Contrôleur de moteur basé sur des réseaux MLP et RBF

Philippe Saint-Jacques

École Polytechnique de Montréal

22 avril 2010

Contrôle linéaire (par modèle d'état ou PID)

1. Équation caractéristique correspondant à la dynamique désirée

Contrôle linéaire (par modèle d'état ou PID)

1. Équation caractéristique correspondant à la dynamique désirée
2. Placement de pôles dominants (ζ, d)

Contrôle linéaire (par modèle d'état ou PID)

1. Équation caractéristique correspondant à la dynamique désirée
2. Placement de pôles dominants (ζ, d)
3. Transformée en z de la fonction de transfert du contrôleur

Contrôle linéaire (par modèle d'état ou PID)

1. Équation caractéristique correspondant à la dynamique désirée
2. Placement de pôles dominants (ζ, d)
3. Transformée en z de la fonction de transfert du contrôleur
4. Algorithme de contrôle
   $u_{k+1} = f(u_k, u_{k-1}, y_k, y_{k-1} \ldots u_{k-d_0}, y_{k-d_0})$

Hypothèses pour concevoir un contrôleur linéaire

1. Réponse du modèle du système en BF connue

Hypothèses pour concevoir un contrôleur linéaire

1. Réponse du modèle du système en BF connue
2. Optimisation possible de la boucle de contrôle

Hypothèses pour concevoir un contrôleur linéaire

1. Réponse du modèle du système en BF connue
2. Optimisation possible de la boucle de contrôle
3. **Fonction de coût quadratique (pour contrôle optimal)**

Hypothèses pour concevoir un contrôleur linéaire

1. Réponse du modèle du système en BF connue
2. Optimisation possible de la boucle de contrôle
3. **Fonction de coût quadratique (pour contrôle optimal)**
4. **Action de contrôle u ne peut être borné**

Moteur à aimants permanent

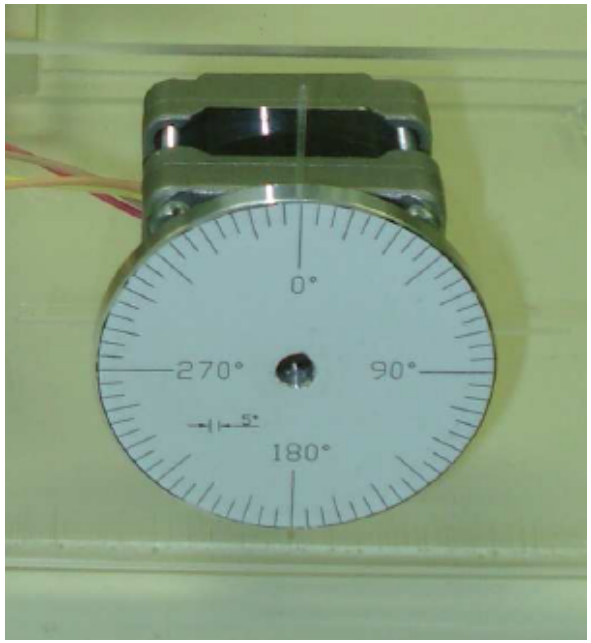


Figure: Moteur utilisé pour les simulations

Les constantes de ce moteur on été identifiées à $\tau_m = 0,031s$ et $K_m = 100$.

Modèle théorique du système (1 de 3)

La fonction de transfert du moteur peut être écrite comme suit en considérant la position comme la sortie du système et en négligeant la constante de temps électrique du moteur :

$$G(s) = \frac{K_m}{s(\tau_m s + 1)} = \frac{100}{s(0.031422s + 1)}$$

En posant le vecteur d'état suivant,

$$\vec{x} = \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix}$$

On obtient le modèle d'état correspondant au moteur :

$$\dot{\vec{x}} = A\vec{x} + B\vec{u} = \begin{bmatrix} 0 & 1 \\ 0 & -\frac{1}{\tau_m} \end{bmatrix} \vec{x} + \begin{bmatrix} 0 & 0 \\ \frac{K_m}{\tau_m} & -\frac{K_d}{\tau_m} \end{bmatrix} \vec{u}$$

Modèle théorique du système (2 de 3)

Le vecteur de commande du système vaut dans ce cas :

$$\vec{u} = \begin{bmatrix} e_a \\ T_d \end{bmatrix}$$

Où $e_a(t)$ est la tension aux bornes du moteur et T_d est le couple résistant aux bornes du moteur. T_d est considéré comme une perturbation. La sortie du système vaut :

$$\vec{y} = C\vec{x} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \vec{x}$$

Modèle théorique du système (3 de 3)

Le système doit être discrétisé pour pouvoir être simulé. Par la transformée de Laplace et la transformée en Z, et en posant $T_s = 0.010s$, on obtient :

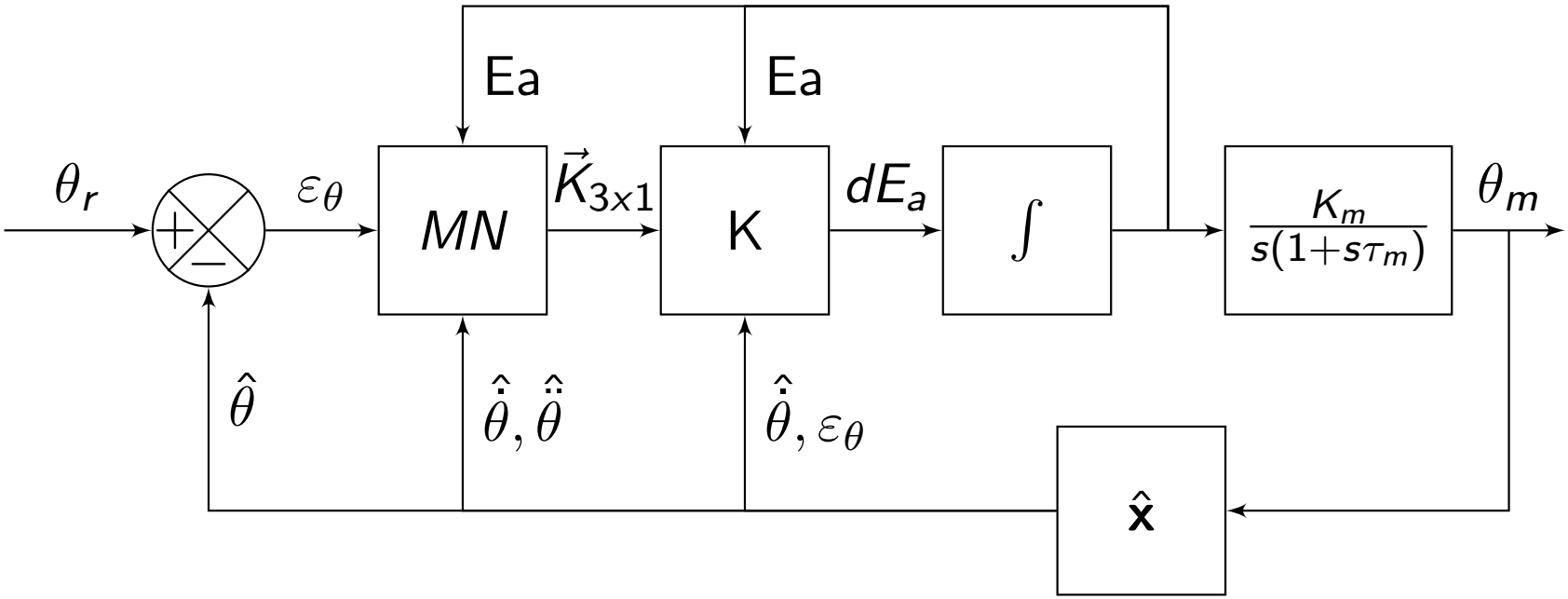
$$\vec{x}_{k+1} = A_d\vec{x}_k + B_d\vec{u}_k = \begin{bmatrix} 1.0000 & 0.0085 \\ 0 & 0.7243 \end{bmatrix}\vec{x}_k + \begin{bmatrix} 0.1453 & 0 \\ 27.5722 & 0 \end{bmatrix}\vec{u}_k$$

On a donc ici un système d'ordre d'observabilité 1.

