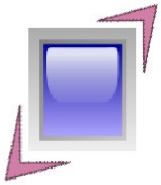
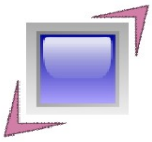



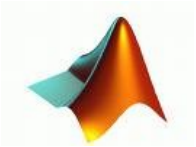
DEVSimPy
*SPE Python DEVS Framework*


# Étude Comparative

## DEVSimPy, Simulink©, PowerDEVS, Maple©

*Simulation du Stator*






*Laurent Capocchi*
*SPE UMR-CNRS 6134*
*Université de Corse*
*16 janvier 2009*

# Table des matières

## Objectif

Le but de ce document et de présenter les simulations effectuées sur le modèle du stator (Mada 5.5kw) avec Matlab/Simulink©, PowerDEVS, Maple© et DEVSimPy. Le modèle du stator est issue du modèle complet d'une machine asynchrone triphasée. Cette comparaison nous permettra de voir de quelle manière un modèle mathématique basé sur un système d'équations différentielles du premier ordre à coefficient non constant peut être résolue à la fois par la simulation à événement (DEVSimPy et PowerDEVS) et par la résolution numérique (Matlab/Simulink©) ou la résolution analytique (Maple©).

## Simulation avec Matlab/Simulink©

La version du logiciel est 7.0.1. Le modèle du stator est représenté sur la figure 1.

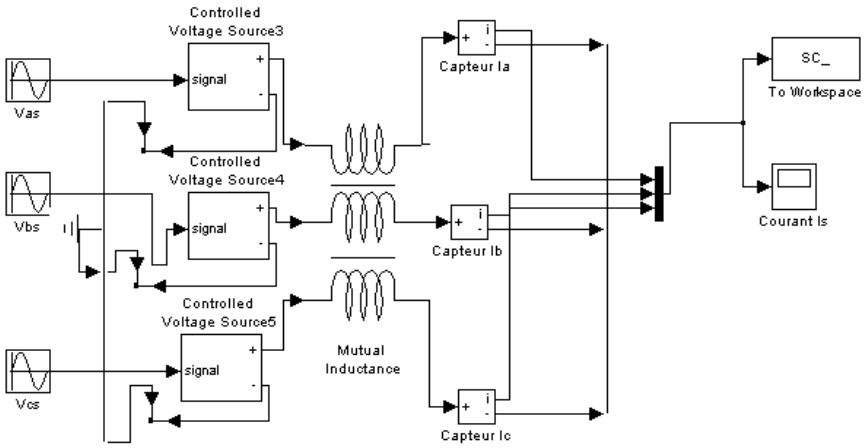
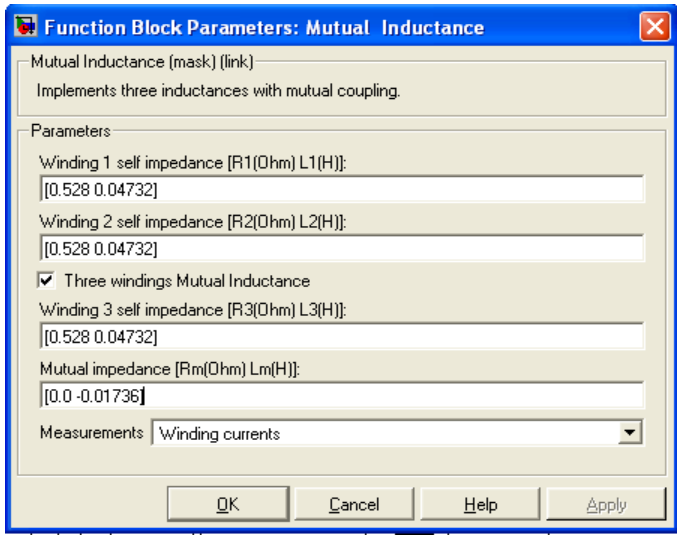
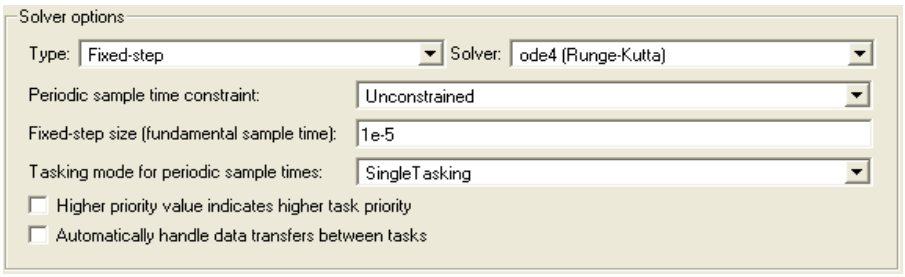



