Objectifs ***en physique :*** *comparer graphiquement les différents régimes d'amortissement des oscillations libres amorties ; mettre en évidence l'influence de la valeur du facteur d'amortissement et des conditions initiales.*
en informatique : *définir une fonction dans Scilab, l'intégrer à un script, tracer le graphe d'une fonction, superposer plusieurs graphes, et accessoirement sauvegarder le résultat final sous forme d'un fichier image ; utiliser des instructions conditionnelles (if, then, else) et des instructions itératives (boucles conditionnelles while).*

NB ***Compte-rendu :*** *Exercices F2 et G2*
► *scripts à déposer sur Éclat > Espace des classes > Classe ATS > Dossiers partagés > TP INFO physique - 1*

PRÉREQUIS

Vous devez à ce stade être capable de :

- Réaliser des calculs simples dans la console.
- Réinitialiser les variables, vider l'écran de la console.
- Écrire un script dans Scinotes.
- Ouvrir et exécuter un script, arrêter l'exécution d'un script.

Dans le cas contraire, reportez-vous au document annexe : *http://www.phycats.plaf.org/docs/I_annexe.pdf*

Écriture dans Scilab, rappel de l'essentiel

écriture et opérations simples

vraie vie	=	$a+b$	$a-b$	$2,5$	$a\times10^n$	π	$\sqrt{x}$	e^x	$\ln x$
Scilab	=	a+b	a-b	2.5	aEn	%pi	sqrt(x)	exp(x)	log(x)

multiplication, division, puissance dans $\mathbb{R}$

vraie vie	$a\times b$	$a\div b$	a^b
Scilab	a*b	a/b	a^b

opérateurs de comparaison

=	≠	<	⩽	>	⩾
==	<>	<	<=	>	>=

multiplication, division, puissance, appliquées aux vecteurs, aux matrices (tableaux)

	opérations termes à termes (→ graphes)			opérations matricielles	
vraie vie	$a\times b$	$a\div b$	a^b	$a\times b$	$a\div b$
Scilab	a.*b	a␣./b	a.^b	a*b	a/b

␣ : espace

1. FONCTIONS

1.1. Listes à distribution régulière

a) <u>généralités</u>

Scilab est surtout fait pour manipuler des tableaux et des vecteurs. En général, une fonction sera donc une liste de valeurs. Par exemple, si on veut manipuler la fonction cosinus sur l'intervalle $[0, 2\pi]$, pour faire un graphique ou pour simuler un phénomène, on manipulera un vecteur du type `y=cos(liste)`, où `liste` désigne ce qu'on appelle une liste à distribution régulière.

Notez qu'en général, les données réelles ne sont effectivement qu'une liste de nombres (par exemple la température mesurée toutes les secondes par une sonde, envoyée dans LatisPro ou tout autre logiciel d'acquisition et de traitements de données).

b) <u>syntaxe</u>

Rappel (vu en info-maths) : les tableaux à 1 ligne ou à 1 colonne sont nommés *listes*. Les premiers sont également désignés par *vecteurs-ligne*, et les seconds, très utiles pour les graphes, sont des *vecteurs-colonne*.

Voici deux façons de créer des listes dont les éléments sont régulièrement espacés :

- `début:pas:fin` permet de créer une liste dont les éléments commencent à la valeur de `début`, sont espacés de la valeur du `pas` et ne dépassent pas la valeur limite `fin`. En principe, vous connaissez cette première méthode pour l'avoir utilisée en maths.

- `linspace(début,fin,nb)` permet de créer une liste comportant *nb* éléments et allant de la valeur *début* à la valeur *fin* (*linspace* pour `lin`*early* `spaced` *vector* – vecteur à composantes linéairement espacées). Cette seconde méthode garantit que les bornes sont incluses.

c) transposition

Vous apprendrez en maths, un jour, ce qu'est une transposée de matrice. Cela revient, dans un tableau, à inverser les lignes et les colonnes. On utilise l'opérateur « ' » placé après le tableau. Exemple, pour transposer le tableau T : `T'`
Cette opération nous permettra de transformer un vecteur-ligne en vecteur-colonne.

