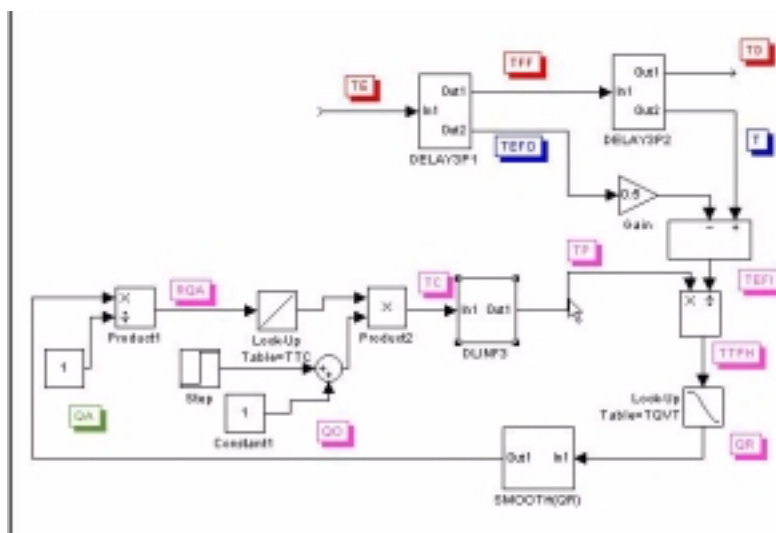


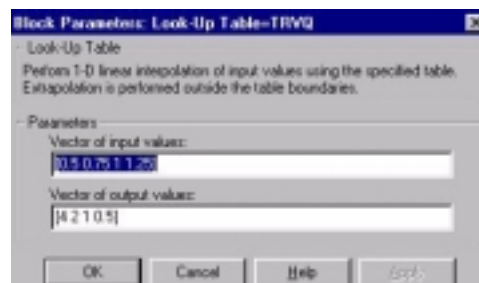
FE020

Comme pour le DELAY3 on transforme cet assemblage en sous-système. On a ainsi globalement, après jonction avec la flèche TP entrant dans le bloc donnant $TTPH$



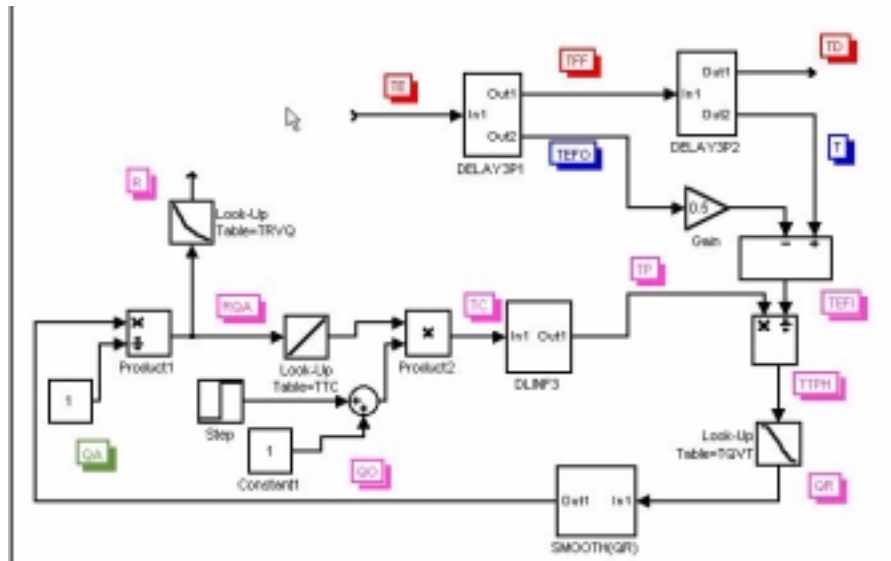
FE021

Pour ce qui est de la boucle allant vers TE , RQA sert d'intrant à une table donnant R . On introduit une Look-Up Table telle que



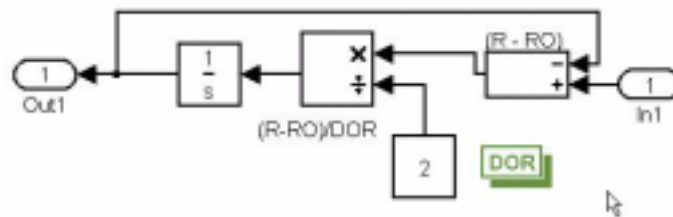
FE022

D'où le schéma global



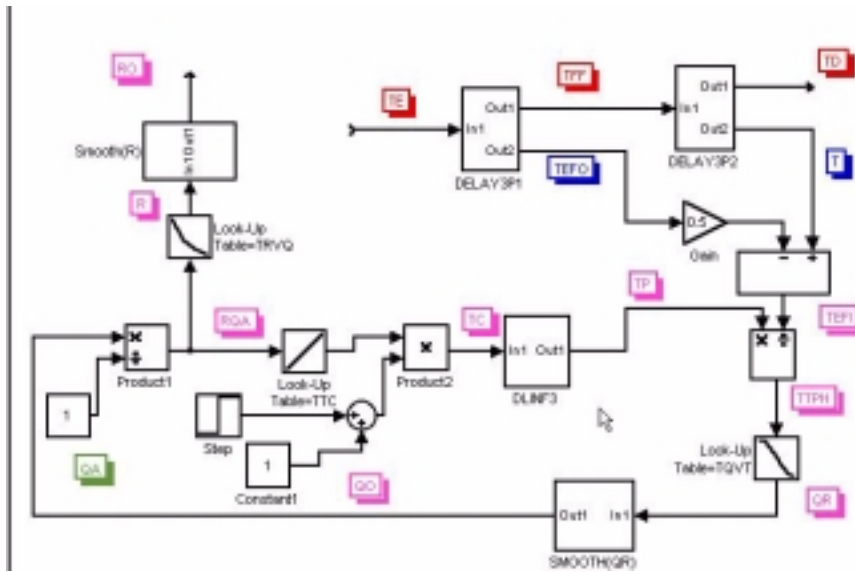
FE023

L'auxiliaire R alimente un $SMOOTH$ pour se transformer en RO . On recopie le $SMOOTH(QR)$ que l'on transforme en $SMOOTH(R)$ de façon adéquate.



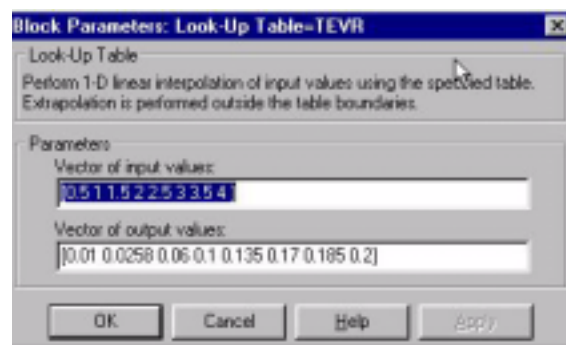
FE024

Ce $SMOOTH(R)$ est mis sous forme d'un sous-système. Il vient



FE025

L'auxiliaire RO sert d'intrant à une table $TEVR$ donnant FEM , les caractéristiques de cette Look-Up Table sont

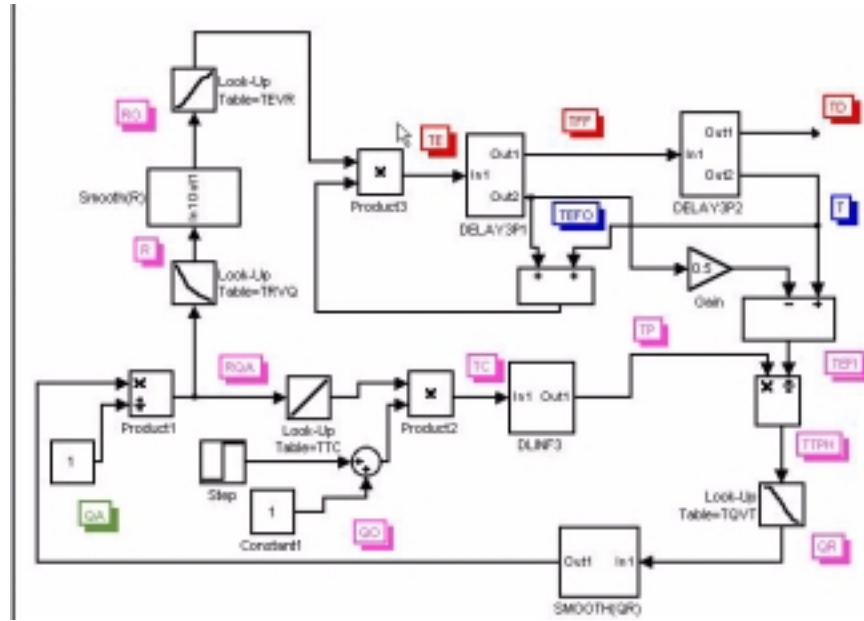


FE026

Ayant FEM on peut obtenir TE sachant que

$$TE = FEM * (TEFO + T) \quad (1, R)$$

On commence par additionner $TEFO$ et T , à l'aide d'un bloc Sum , puis on introduit un bloc $Product$ pour réaliser le produit de cette somme et FEM et on fait la jonction avec la flèche TE . On obtient ainsi le schéma global



FE027

Pour que ce schéma soit opérationnel il est nécessaire de donner des valeurs initiales aux variables d'état, c'est à dire aux différents blocs Integrator. En outre il faut sélectionner les variables dont on veut connaître l'évolution des valeurs.

3.2.2 Initialisations et sélection des variables pertinentes.

On va déduire toutes les initialisations des niveaux constituant sachant que l'on a à l'origine

$$TE_0 = 20/69 \# 0.28986$$

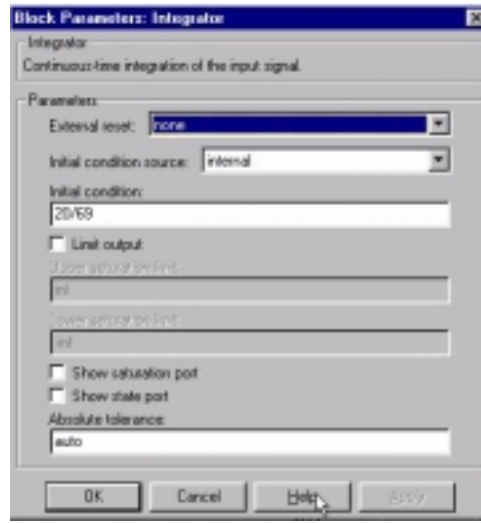
On note $L11$, $L21$ et $L31$ les trois niveaux constitutifs du $DELAY3P$ donnant $TEFO$. Si Q est le flux initial, puisque $DDF = 3$ est le délai associé, les trois niveaux sont initialisés à

$$Q * \frac{DDF}{3} = Q \equiv \frac{20}{69}$$

Ainsi on a originellement

$$TEFO_0 = \sum_{i=1}^3 Li1 = 3 * \frac{20}{69} = \frac{20}{23} \# 0.86957$$

⇒ On fait un double clic gauche sur le sous-système $DELAY3P1$, puis un double clic gauche sur chacun des blocs Integrator et on donne la valeur $20/69$ à Initial Condition.



FE028

On note $L12$, $L22$ et $L32$ les trois niveaux constitutifs du $DELAY3P$ donnant T . Le flux initial entrant est

$$TFF_0 = 20/69$$

puisque $DMS = 36$ on a pour initialisation

$$TFF_0 * \frac{DMS}{3} = \frac{20}{69} * \frac{36}{3} = \frac{80}{23} \# 3.4783$$

et donc originellement

$$T_0 = \sum_{i=1}^3 \$Li2 = 3 * \frac{80}{23} = \frac{240}{23} \# 10.435$$

$\Rightarrow$ On réitère l'opération pour le sous-système $DELAY3P2$: double clic d'ouverture et initialisation de chacun des blocs Integrator à $(20/69)*12$.

On a donc au départ

$$\begin{aligned} TEFI_0 &= T_0 - 0.5 * TEFO_0 \\ &= \frac{240}{23} - 0.5 * \frac{20}{23} = 10 \end{aligned}$$

et

$$TTPH_0 = \frac{TP_0}{TEFI_0}$$

On nous dit que

$$QO_0 = 1$$

$\Rightarrow$ double clic d'ouverture du sous-système $Smooth(QR)$ et initialisation du bloc Integrator à 1.

