Rapport Interne
Programmation sous Matlab
2010

Description des classes Matlab.

Objet

Le rapport contient la description des classes développées sous Matlab.

Félicien Bonnefoy
Laboratoire de Mécanique des Fluides
Centrale Nantes

Table des matières

Principe

On décrit dans ce document les classes et les méthodes développées sous Matlab pour la suppression des ondes libres. Chaque classe comporte en général les cinq méthodes de base suivantes

display affiche les propriétés de l'objet,

get retourne la propriété souhaitée de l'objet,

set modifie les propriétés de l'objet. Elle nécessite en argument un liste de paires propriété–valeur.

convert2nondim passe les données en version adimensionnelle, à l'aide de la profondeur donnée en entrée et de l'accélération de la pesanteur $g = 9.81$ m.s^{-1},

convert2dim passe les données en version dimensionnelle, à l'aide de la profondeur qu'on donne en entrée et de l'accélération de la pesanteur $g = 9.81$ m.s^{-1},

On détaille dans la suite ces méthodes de base si elles comportent des spécificités et les autres méthodes le cas échéant.

J'essaie d'écrire les fichiers avec des nouvelles fonctionnalités de Matlab, notamment la définition des classes. Une classe est définie par ces propriétés, ces méthodes et ces évènements.

Une classe peut être définie par valeur (comme un un tableau classique) ou par handle (pointeur vers l'instance).

Chapitre 1

Théorie potentielle

On a défini plusieurs objets dans le but d'étudier la houle générée en bassin. On se base d'une part dans le cadre de la théorie potentielle. L'objet potentiel permet d'évaluer les champs de vitesse dans le bassin de houle.

D'autre part, on se place également dans le cadre des expériences réalisées à Centrale Nantes. Le premier objet décrit ci-après introduit par exemple la notion d'harmonique ; elle est liée au fonctionnement du programme **Ocean** qui permet de préparer les fichiers de houle générés ensuite dans le bassin.

1.1 Objet harmonic

1.1.1 Description

Un objet de la classe *harmonic* contient les informations élémentaires d'une onde (harmonique, amplitude et phase). La notion d'harmonique fait intervenir celle d'un fondamental, qui est lié au fonctionnement du batteur et défini plus tard dans l'objet *wave* (voir période de répétition). Plus de détails sont fournis dans la section 1.1.5.

1.1.2 Contenu

Un objet de cette classe contient les éléments suivants :

dim entier qui vaut un (par défaut) lorsque l'amplitude de la houle est donnée en mètre ou 0 si elle est donnée en adimensionnel,

n_harmo entier égal au nombre d'harmoniques décrivant le spectre,

harmo numéro d'harmonique, *i.e.* entier tel que la fréquence de la houle vaut *harmo* fois une fréquence fondamentale (la fréquence de répétition dans un objet *wave*),

ampli amplitude de la houle,

phase phase de la houle,

angle direction de propagation.

1.1.3 Construction

On construit un objet *harmonic* en donnant le numéro d'harmonique, l'amplitude (en m), la phase et l'angle de propagation (en degré). La phase et l'angle sont nuls par défaut (voir l'aide MatLab sur *harmonic* pour plus de détails).

1.1.4 Méthodes

On définit les méthodes suivantes :

set ne permet pas de modifier la valeur de *dim* (modifiable seulement grâce à *convert2nondim* et *convert2dim*),

plus ou +, opération addition de deux *harmonic*,

uminus ou −, opération inversion de signe d'un objet *harmonic* (ajout de π à la phase),

minus ou −, opération soustraction de deux *harmonic*,

times ou .∗, opération multiplication par un scalaire,

mtimes ou ∗, opération multiplication au sens des matrices, limitée ici à la multiplication par un scalaire (pour compatibilité),

rotate ajoute un angle donné à la direction de propagation de toutes les composantes de l'objet *harmonic*,

length donne le nombre d'harmoniques contenu dans le spectre (équivalent à *get* de la propriété *harmo*),

isempty teste si l'objet *harmonic* contient un harmonique ou non,

subsref définit la référence pour un spectre,

subsassgn définit l'assignement pour un spectre.

convert2nondim **et** ***convert2dim***

1.1.5 Principe

Un objet *harmonic* ne définit pas complètement une onde dans le bassin (il manque par exemple la fréquence fondamentale, la profondeur et le type de batteur, loi de contrôle utilisés (*cf.* les objets *wave*)). L'intérêt principale de la classe *harmonic* est la possibilité offerte de définir simplement un spectre. En effet, un spectre peut être représenté par un vecteur d'objets *harmonic* ; dans ce cas le nombre d'harmonique est plus grand que 1.