Exercice L1

1) Dans la console, créer et afficher une liste nommée `K` allant de 0 à 2π par pas de $1{,}5$.
2) De même, créer et afficher une liste nommée `L` comportant 5 valeurs régulièrement espacées entre 0 et 2π.

Exercice L2

Considérons un mouvement oscillatoire $X\cos(\omega_0 t)$ avec $X = 10{,}0$ cm et une période propre $T_0 = 1{,}00$ s.

On déduira ω_0 de T_0.
Calculer dans la console $X\cos(\omega_0 t)$ pour $t = 0{,}5$ s.

Calculer ensuite $X\cos(\omega_0 t)$ pour 20 valeurs de t comprises entre 0 et 2 s.

Exercice L3 (facultatif)

1) Créer dans la console un vecteur-colonne `V` $\begin{pmatrix}1\\2\\3\end{pmatrix}$.

Revoir si besoin l'écriture des tableaux dans l'annexe disponible sur *PhyCats*.
http://www.phycats.plaf.org/docs/I_annexe.pdf
2) Créer un vecteur `W` identique sans utiliser « `V` » ou « `;` », et en n'utilisant ↵ qu'une seule fois.

1.2. Fonctions à 1 variable

On peut avoir besoin de définir une fonction afin de l'utiliser plusieurs fois dans un script ou pour rendre la programmation plus claire. La méthode la plus polyvalente consiste alors à utiliser la commande `function`. Limitons-nous pour le moment aux fonctions à une variable, donc aux fonctions du type $y = f(x)$.

a) syntaxe

Soit par exemple la fonction $f(x) = 2x + 3$. Voici comment la définir dans la console :

```
--> function y=f(x); y=2*x+3; endfunction
```

Pour plus de clarté, les « `;` » peuvent être remplacé par des ↵ et des espaces peuvent entourer le signe « = » :

```
--> function y = f(x)
--> y = 2*x+3
--> endfunction
```

La fonction se définit donc en 3 parties (les 3 lignes précédentes) :

- La première partie, ici `function y=f(x)`, est la ligne de déclaration formelle : elle précise le nom de la fonction (ici `f`), le nom des variables d'entrée (ici il n'y en a qu'une, nommée `x`) et le nom des variables de sortie (ici il n'y en a qu'une, comme ce sera le plus souvent le cas, nommée `y`). Notez que ces variables sont muettes, c'est-à-dire qu'elle ne resteront pas en mémoire, contrairement à la fonction `f`.
- La deuxième partie est suite d'instructions Scilab. Ici il n'y en a qu'une : `y = 2*x+3`
- La dernière partie, `endfunction`, marque évidemment la fin de la définition de la fonction.

La fonction ayant été définie, elle peut être utilisée pour calculer des valeurs :

```
--> f(10)
 ans  =
    23
```

Elle peut également s'appliquer en une seule opération à une liste de valeurs (série d'abscisses) pour obtenir une autre liste de valeurs (série d'ordonnées).

```
--> f(0:3:10)
 ans  =
    3.    9.    15.    21.
--> f(linspace(0,10,5))
 ans  =
    3.    8.    13.    18.    23.
```

b) fonction et variables : quels noms utiliser ?

Remarquons tout d'abord que dans l'exemple précédent, on a défini classiquement `y=f(x)` : on manipule ensuite "`f`". Le nom de la fonction est donc `f` et non pas `y` (qui désigne la variable de sortie, muette).

