Inria Bordeaux, EQUIPE STORM



Rapport de Stage

# AFF3CT web site

Mehdi Naciri
Stage 1ere année 2018/2019
ENSEIRB-Matmeca


Superviseur : Adrien Cassagne
Maître de stage : Denis Barthou

4 juillet 2019

# Table des matières

# Introduction

De tout temps, l'être humain a cherché à communiquer. Ses méthodes ont évoluées, se sont modernisées pour ainsi devenir celles que nous connaissons aujourd'hui (WIFI, 3G 4G 5G, etc.).

Pour l'élaboration d'un système de communication, il est important, parmi de nombreux autres paramètres, de bien choisir le type d'encodage. Pour cela, il est nécessaire d'avoir un moyen de comparer ces types de codage, en fonction des besoins du système (vitesse de transmission, débit, taille de trame, algorithme de code correcteur d'erreur, etc.). Il est également important de comparer la justesse du signal reçu par rapport au signal émis.
En effet, pour une certaine perturbation de signal donné, on veut comparer les méthodes de transmission pour choisir la plus juste.

Le logiciel AFF3CT a été implémenté dans ce but. Il s'agit d'un logiciel libre dédié à la conception, l'expérimentation et la validation de codes correcteurs d'erreurs. Il est accompagné d'une large bibliothèque de codes de la littérature, ainsi que d'un ensemble de résultats de simulation pré-calculés.

Il est possible d'explorer les résultats de la base de données disponible mais également d'effectuer ses propres simulations. Jusqu'à aujourd'hui, pour afficher la courbe issue du résultat de la simulation AFF3CT, le logiciel PyBer est utilisé.

## 1. Besoins

Dans le but de promouvoir cet outil en accès libre et de pouvoir comparer les différentes simulations effectuées, un site Internet préliminaire a été mis en place, avec un certain nombre de limitations. Il permet d'affiner sa sélection de codes correcteurs d'erreurs.

Cet outil doit être accessible facilement, la sélection de simulation doit être aisée et une capacité de comparaison assez élevée.
Son utilisation doit être simple, efficace et les informations doivent être facilement lisibles.
L'utilisateur doit également pouvoir charger sa propre simulation, en vue d'un potentiel *"remplacement"* du logiciel Pyber.
Il doit être aisé de partager sa propre comparaison de simulation via la base de données.

De plus, un certain niveau de sécurité doit être mis en place sur le dit site web.

## 2. Solution

Pour répondre à ce besoin, un site web est mis en place. Dans un soucis de transparence, de traçabilité, de liberté d'accès et de prix, La plateforme Github est utilisée pendant le développement du site web mais également pour la publication de celui-ci.
https ://aff3ct.github.io

### Mise en forme

Pour la mise en forme, le langage HTML est utilisé par la quasi-totalité des sites web. Etant donné que l'on affiche toutes les références de la base de données, il peut être intéressant d'éviter la redondance de code et d'anticiper une future extensibilité.

### Design

Le design est géré par différents framework ou bien directement codé par nos soins, le tout en CSS.

### Dynamique

Dans un soucis de sécurité, le langage JavaScript est utilisé pour gérer la dynamique du site web car son traitement s'effectue côté client, donc sur la machine de l'utilisateur. Ainsi, le chargement de fichier locaux depuis la machine cliente vers notre site web ne pose aucun problème de sécurité.

**Frameworks**

Pour une question de compatibilité entre navigateurs, le framework Jquery est utilisé pour l'intéractivité client. Le framework Bootstrap apporte plusieurs formes et design notamment pour les boutons, la mise en forme de la page (organisation avec des colonnes), la barre de chargement, etc.

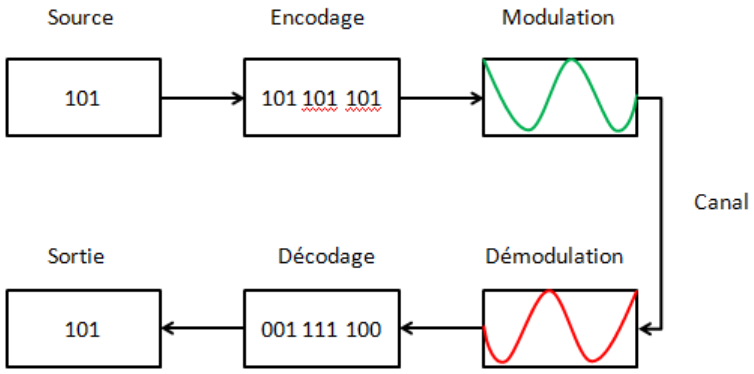
# Contexte

## 1. Notion de code correcteur d'erreur

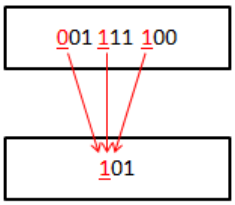
Lorsque deux personnes parlent ensemble, il se peut que la phrase que l'interlocuteur A souhaite prononcer ne soit pas exactement la même que la phrase entendue par l'interlocuteur B.
La phrase n'a probablement pas été correctement prononcée car l'interlocuteur A n'a pas parlé assez fort, a bégayé ou bien qu'il y a assez de bruit entre les deux pour perturber la communication. Cependant, il n'est pas forcement nécessaire de connaitre tous les mots de la phrase pour la reformer afin d'en comprendre le sens. Le principe est le même pour la transmission d'information.

Pour le transfert d'information, c'est lors de la reconstitution du message original qu'intervient les codes correcteurs d'erreurs. Ils vont lire le message reçu, vérifier si le contenu du message envoyé n'a pas été perturbé sur le chemin et le corriger si besoin.

Parmi les différents types de codes correcteurs d'erreurs, nous utiliserons le code de répétition comme exemple. Supposons que nous souhaitons transmettre ce message : 101. Ce message va être encodé et modulé pour pouvoir être transmis via un canal. Une fois réceptionné, ce message sera démodulé puis décodé afin d'extraire le message.

