ATIAM 2018
Rapport de stage

Simulation temps réel d'Ondes Martenot

Stage pluridisciplinaire dans le cadre du Collegium Musicae

Equipe Conservation et Recherche - Musée de la Musique
& Equipe S3AM - Laboratoire STMS

Encadrants : Thomas Hélie, David Roze, Thierry Maniguet, Stéphane Vaiedelich
Collaborateur : Henri Boutin

1er février 2018 - 31 juillet 2018

Judy Najnudel

Résumé

Ce mémoire de stage de master ATIAM traite de la modélisation et la simulation en temps réel d'Ondes Martenot, instrument de musique électro-acoustique du patrimoine. Cette modélisation s'appuie sur le formalisme des Systèmes Hamiltoniens à Ports, formalisme respectant le bilan de puissance et garantissant la passivité des simulations. Les contributions principales lors de ce stage sont constituées d'un algorithme de résolution automatique de conflits de réalisabilité pour la simulation de circuits électroniques et son intégration dans une bibliothèque open-source dédiée à la modélisation, ainsi que la conception d'un plugin temps réel reproduisant le son issu du circuit d'une Onde Martenot.
Dans une première partie, nous décrivons le fonctionnement de l'Onde Martenot en détail et présentons les outils *ad hoc* permettant de la modéliser. Dans une seconde partie, nous exposons nos contributions et résultats.

Mots-clé : Patrimoine - Instrument de musique - Electronique analogique non linéaire - Modèle physique - Simulation numérique - Passivité - Réalisabilité

Abstract

This ATIAM master thesis addresses the modeling and simulation of the Ondes Martenot, a classic electrical musical instrument. This modeling lies on the Port-Hamiltonian System formalism, which respects the power balance and guarantees the passivity of simulations. Principal contributions are an automatic solving algorithm for realizability issues in electronic circuits and its integration in a dedicated open-source library, as well as a real-time plugin for sound reproduction of an Onde Martenot circuit.
First, the Onde Martenot and modeling tools are thoroughly described. Then, our contributions and results are presented.

Keywords : Legacy - Musical instrument - Non-linear analog electronics - Physical modeling - Numerical simulation - Passivity - Realizability

Table des matières

Table des figures

Liste des tableaux

Introduction

Les Ondes Martenot tiennent une place particulière au sein de notre patrimoine culturel. Inventée par un pédagogue français, parmi les tous premiers instruments de musique dont le son est produit électriquement, l'Onde Martenot s'est largement affranchie du rôle de simple curiosité technologique puisque 90 ans après son invention, elle peut se targuer d'un répertoire de milliers d'oeuvres dont une bonne partie de la main de grands compositeurs, et est encore jouée aujourd'hui partout dans le monde par de prestigieux interprètes. Cependant, sa production est arrêtée depuis une trentaine d'années et les instruments en état de jeu se font de plus en plus rares. Fort de sa mission de conservation, le Musée de la Musique à Paris en a acquis plusieurs exemplaires. Certains composants devenus obsolètes, comment préserver et faire entendre son identité sonore dans un contexte muséal interdisant les interventions de nature irréversible, comme le remplacement de composants ? Une solution consiste à produire un fac-simile virtuel jouable, en modélisant l'instrument à l'aide d'outils de modélisation physique adaptés à ses particularités, notamment les non-linéarités. C'est l'objet de ce travail de stage, qui s'articule en deux parties. Dans la première partie, nous établissons un état des connaissances organologiques sur les Ondes Martenot et présentons les outils adoptés pour les modéliser. Ces outils reposent sur le formalisme des Systèmes Hamiltoniens à Ports, formalisme respectant le bilan de puissance et garantissant la passivité et donc la stabilité des simulations. Dans la seconde partie, nous exposons nos contributions et résultats de cette modélisation, à savoir un générateur de code de simulation à bilan de puissance équilibré avec génération automatique de documentation, un algorithme de résolution automatique de conflits de réalisabilité implémenté dans la bibliothèque open-source PyPHS, et un plugin temps réel contrôlable reproduisant le son issu du circuit d'une Onde Martenot, programmé en C++ dans l'environnement JUCE.

Hébergé du 1er février 2018 au 31 juillet 2018 par l'équipe S3AM (Systèmes et Signaux Sonores, Audio/Acoustique, instruMents) du laboratoire STMS à l'IRCAM, ce stage a été financé par l'Equipe Conservation et Recherche du Musée de la Musique à travers le Collegium Musicae. Cet institut de Sorbonne Universités, composé de neuf organismes de recherche et formation dont le STMS et le Musée de la Musique, se donne pour mission de considérer les musiques et leurs pratiques sous différents angles aussi bien scientifiques qu'artistiques, et structure ses projets en trois axes : Analyse | Création, Instruments | Interprètes et Archives | Patrimoine. Dans le cadre de ces deux derniers, ce stage s'inscrit dans un projet plus large répondant à des problématiques de conservation et valorisation des instruments de musique électroniques et prévoyant également un travail de conception de composants programmables et mesures sur un corpus du Musée, objet d'un autre stage en cours.

Première partie

Etat de l'art et approche

Avant de pouvoir simuler une Onde Martenot et concevoir un fac-simile virtuel, nous devons rassembler tous les éléments nécessaires à sa modélisation par modèle physique et comprendre leur fonctionnement. Dans cette première partie, nous analysons donc en premier lieu notre instrument d'étude, à travers une présentation générale des Ondes Martenot d'abord puis une étude détaillée du circuit et composants d'un exemplaire particulier. Nous introduisons ensuite les outils grâce auxquels nous en réaliserons une simulation à passivité garantie.

1 Présentation des Ondes Martenot

Après un rappel du contexte de son invention et de la place culturelle des Ondes Martenot, cette section décrit les éléments constitutifs de l'instrument en insistant particulièrement sur le circuit, ses étages et composants, qui fera l'objet d'une modélisation puis simulation dans la deuxième partie du rapport.

1.1 Contexte historique et musicologique

FIGURE 1 – Onde Martenot 146, E.2017.4.1, Maurice L. E. Martenot, 1937 - Musée de la Musique, Paris.

Les années 1920 furent une période faste en termes de facture instrumentale, ayant en particulier vu émerger un tout nouvel instrumentarium, annonçant la révolution électronique à venir de la production du son. En effet, sont apparus à quelques années d'intervalle le Theremin (1919) en Russie, les Ondes Martenot (1928) en France et le Trautonium (1929) en Allemagne. Ces précurseurs des synthétiseurs électriques, particulièrement expressifs, reposent tous sur le principe du circuit hétérodyne à lampes, conçu pour la détection de signaux en télégraphie sans fil (TSF). Mais seules les Ondes Martenot (figure 1) ont inspiré un répertoire fort de 1200 oeuvres, encore jouées aujourd'hui. Parmi ses compositeurs phares, on peut citer Olivier Messiaen, Tristan Murail, Edgard Varèse ou Darius Milhaud, ou plus près de nous, Jonny Greenwood. On peut également entendre des Ondes Martenot dans de nombreuses chansons populaires (Brel, Ferré, Tom Waits, Radiohead ...) et musiques de films (Mad Max, Mars Attacks, Docteur Jivago, Ghostbusters ...). Malgré leur succès auprès du public et une communauté de musiciens toujours plus importante, la production des Ondes Martenot s'est arrêtée en 1988, à la retraite de l'assistant de Maurice Martenot, leur inventeur. Des 281 instruments originaux, il n'en subsiste aujourd'hui que 70, dont 20 exemplaires sont au Musée de la Musique à Paris, et seulement quelques-uns en état de jeu. Chaque exemplaire a son identité propre, d'une part car les Ondes étaient fabriquées de manière artisanale (une tentative de production en série confiée à la maison Gaveau fut rapidement abandonnée tant les résultats étaient décevants), d'autre part car comme tout inventeur, Maurice Martenot ne se privait pas d'apporter les améliorations qu'il jugeait opportunes à chaque itération. Les dernières versions, à partir des années 1970, sont par exemple dotées de transistors à la place des lampes. Le principe général demeure cependant le même : la tension issue d'un circuit hétérodyne est amplifiée

puis envoyée dans un diffuseur (variante du haut-parleur colorant fortement le son). Le musicien, appelé ondiste, peut contrôler la fréquence de jeu à la main droite grâce un ruban continu ou un clavier note à note permettant tout de même des vibrati, tandis que sa main gauche agit sur une touche d'intensité à grande sensibilité pour les nuances et l'expression. La section suivante présente ces différents éléments de façon plus détaillée.

1.2 Description de l'instrument

1.2.1 Circuit hétérodyne

Le circuit hétérodyne est une technique ancienne de démodulation d'amplitude, inventée en 1901 par l'ingénieur canadien Reginald Fessenden [5] et utilisée depuis dans la quasi-totalité des systèmes de télécommunications. L'expérience de caporal radiotélégraphiste de Maurice Martenot durant la Première Guerre Mondiale l'a fortement familiarisé avec ce principe. Son fonctionnement dans le cadre des Ondes Martenot est le suivant : deux oscillateurs produisent chacun une tension sinusoïdale V1 et V2 de fréquence élevée, autour de 80 kHz. L'une est fixe, l'autre est variable et contrôlable par le musicien. Leur somme produit une tension V3 modulée en amplitude, et c'est cette modulation, après détection et redressement V4 par le mélangeur ou démodulateur, qui constitue le signal audible.

1.2.2 Contrôles

Fréquence de jeu L'Onde Martenot est un instrument monodique : on ne peut jouer qu'une note à la fois. Les premières versions de l'instrument n'ont pas de clavier et sont dotées d'un ruban que le musicien peut faire coulisser entre les lames d'un condensateur placé entre le clavier et le circuit (figure 2). Seule une partie du ruban est conductrice, et permet à la capacité du condensateur de varier en fonction de la position du ruban. Ce condensateur étant relié au circuit résonant de l'oscillateur variable, le geste du musicien modifie directement sa fréquence d'oscillation. Les lames du condensateur sont fendues et déformables pour faciliter l'accord note par note, et le condensateur est subdivisé par octave pour faciliter sa réalisation. En effet, la capacité d'un condensateur plan à diélectrique air est de la forme $C = \epsilon_0 \times \frac{S}{d}$ où ϵ_0 est la permittivité de l'air, S la surface des armatures en regard et d la distance entre ces plans, qui doit donc augmenter avec la fréquence. Or pour couvrir les 7 octaves de jeu, il faudrait une plage d'écartement considérable. Maurice Martenot prend donc le parti de fabriquer un condensateur par octave et de les connecter en parallèle pour obtenir leur capacité équivalente [4]. Dans un premier temps, le clavier n'a pas d'autre fonction que celle de repère pour la position du ruban. Par la suite, Maurice Martenot ajoute un vrai clavier qui commande une inductance variable constituée d'autant d'inductances mises en série que de touches, elle aussi reliée au circuit résonant de l'oscillateur, et le musicien peut passer d'un mode à l'autre à loisir à l'aide d'un commutateur. Une poussée latérale sur les touches permet en outre de jouer en vibrato.

Intensité En plus de contrôler la note jouée, le musicien peut également agir sur son niveau sonore grâce à la touche d'intensité, qui se trouve dans un petit tiroir à gauche du clavier. La touche d'intensité est peut-être l'élément qui a le plus contribué au succès de l'instrument. Elle se comporte comme une résistance variable entre l'alimentation et l'anode de l'amplificateur de puissance, en amont du diffuseur. Lorsque la touche est relâchée, la résistance est infinie et aucun courant ne circule dans l'amplificateur. Lorsque la touche est enfoncée, le courant peut circuler. Le génie de Maurice Martenot réside dans la technique de mise en oeuvre de cette résistance, qui permet les nuances intermédiaires entre les positions ouverte et fermée et peut restituer toute l'expressivité du musicien avec une réponse haptique adaptée. Elle est constituée d'un sac souple de poudres de graphite (conducteur) et mica (isolant) (figure 3), que la touche vient compresser de façon réversible, modifiant ainsi les trajectoires formées par les grains de graphite en contact suivies par le courant et la distance entre les électrodes. Cette touche d'intensité ne sera pas modélisée dans le cadre du stage mais a fait l'objet d'une étude approfondie par Quartier *et al*[6], qui concluent notamment que l'intensité du son dépend de l'enfoncement de la touche et de la force qui lui est appliquée selon une courbe adaptée à l'humain (en position pour les nuances faibles, en force pour les nuances élevées), pas de la vélocité, et que la dynamique résultante est de l'ordre de 50 dB SPL.

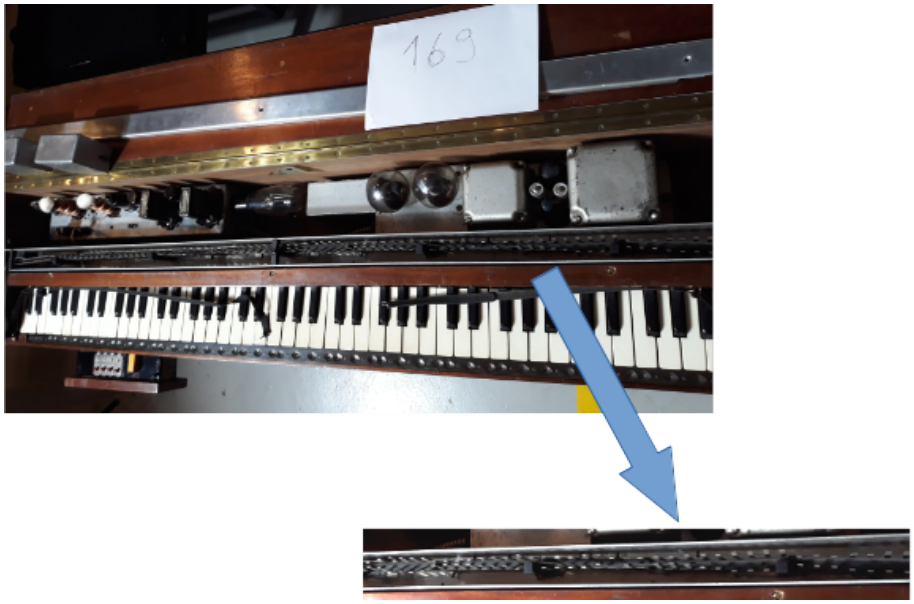

FIGURE 2 – Contrôles et circuit de l'Onde 169 et mise en évidence du condensateur variable - Musée de la Musique, Paris.