*Figure 1: Modèle Matlab/Simulink© du Stator*

Le modèle fait intervenir le bloc « Mutual Inductance » paramétré de la manière suivante:




Nous avons choisi une intégration basée sur la méthode Runge-Kutta d'ordre 4 à pas fixe (0.00001). Les paramètres de simulation sont montrés sur la capture d'écran suivante:




Les résultats de la simulation du modèle précédent pendant une seconde sont les suivants:

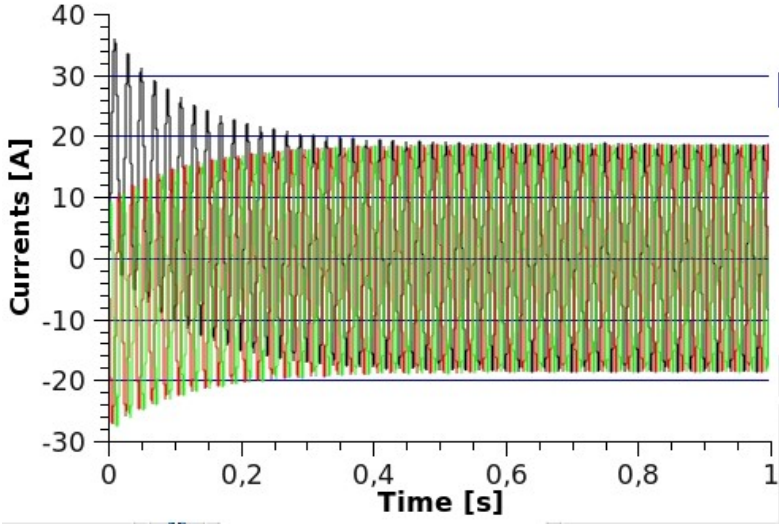



*Figure 2: Simulation Matlab/Simulink©*

# Simulation avec Maple©

Le logiciel Maple© est utilisé dans sa version 11.0. Le modèle est bien entendu littéral et nous calculons les solutions analytiques.

$$i_{as}(t)=\frac{-1}{(1089+122500\pi^2)}\cdot$$
$$\frac{-(2373437500\,e^{(\frac{-66}{7}t)}(33\sqrt{(3)}+350\,\pi)\pi^2)}{(1089+122500\pi^2)}+\frac{(78323437500\,e^{(\frac{-66}{7}t)}\sqrt{(3)}\pi^2)}{(1089+122500\pi^2)}+$$
$$6781250\cos(100\pi t)\pi-\frac{(21099375\,e^{(\frac{-66}{7}t)}(33\sqrt{(3)}+350\,\pi))}{(1089+122500\pi^2)}+$$
$$\frac{(696279375\,e^{(\frac{-66}{7}t)}\sqrt{(3)})}{(1089+122500\pi^2)}-639375\sin(100\pi t)$$

$$i_{bs}(t)=(\frac{1}{(2(1089+122500\pi^2))})\cdot$$
$$156646875000\,e(-(\frac{66}{7})t)\sqrt{(3)}\frac{\pi^2}{(1089+122500\pi^2)}-$$
$$2373437500\,e(-(\frac{66}{7})t)(33\sqrt{(3)}+350\pi)\frac{\pi^2}{(1089+122500\pi^2)}-$$
$$6781250\sqrt{(3)}\sin(100\pi t)\pi+6781250\cos(100\pi t)\pi-21099375\,e(-(\frac{66}{7})t)\frac{(33\sqrt{(3)}+350\pi)}{(1089+122500\pi^2)}+$$
$$1392558750\,e(-(\frac{66}{7})t)\frac{\sqrt{(3)}}{(1089+122500\pi^2)}-639375\sin(100\pi t)-639375\cos(100\pi t)\sqrt{(3)}$$

