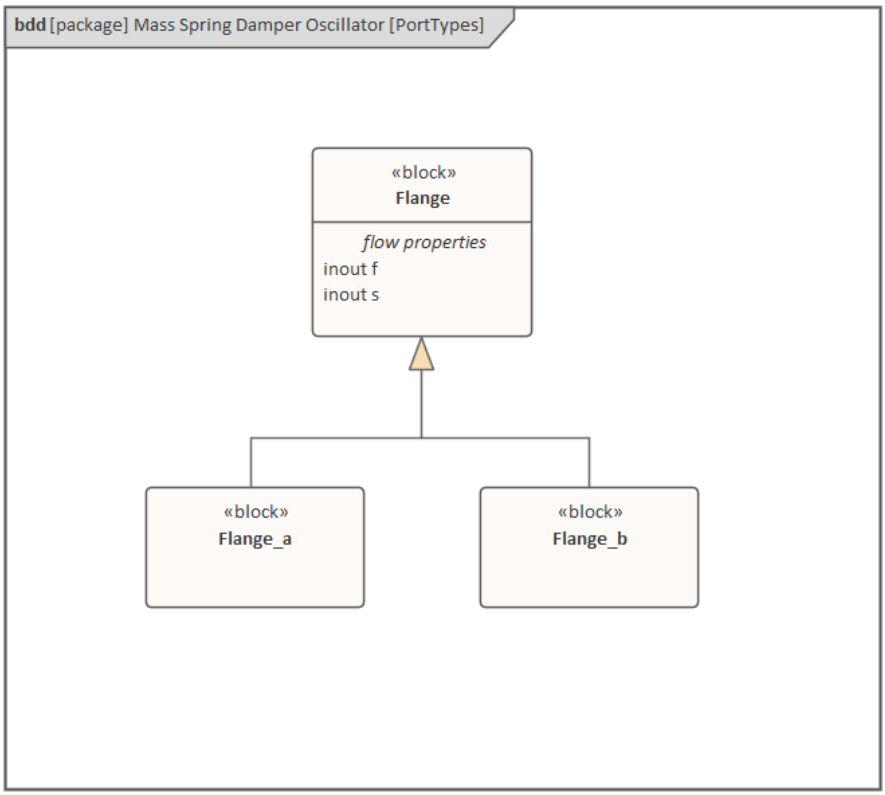
Une masse est suspendue à un ressort et à un amortisseur. Le premier état représenté ici représente le point initial à l'instant = 0, juste au moment où la masse est relâchée. Le deuxième état représente le point final lorsque le corps est au repos et que les forces du ressort sont en équilibre avec la gravité.