On a alors

$$RQA_0 = \frac{QO_0}{QA_0} = \frac{1}{1} = 1$$

et d'après la table *TTC* il vient

$$TC_0 = 1000$$

L'assemblage *DLINF3* donnant *TP* est constitué des 3 niveaux *N1*, *N2* et *N3*. Leurs valeurs initiales sont celles de la variables observée à l'origine soit

$$N1_0 = N2_0 = N3_0 = 1000$$

$\Rightarrow$ pour le sous-système *DLINF3* : double clic d'ouverture et initialisation de chacun des blocs *Integrator* à 1000.

On a donc

$$TP_0 = 1000$$

ainsi

$$TTPH_0 = \frac{TP_0}{TEFI_0} = \frac{1000}{10} = 100$$

On vérifie alors, par le biais de la table *TQVT* que

$$QR_0 = 1$$

et que donc

$$QO_0 = 1$$

Ce n'est qu'une vérification !

Pour ce qui est de *R*, par le biais de la table *TRVQ* on trouve que initialement

$$R_0 = 1$$

et donc

$$RO_0 = 1$$

$\Rightarrow$ double clic d'ouverture du sous-système *Smooth(R)* et initialisation du bloc *Integrator* à 1.

On a une ultime vérification à faire. Par le biais de la table *TEVR*

$$RO_0 = 1 \Rightarrow FEM_0 = 0.0258$$

comme

$$TE_0 = FEM_0 * (TEFO_0 + T_0)$$

on a

$$\begin{aligned} TE_0 &= 0.0258 * \left(\frac{20}{23} + \frac{240}{23} \right) \\ &= 0.0258 * \frac{260}{23} \\ &= 0.29165 \end{aligned}$$

La valeur obtenue est peu différente de

$$\frac{20}{69} \# 0.28986$$

De fait si

$$\frac{x}{69} = 0.29165$$

alors

$$x = 69 * 0.29165 = 20.124$$

Reste à savoir le pourquoi de ce 20/69 donné comme valeur initiale de TE par les concepteurs du modèle ??? Cela relève d'éventuelles simulations ultérieures

Pour ce qui est de variables pertinentes à sélectionner observons le graphe intitulé "Réponse à un accroissement de 20% des commandes" dans l'énoncé. En lisant à gauche du graphe avec quels caractères sont tracés les variables on note que les variables dont on étudie l'évolution temporelle sont

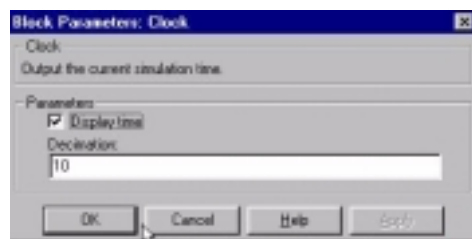
$$\begin{array}{llll} TE \text{ (HR)} & TEFO \text{ (TT)} & T & TTPH \text{ (TRM)} \\ QO \text{ (OQ)} & TC \text{ (OR)} & RO \text{ (OC)} & \end{array}$$

Pour notre part nous prendrons

$$TEFO \quad T \quad TE \quad TC \quad QO \quad RO$$

soit **6 variables**. Cela étant on introduit un bloc Mux de la librairie Signals & Systems. On fixe son nombre d'entrée à 7. De fait sa première entrée sera un bloc Clock, les 6 autres correspondant aux 6 variables sélectionnées.

On introduit un bloc Clock dont on fixe les paramètres comme suit



(Rappel : si on coche l'option Display Time le bloc change de forme et devient rectangulaire). On joint la sorte de ce bloc à la première entrée du bloc Mux.

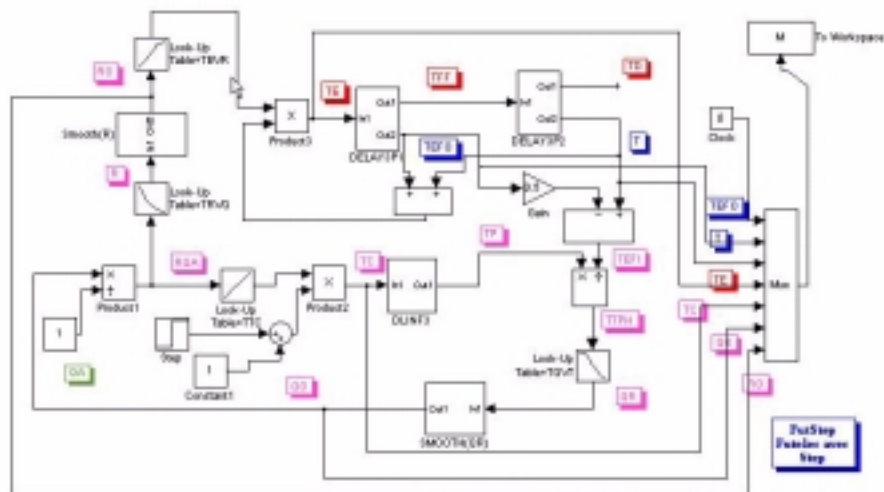
Puis en faisant des "piquages" appropriés sur les flèches on amène successivement nos 6 variables sélectionnées au bloc Mux dans l'ordre énoncé antérieurement (c'est l'ordre vertical de haut en bas pour le bloc).

On introduit enfin un bloc To Workspace dont les paramètres sont tels que



FE030

On obtient ainsi le modèle final

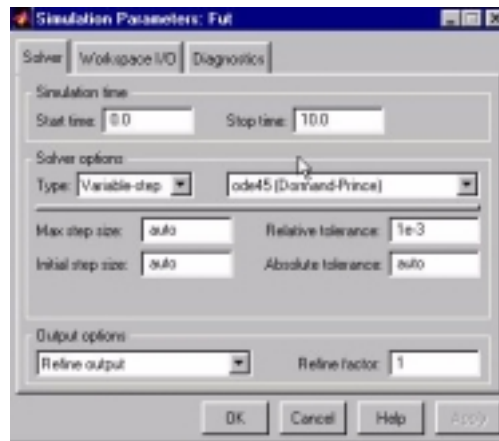


FE031

3.2.3 Quelques simulations.

Reste à savoir si nous n'avons pas commis d'erreur. Ici nous avons un moyen simple de voir ce qu'il en est. Nous avons affaire à un modèle de type micro-économique, c'est à dire que si l'on ne fait pas jouer la fonction STEP on doit rester au Steady State. En toute exactitude nos variables ne doivent enregistrer que de minimes variations, puisque l'on a vu lors de notre dernière vérification sur les valeurs initiales que l'on peut avoir une valeur de TE légèrement différente de 20/69.

Pour désactiver le STEP on fait un double clic gauche sur le bloc correspondant et on passe la valeur de *Final Value* à 0. Avant de lancer la simulation on fixe les paramètres de cette simulation comme suit



FE032

On peut penser qu'une simulation sur un laps de temps allant de 0 à 10 permet de voir si les valeurs des variables bougent ou non. On lance la simulation soit en faisant *Simulation + Start* (ou *Ctrl + T*) ou en appuyant sur l'icone *Start/Pause Simulation* dans le bandeau supérieur (►).

L'ordinateur *beep* pour signaler la fin de la simulation. On va sous la *Matlab Command Window* et on note que l'on a un avertissement : » *Warning: Unconnected output line found on block 'Fut/DELAY3P2' (output port: 1)*

De fait la sortie 1 de DELAY3P2, le flux *TD*, n'est relié à rien d'où l'avertissement, qui n'entrave en rien la simulation (On pourrait utiliser un bloc *Terminator* ou faire une entrée vers le bloc *Mux*). Pour voir ce qu'il en est on peut visualiser la matrice *M* des résultats. On tape *M* dans la *Matlab Command Window* (ce qui n'est guère recommandé si *M* est importante). A priori, hormis dans la première colonne, qui est rappelons-le celle du temps, rien ne bouge. Souvenons nous que les colonnes de *M* correspondent successivement à

time TEFO T TE TC QO RO

Les valeurs de la 4ème colonne (TE) sont égales à

$$1.0e + 003 * 0.0003 = 0.3$$

il faut faire attention que par défaut l’affichage de Matlab, le format numérique, est "short". Changeons la chose en faisant *File + Preferences + Long*. Bien évidemment l’affichage change et la première valeur de TE est

$$1.0e + 003 * 0.00029165217391$$

qui correspond bien à la valeur que nous avons calculée antérieurement. On notera que la dernière valeur est

$$1.0e + 003 * 0.00029213429860$$

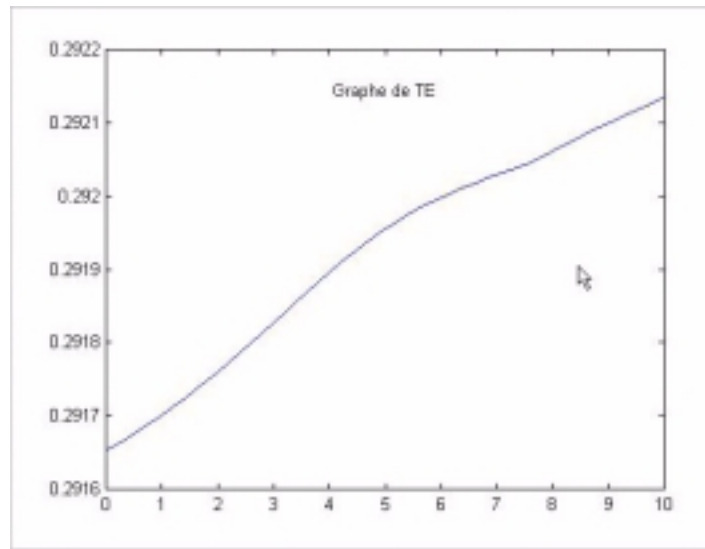
La variation n’est pas énorme.

$$\frac{0.00029213429860 - 0.00029165217391}{0.00029165217391} = 0.0016531$$

On peut agir différemment en faisant un tracé des valeurs de TE . On passe les commandes

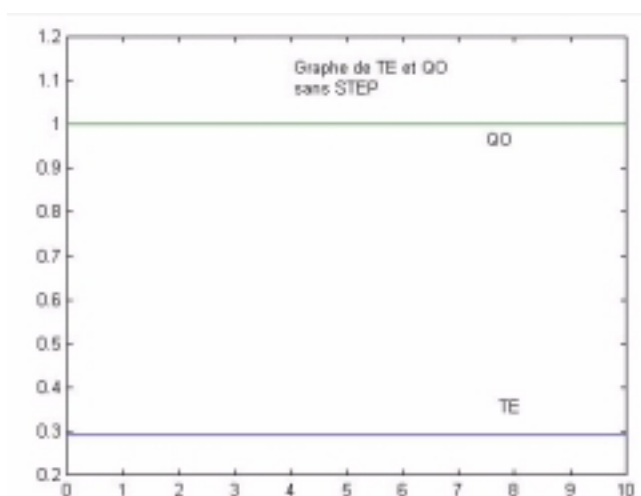
```
» plot(M(:,1),M(:,4))
» set(gcf,'color','w')
```

et on obtient le graphe



FE033

Il faut faire attention à ne pas se laisser abuser par l’échelle des ordonnées. En effet, si l’on fait un graphe simultanée de TE et QO (colonnes 4 et 6 de M) [faire `plot(M(:,1),M(:,4),M(:,1),M(:,6))`] les choses reprennent aussitôt leur juste place.



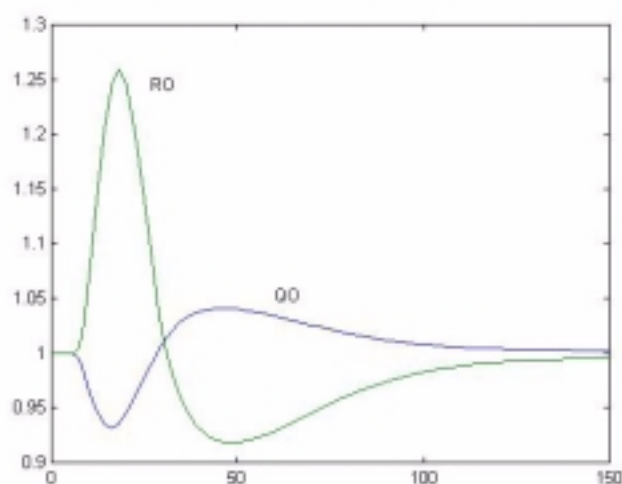
FE034

Avant de nous intéresser au problème de la valeur 20/69 donnée à TE on peut voir ce qu'il en est lorsque l'on fait agir le STEP. Pour ce faire on remet la valeur finale de ce bloc à 0.2. Dans la simulation originelle les cartes de contrôle nous disent que l'on a $LENGTH = 125$ (durée du processus de simulation). Pour notre part nous modifions les paramètres de simulation (*Simulation + Parameters*) de sorte que *Stop time* = 150.

