TP JS 3 - Courbes

Décompressez l'archive déposée sur Moodle pour ce TP. Le dossier résultant contient différents fichiers à réutiliser ou à compléter. Pensez à consulter le site MDN.

On propose une page web permettant de représenter graphiquement différentes fonctions mathématiques (voir Figure 1). L'implémentation `JS` s'appuie sur les canevas `HTML` et l'API `Canvas` en se limitant aux fonctionnalités permettant le tracé de lignes.
La page permet au visiteur de

- sélectionner et paramétrer des fonctions,
- les échantillonner sur un intervalle de valeurs,
- dessiner chaque courbe en reliant les ordonnées des points d'abscisses successives de l'échantillon,
- animer chaque courbe segment par segment selon une fréquence choisie.

La trace de l'exécution `JS` est quant à elle consignée dans les différents onglets de la console : objets utilisés pour le tracé et l'historisation (`Journaux`), points hors-limites (`Avertissements`), levées d'exceptions (`Erreurs`), échantillons générés et autres variables (`Débogueur`).

La page comprend :

- un formulaire scindé en trois panneaux,
- un canevas sur lequel sont dessinés les courbes,
- un tableau faisant office d'historique (log) et listant toutes les fonctions dessinées.

Le panneau "Repère" sert à ajuster les dimensions du repère (coordonnées maximum) dans lequel s'inscriront les courbes et, optionnellement, à surimposer une grille. Le panneau "Echantillonnage" sert à ajuster la taille des échantillons (nombre de points) à générer pour chaque fonction ainsi que la fréquence du traçage. Le panneau "Fonctions" sert à paramétrer les fonctions et à déclencher leur tracé en cochant la case correspondante.

Chaque courbe est dessinée avec une couleur tirée aléatoirement et l'expression de la fonction est ajoutée au tableau. L'unité de fréquence est le dixième de secondes. La fréquence nulle (valeur par défaut) signifie que les courbes seront tracées l'une après l'autre sans délai d'affichage entre segments successifs. Une fréquence f non-nulle signifie que les courbes seront affichées de manière concurrente (en "parallèle") avec un délai de f dixièmes de secondes entre segments successifs (voir l'instantané en Figure 2).
Ce démonstrateur vous aidera à visualiser ce qui est attendu.

Le dossier décompressé contient les fichiers suivants :

- **courbes.html** : le fichier `HTML` de la page web.
- **courbes.css** : la feuille de styles.
- **courbes.js** : le fichier principal `JS`.
- **échantillonneur.js** : le module d'échantillonnage.
- **traceur.js** : le module de dessin.
- **échantillonneur_proto.js** : fichier obfusqué pour l'échantillonnage.
- **traceur_proto.js** : fichier obfusqué pour le dessin.

Le fichier principal **courbes.js** implémente les fonctionnalités suivantes :

1. tracé de la fonction identité par l'IIFE `test`,
2. appel de la fonction `tracer` qui trace une courbe au cochage d'une case,
3. appel de la fonction `regénérerTracés` qui regénère canevas et courbes à chaque clic sur `REGENERER`.

L'IIFE `test` fournit la recette pour tracer et journaliser une fonction (ici, la fonction $f(x) = x$). Ces opérations sont déléguées à un objet de la classe `Traceur` par appel à la méthode `Traceur.dessiner`. Le traceur est construit au préalable à partir du canevas `HTML`, d'un contexte 2D, des marges internes délimitant le dessin du repère et des courbes (20 pixels en horizontal/vertical), et des valeurs maximum autorisées en abscisse et ordonnées pour les fonctions. La méthode `Traceur.dessiner` prend en arguments un descripteur de la fonction contenant notamment son implémentation `JS` sous forme de callback. Elle délègue l'échantillonnage à un objet de classe `Echantillon` par appel à la méthode `Echantillon.points` et le tracé à la méthode `Traceur.tracer`.

Le code qui vous est fourni est commenté et fonctionnel grâce aux modules obfusqués. Vous devrez écraser différentes méthodes et fonctions qui font appel à ces modules pour répondre aux questions.

Exercice 1. Echantillonnage

Un objet `Echantillon` contient

- une fonction à échantillonner sous forme de callback `f`,
- une taille `n` d'échantillon,
- l'intervalle de valeurs `minMaxX` sur lequel est échantillonnée la fonction,
- la méthode statique `abscisse(k,n,minMaxX)` calculant la `k`-ième valeur dans l'intervalle `minMaxX` gradué par `n` valeurs,
- l'échantillon résultant `points` sous la forme `[[x,f(x)] | i=1..n, x=abscisse(i,n,minMaxX)]`.