$$i_{cs}(t)=-(\frac{19375}{2})e(-(\frac{66}{7})t)\frac{(33\sqrt{(3)}+350\pi)}{(1089+122500\pi^2)}+$$
$$\frac{((6781250\sqrt{(3)}\pi-639375)\sin(100\pi t)+6781250\cos(100\pi t)(\pi+(\frac{33}{350})\sqrt{(3)}))}{(2178+245000\pi^2)}$$

Les solutions analytiques sont une somme de multiplication entre des sinus/cosinus est des exponentielles. La figure 3 montre le tracé des solutions sur 100000 points.

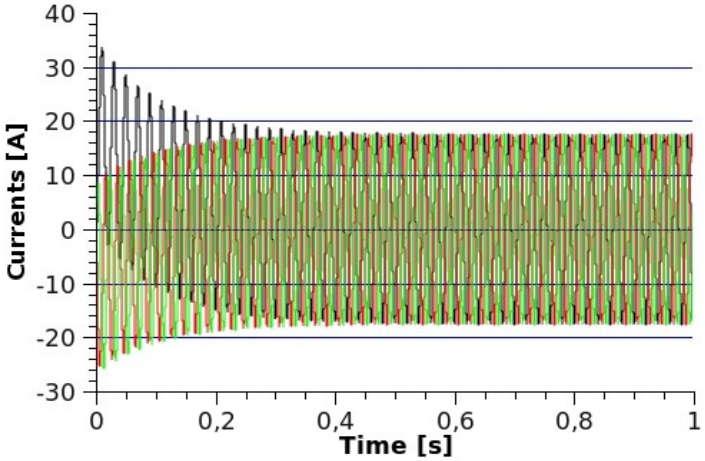



*Figure 3: Simulation analytique Maple©*

# Simulation avec PowerDEVS

La simulation avec PowerDEVS [1] nécessite une approche à événement discret. Le modèle est présenté sur la figure 4 et on remarquera qu'il utilise une représentation en schéma block proche de celle utilisé avec Matlab/Simulink©.

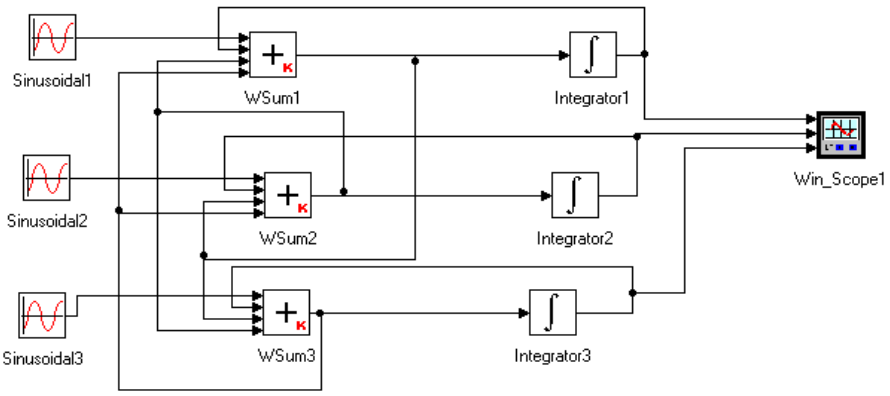



*Figure 4: Modèle PowerDEVS du stator*

De plus, la figure 4 met en évidence la présence de 3 boucles algébriques. Ces bouclages conduisent à une simulation infinie. A l'inverse de Matlab/Simulink©, PowerDEVS ne fournit aucune solution pour ce problème. Dans la littérature [2], les auteurs préconisent d'utiliser des modèles atomiques dont le comportement est de « casser » ces boucles. L'application de ces modèles appelés LB (Loop Breaking) conduit au modèle de la figure 5.