Supposons que nous souhaitions définir la vitesse en fonction du temps.
La variable d'entrée est le temps, noté `t`, ce qui est naturel. On pourrait alors être tenté, par analogie, d'écrire `v=f(t)`. Mais si ensuite on se réfère à la vitesse il faudrait la désigner par `f`, ce qui n'est pas très explicite.
La fonction devra donc de préférence être nommée en référence à la grandeur étudiée, ce qui conduit ici à `v(t)`.
Reste la variable de sortie.
On peut garder `y`, mais en physique on réserve en général `x`, `y` et `z` aux coordonnées d'un point. Ce n'est donc pas idéal.
S'agissant d'une variable muette, qui pas de signification particulière, on peut aussi lui donner un nom mathématiquement et physiquement insignifiant, par exemple « `sortie` », « `resultat` », « `#` »… on a le choix.
L'écriture la plus simple, cependant, si on accepte de nommer de façon identique la fonction et la variable muette qui sert à la définir, consiste à écrire `v=v(t)`. Exemple :

```
--> function v = v(t)
  > v = 12-10*t
  > endfunction
```

Exercice F1

1) Définir dans la console la fonction $x(t) = X\cos(\omega_0 t)$.

2) L'appliquer à différentes valeurs de t.
3) Utilisez cette fonction pour déterminer 10 valeurs de la position à des dates régulièrement espacées entre $t = 0$ et $t = 1$ s. Les résultats devront être présentés en colonne.

1.3. Définition par un script

a) contextualisation

Reprenons le problème des oscillations libres, mais cette fois-ci de manière plus générale, notamment en tenant compte des frottements.

Exercice de physique → à la maison

Considérons sur l'axe Ox le mouvement oscillatoire amorti d'un point autour de sa position d'équilibre x_{eq}.
La période propre T_0 est supposée connue. Nous envisageons différentes valeurs du facteur d'amortissement.
On rappelle que :

- pour $\xi < 1$: $x(t) = x_{eq} + \mathrm{e}^{-\xi\omega_0 t}\left(A\cos(\omega'_0 t) + B\sin(\omega'_0 t)\right)$ avec $\omega'_0 = \omega_0\sqrt{1-\xi^2}$
- pour $\xi > 1$: $x(t) = x_{eq} + A\mathrm{e}^{-\delta_1 t} + B\mathrm{e}^{-\delta_2 t}$ avec $\delta_1 = \omega_0\left(\xi - \sqrt{\xi^2-1}\right)$ et $\delta_2 = \omega_0\left(\xi + \sqrt{\xi^2-1}\right)$
- pour $\xi = 1$: $x(t) = x_{eq} + (At + B)\mathrm{e}^{-\omega_0 t}$

À $t = 0$, ce point passe par x_0 avec la vitesse v_0. Montrer que :

- pour $\xi < 1$: $A = x_0 - x_{eq}$ $\qquad B = \dfrac{v_0 + \xi\omega_0\left(x_0 - x_{eq}\right)}{\omega'_0}$
- pour $\xi > 1$: $A = \dfrac{\delta_2\left(x_0 - x_{eq}\right) + v_0}{\delta_2 - \delta_1}$ $\qquad B = -\dfrac{\delta_1\left(x_0 - x_{eq}\right) + v_0}{\delta_2 - \delta_1}$
- pour $\xi = 1$: $A = v_0 + \omega_0\left(x_0 - x_{eq}\right)$ $\qquad B = x_0 - x_{eq}$

b) rappels sur les scripts

Revoir au besoin comment écrire un script dans Scinotes (annexe) : http://www.phycats.plaf.org/docs/I_annexe.pdf

c) définition d'une fonction dans un script

Exemple : oscillations libres amorties en régime pseudopériodique.
Écrivez (ou copiez-collez) ce script, comportant des exemples de valeurs pour les différents paramètres :

```
// Oscillation libres amorties - Régime pseudopériodique

// paramètres
xeq = -0.5;                 // position d'équilibre
T = 0.8;                    // période propre
ksi = 0.15;                 // facteur d'amortissement
x0 = 1;                     // position initiale
v0 = 0;                     // vitesse initiale
w = 2*%pi/T ;               // pulsation propre

function x=x(t)
    wp = w*sqrt(1-ksi^2);     // pseudopulsation
    A = x0-xeq                // coeff du cosinus
    B = (v0 + ksi*w*A)/wp     // coeff du sinus
    x = xeq + exp(-ksi*w*t).*(A*cos(wp*t) + B*sin(wp*t)) // attention : ".*"
endfunction
```