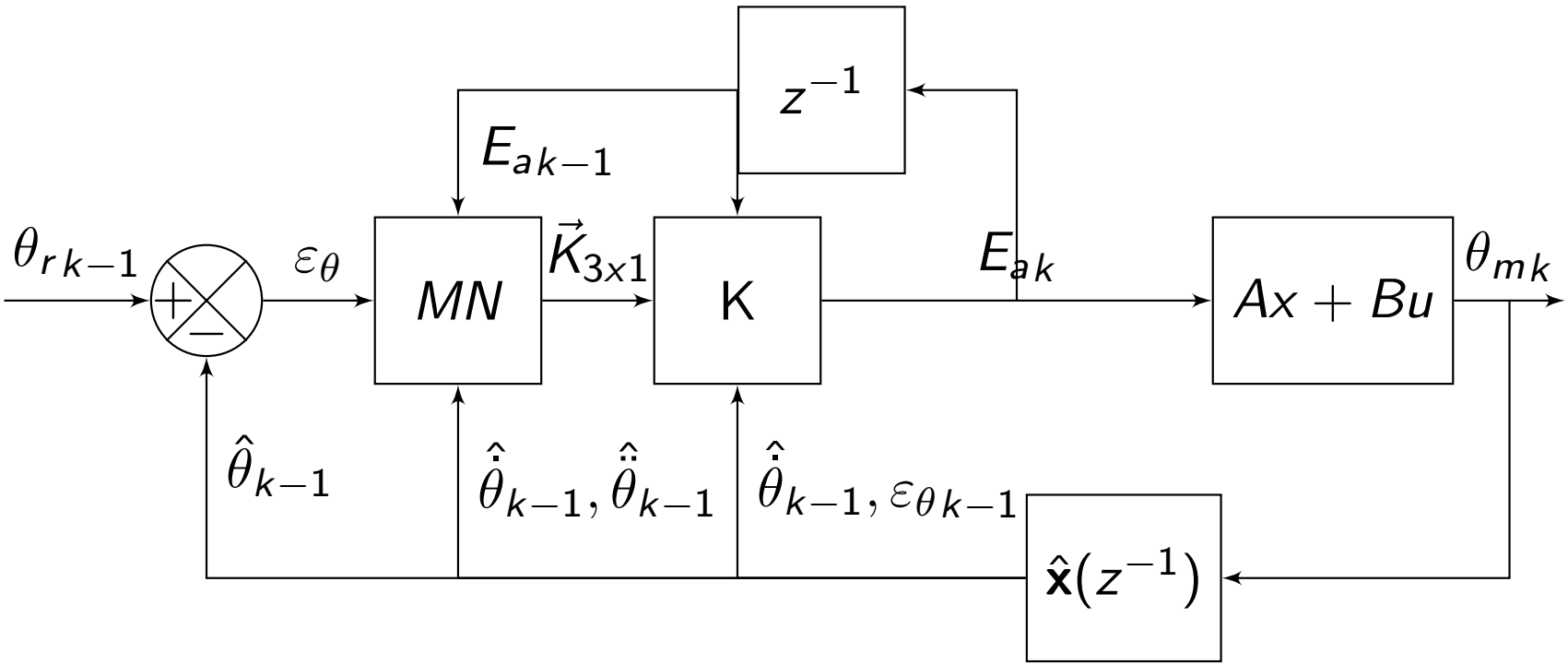
$$\vec{y}_k = C\vec{x}_k = \begin{bmatrix} 1 \\ 0 \end{bmatrix}\vec{x}_k$$

L'élimination de l'erreur en régime permanent d'un système avec correcteur peut être obtenue soit en intégrant l'erreur ou en intégrant la sortie de celui-ci. La deuxième approche est retenue.

Structure de la boucle de contrôle



Structure de la boucle de contrôle (modèle discret)



Fonction de coût à optimiser

La fonction de coût à minimiser doit décrire la dynamique et les contraintes du système asservi.

- $u_i = \mathcal{P}(\vec{\omega}, \varepsilon_\theta, \dot{\theta}, Ea)$

Fonction de coût à optimiser

La fonction de coût à minimiser doit décrire la dynamique et les contraintes du système asservi.

- $u_i = \mathcal{P}(\vec{\omega}, \varepsilon_\theta, \dot{\theta}, Ea)$
- $x_{i+1} = Ax_i + Bu_i$

Fonction de coût à optimiser

La fonction de coût à minimiser doit décrire la dynamique et les contraintes du système asservi.

- $u_i = \mathcal{P}(\vec{\omega}, \varepsilon_\theta, \dot{\theta}, Ea)$
- $x_{i+1} = Ax_i + Bu_i$
- $y_i = Cx_i$

Fonction de coût à optimiser

La fonction de coût à minimiser doit décrire la dynamique et les contraintes du système asservi.

- $u_i = \mathcal{P}(\vec{\omega}, \varepsilon_\theta, \dot{\theta}, Ea)$
- $x_{i+1} = Ax_i + Bu_i$
- $y_i = Cx_i$
- $C(\vec{\omega}) = \{\sum_{i=1}^{N_{iter}} \alpha(y_i - r_i)^2 + \gamma \dot{y}_i^2 + \beta \ddot{y}_i^2 + \mathcal{C}\}\Big|_{\vec{\omega}}$
- Le terme $\mathcal{C}$ contient des termes non-linéaires pouvant servir à limiter par exemple le dépassement en pénalisant lourdement un changement de signe de ε_θ.

Optimisation globale

L'ensemble des paramètres du réseau de neurones (poids et biais pour un MLP) peuvent être regroupés dans un vecteur de paramètres $\vec{\omega}$.

- La dimension de $\vec{\omega}$ croit très rapidement (43 pour un MLP (4,5,3)).

Optimisation globale

L'ensemble des paramètres du réseau de neurones (poids et biais pour un MLP) peuvent être regroupés dans un vecteur de paramètres $\vec{\omega}$.

- La dimension de $\vec{\omega}$ croit très rapidement (43 pour un MLP (4,5,3)).
- Algorithmes de rétropropagation peu ou pas utilisables (ε_θ ne dépend pas directement de u).

Optimisation globale

L'ensemble des paramètres du réseau de neurones (poids et biais pour un MLP) peuvent être regroupés dans un vecteur de paramètres $\vec{\omega}$.

- La dimension de $\vec{\omega}$ croit très rapidement (43 pour un MLP (4,5,3)).
- Algorithmes de rétropropagation peu ou pas utilisables (ε_θ ne dépend pas directement de u).
- Allure de la surface inconnue.

Optimisation globale

L'ensemble des paramètres du réseau de neurones (poids et biais pour un MLP) peuvent être regroupés dans un vecteur de paramètres $\vec{\omega}$.

- La dimension de $\vec{\omega}$ croit très rapidement (43 pour un MLP (4,5,3)).
- Algorithmes de rétropropagation peu ou pas utilisables (ε_θ ne dépend pas directement de u).
- Allure de la surface inconnue.
- Une approche numérique a été utilisée.

Optimisation globale

- Un ensemble de $2D$ points de départs sont créés de façon aléatoire et homogène dans l'espace $\mathcal{W}$.

$$\vec{\omega}_n = \{\pm 1, \hat{\epsilon}_2, ..., \hat{\epsilon}_D\}$$

Optimisation globale

- Un ensemble de $2D$ points de départs sont créés de façon aléatoire et homogène dans l'espace $\mathcal{W}$.

$$\vec{\omega}_n = \{\pm 1, \hat{\epsilon}_2, ..., \hat{\epsilon}_D\}$$

- Ces points servent de points de départ pour $2D$ optimisations locales.

Optimisation locale - méthode du gradient conjugué

La méthode du gradient de second ordre est utilisée car elle converge plus vite que le gradient du premier ordre.

- Faire une recherche linéaire dans la direction $\mathbf{h}_{k-1}$ pour obtenir le point ω_k donnant un coût minimal.

On boucle tant que $\Delta\omega \geq \epsilon$.

Optimisation locale - méthode du gradient conjugué

La méthode du gradient de second ordre est utilisée car elle converge plus vite que le gradient du premier ordre.

- Faire une recherche linéaire dans la direction $\mathbf{h}_{k-1}$ pour obtenir le point ω_k donnant un coût minimal.
- Calculer le gradient $\mathbf{g}_k$ au point ω_k :

$$\nabla_\omega C = \sum_i^D \frac{C(\omega + \mathbf{e}_i\delta) - C(\omega)}{\delta}$$

On boucle tant que $\Delta\omega \geq \epsilon$.

Optimisation locale - méthode du gradient conjugué

La méthode du gradient de second ordre est utilisée car elle converge plus vite que le gradient du premier ordre.

- Faire une recherche linéaire dans la direction $\mathbf{h}_{k-1}$ pour obtenir le point ω_k donnant un coût minimal.
- Calculer le gradient $\mathbf{g}_k$ au point ω_k :

$$\nabla_\omega C = \sum_i^D \frac{C(\omega + \mathbf{e}_i\delta) - C(\omega)}{\delta}$$

- Calculer la prochaine direction de recherche $\mathbf{h}_k$ (Polak-Ribiere) :

$$\mathbf{h}_k = \mathbf{g}_k + \frac{\mathbf{g}_k^T \mathbf{g}_k}{\mathbf{g}_{k-1}^T \mathbf{g}_{k-1}} \mathbf{h}_{k-1}$$

On boucle tant que $\Delta\omega \geq \epsilon$.

Conditions initiales et commandes

Deux cas de figures ont été utilisés pour l'entraînement des réseaux :

- Front montant : D = [0 0], W = Pi/6

Conditions initiales et commandes

Deux cas de figures ont été utilisés pour l'entraînement des réseaux :

- Front montant : D = [0 0], W = Pi/6
- Front descendant : D = [Pi/2 Pi/2], W = 0

Conditions initiales et commandes

Deux cas de figures ont été utilisés pour l'entraînement des réseaux :