On réalise la simulation. A titre de première vérification on peut faire un graphe simultané de la qualité observée QO (6ème colonne de M) et des réclamations observées RO (7ème colonne de M). On passe les ordres

```
» plot(M(:,1),M(:,6),M(:,1),M(:,7))
» set(gcf,'color','w')
```

On obtient le graphe suivant



FE035

Les courbes obtenues ont la même allure que celle du graphe de l'énoncé encore que l'on note un certain décalage. De fait dans le graphe de l'énoncé l'échelle de QO ($= OQ$) tracé avec la lettre Q va de 0.85 à 1.05 alors que celle de RO ($= OC$) tracé avec la lettre C va de 0.9 à 1.3. D'où un décalage des tracés dans ce graphe, puisque la valeur initiale 1, commune à ces deux variables, ne se trouve pas au même endroit de l'échelle.

Est-il possible de réaliser un tracé simultané de courbe comme dans la figure de l'énoncé. Commençons par observer la champ de variation de nos variables en passant des ordres de la forme

$$\gg \min(M(:,i)), \max(M(:,i)) \quad i \in [2, 7]$$

on a

	min	max	Δ
TEFO	0.869565	1.680099	0.81053
T	10.434783	14.517676	4.0829
TE	0.291652	0.567625	0.27597
TC	999.9804042	1249.296435	249.32
QO	0.931883	1.041080	0.1092
RO	0.918511	1.258511	0.34

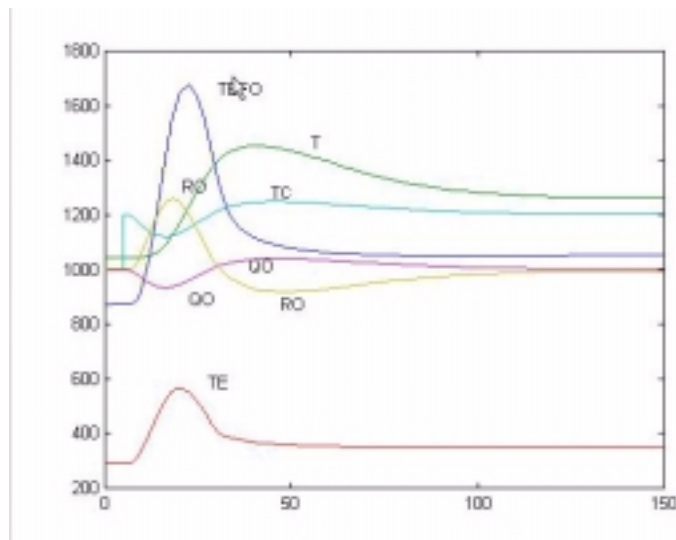
On note la disparité des variations et des valeurs absolus. On peut prendre comme échelle celle de TC qui est de l'ordre du millier et faire un tracé de

$$\begin{aligned} &1000 * TEFO \quad 100 * T \quad 1000 * TE \\ &1000 * QO \quad 1000 * RO \end{aligned}$$

Ainsi on passe l'ordre

$$\gg \text{plot}(M(:, 1), 1000 * M(:, 2), M(:, 1), 100 * M(:, 3), M(:, 1), \\ 1000 * M(:, 4), M(:, 1), M(:, 5), M(:, 1), 1000 * M(:, 6), M(:, 1), 1000 * M(:, 7))$$

On obtient la figure suivante



FE036

On obtient une figure similaire à celle de l'énoncé. Les courbes ont été nommées en observant les valeurs maximales dans les nouvelles échelles. Cependant, il vaut mieux faire plusieurs tracés en prenant à chaque fois des courbes comparables. On pourrait ainsi avoir les graphes

$TEFO$ & $T \rightarrow$ (hommes)
 $TE \rightarrow$ hommes/mois
 $TC \rightarrow$ unités/mois
 QO et $RO \rightarrow$ similitude des valeurs
 et non pas des unités

NB : On pourrait faire un graphe comparatif de TE et TD , cela impliquerait de relier la flèche donnant TD au bloc *Mux* auquel on ajouterait une entrée. Réalisons cette modification.

On peut maintenant examiner le problème de la valeur 20/69 donnée à TE . Pour formuler les choses autrement : si l'on donne des valeurs initiales quelconques aux niveaux constitutifs des *DELAY3P* obtiendra-t-on une valeur d'état stationnaire de 20/69 pour TE ?

En d'autres termes, compte tenu de l'ensemble des valeurs paramétriques employées en dehors de la ligne de flux des testeurs doit on avoir nécessairement une valeur stationnaire de TE de 20/69 en l'absence de toute perturbation extérieure ?

On peut commencer par voir si cette version de base est sensible aux diverses méthodes d'intégration.

ode45/ode23/ode113/ode15s/ode23s/ode23t/ode23tb

$\Rightarrow$ on réalise 7 simulations avec la version de base, **le STEP ne jouant pas**, sur **200 périodes**. On stocke chaque vecteur TE obtenu dans une même matrice ($CompTE$) afin de faire une comparaison. Mais on retrouve un problème déjà évoqué. En réalisant une simulation avec Simulink on utilise le solveur *ode45* et si l'on n'y prend pas garde lorsque l'on fait varier les initialisations les vecteurs colonnes renvoyés peuvent varier en dimension.

De fait dans l'état actuel des choses (le paramètre *Sample Time* du bloc *To Workspace* est à la valeur -1) si on réalise une simulation on obtient une matrice M est de format (81×8) .

On va donc fixer le Paramètre *Sample Time* à 1. rappelons que

Le fait de choisir un Sample Time unité revient à faire comme si on avait un solveur à pas constant qui travaille à l'unité de temps, alors même que le solveur est à pas variable.

On définit

$$compTE = \text{zeros}(201, 7)$$

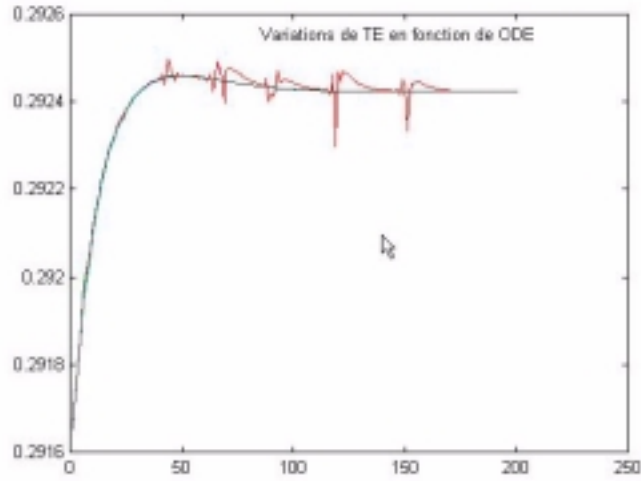
(!!) on a 201 lignes car *start time* = 0 et *stop time* = 200 et

$$(200 - 0) + 1 = 201$$

Le stockage se fait à l'aide du petit programme suivant

```
%CompTE
%Programme de stockage des vecteurs TE dans une matrice
i = input('n° de simulation ')
compTE(:,i)=M(:,4);
```

On effectue les 7 simulations et l'on prend la précaution de sauver l'espace de travail (sinon on devrait tout refaire). Notons Fut1 cette MAT-file. (*.mat). L'ordre *plot(compTE)* nous donne le graphe suivant



FE037

On constate les diverse méthodes d'intégration donnent (à peu près) la même chose à l'exception de l'une d'entre elles. Pour la repérer on fait tout bêtement `plot(compTE(:,i))` en faisant varier i de 1 à 7. On trouve qu'il s'agit de la troisième méthode `ode113`. Il vaut donc mieux éviter cette méthode qui engendre de faibles "oscillations".

Pour ce qui est des valeurs maximales enregistrées l'ordre `max(compTE)` nous donne

ode45	0.29246002365432
ode23	0.29246002404317
ode113	0.29249729529584
ode15s	0.29245909819575
ode23s	0.29246004136651
ode23t	0.29246005776949
ode23tb	0.29246006004883

Sauf pour le troisième solveur on note les variations plus que minime (de l'ordre de 10^{-8}). Par la suite on peut s'en tenir à `ode45`.

Pour en revenir à notre problème on ne fait pas jouer le bloc `STEP` et on réalise un ensemble de simulations en faisant varier les initialisations des niveaux constitutifs de `DELAY3P1` et `DELAY3P2`. On peut se fixer le plan de travail suivant on suppose que initialement `TEFO` peut varier de 0 à 4 et T varie de 1 à 4. De fait on a

$$TEFI = T - 0.5 * TEFO \quad (4, A)$$

si $T = 0$ l'auxilaire est négative et une simulation effectuée dans ce cas conduira au message

Warning: Unable to meet integration tolerances without violating a minimum machine step size of 1.252297431387592e-015 at time 0.352490390334638 for model 'Fut'. It appears that there is a singularity in the solution. If not, try reducing the step size (either by reducing the fixed step size or by tightening the error tolerances).

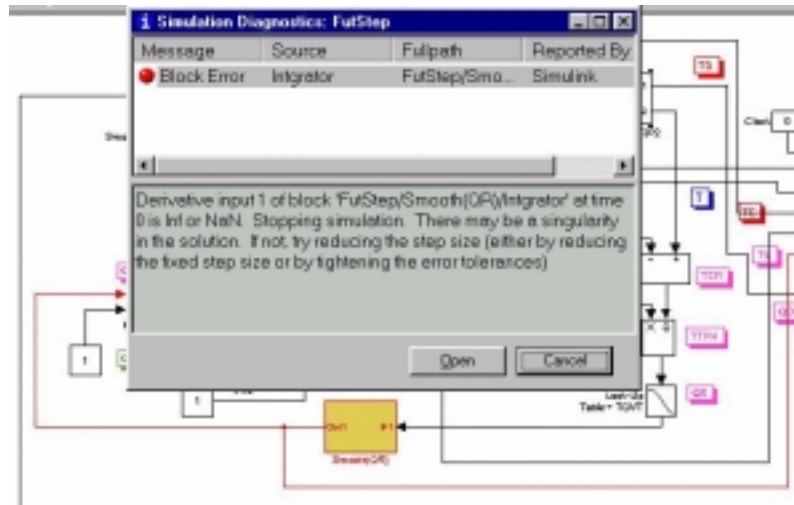
La simulation ne s'effectuera pas. Ainsi T ne peut jamais être nul ! Par ailleurs on ne peut avoir

$$T = 0.5 * TEFO \Rightarrow TEFI = 0$$

car alors

$$TTPH = \frac{TP}{TEFI} \rightarrow \infty$$

C'est à dire que pour Matlab $TTPH$ n'est pas un nombre, ce qui se répercute sur la table $TQVT$ et donc sur l'entrée du SMOOTH. On a un blocage de simulation qui se traduit comme suit à l'écran



FE038

(Noter le passage en jaune du bloc SMOOTH et le passage en rouge de la flèche associée). En outre dans la Fenêtre de commande de MATLAB on a le message

Warning: Division by zero in 'FutStep/Product1'.

Si on dresse un tableau à double entrée, en portant les valeurs de $TEFO$ en ligne et celle de T en colonne, on met dans le coeur du tableau les numéros

de simulations selon un codage assez évident.

	T			
TEFO	1	2	3	4
0	1	2	3	4
1	5	6	7	8
2	∅	9	10	11
3	∅	12	13	14
4	∅	∅	15	16

On note que, dans chaque cas, il faudra donner aux niveaux constitutifs des *DELAY3P* des valeurs égales au 1/3 de *TEFO* ou *T*. Pour pouvoir exploiter les résultats de nos simulations il faut pouvoir les stocker. On sait que à chaque simulation la matrice *M* est renvoyée dans l'espace de travail de Matlab, toute nouvelle simulation écrase les anciennes valeurs de *M*. Encore une fois on va recourir à un tableau multidimensionnel (un "cube") Sa première dimension (*time*) représente les lignes d'une matrice *M*, la deuxième dimension correspond au nombre de colonnes de *M*, la troisième correspond à une simulation donnée.