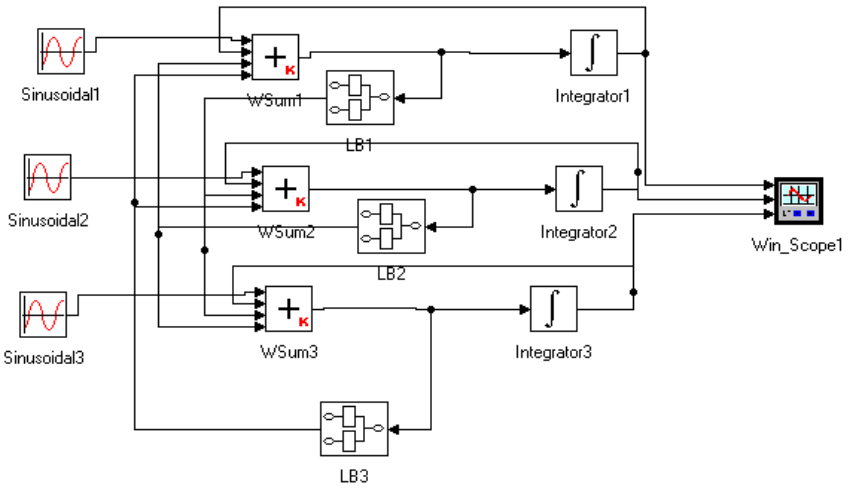
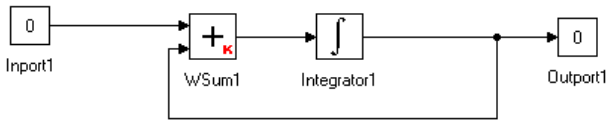



*Figure 5: Ajout des LB dans le modèle PowerDEVS*

L'utilisation des modèles LB nécessite de définir un facteur de tolérance. Le choix de ce paramètre est une contrainte de modélisation car il n'existe aucune règle de détermination de la meilleur valeur pour cette tolérance. Bien entendu plus elle sera petite plus le résultat sera meilleur au détriment d'un temps de simulation important.

Une autre approche consiste à remplacer des LB par des filtres passe-bas du premier ordre (LPF ) dont la constante de temps est facilement calculable (car elle dépend des caractéristique du système). La figure ci-dessous montre le schéma-bloc d'un tel filtre:




*Filtre passe-bas du première ordre*

Les résultats de la simulation du modèle du stator comportant des LPF sont données sur la figure 6:

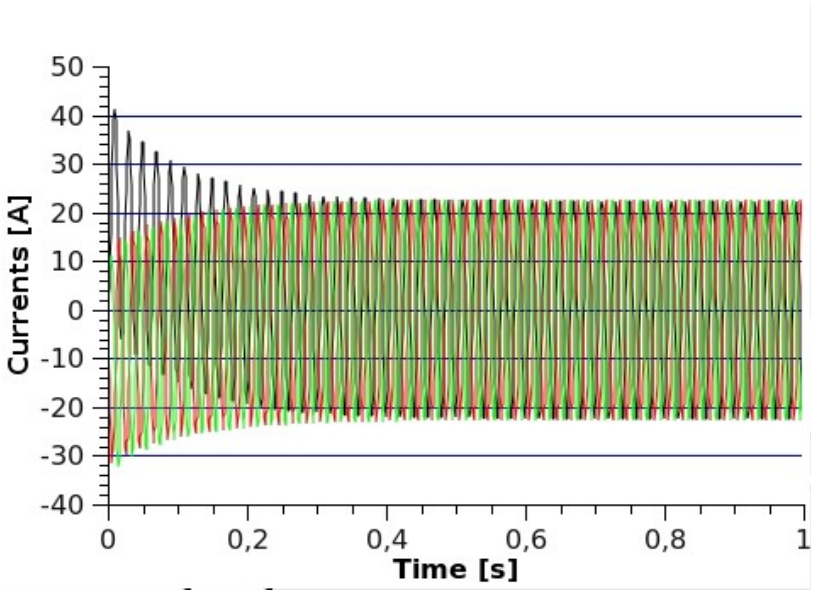



*Figure 6: Simulation PowerDEVS avec LPF*

Les résultats montrent une amplitude des signaux plus importante que pour Maple© et Matlab/Simulink©.