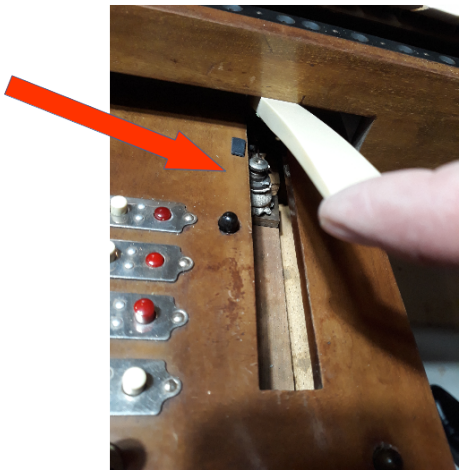

FIGURE 3 – Dessous de la touche d'intensité de l'Onde 146 et mise en évidence du sac de poudre - Musée de la Musique, Paris

1.2.3 Filtres

Le tiroir contenant la touche d'intensité permet également de modifier le timbre des notes en activant ou non un à quatre filtres constitués d'une résistance et de condensateurs en cascade.

1.2.4 Diffuseurs

Les diffuseurs (figure 4) sont des dispositifs conçus par Maurice Martenot pour rayonner le son avec une coloration spécifique. De plusieurs types, ils peuvent être utilisés seuls ou combinés. Le D1, ou Principal, est un haut-parleur simple dont la dynamique est à même de traduire toutes les nuances possibles de la touche d'intensité. Le D2, ou Résonance, se comporte comme une réverbération grâce à un système de ressorts [7]. Le D3, ou Métallique, est fortement non-linéaire car couplé à un gong. La Palme est couplée avec 2x12 cordes montées sur une caisse de résonance qui vibrent par sympathie. Ces diffuseurs ne seront pas modélisés dans le cadre du stage mais font l'objet d'un projet de travail d'identification en lien avec les travaux de Damien Bouvier [8] et Tristan Lebrun [9].

FIGURE 4 – Onde et diffuseurs : de gauche à droite Palme, Métallique et Résonance - Wikipedia.

1.3 Circuit étudié

1.3.1 Choix de l'Onde 169

Le circuit des Ondes Martenot étant bien entendu un élément clé de la production du son, une analyse approfondie préalable est indispensable à la construction d'un modèle pour simulation. On garde cependant à l'esprit qu'établir un modèle "type" des ondes Martenot n'a pas grand sens puisque il y a quasiment autant de schémas électriques que d'instruments. On concentrera donc nos travaux sur une version particulière, l'Onde 169 (figure 5). Cette Onde est bien adaptée au cadre du stage car d'une part c'est un spécimen de la collection du Musée, ayant fait l'objet d'une analyse de fonctionnement par Thierry Courrier, électronicien mandaté par le Musée en 2012 mettant ainsi à notre disposition un schéma détaillé et de nombreuses autres informations de facture [4] ; on évite ainsi nombre de manipulations sur des instruments coûteux et fragiles. D'autre part c'est une version à la fois suffisamment "avancée" pour posséder les caractéristiques complètes que la plupart des ondistes seraient susceptibles de souhaiter retrouver dans une reproduction virtuelle (notamment un clavier), et suffisamment "ancienne" pour que son étude ait un intérêt patrimonial indiscutable, en raison de la présence de lampes triodes.

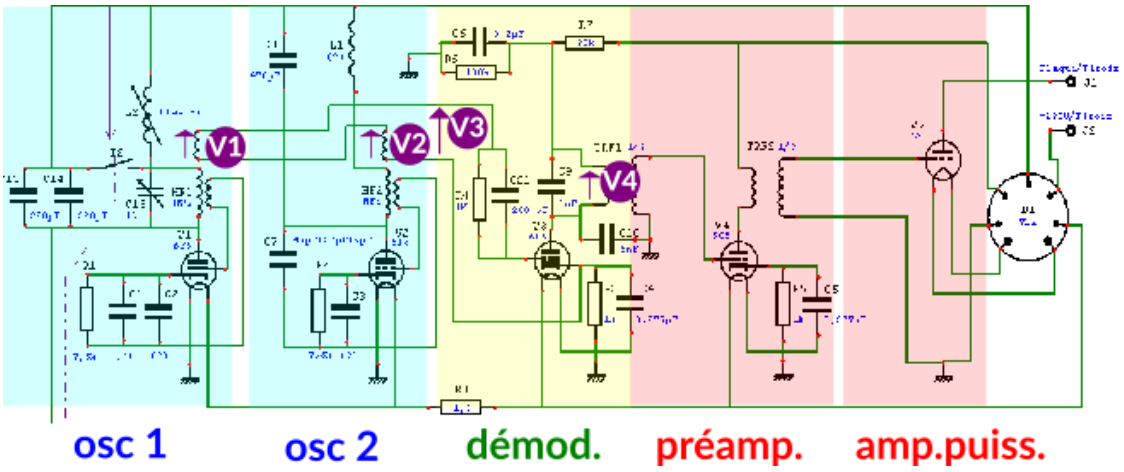

FIGURE 5 – Schéma électronique de l'Onde 169 fourni par le Musée de la Musique avec ses étages constitutifs. V1 est la tension en sortie de l'oscillateur fixe, V2 la tension en sortie de l'oscillateur variable, V3 la tension en entrée du mélangeur et V4 la tension en sortie du mélangeur.

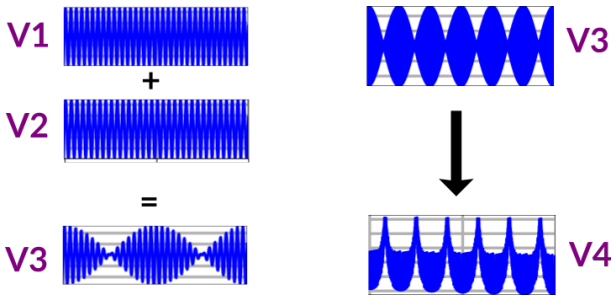

FIGURE 6 – Signaux obtenus par simulation à chaque étage de l'Onde 169.

1.3.2 Fonctionnement général de la triode

Les 5 étages (oscillateur fixe, oscillateur variable, mélangeur, préamplificateur et amplificateur de puissance) du circuit reposent tous sur l'utilisation de lampes triodes, composants non linéaires. Ces non-linéarités apportent des harmoniques et jouent un rôle prépondérant dans le timbre de l'instrument en particulier au niveau du mélangeur (figure 6). La triode, inventée en 1906 par Lee de Forest [10], a rapidement trouvé des applications dans l'ensemble des télécommunications grâce à ses propriétés amplificatrices. L'Onde 169 en l'occurrence exploite différents modèles dont la 6F5 et la 2A3 [4] (figures 7 et 8). Une description du fonctionnement de la triode peut être trouvée

FIGURE 7 – Triode 6F5 utilisée dans les oscillateurs de l'Onde 169.

FIGURE 8 – Triode 2A3 de puissance utilisée dans l'amplificateur de l'Onde 169.

dans [11]. Nous en résumons les principaux éléments ci-dessous.

Dans la triode, une grille est placée entre l'anode (*plate* en anglais), mise à un potentiel V_p positif par une source extérieure haute tension continue, et la cathode mise à un potentiel V_c négatif, afin de contrôler le courant I_p qui traverse le composant (figure 9). En effet, si la grille est à un potentiel V_g trop négatif par rapport à la cathode, aucun courant ne circule entre l'anode et la cathode. Si V_g est légèrement négatif par rapport à la cathode, le courant peut circuler de l'anode vers la cathode. Si V_g est positif par rapport à la cathode, un courant de grille apparaît et "capte" une partie du courant entre l'anode et la cathode. Lorsque V_g dépasse un certain seuil, le courant d'anode n'augmente plus quelle que soit la tension appliquée à la grille et on dit alors que la lampe est saturée par la grille. Il existe également un phénomène de saturation par la plaque : à partir d'une certaine valeur, l'augmentation du potentiel d'anode n'a plus d'influence sur le courant traversant la triode. Exceptés ces phénomènes de saturation responsables d'une partie des non-linéarités, le

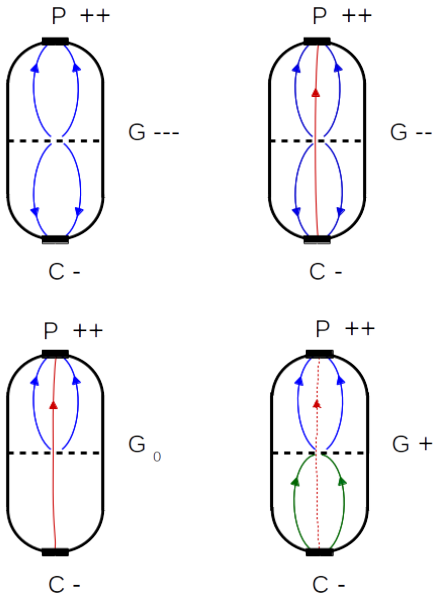

FIGURE 9 – Comportements possibles du courant dans une triode, en fonction des potentiels d'anode P, de cathode C et de grille G (les flèches indiquent le sens des électrons). En rouge : courant d'anode I_p ; en vert : courant de grille I_g ; en bleu : courant résiduel.

courant d'anode suit les modulations de la tension appliquée à la grille. Ce courant est transformé en tension en plaçant une résistance de charge R_p entre l'anode et l'alimentation de la triode : plus la valeur de R_p sera élevée, plus la tension résultante sera forte. Les courbes caractéristiques en figure 10 permettent de mieux comprendre les propriétés amplificatrices du composant. Le courant d'anode I_p est tracé en fonction de la tension V_{pc} pour plusieurs valeurs de tension V_{gc} données. Deux coefficients permettent de caractériser le comportement de la triode : la transconductance (ou pente) $gm = \partial I_p/\partial V_{gc}$ et la résistance d'anode $ra = \partial V_{pc}/\partial I_p$. Pour obtenir la transconductance (en Siemens, correspondant à l'inverse d'une résistance), on fixe V_{pc} et on calcule le rapport entre la variation de I_p et la variation de V_{gc}. Pour obtenir la résistance d'anode, on fixe V_{gc} et on calcule le rapport entre la variation de V_{pc} et la variation de I_p.

Dans le cas d'une triode 6F5, pour une tension d'anode de 100V : lorsque V_{gc} varie de 0.5V, I_p varie de 0.8mA et gm vaut donc $(0.8 \times 10^{-3})/0.5 = 1.6mS$. Pour une tension de grille de -0.5V : lorsque V_{pc} varie de 50V, I_p varie de 0.75mA et ra vaut donc $50/(0.75 \times 10^{-3}) = 67k\Omega$. L'amplification μ étant définie par $\mu = \partial V_{pc}/\partial V_{gc}$, on obtient immédiatement $\mu = ra \times gm = 107$. On se rend cependant rapidement compte en regardant les courbes caractéristiques en figure 10 que selon le domaine d'utilisation de la triode, les non-linéarités sont plus ou moins importantes, les écarts entre les courbes ainsi que leurs pentes varient et donc également les valeurs de gm, ra et μ. On remarque également qu'en plus de l'amplification, la triode réalise une inversion en tension : une variation négative de V_{gc} implique une variation positive de V_{pc}.