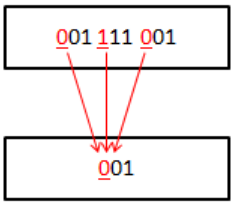



L'encodage par répétition consiste à envoyer plusieurs fois à la suite le même message. Dans notre exemple, le message est envoyé en triple exemplaire. Durant la transmission dans le canal, de nombreux éléments peuvent perturber le signal et ainsi modifier le message reçu. Cependant, le code à répétition permet de comparer bit à bit les différents exemplaires du message source et ainsi reformer le message.




Dans notre exemple, on peut voir que le premier mot a vu son premier bit corrompu. En revanche, le premier bit des deux autres mots n'ont pas changé. Le premier bit du message reconstitué sera donc fidèle au message source. Le système réitière sur tous les bits composants le mot.

Le système a ses limites car si deux mots sur 3 ont le même bit corrompu, le message reçu sera corrompu.




Pour remédier à ce problème on peut augmenter le nombre de répétition. Cette opération a un prix car on augmente la quantité d'information transmise. C'est pourquoi il est important de comparer les différents codes correcteurs d'erreurs afin de trouver celui correspondant aux besoins de son système. Un bon compromis entre coût de transfert et fiabilité est de rigueur.

## 2. Les types de code

Pour présenter brièvement une partie des différents types de codes correcteurs d'erreurs :

-LDPC : Ce type de code est utilisé pour les transmissions bruitées. Il a été le code correcteur d'erreurs retenu pour le standard DVB-S2 utilisé pour les transmission par satellite de la télévision numérique de par sa faible complexité de décodage. Il est également disponible pour la transmission par WIFI. WiMAX embarque aussi le code correcteur d'erreur LDPC. Il s'agit d'un standard de communication sans fil par micro-ondes (Canal Hertzien). WiMAX est utilisé notamment pour fournir un accès Internet à haut débit sur une zone géographique étendue.

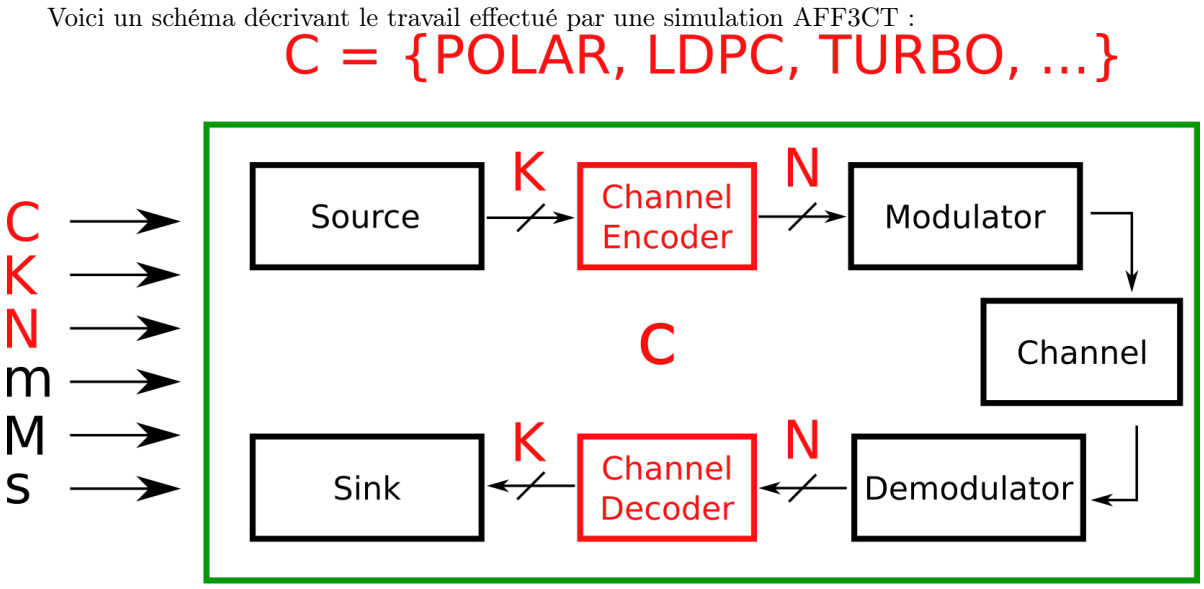
-Turbo Codes : Les Turbo Codes sont utilisés dans les standards de téléphonie mobile 3G et 4G notamment grâce à leurs excellentes performances de décodage et à la complexité calculatoire modérée des turbo décodeurs. La NASA et l'Agence Spatiale Européenne les utilisent pour leurs sondes spatiales. Ils sont également utilisés par l'ADSL 2.

-Codes Polaires : Les codes polaires sont/seront employés pour la transmission de données mobiles 5G.

## 3. Description d'AFF3CT

Pour effectuer une simulation, il suffit de taper une commande AFF3CT, par exemple : "aff3ct -C "POLAR" -K 1723 -N 2048 -m 1.0 -M 4.0 -s 1.0". Le paramètre C est le type de code correcteur choisi pour l'encodage, N étant la taille de la frame en bits et K le nombre de bits alloués à l'information. Les autres paramètres étant alloués à l'intervalle de définition de la simulation.

Voici un schéma décrivant le travail effectué par une simulation AFF3CT :