Cette classe est définie pour pouvoir reproduire les fonctionnalités du langage **Ocean**.

1.1.6 Exemples

- `harmonic()` crée un objet *harmonic*.
- `harmonic()` crée un objet *harmonic*.
- `harmonic()` crée un objet *harmonic*.
- `harmonic()` crée un objet *harmonic*.
- `harmonic(32,0.1)` crée une houle d'amplitude 10 cm.
- `harmonic([26,32],0.1)` crée une houle bi-chromatique dont les composantes ont toutes les deux une amplitude de 10 cm.
- `harmonic([26,32],[0.1,0.15],pi)` crée une houle bi-chromatique dont les composantes ont pour amplitude 10 et 15 cm et pour phase π.

1.1.7 à faire :

écrire la page d'aide en java

1.2 Objet wavemaker

1.2.1 Description

Un objet de la classe *wavemaker* contient les informations géométriques d'un générateur de houle.

1.2.2 Contenu

Un objet de cette classe contient les éléments du tableau 1.1.

Type d'entrée	Variable	Unité	Description
type			Type de batteur
type_ramp			Type de rampe temporelle
depth	h	m	Profondeur
hinge_bottom	d	m	Hauteur charnière du bas
middle_flap	D	m	Hauteur du panneau du milieu
ramp	T_r	s	Durée de la rampe de démarrage
n_paddles			Nombre de volets (si batteur segmenté)
Ly	L_y	m	Largeur du bassin

TABLE 1.1 – Description des éléments contenus dans un objet *wavemaker*

1.2.3 Construction

On construit un objet *wavemaker* en donnant la profondeur, le type de générateur, la distance entre le fond et la charnière du bas, la distance entre les deux charnières, la durée de la rampe et la largeur du bassin. On dispose d'un nombre de bassin prédéfinis, *ECN_wave*, *ECN_small*, *ECN_towing* et *B600* (voir l'aide MatLab sur *wavemaker* pour plus de détails).

1.2.4 Méthodes

On définit les méthodes suivantes :

set ne permet pas de modifier la valeur de **dim** (modifiable seulement grâce à *convert2nondim* et *convert2dim*),

eq teste l'égalité entre deux objets *wavemaker*,

ne teste la différence entre deux objets *wavemaker* (utile lorsqu'on ajoute deux objets *wave*),

export exporte l'objet *wavemaker* dans un fichier `.cfg`.

convert2nondim* et *convert2dim

1.2.5 à ajouter

origine du repère (*cf.* bassin de houle = au milieu dans **Ocean**, théorie = au bord car plus pratique)

1.2.6 Exemples

- `wavemaker()` crée un objet *wavemaker*.
- `wavemaker('ECN_wave')` pour le bassin de houle
- `wavemaker('ECN_small')` pour le petit bassin de houle
- `wavemaker('ECN_towing')` pour le bassin de traction
- `wavemaker('B600')` pour le B600 à DGAth

1.3 Objet wave

1.3.1 Description

Cet objet décrit complètement une onde élémentaire, aussi bien droite qu'oblique. Il permet d'évaluer l'élévation de surface libre théorique ainsi que le mouvement batteur correspondant. Plus de détails sont fournis dans la section 1.3.6

1.3.2 Contenu

Il est composé de la période de répétition T_{repeat}, de la fréquence[1] d'échantillonnage f_{samp}, d'un objet de classe *harmonic*, d'un objet de classe *wavemaker*, et d'un objet de classe *control_law*.

1.3.3 Construction

On construit un objet *wave* en donnant la période de répétition, la fréquence d'échantillonnage, un objet *harmonic*, un objet *wavemaker* et un objet *control_law*. Ce dernier est défini par défaut comme le principe du serpent (voir l'aide MatLab sur *wave* pour plus de détails).