On dimensionne un "cube" de résultat bourré de zéros soit

$$res = zeros(201, 8, 16)$$

(!!) on a 201 lignes. A chaque simulation n°*i* on demande le nombre de lignes de *M* par

$$q = size(M(:, 1), 1)$$

ce nombre de ligne doit être constant et égal à 201, et l'on fait exécuter un programme (une *M-file*) "Stockage" du type

```
%Stockage
i = input('n° de simulation ');
q = size(M(:,1),1);
for j = 1:8
    res(1:q,j,i)=M(:,j);
end
```

On effectue les 16 simulations et l'on prend la précaution de sauver l'espace de travail (sinon on devrait tout refaire). Reprenons Fut1 pour cette MAT-file. (*.mat)

On s'intéresse à *TE* soit à *res(:,4,1:16)*. On crée une matrice *TE* (201×16) à l'aide du programme *PgTE*

```
% pgTE
for i=1:16
    TE(:,i)=res(:,4,i);
end
```

Le fait de transformer une coupe du cube en matrice à 2D permet d'utiliser un ordre du type

`plot(TE)`

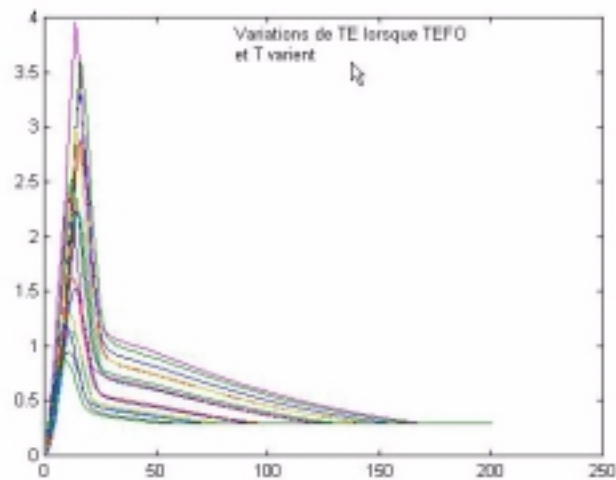
alors que l'ordre

`plot(res(:,4,1:16))`

conduit au message

??? Error using ==> plot
Data may not have more than 2 dimensions.

Si l'on fait `plot(TE)` on obtient le graphe



FE039

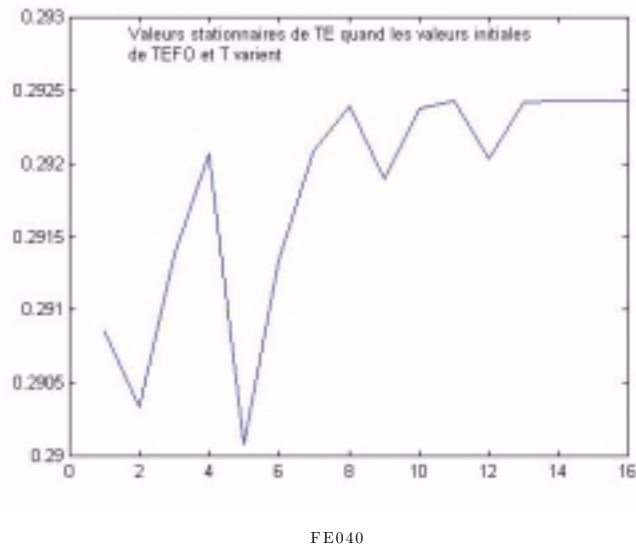
Ce graphe n'est guère lisible (on superpose 16 courbes) mais il a le mérite de montrer la convergence vers une sorte de "valeur commune". Si on tape l'ordre suivant en ayant pris un format numérique *short E*

» `TE(201,:)`

il vient

```
ans =
Columns 1 through 6
2.9085e-001 2.9033e-001 2.9137e-001 2.9206e-001 2.9008e-001 2.9134e-001
Columns 7 through 12
2.9208e-001 2.9239e-001 2.9189e-001 2.9238e-001 2.9243e-001 2.9204e-001
Columns 13 through 16
2.9242e-001 2.9243e-001 2.9242e-001 2.9243e-001
```

soit graphiquement [`plot(TE(201,:))`]



La valeur commune est de l'ordre de $0.29 \# 20/69$, sans pour autant être exactement égal à ce terme fractionnaire. Le $20/69$ n'est pas une fatalité. Regardons de plus près ce qu'il en est avec notre tableau à double entrée

	T			
TEFO	1	2	3	4
0	0.290854	0.290333	0.291368	0.292063
1	0.290080	0.291337	0.292082	0.292389
2	∅	0.291889	0.292376	0.29242554
3	∅	0.292036	0.292419	0.29242776
4	∅	∅	0.29242486	0.29242762

On peut réaliser un tracé de la surface correspondante. Pour simplifier la vision on remplace les valeurs inexistantes ($\emptyset$) par 0.29. On note S la matrice (5×4) ainsi formée. Sous Matlab faire

```
» TE(201,:)
```

```
» S=[s1 s2 s3 s4; s5 ... ,s16]
```

où les s_i sont les valeurs successives de $TE(201,:)$ que l'on copie et l'on colle de façon adéquate.

C'est un peu long et beta, mais cela a le mérite de la simplicité !

On passe les ordres

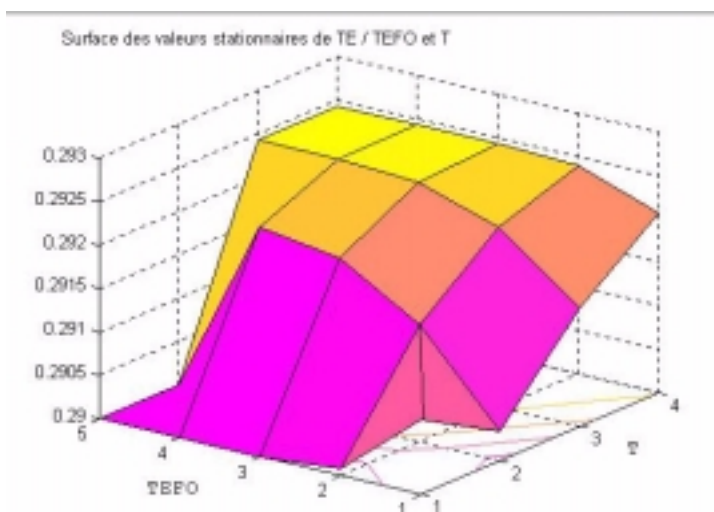
```
» surf(S)
```

```
» set(gcf,'color','w')
```

```
» axis([1 4 1 5 0.29 0.293])
```

```
» colormap (spring)
```

Faire attention à "axis", x correspond aux colonnes de S (T) et y aux lignes ($TEFO$). Si x tout comme T varie de 1 à 4, y lui varie de 1 à 5 (n° de lignes en fait). On a le graphe



FE041

On note que la surface n'est pas simple notamment dans les premières variations de $TEFO$ et T . Le point $(x, y) = (1, 1)$ correspond à un maximum relatif et à terme on a une surface qui tend à être concave. On pourrait faire une étude plus poussée : faire varier T sur $[1, 2]$ par pas de 0.25 et $TEFO$ sur $[0, 1]$ avec le même pas. Aller voir ce qui se passe pour des valeurs de T et $TEFO$ égales à 5.

La valeur de convergence de la surface semble être de l'ordre de 0.29242. Pour vérifier cette intuition faisons une rapide simulation avec

$$TEFO_0 = T_0 = 5$$

on obtient

$$TE_{200} = 0.29242668119956$$

Il semblerait que la surface redescende (?) [A voir éventuellement]

On note par ailleurs que la convergence vers une valeur de 0.29 s'accompagne d'un processus d'ajustement qui fait que TE peut atteindre des valeurs proches de 4. En termes de maximum on a le tableau suivant

	T			
TEFO	1	2	3	4
0	3.305052	3.593023	2.876248	2.235917
1	3.954490	2.983244	2.232762	1.526760
2	∅	2.519379	1.607984	1.123524
3	∅	2.391127	1.293437	0.945926
4	∅	∅	1.187112	0.877295

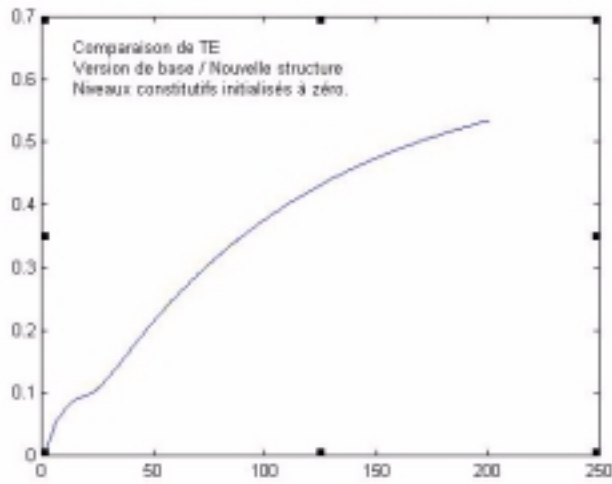
Ici encore on a une évolution non linéaire. Notons $S1$ la nouvelle matrice (5×4) formé à partir de ces valeurs, en portant à 0 les valeurs inexistantes. On la surface

On note la faiblesse de la différence dont le champ est

$$[-7.4008 * 10^{-7}, 3.0269 * 10^{-7}]$$

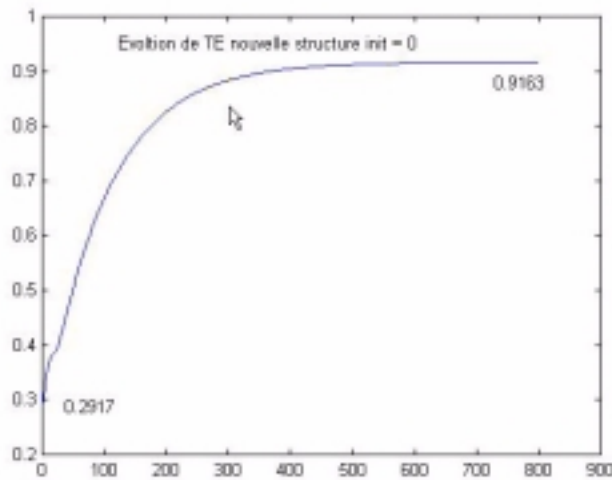
On peut aussi regarder ce qu'il en est si on initialise $TEFO$ et T respectivement à 60/69 et 240/23, en mettant à zéro les niveaux constitutifs des DELAY3. Ce faisant on présuppose que la faible erreur ainsi commise doit s'estomper au fil du temps.

On recommence un tracé comparatif de $M(:,4)$ avec $compTE(:,1)$ qui montre que notre présupposé est faux.



FE046

De fait le flux TE tend à croître à taux moins que proportionnel. Il faut réaliser une simulation sur 800 périodes (!) pour obtenir une valeur stable.

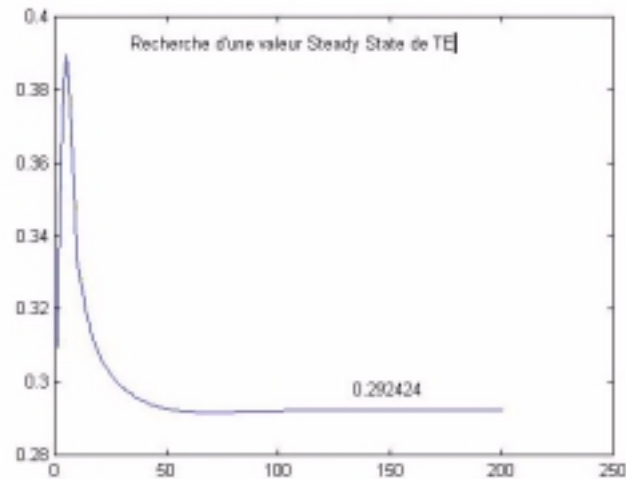


FE047

On peut penser que cette évolution tient à ce que l'on déconnecte d'une certaine façon la ligne de flux du DELAY3 et celle du niveau parallèle en ne prenant pas des initialisations cohérentes.

⇒ on a un très sérieux problème d'initialisation pour des modèles de ce type lorsqu'on ne connaît pas la valeur initiale du flux. Comment faire ?