## Simulation avec DEVSimPy

DEVSimPy [3] implémente les mêmes modèles que PowerDEVS mais expérimente des approches de modélisation différentes. Dans un premier temps nous allons simuler le modèle du stator avec l'introduction des LPF afin de valider nos modèles par rapport à PowerDEVS. Les résultats de la simulation sont données sur la figure 7.

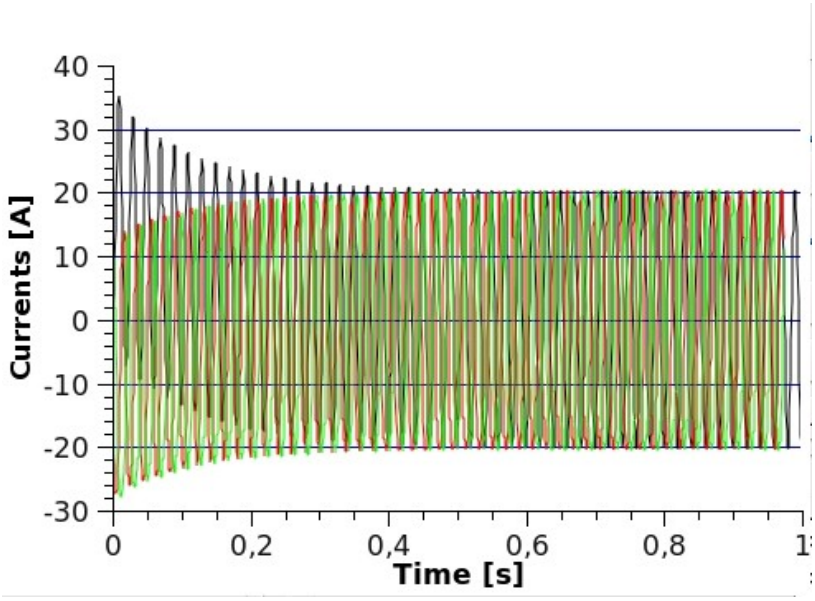



*Figure 7: Simulation avec DEVSimPy et les LPF*

Au vue des figures 6 et 7, on peut dire qu'il existe une différence d'amplitude. Cela peut être due à la résolution numérique différente dans les deux programmes (précision de la méthode de résolution utilisé par l'intégrateur d'ordre 2 et 3). Cela dit, la simulation avec DEVSimPy est plus proche de la solution analytique obtenue avec Maple.

Nous allons à présent développer une autre solution pour surpasser le problème des boucles algébrique. Considérons que dans la plupart des cas, le modèle « sommateur » est présent de ce type de boucle. Si le « sommateur » reçoit, à deux temps successif différent, les mêmes entrées, il n'a aucun intérêt de régénérer une sortie. En effet, c'est le fait que celui-ci reste indifférent à ce type de remarque qui fait que la boucle conduit à une simulation infinie. Par conséquent, si le modèle « sommateur » détecte qu'il se trouve dans une boucle algébrique, celui-ci ne renverra pas le même signal de suite et donc détruira la boucle. Il faudra ensuite interpoler le signal de sortie car le nombre de points sera inévitablement plus petit que dans le cas ou l'on utilise des LPF.

Nous avons implémenté ce type de comportement dans nos modèles « sommateur » et nous avons supprimé les LPF. Les résultats de la simulation sont sur la figure 8.

