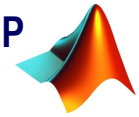
The MathWorks MATLAB&SIMULINK

Cours MATLAB et Simulink

appliqué au traitement vidéo sur DSP

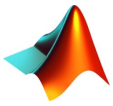


Daniel MARTINS
Ingénieur d'applications
01 41 14 88 40
daniel.martins@mathworks.fr

The MathWorks MATLAB&SIMULINK

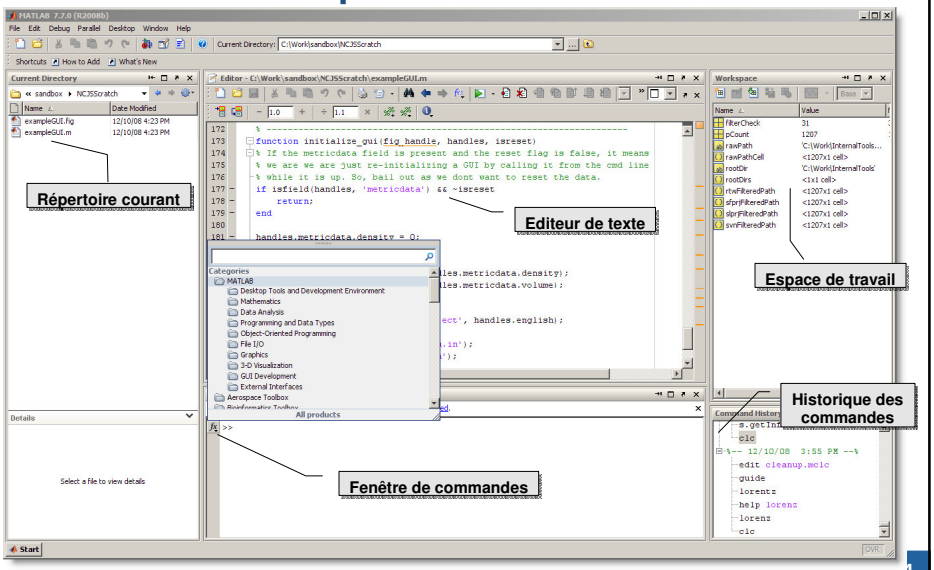
Agenda

- 9h00 : Introduction à MATLAB : quelques commandes à savoir
- 9h30 : Introduction à Simulink : Comment fonctionne le moteur de calcul
- 9h45 : Modélisation d'un filtre de Sobel : prise en main de l'outil, gestion de la taille des données, du type.
- 10h30 : Pause
- 10h45 : Création de ses propres composants : import de code C dans Simulink
- 11h00 : Génération de code C générique
- 11h15 : Application à un DSP Texas Instruments avec création d'un projet Code Composer Studio
- 11h30 : Vérification du code objet dans Simulink : Processor In the Loop (PIL)
- 11h45 : Profile du code DSP
- 12h00 : Fin



Introduction à MATLAB

MATLAB Desktop



Création de données

- **Création d'un scalaire**

```
>> a = 1.3
a =
    1.3000
```

Le point virgule après l'expression supprime l'affichage

```
>> a = 1.3;
```

- **Création d'un vecteur :**

```
>> b = [1 -2 13 41]
b =
     1     -2     13     41
>> c = 1:5
c =
  1    2    3    4    5
>> d = 4:-0.5:2.5
d =
    4.0000    3.5000    3.0000    2.5000
```

n:inc:m est identique à [n, n+inc, n+2*inc, ...] et ne dépassera pas m

Création de données

- **Création d'une matrice:**

```
>> A = [1, 2; 3, 4]
A =
     1     2
     3     4
```

La virgule (ou espace) est un séparateur de colonnes
Le point-virgule est un séparateur de lignes
On peut concaténer plusieurs éléments:

```
>> B = [A 2*A]
B =
     1     2     2     4
     3     4     6     8
>> C = [A; 2*A]
C =
     1     2
     3     4
     2     4
     6     8
```

Création de données

- **Transposé d'une matrice**

```
>> D = B'
D =
     1     2
     3     4
     2     4
     6     8
```

- **Opération sur les matrices**

```
Multiplication matricielle -> *
>> E = B*C
E =
    35    50
    75   110
Multiplication élément par élément -> .*
>> F = C.*D
F =
     1     6
     6    16
     4    24
    24    64
```


Indexation

- **Passer le numéro de ligne et le numéro de colonne en paramètres d'entrée**
- **L'indexation commence à 1, et le dernier élément est `end`**
- **Les vecteurs de nombres en ligne et/ou en colonne produisent un tableau qui est l'intersection des indices de ligne et de colonne**
- **Utiliser les deux-points pour spécifier un intervalle**
- **Les deux-points seuls (`:`) retournent tous les éléments dans cette dimension, ce qui est égal à `1:end`**

	1	2	3
1	1,1	1,2	1,3
2	2,1	2,2	2,3
3	3,1	3,2	3,3
4	4,1	4,2	4,3

Indexation

```
>> M(1,3)
ans =
     1.3
>> M(2,:)
ans =
     2.1     2.2     2.3
>> M(3:4,2:3)
ans =
     3.2   3.3
     4.2   4.3
>> M([1 3:4], 3)
ans =
     1.3
     3.3
     4.3
>> M(1,end)
ans =
     1.3
```

	1	2	3
1	1,1	1,2	1,3
2	2,1	2,2	2,3
3	3,1	3,2	3,3
4	4,1	4,2	4,3

M

Quelques commandes utiles

- `whos` Affiche les variables de l'espace de travail avec leur type et leur dimension
- `size` Détermine les dimensions d'une matrice. Accepte 2 arguments, le premier étant la matrice et le deuxième étant la dimension que l'on cherche (1 pour le nombre de lignes et 2 pour le nombre de colonnes)

```
>> A = [1 2 3 4; 5 6 7 8];
>> size(A,1)
ans =
   2
>> size(A,2)
ans =
   4
```

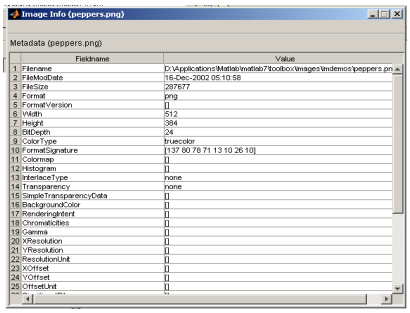
Lecture et écriture d'images

La lecture d'images se fait par imread.

L'écriture par imwrite.

imfinfo permet d'obtenir des informations sur l'image contenue dans un fichier.

imageinfo permet de récupérer les mêmes informations que la fonction imfinfo mais sous une interface graphique.



Les formats supportés : bmp, hdf, jpeg, pcx,tiff, xwd.

Représentation Graphique de l'importation d'une image

file.fmt

imfinfo('file.fmt')

Colormap?

oui

non

[x,map]=imread('file.fmt');

x=imread('file.fmt');

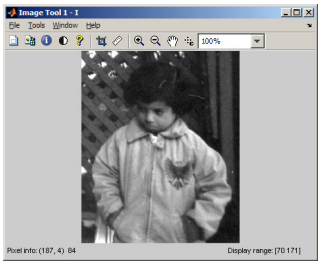
Image Type	Indexed
Double Data	[MXN]
Uint8 Data	[MXN]

Image Type	Intensity	Binary	RGB
Double Data	1 0.5 0.2 0 0.1 0.9 0 0.7 0.8	1 0 1 0 1 0 0 1 1	1 0.4 0.7 0.3 0.5 0.9 0.2
Uint8 Data	1 45 220 100 78 110 200 7 98	1 0 1 0 1 0 0 1 1	140 30 50 220

Affichage d'images

- `imtool` - Affiche l'image dans le visualiseur d'images.
- `imshow` - Affiche l'image.
- `image` - Créé et affiche un objet image (MATLAB).
- `imagesc` - Met à l 'échelle et affiche les images (MATLAB).
- `colorbar` - Affiche la barre des couleurs (MATLAB).
- `colormap` - Assigne une carte des couleur à l'image (MATLAB).
- `truesize` - Ajuste l'affichage à la taille de l'image.