Dans la plupart des montages à triode à cathode commune (l'entrée V_{gc} et la sortie V_{pc} de la triode ont la cathode en commun), les électroniciens insèrent en outre une résistance d'autorégulation R_k et un condensateur de découplage C_k entre la cathode et la masse (figure 11). La résistance d'autorégulation permet de s'assurer que le potentiel de la grille reste inférieur à celui de la cathode (cadre souhaité de fonctionnement de la triode) et évite ainsi le recours à une source extérieure de polarisation. En effet, à la mise sous tension de la lampe et au repos (pas de tension appliquée à la grille), un courant circule aux bornes de R_k, créant une différence de potentiel entre V_c et la masse. V_c devient donc positif par rapport à V_g resté à 0, et la tension V_{gc} devient par conséquent négative. On souhaite par ailleurs fixer le potentiel V_c afin que la tension V_{gc} amplifiée soit la tension d'entrée appliquée à la grille à une constante près. Dans ce but, le condensateur C_k agit comme une réserve de courant et compense les variations de tension aux bornes de R_k dues aux variations du courant I_p. Plusieurs modèles physiques de triode sont disponibles [12] [13] et utilisables en audio [14] [15]. Le modèle choisi dans le cadre du stage est présenté en section 3. Ces principes étant posés, nous pouvons nous pencher de façon plus détaillée sur le fonctionnement du

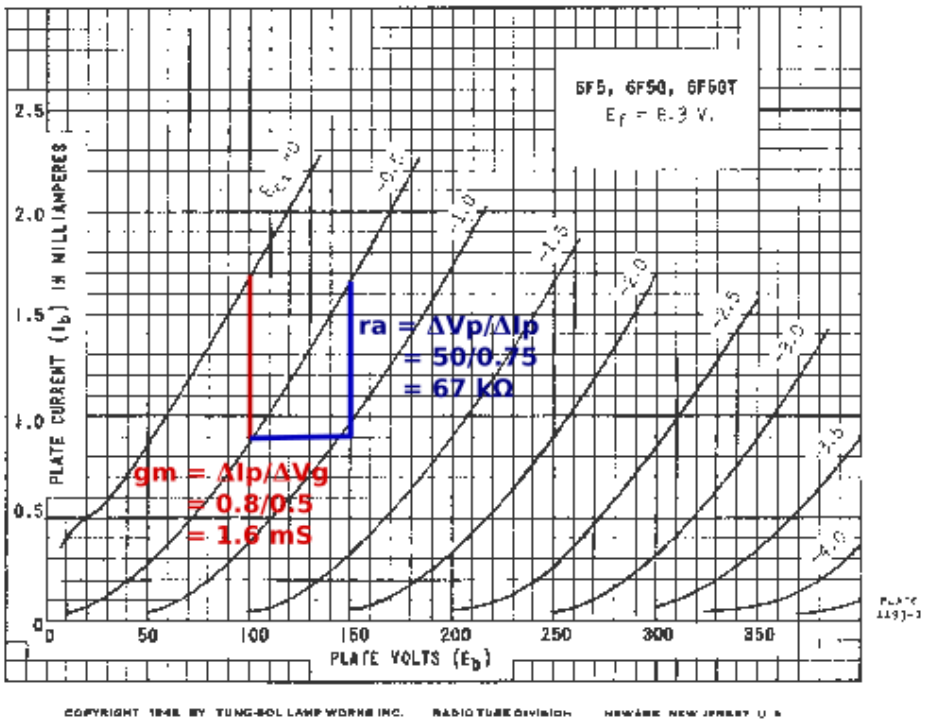

FIGURE 10 – Caractéristiques statiques d'anode de la triode 6F5 fournies par le constructeur Tung-Sol [1] et calcul de gm et ra pour un point de fonctionnement donné. Chaque courbe représente I_p (en ordonnée) en fonction de V_{pc} (en abscisse) pour une valeur de V_{gc} donnée.

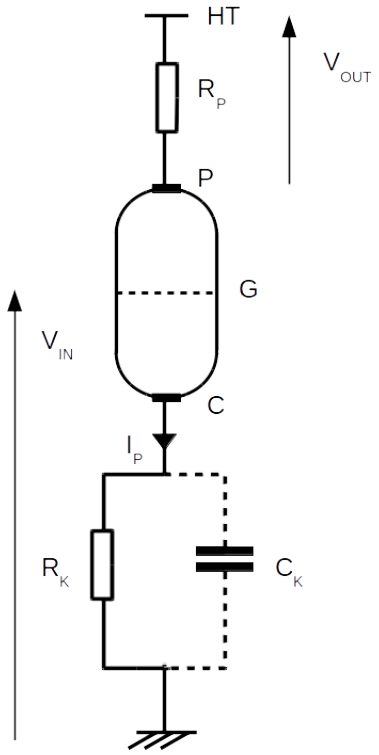

FIGURE 11 – Montage de triode à cathode commune avec résistance d'autorégulation R_k et condensateur de découplage C_k

circuit, étage par étage.

1.3.3 Oscillateurs

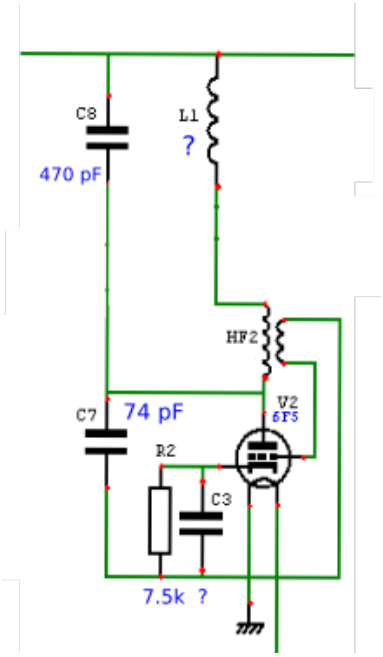

FIGURE 12 – Oscillateur à fréquence fixe de l'Onde 169 extrait du schéma fourni par le Musée de la Musique.

Les deux oscillateurs, fixe et variable, sont des oscillateurs dits Meissner [16]. Ce sont des oscillateurs à rétroaction : à la mise sous tension, V_g vaut 0 et le bruit résiduel de la grille est amplifié par la triode, filtré par un circuit LC passe-bande et réinjecté dans la grille de la triode grâce à un transformateur pour nouvelle amplification et filtrage, en boucle (figure 12). L'amplification est maximale à la fréquence de résonance puisque l'impédance de charge est maximale dans ce cas. Ces oscillateurs, lorsqu'ils sont correctement dimensionnés, permettent de générer une tension stable parfaitement [1] sinusoïdale de fréquence $f \simeq \frac{1}{2\pi\sqrt{LC}}$. La valeur de l'inductance et celle du rapport de transformation ne sont cependant pas renseignées sur le schéma de l'Onde 169 à notre disposition. Nous proposons une méthode pour les estimer en section 3.2. Les oscillateurs sont par ailleurs chacun dotés d'une résistance d'autorégulation et d'un condensateur de découplage (C_3 sur la figure 12), ainsi que d'une capacité en parallèle entre l'anode et la masse (C_7), ajoutée pour supprimer la composante continue de la tension d'anode. La triode utilisée est une 6F5 dans les deux cas, à haut facteur d'amplification (environ 100) et alimentée par une tension continue de 90V [4].

1.3.4 Mélangeur

Les transformateurs utilisés dans les oscillateurs pour les bouclages de rétroaction possèdent en outre chacun un secondaire supplémentaire. Ces deux secondaires sont mis en série en entrée du mélangeur. La tension d'entrée du mélangeur est donc la somme des tensions issues des oscillateurs, c'est une tension modulée en amplitude : $cos(a) + cos(b) = 2cos(\frac{a+b}{2})cos(\frac{a-b}{2})$. Le mélangeur est constitué d'un circuit RC en série avec une triode 6C5, chargée par un condensateur (figure 13). Le condensateur joue le rôle d'un passe-bas : en effet, son impédance $Z = \frac{1}{j\omega C}$ est maximale pour les basses fréquences et l'amplification du montage augmente avec la résistance de charge de la triode. En réalité, le circuit de charge est un passe-bande car le primaire du transformateur vers le préamplificateur est en parallèle de ce condensateur. La triode du mélangeur est un autre modèle que celui utilisé dans les oscillateurs, à facteur d'amplification plus faible (environ 20) et alimentée par une tension continue de 100V (figure 14). Telle qu'elle est montée, son point de polarisation

1. Taux de distorsion harmonique à 0.1% pour l'harmonique 2 sur les signaux simulés

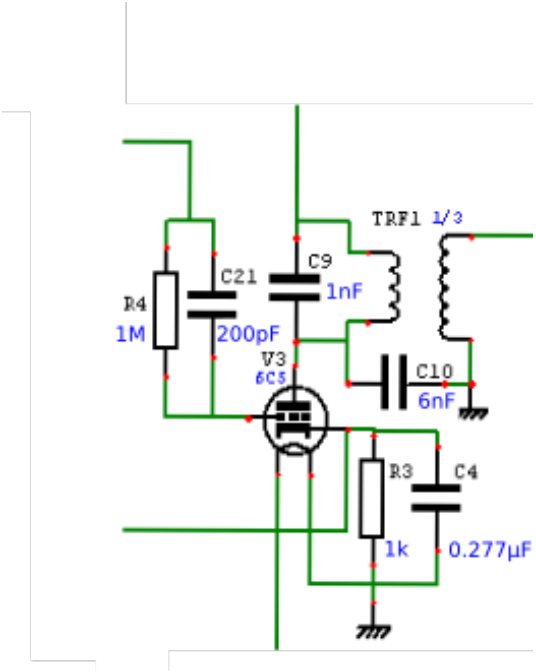

FIGURE 13 – Mélangeur de l'Onde 169 extrait du schéma fourni par le Musée de la Musique.

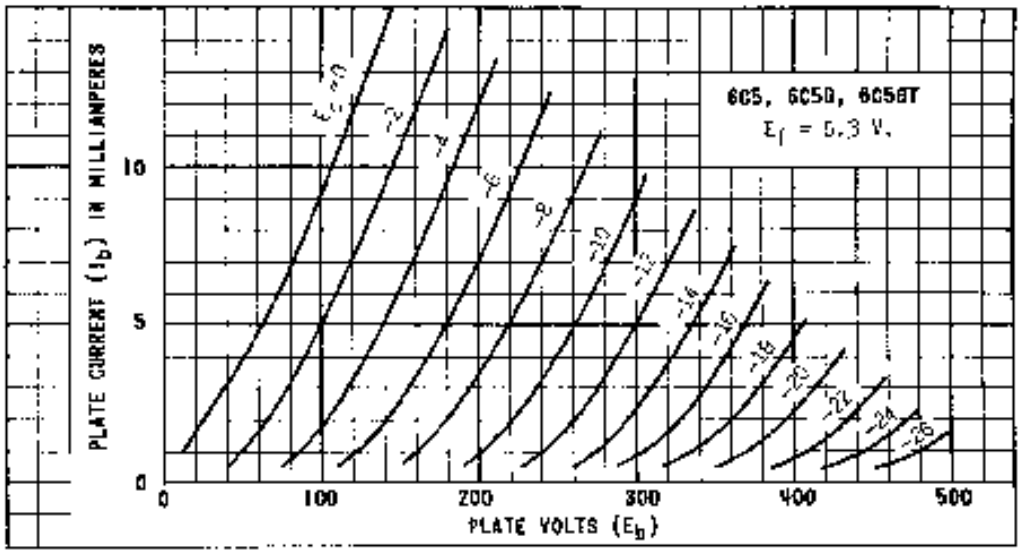

FIGURE 14 – Caractéristiques statiques d'anode de la triode 6C5 fournies en l'état par le constructeur Tung-Sol [2]. Chaque courbe représente I_p (en ordonnée) en fonction de V_{pc} (en abscisse) pour une valeur de V_{gc} donnée.

est très proche de 0 (-0.1mV, valeur obtenue par simulation en partie 2) ; la triode se comporte par conséquent comme une diode entre la grille et la cathode et on retrouve un circuit détecteur d'enveloppe classique (figure 15), dont le comportement est déterminé par la constante de temps $\tau = RC$, où τ est le temps de décharge du condensateur C_{21}. Si τ est trop faible, le résidu de porteuse est trop important. A contrario si τ est trop élevé, l'enveloppe sera distordue par rapport à l'enveloppe cible (figure 16). Si on note f_p la fréquence de la porteuse et f_m la fréquence de la modulation, on cherche en pratique à dimensionner le circuit pour avoir $10f_m \leq \frac{1}{\tau} \leq \frac{f_p}{10}$. Maurice Martenot a choisi $R_4 = 1M\Omega$ et $C_{21} = 200pF$ ce qui donne $\tau = 0.0002$. La fréquence de la porteuse est d'environ 80kHz donc on a bien $\frac{1}{\tau} = 5000 \leq \frac{f_p}{10}$; en revanche la tessiture de l'instrument s'étend du Do1 ($\simeq 32Hz$) au Si7 ($\simeq 3951Hz$) et le risque de distorsion augmente au-delà de 500Hz. Ces distorsions, en plus de celles de la triode, jouent également un rôle dans

l'identité sonore de l'instrument.

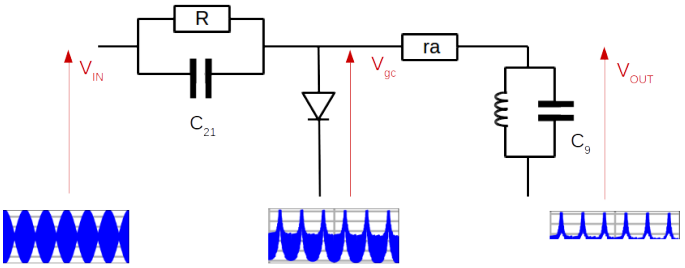

FIGURE 15 – Synoptique du mélangeur.

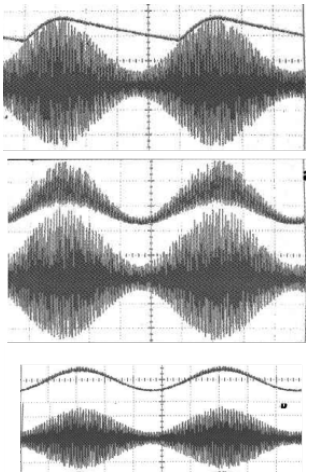

FIGURE 16 – Résultat de la démodulation pour différentes valeurs de τ. De haut en bas : τ trop élevé, τ trop faible, τ correct.

.

1.3.5 Préamplificateur

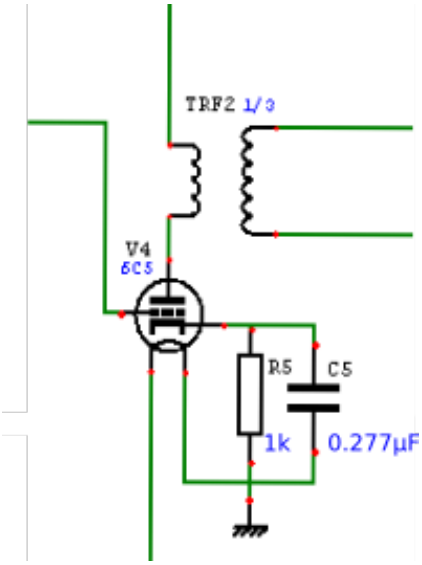

FIGURE 17 – Préamplificateur de l'Onde 169 extrait du schéma fourni par le Musée de la Musique.