Enregistrez-le sous ' *oscillations_pseudopériodique - fonction.sce* ' (par exemple).
Lancez le script. Un message nous prévient que le nom `x` a déjà été utilisé. Pas de problème.

```
Attention : Redéfinition de la fonction : x
```

Il ne se passe rien de plus, forcément.
Essayez d'utiliser la fonction dans la console, pour calculer *x(1)* par exemple :

```
--> x(1)
 ans  =
  - 0.3892332
```

Ça marche !
Essayez de lancer le script après avoir modifié certains paramètres : le résultat sera évidemment différent.

1.4. Rappel sur les instructions conditionnelles

Nous parlons ici des instructions `if... then... else... end`. Revoir le doc *info-math* si nécessaire.
Revoyons ces notions sur un exemple : selon le temps qu'il fait, comment occuper son après-midi ? Pour décrire la météo, nous introduirons tout d'abord la variable `ciel` prenant les valeurs `dégagé`, `couvert` ou `pluvieux`.

a) structure à deux branches

```
if ciel == "dégagé" then
   activité = "baignade"
else
   activité = "promenade"
end
```

b) structure à trois branches (ou plus)

```
if ciel == "dégagé" then
   activité = "baignade"
elseif ciel == "pluvieux" then
   activité = "cinéma"
else
   activité = "promenade"
end
```

c) structures imbriquées

Affinons le raisonnement en ajoutant la variable `température` prenant les valeurs `chaud` ou `froid`.

```
if ciel == "dégagé" then
   if température == "chaud"
      activité = "baignade"
   else
      activité = "promenade"
   end
elseif ciel == "pluvieux" then
   activité = "cinéma"
else
   activité = "promenade"
end
```

Exercice F2

Définir une fonction permettant d'obtenir l'équation horaire *x(t)* quel que soit le régime d'amortissement, en utilisant des *instructions conditionnelles*. Le script de la fonction contiendra la position d'équilibre, la période propre ainsi que les position et vitesse initiales. Il demandera à l'utilisateur d'entrer la valeur du facteur d'amortissement en utilisant la fonction `input` : `ksi = input("valeur de ksi : ");`

Pour la suite, le but sera d'obtenir le graphe de *x(t)*. Nous devons tout d'abord rappeler comment tracer un graphe avec Scilab.

2. GRAPHE D'UNE FONCTION À 1 VARIABLE

2.1. Les séries de données

Pour tracer une courbe, il faut fournir à Scilab une série d'abscisses et une série d'ordonnées (calculées, à l'aide de la fonction, à partir des abscisses). Pour disposer d'une méthode adaptée à toute situation, notamment la superposition simultanée de plusieurs courbes, ces séries doivent être des *vecteurs-colonnes*.

Exercice G1

Pour le problème qui nous intéresse, c'est le temps qui est en abscisse. Nous choisirons un intervalle de temps allant de *0* à *6* s, avec 15 points. Dans la console :
- définissez le vecteur-colonne `t` correspondant aux abscisses.
- définissez le vecteur-colonne correspondant aux ordonnées.