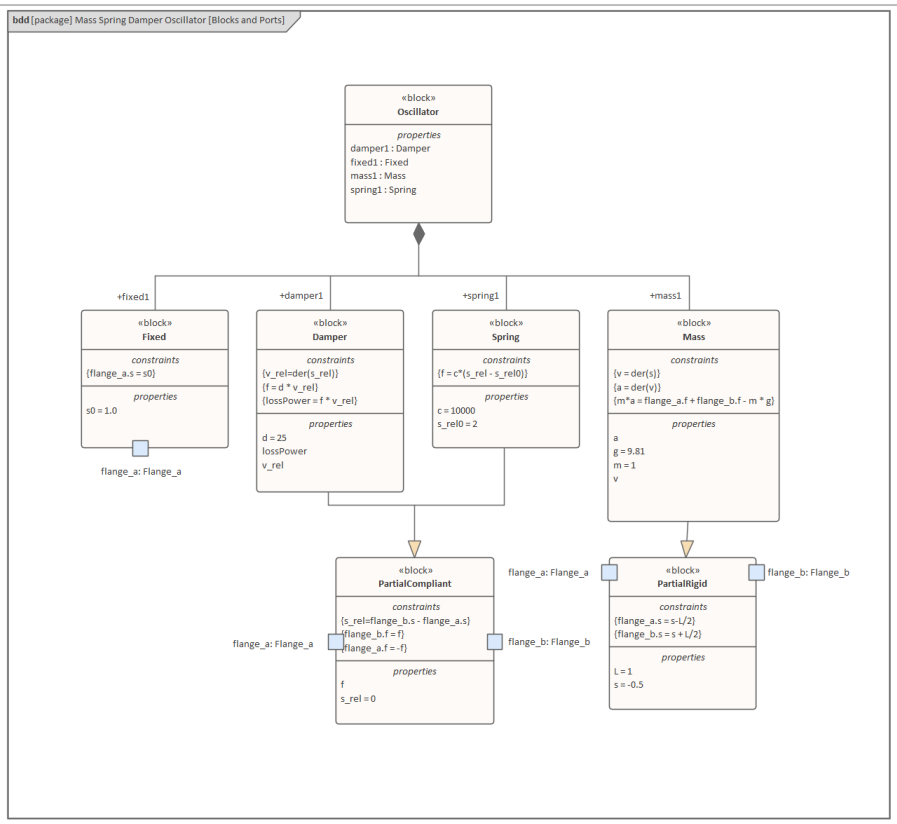


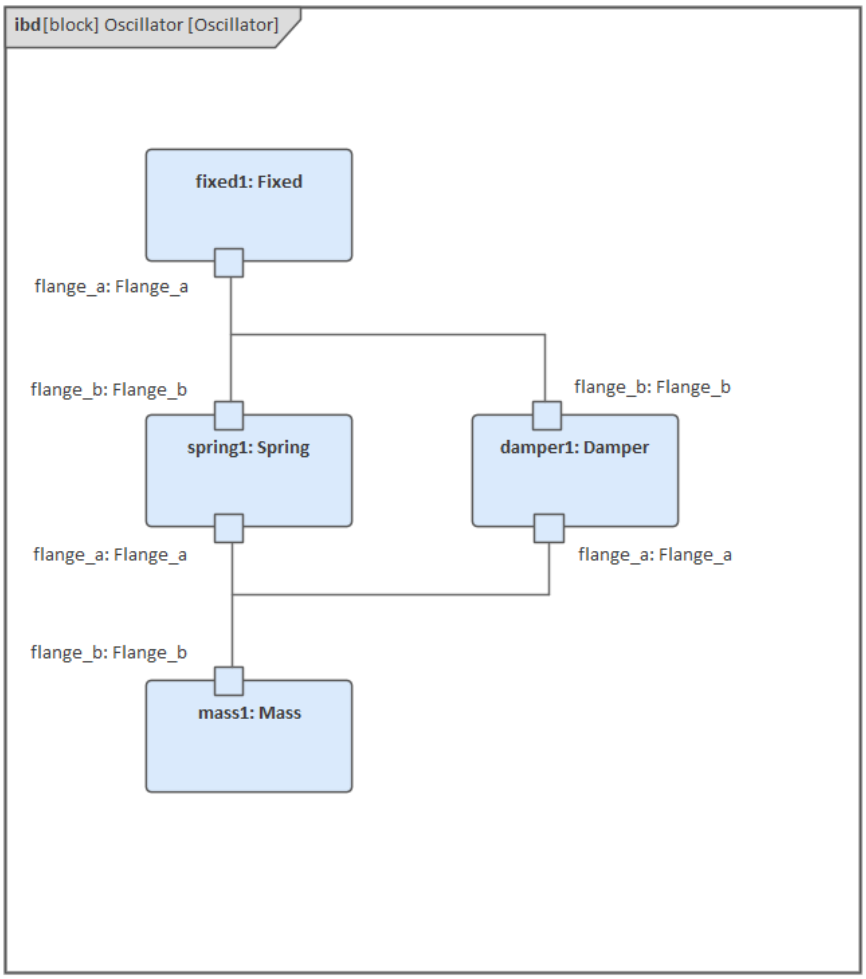
Créer Modèle SysML

Le modèle MassSpringDamperOscillator dans SysML possède un Bloc principal, l' *Oscillateur* . L'Oscillateur est composé de quatre parties : un *plafond* fixe, un *ressort* , un *amortisseur* et un *corps de masse* . Créez un Bloc pour chacune de ces parties. Les quatre parties du Bloc Oscillateur sont connectées via des Ports, qui représentent des brides mécaniques.

Composants	Description
Types de ports	Les blocs « Flange_a » et « Flange_b » utilisés pour les brides dans le domaine mécanique de transition 1D sont identiques mais ont des rôles légèrement différents, quelque peu analogues aux rôles de PositivePin et NegativePin dans le domaine électrique. Les forces sont transmises à travers les brides. L'attribut *isConserved* de la propriété d'écoulement *Flange.f* doit donc être défini sur True.

	
Blocs et ports	• Créez les blocs « Ressort », « Amortisseur », « Masse » et « Fixe » pour représenter respectivement le ressort, l'amortisseur, le corps de masse et le plafond • Créez un Bloc « PartialCompliant » avec deux ports (brides), nommés « flange_a » et « flange_b » — ceux-ci sont respectivement de type Flange_a et Flange_b ; les blocs « Spring » et « Damper » généralisent à partir de « PartialCompliant » • Créez un Bloc « PartialRigid » avec deux ports (brides), nommés « flange_a » et « flange_b » — ceux-ci sont respectivement de type Flange_a et Flange_b ; le Bloc « Mass » se généralise à partir de « PartialRigid » • Créez un Bloc « Fixe » avec une seule bride pour le plafond, qui n'a que le port « flange_a » typé sur Flange_a

	
Structure interne	Créez un diagramme Bloc internes (IBD) pour « Oscillateur ». Ajoutez des propriétés pour le plafond fixe, le ressort, l'amortisseur et le corps de masse, typées par les blocs correspondants. Connectez les ports avec des connecteurs. • Connectez « flange_a » de « fixed1 » à « flange_b » de « spring1 » • Connectez « flange_b » de « damper1 » à « flange_b » de « spring1 » • Connectez « flange_a » de « damper1 » à « flange_a » de « spring1 » • Connectez « flange_a » de « spring1 » à « flange_b » de « mass1 »

	
Contraintes	Pour plus de simplicité, nous définissons les contraintes directement dans les éléments Bloc ; vous pouvez éventuellement définir des ConstraintBlocks, utiliser des propriétés de contrainte dans les Blocks et lier leurs paramètres aux propriétés du Bloc .

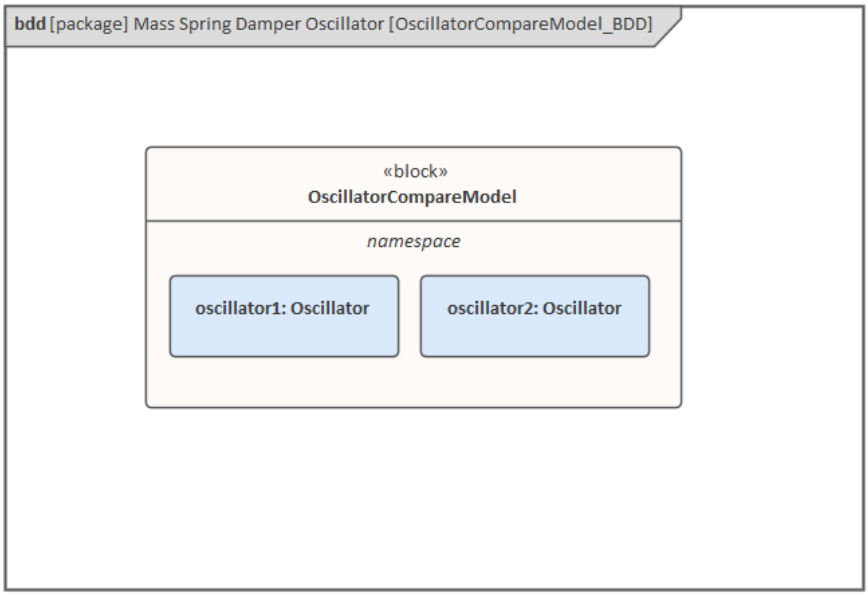
Comparaison de deux plans d'oscillateurs

Après avoir modélisé l'oscillateur, nous souhaitons effectuer une analyse hypothétique. Par exemple :

- Quelle est la différence entre deux oscillateurs avec des amortisseurs différents ?
- Et s'il n'y a pas d'amortisseur ?
- Quelle est la différence entre deux oscillateurs avec des ressorts différents ?
- Quelle est la différence entre deux oscillateurs avec des masses différentes ?

Voici les étapes pour créer un modèle de comparaison :

- Créer un Bloc nommé « OscillatorCompareModel »
- Créez deux Propriétés pour « OscillatorCompareModel », appelées *oscillator1* et *oscillator2,* et saisissez-les avec le Bloc *Oscillator*



Configurer DataSet et Exécuter Simulation

Créez un artefact de configuration SysMLSim et attribuez-le à ce Paquetage . Créez ensuite ces ensembles de données :