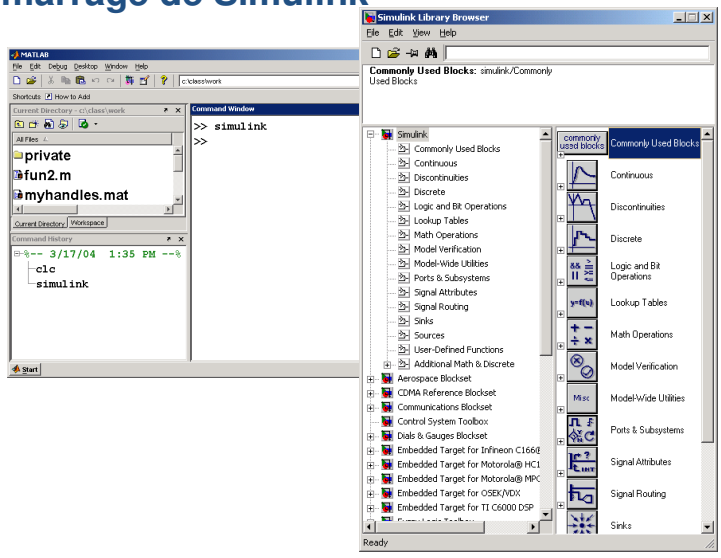
```
>> I = imread('pout.tif');
>> imtool(I)
```



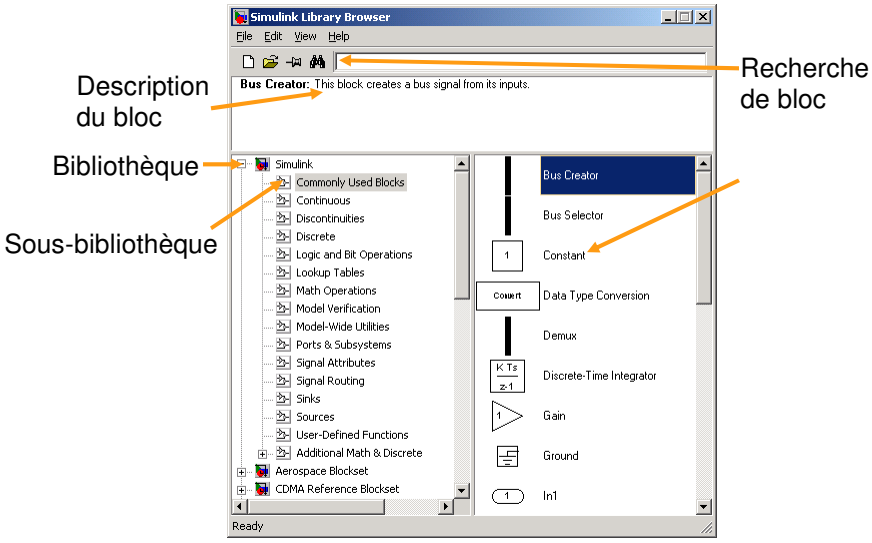
Introduction à Simulink

Création d'un modèle Simulink

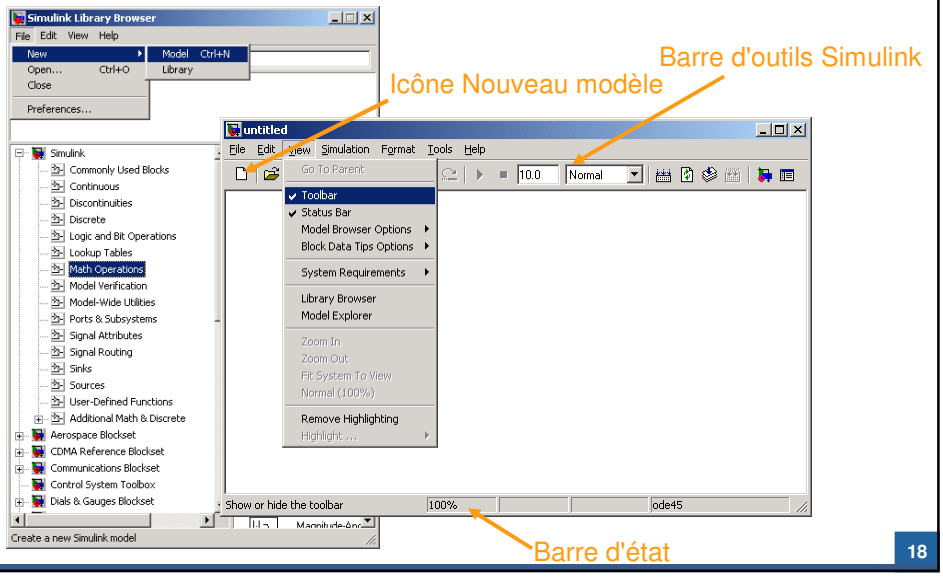
Démarrage de Simulink



Explorateur de bibliothèques Simulink

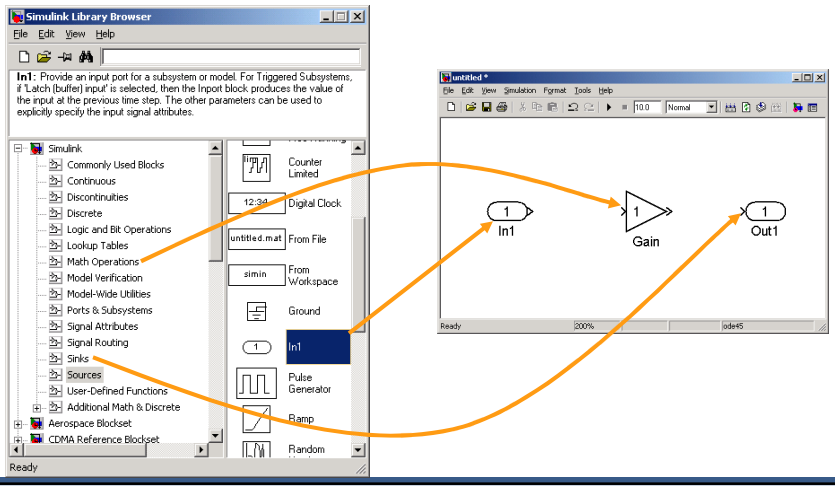


Création d'un nouveau schéma-bloc



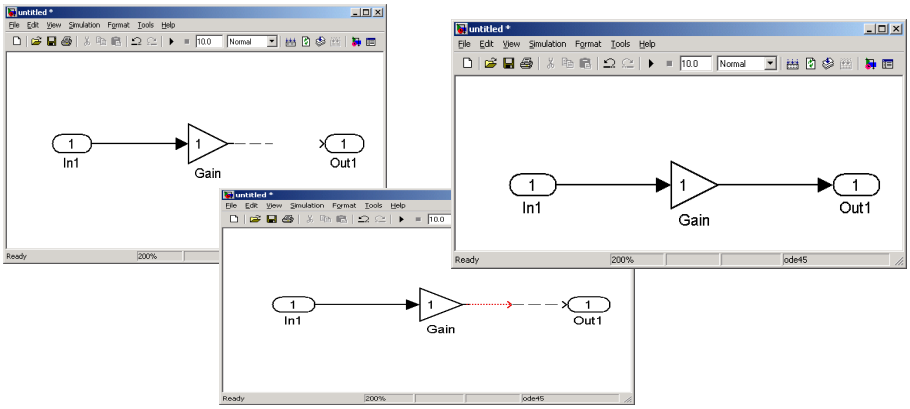
Création d'un nouveau schéma : ajout de blocs

Pour ajouter un bloc à un modèle, faites-le glisser dans le schéma-bloc depuis l'explorateur de bibliothèques. Les blocs sont donnés en liste alphabétique dans chaque bibliothèque.



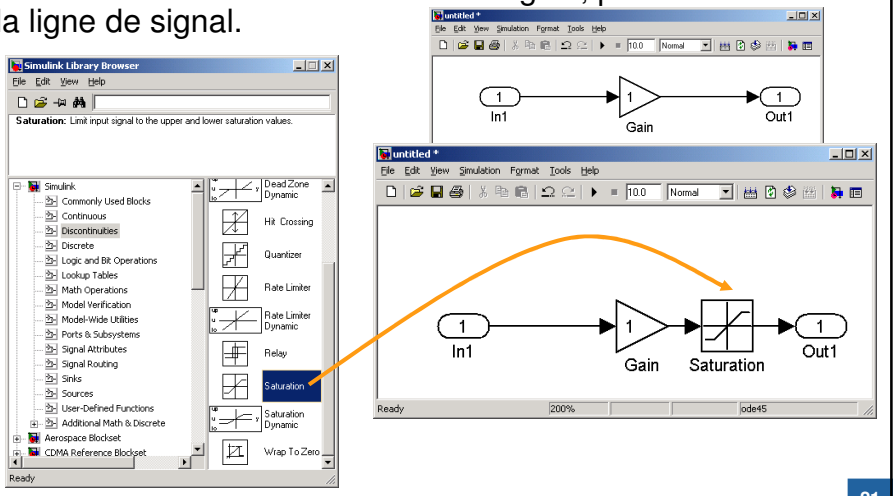
Création d'un nouveau schéma : ajout de signaux

Pour connecter deux blocs avec un signal, glissez depuis le port de sortie du bloc source vers le port d'entrée du bloc cible.



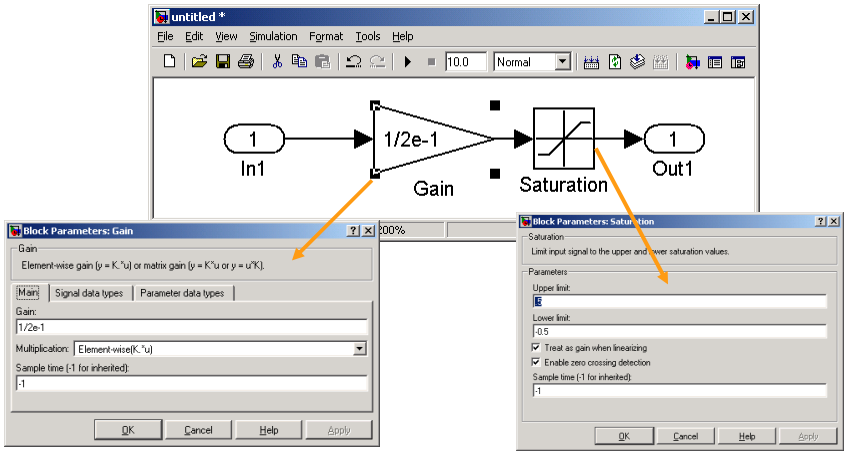
Édition du schéma-bloc : insertion de blocs

Pour insérer un bloc au milieu d'un signal, placez le bloc sur la ligne de signal.