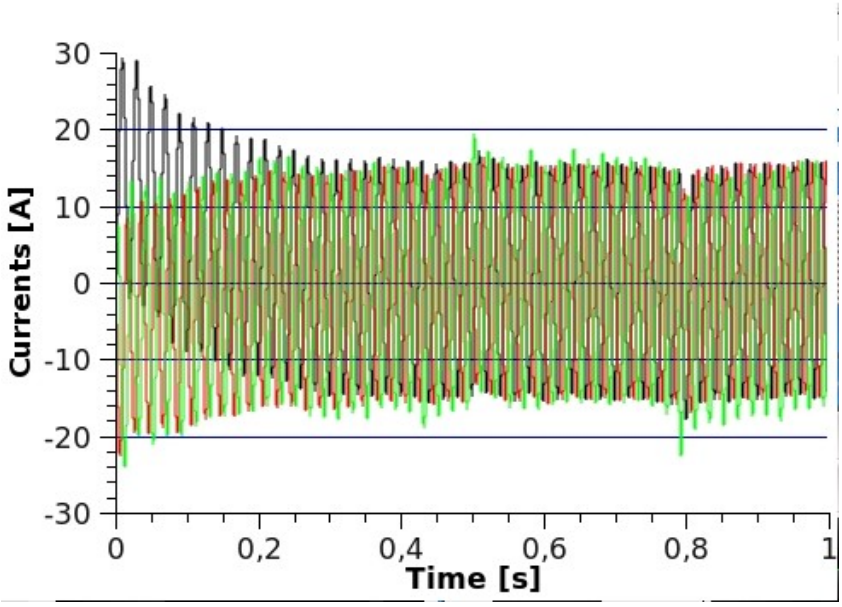



*Figure 8: Simulation DEVSimPy avec modif. des sommateurs*

Les signaux présentés sur la figure 8 ont été interpolé avec une méthode cubic sur 100000 points. Cette opération est indispensable car les modèles atomiques « sommateur » conduisent à des bases de temps différentes pour les trois signaux.

Nous présentons, sur la figure 9 ,les résultats de simulation avec l'introduction des LB à la place des LPF. La tolérance à été fixée à 0.00001.

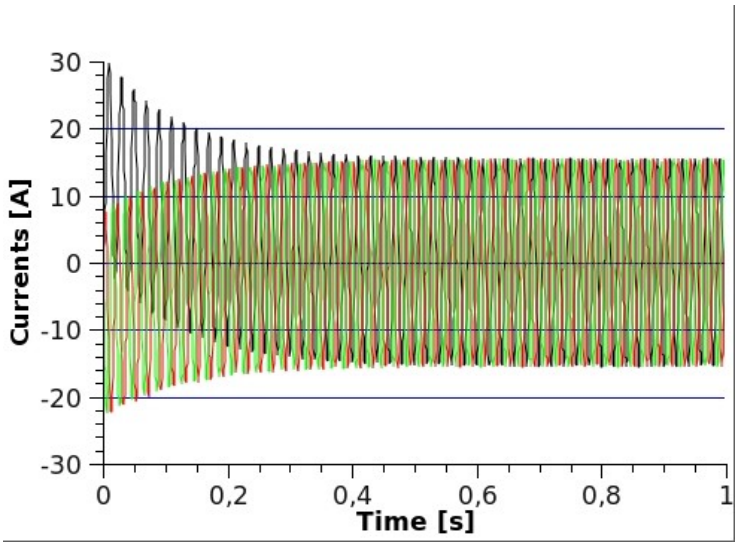



*Figure 9: Simulation DEVSimPy avec LB*

Les signaux sont plus réguliers que dans le cas des LPF. Les amplitudes sont diminuées par rapport aux résultats obtenus avec Matlab/Simulink©.