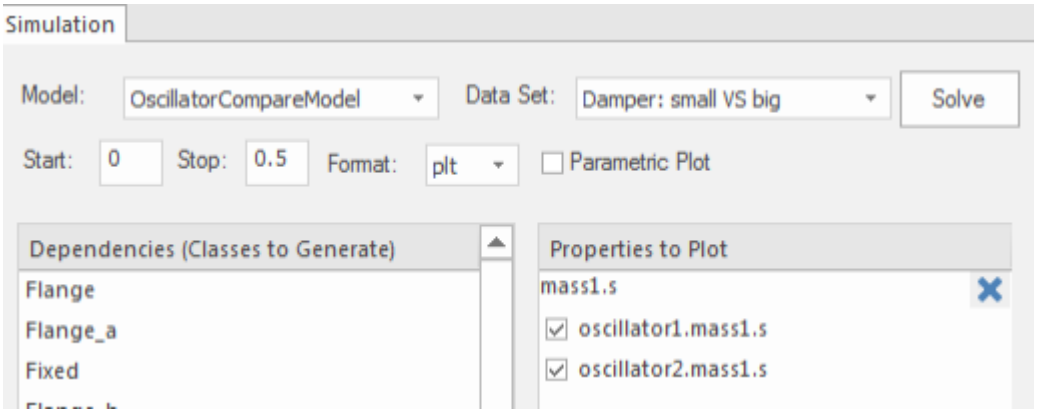
- Amortisseur : petit ou grand
  fournir 'oscillator1.damper1.d' avec la valeur 10 et 'oscillator2.damper1.d' avec la valeur la plus élevée 20
- Amortisseur : non ou oui
  fournir à 'oscillator1.damper1.d la valeur 0 ; ('oscillator2.damper1.d' utilisera la valeur par défaut 25)
- Printemps : petit vs grand
  fournir 'oscillator1.spring1.c' avec la valeur 6000 et 'oscillator2.spring1.c' avec la valeur plus grande 12000
- Masse : légère vs lourde
  fournir 'oscillator1.mass1.m' avec la valeur 0,5 et 'oscillator2.mass1.m' avec la valeur la plus élevée 2

La page configurée ressemble à ceci :

OscillatorCompareModel	SysMLSimModel
Part	
Damper: small VS big	Click button to configure...
oscillator2.damper1.d	20
oscillator1.damper1.d	10
Spring: small VS big	Click button to configure...
oscillator2.spring1.c	12000
oscillator1.spring1.c	6000
Damper: no VS yes	Click button to configure...
oscillator1.damper1.d	0
Mass: light VS Heavy	Click button to configure...
oscillator2.mass1.m	2
oscillator1.mass1.m	0.5

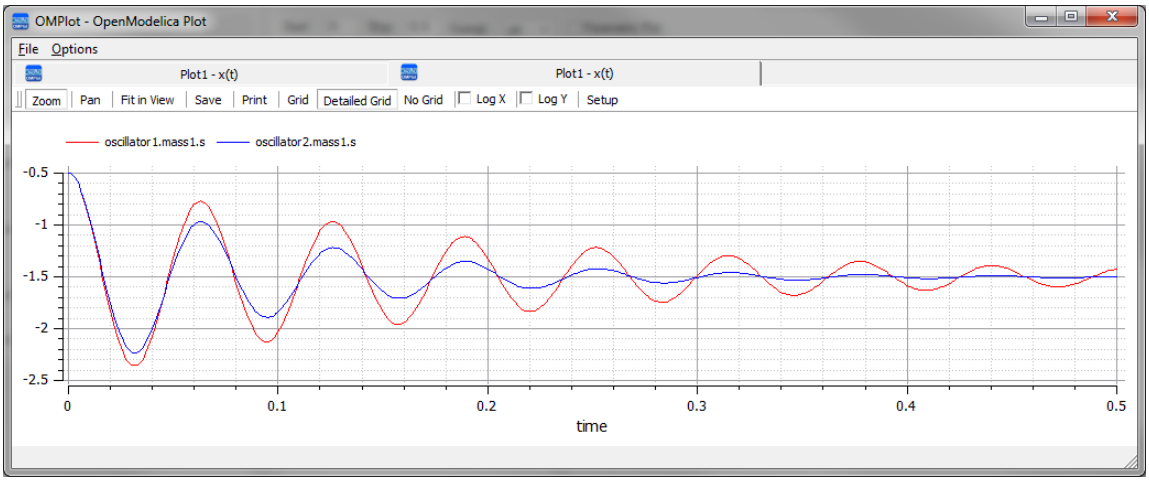
Sur la page « Simulation », sélectionnez « OscillatorCompareModel », tracez « oscillator1.mass1.s » et « oscillator2.mass1.s », puis choisissez l'un des ensembles de données créés et exécuter la simulation.

Conseil : S'il y a trop de propriétés dans la liste des parcelles, vous pouvez basculer la barre de filtre en utilisant le menu contextuel sur l'en-tête de la liste, puis taper 'mass1.s' dans cet exemple.

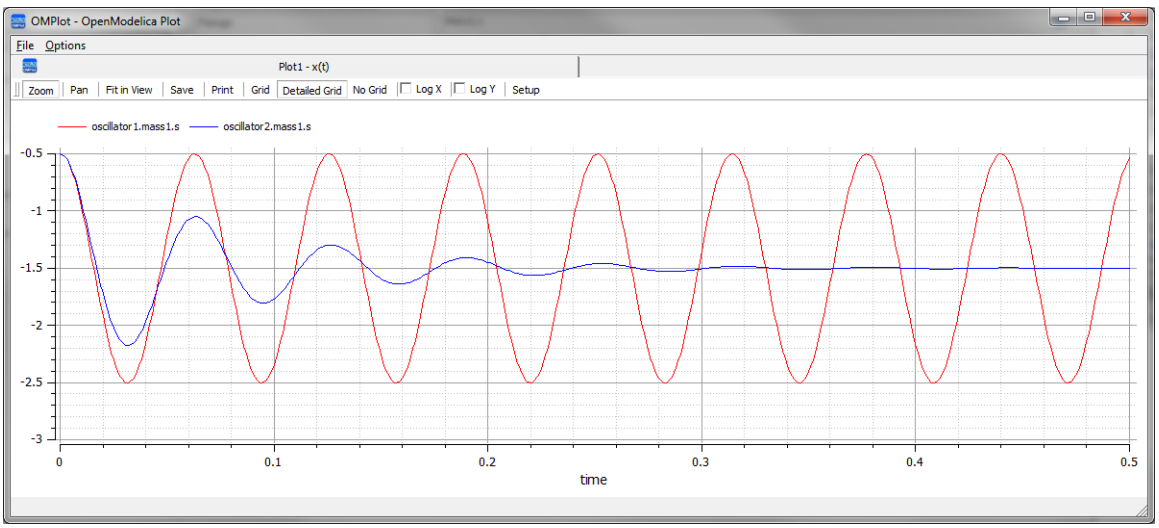


Voici les résultats de la simulation :