2.2. Tracé avec plot2d

Nous disposons désormais de tous les éléments et il nous suffit, pour obtenir le tracé, d'utiliser la fonction `plot2d`, en lui fournissant comme arguments les séries d'abscisses et d'ordonnées : `plot2d(abscisses,ordonnées)`

```
--> plot2d(t,x(t))
```

Comment ça, c'est pas terrible ? Trouvez une solution.
Pour info : *to plot points on a graph* se traduit par *reporter des points sur un graphique*. Donc en fait, Scilab pose les points et les relie par des segments…

Remarque : il existe également une fonction nommée `plot` qui fait à peu près la même chose, mais avec un peu moins d'options. La différence n'est pas flagrante…

NB : si on retrace un graphique, la deuxième courbe se superposera à la précédente.
Pour nettoyer le dernier graphique, on utilise la commande `clf` (`cl`*ear* `f`*igure*).
Pour le supprimer, on utilise `xdel` (*graph*`ics` *window* `del`*ete)*

Exercice G2

Reprendre le script précédent destiné à utiliser la fonction *x(t)* définissant des oscillations faiblement amorties (désigné ici par '*oscillations_pseudopériodique - fonction.sce*'). Ajouter les lignes nécessaires pour obtenir le tracé (amélioré, tant qu'à faire) de *x(t)*.
Enregistrer cette version du script sous '*oscillations_pseudopériodique - fonction + graphe.sce*', par exemple.

C'est en noir sur noir, les axes ne sont pas vraiment bien placés… Bref, on peut faire mieux.

2.3. Mise en forme sommaire

a) <u>couleur de la courbe</u>

Il s'agit d'un paramètre facilement accessible par les options de la commande `plot2d`.

`plot2d(abscisses,ordonnées,`*`style=n`*`)`

Rappel : `abscisses` et `ordonnées` sont les séries d'abscisses et d'ordonnées sous forme de vecteurs-colonne.
`style=`*`n`* définit la couleur, ou le type de tracé (trait plein, pointillés…). $n \in \mathbb{Z}$:

- si $n = 1$ la courbe sera en trait plein noir (valeur par défaut si aucun style n'est indiqué)
- si $n > 1$ la courbe sera en couleur ; on peut voir la correspondance entre le nombre et la couleur (carte des couleurs) grâce à la commande `getcolor()`
  Exemple : `plot2d(x,y,style=5)` permet d'obtenir une courbe rouge.
- si $n \leqslant 0$ la courbe sera remplacée par des marqueurs (prévoir d'espacer les points, pour plus de visibilité) :
  $0 \rightarrow \cdot$ $-1 \rightarrow +$ $-2 \rightarrow \times$ ... $-9 \rightarrow \circ$ $-10 \rightarrow \star$

Exercice G3

Modifier le script précédent de manière à ce que le graphe de $x(t)$ soit tracé en bleu.

b) couleur de la courbe : choix au moment du tracé

Voici une variante très intéressante pour laisser le choix de la couleur à l'utilisateur :

```
plot2d(abscisses,ordonnées,style=getcolor())
```

Au moment du tracé, l'utilisateur choisit sa couleur dans la palette affichée et clique sur [OK].

c) ajout d'une légende

On utilise la commande `legend` dont la syntaxe est la suivante :

```
hl=legend("texte_de_la_légende");
```

⚠ confusion possible entre la lettre `l` (ℓ) et le chiffre `1` : `hl` ≠ `h1` (`hl` = h*andle* l*egend*, probablement).

2.4. Mise en forme approfondie

L'objet de cette courte formation à Scilab n'étant pas la mise en forme la plus élégante possible du graphe d'une fonction (cela nous prendrait beaucoup trop de temps), je vous propose de passer par un script "fait maison" pour ajouter un fond, ajouter une grille, ajouter et positionner un titre et nommer les axes et les faire passer par l'origine.

a) présentation du script

Ce script se nomme '*options_graphes.sce*' et sa dernière version est téléchargeable sur *PhyCats* :
http://www.phycats.plaf.org/docs/index.php#I

Je vous propose d'analyser son contenu, en particulier l'**instruction itérative** `while…end` (boucle conditionnelle), l'**instruction conditionnelle** `if…end` (voir exercice G4 → compte-rendu à faire) et, mais c'est moins important, l'utilisation de la commande `input`.