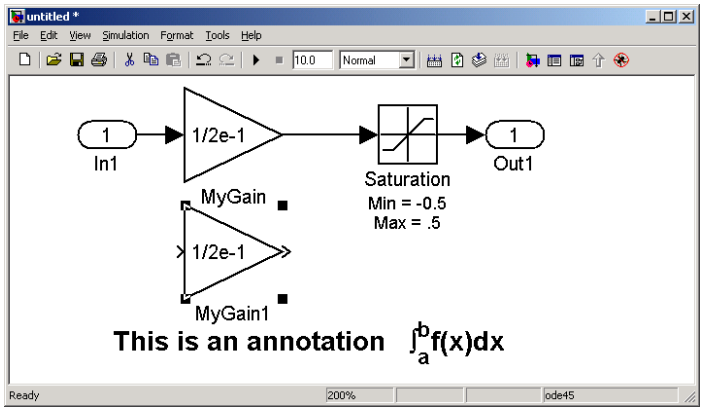
Définition des paramètres d'un bloc

- Pour ouvrir le dialogue des paramètres de bloc, double-cliquez sur le bloc
- Entrez une valeur correcte ou une sélection dans chaque champ.



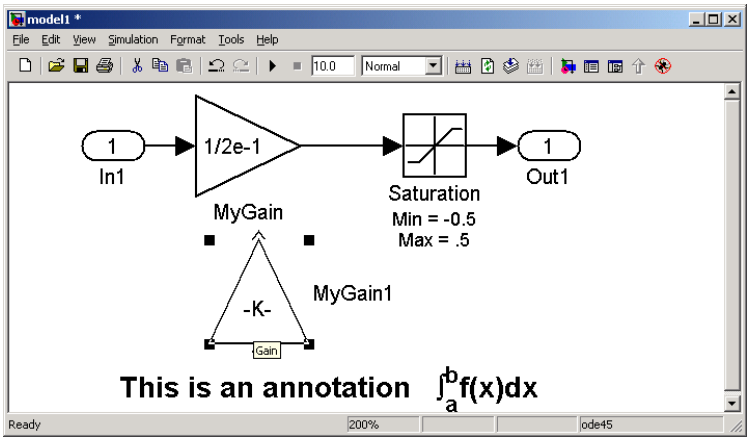
Copie de blocs

Pour copier un bloc, cliquez sur le bloc en utilisant le bouton droit de la souris et faites-le glisser.



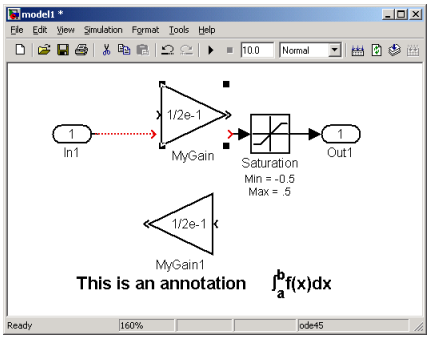
Réorientation des blocs

- **Pour retourner un bloc de gauche à droite, sélectionnez Flip block dans le menu Format après avoir sélectionné le bloc.**
- **Pour faire pivoter un bloc dans le sens des aiguilles d'une montre 90°, sélectionnez Rotate block dans le menu Format après avoir sélectionné le bloc.**



Déconnexion des blocs

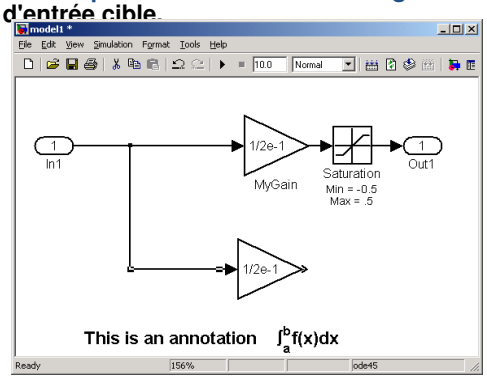
Pour déconnecter un bloc des signaux qui y sont attachés, cliquez sur le bloc avec le bouton droit de la souris et éloignez-le tout en pressant la touche Maj.



Maj + clic droit

Raccordement de signaux

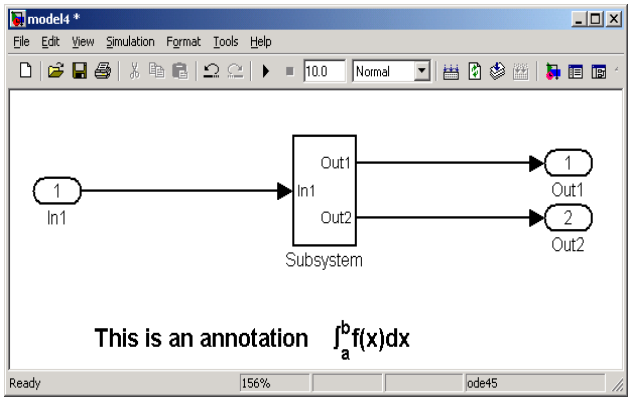
Pour ajouter un embranchement de signal, cliquez bouton droit sur le signal et glissez depuis l'endroit voulu de la ligne de signal jusqu'au port d'entrée cible.



>> model1

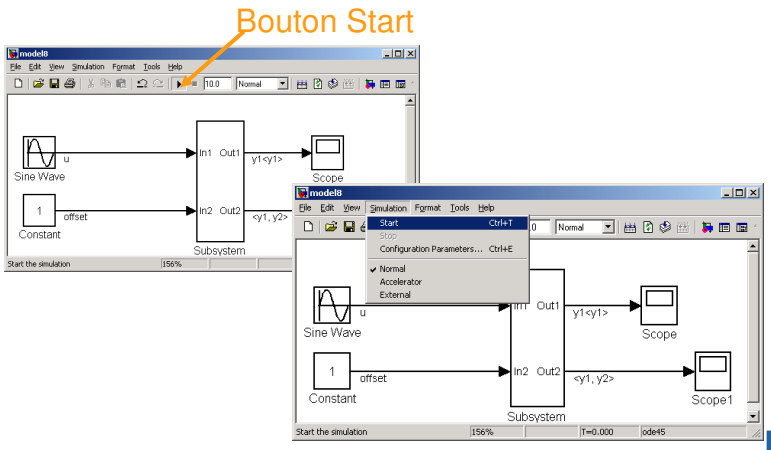
Création de sous-systèmes

Pour grouper des blocs en un sous-système, sélectionnez les blocs dont vous voulez qu'ils soient contenus dans le sous-système, puis sélectionnez Create subsystem dans le menu Edit.



Exécution du modèle avec les paramètres par défaut

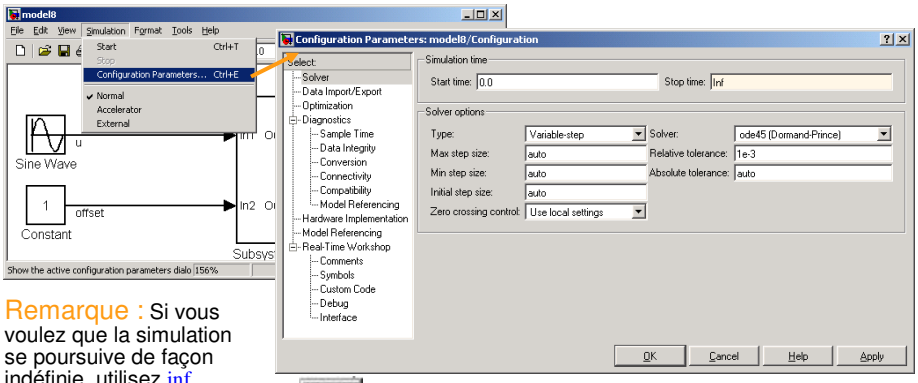
Pour simuler le modèle, pressez le bouton ▶ ou sélectionnez l'option Start dans le menu Simulation.



>> model8

Réglage du temps de simulation

- **Ouvrez les paramètres de configuration en sélectionnant Configuration Parameters dans le menu Simulation.**
- **Réglez les temps de simulation en utilisant les champs Start time et Stop time dans la section Solver de la boîte de dialogue Configuration Parameters.**

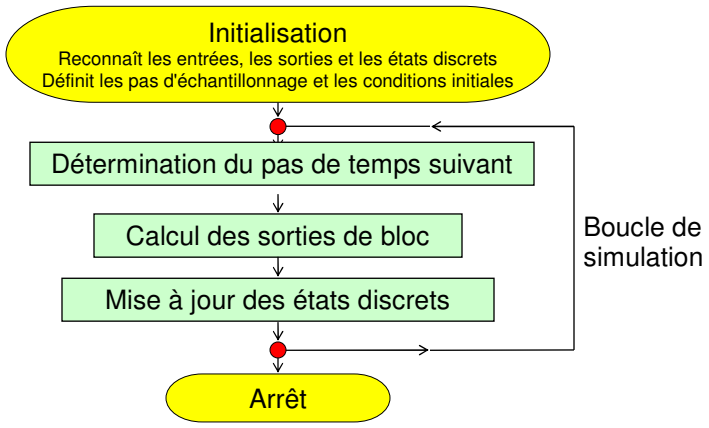


Remarque : Si vous voulez que la simulation se poursuive de façon indéfinie, utilisez inf comme Stop time.