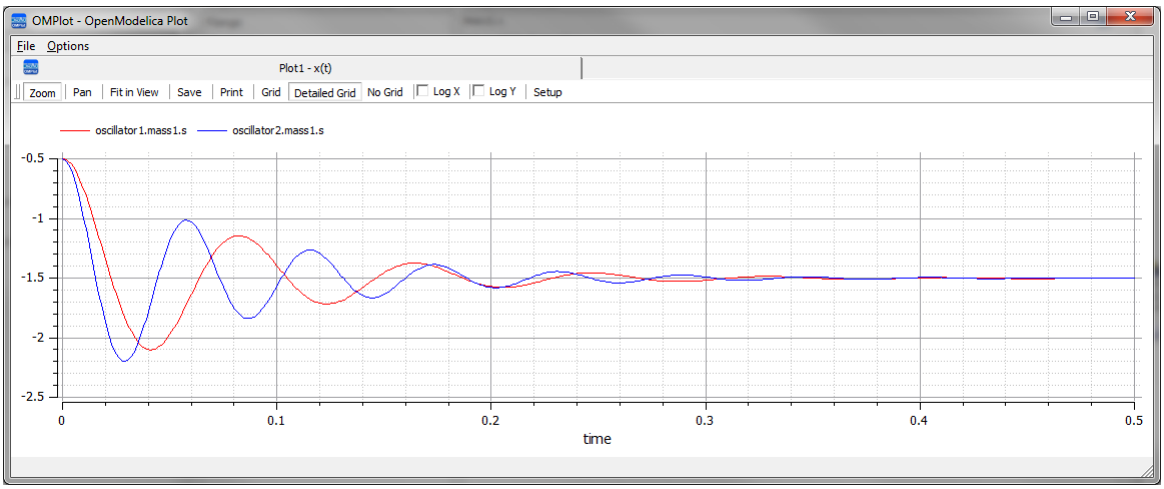
- Amortisseur, petit ou grand : le plus petit amortisseur permet au corps d'osciller davantage



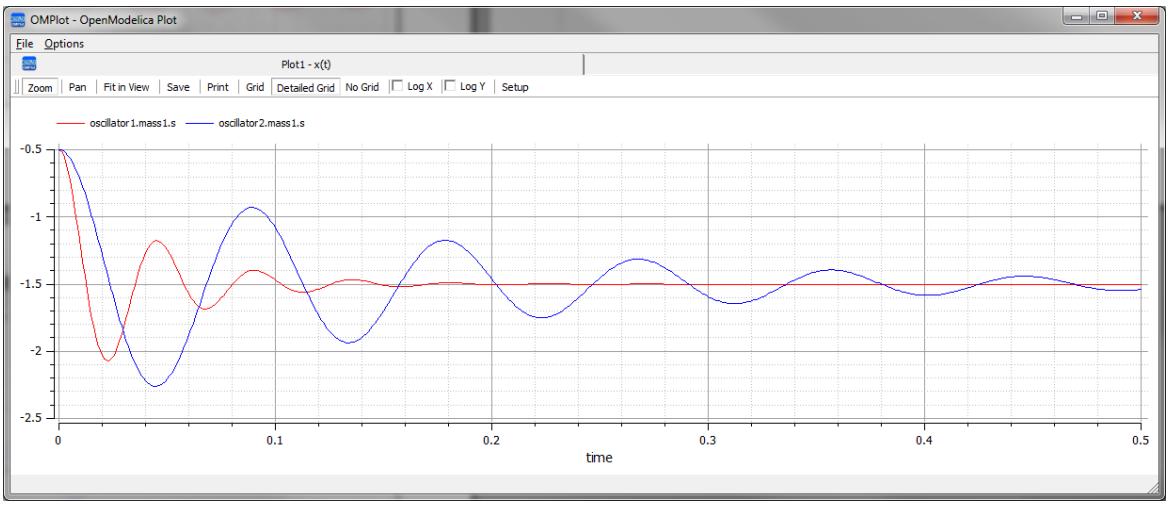
- Amortisseur, non vs oui : l'oscillateur ne s'arrête jamais sans amortisseur



- Ressort, petit ou grand : le ressort avec le plus petit « c » oscillera plus lentement



- Masse, légère vs lourde : l' object avec la plus petite masse oscillera plus vite et régulera plus rapidement



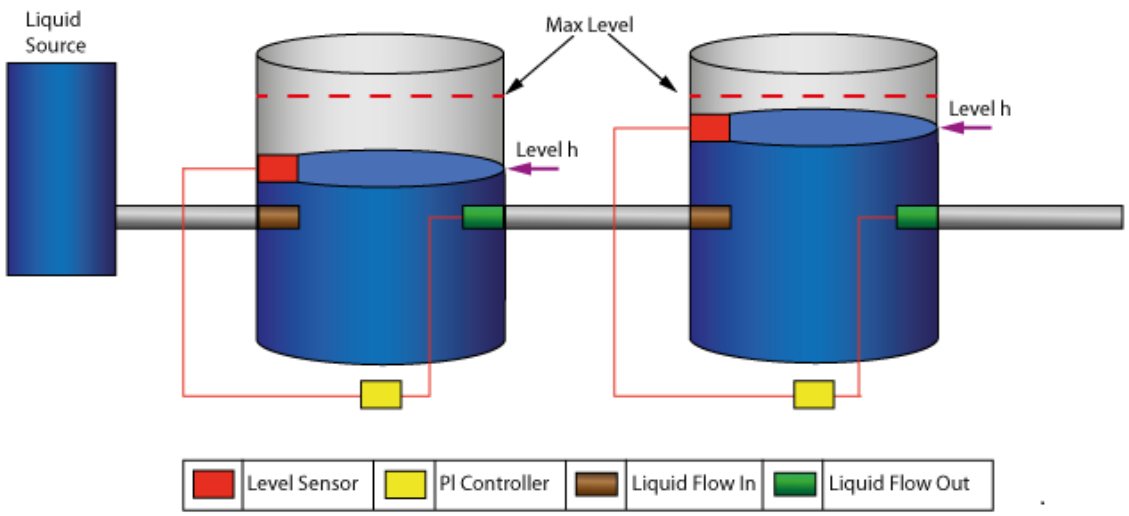
Régulateur de pression du réservoir d'eau

Dans cette section, nous allons parcourir la création d'un modèle SysML Paramétriques pour un régulateur de pression de réservoir d'eau, composé de deux réservoirs connectés, d'une source d'eau et de deux contrôleurs, chacun surveillant le niveau d'eau et contrôlant la vanne pour réguler le système.

Nous expliquerons le modèle SysML, le créerons et configurerons les Configurations SysMLSim. Nous exécuter ensuite la Simulation avec OpenModelica.