Supposons que l'on prenne $TEFO = 3$ et $T = 9$. On initialise chacun des niveaux constituant des DELAY3 respectifs au 1/3 de ces valeurs. On réalise une simulation sur 200 périodes et on obtient l'évolution suivante de TE



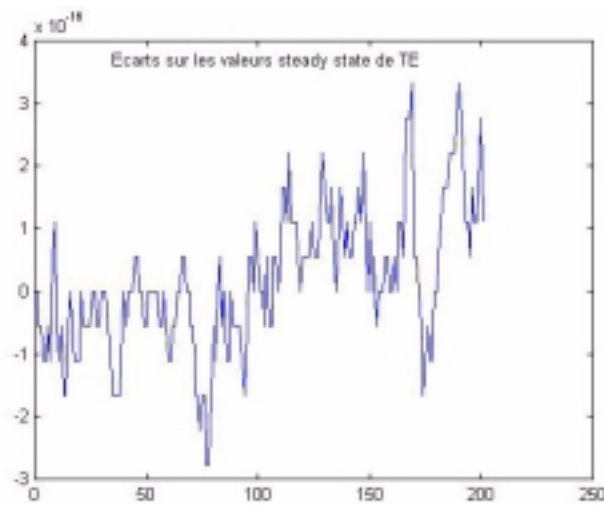
FE048

On trouve une valeur stationnaire de l'ordre de 0.292424, qui est bien de l'ordre des valeurs antérieures. **On voit bien ainsi qu'il faut maintenir un lien entre les DELAY3 et les niveaux parallèles.**

Sauvons la matrice M courante dans une matrice $M1$. Et reprenons le modèle avec les DELAY3P. On réitère l'opération, mais désormais on ne peut qu'initialiser les niveaux constitutifs de ces assemblages. On recommence une simulation sur 200 périodes et on sauve la matrice M courante dans une matrice $M2$. Puis on fait

» `plot(M2(:,4)-M1(:,4))`

il vient



FE049

On n'enregistre que de faibles écarts puisque les valeurs de l'axe des ordonnées sont multipliées par 10^{-16} !

3.2.4 Un passage par le discret.

On peut voir ce que donne l'introduction de fonctions de type "pipe". Celles-ci sont disponibles dans simulink sous deux formes

- Transport delay,
- Variable transport delay

Le bloc "*Transport Delay*" décale l'input selon un certain délai de temps spécifié. Au départ de la simulation on sort du bloc le paramètre *Initial Input* jusqu'à ce que le temps de la simulation dépasse la paramètre *Time delay*, moment où le bloc commence à engendrer l'input décalé. Le paramètre *Time delay* doit être non négatif.

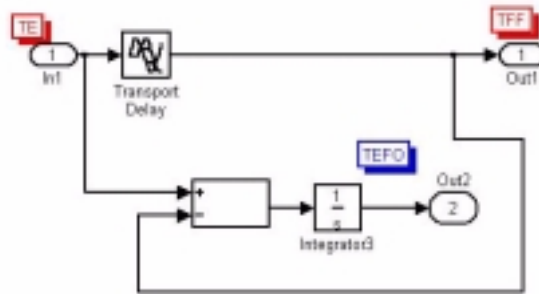
Le bloc "*Variable Transport Delay*" possède deux entrées. Il applique un décalage au premier signal d'entrée. La seconde entrée sert à spécifier le délai de temps. Les paramètres du bloc sont ainsi *Maximum delay* : la valeur maximale du délai de temps. Cette valeur ne peut être négative, sa valeur par défaut est de 10 et *Initial Input* (même sens que précédemment).

Nous utilisons ici des blocs Transport Delay à l'intérieur de ce qui était auparavant les DELAY3P. Nous sauverons notre modèle sous le nom *FutDis* (Future Electronics Discret).

Soulignons que de fait nous allons travailler en temps mixte, la ligne de flux des testeurs étant soumise partiellement à un temps discret.

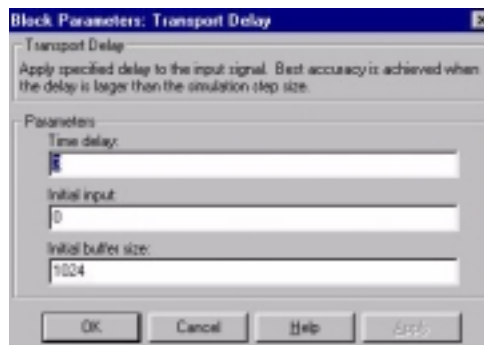
Nous ferons les **hypothèses de travail** suivante : initialement on a 9 testeurs et 0 testeurs en formation.

Le sous-système DELAY3P1 est transformé en sous-système TD1 qui a la forme suivante



FE050

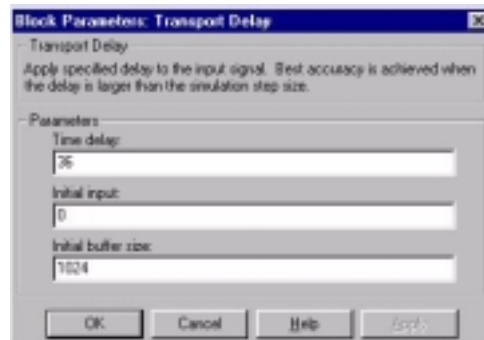
Les paramètres du bloc *Transport delay* sont



FE051

Le bloc Integrator (*TEFO*) est initialisé à 0 : à l'origine on n'embauche pas et n'on n'a pas de testeurs en formation.

On fait subir une transformation similaire au sous-système DELAY3P2 qui devient le sous-système TD2. Les paramètres du bloc *Transport delay* sont



FE052

Le bloc Integrator (T) est initialisé à 9 : à l'origine on a 9 testeurs qui quitteront l'entreprise au bout de 36 mois.

Pour vérifier si notre conception a priori est bonne lançons une simulation sur 50 périodes avec le solveur *ode45*, **sans action du STEP**. Rappelons que la structure de la matrice M selon ses colonnes est, dans l'état actuel des choses

time TEFO T TE TC QO RO TD

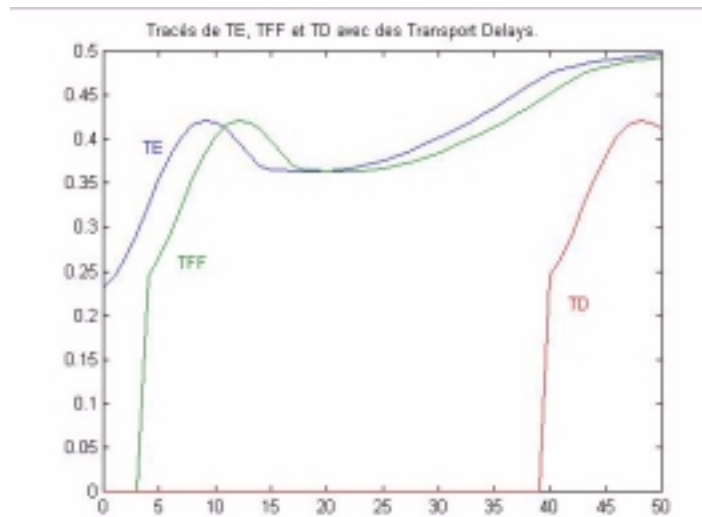
On réorganise les sorties de la façon suivante. On supprime l'ancien bloc Mux, on en prend un nouveau à **9 entrées** qui seront dans l'ordre

time TEFO T TE TFF TD TC QO RO

Réalisons un tracé simultané des 3 flux TE TFF TD . On passe les ordres

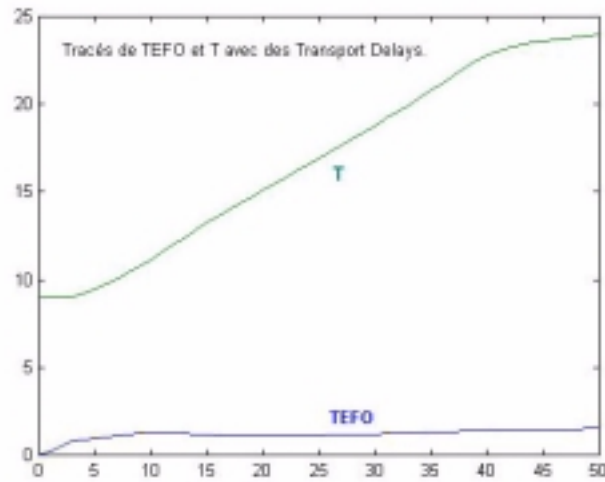
```
» plot(M(:,1),M(:,4),M(:,1),M(:,5),M(:,1),M(:,6))
» set(gcf,'color','w')
```

Il vient



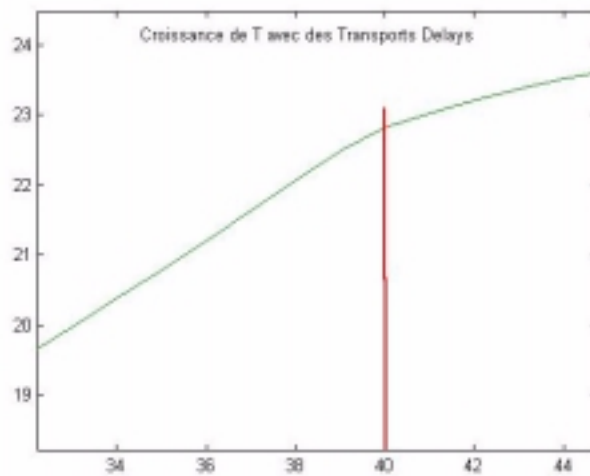
FE053

Les décalages sont bien visibles. TFF passe à 0.2442 en $t = 4$ (instant 0 + délai = 3). Pour TD il passe à 0.2442 en $t = 40$ (instant 0 + délai 3 + délai = 36). Ces décalages se répercutent bien évidemment sur l'évolution de $TEFO$ et T .



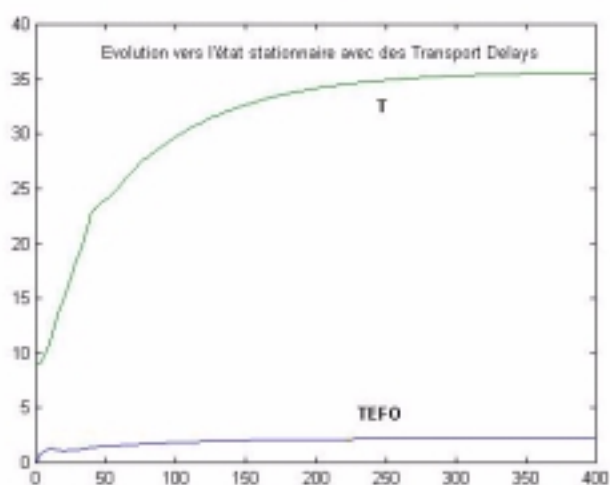
FE054

De $t = 0$ à $t = 3T$ reste constant et égal à 9, au delà il croît de façon quasi-linéaire (quasi-constance de TFF) jusqu'en $t = 40$, où il commence à croître à taux moins que proportionnel (influence de TD) cela est bien visible sur le graphe agrandi (utiliser le *Zoom In*, loupe avec un +, dans le graphe antérieur)



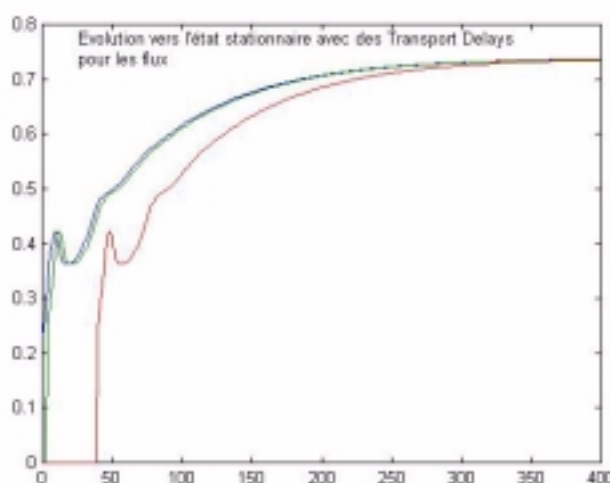
FE055

On peut maintenant faire une simulation sur 400 périodes, pour voir comment le système se stabilise (la durée a été choisi après un premier essai sur 200 périodes !). La variable T se stabilise au delà de 350, aux environs de 35.5, pour $TEFO$ la stabilisation est quasiment acquise en $t=250$ (2.18 pour 2.22 in fine).



FE056

En ce qui concerne les flux de testeurs on a



FE057

La convergence vers une valeur commune (pourquoi cette unicité de valeur ?) compte tenu des décalages sous-jacents est patente. Les valeurs finales des 3 flux sont très précisément

TE	0.73692118
TFF	0.73683224
TD	0.73542993

soit une valeur approchée de 0.73, qui n'a plus rien à voir avec le 0.29 antérieur.