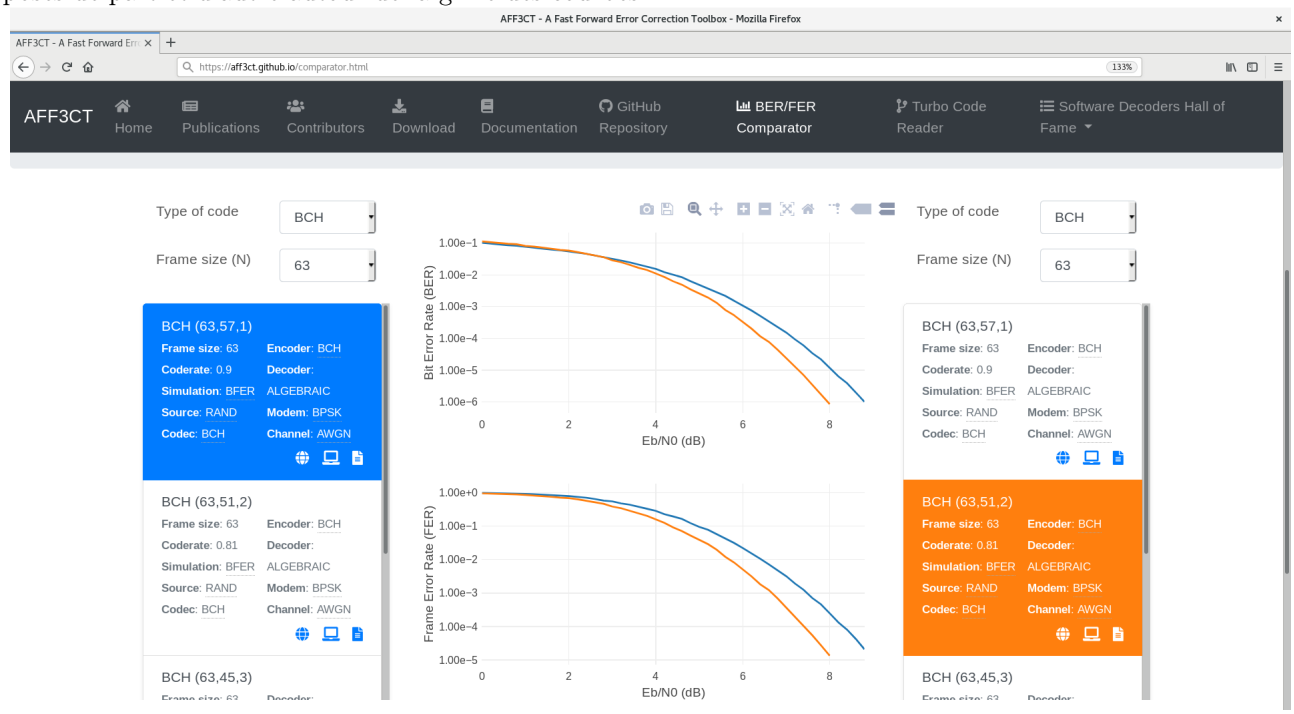



Simulation

Le message à envoyer est appelé Source de taille K (quantité d'information en bits). La source est envoyée en entrée au code correcteur d'erreurs qui va ajouter une surcouche au message pour ainsi obtenir une trame de taille N prête pour être modulée pour la transmission via un environnement appelé Canal. Le récepteur va démoduler l'information reçue puis la décoder pour enfin pouvoir lire l'information de taille K. Le logiciel AFF3CT va, entre autres choses, effectuer plusieurs simulations pour obtenir une courbe montrant l'écart entre le nombre émis et le nombre reçu en fonction de la force du signal, par exemple.

## 4. Site web d'AFF3CT

Le site web d'AFF3CT est hébergé à l'adresse https ://aff3ct.github.io. Mon travail au cours de ce stage portait uniquement sur la page https ://aff3ct.github.io/comparator.html.
Il s'agit du comparateur de codes correcteurs d'erreurs. On peut comparer 2 courbes dont les sélecteurs distincts sont disposés de part et d'autre autour de la grille des courbes.



Sur la version de base, on ne peut choisir qu'un type de code et qu'une taille de frame. On sélectionne les courbes en cliquant sur le rectangle affichant les informations de cette référence. On peut afficher la commande AFF3CT permettant d'obtenir les mêmes résultats qui servent à afficher cette courbe et également afficher le fichier lui-même.
En copiant l'URL, on peut partager sa propre comparaison.

### 5. Cahier des charges du nouveau site web

#### Fonctionnalités

-**Affichage de plusieurs courbes :** La première version du site AFF3CT permet d'afficher uniquement 2 courbes différentes. On souhaite afficher plus de courbes tout en maintenant une bonne lisibilité.

-**Possibilité d'afficher ses propres courbes :** En plus de celles de la base de données du site, l'utilisateur doit pouvoir charger ses propres courbes. Le chargement de fichier via le site doit se faire en toute sécurité. On ne souhaite pas créer une faille de sécurité en ajoutant cette fonctionnalité. On souhaite également garantir une certaine confidentialité au client s'il ne souhaite pas partager sa propre simulation.

-**Sélection de plusieurs courbes :** La sélection des références doit être plus élargie et plus élaborée. Sur la version de base du site, on ne pouvait afficher les références d'un seul type de code et d'une seule taille de frame de manière conjointe. On souhaite donc par exemple pouvoir afficher toutes les références d'un type de code toutes tailles de frame confondues. On souhaite également avoir une sélection multiple et plus de critères de sélection.

-**Possibilité d'envoyer la comparaison sélectionnée via l'URL :** Cette fonctionnalité était déjà présente sur la version de base du site web. Cependant, on souhaite conserver cette fonctionnalité, y compris avec une courbe chargée par l'utilisateur en local. Le tout doit être contenu dans l'URL comme à l'origine.

#### Organisation

-**Tri et organisation des références :** Les différentes références devront être classées avec finesse. En effet, les types de code et les tailles de frame des références seront diverses et variées.

#### Performances

-**Temps de chargement :** Le temps de chargement des références est assez long sur la version de base. On souhaite le réduire au maximum tout en conservant le même processus de chargement.

#### Visuel/Disposition

-**Réorganiser la disposition du site web :** La disposition de la version de base du site est quelque peu binaire. Sont présents deux sélecteurs, disposés à droite et à gauche avec au centre la grille accueillant le tracé des références sélectionnées. La disposition du nouveau site doit s'adapter aux nouvelles améliorations souhaitées.

-**Améliorer la lisibilité des références :** On souhaite également améliorer la lisibilité des références. Un bon compromis entre quantité d'information et nombre de références affichées est demandé. On souhaite également pour choisir d'afficher ou non les courbes sélectionnées.