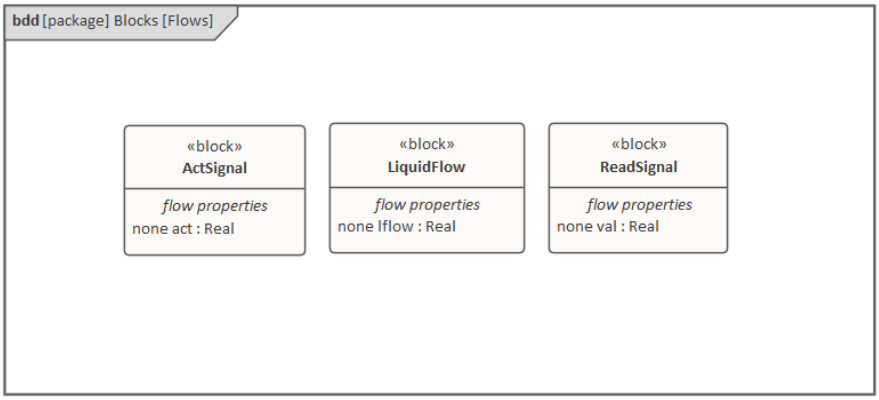
Système en cours de modélisation

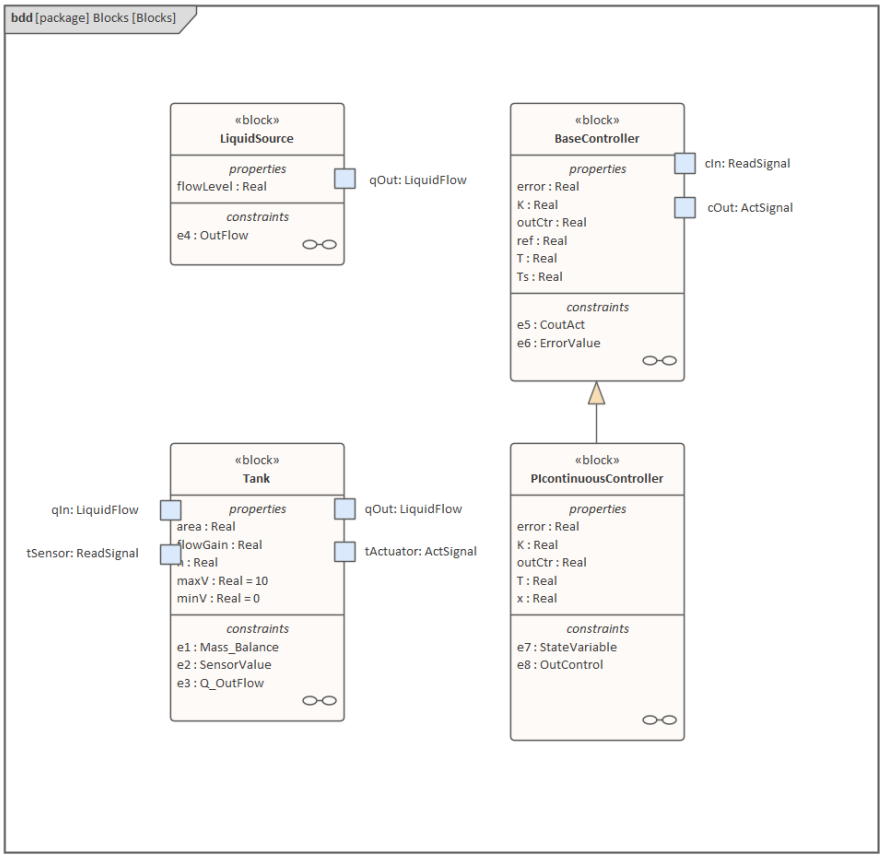
Ce diagramme représente deux réservoirs reliés entre eux et une source d'eau qui remplit le premier réservoir. Chaque réservoir est relié à un contrôleur continu proportionnel-intégral (PI), qui régule le niveau d'eau contenu dans les réservoirs à un niveau de référence. Pendant que la source remplit le premier réservoir d'eau, le contrôleur continu PI régule le débit sortant du réservoir en fonction de son niveau réel. L'eau du premier réservoir s'écoule dans le deuxième réservoir, que le contrôleur continu PI tente également de réguler. Il s'agit d'un problème physique naturel, non spécifique à un domaine.



Créer Modèle SysML

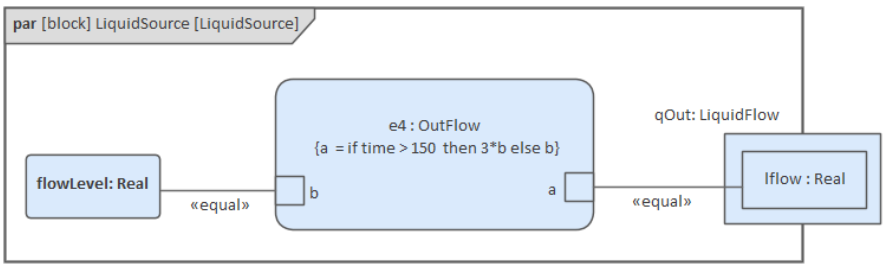
Composant	Discussion
Types de ports	Le char dispose de quatre ports qui sont typés sur ces trois blocs : • ReadSignal : lecture du niveau de liquide ; cette propriété a une unité « m » • ActSignal : Le signal envoyé à l'actionneur pour régler la position de la vanne • LiquidFlow : Le débit de liquide aux entrées ou aux sorties ; il a une propriété « lflow » avec l'unité « m^3/s »

	
Interrompre lorsqu'une Variable Change de Valeur	LiquidSource : l'eau entrant dans le réservoir doit provenir de quelque part, c'est pourquoi nous avons un composant de source liquide dans le système de réservoir, avec la propriété *flowLevel* ayant une unité de « m ^{3} /s ». Un port « qOut » est typé sur « LiquidFlow ». Réservoir : Les réservoirs sont connectés aux contrôleurs et aux sources de liquide via des ports. • Chaque réservoir dispose de quatre ports : - qIn : pour le flux d'entrée - qOut : pour le flux de sortie - tSensor : pour fournir des mesures de niveau de fluide - tActionneur : pour régler la position de la vanne à la sortie de le réservoir • Propriétés : - volume (unité='m ^{3} ') : capacité du réservoir, intervenant dans le *bilan massique* équation - h (unité = 'm') : niveau d'eau, intervenant dans le *bilan de masse* équation ; sa valeur est lue par le capteur - flowGain (unité = 'm ^{3} /s') : le débit de sortie est lié à la vanne position par *flowGain* - minV, maxV : Limites du débit de la vanne de sortie BaseController : ce Bloc peut être le parent ou l'ancêtre d'un contrôleur continu PI et d'un contrôleur discret PI. • Ports: - cIn : Niveau du capteur d'entrée - cOut : Contrôle vers l'actionneur • Propriétés : - Ts (unité = 's') : Période de temps entre les échantillons discrets (non utilisée dans cet exemple) - K : Facteur de gain - T (unité = 's') : Constante de temps du contrôleur - ref : niveau de référence - erreur : différence entre le niveau de référence et le niveau réel niveau d'eau, obtenu à partir du capteur - outCtr : signal de commande vers l'actionneur pour contrôler la vanne position PIcontinuousController : spécialisation à partir de BaseController • Propriétés : - x : la variable d'état du contrôleur