Deuxième modèle

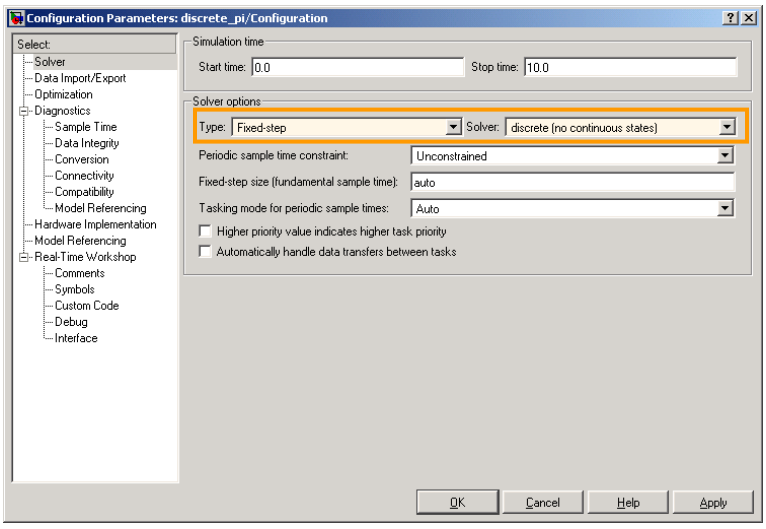
Le moteur de calcul de Simulink

Simulation de systèmes discrets

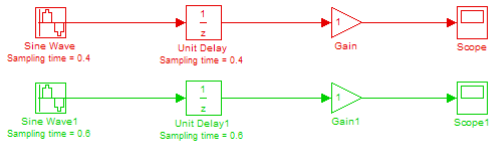


Systèmes discrets

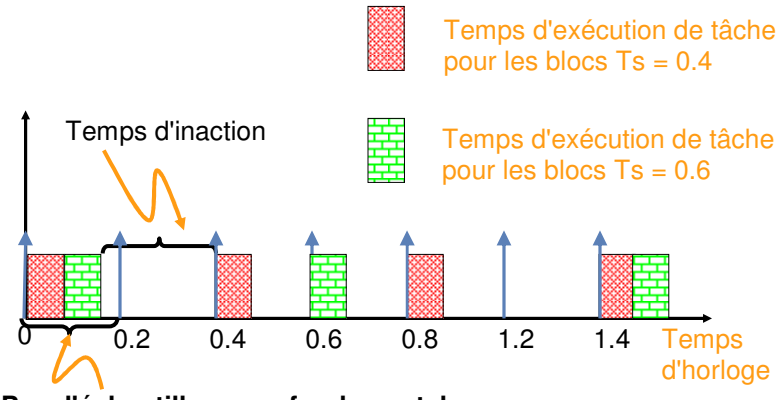
Choix d'un solveur pour un modèle à période d'échantillonnage unique



Solveurs discrets à périodes multiples



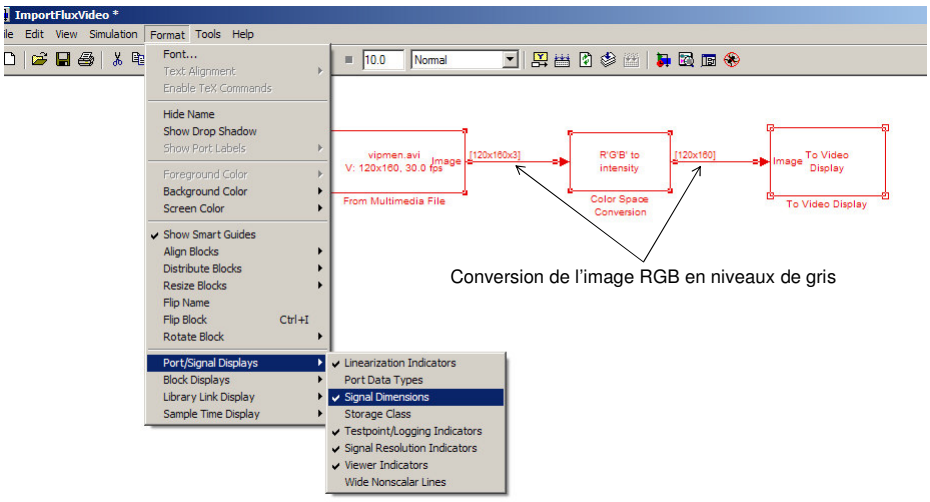
Périodes multiples en monotâche (suite)



Pas d'échantillonnage fondamental
- doit être plus grand que la somme des durées d'exécution des tâches rapides et lentes
- doit être un diviseur commun de toutes les périodes d'échantillonnage

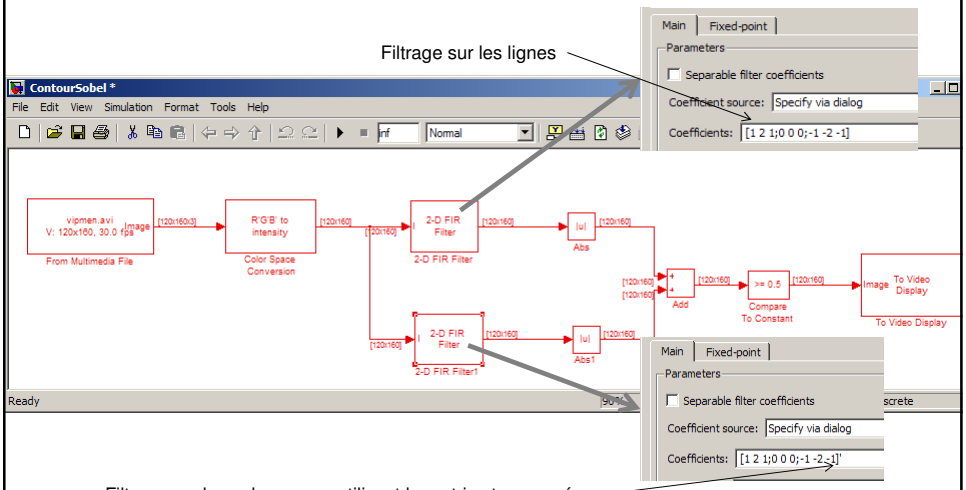
Détection de contour par filtrage de Sobel

Import d'un flux video

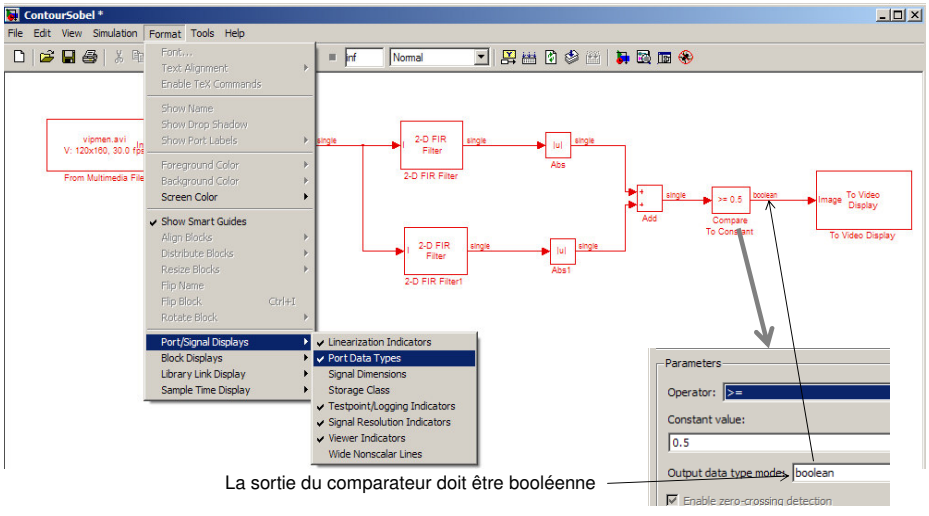


Conversion de l'image RGB en niveaux de gris

Détection de contour par filtrage de Sobel



Détection de contour par filtrage de Sobel



Import de code C

Import de code C

Pour importer un code C dans Simulink, il faut passer par la création d'une S-function. Cette création peut être automatiser apr le biais d'un script à base du legacy_code tool

Simulink supporte les pointeurs sur des entiers, des flottants mais pas des charactères. Pour importer une fonction C dans Simulink, celle-ci doit avoir un prototype avec des arguments valides constitués de pointeurs ou valeurs sur des types entiers ou flottants. Au besoin il faudra créer une fonction Wrapper acceptant des arguments valides qui appellera la fonction souhaitée utilisant des arguments non valides

Exemple de fonction code C

La fonction GradientImage utilise le type non valide PTR_PICTURE pour lire et modifier des images

```
void GradientImage( PTR_PICTURE pImgSource, PTR_PICTURE pImgGraduee ){
  register PTR_PIXEL pGraduee;
  register PTR_PIXEL pSource0;
  ...}
```

Il faut créer une fonction wrapper qui n'accepte que des pointeurs et/ou valeurs sur des types entiers ou flottants

```
void wrapperGradient(int *imgSource, int *imgLissee, int iLargeur, int iHauteur)
{
  PICTURE PictureSource, PictureLissee;

  PictureSource.iHauteur = iHauteur;
  PictureSource.iLargeur = iLargeur;
  PictureSource.pPixels  = imgSource;
  PictureLissee.iHauteur = iHauteur;
  PictureLissee.iLargeur = iLargeur;
  PictureLissee.pPixels  = imgLissee;
  GradientImage(&PictureSource, &PictureLissee);
}
```