1.3.4 Héritage

Cette classe hérite de la classe *wavemaker*. Un spectre peut être représenté par un objet *wave* contenant un vecteur d'objet *harmonic*,

1.3.5 Méthodes

On définit les méthodes suivantes :

display	affiche les propriétés d'un objet *wave*,
get	retourne les propriétés d'un objet *wave*, notamment celles des objets *harmonic* et *wavemaker* qui le composent,
set	modifie les propriétés d'un objet *wave*, notamment celles des objets *harmonic* et *wavemaker* qui le composent,
plus	ou +, opération addition de deux *wave*,
times	ou $*$, opération multiplication par un scalaire,
rotate	ajoute un angle donné à toutes les composantes de l'objet *wave* (*cf.* objet *harmonic*),
eval_data	donne accès aux informations spatio-temporelles de l'onde (période de répétition, période, fréquence, pulsation, nombre d'onde, longueur d'onde),
clean	regroupe les ondes qui ont la même fréquence en une seule[2] et enlève les composantes d'amplitude nulle,

1. ou de manière équivalente de f_{samp} et d'un entier n_{repeat} tel que

$$n_{repeat} = T_{repeat}\, f_{samp},$$

2. Attention, en 3D ce ne sera plus valable.

time_shift* et *position_shift
décalent les composantes respectivement en temps et en espace,

phase_shift décale les phases des composantes,

eval_eta évalue l'élévation de surface libre en une position x donnée en fonction d'un vecteur temps donné en entrée,

write_inp écrit l'élévation de surface libre en $x = 0$ correspondante au spectre *wave* donné en entrée. Cette routine est utile pour le pilotage du B600. On peut éventuellement modifier la fréquence d'acquisition en la rentrant en paramètre. Pour le B600, il faudrait inclure la conversion en fichier batteur $*$.des grâce au fichier `WhtextConvert.m.m`,

calc_TF_TF_n
permet de calculer les fonctions de transfert (appel interne au calcul des intégrales I et J_n, des coefficients p_n et q_n si nécessaire)

eval_X_3D permet de tracer le mouvement batteur. On donne en entrée le vecteur temps et optionnellement la position des volets.

eval_bound_waves

eval_free_waves

eval_X
convert2nondim* et *convert2dim

1.3.6 Principe

La fréquence d'échantillonnage fournie lors de la construction de l'objet sert à construire le vecteur temps dans les routines qui en ont besoin.

1.3.7 Précaution

Par défaut, on attribue une loi de commande serpent (voir les objets *control_law*) si aucune loi n'est spécifiée en argument. Ce sera le cas par exemple en cas de houle droite, ce qui est sans effet. Dans le cas d'une houle oblique, il ne faudra pas oublié de spécifier la loi de commande en argument si on désire une loi élaborée autre que le principe du serpent.

Côté pratique, on crée l'objet en donnant la période de répétition T_{repeat} plutôt que n_{repeat}. On peut faire le lien entre T_{repeat} et la valeur $rnum$ entrée lors de la création des houles avec **Ocean**. On a en effet

$$T_{repeat} = \frac{2^{\mathrm{rnum}}}{\mathrm{clock}}$$

où *clock* vaut 32 Hz. Ainsi, une valeur $rnum = 11$ donne $T_{repeat} = 64$ s. Une onde créée dans **Ocean** sera définie avec $f_{samp} = 32$ Hz. Le tableau 1.2 donne les périodes de répétition associées aux valeurs usuelles de $rnum$.

Les fonctions *front* et *single* seraient à définir en dehors des méthodes pour ne pas nécessiter d'argument *wave* en entrée. Les fonctions *gensea2wave* et *ocean_txt2wave* assurent la lecture des fichiers `.inp` et `.txt` et la conversion en un objet *wave*.

$rnum$	10	11	12	13	14	15
T_{repeat} (en s)	32	64	128	256	512	1024

TABLE 1.2 – Périodes de répétition T_{repeat}

1.3.8 Exemples

•

1.4 Objet control_law

Cet objet est nécessaire pour la génération de houle oblique. En houle droite, on peut s'en passer (*cf.* l'objet *control_law* de type serpent par défaut dans la classe *wave*).

1.4.1 Description

Un objet de la classe *control_law* regroupe les éléments décrivant la loi de contrôle utilisée pour la génération de houle directionnelle (oblique).

1.4.2 Contenu

Un objet de cette classe contient les éléments suivants :

active_paddles le numéro des volets actifs (cela fait référence au nombre n_paddles de volets qui est défini dans un objet *wavemaker*),

law loi de contrôle (parmi snake, dalrymple, disc et rectangle),

parameters paramètres correspondant à la loi de contrôle choisie (respectivement $\emptyset$, X_d, (x_0, y_0, R) et (x_0, y_0, x_1, y_1)).