Le préamplificateur est un montage classique utilisant le même modèle de triode que le mélangeur, alimentée par une tension continue de 180V (figure 17). On ne connaît cependant pas la valeur de l'inductance qui constitue le primaire du transformateur vers l'amplificateur de puissance et donc la valeur de la résistance de charge. Une méthode d'estimation est proposée dans la section 3.4.

1.3.6 Amplificateur de puissance

L'amplificateur de puissance est simplement constitué d'une triode 2A3 alimentée par une tension continue de 230V, chargée par le diffuseur et la touche d'intensité en série. Son point de fonctionnement varie avec la position de la touche d'intensité mais à pleine intensité de jeu, c'est un amplificateur de classe A qui ne colore pas le son : la charge du diffuseur étant assez faible (de l'ordre de $1.5k\Omega$ [4]), la droite de charge se situe plutôt dans la partie linéaire des courbes (figure 18 en rouge). En effet, la droite de charge représente l'ensemble des points parcourus lorsqu'une tension alternative est appliquée à la grille. Elle est tracée en reliant deux points limites du fonctionnement de la triode. Le premier point est déterminé en imaginant la lampe éteinte : l'intégralité de la tension d'alimentation est appliquée à l'anode, les coordonnées du premier point sont donc (230V, 0A). Le deuxième point est déterminé en imaginant la lampe en court-circuit : l'intégralité de la tension d'alimentation se retrouve aux bornes de la résistance de charge, les coordonnées du deuxième point sont donc (0, 230/1500) = (0V, 153 mA) puisqu'à pleine intensité la résistance de touche est nulle. Plus la charge sera élevée, plus la pente de cette droite sera faible et plus la droite traversera les zones non linéaires des courbes caractéristiques. Dans les zones de jeu pianissimo, le comportement de l'amplificateur sera donc moins linéaire.

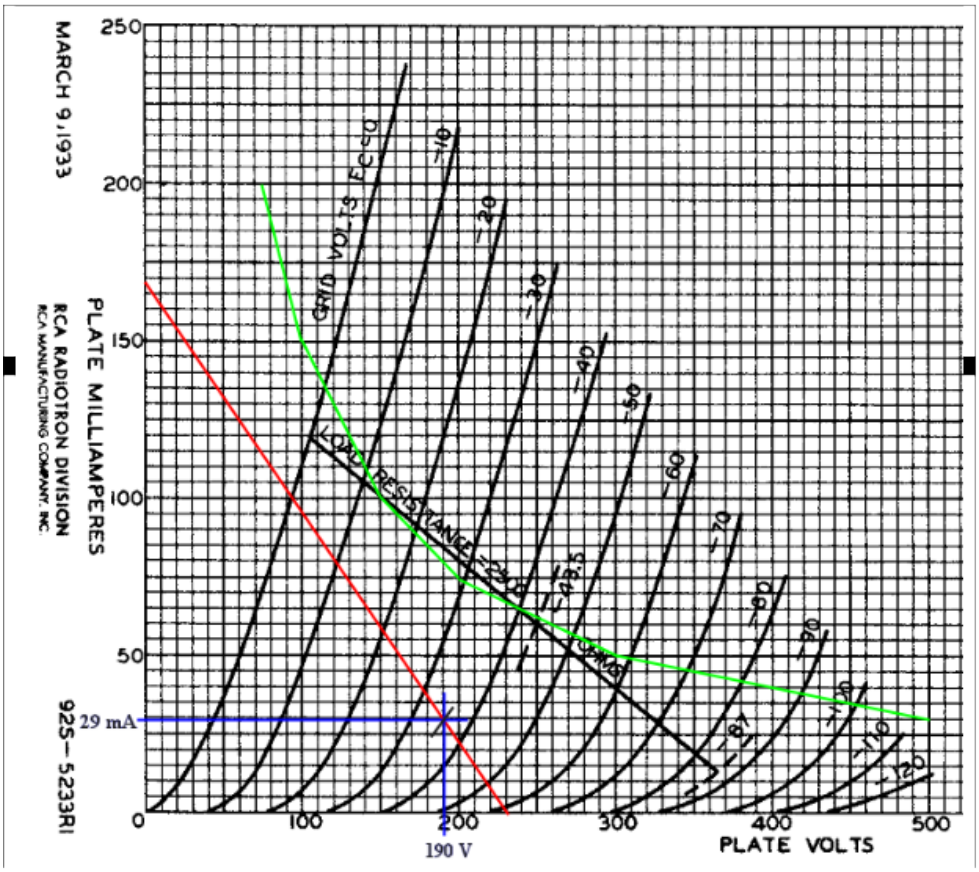

FIGURE 18 – Caractéristiques statiques d'anode de la triode 2A3 fournies par le constructeur RCA [3] et droite de charge tracée par Thierry Courrier [4] en rouge. En bleu : point de fonctionnement de la triode dans l'amplificateur de l'Onde 169 pour une résistance de touche nulle. En vert : courbe limite d'utilisation de la triode (puissance admissible).

2 Outils de modélisation physique et simulation

Dans cette section, nous présentons les outils permettant de réaliser une simulation du circuit des Ondes Martenot. Après avoir replacé les Systèmes Hamiltoniens à Port dans le contexte plus large de la modélisation physique, nous en définissons le formalisme. Nous montrons ensuite une méthode numérique adaptée à ce formalisme qui permet d'en conserver les propriétés lors d'une simulation. Enfin nous décrivons les principales fonctionnalités de la bibliothèque PyPHS [17], dédiée à la modélisation et simulation par SHP.

2.1 Contexte et motivation

L'idée sous-jacente derrière la modélisation physique est de reconstituer le dispositif producteur de son plutôt que le son lui-même, comme c'est le cas des méthodes de synthèse plus classiques telles les synthèses additive, source-filtre, granulaire ou encore la modulation de fréquence. Les systèmes mécano-acoustiques dont font partie les instruments de musique bénéficient d'outils de modélisation éprouvés comme la synthèse modale ou les guides d'ondes. Concernant la simulation de circuits électroniques, plusieurs méthodes existent comme les Wave Digital Filters [18], assez similaires aux guides d'ondes mais dont l'approche est davantage axée traitement du signal, ou Nodal DK [19], fondée sur l'inversion de matrices pour la résolution d'équations d'état dynamiques. Cependant dès lors que des non-linéarités entrent en jeu ou que les composants varient dans le temps, la stabilité de la simulation n'est pas nécessairement garantie et les calculs peuvent devenir assez considérables. Par ailleurs, ces méthodes ne permettent pas de représenter aisément un système multi-physique. Les Systèmes Hamiltoniens à Ports (on utilisera l'abréviation SHP dans la suite du document) ont été introduits en 1992 par B.M.Maschke et A.J.van der Schaft [20] et leur formalisme respecte le bilan de puissance. Ils sont multi-physiques, c'est-à-dire à même de représenter un système constitué de sous-systèmes mécaniques, électroniques ou thermiques (comme un haut-parleur par exemple), et modulaires : un ensemble de SHP connectés est toujours un SHP. Ce formalisme est donc particulièrement adapté à la modélisation d'un système complexe avec de forts couplages et des non-linéarités, pouvant être décomposé en étages élémentaires, comme c'est le cas des Ondes Martenot.

2.2 Formalisme des Systèmes Hamiltoniens à Ports

2.2.1 Principe et formulation

Des présentations détaillées des SHP sont données dans [21] et [22]. Nous proposons ici d'utiliser une formulation algébro-différentielle en représentation d'état adaptée à une approche par composants multi-physiques [23] [24]. Dans cette formulation, on peut représenter tout système physique comme un réseau de composants stockants, dissipatifs et sources ; les composants stockants sont définis par leur variable d'état x et l'expression de l'énergie stockée, donnée par la fonction $x \mapsto H(x)$, définie positive et supposée de classe $\mathcal{C}^1$; les composants dissipatifs par leur variable w et leur loi constitutive $z(w)$; les sources par les entrées u et leurs sorties associées y telles que $u^\intercal y$ est la puissance extérieure apportée au système. u, x et w dépendent du temps et peuvent être des vecteurs. L'ensemble des flux $\mathcal{F}$ et efforts $\mathcal{E}$ des composants et ports extérieurs sont couplés via une matrice anti-symétrique $S = -S^\intercal$, qui peut dépendre des variables x, w ou non :

$$\underbrace{\begin{pmatrix} \frac{dx}{dt} \\ w \\ -y \end{pmatrix}}_{\mathcal{F}\,(flux)} = \mathbf{S} \times \underbrace{\begin{pmatrix} \nabla H(x) \\ z(w) \\ u \end{pmatrix}}_{\mathcal{E}\,(efforts)} \tag{1}$$

La matrice S antisymétrique garantit la passivité du système, i.e. qu'il n'y a pas de création spontanée d'énergie. En effet, le produit scalaire des efforts et des flux, homogène à une puissance, est nul pour le système (1) puisque

$$\begin{aligned} \mathcal{E}^\intercal \mathcal{F} &= \mathcal{E}^\intercal S \mathcal{E} \\ &= (\mathcal{E}^\intercal S \mathcal{E})^\intercal \\ &= \mathcal{E}^\intercal S^\intercal \mathcal{E} \\ &= -\mathcal{E}^\intercal S \mathcal{E} \end{aligned} \tag{2}$$

or si on note $E = H(x)$ l'énergie du système, on a

$$\mathcal{E}^\intercal \mathcal{F} = \underbrace{\frac{dE}{dt}}_{\nabla H(x)^\intercal \frac{dx}{dt}} + \underbrace{P_{diss}}_{z(w)^\intercal w \geq 0} - \underbrace{P_{ext}}_{u^\intercal y} = 0 \tag{3}$$

et on retrouve bien

$$\underbrace{\frac{dE}{dt}}_{\nabla H(x)^\intercal \frac{dx}{dt}} = \underbrace{P_{ext}}_{u^\intercal y} - \underbrace{P_{diss}}_{z(w)^\intercal w \geq 0} \tag{4}$$

2.2.2 Exemple

Etudions la mise en oeuvre de ce formalisme à travers l'exemple élémentaire du RLC parallèle (figure 19).

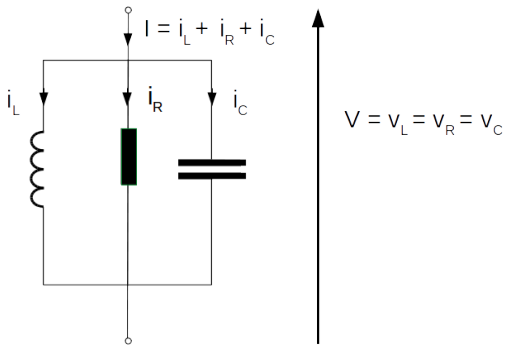

FIGURE 19 – Circuit RLC parallèle

	x	$\frac{dx}{dt}$	$H(x)$	$\nabla H(x)$
C	q	$\dot{q} = i_C$	$\frac{q^2}{2C}$	$\frac{q}{C} = v_C$
L	ϕ	$\dot{\phi} = v_L$	$\frac{\phi^2}{2L}$	$\frac{\phi}{L} = i_L$
		w		$z(w)$
R		v_R		$\frac{v_R}{R} = i_R$

Tableau 1 – Variables d'état et lois constitutives des composants d'un circuit RLC parallèle.

Le condensateur et la bobine sont des composants stockants dont les états sont donnés respectivement par la charge q et le flux magnétique ϕ ; la résistance est un composant dissipatif gouverné par la loi d'Ohm et le circuit est commandé par une source de courant I, la sortie associée est une tension V. Le tableau 1 récapitule les variables d'état et lois constitutives de ces composants, et l'application des lois de Kirchhoff en convention récepteur au noeud et à la maille constitués par la mise en parallèle des trois composants nous donne

$$\begin{pmatrix} i_C \\ v_L \\ v_R \\ -V \end{pmatrix} = \begin{pmatrix} 0 & -1 & -1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \end{pmatrix} \times \begin{pmatrix} v_C \\ i_L \\ i_R \\ I \end{pmatrix}$$

Les intensités sont des flux, les tensions sont des efforts et leurs produits des puissances ; de plus on retrouve une matrice d'interconnexion S antisymétrique, préservant ainsi le bilan de puissance. Dans ce cas très simple, la matrice S est creuse et tous ses coefficients sont constants. Mais les propriétés de ce formalisme restent les mêmes pour des systèmes non linéaires ou couplés.

2.3 Méthode numérique

La passivité du système garantie à temps continu par le formalisme des SHP peut également être préservée à temps discret lors d'une simulation grâce à l'introduction du gradient discret dans le schéma numérique. Un schéma numérique cherche à calculer l'échantillon courant $x(k+1)$ en fonction des échantillons précédents $x(k-i), i \in [0,k]$ et des données du problème. On se place ici dans le cadre d'un schéma numérique à un pas ($i=0$) ce qui nous donne

$$x(k+1) = x(k) + \delta x(k, t_e) \tag{5}$$

où t_e est le pas d'échantillonnage. Dans le cas de composants mono-variables ($H(x) = \sum_{n=1}^{N} H_n(x_n)$ où N est le nombre de composants stockants), on définit le gradient discret $[\bar{\nabla} H(x, \delta x)]_n$ par

$$[\bar{\nabla} H(x, \delta x)]_n = \begin{array}{l} \dfrac{H_n(x_n + \delta x_n) - H_n(x_n)}{\delta x_n} \text{ si } \delta x_n \neq 0 \\ H_n'(x_n) \text{ sinon} \end{array} \tag{6}$$

On retrouve la variation discrète d'énergie par différentiation en chaîne :

$$\frac{\delta E(k, t_e)}{t_e} = \bar{\nabla} H(x, \delta x(k, te))^{\intercal} \frac{\delta x(k, t_e)}{t_e} \tag{7}$$

Ici les données sont la matrice S, les lois H et z, et la séquence des entrées discrétisées u_k. La simulation s'effectue en remplaçant dans (1) $\frac{dx}{dt}$ par $\frac{\delta x(k,t_e)}{t_e}$ et $\nabla H(x)$ par $\bar{\nabla} H(x, \delta x)$ ce qui donne une équation dynamique en $\delta x(k)$ de type

$$\delta x(k, t_e) = t_e f_k(x(k), \delta x(k, t_e)) \tag{8}$$

où f_k est une fonction dépendant des données du problème à savoir H, z, u et S. Sa résolution (par la méthode de Newton-Raphson par exemple) permet ensuite de calculer $x(k+1)$ et $y(k)$. Cette méthode est consistante d'ordre 2 si les coefficients de S sont constants et d'ordre 1 sinon, c'est à dire que sur un même pas de temps, l'approximation discrète de l'accroissement $\frac{\delta x(k,t_e)}{t_e}$ tend à se rapprocher de sa valeur continue lorsque le pas de temps diminue, et stable aux erreurs, c'est à dire que l'accumulation des erreurs de consistance, des erreurs d'arrondis et d'initialisation sur la totalité des pas peut être majorée. Le détail des calculs et des démonstrations peut se trouver dans [23] et [25], chapitre 2.