Le legacy_code tool

Le script commence par la création d'un objet

```
def = legacy_code('initialize');
```

Puis d'un nom pour la S-function

```
def.SFunctionName = 'sfun_Gradient';
```

Puis du prototype de la fonction valide

```
void wrapperGradient(int *imgSource,
int *imgLissee, int iLargeur, int iHauteur)
```

```
def.OutputFcnSpec = 'void wrapperGradient(uint8 u1[][],
uint8 y1[size(u1,1)][size(u1,2)], int16 size(u1,1), int16 size(u1,2))';
```

u1, u2, …ui est utilisé pour les entrées 1, 2, …i
y1, y2, …yj est utilisé pour les sorties 1, 2, …j
p1, p2, …pk est utilisé pour les paramètres 1, 2, …k
u1 est utilisé pour une valeur
u1[] est utilisé pour un pointeur sur un vecteur
u1[][] est utilisé pour un pointeur sur une matrice
y1[size(u1,1)][size(u1,2)] : pointeur sur une matrice de même taille que u1
size(u1,1) pour iLargeur
size(u1,2) pour iHauteur

Le legacy_code tool (suite)

On termine le script par:
Les fichiers .C et .H nécessaires

```
def.HeaderFiles    = {'Algo.h'};
def.SourceFiles    = {'Roberts.c','WrapRoberts.c'};
```

La création de la S-function proprement dit

```
legacy_code('sfcn_cmex_generate', def);
```

La compilation de la S-function

```
legacy_code('compile', def);
```

La création d'un bloc pour Simulink

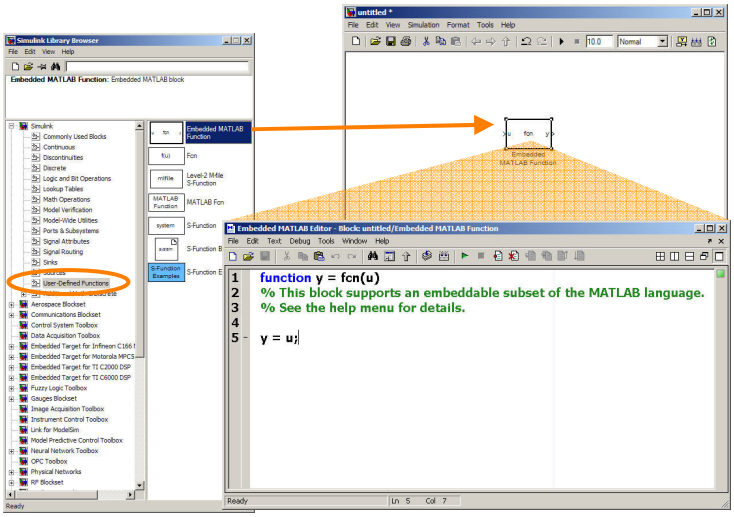
```
legacy_code('slblock_generate', def);
```

La création d'un fichier utile pour la génération de code

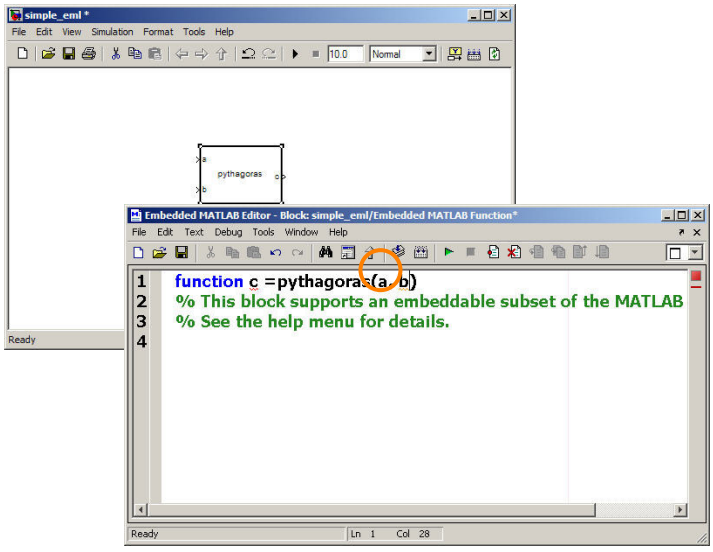
```
legacy_code('sfcn_tlc_generate', def);
```


Import de code MATLAB

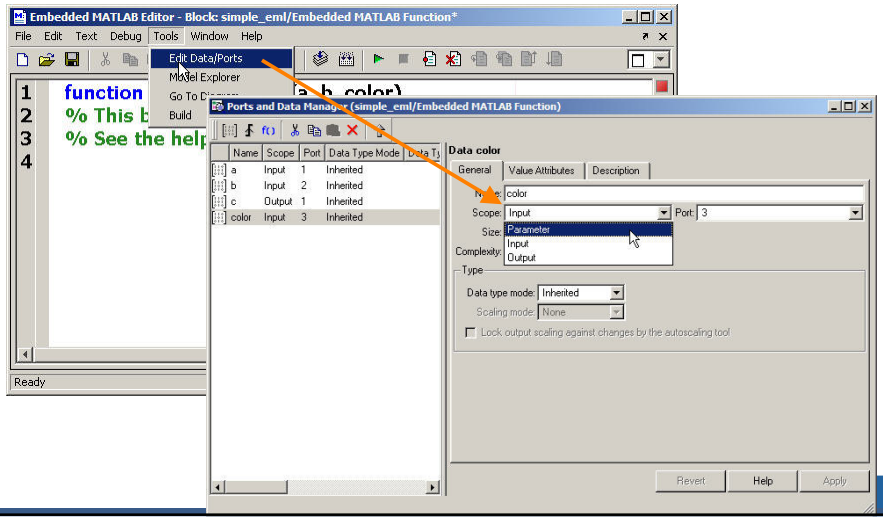
Ecrire une fonction Embedded MATLAB



Definir les entrées et sorties du bloc



Definir des paramètres

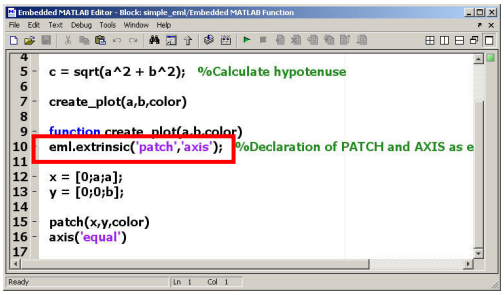


Utilisation des fonctions de la librairie Run-Time

- Sous-ensemble des fonctions MATLAB
- Même nom, arguments, et fonctionalités qu'avec MATLAB
- Generation de code C avec Real-Time Workshop

Appel de fonctions MATLAB non run-time

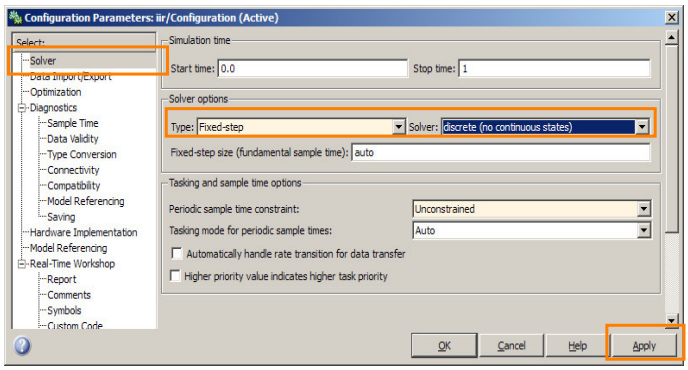
- **Executées par MATLAB pendant la simulation**
  - **La simulation sera plus lente**
- **Avant l'appel de telles fonctions, les déclarer en extrinsic**
  - **Extrinsic declaration → `eml.extrinsic`**
- **Aucun code de ces fonctions ne sera généré**



