Informatique

**Dérivation numérique**

**Méthode d’Euler**

***Résumé***

|  |  |  |
| --- | --- | --- |
|  | Dérivée première | |
| Type d’Euler | Explicite | Implicite |
| Expression de |  |  |
| Nouvelle valeur lors d’une résolution |  |  |
| Bilan |  | |
| Calcul de la nouvelle valeur |  | Dans , est inconnu ☹  Approximation dans : |

|  |  |
| --- | --- |
| Dérivée seconde | |
| **Méthode à privilégier**  Double Taylor à l’ordre 2 |  |
| Double Euler explicite |  |
| Double Euler implicite |  |
| Euler implicite sur la dérivée seconde et explicite sur la dérivée première |  |
| Explicite sur la dérivée seconde et implicite sur la dérivée première |  |
| Bilan |  |

|  |  |
| --- | --- |
| Une équation différentielle | |
| 1° ordre | 2° ordre |
|  |  |
| Exemples | |
| Chute libre | Pendule |

|  |  |
| --- | --- |
| Système d’équations différentielles | |
| 1° ordre | 2° ordre |
|  |  |

|  |  |  |
| --- | --- | --- |
| Programmation de la méthode sous python  Eviter les cumuls d’erreurs d’arrondis  « t **+=** dt »  Ne pas mettre à jour y mais bien écrire « y **=** » sinon unique y stocké dans Y | **def** Euler\_Explicite**(**f**,**y0**,**t0**,**t1**,**dt**):**  t**,**y **=** t0**,**y0  T**,**Y **=** **[**t**],[**y**]**  i **=** 0  **while** t **<** t1**:**  yp **=** f**(**y**,**t**)**  n **=** len**(**y**)**  y **=** y **+** yp**\***dt # 1 eq ordre 1  y **=** **[**y**[**i**]** **+** yp**[**i**]\***dt **for** i **in** range**(**n**)]** # Sinon  i **+=** 1  t **=** t0 **+** i**\***dt  T**.**append**(**t**)**  Y**.**append**(**y**)**  **return** T**,**Y | |
| **def** f**(**y**,**t**):** # 1° ordre  yp **=** **…**  **return** yp | **def** f**(**Y**,**t**):** # 2° ordre  y**,**yp **=** Y  ypp **=** **…**  **return** ypp  **def** F**(**Y**,**t**):**  yp **=** Y**[**1**]**  ypp **=** f**(**Y**,**t**)**  Sol **=** **[**yp**,**ypp**]**  **return** Sol |
| Remarques | Les erreurs se propagent | |
| Plus les pas (dt, dx…) sont faibles, plus la solution est correcte mais plus les temps de calcul sont longs | |
| Préférer le calcul absolu (du temps par exemple) : t **=** t0 **+** i **\*** dt au calcul relatif qui cumule les erreurs d’arrondi en machine : t **=** t **+** dt | |

|  |  |  |
| --- | --- | --- |
| Résolutions avec Odeint | | |
| 1° ordre - 1 équation | | |
|  | **def** F**(**y**,**t**):**  k**,**g**,**m **=** 0.001**,**9.81**,**0.1  dy **=** **(**k**/**m**)\*(**y**\*\***2**)-**g  **return** dy | |
| Y0 **=** 0 | |
| 1° ordre - 2 équations | | |
|  | **def** F**(**Y**,**t**):**  a**,**b**,**c**,**d **=** 3**,**1**,**1**,**2  H**,**A **=** Y  dH **=** a**\***H **-** b**\***A**\***H  dA **=** **-**c**\***A **+** d**\***A**\***H  dY **=** **[**dH**,**dA**]**  **return** dY | |
| H0**,**A0 **=** 10.0**,**1.0  Y0 **=** **[**H0**,**A0**]** | |
| 2° ordre - 1 équation | | |
|  | **def** f**(**Y**,**t**):**  m**,**R**,**k**,**g **=** 0.1**,**0.1**,**0.001**,**9.81  J **=** m**\***R**\*\***2  y**,**yp **=** Y  ypp **=** **(**1**/**J**)\*(-**k**\***yp**-**R**\***m**\***g**\***sin**(**y**))**  **return** ypp  **def** F**(**V**,**t**):**  y**,**yp **=** V  ypp **=** f**(**V**,**t**)**  dY **=** **[**yp**,**ypp**]**  **return** dY | |
| **from** math **import** pi**,**sin  y0**,**yp0 **=** 45**\***pi**/**180**,**0  Y0 **=** **[**y0**,**yp0**]** | |
| 2° ordre - 2 équations | | |
|  | **def** f**(**Y**,**t**):**  m**,**B**,**g **=** 1**,**1**,**9.81  x**,**xp**,**y**,**yp **=** Y  Vit **=** **(**xp**\*\***2**+**yp**\*\***2**)\*\*(**0.5**)**  xpp **=** **(**1**/**m**)\*(-**B**\***Vit**\***xp**)**  ypp **=** **(**1**/**m**)\*(-**g**-**B**\***Vit**\***yp**)**  **return** xpp**,**ypp  **def** F**(**Y**,**t**):**  x**,**xp**,**y**,**yp **=** Y  xpp**,**ypp **=** f**(**Y**,**t**)**  dY **=** **[**xp**,**xpp**,**yp**,**ypp**]**  **return** dY | |
| x0**,**y0 **=** 0**,**0  xp0**,**yp0 **=** 10**,**10  Y0 **=** **[**x0**,**xp0**,**y0**,**yp0**]** | |
| Code de résolution | |
| t0**,**t1 **=** 0**,**10  N **=** 1000  dt **=** **(**t1**-**t0**)/(**N**-**1**)**  Lt **=** **[**t0**+**i**\***dt **for** i **in** range**(**N**)]**  **from** scipy**.**integrate **import** odeint  Sol **=** odeint**(**F**,**Y0**,**Lt**)**  **print(**Sol**)** | |

|  |  |
| --- | --- |
| Equations aux différences | |
| Principe | Possibilité de résoudre des équations du type :  Donc, des équations qui lient les dérivées de variables selon plusieurs paramètres, comme le temps et l’espace dans l’équation de la chaleur  L’objectif est de discrétiser toutes les dérivées puis d’exprimer :  Solution dans laquelle on trouve s à l’instant d’après , connaissant et à l’instant présent et aux instants précédents |
| Exemple  Chute libre  Un seul paramètre |  |
| Calculer à chaque pas de temps avec Euler implicite :  Cela permet de calculer la valeur de pour chaque pas de temps, puis :  Très intéressant si par exemple, … |
| Prise en compte des conditions initiales | Traiter le/les premier(s) pas de temps de manière particulière en imposant des valeurs aux variables en temps négatif (ex ) permettant d’obtenir la condition initiale voulue.  Exemple : |