## Propositions

---

### 1. Etude des technologies utilisables

#### Hébergement

Pour ce qui est du choix pour l'hébergement du site web, les possibilités gratuites offrant une bande passante correcte et un système d'intégration continue nous permet de réduire notre sélection à uniquement github.

### Choix des languages de programmation

Pour le visuel et la disposition des éléments du site, le HTML et le CSS sont de rigueur. En revanche, pour gérer l'intéractivité avec l'utilisateur, 2 possibilités s'offrent à nous : le PHP et le JavaScript. La différence de fonctionnement entre ces deux langages réside dans la localisation du traitement de l'interactivité. Le PHP sera exécuté sur le serveur hébergeant le site web tandis qu'en JavaScript, c'est la machine du client qui gère ce traitement.

### Choix de frameworks

Les frameworks pouvant être utilisés se limitent à ceux possédant une License MIT. Il s'agit d'une License logicielle libre, open source, dont l'utilisation peut être à but commercial ou non. Il est cependant obligé d'afficher le nom des auteurs du framework avec la notice de copyright.

## 2. Précision des concepts choisis

### Interaction côté client

Nous avons choisi de gérer la dynamique du site web uniquement côté client pour les avantages suivants : Interaction avec le serveur hébergeur uniquement au chargement de la page, sécurisation du serveur (aucun fichier ou requête n'est transmise au site web après le chargement effectué), délocalisation du temps de calcul. Notre choix s'est donc porté sur le JavaScript.

### Frameworks

En plus des frameworks déjà utilisés dans la version de base, voici la liste des frameworks, ajoutée par nos soins durant mon stage :

-Mustache : il s'agit d'un framework JavaScript permettant de créer des templates HTML. Son fonctionnement est simple : il suffit de créer un template HTML dans une balise <script> en insérant des variables reconnaissables car entourées par des doubles accolades. Du côté du JavaScript, à l'aide des fonctions fournies par le framework, il suffit de charger la balise script dans une variable, d'effectuer un rendu en indiquant quelle variable mustache correspond à quelle variable/valeur en JS (la syntaxe est similaire au JSON) puis à insérer le résultat dans la balise HTML souhaitée, notamment avec JQuery.
Cela nous permet de factoriser le code HTML et de bien distinguer le HTML du JavaScript afin de faciliter le débogage.

-Tippy : Tippy est un framework servant à afficher des tooltips. Il a été utile jusqu'à ce qu'on trouve le moyen d'afficher les titres de références entièrement.

-lz-string : Afin de conserver la transportabilité de la comparaison via l'URL, il nous fallait trouver un template afin de compresser le fichier texte permettant d'afficher la courbe et de l'insérer dans l'URL. lz-string permet de compresser une chaîne de caractère en JavaScript. Il possède également une méthode de compression adaptée à l'encodage pour l'insertion dans l'URL. Une fois l'entièreté de la base de données convertie en JSON, ce framework ne nous est plus d'aucune utilité.

### Hébergement

Github : Les avantages à héberger le site web sur github sont les suivants :

-L'hébergement est gratuit
-Grace à l'intégration continue git, il est aisé de travailler en équipe sur le site web. De plus, il dispose d'une traçabilité efficace en termes de contribution et une facilité à effectuer les tests. En cas de problème avec une nouvelle version, une seule ligne de commande suffit pour revenir à une version antérieure du site web.
Cependant, les paramètres du serveur ne peuvent pas être modifiés par nos soins. Il est donc impossible, entre autres exemples, de contrôler l'utilisation du cache navigateur sur le site.

Gitlab : La base de données des références du site sont stockées sur un serveur Gitlab. Ainsi, depuis le github, nous effectuons une requête Ajax au serveur Gitlab afin que le client reçoive la base de données.

CouchDB : Aujourd'hui, Le site n'utilise plus Gitlab pour stocker la base de données mais CouchDB qui est réservé au stockage de fichiers au format JSON. Actuellement, la base donnée d'AFF3CT est stocké au format JSON. De plus, lorsqu'un utilisateur charge sa propre courbe sur le site et souhaite obtenir un PermaLink pour l'envoyer à un autre utilisateur, sa courbe est convertie au format JSON et stockée sur CouchDB.

# Implémentations

## 1. Possibilité d'afficher jusqu'à 10 courbes

### Ré-écriture du moteur de courbe

Désormais, les courbes sont stockées dans un objet appelé Curves.

```
const Curves = {
    max: 10,
  length: 0,//number of displayed curves
  disponibility: [],//index==id! disponibility[id]=1 => available && disponibility[id]=0 => unavailable
  hidden: [],//1 if hidden else 0
  id: [],
  input: [],//input[nb_curve]= -1:no_curve||0:intern_curve||1:uploaded_curve
  names: [],//names[id]=name_of_the_curve
  referenceColors: ['#1f77b4', '#ff7f0e', '#2ca02c', '#d62728', '#9467bd', '#8c564b', '#e377c2', '#7
      f7f7f', '#bcbd22', '#17becf'], // Do not modify this tab!!! Use it as a reference
  colors: ['#1f77b4', '#ff7f0e', '#2ca02c', '#d62728', '#9467bd', '#8c564b', '#e377c2', '#7f7f7f', '#
      bcbd22', '#17becf'], // muted blue //safety orange // cooked asparagus green // brick red //
      muted purple // chestnut brown // raspberry yogurt pink // middle gray // curry yellow-green //
      blue-teal
  colorsOrder: [], // From 0 to 9
  refs: [],
    ...
};
```

Une fois la base de données téléchargée au chargement de la page, elle est stockée dans Curves.refs.

Avec l'aide d'Adrien CASSAGNE, nous avons pu simplifier la structure de la variable Curves pour ainsi obtenir ceci :

```
var Global = {
  refs: {}, // contains all the references indexed by id/hash (GitLab, Kaiserslautern, CouchDB & local)
  selectors: {
    dataBase:    { showZeros: true,  path: "metadata.source"             , selection: ["AFF3CT"] },
    codeType:    { showZeros: true,  path: "headers.Codec.Type"          , selection: [        ] },
    frameSize:   { showZeros: false, path: "headers.Codec.Frame size (N)", selection: [        ] },
    modemType:   { showZeros: false, path: "headers.Modem.Type"          , selection: [        ] },
    channelType: { showZeros: false, path: "headers.Channel.Type"        , selection: [        ] }
  },
  selectedIds: [], // the ids of the references that have been selected to be displayed
  filteredValueIds: [], // the ids of the reference filtered by the search bar
};
```

### Utilisation de templates html avec Mustache

A de nombreuses reprises, nous avons utilisé des templates Mustache afin d'éviter toute redondance de code HTML et également dans le but de dissocier le code HTML du code JavaScript.