2.4 La bibliothèque PyPHS

La bibliothèque open-source PyPHS [17], développée par Antoine Falaize lors de sa thèse [25], est un ensemble de fonctions dédiées à la modélisation et la simulation de systèmes physiques par SHP dans Python. L'utilisateur définit le système considéré, soit manuellement, soit à l'aide d'une netlist de composants faisant partie d'un dictionnaire (listing 1) ; les connexions des composants sont représentées par des noeuds (N_i ou # pour la masse). Le dictionnaire répertorie les com-

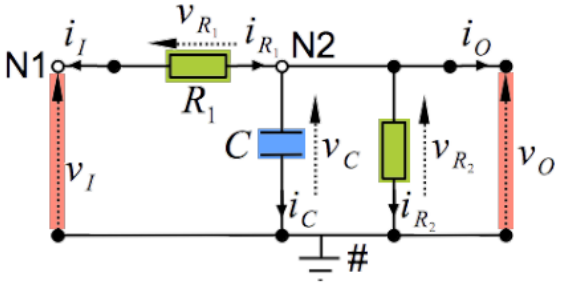

FIGURE 20 – Circuit passe-bas (image issue du tutoriel PyPHS).

posants usuels d'une même classe de systèmes physiques (électronique, mécanique, thermique ...)

et encode leurs lois constitutives sous forme d'expression symbolique. PyPHS instancie ensuite le graphe associé à la netlist et effectue une analyse de réalisabilité [2] sur ce graphe. Si le système est réalisable, il peut être mis en équations automatiquement et l'utilisateur a la possibilité de lancer une simulation grâce aux méthodes numériques de son choix dont celle précédemment décrite. PyPHS permet par ailleurs d'introduire des paramètres de contrôle, de générer du code C++ et FAUST pour la conception de plugins ainsi que de la documentation automatique. L'intégralité des fonctions disponibles et leur documentation se trouve sur https://github.com/pyphs/pyphs. Nos contributions lors de ce stage se fondent en grande partie sur les outils de cette bibliothèque et pour certains aspects, en sont une amélioration.

```
electronics.source I ('N1', '#'): type=voltage;
electronics.resistor R1 ('N1', 'N2'): R=('R1', 500.0);
electronics.capacitor C ('N2', '#'): C=('C', 1e-06);
electronics.resistor R2 ('N2', '#'): R=('R2', 1000.0);
electronics.source O ('N2', '#'): type=current;
```

Listing 1: Netlist du circuit passe-bas montré en fig.20

2. voir [26] et [27] pour une définition de la réalisabilité

Deuxième partie
Contributions

Dans cette seconde partie, nous modélisons l'Onde 169 en SHP après avoir effectué un travail d'investigation pour estimer les valeurs manquantes de ses composants en section 3, et assuré la réalisabilité du système en section 4. Nous en réalisons ensuite en section 5 une simulation à passivité garantie en temps différé d'abord dans PyPHS, puis en temps réel grâce à l'environnement JUCE.

3 Modélisation et estimation des paramètres des composants de l'Onde 169

Afin de modéliser l'Onde 169 dans PyPHS, il est nécessaire de renseigner les paramètres de tous les composants ainsi que leurs lois constitutives. Le circuit ne comprend que des résistances, bobines, condensateurs, triodes et transformateurs. En l'absence de mesures disponibles sur ce circuit ou de la possibilité d'en effectuer, leurs comportements sont donnés a priori par les lois constitutives linéaires pour les condensateurs et bobines, la loi d'Ohm pour les résistances et le modèle Norman-Koren amélioré par Cohen [28] pour les triodes. On considère en première approximation les transformateurs comme parfaits. Cependant les valeurs d'un certain nombre de composants du circuit sont manquantes, en particulier les rapports de transformation et capacités de découplage des oscillateurs, ainsi que les inductances de charge pour chaque étage. Par ailleurs, si le modèle Norman-Koren est effectivement un modèle paramétré de triode permettant d'exprimer sa loi de dissipation, ces paramètres sont à déterminer pour les triodes qui nous intéressent. Les valeurs manquantes des composants sont estimées soit par calcul lorsque cela est possible, soit par raisonnement inductif reposant sur les documents fournis par le Musée de la Musique. Dans ce cas, davantage que la valeur réelle, on donne une valeur *plausible*. Les méthodes d'estimation pour chaque composant sont détaillées ci-après, en suivant l'ordre des informations récoltées et en adoptant le point de vue d'un électronicien. En effet, au départ nous n'avons que les courbes caractéristiques des triodes à disposition. Nous commençons donc par estimer leurs paramètres. Ensuite, nous savons que les condensateurs de découplage sont essentiels pour fixer les points de fonctionnement de la triode et nous en donnons un ordre de grandeur. Après cela nous pouvons donner une plage de valeurs pour le rapport de transformation des oscillateurs puisque les paramètres de la triode sont à présent connus. Enfin les valeurs des inductances de charge de chaque étage sont estimées en fonction du rôle qu'elles jouent dans le montage.

3.1 Triodes

Le circuit de l'Onde 169 fait appel à 3 types de triodes : une 6F5 pour chaque oscillateur, deux 6C5 pour le mélangeur et le préamplificateur et une 2A3 pour l'amplificateur de puissance. Ces triodes sont proposées par plusieurs constructeurs, mais les caractéristiques d'une même triode doivent demeurer identiques d'un constructeur à l'autre. Ainsi les *datasheets* pour une même triode sont en théorie interchangeables. On dispose des caractéristiques RCA pour la 2A3 et Tung-Sol pour les autres. A partir de ces caractéristiques, on souhaite retrouver les paramètres permettant de modéliser ces triodes dans PyPHS grâce au modèle Norman-Koren amélioré [14]. Ce modèle exprime les courants d'anode I_p et de grille I_g en fonction des tensions grille-cathode V_{gc} et anode-cathode V_{pc} :

$$I_p = \begin{cases} 2E_1^{E_x}/K_g \text{ si } E_1 \geq 0 \\ 0 \text{ sinon} \end{cases} \tag{9}$$

$$I_g = \begin{cases} 0 \text{ si } V_{gc} < V_a \\ \frac{V_{gc}-V_a}{R_{gk}} \text{ sinon} \end{cases} \tag{10}$$

avec

$$E_1 = \frac{V_{pc}}{K_p} \ln(1 + \exp(K_p(\frac{1}{\mu} + \frac{V_{gc}+V_{ct}}{\sqrt{K_{vb}+V_{pc}^2}}))))$$

ce qui rend possible une modélisation de la triode par SHP en tant que composant dissipatif bivariable. Afin d'implémenter ce modèle, on doit déterminer $\theta = (\mu, E_x, K_g, K_p, K_{vb}, V_{ct}, V_a, R_{gk})$. Pour chaque triode, les caractéristiques statiques d'anode (I_p en fonction de V_{pc} pour plusieurs valeurs de V_{gc}) sont donc scannées puis retracées à l'aide d'un logiciel de dessin vectoriel. La différence entre les ordonnées ainsi relevées et l'évaluation des abscisses par le modèle Norman-Koren fait ensuite l'objet d'une minimisation par moindres carrés, conjointe sur plusieurs valeurs de V_{gc}. Les figures 21, 22 et 23 montrent le résultat de cette minimisation et le tableau 2 indique les valeurs des paramètres ainsi estimés. On ne dispose par contre pas de données pour le courant de grille I_g, qui ne fait pas partie du modèle Norman-Koren original [12] mais qu'on souhaite ne pas négliger. Les valeurs de V_a et R_{gk} sont donc choisies arbitrairement en attendant d'être remplacées par des valeurs issues de mesures, objet d'un autre stage en cours. On prend $V_a = 0.33V$ et $R_{gk} = 1.3k\Omega$ en se fondant sur les résultats de [14] bien qu'il ne s'agisse pas du même type de triode. Pour ce jeu de paramètres θ, on vérifie que pour $w = \begin{pmatrix} V_{pc} \\ V_{gc} \end{pmatrix}$ et $z_\theta(w) = \begin{pmatrix} I_p \\ I_g \end{pmatrix}$, on a bien $P_{diss} = z_\theta(w)^\intercal w \geq 0$.

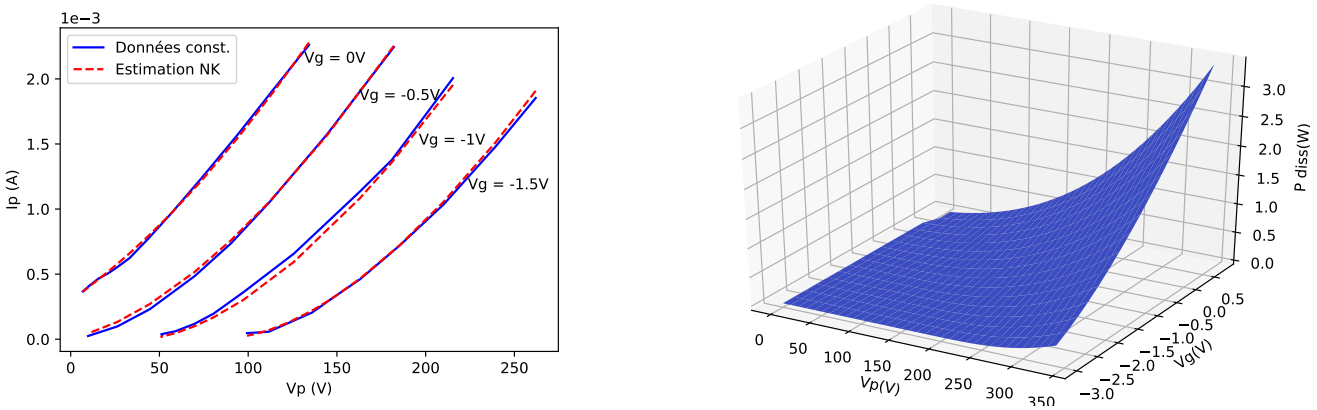

FIGURE 21 – Comparaison entre les données constructeur et les données estimées, et puissance dissipée pour la triode 6F5.

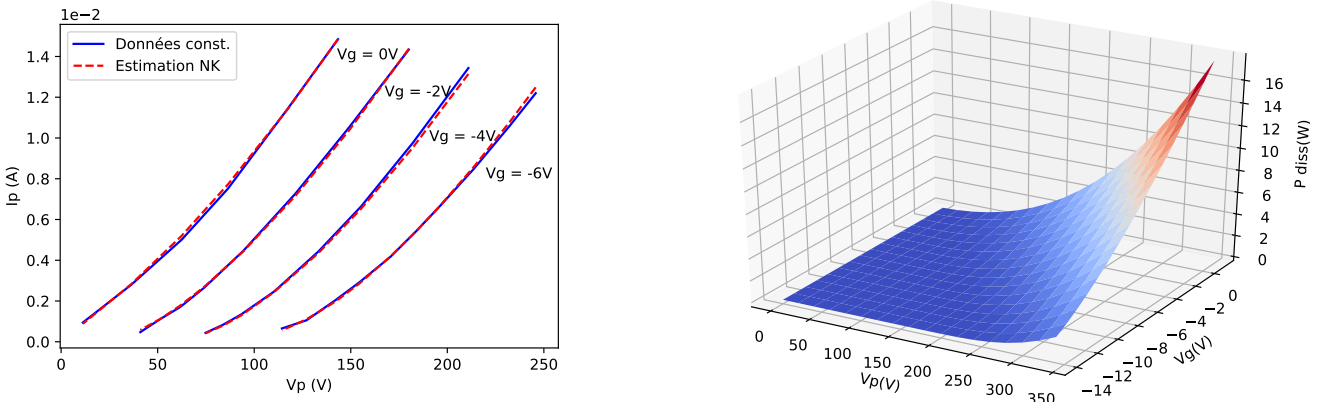

FIGURE 22 – Comparaison entre les données constructeur et les données estimées, et puissance dissipée pour la triode 6C5.