1. Réimplementez la propriété `Echantillon.points`.

Exercice 2. Tracé et journalisation

Le tracé d'une fonction f s'effectue en convertissant chaque point `(x,f(x))` d'un échantillon par une paire de coordonnées (appelée "transformé" du point) qui sont exprimées en pixels relativement au coin supérieur gauche du canevas (voir l'API `Canvas`).

Les objets de la classe `Traceur` implémentent des conversions obéissant aux règles suivantes :

- les points autorisés appartiennent au domaine `D=[-maxXY.X,+maxXY.X]`$\times$`[-maxXY.Y,+maxXY.Y]` où `maxXY` est donné,
- les transformés autorisés appartiennent au rectangle `R` délimité par les marges internes du canevas,
- tout point en dehors de `D` est omis du tracé,
- tout transformé en dehors de `R` est omis du tracé,
- toute omission donne lieu à un avertissement en console,
- le transformé du point `O=(0,0)` apparaît au centre de `R` et correspond à l'origine du repère,
- le transformé d'un point `(x,y)` est obtenu en redimensionnant `x` et `y` du fait du changement d'échelle entre `D` et `R` puis par décalage relativement au transformé du point `O`.

Un objet `Traceur` encapsule

- un canevas `canevas` sur lequel dessiner,
- sa largeur `L` définie par l'attribut `HTML width`, sa hauteur `L` définie par l'attribut `HTML height`, et ses marges internes `marge` qui déterminent conjointement les dimensions du rectangle `R`,
- son contexte 2D `contexte`,
- les valeurs maximum `maxXY` déterminant le domaine `D`,
- les coefficients de redimensionnement `dimensionXY` basés sur les ratios entre `R` et `D`,
- les transformés des points extrêmes des axes de `D` et de son centre (`repère`).

1. Réimplementez la méthode `Traceur.point(x,y)` qui calcule et renvoie le transformé `(u,v)` d'un point `(x,y)` au format `{"X": u, "Y": v}`. La méthode renvoie `{"X": false, "Y": false}` pour tout point ou tout transformé hors-limites. Par exemple, un point est hors-limite si la fonction est évaluée en dehors de son domaine de valeurs (valeur `NaN`) ou si l'évaluation donne lieu à un débordement (valeurs `+/-Infinity`).

2. Réimplementez la méthode `Traceur.tracerGrille` qui dessine la grille correspondant aux graduations des axes du repère. Inspirez-vous de la méthode `Traceur.tracerRepère` qui dessine le repère. Les axes de la grille sont tracés en pointillés et en noir (voir Figure 2).

3. Réimplementez la méthode `tracer(P,style)` qui dessine la courbe de l'échantillon de points `P` en utilisant la couleur `CSS style`. La méthode omet le tracé des segments reliant tout point de valeur `{"X": false, "Y": false}` et émet un avertissement le cas échéant. La méthode lève une exception si les abscisses des points consécutifs du tableau `P` ne sont pas ordonnées, cad. ne vérifient pas `P[i].X < P[i+1].X (1<=i<n)`.

4. Réimplementez la méthode `Traceur.dessiner(n, meta, log)` qui (1) génère un échantillon de taille `n` pour la fonction décrite par `meta`, (2) trace sa courbe, et (3) journalise son expression dans le tableau `log` et l'affiche dans le tableau `HTML`.

L'échantillonnage s'appuie sur la classe `Echantillon`. Le tracé utilise la méthode `tracer(P,style)` en considérant deux cas de figure :

1. A fréquence nulle, le tracé s'effectue de manière synchrone.
2. A fréquence non nulle, le tracé s'effectue segment par segment par appel périodique à `tracer(P,style)`. Pour ce faire, utilisez setInterval en veillant à lier "l'objet `this`" à ces appels (voir Le problème de this) !

La journalisation et l'affichage dans le tableau `HTML` suppose de construire l'expression de la fonction à partir de son nom et des paramètres stockés dans le descripteur `meta` : privilégiez les littéraux de gabarits (*template literals*).