On peut penser que l'action de la fonction STEP va peut-être perturber les choses. Pour bien voir ce qui se passe on fixe les paramètres du bloc comme suit

<i>Step time</i>	20
<i>Final value</i>	0.2

Avant de refaire une simulation sur 400 périodes on sauve le Workspace sous *Fut2*. On effectue la simulation. Un tracé des nouvelles valeurs des niveaux *TEFO* et *T* ne laisse pas apparaître de grosses différences. Cela étant comment faire une comparaison des valeurs avec et sans le STEP ? Il ne sert à rien de charger le Workspace *FUT2.mat*, puisque ce faisant on écrase l'espace de travail courant. On contourne la difficulté comme suit

Créons une matrice *niv=zeros(401,4)*. Sauvons les valeurs présentes de *TEFO* et *T* dans les deux premières colonnes de cette matrice par les ordres

```
» niv(:,1)=M(:,2);
» niv(:,2)=M(:,3);
```

Repassons à 0 le paramètre *Final value* du bloc STEP. Refaisons une simulation et passons les ordres

```
» niv(:,3)=M(:,2);
» niv(:,4)=M(:,3);
```

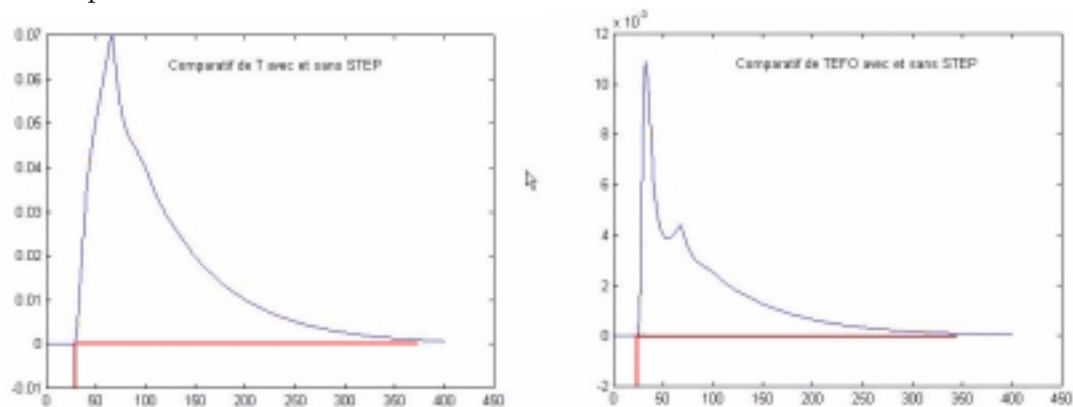
On peut alors faire le tracé graphique pour *T* avec et sans STEP

```
» plot(niv(:,2)-niv(:,4))
```

et celui pour *TEFO* avec et sans STEP

```
» plot(niv(:,1)-niv(:,3))
```

Soit respectivement



Le choc engendré par le STEP en $t = 20$ est bien visible, il entraîne une croissance consécutive des niveaux suivie d'un retour aux valeurs antérieures. On notera malgré tout la faiblesse des écarts puisque

$$\begin{aligned}\max(\text{niv}(:,2)-\text{niv}(:,4)) &= 0.069887... \\ \max(\text{niv}(:,1)-\text{niv}(:,3)) &= 0.010887...\end{aligned}$$

Pour bien voir la différence avec le cas continu pur, il convient de faire une comparaison dans les mêmes conditions d'initialisation. De fait ici nous avons perturbé avec le STEP un processus non stable. On revient au cas antérieur où $TE_0 = 20/69$.

Pour le cas continu on prend le modèle avec des DELAY3 et des niveaux associés. Les conditions initiales sont

$$\begin{aligned}TEFO_0 &= 60/69 \\ L11_0 = L21_0 = L31_0 &= 20/69 \\ T_0 &= 240/23 \\ L12_0 = L22_0 = L32_0 &= 80/23\end{aligned}$$

Pour le cas mixte (continu + discret)

→ dans TD1 on prend $TEFO_0 = 60/69$ et pour le Transport delay

$$\begin{aligned}\text{Time delay} &= 3 \\ \text{Initial input} &= 20/69\end{aligned}$$

→ dans TD2 on prend $T_0 = 240/23$ et pour le Transport delay

$$\begin{aligned}\text{Time delay} &= 36 \\ \text{Initial input} &= 20/69\end{aligned}$$

Dans chaque cas on réalise une simulation **sans STEP** et une **avec STEP**. Pour ce bloc on prendra les paramètres

$$\begin{aligned}\text{Step time} &= 10 \\ \text{Final value} &= 0.2\end{aligned}$$

On a donc le plan de simulation

Continu		Mixte	
sans STEP	avec Step	sans STEP	avec STEP

On réinitialise le Workspace $\Rightarrow$ fermer et réouvrir MATLAB ou prendre le *Workspace Browser*, sélectionner tous les éléments et faire *delete*. On va réaliser 4 simulations sur 200 périodes avec *ode45*. Pour stocker les divers résultats on utilise à nouveau un cube. La matrice M est de format (201×9) ,

le cube comprendra 4 "feuilles" correspondant aux 4 matrices M successives. On modifie l'ancien programme `Stockage` en passant la valeur finale de l'indice de boucle j à 9. On commence par dimensionner le cube avec l'ordre

$$res = \text{zeros}(201, 9, 4)$$

On réalise nos 4 simulation

- (#1) : continu sans STEP,
- (#2) : continu avec STEP,
- (#3) : mixte sans STEP,
- (#4) : mixte avec STEP.

Pour exploiter nos résultats il faut extraire les données du cube pour former des tableaux 2D (matrices).

Pour les niveaux on peut construire une matrice $Niv(201 \times 8)$ [$Niv = \text{zeros}(201, 8)$] telle que

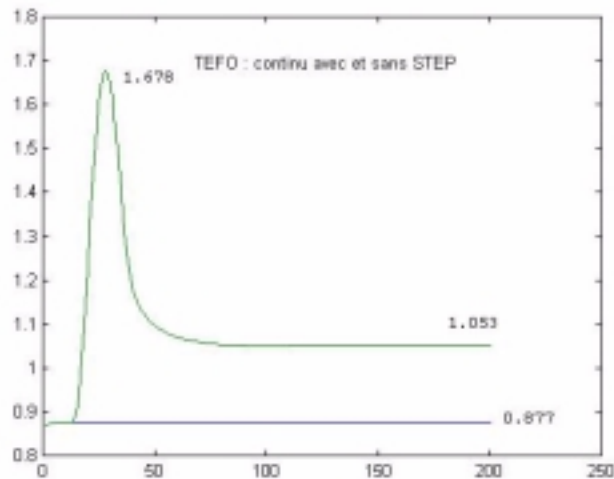
$$\begin{aligned} Niv(:, 1 : 4) &= res(:, 2, 1 : 4) \rightarrow TEFO \\ Niv(:, 5 : 8) &= res(:, 3, 1 : 4) \rightarrow T \end{aligned}$$

C'est à dire que la structure colonne de cette matrice est telle que

TEFO				T			
Continu		Mixte		Continu		Mixte	
∅	STEP	∅	STEP	∅	STEP	∅	STEP

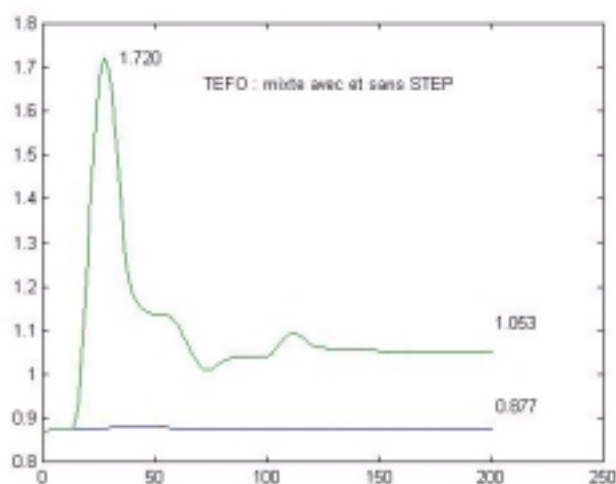
On peut alors faire plusieurs graphes comparatifs.

- Pour $TEFO$: continu sans et avec $STEP$ [» $\text{plot}(Niv(:, 1:2))$]



La perturbation du *STEP* se traduit par une hausse de la valeur d'état stationnaire (1.053 / 0.877) après un processus d'ajustement qui amène *TEFO* à la valeur maximale 1.678.

→ Pour *TEFO* : mixte sans et avec *STEP* [» `plot(Niv(:,3:4))`]



FE061

Du point de vue des valeurs stationnaires il n'y a guère de changement, seul le processus d'ajustement lors du jeu du *STEP* est légèrement modifié (valeur maximale un peu plus élevée 1.720 / 1.678). Les valeurs d'état stationnaire exactes sont

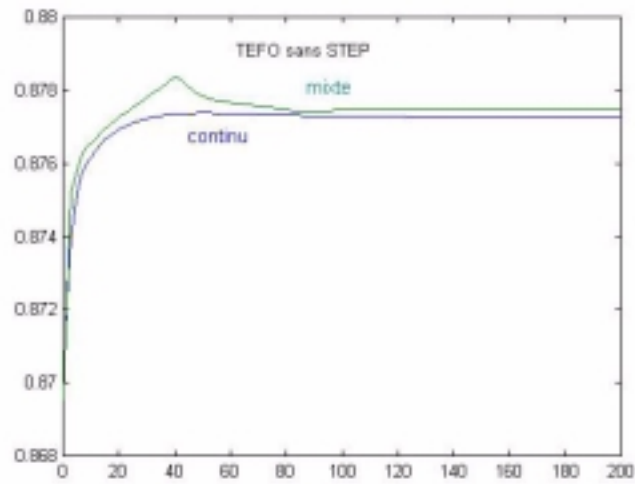
<i>TEFO</i>			
<i>Continu</i>		<i>Mixte</i>	
$\emptyset$	<i>STEP</i>	$\emptyset$	<i>STEP</i>
0.8772692	1.0527244	0.8774798	1.0527391

Les écarts sont on ne peut plus minimes.

→ Pour *TEFO* : sans *STEP* mixte et continu [» `plot(t, Niv(:,1), t, Niv(:,3))`] ou *t* est le vecteur temps tel que

$$\gg t = [0 : 200]'$$

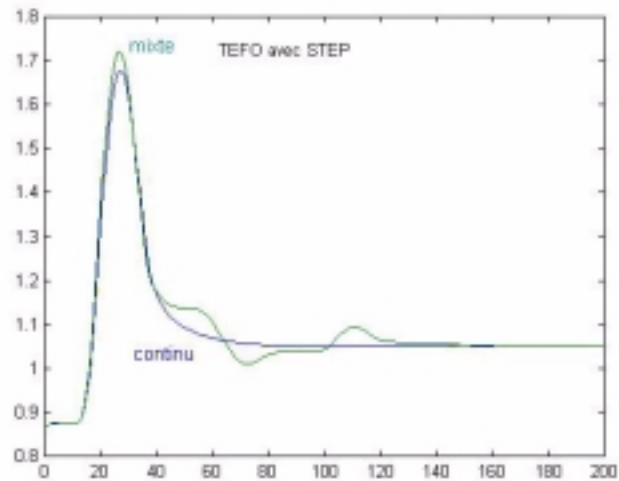
Il vient



FE062

L'introduction des *Transports Delays* se traduit par une légère accumulation des testeurs en formation avant "vidage". A noter la faiblesse des différences ($< 1/1000$).

→ Pour *TEFO* : avec *STEP* mixte et continu [$\gg \text{plot}(t, \text{Niv}(:,2), t, \text{Niv}(:,4))$]



FE063

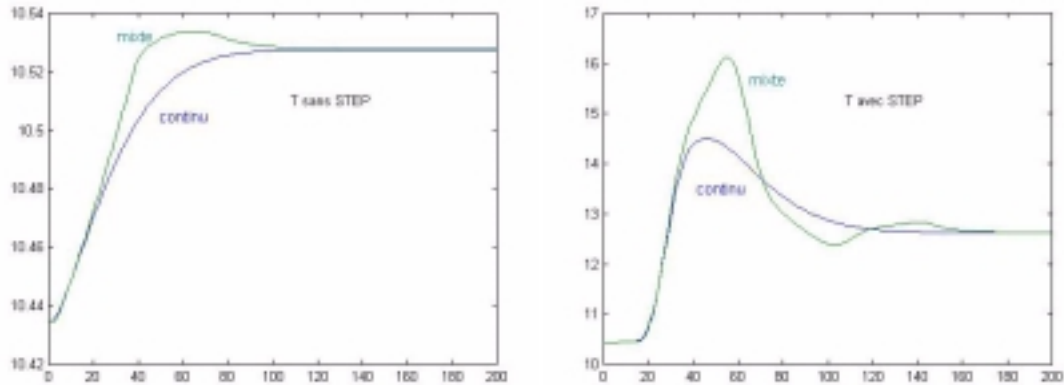
On note qu'il n'y a guère de différences pendant la phase première de croissance jusqu'à $t \# 25$, par contre au delà, l'introduction des délais purs engendre une évolution moins lissée.

On peut essayer de voir ce qu'il en est pour T , où les valeurs mises en jeu sont plus importantes.