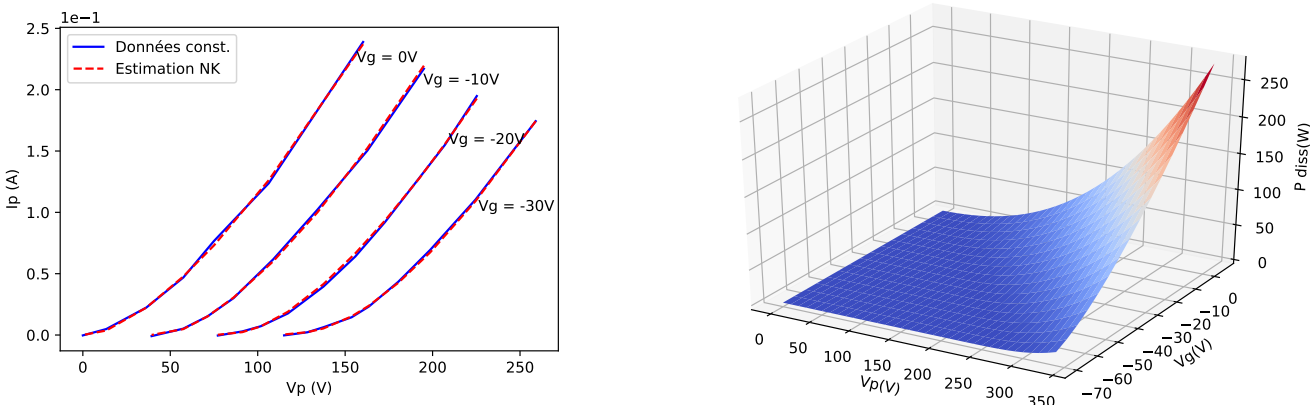

FIGURE 23 – Comparaison entre les données constructeur et les données estimées, et puissance dissipée pour la triode 2A3.

	μ	E_x	K_g	K_p	K_{vb}	V_{ct}
6F5	98	1.6	2614	905	1.87	0.5
6C5	20	1.5	2837	138	89	0.8
2A3	4.3	1.5	1685	43	102	-1.2

Tableau 2 – Paramètres estimés du modèle Norman-Koren pour les triodes 6F5, 6C5 et 2A3.

3.2 Capacités de découplage des oscillateurs

Pour rappel de la section 1.3.2, dans la plupart des montages à triode avec résistance d'autorégulation, un condensateur de découplage C_k est placé en parallèle de cette résistance d'autorégulation, entre la cathode et la masse. Ce condensateur de découplage joue un rôle essentiel dans le comportement de la triode. Or les valeurs des capacités de découplage des oscillateurs ne sont pas renseignées sur le schéma électrique de l'Onde 169. Elle doivent donc être estimées. C_k fixe le potentiel de cathode et agit sur la réponse en fréquence du montage ; en effet un condensateur et une résistance en parallèle forment un filtre passe-haut en courant-courant de fréquence de coupure $f_c = \frac{1}{2\pi RC}$ et l'amplification favorise les fréquences supérieures à f_c. On peut imaginer rechercher dans ce cas la plus basse fréquence de coupure techniquement possible afin de favoriser tout le spectre. Cependant les condensateurs de capacité élevée (plus de $1\mu F$) étant à la fois forts coûteux et peu fiables à l'époque de Maurice Martenot [4], on choisit plutôt de reproduire le compromis qu'il a probablement effectué. L'oscillateur génère une tension de fréquence $f_{osc} \simeq 80kHz$. On souhaite que cette tension soit stable, et donc que le découplage soit maximal à cette fréquence. On doit par conséquent avoir $f_{osc} \geq 10 f_c$ soit $C \geq \frac{10}{2\pi R f_{osc}} = \frac{10}{2\pi \times 7500 \times 80000} = 3nF$, pour une résistance d'autorégulation valant $7.5k\Omega$. La figure 24 montre l'allure de la tension en sortie d'une triode 6F5, utilisée dans les oscillateurs, pour plusieurs valeurs de C_k lorsqu'une tension d'amplitude $1V$ à la fréquence f_{osc} est appliquée à sa grille. Ces courbes montrent que conformément aux prédictions, plus C_k est élevée, meilleure est l'amplification et plus court est le transitoire. On choisit $C_k = 0.22\mu F$ qui est à la fois une valeur de capacité standard et donne une fréquence de coupure égale à 96Hz, suffisante pour notre oscillateur.

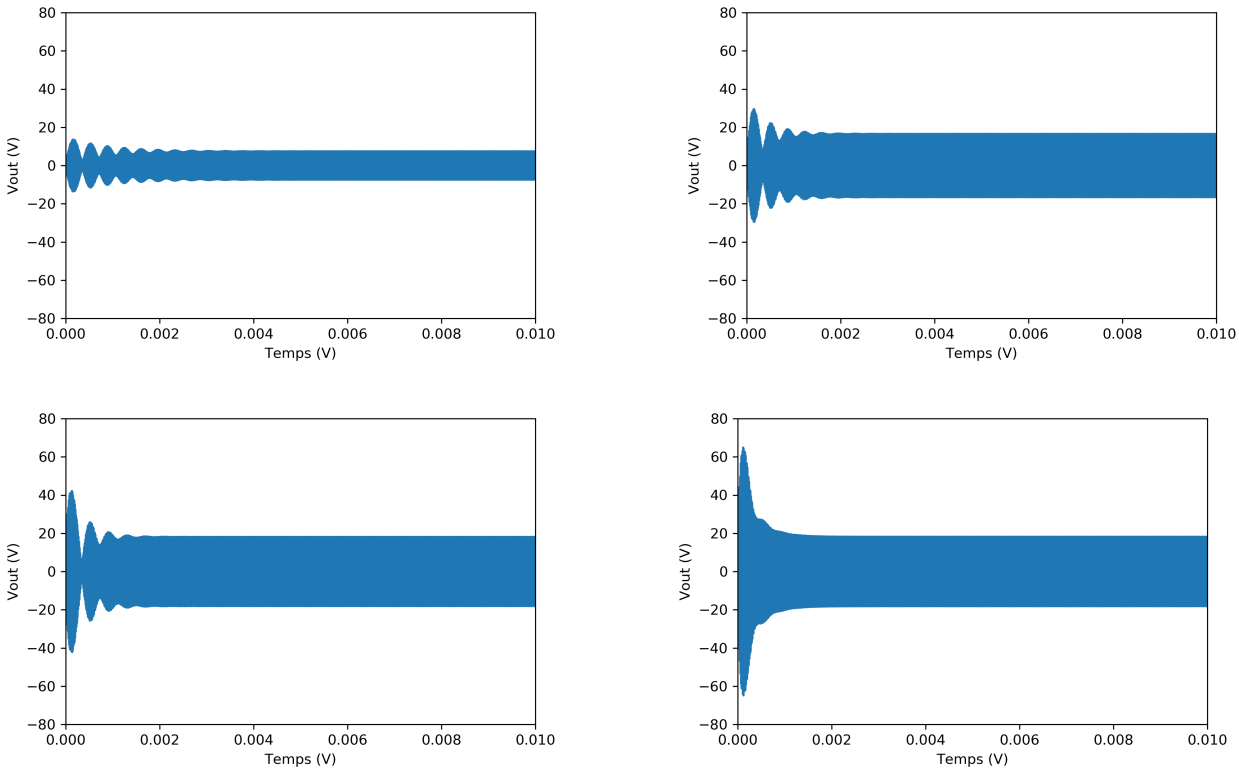

FIGURE 24 – Tensions simulées en sortie d'une triode 6F5 chargée par un circuit LC accordé sur $f_{osc} = 80kHz$ pour une tension d'entrée sinusoïdale de fréquence f_{osc} et d'amplitude $1V$, selon la valeur de la capacité de découplage. De gauche à droite et de haut en bas : $C_k = 0.2nF, 2nF, 20nF$ et $0.2\mu F$.

3.3 Transformateurs des oscillateurs

Pour rappel de la section 1.3.3, les oscillateurs de l'Onde 169 sont des oscillateurs dits Meissner [16]. La figure 25 en donne le synoptique, avec Z l'impédance de charge de la triode, ra la résistance d'anode de la triode et M le rapport de transformation entre la tension aux bornes de Z et la tension de grille. Le dimensionnement de M conditionne la naissance d'oscillations à partir d'un bruit de

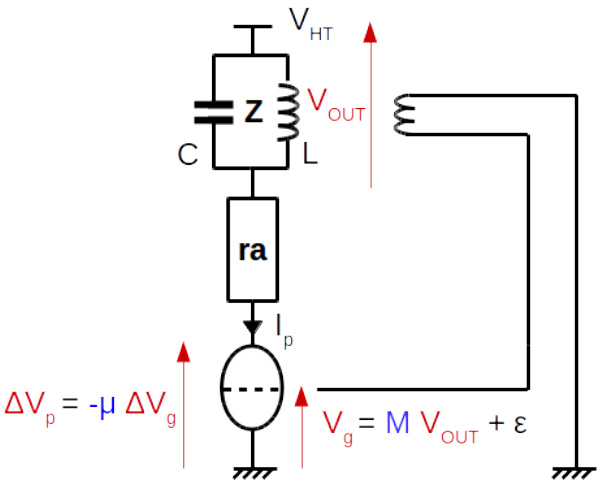

FIGURE 25 – Synoptique de l'oscillateur Meissner en petits signaux.

tension sur la grille. Or ce rapport n'est pas renseigné sur le schéma électrique de l'Onde 169. Pour le déterminer, nous allons l'exprimer en fonction des données dont nous disposons, à savoir

les caractéristiques de la triode et l'impédance de charge. On linéarise le problème autour d'un point de fonctionnement V_{eq} (qui diffère pour chaque tension observée) en se plaçant dans le cadre d'un modèle petits signaux et en considérant que les coefficients de la triode sont constants au voisinage de ce point. En première approximation, on modélise par ailleurs le système en circuit ouvert (signaux peu perturbés par le 2e secondaire du transformateur vers le mélangeur). Pour chaque tension V, on note $\Delta V = V - V_{eq}$. On note μ le facteur d'amplification de la triode et ϵ le bruit de grille de démarrage. La charge est constituée par un circuit LC parallèle, on a donc dans le domaine de Laplace

$$Z = \frac{sL}{1 + s^2LC} \tag{11}$$

Par ailleurs

$$V_{out} = \frac{Z}{Z + ra}(V_{HT} - V_p) \tag{12}$$

ce qui implique

$$\Delta V_{out} = -\frac{Z}{Z + ra}\Delta V_p \tag{13}$$

puisque la tension d'alimentation est constante. Par définition de l'amplification de la triode et du rapport de transformation, l'équation (13) donne

$$\Delta V_{out} = \frac{Z}{Z + ra}\mu\Delta V_g = \frac{Z}{Z + ra}\mu(M\Delta V_{out} + \epsilon) \tag{14}$$

ce qui implique

$$\frac{\Delta V_{out}}{\epsilon} = \frac{Z\mu}{Z(1 - \mu M) + ra} \tag{15}$$

Les pôles de la fonction de transfert (15) sont obtenus en calculant

$$\begin{aligned} 0 &= Z(1 - \mu M) + ra \\ &= \frac{sL}{1 + s^2LC}(1 - \mu M) + ra \\ &= s^2LCra + sL(1 - \mu M) + ra \\ &= s^2 + s\frac{1 - \mu M}{raC} + \omega_0^2 \end{aligned} \tag{16}$$

avec $\omega_0 = \frac{1}{\sqrt{LC}}$. On reconnaît l'équation caractéristique d'un oscillateur harmonique dont les racines sont de la forme $\frac{-B \pm \sqrt{\Delta}}{2}$ avec $B = \frac{1-\mu M}{raC}$ et $\Delta = B^2 - 4\omega_0^2$. On recherche les conditions de naissance de l'oscillation, donc un pôle complexe à partie réelle positive, ce qui implique $\Delta < 0$ et $B \leq 0$. Ces deux conditions nous donnent un encadrement sur M :

$$\begin{aligned} B \leq 0 \text{ et } \Delta < 0 &\Rightarrow B^2 < 4\omega_0^2 \\ &\Rightarrow -2\omega_0 < B \leq 0 \\ &\Rightarrow -2\omega_0 < \frac{1 - \mu M}{raC} \leq 0 \\ &\Rightarrow \frac{1}{\mu} \leq M < \frac{1}{\mu} + \frac{2\omega_0 raC}{\mu} \\ &\Rightarrow \frac{1}{\mu} \leq M < \frac{1}{\mu} + \frac{2\omega_0 C}{gm} \end{aligned} \tag{17}$$

La plage de valeurs de M pour qu'il y ait apparition d'oscillations en circuit ouvert est donc

$$\frac{1}{\mu} \leq M < \frac{1}{\mu} + \frac{2\omega_0 C}{gm} \tag{18}$$

La figure 26 illustre la trajectoire parcourue par les racines (sans leurs complexes conjugués) de l'équation caractéristique selon la valeur de M, dans le plan complexe. Plus M sera choisi proche de $1/\mu$, plus la solution tendra vers une solution imaginaire pure et plus l'oscillation sera stable.

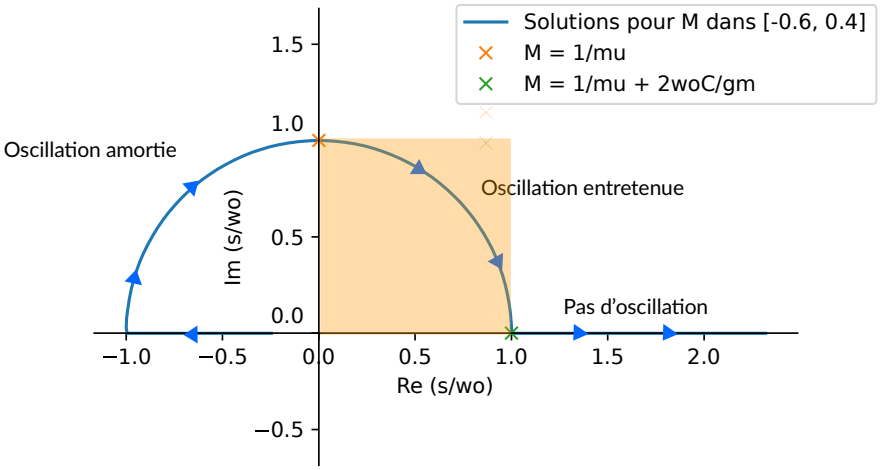

FIGURE 26 – Solutions de l'équation caractéristique de l'oscillateur pour plusieurs valeurs de M dans le plan complexe.