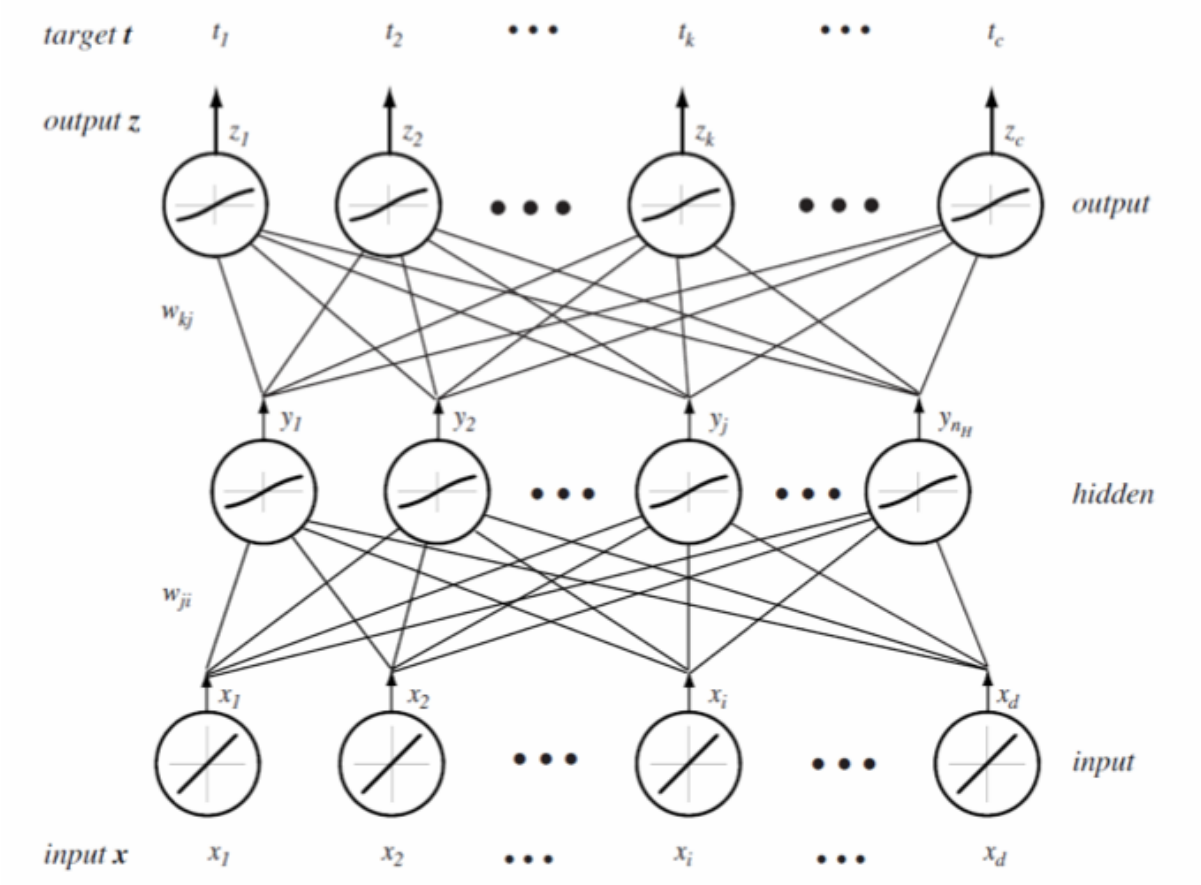
- Front montant : D = [0 0], W = Pi/6
- Front descendant : D = [Pi/2 Pi/2], W = 0
- Simulation sur 400 itérations (4 s)

Conditions initiales et commandes

Deux cas de figures ont été utilisés pour l'entraînement des réseaux :

- Front montant : D = [0 0], W = Pi/6
- Front descendant : D = [Pi/2 Pi/2], W = 0
- Simulation sur 400 itérations (4 s)
- Hyperparamètres $\alpha, \gamma, \beta, \lambda, \epsilon_w, \mathcal{C}$, *etc*

Concepts sous-jacents aux MLP

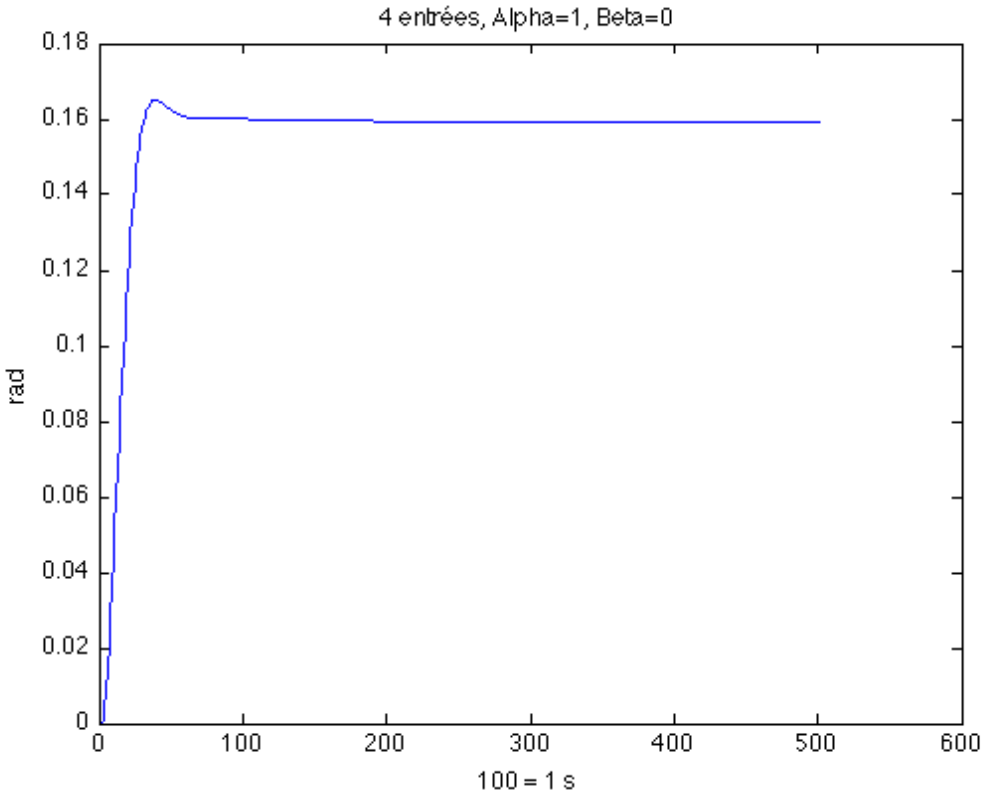


Réseau MLP avec 4 entrées

- Fonction d'activation sigmoïdale
- Quatre entrées : $\varepsilon_\theta, \dot{\theta}, \ddot{\theta}, Ea$
- Trois sorties : multiplicateurs de $\varepsilon_\theta, \dot{\theta}, Ea$
- Cinq perceptrons sur la couche cachée
- Entrées normalisées
- Fonctions *Matlab (NetLAB)* utilisées : mlp, mlpfwd, mlppak, mlpunpak

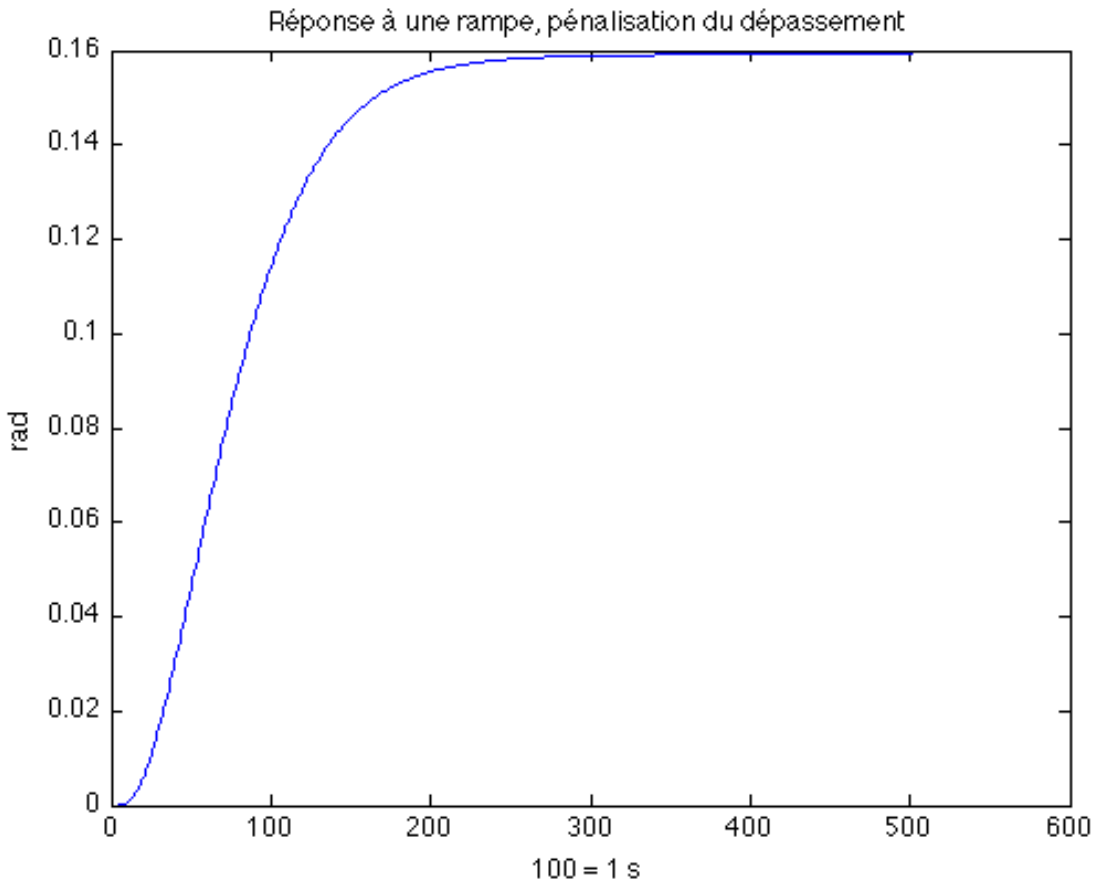
Réseau MLP avec 4 entrées

Paramètres de la fonction de coût pour ces graphes :
$\alpha = 10^3, \gamma = 0, \beta = 0.1$