1.4.3 Construction

On construit un objet *control_law* en donnant le numéro des volets actifs (0 volet continu), le type de loi de commande et ses paramètres associés, et la dimension (dim=1 : dimensionnel (par défaut), 0 adimensionnel) (voir l'aide MatLab sur *control_law* pour plus de détails).

1.4.4 Exemples

-

1.5 Objet potential_2D

1.5.1 Description

Un objet de la classe *potential_2D* regroupe les éléments nécessaires à un calcul de suppression d'onde libre en 2D.

1.5.2 Contenu

Un objet de cette classe contient les éléments suivants :

wave un objet *wave*,

TF_type le type de mouvement batteur pour la fonction de transfert,

alpha_n les nombres d'onde verticaux $\alpha_n \in$,

TF la fonction de transfert $\underline{TF}$,

TF_n les fonctions de transfert des modes évanescents $\underline{TF_n}$,

sigma_n les nombres d'onde verticaux pour les modes libres indépendant du temps $\sigma_n \in$.

1.5.3 Construction

On construit un objet *potential_2D* en donnant un objet *wave*, les nombres de modes évanescents en linéaire et pour les ondes libres (par défaut 100 et 0 respectivement, voir l'aide MatLab sur *potential_2D* pour plus de détails).

1.5.4 Méthodes

On définit les méthodes suivantes :

init_data évalue les éléments alpha_n, TF, TF_n, et sigma_n,

calc_eta_lin évalue l'amplitude de la houle linéaire (en ω)

calc_ampli_lin
évalue l'élévation de surface libre linéaire (en ω)

calc_UW_lin évalue les composantes horizontale et verticale de vitesse linéaire (en ω), en séparant modes progressif et évanescents, au cours du temps, sur un maillage (x, z) donné.

calc_UW_bound_indep
évalue les composantes horizontale et verticale de vitesse au second ordre (indépendante du temps), associées aux ondes liées

calc_U_free_indep
évalue les composantes horizontale et verticale de vitesse au second ordre (indépendante du temps), associées aux ondes libres, en séparant le courant de retour et les modes évanescents.

calc_U_Stokes
évalue le courant de Stokes à la surface du bassin.

calc_return_current
évalue le courant de retour.

calc_eta_bound_indep
: évalue la variation de niveau moyen à la surface du bassin.

calc_eta_bound_2w
: évalue l'élévation de surface libre liée (en 2ω)

calc_eta_free : évalue l'élévation de surface libre libre (en 2ω)

calc_UW_bound_2w
: évalue les composantes horizontale et verticale de vitesse au second ordre (en 2ω), associées aux ondes liées

calc_UW_free_2w
: évalue les composantes horizontale et verticale de vitesse au second ordre (en 2ω), associées aux ondes libres

***convert2nondim* et *convert2dim* ?**

Les intégrales I et J_n, les coefficients p_n et q_n (les J_n, p_n et q_n sont séparés en une partie monotone et une partie oscillante), l'ordre et les coefficients de leur développement asymptotique. On construit ces différentes composantes à la création de l'objet.

1.6 Objet potential_3D

1.6.1 Description

Un objet de la classe *potential_3D* regroupe les éléments nécessaires à un calcul de suppression d'onde libre en 3D.

1.6.2 Contenu

Un objet de cette classe contient les éléments suivants :

potentiel_2D un objet *potentiel_2D*,

N_1 le nombre de modes transverses progressifs,

mu_n les nombres d'onde transverses,

I_n les intégrales transverses,

k_0n les nombres d'onde des modes progressifs,

a_0n les amplitudes des modes progressifs,

k_mn les nombres d'onde,

TF_mn les fonctions de transferts des modes évanescents,

1.6.3 Construction

On construit un objet *potential_3D* à partir d'un objet *wave*, et des nombres de modes évanescents verticaux et transverses (par défaut 100, voir l'aide MatLab sur *potential_3D* pour plus de détails).