3.4 Inductances de charge

3.4.1 Inductance de l'oscillateur fixe

On choisit une fréquence pour le circuit LC de l'oscillateur fixe $f = 80kHz$. Dans ce cas, $L = \frac{1}{C_8(2\pi f)^2} = \frac{1}{470e^{-12}(2\pi 80000)^2} = 8.4mH$.

3.4.2 Inductance du mélangeur

On souhaite déterminer la valeur de l'inductance qui constitue le primaire du transformateur vers le préamplificateur. Cette inductance est en parallèle avec le condensateur C_9, formant un filtre passe-bande. La tessiture des Ondes Martenot s'étend de $30Hz$ à $4000Hz$ environ. On choisit donc comme fréquence de résonance la fréquence médiane (en bandes d'octave), soit environ $500Hz$, ce qui donne $L = \frac{1}{C_9(2\pi f)^2} = \frac{1}{1e^{-9}(2\pi 500)^2} = 101H$.

Par ailleurs, pour cette valeur, le facteur de qualité du filtre vaut $Q = ra\sqrt{\frac{C_9}{L}} = 15000 \times \sqrt{\frac{1e^{-9}}{101}} = 0.04$, et la bande passante est donc de $500/0.04 = 12.5kHz$.

3.4.3 Inductance du préamplificateur

La charge du préamplificateur n'est constituée que du primaire du transformateur vers l'amplificateur de puissance. Cependant une inductance seule n'est pas réalisable, l'inductance et la triode étant en série et toutes les deux contrôlées en tension (voir la section suivante pour une description du conflit de réalisabilité). On se place donc dans le cadre d'un modèle petits signaux pour faire apparaître la résistance interne de la triode ra, considérée constante autour du point de fonctionnement et de valeur $10k\Omega$ (figure 27). En notant Z_L l'impédance de l'inductance et en reprenant les notations de la section 3.3, on a

$$\begin{aligned} \Delta V_{out} &= -\frac{Z_L}{Z_L + ra}\Delta V_p \\ &= -\frac{Z_L}{Z_L + ra} ra I_p \end{aligned} \tag{19}$$

Le montage se comporte donc en réalité comme si ra et Z_L étaient en parallèle. On a par conséquent une impédance de charge équivalente $Z_{eq} = \frac{Z_L ra}{Z_L + ra}$ qui joue le rôle d'un passe-haut puisque $Z_{eq} = \frac{rasL}{ra+sL}$ dans le domaine de Laplace tend vers ra dans les hautes fréquences, 0 dans les basses

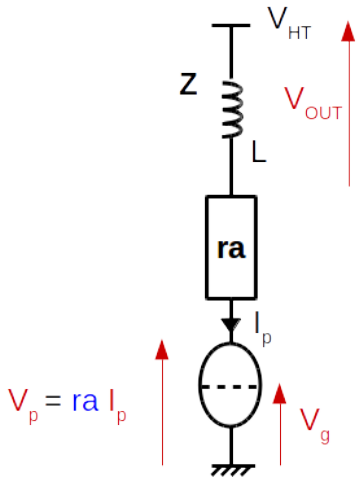

FIGURE 27 – Synoptique du préamplificateur en petits signaux.

fréquences, et l'amplification augmente avec la charge de la triode. On choisit une fréquence de coupure $f_c = 20Hz$, et on trouve $L = \frac{ra}{2\pi f_c} = \frac{10000}{2\pi \times 20} = 79H$.

4 Circuit complet et résolution des conflits de réalisabilité

Nous disposons à présent de tous les éléments nécessaires à la modélisation des composants de l'Onde 169 : le dictionnaire électronique de PyPHS encode les lois constitutives des résistances, condensateurs, bobines et triodes à partir de leurs paramètres et nous avons estimé dans la section précédente les paramètres manquants de tous les composants. Afin de modéliser l'ensemble du circuit, nous devons à présent l'écrire sous forme de graphe, cependant cette mise en forme fait apparaître des conflits dits de réalisabilité. Après avoir décrit ces conflits, on propose une méthode de résolution partielle pour les composants mono-variables. Une implémentation dans PyPHS pour les composants mono-variables avec lois constitutives affines par morceaux est ensuite suggérée. Ce travail a fait l'objet d'un article accepté pour présentation à la 145e Convention AES.

4.1 Présentation du problème

En théorie des systèmes, on dit qu'un système est réalisable si à partir d'une relation entrée-sortie, on est capable de construire une représentation d'états de ce système [27]. Ainsi, la représentation d'un circuit électronique dans la formulation (1) n'est pas toujours directement possible. L'exemple classique est celui de condensateurs mis en parallèle (qu'on retrouve notamment dans l'oscillateur variable de l'Onde 169 et ses contrôles) qui forment du point de vue physique un unique composant équivalent. Le problème est bien connu et résolu par les électroniciens dans le cas linéaire : le composant équivalent est immédiatement obtenu par calcul d'impédance. Prenons le système en figure 28 et notons C le composant équivalent. $Z_C = \frac{Z_A Z_B}{Z_A + Z_B}$ conduit à $C_C = C_A + C_B$, avec $Z = \frac{1}{j\omega C}$. Pour deux bobines en série, on obtiendrait le même résultat avec $Z_C = Z_A + Z_B$ et $Z = j\omega L$. En revanche, dans le cas de composants non linéaires ou variables dans le temps, ce calcul d'impédance n'est plus approprié. Les deux condensateurs sont caractérisés par leur état de charge $x_{A,B} = q_{A,B}$ et contrôlés en courant ($\frac{dx_{A,B}}{dt} = \dot{q}_{A,B} = i_{A,B}$). Lorsqu'on les connecte en parallèle, leurs tensions sont égales et on ne peut trouver de représentation algébro-différentielle du système par SHP :

$$\begin{pmatrix} i_A \\ i_B \\ -V \end{pmatrix} = ? \times \begin{pmatrix} v_A \\ v_B \\ I \end{pmatrix}$$

Ce type de conflit apparaîtrait également avec deux bobines en série. Du point de vue général, le problème d'analyse de réalisabilité dépend du type de connexion et de la nature du contrôle du composant considéré (courant ou tension). L'analyse de réalisabilité est disponible dans PyPHS grâce à un critère [25] déterminant la réalisabilité d'un système d'après sa représentation en graphe. Dans cette représentation, un noeud est constitué par la connexion de deux composants et une

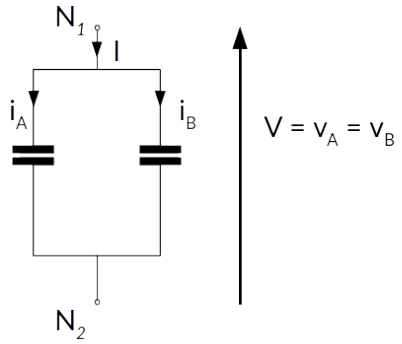

FIGURE 28 – Condensateurs en parallèle.

branche est constituée par le composant lui-même. Par convention, l'orientation de la branche est donnée par le sens du courant qui la parcourt. Le critère est issu de la matrice d'incidence Γ définie comme suit :

$$\Gamma_{n,b} = \begin{cases} -1 \textit{ si la branche } b \textit{ sort du noeud } n \\ 1 \textit{ si la branche } b \textit{ entre dans le noeud } n \\ 0 \textit{ sinon} \end{cases} \tag{20}$$

Les branches sont préalablement triées entre composants contrôlés en courant et composants contrôlés en tension. On peut ensuite extraire de Γ une matrice γ des composants uniquement contrôlés en courant. Le critère établit que si γ est inversible, alors le système est réalisable par SHP. En effet, pour l'exemple des condensateurs en parallèle, la matrice γ donne

$$\gamma = \begin{array}{c} \begin{matrix} C_A & C_B \end{matrix} \\ \begin{pmatrix} -1 & -1 \\ 1 & 1 \end{pmatrix} \end{array} \begin{matrix} N_1 \\ N_2 \end{matrix} \tag{21}$$

Son déterminant étant égal à 0, γ n'est pas inversible et le conflit de réalisabilité est confirmé.

4.2 Méthode de résolution formelle pour les composants stockants monovariables

Dans le travail qui suit, nous proposons une méthode de substitution de l'ensemble des composants en conflit par un composant équivalent pour une classe de composants stockants, monovariables et non linéaires. L'objectif est alors de déterminer la fonction d'énergie du composant équivalent, sous réserve que les fonctions d'énergie des composants à remplacer satisfassent certaines hypothèses :
(i) H est définie positive et $\mathcal{C}^1$
(ii) H' est strictement croissante
(iii) $H'(0) = 0$
Soient $q_A \mapsto H_A(q_A)$ et $q_B \mapsto H_B(q_B)$ les fonctions d'énergie de deux condensateurs A et B en parallèle, satisfaisant les hypothèses (i) à (iii). On propose de déterminer $q_C \mapsto H_C(q_C)$ décrivant le composant C équivalent à $A//B$ et de montrer qu'elle satisfait les hypothèses (i) à (iii). Pour cela, on exprime q_A et q_B en fonction de v_A et v_B. On introduit ensuite l'état q_C de charge totale en fonction de la tension $v_A = v_B = v_C$, ce qui nous permet d'exprimer ensuite v_C en fonction de q_C. On tire de ces expressions la fonction d'énergie totale $q_C \mapsto H_C(q_C)$ et on vérifie que cette description de C fournit bien un composant équivalent satisfaisant (i) à (iii).

H_A et H_B sont les fonctions d'énergie des composants A et B et on note leurs gradients H'_A et H'_B. D'après nos hypothèses, H'_A et H'_B sont bijectives ce qui garantit l'existence de H'^{-1}_A et H'^{-1}_B. Par définition,

$$v_A = H'_A(q_A) \tag{22}$$

et

$$v_B = H'_B(q_B) \tag{23}$$

ce qui donne

$$q_A = H_A'^{-1}(v_A) \tag{24}$$

et

$$q_B = H_B'^{-1}(v_B) \tag{25}$$

Les composants A et B sont extensifs donc

$$q_C = q_A + q_B \tag{26}$$

et par construction,

$$v_C = v_A = v_B \tag{27}$$

On note

$$\tilde{H}_A(v_A) = (H_A \circ H_A'^{-1})(v_A) = (H_A \circ H_A'^{-1})(v_C) \tag{28}$$

et

$$\tilde{H}_B(v_B) = (H_B \circ H_B'^{-1})(v_B) = (H_B \circ H_B'^{-1})(v_C) \tag{29}$$

L'énergie du composant équivalent est la somme des énergies des composants A et B et on définit $\tilde{H}_C(v_C)$ comme :

$$\begin{aligned} \tilde{H}_C(v_C) &= \tilde{H}_A(v_C) + \tilde{H}_B(v_C) \\ &= (\tilde{H}_A + \tilde{H}_B)(v_C) \end{aligned} \tag{30}$$

Les équations (26) et (27) donnent

$$\begin{aligned} q_C &= H_A'^{-1}(u_A) + H_B'^{-1}(v_B) \\ &= H_A'^{-1}(v_C) + H_B'^{-1}(v_C) \\ &= (H_A'^{-1} + H_B'^{-1})(v_C) \end{aligned} \tag{31}$$

$H_A'^{-1}$ et $H_B'^{-1}$ sont bijectives, leur somme est donc aussi bijective. Les équations (31) et (30) donnent alors

$$\begin{aligned} \tilde{H}_C(v_C) &= (H_A \circ H_A'^{-1} + H_B \circ H_B'^{-1})(v_C) \\ &= (H_A \circ H_A'^{-1} + H_B \circ H_B'^{-1}) \circ (H_A'^{-1} + H_B'^{-1})^{-1}(q_C) \end{aligned} \tag{32}$$

On définit donc $H_C(q_C) = (H_A \circ H_A'^{-1} + H_B \circ H_B'^{-1}) \circ (H_A'^{-1} + H_B'^{-1})^{-1}(q_C)$ ce qu'on peut généraliser pour N composants connectés :

$$H_C(q_C) = [\sum_{i=1}^{N} H_i \circ H_i'^{-1}] \circ [\sum_{i=1}^{N} H_i'^{-1}]^{-1}(q_C) \tag{33}$$

Par ailleurs, on peut calculer l'état des composants originaux à tout instant :

$$\begin{aligned} q_i &= H_i'^{-1}(v_C) \\ &= (H_i'^{-1} \circ H_C'^{-1})(q_C) \end{aligned} \tag{34}$$

4.3 Cas des lois affines par morceaux et intégration dans PyPHS

L'utilisation dans PyPHS de la méthode présentée précédemment n'est possible que pour une classe de composants dont lois constitutives $f : x \mapsto H'(x)$ sont numériquement intégrables, inversibles, sommables et composables. On considère alors les lois constitutives affines par morceaux $\mathcal{C}^0$, qui remplissent ces conditions, et les H_i' ne sont plus définies de façon analytique mais par une collection de points triés $(X, Y) \in \mathbb{I}_N^2$, $N > 0$ avec $\mathbb{I}_N = \{(X_0, ..., X_N) \in \mathbb{R}^{N+1}$ tels que $X_0 < X_1 < ... < X_N\}$:
(i) si $X_n \leq x < X_{n+1}$ avec $0 \leq n < N$, alors $f_{X,Y}(x) = Y_n + \frac{Y_{n+1}-Y_n}{X_{n+1}-X_n}(x - X_n)$
(ii) si $x < X_0$, alors $f_{X,Y}(x) = Y_0 + \frac{Y_1-Y_0}{X_1-X_0}(x - X_0)$
(iii) si $x > X_N$, alors $f_{X,Y}(x) = Y_{N-1} + \frac{Y_N-Y_{N-1}}{X_N-X_{N-1}}(x - X_N)$
(iv) il existe n tel que $0 \leq n < N$ et $X_n = Y_n = 0$.
Pour cette classe de fonctions, les lois inverse, somme et composition sont des lois internes :