→ Pour T : sans $STEP$ mixte et continu [$\gg \text{plot}(t, \text{Niv}(:,5), t, \text{Niv}(:,7))$]

→ Pour T : avec $STEP$ mixte et continu [$\gg \text{plot}(t, \text{Niv}(:,6), t, \text{Niv}(:,8))$]

soit respectivement



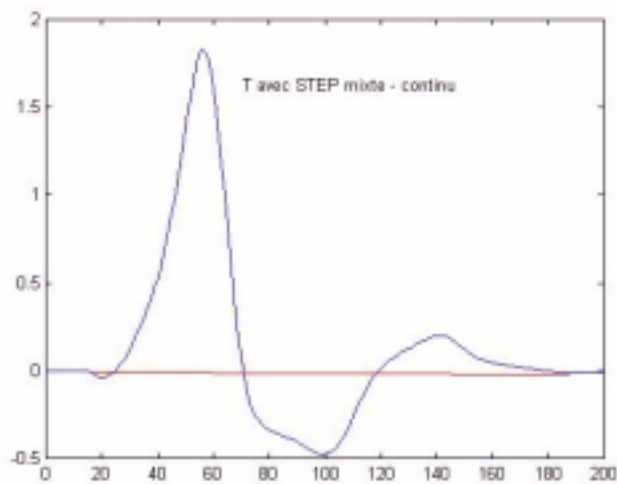
Pour le cas non perturbé (sans STEP) les différences sont minimales on a ainsi

$$\max(\text{Niv}(:,7) - \text{Niv}(:,5)) = 0.021\dots$$

Dans le cas où le STEP joue les différences sont un peu plus forte faisons

$$\gg \text{plot}(t, \text{Niv}(:,8) - \text{Niv}(:,6))$$

il vient



FE066

On obtient certes, à terme, des valeurs similaires, mais entre temps on a des différences (celles-ci peuvent être fortes dans le cas de certains modèles). On pourrait alors se demander quel est l'intérêt de l'introduction de délai purs ?

Comme l' a dit Keynes en substance, " à long terme nous serons tous morts ". De fait une période de 200 mois (le mois est l'unité de temps de notre modèle) représente environ 16 ans et demi. C'est très long dans un contexte économique. Un temps moyen d'observation peut être estimé à environ 4 à 5 ans (48 à 60 mois) sur cette période T varie. Quand bien même l'économiste analyste tend à faire des passages à la limite ($t \rightarrow \infty$) pour trouver de valeurs d'équilibre, il semblerait que ce soit surtout les processus de transition qui importent. Dans ce cas l'introduction de délais purs peut engendrer des différences notables.

On peut se livrer à une analyse similaire sur les flux. On dimensionne une matrice $Flux$ (201×12) dont la structure colonne est telle que l'on ait

$TE \leftrightarrow M(:, 4)$				$TFF \leftrightarrow M(:, 5)$				$TD \leftrightarrow M(:, 6)$			
Cont.		Mix.		Cont.		Mix.		Cont.		Mix.	
$\emptyset$	STEP	$\emptyset$	STEP	$\emptyset$	STEP	$\emptyset$	STEP	$\emptyset$	STEP	$\emptyset$	STEP

Ainsi on a

$$Flux(:, 1 : 4) = res(:, 4, 1 : 4) \rightarrow TE$$

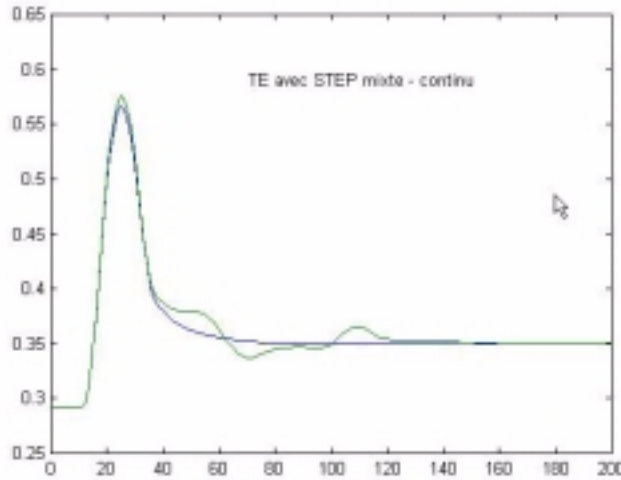
$$Flux(:, 5 : 8) = res(:, 5, 1 : 4) \rightarrow TFF$$

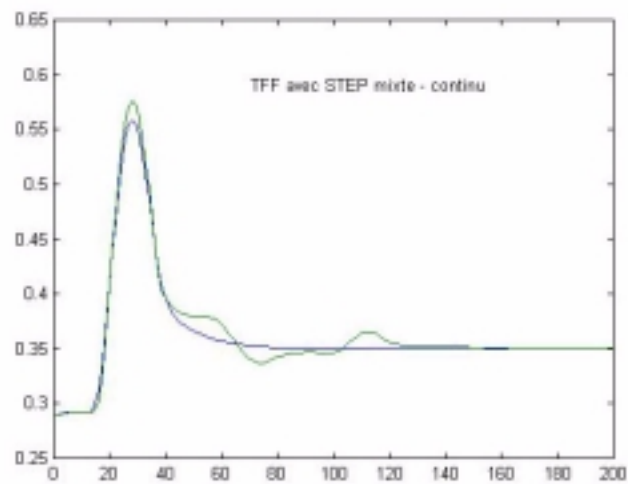
$$Flux(:, 9 : 12) = res(:, 6, 1 : 4) \rightarrow TD$$

On peut tracer les graphes comparatifs pour TE , TFF et TD lorsque le STEP joue dans les cas continu et mixte soit

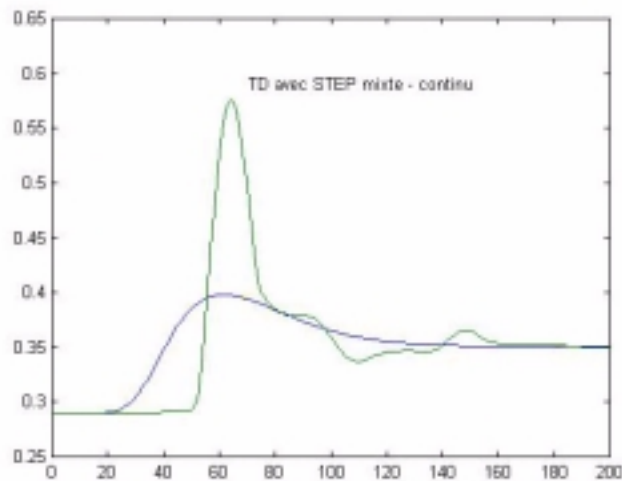
- » $plot(t, Flux(:, 2), t, Flux(:, 4))$
- » $plot(t, Flux(:, 6), t, Flux(:, 8))$
- » $plot(t, Flux(:, 10), t, Flux(:, 12))$

On obtient les 3 graphes ci-après





FE068

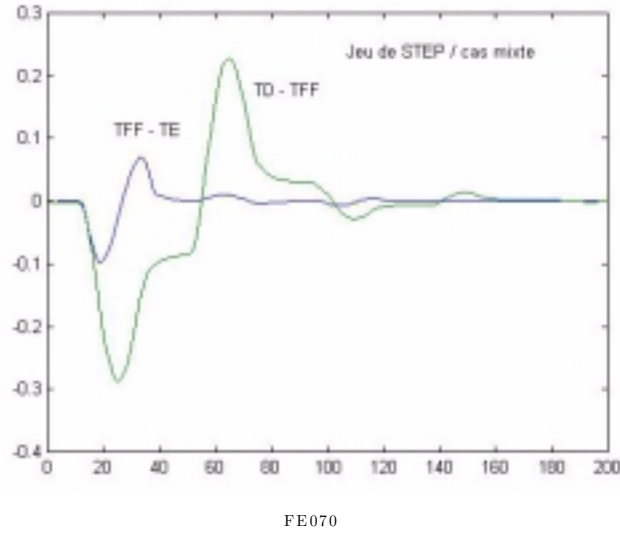


FE069

On constate que le comportement de TFF est un quasi décalque de celui de TE (au délai 3 près), le comportement de TD est plus diversifié entre le cas continu et le cas mixte.

Compte tenu du jeu des délais purs on note des décalages transitoires entre TFF et TE et entre TD et TFF dans le cas où $STEP$ joue.

» $plot(t, Flux(:,8)-Flux(:,4), t, Flux(:,12)-Flux(:,4))$



Nous avons ainsi réalisé un certain nombre de simulations. Il y a bien d'autres possibilités. Dans la pratique si l'on devait effectivement traiter le sujet on devrait réaliser un second modèle intégrant une "nouvelle politique", (on peut le faire à titre d'exercice) et voir ce qu'il en est.

Cela étant on constate que Simulink permet de simuler des modèles non linéaires relativement compliqués. On va voir qu'il serait possible d'effectuer un travail similaire, en travaillant directement sous Matlab, mais que l'on a n'a pas alors la même flexibilité.

3.3 Future Electronics et Matlab.

3.3.1 Le cadre général.

Nous avons vu que sous Matlab il est possible d'intégrer des équations différentielle en utilisant une *M-File*. Si l'on a un *M-File* "*toto*" contenant des équations différentielles l'intégration peut se faire en utilisant un des solveurs possibles et en écrivant

$$[t, y] = \text{solver}('toto', tspan, y0)$$

avec $tspan = [t0 \ tfinal]$, c'est un vecteur spécifiant l'intervalle d'intégration. Cet ordre permet d'intégrer le système d'équations différentielles

$$y' = F(t, y)$$

de l'instant $t0$ à l'instant $tfinal$ avec les conditions initiales $y0$. La fonction $F(t, y)$ doit retourner un vecteur colonne. Chaque ligne du vecteur solution y correspond à un instant retourné dans le vecteur colonne t .

Un programme de calcul usuel d'un modèle SSD peut se décomposer comme suit

Constantes et Initialisations
Pour t allant de t_0 à t_{final} faire
Calcul des auxiliaires
Calcul des flux
Calcul des niveaux
Stockage des résultats
Fin
Fin

Dans le cas présent on peut écrire une M-File correspond aux équations de Future Electronics. Mais on voit tout de suite qu'il existe un certain nombre de problèmes qui vont perturber le schéma général antérieur.

3.3.2 Problème du tri des auxiliaires.

Lorsque l'on édite notre M-File reprenant les équations initiales de Future Electronics il est nécessaire de calculer les auxiliaires dans un certain ordre sachant que l'on ne connaît initialement que

- les valeurs des constantes
- les valeurs initiales des variables d'état (les éléments de y).

On notera que sous Simulink, ce problème d'ordre de calcul des auxiliaires est transparent pour l'utilisateur.

On pourrait faire le tri des auxiliaires en se servant du diagramme de flux originel, en partant de niveaux que l'on connaît et en remontant les chaînes d'information d'auxiliaire en auxiliaire. La chose peut se révéler bien vite fastidieuse. Aussi on peut songer à automatiser le processus, en utilisant un programme qui revient in fine à trianguler un tableau. Le programme en question est une M-File dénommée "Triaux".

3.3.2.1 Le programme "Triaux".

Il se présente comme suit

```
%Programme Triaux
%VS1.1 du 19.06.2000
M=MO;
N=size(M,1);
for L=1:N
    NE(L)=0;
end
C=0; I=0; SP=1;
while SP ~= 0
```

```

for L=1:N
  if NE(L)==0
    P(L)=0;
    for K=1:N
      P(L)=P(L)+M(L,K);
    end
  end
end
for L=1:N
  if P(L)==0 & NE(L)~=1
    I=I+1;
    AUXTRI(I)=AUX(L);
    ORDCAL(I)=C;
    NE(L)=1;
  end
end
for K=1:N
  if P(K)==0
    for L=1:N
      M(L,K)=0;
    end
  end
end
C=C+1;
SP=0;
for L=1:N
  SP=SP+P(L);
end
end

```

3.3.2.1 Commentaires sur le programme "Triaux".

Tout repose sur la "triangulation" d'une matrice booléenne comportant en lignes et en colonnes les auxiliaires du modèle dans un certain ordre. Par exemple l'ordre de lecture des auxiliaires dans le modèle originel.

Si une auxiliaire donnée ne dépend que de niveaux et de constantes la ligne correspondante à cette auxiliaire ne comporte que des 0. Si elle dépend d'autres auxiliaires sa ligne comporte des 1 dans les colonnes correspondant à ces auxiliaires là.