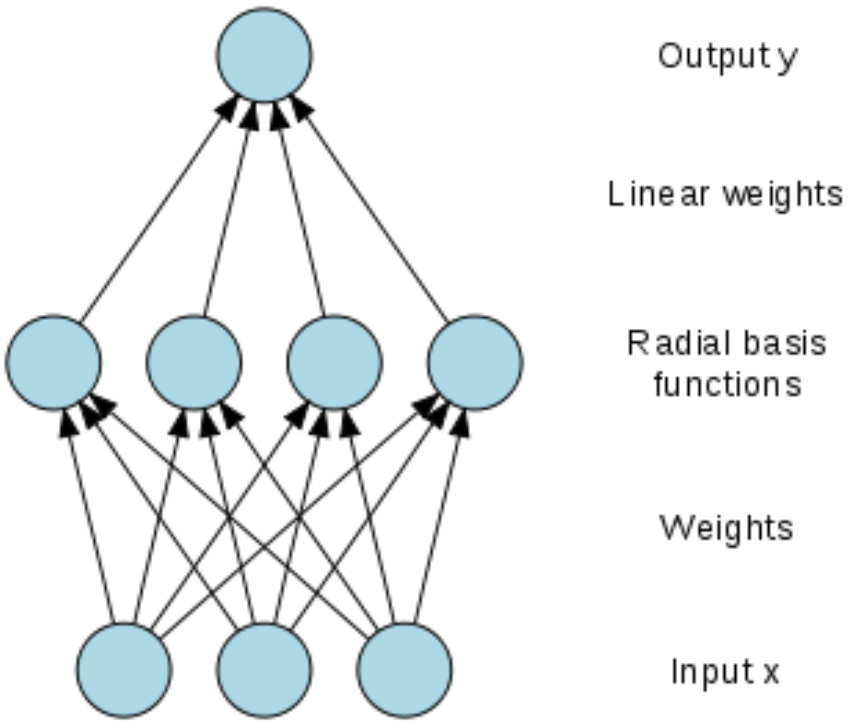


Réseau MLP avec 4 entrées

Réponse à une rampe avec pénalisation du dépassement (terme $\mathcal{C}$)



Concepts sous-jacents aux RBF



Concepts sous-jacents aux RBF

La sortie d'un réseau RBF

$$\varphi(x) = \sum_{i=1}^{N} w_i \rho(\|\mathbf{x} - \mathbf{c_i}\|)$$

La fonction d'activation de chaque noeud de la couche cachée

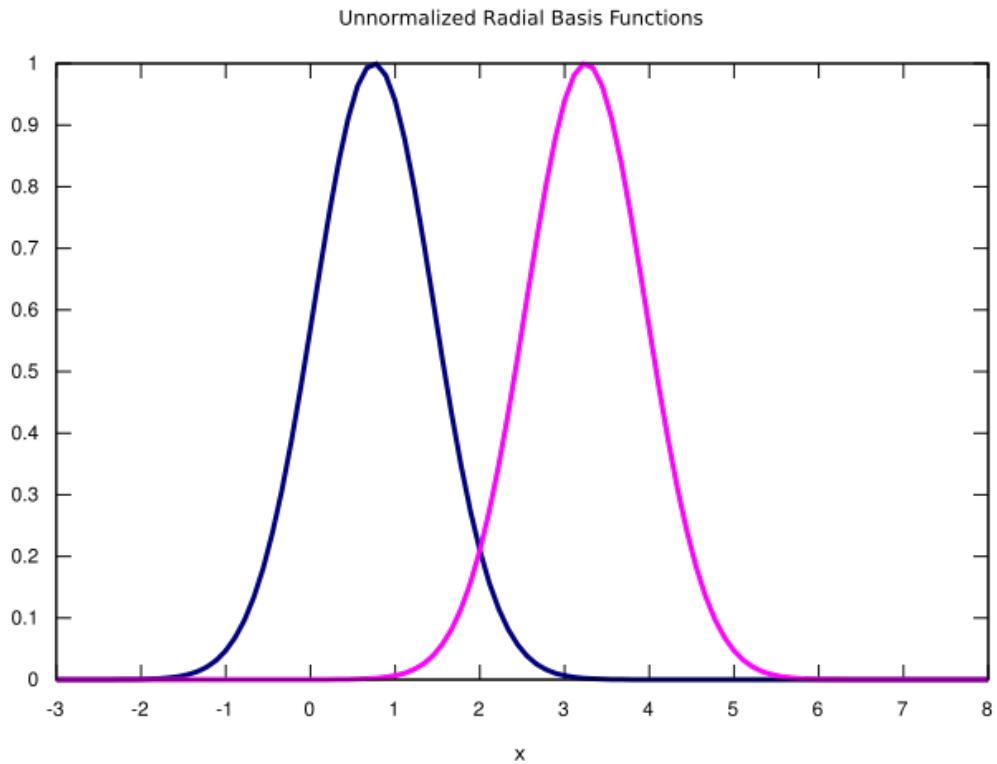
$$\rho(\|\mathbf{x} - \mathbf{c_i}\|) = exp\left[-\beta\|\mathbf{x} - \mathbf{c_i}\|^2\right]$$

Chaque fonction d'activation est dîte locale

$$\lim_{\|x\| \to \infty} \rho(\|\mathbf{x} - \mathbf{c_i}\|) = 0$$

Paramètres à optimiser : $w_i, \mathbf{c_i}, \beta$

Concepts sous-jacents aux RBF

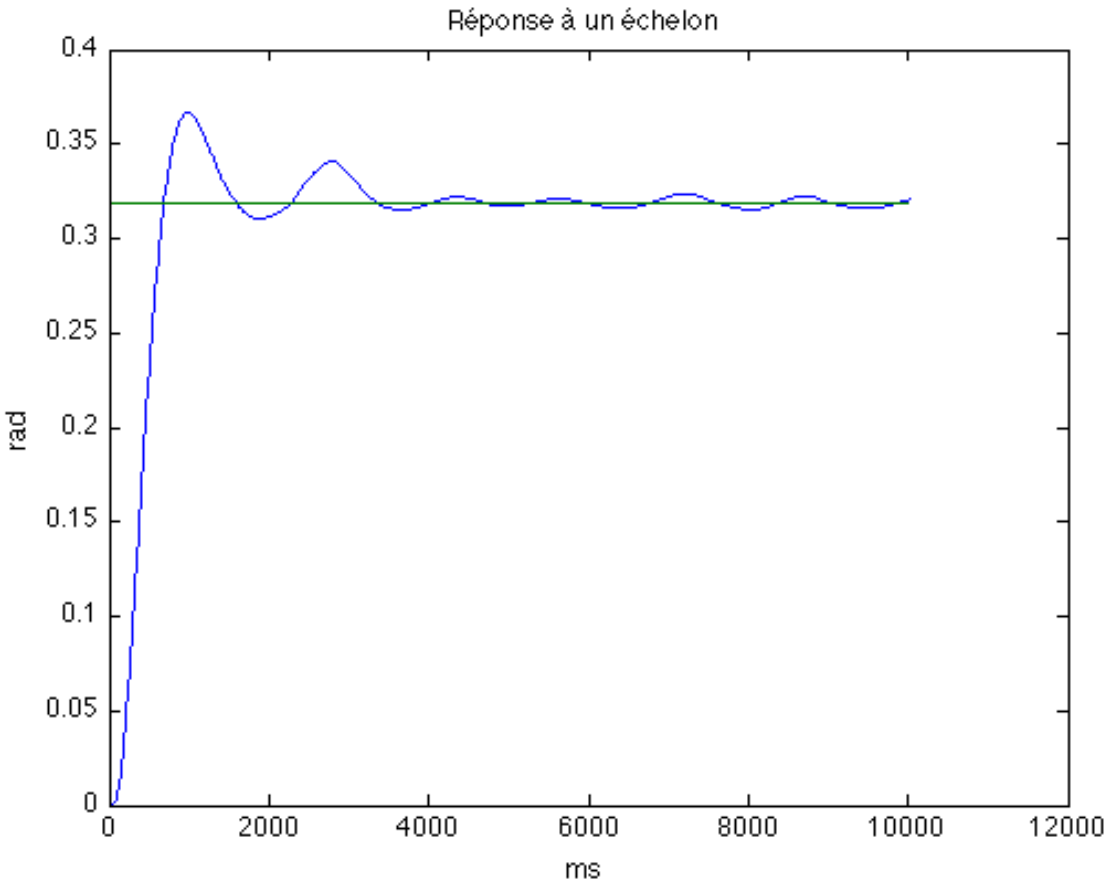


Réseau RBF avec 4 entrées

- Fonction d'activation gaussienne
- Quatre entrées : $\varepsilon_\theta, \dot{\theta}, \ddot{\theta}, Ea$
- Trois sorties : multiplicateurs de $\varepsilon_\theta, \dot{\theta}, Ea$
- Huit perceptrons sur la couche cachée
- Entrées normalisées
- Fonctions *Matlab (NetLAB)* utilisées : rbf, rbffwd, rbfpak, rbfunpak