Exercice 3. Ecouteurs

1. Réimplementez la fonction `tracer(traceur,log)` qui trace la courbe d'une fonction f lorsque la case correspondante est cochée. La fonction procède en 3 étapes :

1. Elle construit le descripteur de f à partir des paramètres renseignés dans le formulaire. Consultez le descripteur `meta_f` initialisé dans l'IIFE `test` pour le format attendu. La fonction f sera implémentée à l'aide des méthodes et constantes de l'objet `Math`. La couleur `CSS` sera aléatoirement générée sous forme de chaîne de caractères du type `"rgb(12,255,0)"` par appel à la fonction `rgb` qui est à implémenter.
2. Elle invoque la méthode `traceur.dessiner(n, meta_f, log)` pour le tracé.
3. Elle laisse la case cochée pendant 2 secondes avant de la décocher.

2. Réimplementez la fonction `regénérerTracés(log)` qui, à chaque clic sur `REGENERER`, regénère le canevas et les courbes tracées. La fonction procède en 4 étapes :

1. Suppression et remplacement du canevas existant par un canevas de même taille.
2. Construction d'une instance de `Traceur` pour ce canevas selon les coordonnées maximum renseignées dans le formulaire et avec des marges de 20 pixels.
3. Traçage du repère, et éventuellement de la grille si le bouton radio a été coché, à l'aide de l'objet traceur.
4. Traçage de toutes les courbes des fonctions stockées dans l'historique.