1.6.4 Méthodes

On définit les méthodes suivantes :

init_data évalue les éléments N_1, μ_n, k_{mn}, k_{0n}, I_n, a_{0n}, TF_{mn} en linéaire,

calc_eta_lin calcule l'élévation de surface libre sur la surface du bassin au cours du temps. Il faut noter qu'on doit utiliser $N+1$ points dans la direction transverse si l'on veut bénéficier d'une accélération des calculs par FFT, où N est le nombre de modes transverses

calc_U_lin calcule la composante horizontale de vitesse linéaire.

***convert2nondim* et *convert2dim* ?**

1.6.5 à faire

évaluer les champs de vitesse dans des plans xz et xy. Tout le second ordre voir les fichiers dans le répertoire from_these

1.7 Exemples

1.7.1 En deux dimensions

- Choix du générateur de houle,
```
wmk = wavemaker('ECN_wave');
```

- Choix des harmoniques
```
harmo = harmonic(32, 0.1);
```

- Construction de l'objet *wave*
```
wv = wave(64, 40, wv, wmk);
```

- Prédiction de l'onde libre avec en argument le nombre de modes évanescents utilisés pour le calcul et suppression par déphasage de π
```
wv_free = eval_free_waves(wv, 100);
wv_free = phase_shift(wv_free,pi);
```

- Sauvegarde des données sous forme d'un fichier .dat pour le batteur
```
export(wv+wv_free,'dat', 'filename')
```

- Construction du *potentiel_2D*
```
pot = potential_2D(wv, 100);
```

- Evaluation des vitesses en linéaire
```
[U_prog_1st U_evan_1st W_prog_1st W_evan_1st]= calc_UW_lin(pot,x,z,time);
```

- Evaluation des vitesses libres au second ordre
```
[U, U_evan, W_evan] = calc_UW_free_indep(pot, x, z, n_evan_free);
```

- Evaluation du courant de Stokes
```
U_Stokes = calc_U_Stokes(pot, x);
```

1.7.2 Pour le B600

On doit en plus spécifier la fonction de transfert utilisée avec le batteur biflap :

- Choix du générateur de houle,
```
wmk = wavemaker('B600');
```

- Choix des harmoniques
```
harmo = harmonic(32, 0.1);
```

- Construction de l'objet *wave*
```
wv = wave(64, 40, wv, wmk);
```

- Choix de la fonction de transfert

```
wv = set(wv, 'TF_type', 1);
```

Chapitre 2

Classes Tecplot

On définit ici deux classes d'objet utiles pour faire des sorties fichiers dans un format compatible Tecplot. On peut alors bénéficier des outils de visualisation de Tecplot. On s'est tourné pour l'instant vers une écriture zone par zone.

Principe

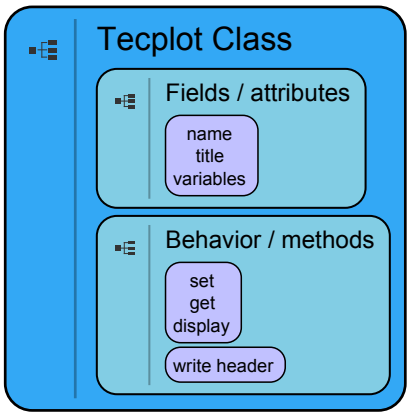
On a choisi de définir une classe *tecplot* qui contient les informations relatives à un fichier de sortie au format Tecplot® et une classe *zone* qui contient les informations relatives à une zone Tecplot®. Lorsqu'on veut sauvegarder des données en format Tecplot®, on utilise donc un objet *tecplot* pour le fichier de sauvegarde (ouverture du fichier, écriture d'un entête) puis un objet zone par pas de temps.

Exemples

- on crée un objet *tecplot* avec pour fichier `file.dat`, pour titre `title` et trois variables `x`, `y` et `z` : `tec = tecplot('file.dat', 'title', char('x','y','z'));`,
- on ouvre le fichier et on écrit l'entête : `fid = write_header(tec);`,
- Pour chaque zone, on crée un objet *zone* : `zon = zone(0,3,[Nx, Ny],[],[],data);`,
- On écrit la zone dans le fichier : `write_zone(zon, fid);`,
- Une fois toutes les zones écrites, on ferme le fichier : `fclose(fid);`.

2.1 Classe tecplot

2.1.1 Description



Un objet de la classe *tecplot* contient les informations relatives à un fichier de sortie au format Tecplot®.

2.1.2 Contenu

Un objet de cette classe contient trois éléments :

name le nom du fichier,

title le titre, qui s'affichera dans l'entête de la frame,

variables le nom des variables, rentré sous forme de cell.