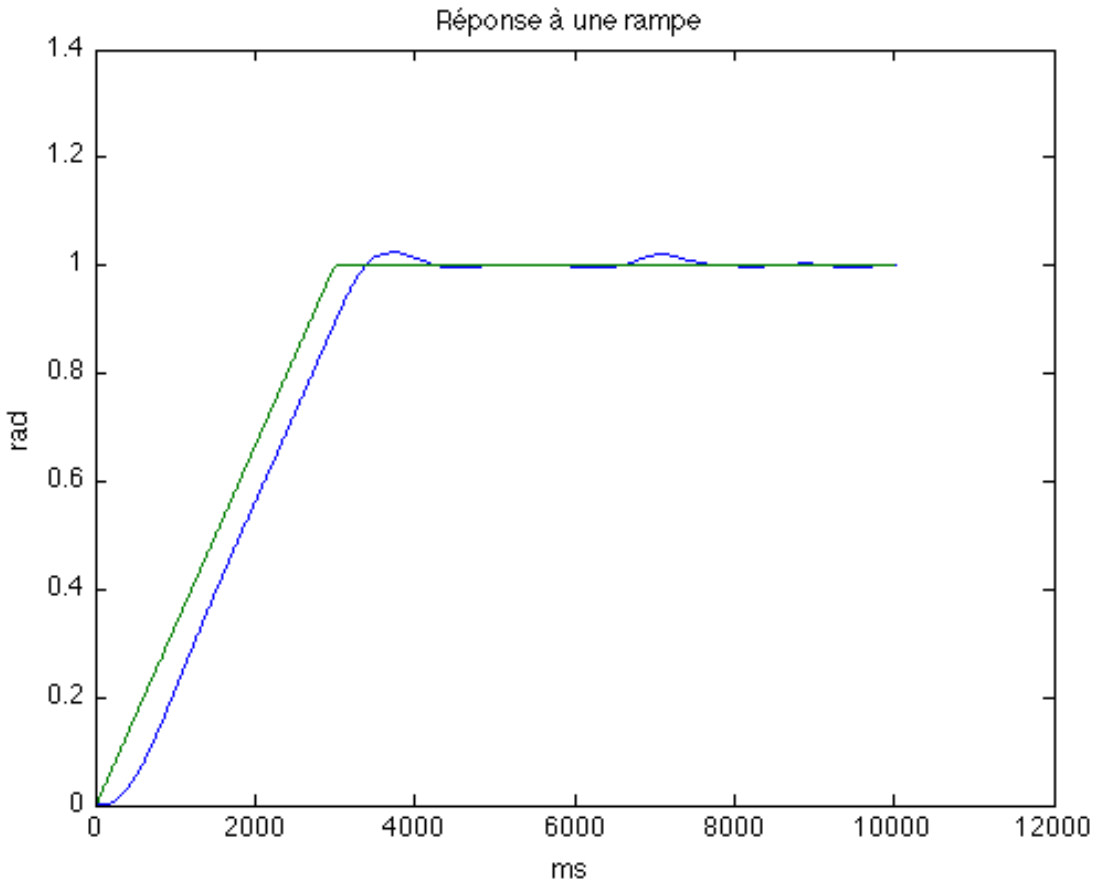
Réseau RBF avec 4 entrées

Réponse à un échelon



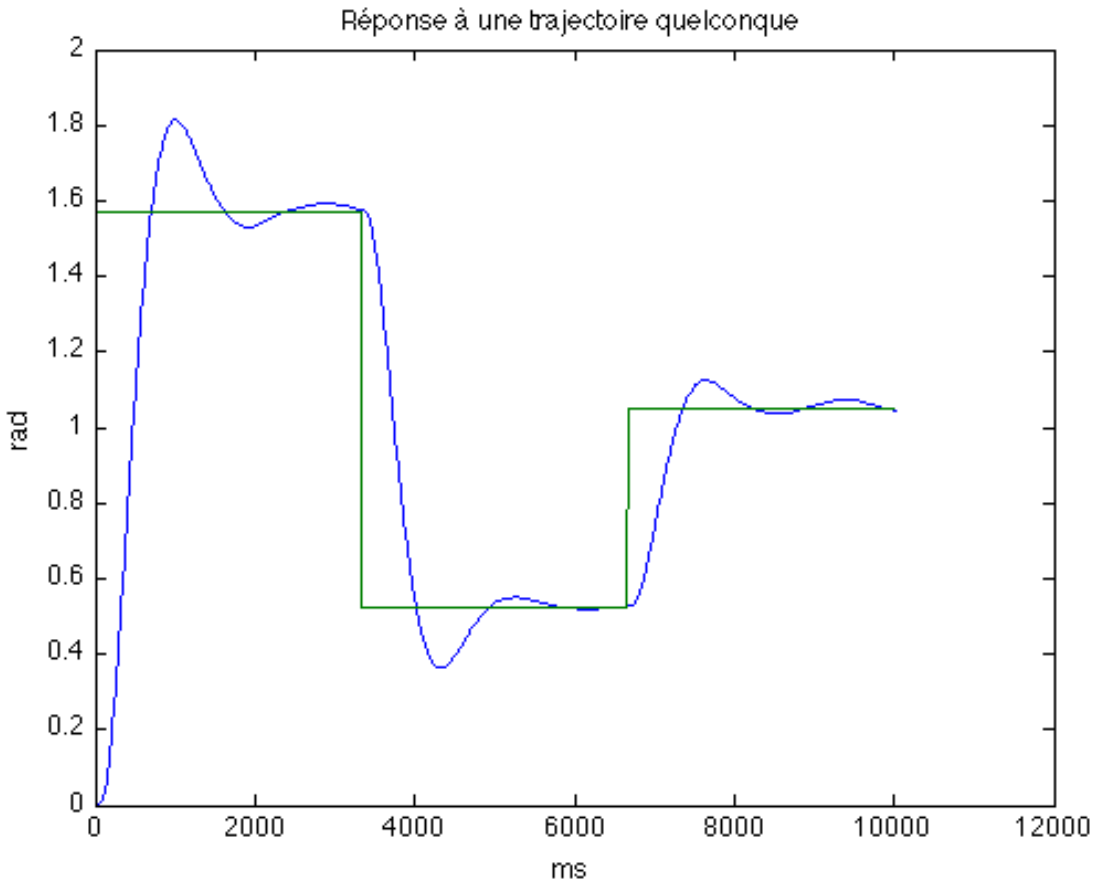
Réseau RBF avec 4 entrées

Réponse à une rampe



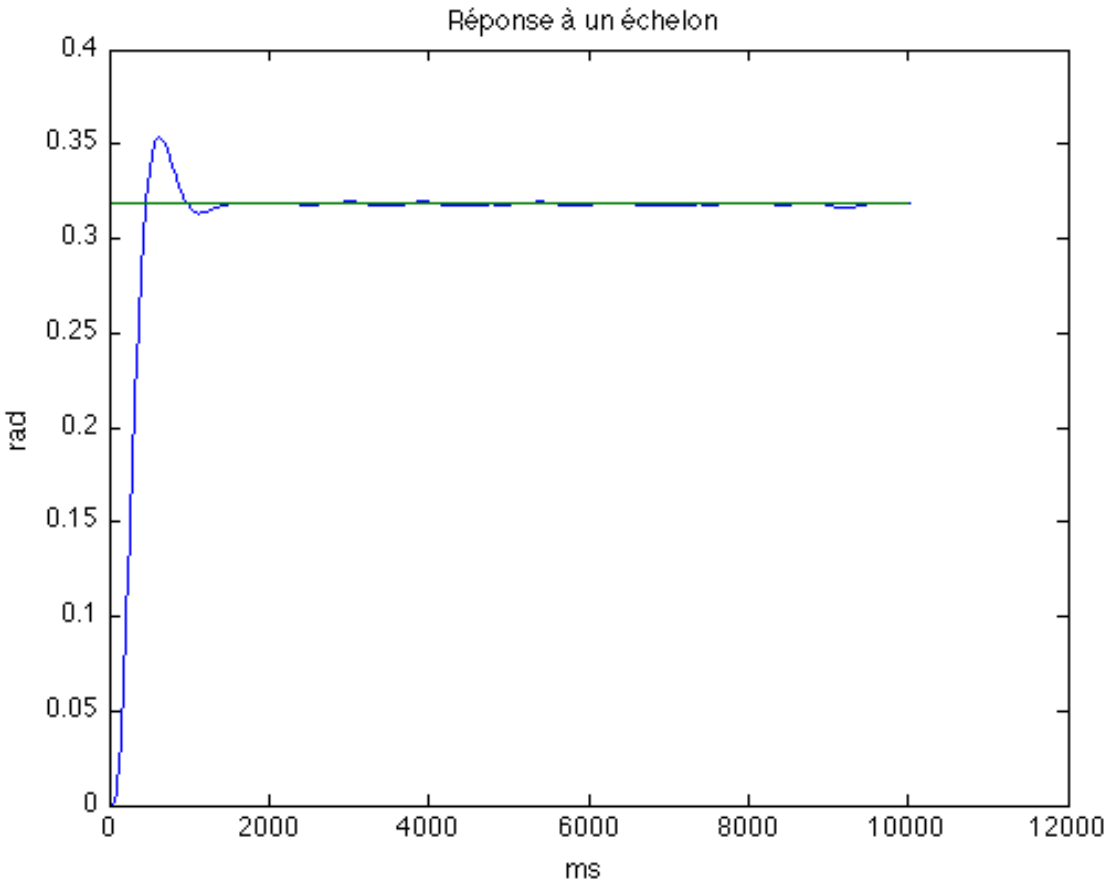
Réseau RBF avec 4 entrées

Réponse à une trajectoire



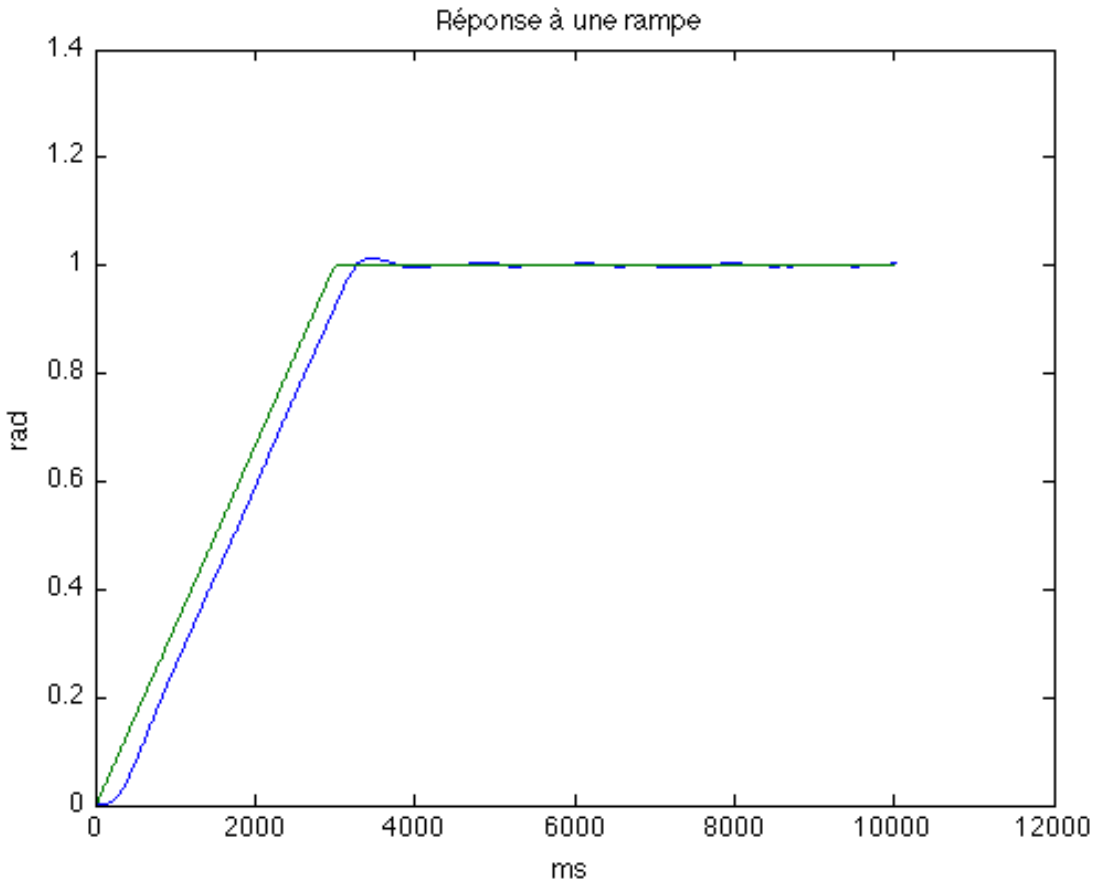
Réseau RBF avec 4 entrées

Réponse à un échelon avec limitation du dépassement



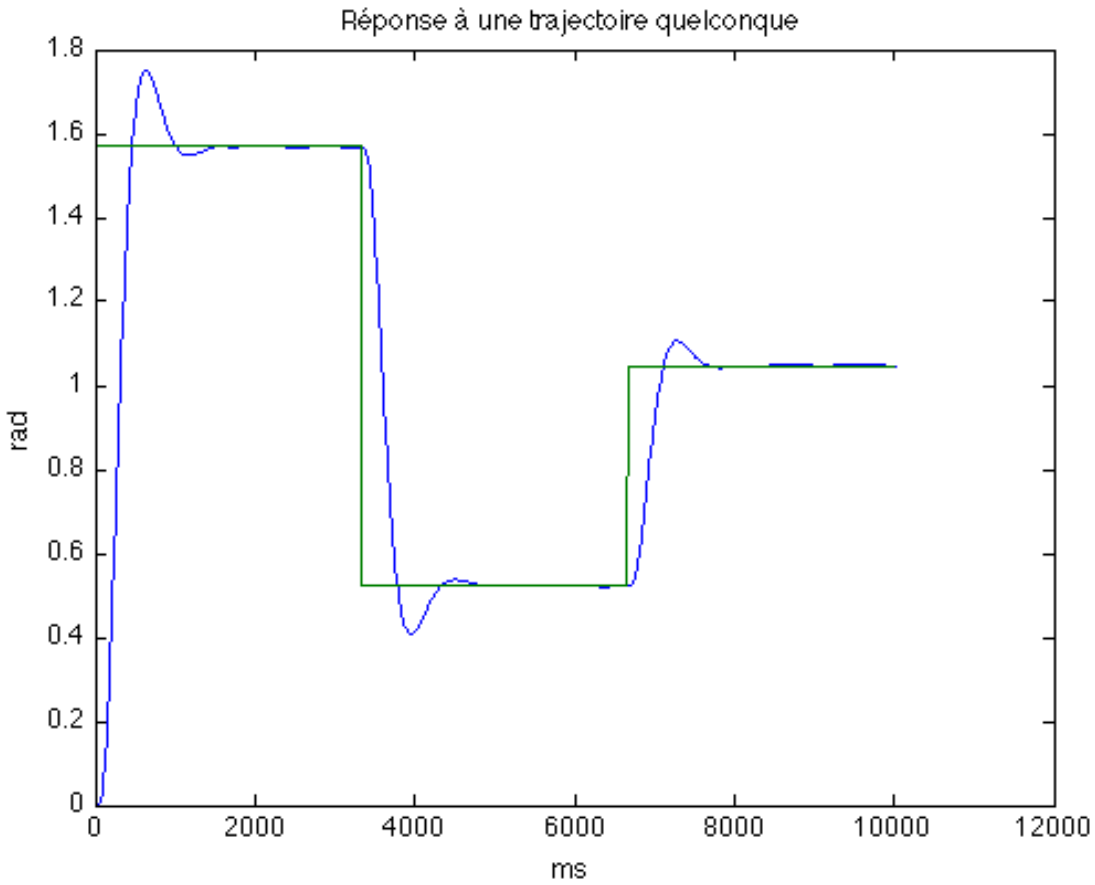
Réseau RBF avec 4 entrées

Réponse à une rampe avec limitation du dépassement



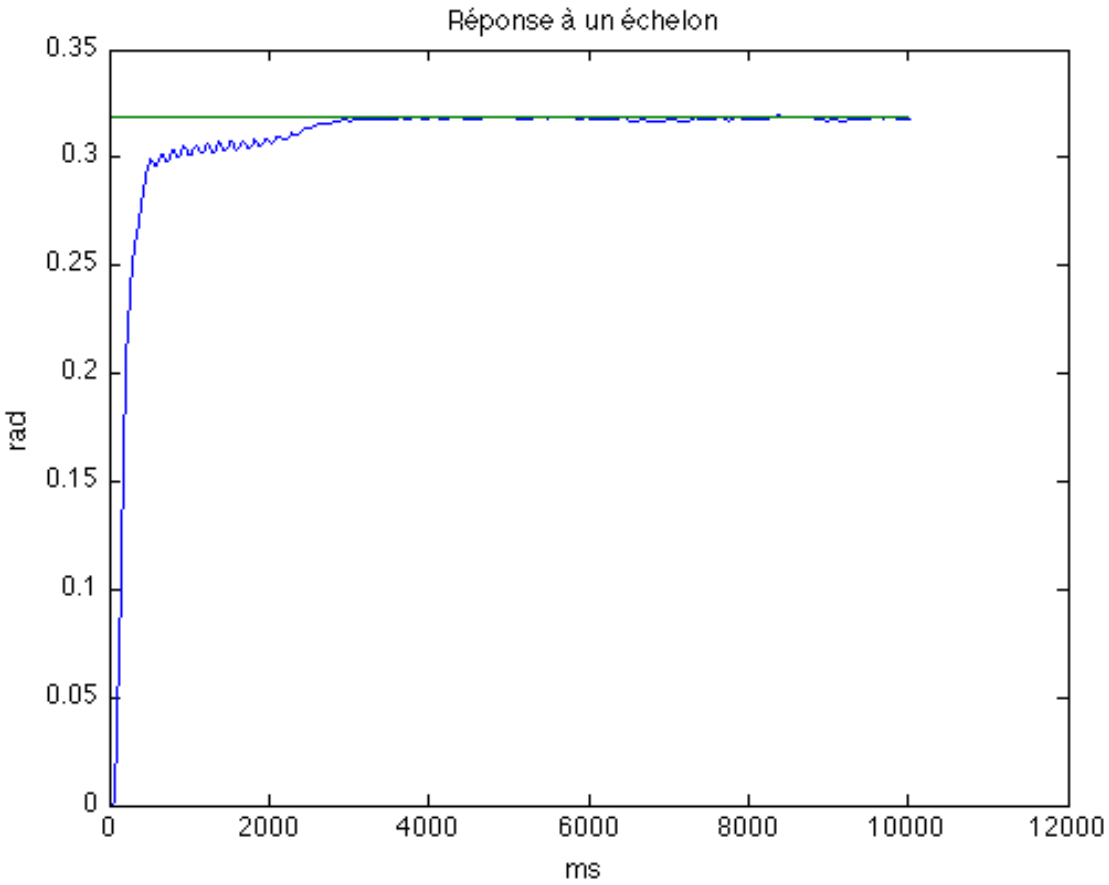
Réseau RBF avec 4 entrées

Réponse à une trajectoire avec limitation du dépassement



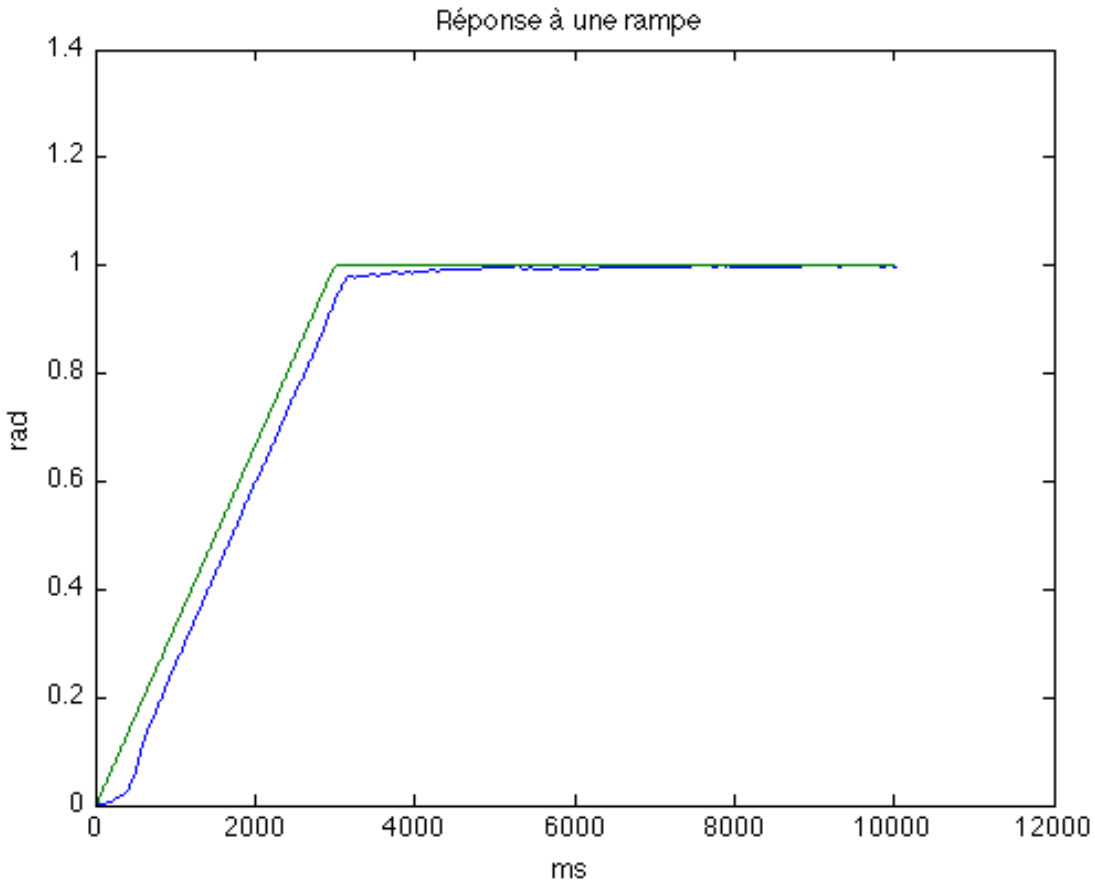
Réseau RBF avec 4 entrées

Réponse à un échelon avec restriction du dépassement



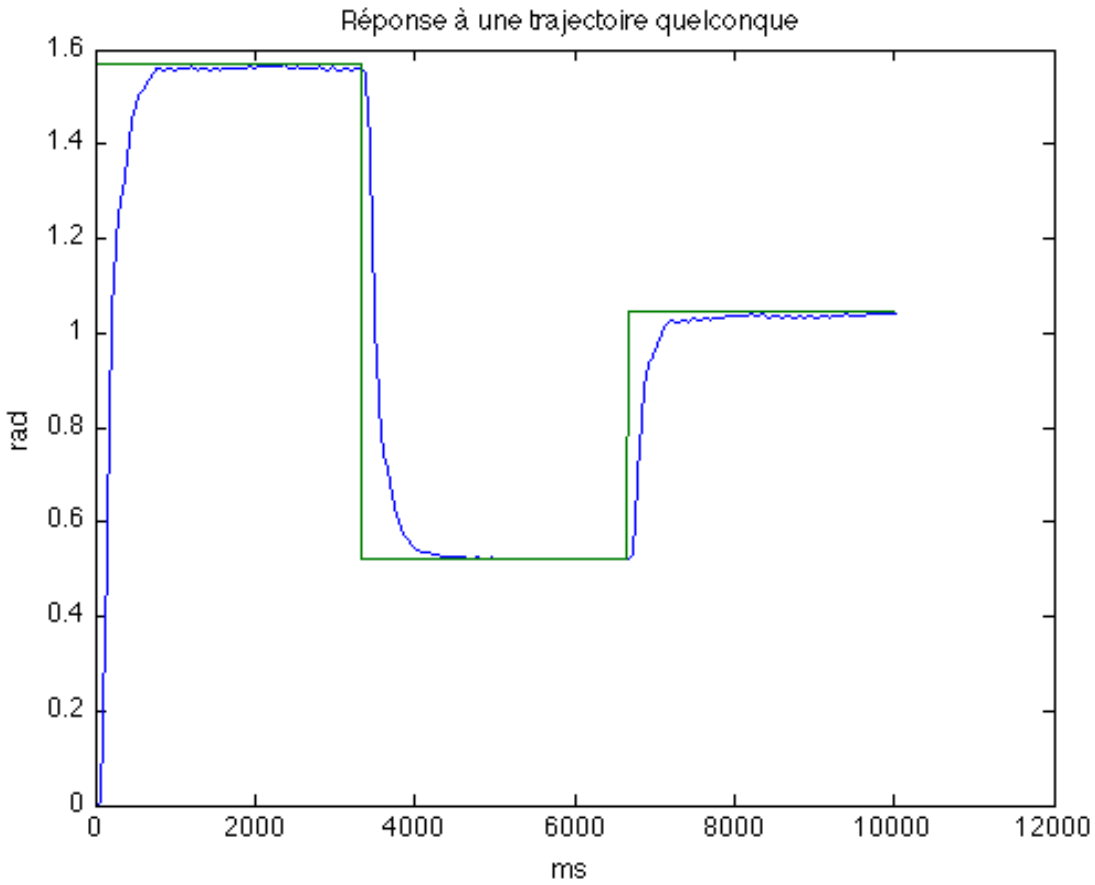
Réseau RBF avec 4 entrées

Réponse à une rampe avec restriction du dépassement