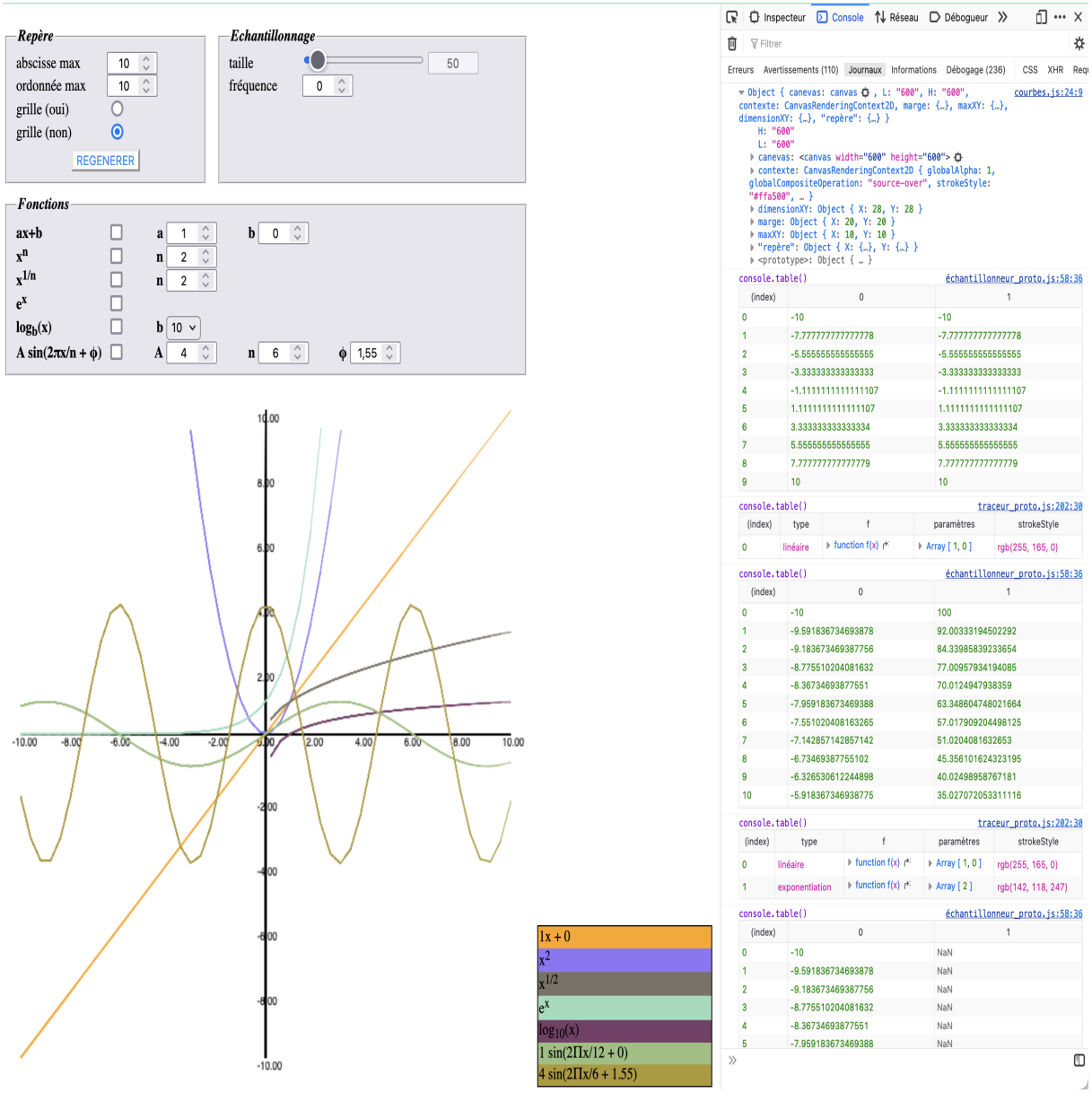

FIGURE 1 – Tracé de courbes

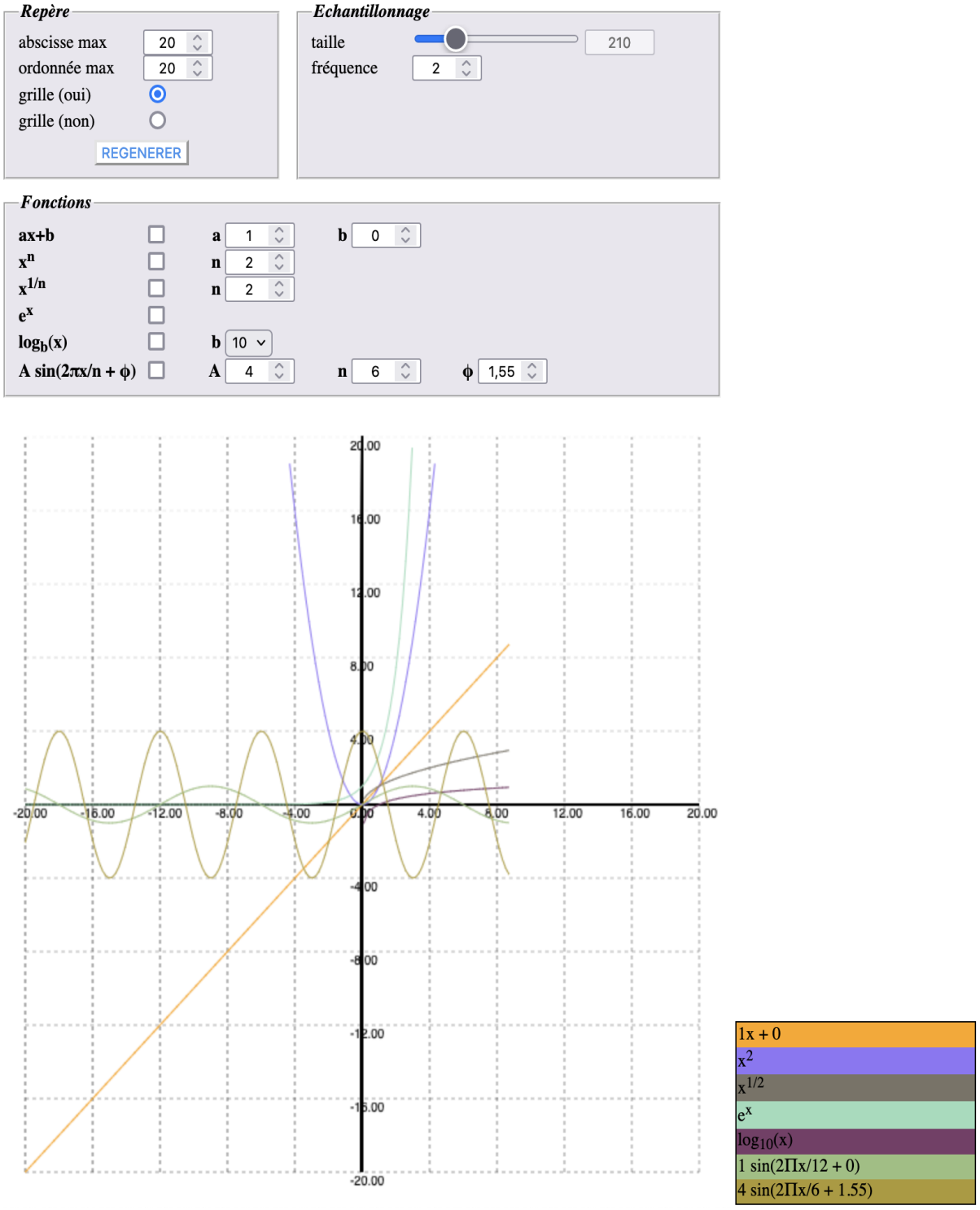

FIGURE 2 – Instantané d'un tracé dynamique de courbes