Exercice G4

Expliquer le fonctionnement et le rôle des instructions itératives et conditionnelles du script précédent. En résumé, il s'agit de répondre à la question : que font ces instructions ? Proposer éventuellement une amélioration possible.

b) utilisation basique

Lancer l'exécution du script après avoir réalisé le tracé par `plot2d`.

c) autre méthode : lancement du script '*options_graphes.sce*' depuis le script initial

Il s'agit donc de modifier le script de l'exercice G3 de manière à ce que le graphe de $x(t)$ soit enrichi par '*options_graphes.sce*'. Ceci peut se faire en ajoutant au script la commande d'exécution de '*options_graphes.sce*'.
Cette commande se nomme `exec`, on l'a déjà vue dans la console.
Il faut donner à exec l'adresse du script à lancer, adresse dans l'explorateur Windows qu'il faut donc noter. Cette adresse étant désignée ici par "…chemin", la commande s'écrit :

```
exec('…chemin\options_graphes.sce', -1)
```

Sauvegarder cette version du script, ' *oscillations_pseudopériodique - fonction + graphe amélioré.sce*', par exemple.

2.5. Superpositions de graphes (pour mémoire)

Il existe différentes méthodes, plus ou moins évoluées. Je vous propose d'aller au plus simple.

a) tracé dans une fenêtre existante

Dans le cas où une seule figure (fenêtre graphique) est ouverte, il vous suffit de commander le tracé par `plot2d` pour l'obtenir automatiquement dans cette figure, à condition évidemment de ne pas utiliser entre-temps la commande `xdel(winsid())` . Si plusieurs figures coexistent, `plot2d` utilisera la dernière figure créée par Scilab.
Si un changement d'échelle est nécessaire pour que chaque abscisse ait son image par la fonction, la figure sera automatiquement agrandie.

b) plusieurs tracés successifs dans un même script

Il suffit d'utiliser plusieurs fois `plot2d` (avec des styles différents, tant qu'à faire). En cas de besoin, la série d'abscisses peut être redéfinie entre les deux tracés (largeur de l'intervalle, nombre de points).

c) tracés simultanés

Cette méthode offre l'avantage de permettre l'ajout d'une légende. Syntaxe, avec deux fonctions du type $x(t)$:

```
plot2d(t,[x1(t),x2(t)],[style1,style2])
hl=legend(["légende1";"légende2"]);
```

Remarque : dans le cas de deux fonctions désignées de la même façon, par exemple `x(t)`, vous pouvez très bien, au lieu de renommer les fonctions, utiliser la parade suivante :

```
funcprot(0)  //pour désactiver la protection des fonctions,
// et ainsi éviter le message "Attention : redéfinition de la fonction x"
// intervalle de temps (exemple)
t = linspace(0,2,100)';

//définition de la 1e fonction
function x=x(t)
    x = …
endfunction
x1=x(t);

//définition de la 2e fonction
function x=x(t)
    x = …
endfunction
x2=x(t);

// tracé
plot2d(t,[x1,x2],[2,5])
hl=legend(["1er cas";"2e cas"]);
funcprot(1)  //pour réactiver la protection des fonctions (mode par défaut)
```

2.6. Sauvegarde du graphique

a) sauvegarde du fichier Scilab

Menu Fichier > Enregistrer…
Vous obtenez un fichier SCG exploitable par Scilab. Bof.

b) sauvegarde directe en image bitmap

Menu Fichier > Exporter vers…
Choisissez le format PNG de préférence. Attention, il faut entrer le nom du fichier, mais également son extension *.png*, ce n'est pas automatique. Le résultat est une image à relativement basse résolution, ce qui ne convient pas parfaitement à une impression. On peut largement s'en contenter, mais il y a mieux.

c) sauvegarde en image vectorielle

Menu Fichier > Exporter vers…
Choisissez le format SVG. Là aussi, il faut entrer le nom du fichier suivi de son extension *.svg*.
Si vous disposez d'un logiciel dessin vectoriel (le logiciel libre *inkscape* – https://inkscape.org/fr, par exemple), vous pouvez alors rouvrir le fichier SVG dans ce logiciel et l'exporter en PNG avec une meilleure résolution.