# Étude comparative

## *Calcul de l'erreur*

Après avoir interpolé les signaux pour obtenir la même base temporelle, nous pouvons calculer l'erreur entre deux courbes C1 et C2 résultant de deux simulations différents (méthode et/ou logiciel) par la formule:

$$E_i = \frac{\left|(C1_i - C2_i)\right|}{C2_i}$$

Pour chaque valeur i on peut reporter ensuite la courbe des erreurs. Nous allons calculer les erreurs entre les différentes méthodes expérimentées par rapport aux résultats analytique obtenu avec Maple©. Nous ne prenons en compte que le régime permanent (à partir de 0.4s).

| | ***Simulink*** | ***PowerDEVS avec LPF*** | ***DEVSimPy avec LPF*** | ***DEVSimPy avec LB*** | ***DEVSimPy avec sommateur mod.*** |
|---|---|---|---|---|---|
| Erreur moyenne / Sol. analytique | 0.00986459 | 0.914637 | 0.922846 | 0.228135 | 1.12371 |

Le tableau ci-dessus nous montre que la solution la plus proche de celle obtenue avec Maple© est celle obtenue avec DEVSimPy en appliquant les modèles LB. La solution la plus mauvaise et celle utilisant les « sommateurs » modifiés.

## *Calcul des temps de simulation*

Le tableau ci-dessous rassemble les temps de simulation pour les méthodes mises en œuvre dans cet étude.

| | ***Maple*** | ***Simulink*** | ***PowerDEVS*** | ***DEVSimPy*** | | |
|---|---|---|---|---|---|---|
| | | | | **Avec LPF** | Avec LB | Avec Sommateur |
| *Temps* | ~2s | ~2s | ~5s | **real-1m6.606s**<br>**user-1m3.052s**<br>**sys -0m0.404s** | real-15m14.788s<br>user-14m43.475s<br>sys- 0m2.116s | real -26m39.721s<br>user-25m17.643s<br>sys-0m6.112s |

Au vue de ces temps de simulation, la solution la plus rapide est celle utilisant les LPF.

## Conclusion

Si l'on fait un compromis entre le temps de simulation et l'erreur numérique par rapport au résultat obtenu par Maple©, la meilleur méthode est celle qui exploite les filtres passe-bas du premier ordre pour la simulation du stator.