La matrice booléenne d'origine est notée MO . La matrice courante est M telle que au départ

$$M = MO$$

Le paramètre N représente la taille de cette matrice carrée.

On note aussi AUX le vecteur originel des auxiliaires, constitué tout simplement par leur numéro d'ordre d'entrée.

On note NE le vecteur de "non existence" d'une ligne, i.e. à partir du moment où le poids d'une ligne (somme des éléments de la ligne de M) est nul on élimine cette ligne. Au départ tous les éléments de NE sont mis à 0, toutes les lignes existent. C'est la première boucle

```
for L=1:N
    NE(L)=0;
end
```

Puis on initialise un certain nombre de paramètres. Le paramètre C permet de déterminer l'ordre de calculabilité des auxiliaires triées. A l'origine $C = 0$.

I est un compteur de boucle servant à détecter le nombre de variables auxiliaires effectivement triées. A l'origine $I = 0$ et on doit avoir $I = N$ en fin de processus. Si $I > N$ il y a une erreur quelque part soit dans la constitution de la matrice booléenne soit dans la conception du modèle. Ainsi on peut détecter l'existence d'une boucle fermée de variables auxiliaires.

Le paramètre SP représente la somme des poids lignes. Il faut amener cette somme des poids à 0. A l'origine on pose $SP = 1$, on pourrait prendre toute autre valeur non nulle.

Tant que la somme des poids est différente de 0 on réitère les calculs, c'est la boucle principale

```
while SP ~= 0
    ...
end
```

Si une ligne existe, i.e si $NE(L) == 0$, on calcule son poids courant, on met $P(L)$ à 0 et on fait une sommation en colonne. C'est la première boucle de la boucle majeure.

Puis si une ligne a un poids nul ($PE(L) == 0$) **et** si elle existe encore ($NE(L) \sim 1$) il faut l'éliminer $\Rightarrow$ elle n'existe plus, elle n'est plus prise en compte.

On commence par incrémenter le compteur I d'une unité. On remplit par pas successifs le vecteur $AUXTRI()$, il est constitué par les numéros des lignes couramment éliminées.

```
AUXTRI(I) = AUX(L);
```

On remplit aussi par pas successifs le vecteur $ORDCAL()$ donnant l'ordre de calculabilité des auxiliaires triées.

```
ORDCAL(I) = C;
```

Une auxiliaire qui ne dépend que de variables d'état ou de constantes est d'ordre de calculabilité $C = 0$. Si elle dépend d'une autre auxiliaire on a $C = 1$, etc.

La ligne ayant été triée elle n'existe plus on met à 1 l'élément correspondant de NE

$$NE(L) = 1;$$

On passe ainsi en revue toutes les lignes $\rightarrow$ seconde boucle de la boucle majeure.

Lorsqu'une ligne est de poids nul, elle n'existe plus. Il en va de même pour la colonne correspondante. D'où l'annulation des colonnes de même numéro que les lignes de poids zéro.

```

for K=1:N
  if P(K) == 0
    for L=1:N
      M(L,K) = 0;
    end
  end
end
end

```

Les lignes courantes ayant été triées on incrémente le compteur C d'ordre de calcul.

$$C = C + 1;$$

On calcule alors la somme des poids courants, en remettant d'abord cette somme à 0

```

SP=0;
for L=1:N
  SP=SP+P(L);
end

```

Tant que cette somme est nulle on recommence le tri.

Dans le cas du modèle *FUTELEC* on va constituer *MO* en lisant les auxiliaires au fur et à mesure. A noter qu'une auxiliaire possédant une initialisation

$\rightarrow$ parce qu'elle dépend de variables d'états et de constantes (cas de *TEFI*)

$\rightarrow$ parce qu'elle fait appel à une fonction de lissage (cas de *QO*)
est censée ne dépendre de personne.

Pour constituer *MO* et *AUX* on se sert de Excel, c'est la solution optimale en l'espèce. On a dans le cas présent le tableau suivant dans lequel on ne fait apparaître, pour des raisons pratiques, que les 1 de la matrice booléenne.

	TEFI	TTPH	QR	QO	RQA	R	RO	FEM	TC	TP
	1	2	3	4	5	6	7	8	9	10
TEFI	1									
TTPH	2	1								1
QR	3		1							
QO	4									
RQA	5			1						
R	6				1					
RO	7					1				
FEM	8						1			
TC	9				1					
TP	10								1	

La matrice MO est constituée par la matrice interne formée de 0 et de 1. Pour la rentrée dans le *Workspace* de Matlab on commence par la copier sous Excel.

Dans la fenêtre de commande de Matlab on écrit

$$MO = [$$

on fait *Ctrl+C* (= copier) on termine par

]

C'est fait ! Pour AUX on prend le vecteur (ligne ou colonne) des numéros d'ordre.

On demande d'effectuer *Triaux* (taper *triaux + Enter*). Puis on demande *AUXTRI* et *ORDCAL* on obtient

» AUXTRI

AUXTRI =

1 4 5 6 9 7 10 2 8 3

» ORDCAL

ORDCAL =

0 0 1 2 2 3 3 4 4 5

»

En d'autres termes les auxiliaires triées avec leur ordre de calcul sont

Aux.	C
TEFI	0
QO	0
RQA	1
R	2
TC	2
RO	3
TP	3
TTPH	4
FEM	4
QR	5

On peut vérifier sous Excel, en manipulant les lignes puis les colonnes, dans l'ordre obtenu, que l'on obtient bien une matrice booléenne triangulaire inférieure.

Il n'apparaît pas utile de faire sous Matlab un traitement d'un vecteur de chaînes de caractères constitué par les noms des auxiliaires.. La mise en forme est prohibitive en temps. Il faudrait copier chacun des éléments chaîne de caractère à partir d'Excel (en suivant la même procédure que précédemment). Il faudrait en outre mettre chaque variable texte entre deux apostrophes (*single quotes*), par ex

'TEFI'

et faire en sorte que toutes ces variables soient de même longueur, ainsi il faudrait rajouter trois blancs (β) à la variable R

'R $\beta\beta\beta$ '

Ce n'est pas des plus rapide.

3.3.3 Problème des modulateurs (tabulations).

Sous Simulink on réalise les interpolations linéaires des modulateurs à l'aide du bloc *Look-Up Table*



On notera que ce bloc réalise une extrapolation en dehors des bornes de la table.

On a une fonction similaire sous Matlab qui est *interp1*. Sa syntaxe est

$$yi = \text{interp1}(x, y, xi)$$

cet ordre renvoie un vecteur yi qui contient des éléments correspondant aux éléments de xi et qui sont déterminés par interpolation entre les vecteurs $\mathbf{x}$ et $\mathbf{y}$. Le vecteur $\mathbf{x}$ spécifie les points auxquels on fait correspondre les données $\mathbf{y}$. Ce vecteur $\mathbf{x}$ représente le découpage discret de l'axe des abscisses.

Le problème avec cette fonction est que : "les valeurs hors champ sont retournées sous la forme *NaN = Not-a-Number*". D'où un blocage de la simulation si l'on n'y prend pas garde.

Ainsi pour décrire une table on utilise la fonction *interp1*. On différencie la variable d'entrée selon les cas, en utilisant une variable intermédiaire xt . En outre on automatise le processus en ayant recours à une *M-file* dénommée *Table* qui est la suivante

```
% Programme Table
tab(1,:)=(minx:incr:maxx);
if xent <= minx
    xt=minx;
elseif xent>minx & xent<maxx
    xt=xent;
else
    xt=maxx;
end
yout=interp1(tab(1,:),tab(2,:),xt);
```

3.3.4 La M-File pour Future Electronics.

En fait cette *M-File* nommée *Fut2* correspond à une fonction. le programme commenté est le suivant.

```
function dy=Fut(time,y)
% Vs 1.2 du 21.06.2000 = Fut2
% On met d'abord les constantes.
ddf=3; D1=ddf/3;
dms=36; D2=dms/3;
doq=6; dor=2;
```

```

ddp=3; D3=ddp/3;
qa=1;

% Les initialisations seront fixées à l'extérieur
% Cf. la M-file Calfut2

% Les variables d'état sont les éléments d'un vecteur y
% On doit définir un vecteur colonne dy sous la forme
% dy=zeros(n,1) où n est le nombre de variables d'état
dy=zeros(11,1);

% On a les correspondance suivantes des variables d'état.
L11=y(1); L21=y(2); L31=y(3);
L12=y(4); L22=y(5); L32=y(6);
TEFO = L11+L21+L31;
T=L12+L22+L32;
QO=y(7); RO=y(8);
N1=y(9); N2=y(10); N3=y(11);

% On calcule les auxiliaires dans l'ordre fourni par Triaux.
TEFI= T-0.5*TEFO;

% QO est une "fausse" auxiliaire décrite par une variable d'état y(7)
RQA=QO/qa;

% Pour décrire une table on utilise la fonction interp1
% Malheureusement si l'entrée dépasse les bornes on a une erreur
% Il faut différencier la variable d'entrée selon les cas
% en utilisant une variable intermédiaire xt
% En outre on automatise le processus en ayant recours à une M-file
Table

% On donne la variable d'entrée
xent=RQA;

% On décrit d'abord l'axe des abscisses avec 3 variables réservées
minx=0.5; maxx=1.25; incr=0.25;

% On décrit une table remise à 0 à chaque fois
tab=zeros(2,(maxx-minx)/incr +1);

```

```

% On donne les ordonnées
tab(2,:)= [4 2 1 0.5];
% On appelle le programme table
Table;
R=yout;

xent=RQA;
minx=0; maxx=1.5; incr=0.5;
tab=zeros(2,(maxx-minx)/incr +1);
tab(2,:)= [0 500 1000 1500];
Table;
TCT=yout;
% TCT est la "sortie" de la table TTC

if time >=5
    Step =0.2;
else
    Step=0;
end
TC=TCT*(1+Step);
% RO est une "fausse" auxiliaire décrite par une variable d'état y(8)
TP=N3; % Attention au DLINF3
TTPH=TP/TEFI;

xent=RO;
minx=0.5; maxx=4; incr=0.5;
tab=zeros(2,(maxx-minx)/incr +1);
tab(2,:)= [0.01 0.0258 0.06 0.1 0.135 0.17 0.185 0.2];
Table;
FEM=yout;

xent=TTPH;
minx=50; maxx=200; incr=25;
tab=zeros(2,(maxx-minx)/incr +1);
tab(2,:)= [1.1 1.1 1 0.85 0.7 0.6 0.55];
Table;

```

```

QR=yout;

%On termine par les flux et les variations de niveaux associées
TE=FEM*(TEFO+T);
R11=L11/D1; R21=L21/D1; TFF=L31/D1;
dy(1)=TE-R11;
dy(2)=R11-R21;
dy(3)=R21-TFF;
R12=L12/D2; R22=L22/D2; TD=L32/D2;
dy(4)=TFF-R12;
dy(5)=R12-R22;
dy(6)=R22-TD;
% Traduction du smooth pour QO = y(7)
dy(7)=(QR-QO)/doq;
% Traduction du smooth pour RO = y(8)
dy(8)=(R-RO)/dor;
% Traduction du DLINF3 donnant TP
dy(9)=(TC-N1)/D3;
dy(10)=(N1-N2)/D3;
dy(11)=(N2-N3)/D3;

% L'intégration se fait en dehors par ode45('Fut',tspan,y0)
% On n'obtiendra que les variables d'état en sortie du processus !

```

3.3.5 Le calcul proprement dit.

De façon à éviter de répéter des ordres à l'écran on crée un petit programme de calcul, correspondant à la donnée de l'ordre d'intégration, dénommé *Cal-fut2*. C'est le suivant

```

% Programme de calcul de Futelec
% CalFut2
% Vs 1.2 du 21.06.2000
% On pose
ddf=3; D1=ddf/3;

```

```

dms=36; D2=dms/3;
tei=20/69; % te à l'origine pour les delay3p
L11i=tei*D1; L21i=L11i; L31i=L21i;
L12i=tei*D2; L22i=L12i; L32i=L12i;
QOi=1;
ROi=1;
N1i=1000; N2i=N1i; N3i=N2i;
tfin=150;
tspan=[0 tfin];
y0=[L11i L21i L31i L12i L22i L32i QOi ROi N1i N2i N3i];
[time,y]=ode45('Fut2',tspan,y0);

```

On vérifiera que l'on obtient bien les mêmes valeurs des **variables d'état (niveaux)** avec ce programme et Simulink.

Le gros problème est que ici il n'est pas question d'avoir une information numérique sur les auxiliaires ou les flux. Ainsi il vaut mieux utiliser Simulink en couplage avec Matlab (pour l'exploitation des résultats) que ce genre de programme direct sous Matlab.