2.1.3 Méthodes

On définit les méthodes suivantes :

write_header ouvre le fichier, écrit l'entête du fichier (le titre et le nom des variables) et retourne le numéro (fid) du fichier.

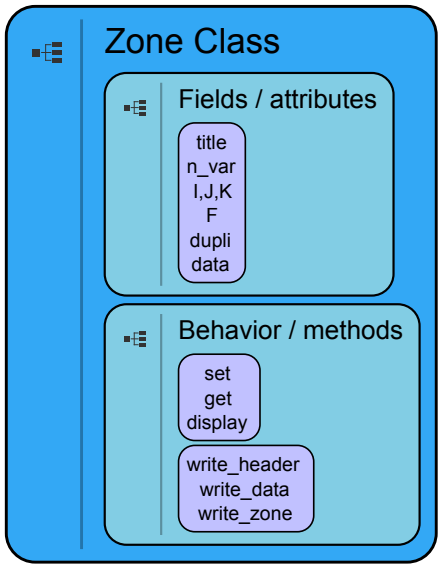
2.1.4 Exemples

- `tec = tecplot('file.dat', 'title', char('x','y','z'));` crée un objet *tecplot* avec pour fichier `file.dat`, pour titre `title` et trois variables `x`, `y` et `z`.
- `fid = write_header(tec);` permet d'ouvrir le fichier et d'écrire l'entête. La variable `fid` est réutilisée ensuite.

2.2 Classe zone

2.2.1 Description

Un objet de la classe *zone* contient les informations relatives à une zone d'un fichier de sortie au format Tecplot®.



2.2.2 Contenu

Un objet de cette classe contient les éléments suivants :

title le titre de la zone (son nom), ou le temps en `SOLUTIONTIME`

n_var le nombre de variables,

I,J,K les entiers pour les fichiers I-, J- ou IJK-ordered,

F le format d'écriture (POINT ou BLOCK),

dupli le numéro des variables éventuellement dupliquées,

data les données de la zone.

2.2.3 Méthodes

On définit les méthodes suivantes :

write_header écrit l'entête de la zone (le titre, les I,J et K, le format et la duplication). Si le titre est une chaîne de caractère, il s'agit d'un titre passé en `T=...`, si le titre est une valeur numérique, il est passé en temps dans `SOLUTIONTIME=...`,

write_data écrit les données,

write_zone écrit entête et données.

2.2.4 Principe

On peut initialiser la zone sans toutes les composantes. Si I, J, K sont spécifiés, les données sont initialisées à un tableau nul. La taille des données est vérifiée si on spécifie data.

On pourrait hériter d'un objet *tecplot* dans lequel on aurait stocker la fid du fichier. Cela permettrait lorsqu'on écrit la zone, de vérifier si le fichier est ouvert.

2.2.5 Exemples

- `zon = zone(0,3,[Nx, Ny],[],[],data);` crée un objet *zone*
- `write_zone(zon, fid);` écrit la zone
- `fclose(fid);` ferme le fichier.

Annexe A

Autres objets

A.1 Objet info

Cette classe sert à calculer simplement les caractéristiques d'une onde, telles que la fréquence, période, longueur d'onde...

A.1.1 Description

Un objet de la classe *info* contient les informations spatio-temporelles d'une onde (fréquence et longueur d'onde).

A.1.2 Contenu

Un objet de cette classe contient les éléments du tableau A.1.

Variable	Description
input	Type d'entrée
dim	1 si dimensionnel, 0 sinon
depth	Profondeur

Type d'entrée	Variable	Unité	Description
pulsation	omega	$Rad.s^{-1}$	Pulsation
period	period	s	Période
frequency	freq	Hz	Fréquence
wavenumber	k	m^{-1}	Nombre d'onde
wavelength	lambda	m	Longueur d'onde

TABLE A.1 – Description des éléments contenus dans un objet *info*

A.1.3 Méthodes

On définit les méthodes *display* et *get*.

A.1.4 Méthodes

Un objet *info* sert principalement à jongler avec les informations spatio-temporelles de l'onde. Il ne contient pas d'information de phase, d'amplitude ou de direction (*cf.* les objets *harmonic* et *wave*). Dans le même registre, on ne peut effectuer d'addition, de multiplication par un scalaire *etc.*