Pour ce faire, il suffit de créer une balise script dans le fichier HTML et entourer les variables de doubles accolades comme ceci :

```
<script id="checkboxTemplate" type="x-tmpl-mustache">
    <input type="checkbox" class="form-check-input" id="{{element}}" title="{{element}}" onclick="{{
        function}}{{element}})" {{disabled}}><label class="form-check-label" for="{{element}}" title="
        {{element}}">{{{startBlackFont}}}{{element}} ({{length}}){{{endBlackFont}}}</label>
</script>
```

Puis dans le fichier JavaScript, il suffit de charger la balise script, insérer les variables à notre guise et charger le tout dynamiquement dans une balise HTML :

```
function displayCheckbox(length, font, endFont, disabled, fonction, i, div) {
  let checkboxTemplate = $('#checkboxTemplate').html();
  Mustache.parse(checkboxTemplate);
  let checkboxRendered=Mustache.render(checkboxTemplate, {
    element: i,
    disabled: disabled,
    startBlackFont: font,
    endBlackFont: endFont,
    length: length,
    function: fonction
  });
  $(div).append(checkboxRendered);
}
```

Cet exemple est utilisé dans le site pour afficher les checkboxs de sélection.

## 2. Améliorer l'expérience utilisateur

### Possibilité d'afficher/cacher les courbes

Pour cacher une courbe, l'astuce a été de réafficher tout le graphe avec uniquement les courbes non-masquées. Pour éviter un décalage de couleur pour ne pas "tromper" l'utilisateur, il a fallu prendre en compte uniquement les couleurs des courbes non-masquées. D'où la nécessité d'avoir un tableau servant de références couleurs et un autre contenant uniquement les couleurs des courbes à afficher.

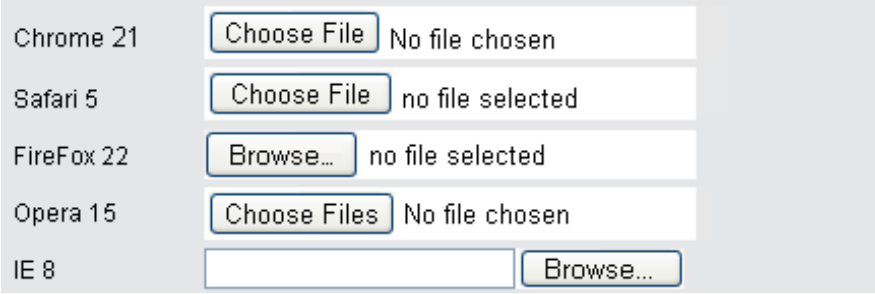
### Pouvoir supprimer les courbes

La suppression de courbe se fait directement avec le framework Plotly utilisé pour afficher les courbes.

```
Plotly.newPlot('myDiv', data, layout);
Plotly.deleteTraces('myDiv', index_of_the_curve);
```

### Charger ses propres courbes

Pour que l'utilisateur puisse charger ses propres courbes, il faut d'abord qu'il puisse sélectionner les fichiers depuis le disque dur de son ordinateur. Nous avons utilisé une balise HTML input. Voici le rendu de la balise :




Ensuite, j'ai réutilisé une fonction déjà présente depuis la version de base du site. Cette fonction parse le fichier texte chargé pour obtenir toute les informations nécessaires à l'affichage de la courbe. La courbe est ensuite affichée avec Plotly de la même manière qu'une courbe de la base de données.

### Stocker les courbes chargées dans l'URL

Pour stocker les courbes chargées dans l'URL, j'ai téléchargé le framework lzstring qui permet de compresser une chaîne de caractères. La méthode de compression est compatible avec l'encodage pour URL. On retrouve donc le fichier texte original dans l'URL. Du fait de la taille des fichiers, le chargement de deux fichiers textes sature l'URL car la plupart des navigateurs limitent les tailles d'URL.
Grâce à CouchDB et la convertion en JSON, nous stockons uniquement l'identifiant à 7 caractères associé à la courbe chargée dans la base de données.

## 3. Réduire le temps de chargement de la base de données

Sur la version de base du site web, le chargement était relativement long. En effet, le site envoyait autant de requête que le nombre de fichiers contenus sur la base de données Gitlab.