3. EXPLOITATION PHYSIQUE

Enfin ! Le but était de comparer graphiquement les différents régimes d'amortissement des oscillations libres amorties et de mettre en évidence l'influence de la valeur du facteur d'amortissement et des conditions initiales.

3.1. Régime pseudopériodique

Reprenons le problème du §1.3.a. Nous avons pour l'instant mis en place les bases de l'exploration du régime pseudopériodique : nous avons obtenu un tracé pour $x_{eq} = -0{,}5$ cm ; $T = 0{,}8$ s ; $\xi = 0{,}15$; $x_0 = 1$ cm ; $v_0 = 0$ m/s.

a) <u>influence de la période</u>

Relancer le script précédent en modifiant le titre : ajouter "influence de T".
Le sauvegarder sous un autre nom, ' *oscillations_pseudopériodique - graphes suivants.sce*', par exemple.
Modifier le script de manière à superposer d'autres courbes à la figure obtenue (voir §2.5).
Modifier la valeur de T.
Tracer sur la même figure (n°0 en principe) le graphe de $x(t)$, dans une couleur différente : utiliser les options de `plot2d`.
Renouveler l'opération avec une 3e valeur de T, et une 3e couleur.
Exporter l'image obtenue.

b) <u>influence du facteur d'amortissement</u>

Relancer le script ' *oscillations_pseudopériodique - fonction + graphe.sce*' en modifiant le titre : ajouter "influence de ksi".
Exécuter le script ' *oscillations_pseudopériodique - graphes suivants.sce*'.
Revenir à la valeur d'origine pour T. Modifier la valeur de ξ.
Etc. Exporter l'image obtenue.

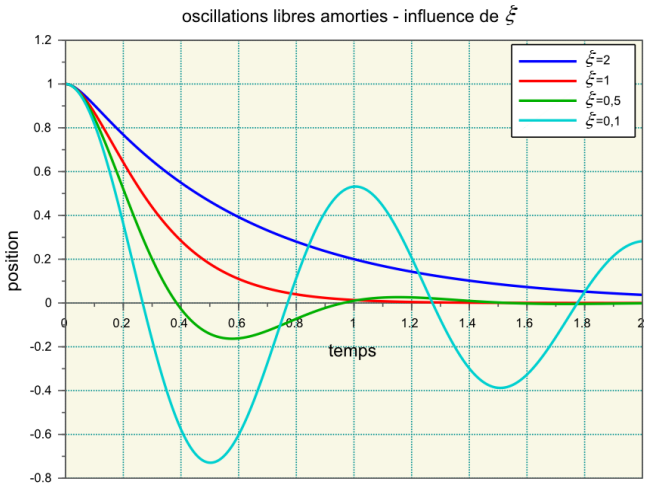
c) <u>influence des conditions initiales</u>

Même chose.

3.2. Régimes apériodique et critique

Les fonctions n'ont pas encore été définies. Mais c'est la même démarche.

Définir la fonction $x(t)$ pour le régime apériodique dans un script.
Ouvrir le script
'*oscillations_ pseudopériodique – fonction + graphe amélioré.sce*'.
Enregistrez-le sous
' *oscillations_ apériodique – fonction + graphe amélioré.sce* ', par exemple. Faites les modifications adéquates pour mettre en évidence l'influence de la valeur du facteur d'amortissement et des conditions initiales.



Mêmes opérations pour le régime critique.
NB : il peut être intéressant de superposer régimes apériodique, critique et pseudopériodique (voir ci-contre).

3.3. Oscillations forcées (plus tard...)

Reportez-vous aux fonctions définies dans le cours de mécanique sur les oscillations forcées (M5) pour tracer :
- Courbes de résonance $x_m = f(\omega)$ pour différentes valeurs de ξ.
- Déphasage $\varphi = f(\omega)$.