1. Simulation d’un MCC
   1. Traduire la fonction de transfert du correcteur en une équation différentielle temporelle

|  |  |
| --- | --- |
|  | |
|  | CIN : La primitive s’annule en 0  En dérivant l’équation : |

Il faut donc faire apparaitre des dérivées (multiplications par p) pour obtenir les équations différentielles attendues rapidement.

* 1. Proposer le système de 3 équations différentielles du premier ordre permettant de simuler cet asservissement avec la méthode d’Euler

* 1. Programmer la résolution Euler explicite des équations pour un échelon de vitesse de 1 rd/s sur 0,2 secondes avec 1000 points (on ne manipulera que des listes et « odeint » ne sera pas utilisé)

|  |
| --- |
| **def** Euler\_Explicite**(**f**,**y0**,**t0**,**t1**,**dt**):**  t **=** t0  y **=** y0  T **=** **[**t**]**  Y **=** **[**y**]**  i **=** 0  **while** t **<** t1**:** # Important: < et non <=  yp **=** f**(**y**,**t**)**  y **=** **[**y**[**i**]** **+** yp**[**i**]\***dt **for** i **in** **range(len(**y**))]**  i **+=** 1  t **=** t0 **+** i**\***dt # Important: Eviter les erreurs de t+=dt  T**.**append**(**t**)**  Y**.**append**(**y**)**  **return** T**,**Y  **def** F**(**Y**,**t**):**  R **=** 0.45  L **=** 0.0046  ke **=** 0.169  kc **=** 0.17  f **=** 0.01  J **=** 0.01  cr **=** 0  i**,**w**,**u **=** Y  di **=** **(**u**-**ke**\***w**-**R**\***i**)/**L  dw **=** **(**kc**\***i**-**f**\***w**-**cr**)/**J  du **=** **-**Kp**\***Kcapt**\***dw**+**Ki**\*(**Uc**-**Kcapt**\***w**)**  Sol **=** **[**di**,**dw**,**du**]**  **return** Sol  wc **=** 1  Kcapt **=** 10  Ka **=** Kcapt  Kp **=** 0.4  Ki **=** 7.29  Uc **=** wc**\***Ka  U **=** Kp**\***Uc # CIN (d./dt et w nulles au départ)  i0 **=** 0  w0 **=** 0  U0 **=** U  V0 **=** **[**i0**,**w0**,**U0**]**  t0 **=** 0  t1 **=** 0.2  N **=** 1000  dt **=** **(**t1**-**t0**)/(**N**-**1**)**  Lt**,**Y **=** Euler\_Explicite**(**F**,**V0**,**t0**,**t1**,**dt**)**  Li **=** **[**Y**[**i**][**0**]** **for** i **in** **range(len(**Y**))]**  Lw **=** **[**Y**[**i**][**1**]** **for** i **in** **range(len(**Y**))]**  Lu **=** **[**Y**[**i**][**2**]** **for** i **in** **range(len(**Y**))]** |

* 1. Tracer l’évolution de la vitesse moteur et de son intensité en fonction du temps

|  |
| --- |
| **import** matplotlib**.**pyplot **as** plt  plt**.**close**(**'all'**)**  **def** f\_Affiche**(**fig**,**Lx**,**Ly**,**Legende**):**  plt**.**figure**(**fig**)**  plt**.**plot**(**Lx**,**Ly**,**label**=**Legende**)**  plt**.**legend**()**  plt**.**show**()**  plt**.**pause**(**0.0001**)**  f\_Affiche**(**1**,**Lt**,**Li**,**'i'**)**  f\_Affiche**(**1**,**Lt**,**Lw**,**'w'**)**  f\_Affiche**(**1**,**Lt**,**Lu**,**'u'**)** |

* 1. Réaliser la même résolution en utilisant « odeint »

|  |
| --- |
| dt **=** **(**t1**-**t0**)/(**N**-**1**)**  Lt **=** **[**t0**+**i**\***dt **for** i **in** **range(**N**)]**  **from** scipy**.**integrate **import** odeint  Sol **=** odeint**(**F**,**V0**,**Lt**)**  Li **=** **[**Sol**[**i**][**0**]** **for** i **in** **range(len(**Sol**))]**  Lw **=** **[**Sol**[**i**][**1**]** **for** i **in** **range(len(**Sol**))]**  Lu **=** **[**Sol**[**i**][**2**]** **for** i **in** **range(len(**Sol**))]**  f\_Affiche**(**2**,**Lt**,**Li**,**'i'**)**  f\_Affiche**(**2**,**Lt**,**Lw**,**'w'**)**  f\_Affiche**(**2**,**Lt**,**Lu**,**'u'**)** |

* 1. Réaliser le modèle XCOS de l’asservissement et comparer les résultats obtenus

|  |  |
| --- | --- |
|  |  |
|  |