### Ajouter un script pour concaténer la base de données

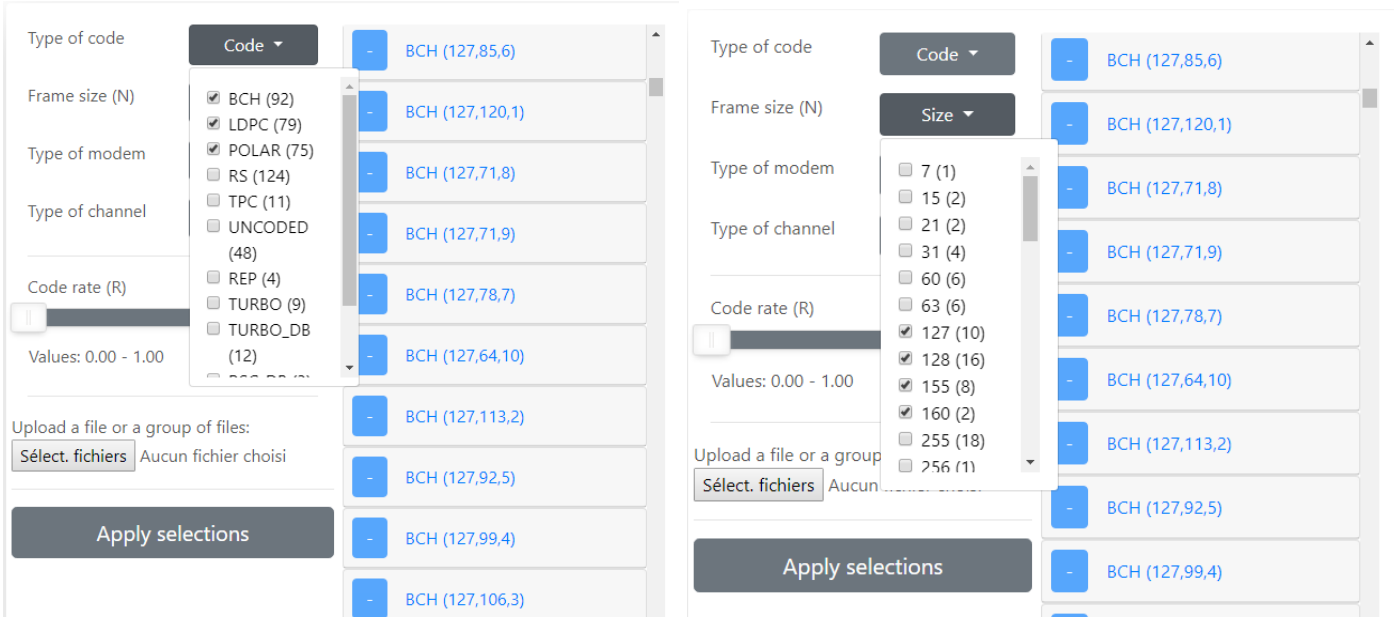
J'ai donc eu comme idée de concaténer tous les fichiers de la base de données afin d'avoir une unique requête et envoyer toute la base de donnée en une fois. Voici mon script écrit en bash :

```
#!/bin/bash
debutFichier='Start_File'
finFichier='End_File'
root=./error_rate_references-master

function directory

{
  for element in `ls $1`
  do
    if [ -d "$1/$element" ]
    then
      echo -e "$1/$element\n" >> ../result.txt
      directory "$1/$element"
    else
      echo -e "$1/$element:\n" >> ../result.txt
      echo -e "$debutFichier\n" >> ../result.txt
      cat "$1/$element" >> ../result.txt
      echo -e "$finFichier\n" >> ../result.txt
    fi
  done
}

echo -e "" > ../result.txt
directory $root
```

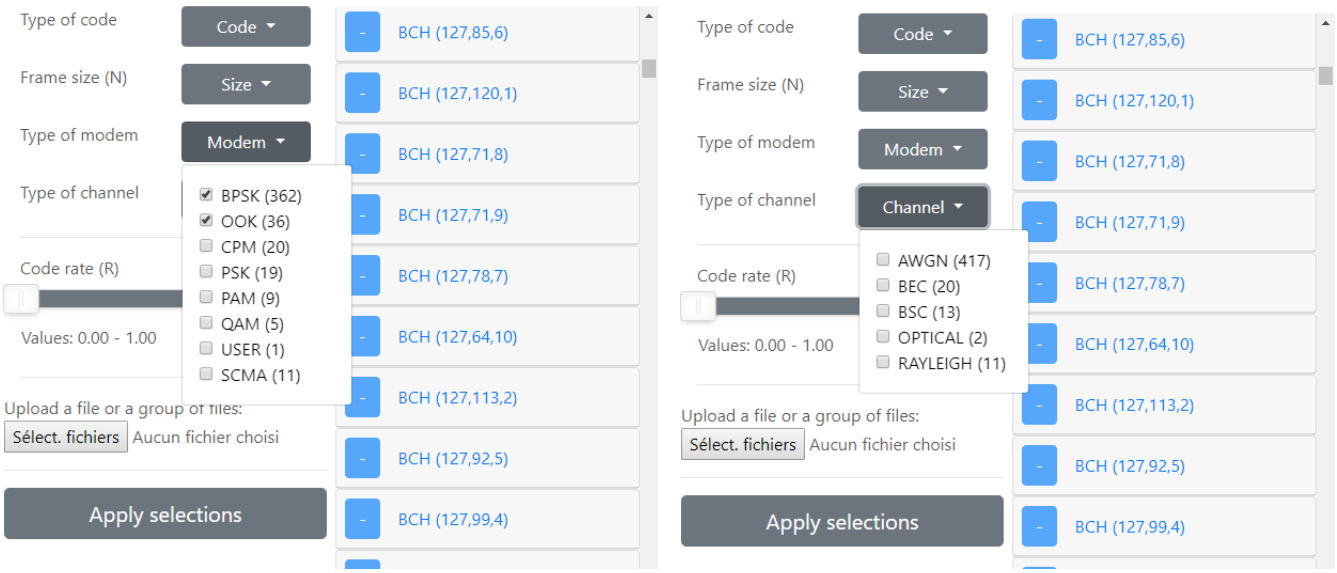
Pour parser plus facilement ce fichier, j'ai ajouté des séparateurs entre les différentes références.