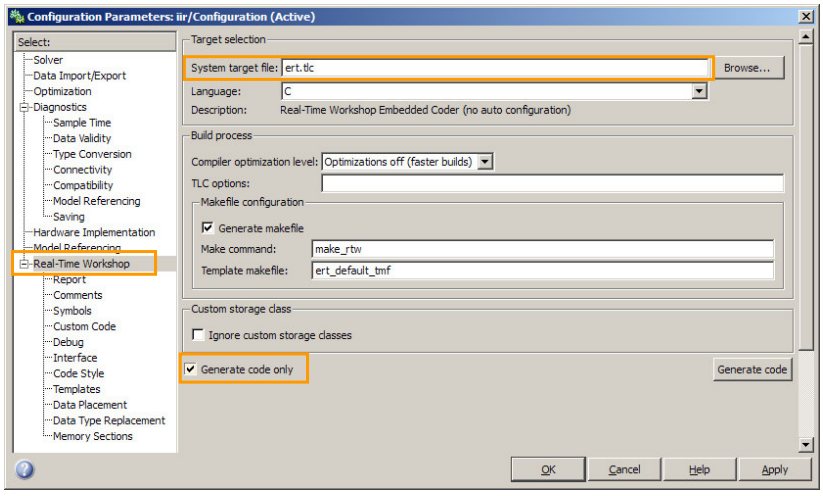
Génération de code C générique

Pré-requis pour la génération de code

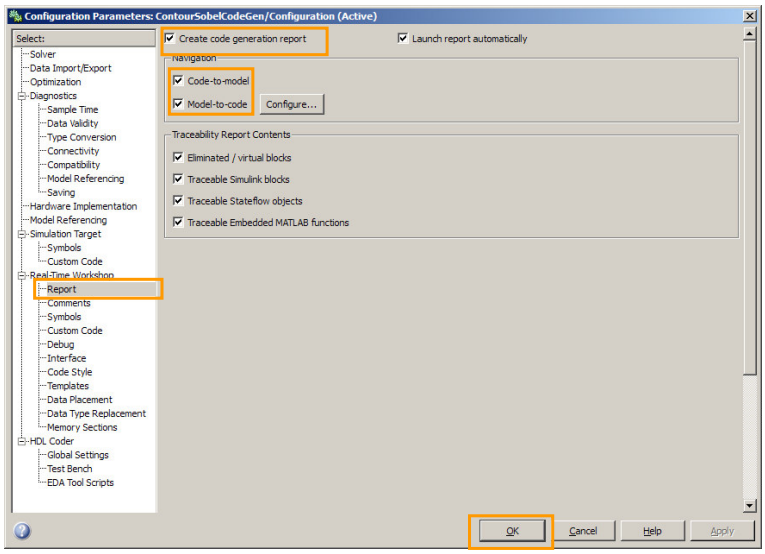
Selectionner Simulation → Configuration Parameters (Ctrl+E)



Configurer la cible

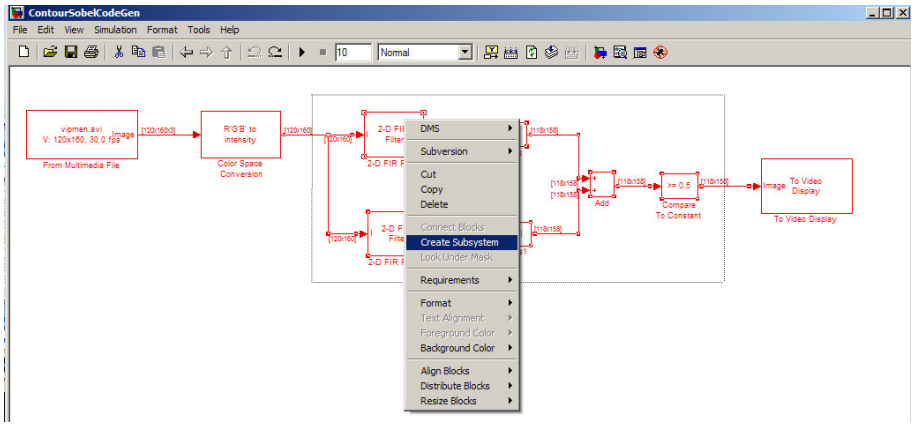


Générer un rapport de génération de code



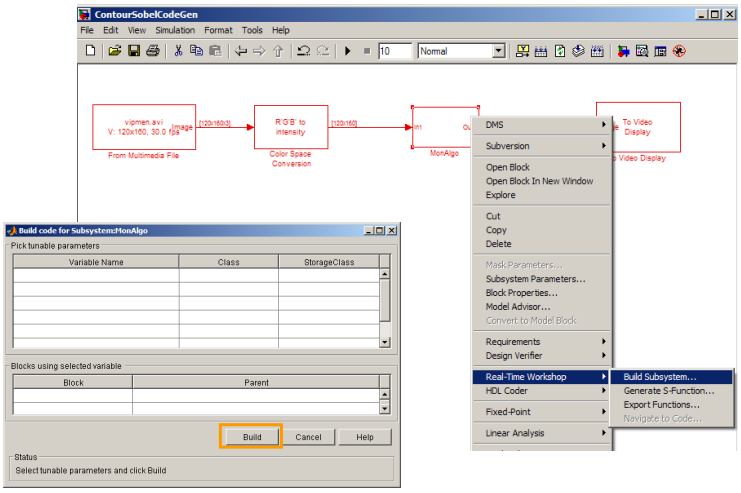
Créer un sous-système

Selectionner les blocs don't on veut générer le code
Click bouton droit sur un des blocs sélectionnés



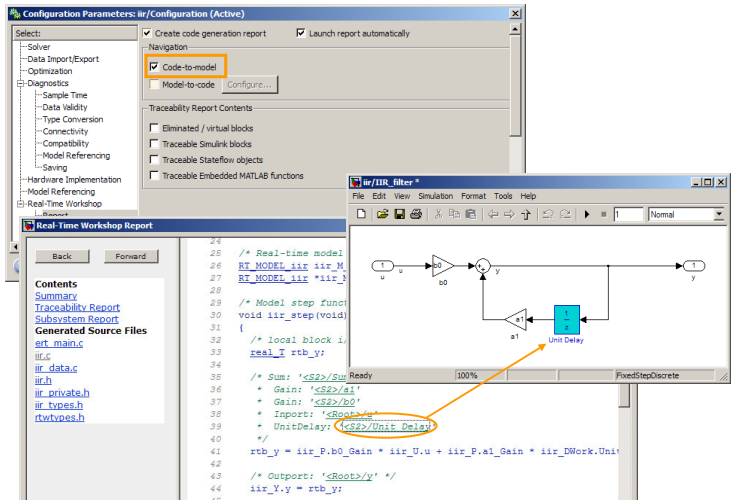
Générer le code

Click bouton droit sur le sous-système
Puis Build sur la 2ème interface



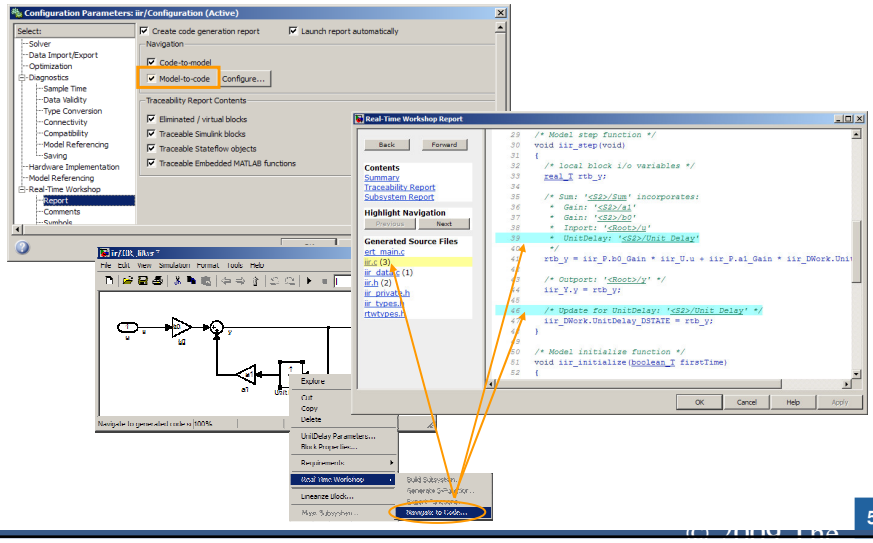
Traçabilité bi-directionnelle

- **Du code au modèle**

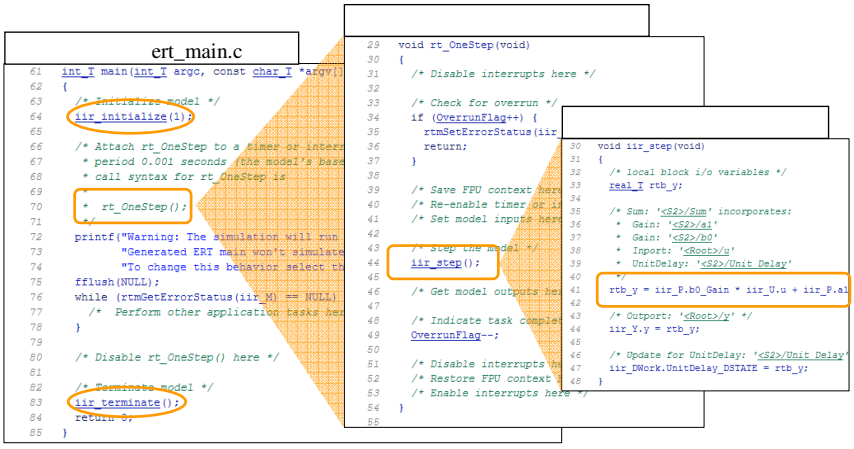


Traçabilité bi-directionnelle

- Du modèle au code



Les fichiers générés don't un main "exemple"