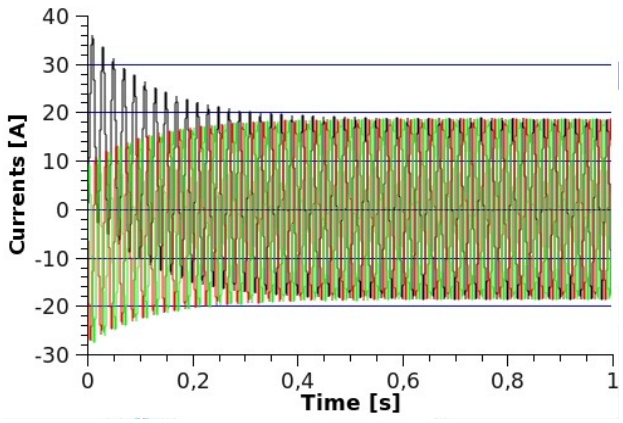



*Figure 2: Simulation Matlab/Simulink©*

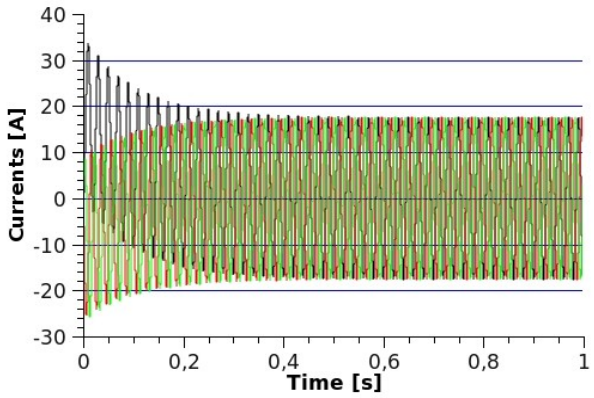



*Figure 3: Simulation Maple©*

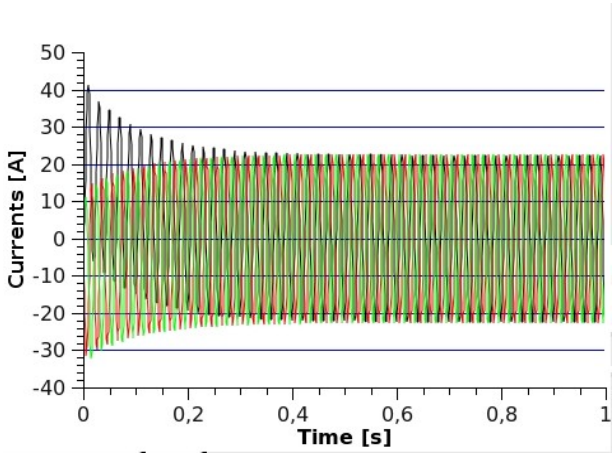



*Figure 6: Simulation PowerDEVS avec LPF*

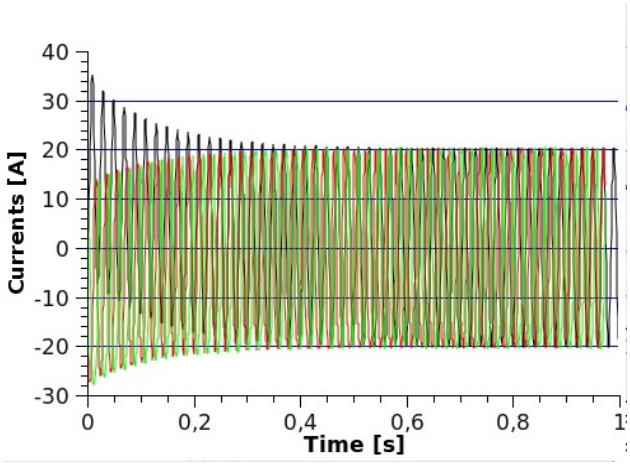



*Figure 7: Simulation avec DEVSimPy et les LPF*

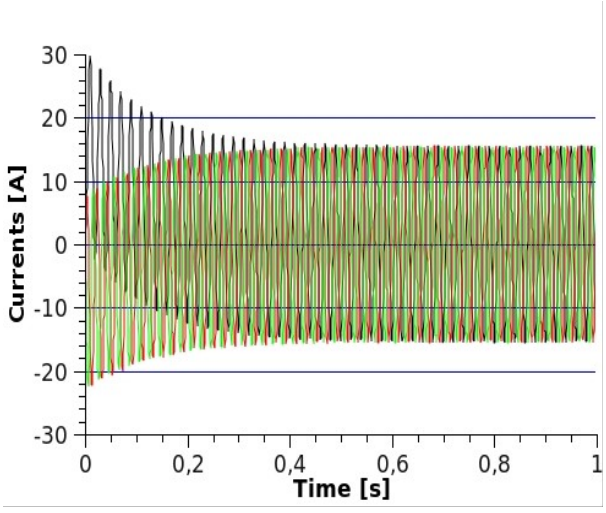



*Figure 9: Simulation DEVSimPy avec LB*

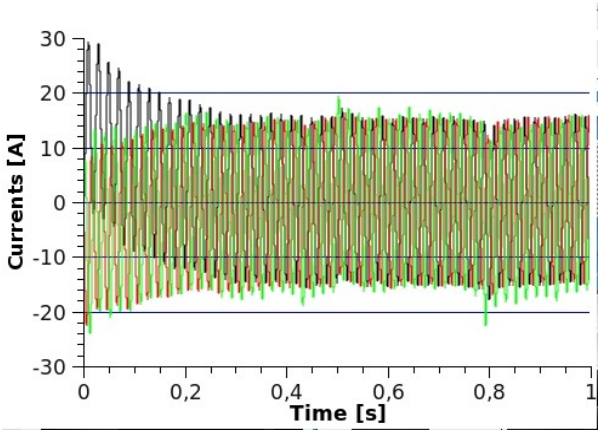



*Figure 8: Simulation DEVSimPy avec modif. des sommateurs*

## Références

[1] PowerDEVS, http://sourceforge.net/projects/powerdevs/

[2] Continuous System Simulation, F. Cellier, E. Kofman, Springer, 643 pages, 2006.

[3] DEVSimPy, https://code.google.com/p/devsimpy/