## 4. Réorganisation de la sélection de fichiers

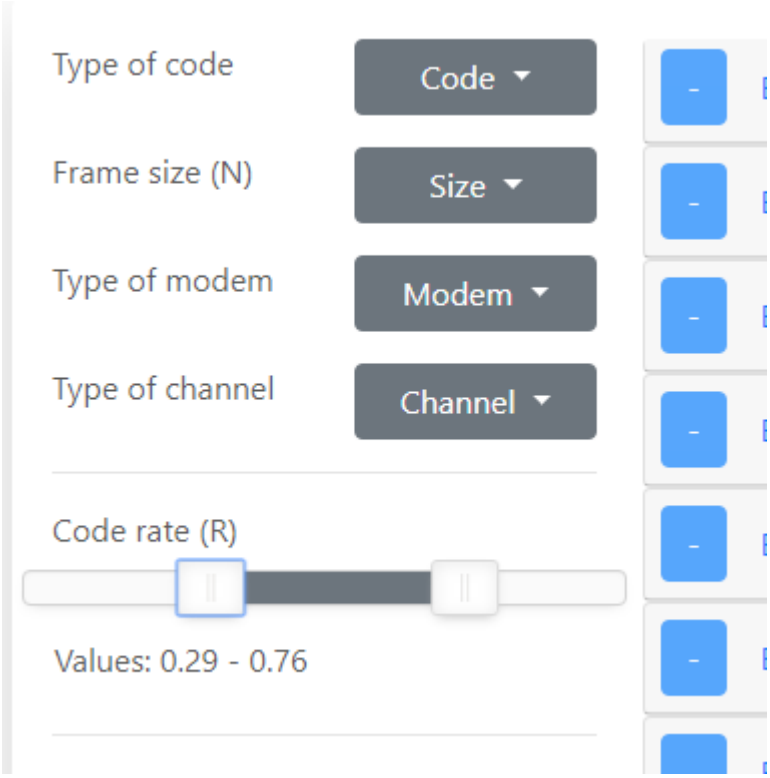
La séléection des références a été totalement revisitée. Désormais, il est possible de choisir plusieurs types de codes différents, ainsi que plusieurs tailles de trames différentes.



On peut également choisir le type de modem et de canal :

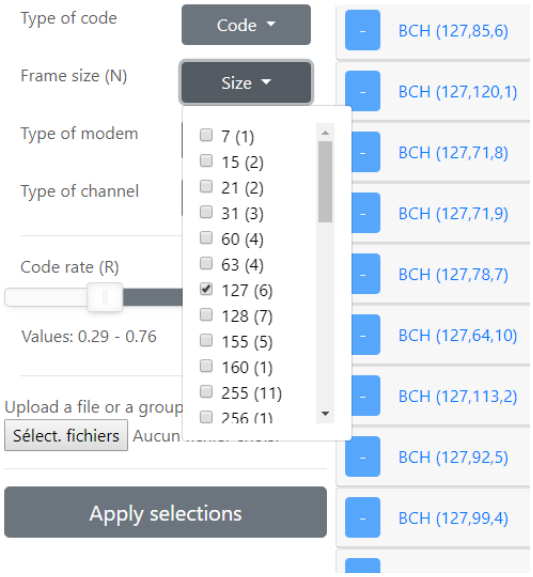



De plus, j'ai intégré une barre pour sélectionner un ensemble de valeur de coderate comprises entre 0 et 1 :

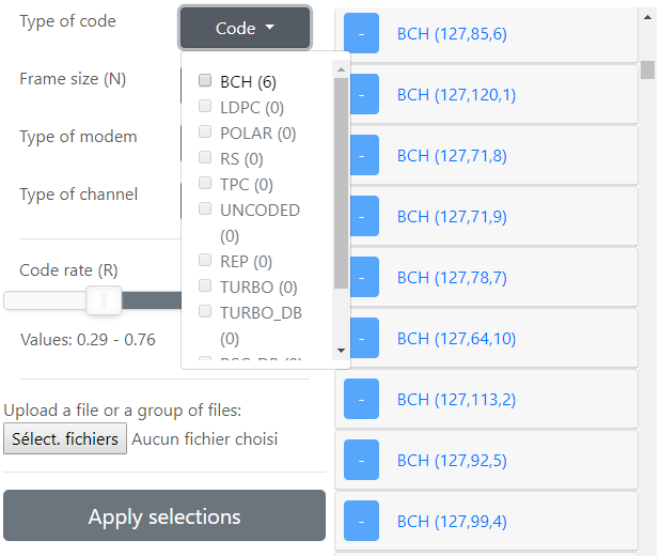



Hormis le choix de coderate, chaque sélection est suivie par un nombre emprisonné entre deux parenthèses. Il indique le nombre de références disponibles.

La sélection de paramètres pour assister l'utilisateur lors du choix de références à comparer est adaptatif. Des choix de paramètres incompatibles est impossible et le nombre de références disponibles se met à jour en fonction des choix effectués. Voici un exemple :



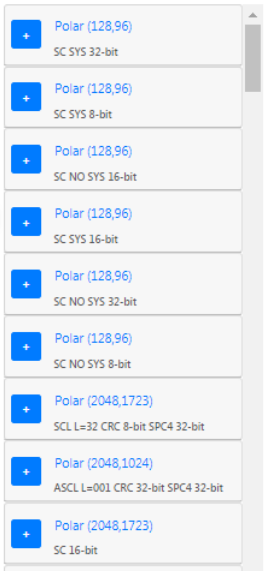

On peut remarquer que j'ai choisi une taille de trame à 127.




En regardant les types de codes, on peut voir que seul BCH possède des références avec une taille de frame à 127. Le site en compte 6. Il est donc impossible de sélectionner d'autres types de codes incompatibles avec une taille de 127.

### Séparation des titres

En parcourant les différentes références, on peut remarquer que certaines ont des titres assez long mais très explicites. Une contrainte du cahier des charges demandait une bonne visibilité des titres. Pour celà, j'ai d'abord pensé à ajouter des tooltips. Puis j'ai décidé de séparer le titre en 2 partie. La première partie contient tout le texte qui précède les nombres entourés de parenthèses ainsi que les parenthèses elles-mêmes. La deuxième partie contient le reste du titre.




**Tooltips pour les titres**

Cependant, il se peut que certains titres ne respectent pas ou ne respecteront pas ce format. Dans ce cas, s'il est trop long, il sera troncaturé et en passant la souris sur le titre, un tooltip apparait pour montrer le nom complet.

# Evaluation

## 1. Tests

Pour pouvoir travailler et faire mes tests sans perturber l'actuel site web d'AFF3CT, j'ai fait un fork du dépôt Git du site d'AFF3CT hébergé sur le site https ://mnaciri95.github.io. Ainsi je travaille en local pour effectuer mes tests depuis le navigateur Mozilla Firefox en mode développeur, le tout en local. Une fois qu'une tâche est accomplie, je push mes changements sur la version en ligne et le site web se met à jour presque instantanément.