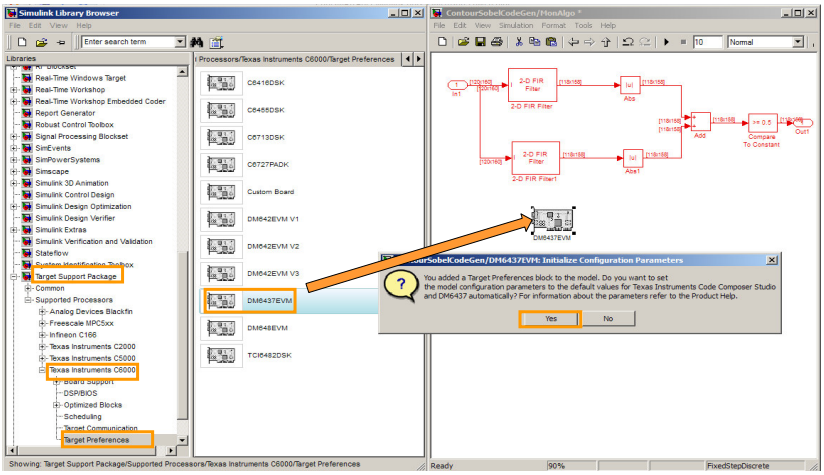


Génération de code C pour DSP TI C6000

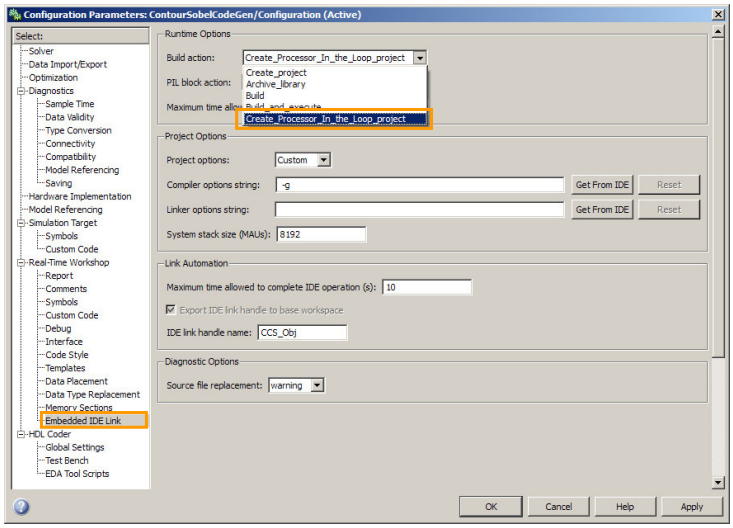
Verification de code (PIL)

Filter Algorithm PIL Verification with C6713DSK

Entrer dans le sous-système et ajouter le bloc DM6437EVM

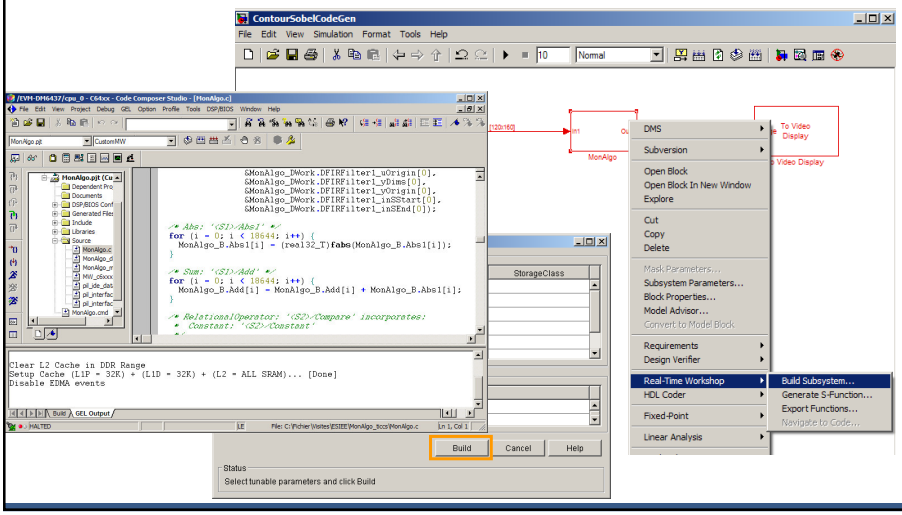


Configurer pour Porcessor in the Loop (PIL)



Générer le code

Code Composer se lance et un nouveau projet est créé, compilé, linké et téléchargé sur la cible

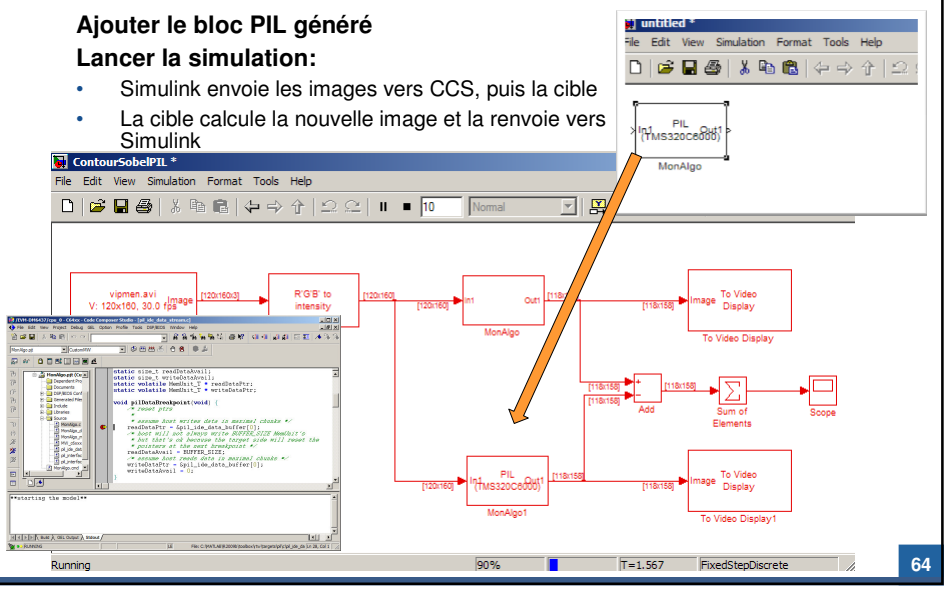


Verification avec l'EVM DM6437

Ajouter le bloc PIL généré

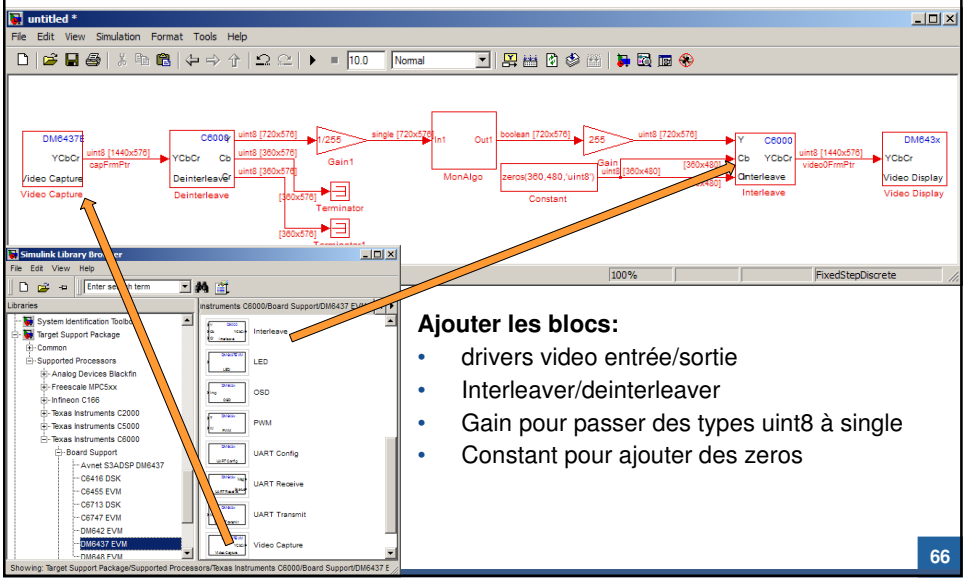
Lancer la simulation:

- Simulink envoie les images vers CCS, puis la cible
- La cible calcule la nouvelle image et la renvoie vers Simulink