Blocs de contraintes

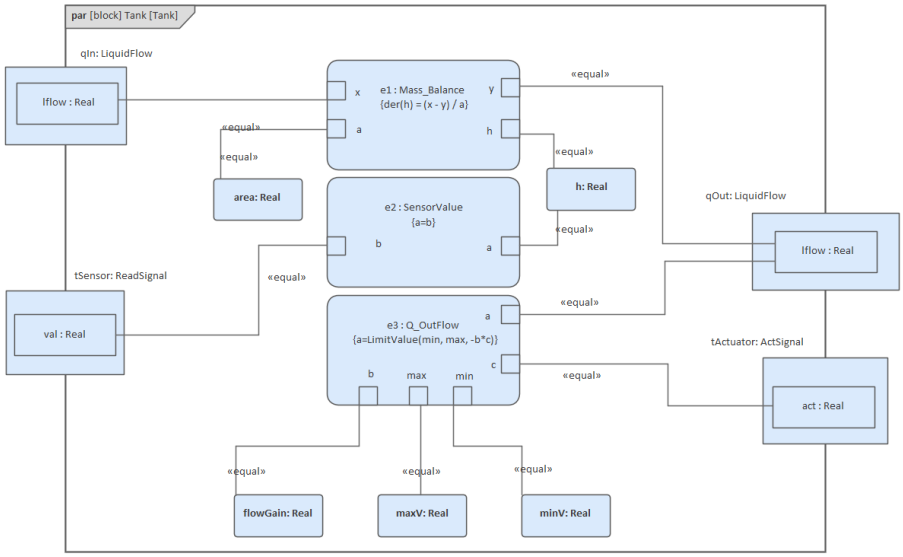
Le débit augmente brusquement à l'instant = 150 jusqu'à un facteur trois du niveau de débit précédent, ce qui crée un problème de contrôle intéressant que le contrôleur du réservoir doit gérer.



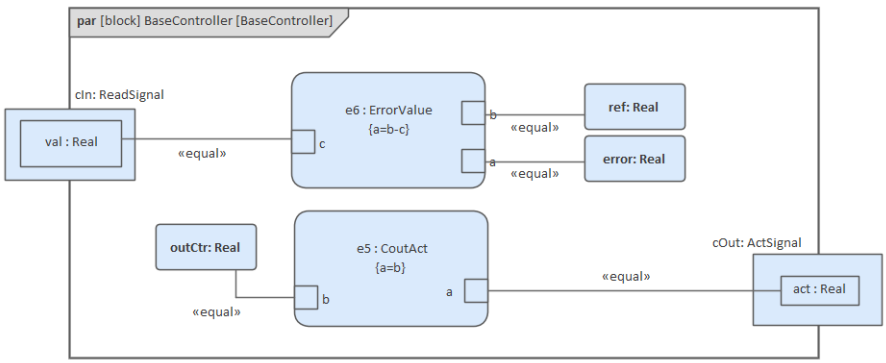
L'équation centrale régulant le comportement du réservoir est l'équation *de bilan massique* .

Le débit de sortie est lié à la position de la vanne par un paramètre « flowGain ».

Le capteur lit simplement le niveau du réservoir.



Les contraintes définies pour « BaseController » et « PIcontinuousController » sont illustrées dans ces figures.



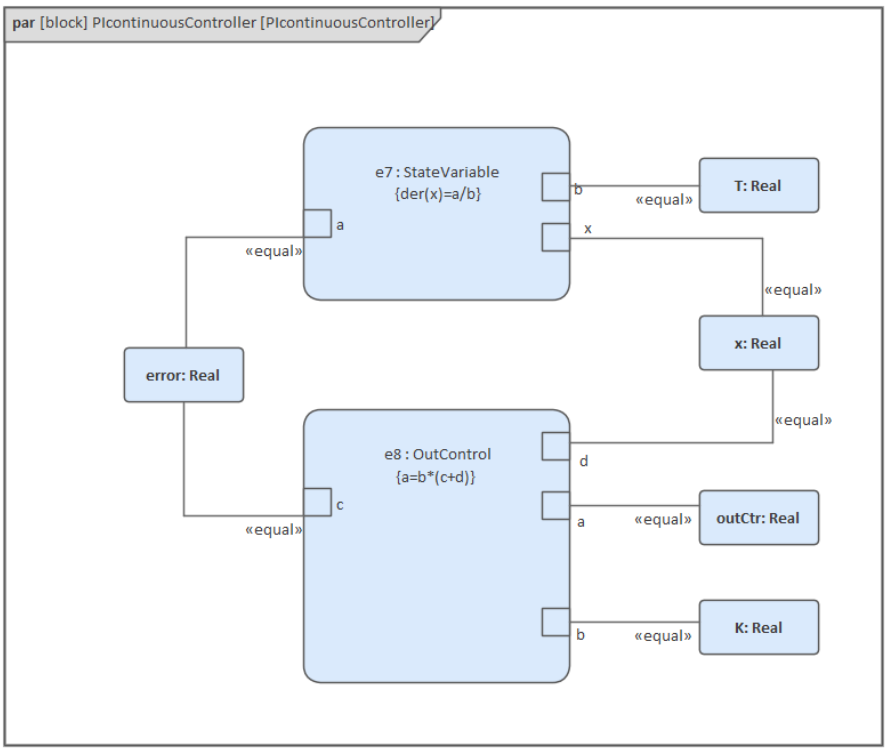
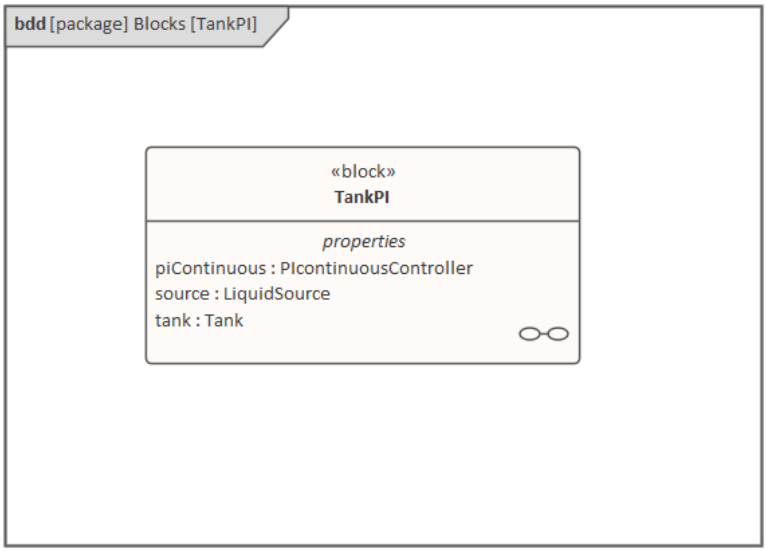
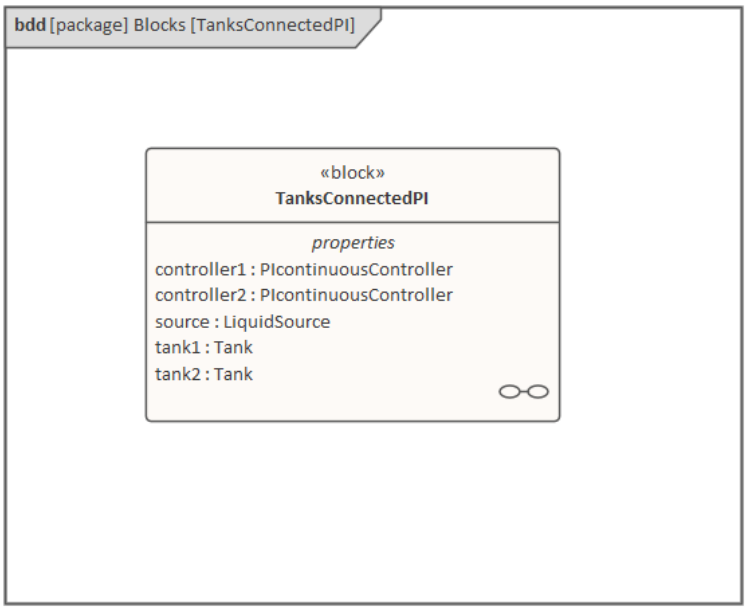


Diagramme Interne de Bloc	Il s'agit du diagramme Bloc interne d'un système avec un seul réservoir.  Il s'agit du diagramme Bloc interne d'un système avec deux réservoirs connectés. 