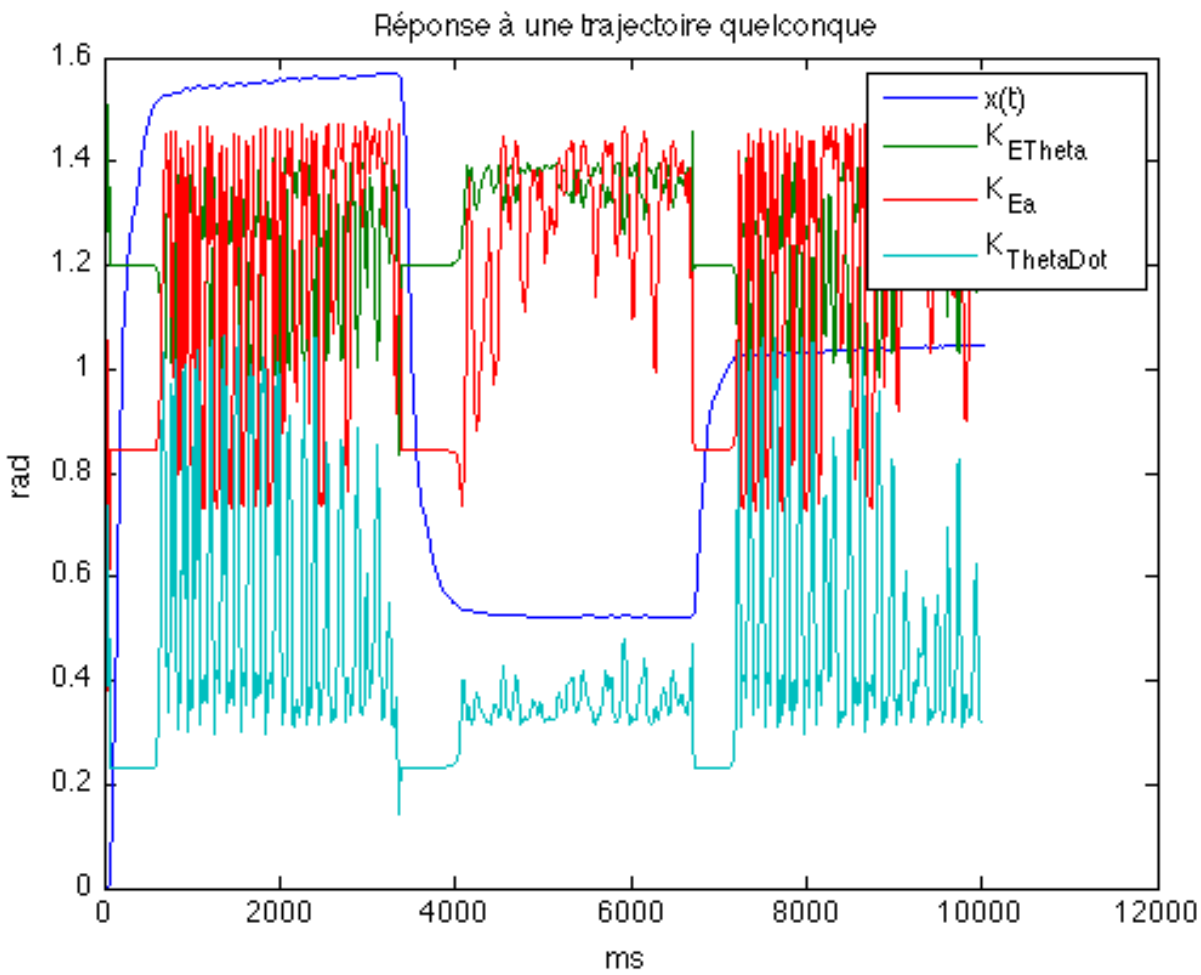


Réseau RBF avec 4 entrées

Réponse à une trajectoire avec restriction du dépassement



Réseau RBF avec 4 entrées - Gains variables dans le temps



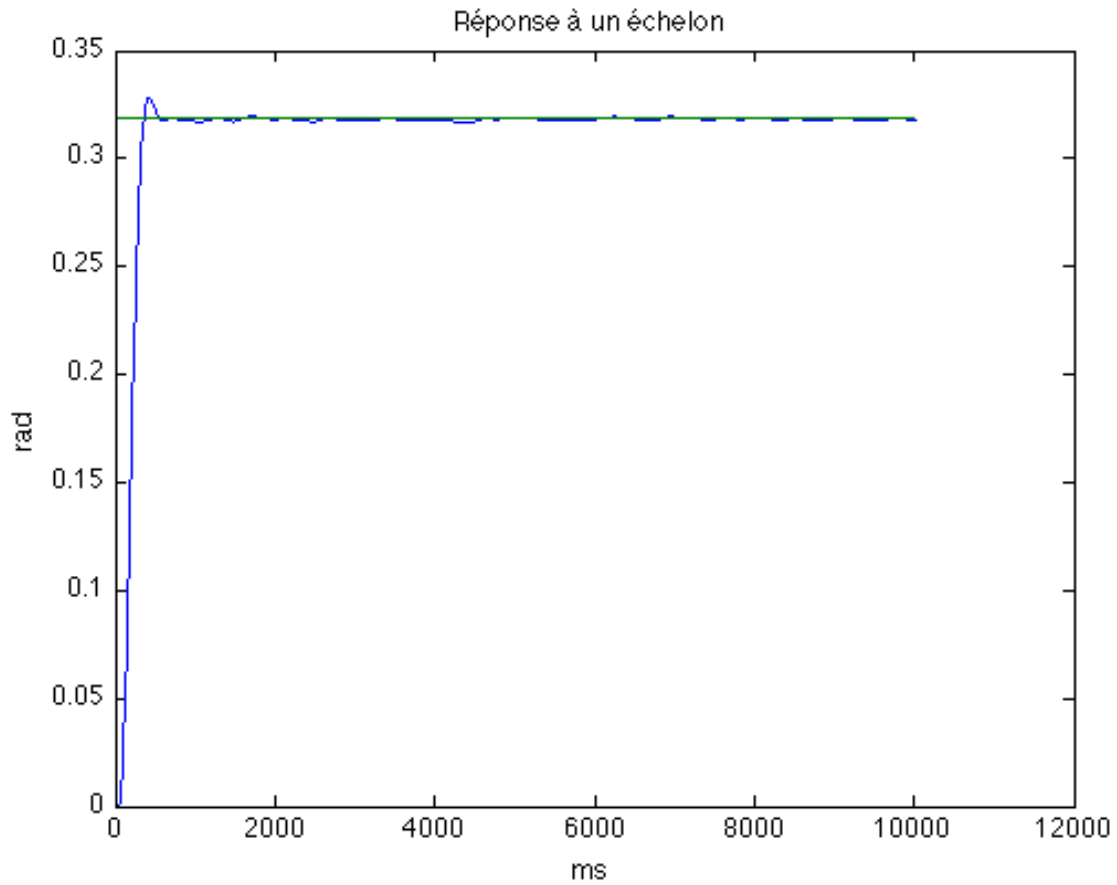
Ajout d'un bruit sur la mesure à la sortie du système

Un bruit blanc est ajouté à la mesure de position y afin de simuler une imprécision au niveau des capteurs du système.

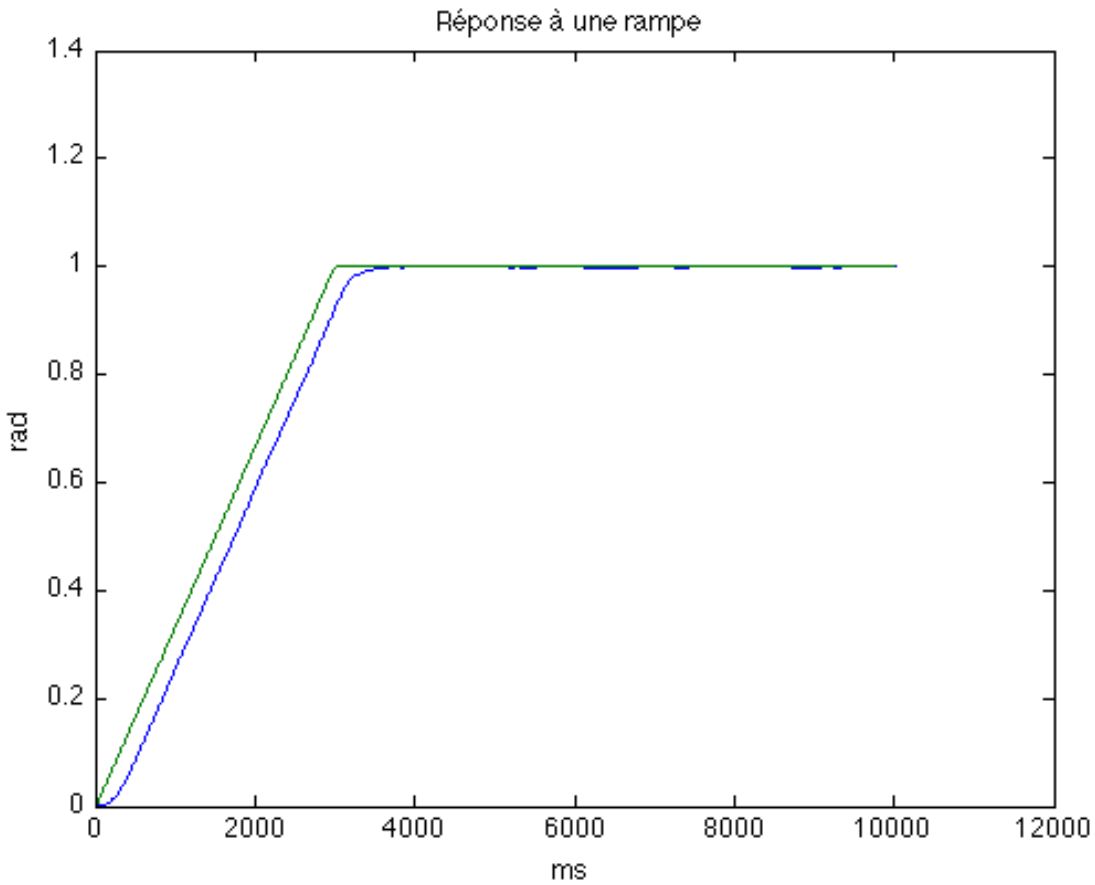
$$y_i = Cx_i + \mathcal{B}(y|0, \lambda)$$

Dans les illustrations suivantes, le réseau MLP a été entrainé avec $\lambda = 2\%$ de y.

Ajout d'un bruit sur la mesure à la sortie du système



Ajout d'un bruit sur la mesure à la sortie du système



Conclusion

- Réseaux RBF beaucoup plus difficiles à faire coller au modèle du contrôleur, oscillations, dépassements

Conclusion