La limite de cette méthode de test a été rencontrée lors de la modification du fonctionnement de l'URL. En effet, l'URL n'est pas disponible en local. De nombreux commits ont été effectués avec peu de changements

## 2. Mise en oeuvre

La mise en oeuvre de se projet devait répondre à certaines contraintes. Le temps était limité. Mon stage ne durait que 12 semaines. Il fallait donc accomplir un maximum de tâches du cahier des charges dans ce délai imparti. Les performances du site ne doivent pas décliner, au contraire. Il m'a notamment été demandé de réduire le temps de chargement. Le site web doit également être compatible avec la plupart des systèmes d'exploitation mais également avec la plupart des navigateurs.

Le site web doit également être compatible avec la plupart des résolutions d'écran. Etant donné le contexte, il n'est pas nécessaire d'adapter le site web aux petites résolutions. Aussi, la résolution minimale pour utiliser le site web est fixée à 1920x1080 pixels.

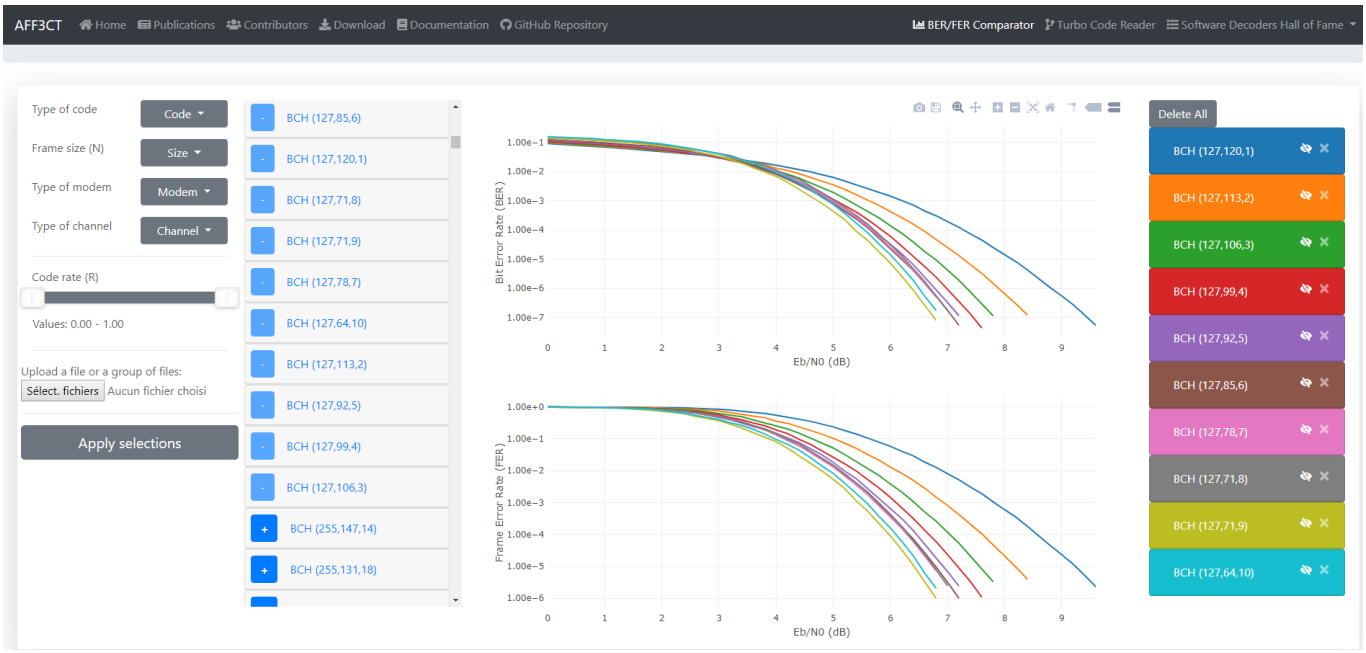
## 3. Problème mémoire

Après avoir concaténé la base de données composée de fichiers texte, il faut extraire chaque références dans le fichier. Or pour l'extraire, il est possible d'utiliser le prototype $.substring(a, b)$ dont $a$ et $b$ sont deux nombres tels que $0 \leq a < b$.

Cependant, ce prototype Javascript fait une copie du fichier entier pour afficher la partie du fichier sélectionnée. Le fichier pesant 7 Mo avec 465 substring effectués correspondant au nombre de références, on se retrouvait avec 1,7 Go de mémoire RAM utilisée. Pour pallier à ce problème, nous avons décidé de stocker la base de données sous format json afin de gagner en mémoire ram consommée et afin de réduire le temps de chargement. En effet, la tâche consistant à passer d'un fichier texte à des données prêtes à être affichées est fait en amont.

# Conclusion

Aujourd'hui, le site web permet à l'utilisateur de comparer jusqu'à 10 courbes différentes. Il est plus facile d'affiner sa recherche en vue d'une comparaison. Il est possible de charger ses propres courbes et de les envoyer facilement à un autre utilisateur. Le temps de chargement a été drastiquement réduit. Voici une capture d'écran du site web à la fin de mon stage :




Le cahier des charges a été respecté. Avec un peu plus de temps, j'aurais voulu ajouter quelques fonctionnalités au site web. Par exemple, il peut être pratique de pouvoir annuler toutes les sélections d'un simple clic. De plus, il aurait été pratique mais peu utile d'ajouter une version mobile au site web.

## 1. Enrichissement personnel

Au cours de ce stage, j'ai pu découvrir le monde de la recherche informatique. Au sein du laboratoire de recherche, on rencontre des personnes aux profils et compétences variés.

J'ai également enrichi mon panel de compétences informatique car j'y ai appris le HTML, le Java Script, le CSS mais j'ai également appris à utiliser bon nombre de frameworks.

Je remercie Monsieur Denis BARTHOU de m'avoir donné l'opportunité et la chance d'effectuer mon stage au sein de l'équipe d'AFF3CT. Je remercie également Adrien CASSAGNE pour m'avoir aidé et accompagné au cours de ce stage.