Exécuter Simulation

Étant donné que *TankPI* et *TanksConnectedPI* sont définis comme « SysMLSimModel », ils seront répertoriés dans la zone de liste déroulante « Modèle » sur la page « Simulation ».

Sélectionnez *TanksConnectedPI* et observez les modifications suivantes dans l'interface graphique :

- Combobox « Ensemble de données » : sera rempli avec tous les ensembles de données définis dans *TanksConnectedPI*
- Liste « Dépendances » : collectera automatiquement tous les blocs, contraintes, fonctions SimFunctions et types de valeurs directement ou indirectement référencés par *TanksConnectedPI* (ces éléments seront générés sous forme de code OpenModelica)
- « Propriétés à tracer » : une longue liste de propriétés de variables « feuille » (c'est-à-dire qui n'ont pas de propriétés)

sera collectée ; vous pouvez en choisir une ou plusieurs à simuler, et les Propriétés seront affichées dans la légende du tracé

Créer un artefact et configurer

Sélectionnez 'Simulate > Comportement du Système > Modelica/Simulink > SysMLSim Configuration Manager'.

Les éléments du Paquetage seront chargés dans le Gestionnaire de Configuration.

Configurez ces blocs et leurs propriétés comme indiqué dans ce tableau .

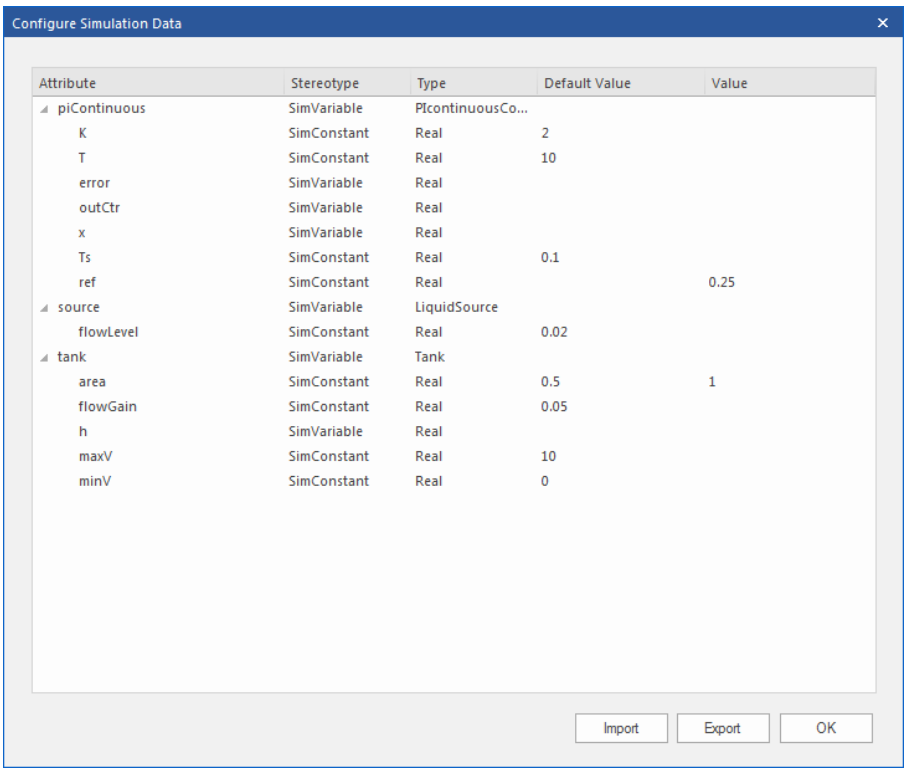
Note : Propriétés non configurées comme 'SimConstant' sont 'SimVariable' par défaut.

Bloc	Propriétés
Source liquide	Configurer comme « SysMLSimClass ». Configuration Propriétés : • flowLevel : défini comme « SimConstant »
Réservoir	Configurer comme « SysMLSimClass ». Configuration Propriétés : • zone : définir comme « SimConstant » • flowGain : défini sur "SimConstant" • maxV : défini comme « SimConstant » • minV : défini comme « SimConstant »
Contrôleur de base	Configurer comme « SysMLSimClass ». Configuration Propriétés : • K : définir comme « SimConstant » • T : définir comme « SimConstant » • Ts : définir comme « SimConstant » • ref : définir comme « SimConstant »
Contrôleur continu PI	Configurer comme « SysMLSimClass ».
RéservoirPI	Configurer comme « SysMLSimModel ».
RéservoirsConnectedPI	Configurer comme « SysMLSimModel ».

Configuration du jeu de données

Cliquez-droit sur chaque élément, sélectionnez l'option « Créer un jeu de données Simulation » et configurez les jeux de données comme indiqué dans ce tableau .

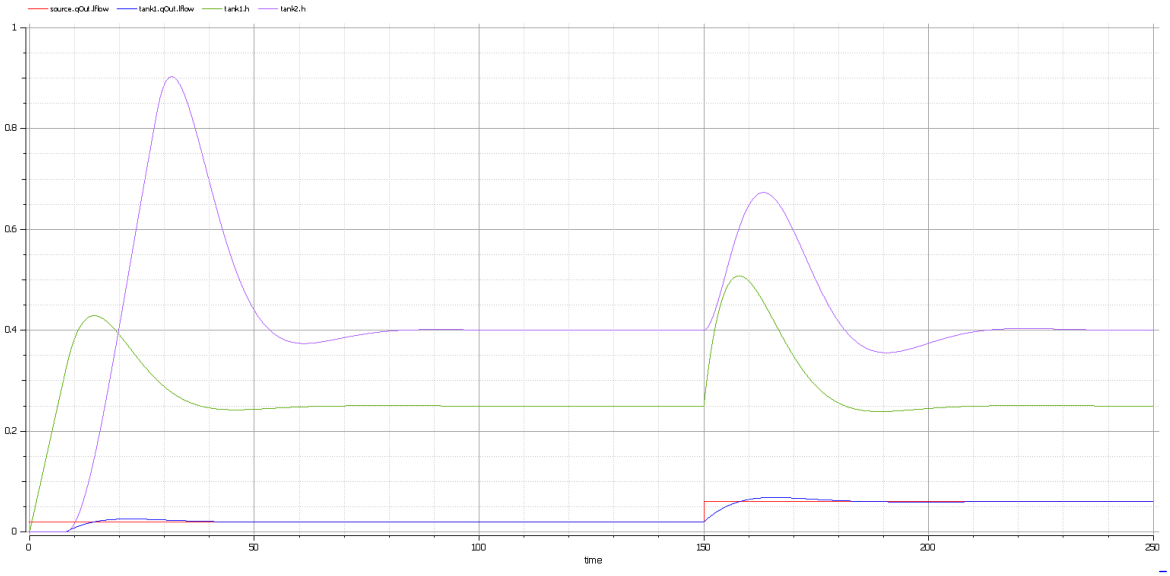
Élément	Ensemble de données
Source liquide	Niveau de débit : 0,02
Réservoir	h.début: 0