- Réseaux RBF beaucoup plus difficiles à faire coller au modèle du contrôleur, oscillations, dépassements
- Nécessité d'un *priori* fort sur les centres c_i des fonctions d'activation

Conclusion

- Réseaux RBF beaucoup plus difficiles à faire coller au modèle du contrôleur, oscillations, dépassements
- Nécessité d'un *priori* fort sur les centres c_i des fonctions d'activation
- Requièrent beaucoup plus de neurones qu'un MLP pour le même résultat

Conclusion

- Améliorer méthode d'optimisation globale

Conclusion

- Améliorer méthode d'optimisation globale
- Augmenter le nombre de situations d'entrainement

Conclusion

- Améliorer méthode d'optimisation globale
- Augmenter le nombre de situations d'entrainement
- Optimisation par paquet trop longue pour un contrôle adaptatif -> Apprentissage itératif et faculté d'oubli

Conclusion

- Améliorer méthode d'optimisation globale
- Augmenter le nombre de situations d'entrainement
- Optimisation par paquet trop longue pour un contrôle adaptatif -> Apprentissage itératif et faculté d'oubli
- Changer la période d'échantillonnage afin de pouvoir avoir un temps de réponse adéquat de sorte que $\tau_m >> T_s$. Puisqu'on intègre la sortie du réseau de neurones, $Ea(t)$ ne varie pas suffisamment rapidement.

Références

- HRYCEJ, Tomas. *Neurocontrol : Towards an industrial control methodology*, Wiley-Interscience Publications, New York, 1997, 388 pp.
- S. KUO, Shan. *Computer applications of numerical methods*, Addison-Wesley publishing company, Philippines, 1972, 418 pp.
- M. BISHOP, Christopher. *Pattern recognition and machine learning*, Springer Science + Business Media LLC, New York, 2007, 742 pp.
- FORTIN, André. *Analyse numérique pour ingénieurs*, Presses Internationales Polytechnique, Montréal, 2008, 478 pp.