Génération d'un projet CCS

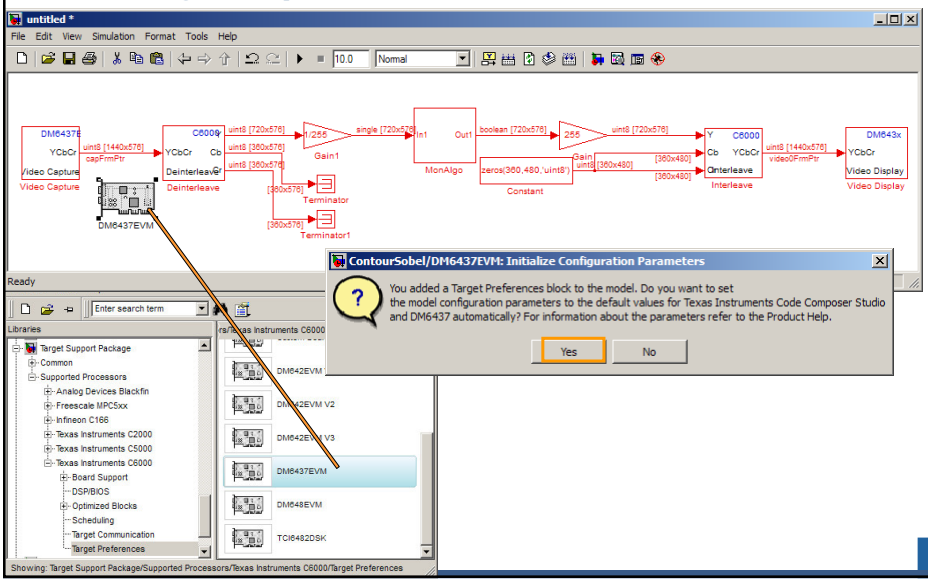
Nouveau modèle à partir du sous-système



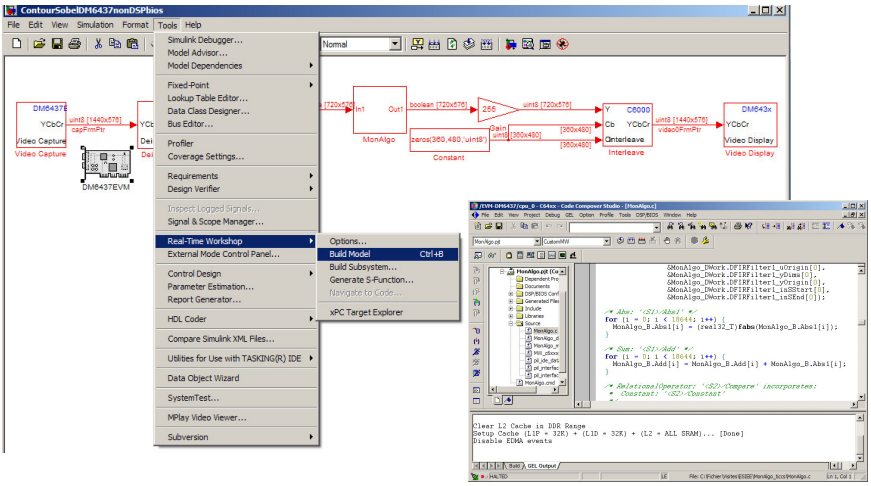
Ajouter les blocs:

- drivers video entrée/sortie
- Interleaver/deinterleaver
- Gain pour passer des types uint8 à single
- Constant pour ajouter des zeros

Configurer pour une cible DM6437

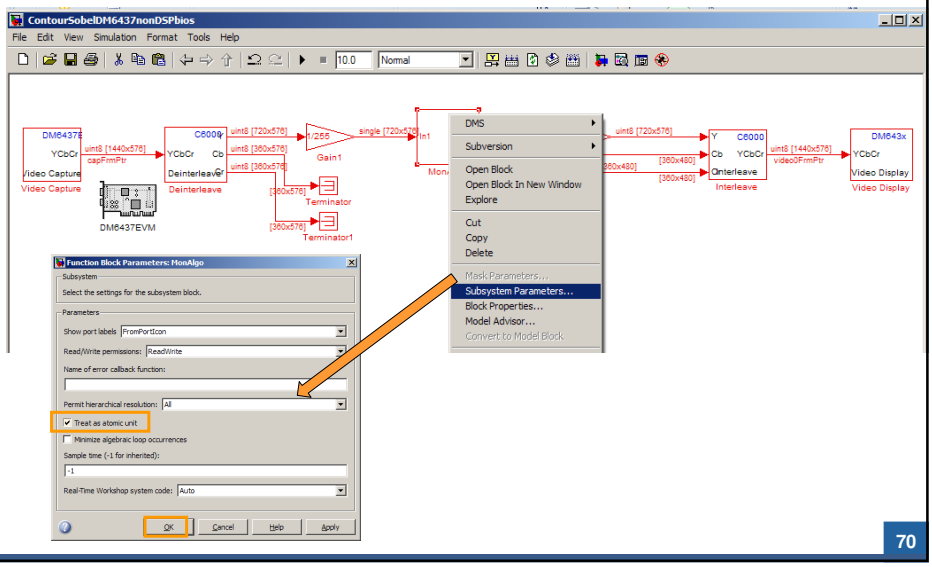


Générer et exécuter le code sur la cible

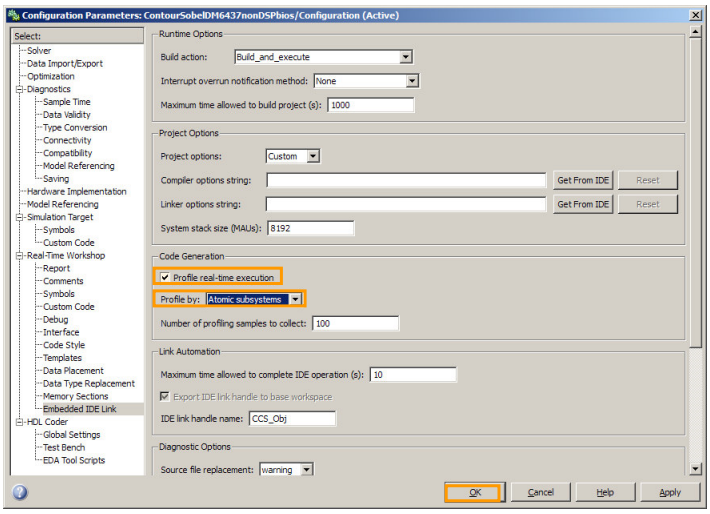


Profile de code

Rendre le bloc MonAlgo “atomic”

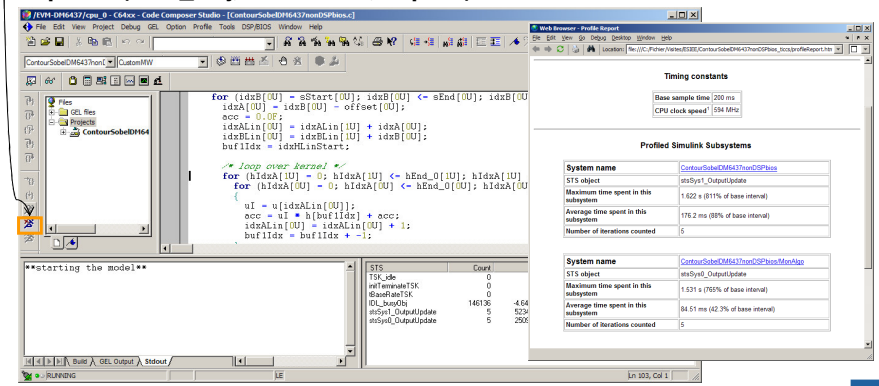


Configurer le modèle pour le profiling (Ctrl+E)

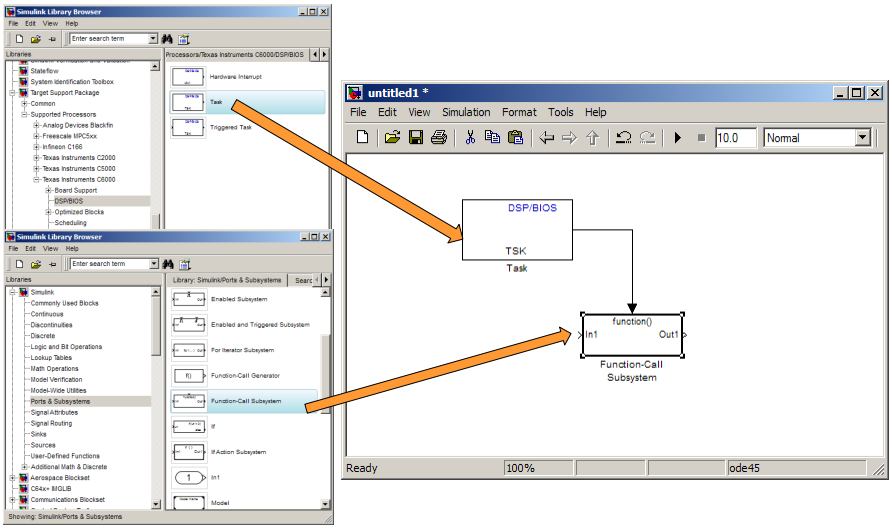


Récupérer le rapport de profile de code

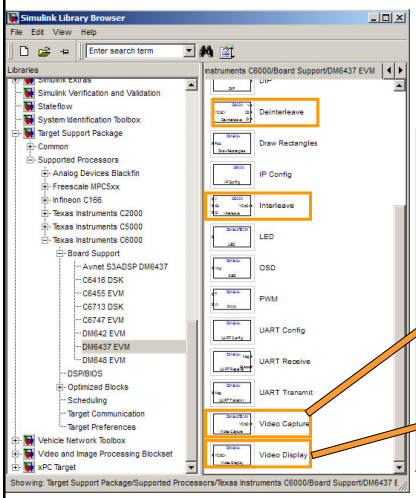
Générer et exécuter le code sur cible (Ctrl+B)
Arrêter l'exécution au bout de quelques secondes
Dans MATLAB lancer la commande :
>> profile(CCS_Obj, 'execution', 'report')



Création d'une tâche DSP/Bios

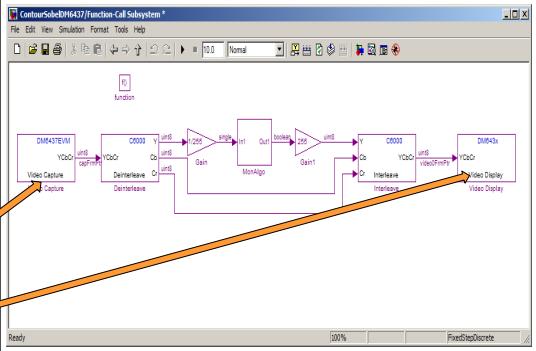


Ajout des drivers Video



Ajouter les blocs propres à la DM6437

- **drivers Entrée/Sortie Video**
- **Entrelaceur/désentrelaceur**
- La cible calcule la nouvelle image et la renvoie vers Simulink



Générer le code au niveau du modèle

Code Composer se lance et un nouveau projet est créé, compilé, linké téléchargé sur la cible et l'exécution démarre