	Gain de débit : 0,05 superficie: 0,5 maxV: 10 minV: 0
Contrôleur de base	T: 10 K: 2 Ts: 0,1
Contrôleur continu PI	Aucune configuration nécessaire. Par défaut, le Bloc spécifique utilisera les valeurs configurées à partir du jeu de données par défaut du super Bloc .
RéservoirPI	Ce qui est intéressant ici, c'est que la valeur par défaut peut être chargée dans la dialogue « Configurer les données Simulation ». Par exemple, les valeurs que nous avons configurées comme dataSet par défaut sur chaque élément Bloc ont été chargées comme valeurs par défaut pour les propriétés de TankPI. Cliquez sur l'icône de chaque ligne pour étendre les structures internes de la propriété à une profondeur arbitraire.  Cliquez sur le bouton OK et revenez au gestionnaire de configuration. Les valeurs suivantes sont alors configurées : • tank.area: 1 ceci remplace la valeur par défaut 0,5 définie dans l'ensemble de données du Tank Bloc • piContinuous.ref: 0.25
RéservoirsConnectedPI	• contrôleur1.ref: 0.25 • contrôleur2.ref: 0.4

Simulation et analyse 1

Sélectionnez ces variables et cliquez sur le bouton Résoudre. Ce graphique devrait prompt :

- source.qOut.lflow
- réservoir1.qOut.lflow
- réservoir1.h
- réservoir2.h



Voici les analyses du résultat :

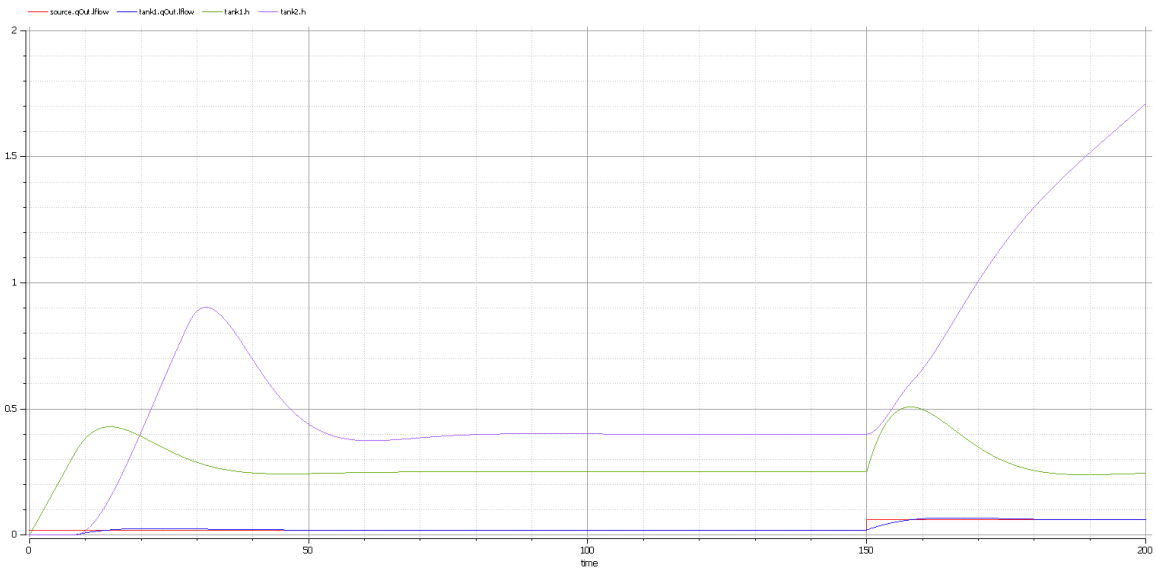
- Le débit du liquide augmente brusquement à l'instant = 150, jusqu'à 0,06 m ^{3} /s, soit un facteur trois du débit précédent (0,02 m ^{3} /s)
- Réservoir 1 régulé à une hauteur de 0,25 et réservoir 2 régulé à une hauteur de 0,4 comme prévu (nous avons défini la valeur du paramètre via l'ensemble de données)
- Les réservoirs 1 et 2 ont été régulés deux fois pendant la simulation ; la première fois avec un débit de 0,02 m ^{3} /s ; la deuxième fois avec un débit de 0,06 m ^{3} /s
- Le réservoir 2 était vide avant qu'il n'y ait un flux provenant du réservoir 1

Simulation et analyse 2

Dans l'exemple, nous avons défini les propriétés du réservoir « minV » et « maxV » sur les valeurs 0 et 10 respectivement. Dans le monde réel, un débit de 10 m ^{3} /s nécessiterait l'installation d'une vanne de très grande taille sur le réservoir.

Que se passerait-il si nous changions la valeur de « maxV » à 0,05 m ^{3} /s ? Sur la base du modèle précédent, nous pourrions effectuer les modifications suivantes :

- Sur le « DataSet_1 » existant de TanksConnectedPI, cliquez-droit et sélectionnez « Dupliquer le DataSet », puis renommez-le en « Tank2WithLimitValveSize »
- Cliquez sur le bouton pour configurer, développez « tank2 » et saisissez « 0,05 » dans la colonne « Valeur » pour la propriété « maxV »
- Sélectionnez « Tank2WithLimitValveSize » sur la page « Simulation » et tracez les propriétés
- Cliquez sur le bouton Résoudre pour exécuter la simulation



Voici les analyses des résultats :

- Notre changement s'applique uniquement au réservoir 2 ; le réservoir 1 peut réguler comme avant sur 0,02 m^3/s et 0,06 m^3/s
- Lorsque le débit de la source est de 0,02 m^3/s, le réservoir 2 peut réguler comme avant
- Cependant, lorsque le débit de la source augmente à 0,06 m^3/s, la vanne est trop petite pour permettre au débit de sortie de correspondre au débit d'entrée ; le seul résultat est que le niveau d'eau du réservoir 2 augmente
- Il appartient alors à l'utilisateur de résoudre ce problème ; par exemple, changer pour une vanne plus grande, réduire le débit de la source ou fabriquer une vanne supplémentaire.

En résumé, cet exemple montre comment ajuster les valeurs des paramètres en dupliquant un DataSet existant.