$f_{X,Y}^{-1} = f_{Y,X}$
$f_{A,B} + f_{C,D} = f_{E,F}$ avec $E = sort(A, C)$ et $F = f_{A,B}(E) + f_{C,D}(E)$
$f_{C,D} \circ f_{A,B} = f_{E,F}$ avec $E = sort(A, f_{A,B}^{-1}(C))$ et $F = [f_{C,D} \circ f_{A,B}](E)$
$sort(A, B)$ étant défini comme l'ensemble de tous les éléments de A et B, triés. Les H_i sont calculées à partir des H_i' en les intégrant par la méthode des trapèzes ($x \mapsto H(x) = \int_0^x f(u)du$) et la fonction d'énergie du composant équivalent est donnée par l'équation (33). L'algorithme d'analyse de graphe et remplacement automatique est donné par l'algorithme (1). La totalité du code est donnée en annexe A.

```
Divide system graph between parallel and series subgraphs;
for each subgraph do
    if subgraph is parallel then
        for each storage component do
            if component is flow-controlled then
                Store component dataset;
            end
        end
        if there are at least two stored components then
            Compute equivalent H;
            Remove stored components from subgraph;
            Replace with new component dataset;
        end
    end
    if subgraph is series then
        for each storage component do
            if component is effort-controlled then
                Store component dataset;
            end
        end
        if there are at least two adjacent stored components then
            Compute equivalent H;
            Remove stored components from subgraph;
            Replace with new component dataset;
        end
    end
end
```

Algorithm 1: Component replacement algorithm based on a dataset (X, Y)$\in \mathbb{I}_N^2$ and an interpolating function $f_{X,Y} \in \mathcal{C}^0(\mathbb{R}, \mathbb{R})$

4.4 Exemple test

L'algorithme est testé sur un système constitué de trois condensateurs non linéaires en parallèle. On génère les données de trois fonctions affines par morceaux représentant H_1', H_2' et H_3' en respectant la contrainte $H_1'(X_1) = H_2'(X_2) = H_3'(X_3)$. On choisit un ensemble X_1 de valeurs q régulièrement espacées de $0.5.10^{-7}$ dans l'intervalle $[0, 10^{-6}]$ (voir le tableau 3). La fonction d'énergie du composant équivalent déterminée analytiquement est $H(x) = \frac{x^4}{4C^3}$ avec $C = \sqrt[3]{C_1} + \sqrt[3]{C_2} + \sqrt[3]{C_3}$. En effet, si on note x la charge du système équivalent, on doit avoir $H'(x) = H_1'(x_1)$ et on a

$$\begin{aligned} x &= x_1 + x_2 + x_3 \\ &= x_1 + x_1 \sqrt[3]{\frac{C_2}{C_1}} + x_1 \sqrt[3]{\frac{C_3}{C_1}} \\ &= \frac{x_1}{\sqrt[3]{C_1}} (\sqrt[3]{C_1} + \sqrt[3]{C_2} + \sqrt[3]{C_3}) \\ &= \frac{x_1}{\sqrt[3]{C_1}} C \end{aligned} \tag{35}$$

ce qui implique

$$\frac{x^3}{C^3} = \frac{x_1^3}{C_1} = H_1'(x_1) \tag{36}$$

D'où $H'(x) = \frac{x^3}{C^3}$ et $H(x) = \frac{x^4}{4C^3}$. La figure 29 montre la comparaison entre cette loi obtenue analytiquement et la loi obtenue automatiquement par l'algorithme (1) et l'équation (33).
On effectue ensuite la simulation dans PyPHS d'un circuit RC série, où dans un cas le condensateur

	C_1	C_2	C_3
capacité	$440pF$	$47pF$	$27pF$
$\mathbf{X} = \{\mathbf{q}\}$	X_1	$X_2 = X_1 \sqrt[3]{\frac{C_2}{C_1}}$	$X_3 = X_1 \sqrt[3]{\frac{C_3}{C_1}}$
$\mathbf{Y} = \mathbf{H}'(\mathbf{X})$	$\frac{X_1^3}{C_1}$	$\frac{X_2^3}{C_2}$	$\frac{X_3^3}{C_3}$

Tableau 3 – Données générées pour trois condensateurs non linéaires en parallèle.

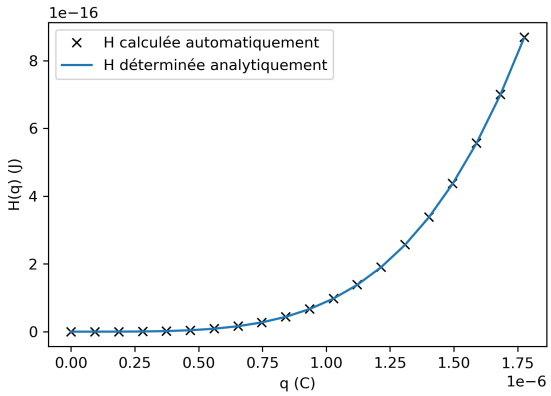

FIGURE 29 – Comparaison entre la fonction d'énergie équivalente déterminée analytiquement et la fonction d'énergie calculée automatiquement pour trois condensateurs en parallèle.

est formé des trois condensateurs précédents en parallèle, et dans l'autre cas le condensateur est unique de valeur équivalente. Les résultats de cette simulation sont montrés en figure 30. Cet algorithme permet donc de calculer avec exactitude une fonction d'énergie équivalente, et de simuler des systèmes qui n'étaient pas réalisables autrement.

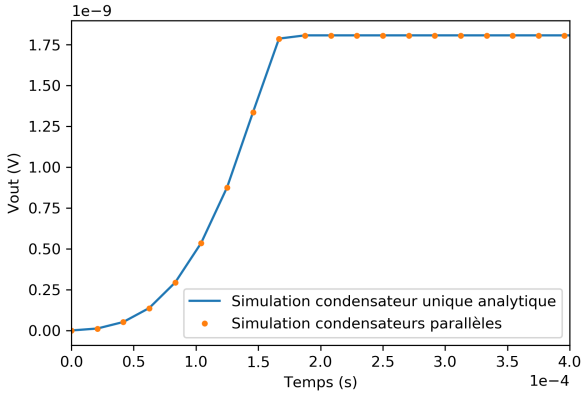

FIGURE 30 – Comparaison entre la simulation de trois condensateurs non linéaires en parallèle et la simulation d'un condensateur unique équivalent.

5 Simulation et résultats

Le travail précédent nous a permis de réunir les éléments nécessaires à la simulation du circuit de l'Onde 169. Le résultat de la simulation de l'oscillateur en particulier nous autorise ensuite à considérer dans la suite les étages 1 et 2 comme un simple générateur de tension modulée et utiliser un modèle de signal plutôt qu'un modèle physique. Le temps de calcul ainsi économisé rend possible une simulation du circuit simplifié en temps réel en C++.

5.1 Temps différé avec implémentation dans PyPHS

5.1.1 Simulation de l'oscillateur fixe et validation de simplification de circuit

La netlist et le code PyPHS de simulation de l'oscillateur fixe sont donnés en annexe B. Le rapport de transformation M est choisi à $\frac{1}{\mu} + \frac{2\omega_0 C}{2000gm}$ ce qui produit une oscillation très légèrement décroissante. Le bruit résiduel de démarrage est à 1mV et le circuit LC est accordé sur une fréquence de 80 kHz. Parce qu'on ne connaît pas à l'avance le contenu harmonique du signal produit par l'oscillateur, on effectue la simulation à une fréquence d'échantillonnage très élevée $f_e \simeq 10 f_{osc} = 768kHz$. La figure 31 montre la tension de sortie du montage obtenue à la simulation. La figure 32 montre le spectre du signal obtenu. La distorsion harmonique est de 60 dB soit

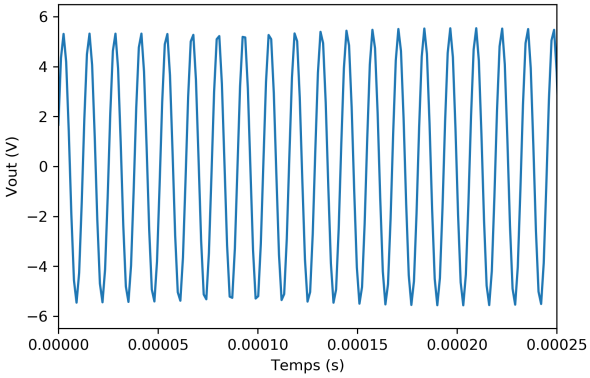

FIGURE 31 – Tension simulée en sortie de l'oscillateur fixe.

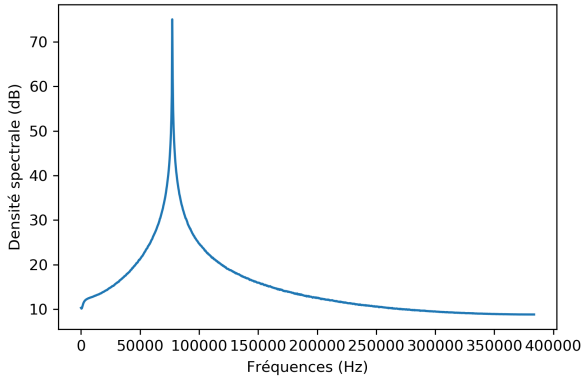

FIGURE 32 – Spectre du signal simulé en sortie de l'oscillateur fixe.

0.1% pour l'harmonique 2. L'oscillateur variable fonctionne sur le même principe et on considère par la suite que le signal obtenu est suffisamment pur pour que les oscillateurs soient modélisés par un générateur sinusoïdal, ce qui permet de réduire considérablement le temps de calcul.

5.1.2 Simulation du mélangeur

La netlist et le code PyPHS de simulation du mélangeur sont donnés en annexe C. On applique une tension $V_e = \frac{sin(2\pi f_p t)+sin(2\pi f t)}{2}$ en entrée du mélangeur avec $f_p = 80kHz$ et $f = f_p - f_m$, où f_m est la fréquence de jeu. La figure 33 montre la tension obtenue en sortie pour $f_m = 220Hz$ à une fréquence d'échantillonnage de $768kHz$. La simulation est lancée pour des conditions initiales nulles, le transitoire correspond donc à l'allumage de la lampe (sans temps de chauffe). Les figures 34 et 35 montrent les spectres du signal obtenu pour différents valeurs de f_m (sans le transitoire). Le signal utile et ses harmoniques sont en fait les produits d'intermodulation $nf_p + mf$ du mélangeur pour $m = -n$. On remarque en effet la présence d'un pic à la fréquence f_m ainsi que les harmoniques 2, 3, 5, 6, 7, 10, 11, etc. On constate également que les fréquences f_p, f et $f + f_p$ ne sont pas totalement supprimées, ni les harmoniques supérieurs, ce qui justifie une telle fréquence d'échantillonnage pour éviter tout repliement spectral. On observe également comme prévu que les distorsions harmoniques sont plus importantes pour les hautes fréquences de jeu.

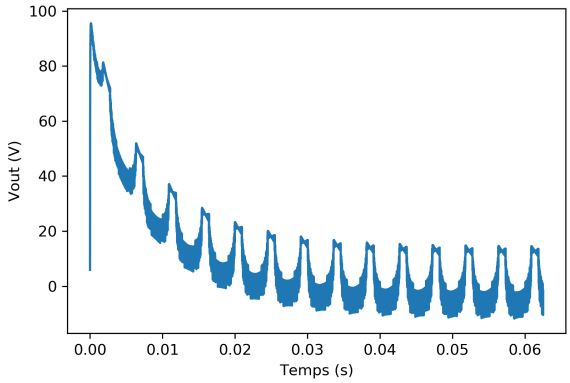

FIGURE 33 – Forme d'onde de la tension simulée en sortie de mélangeur pour $f_m = 220Hz$.

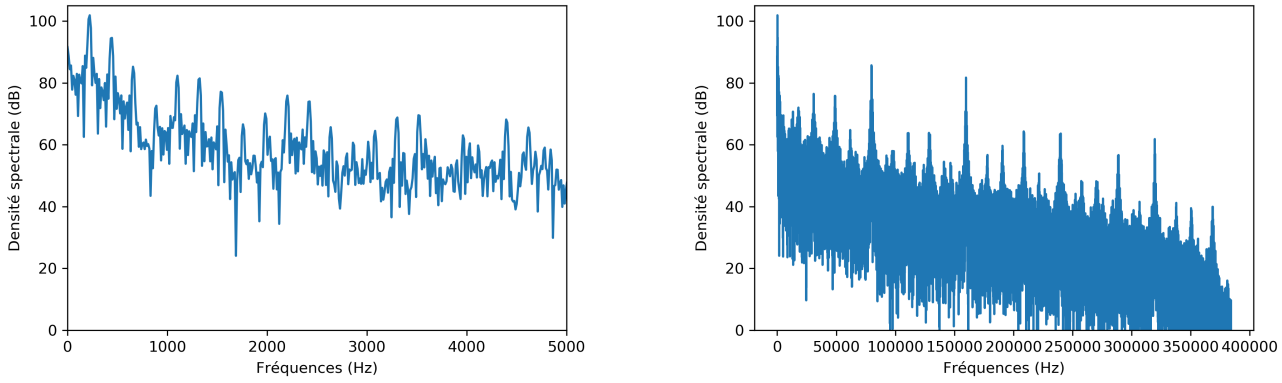

FIGURE 34 – Spectre du signal simulé en sortie du mélangeur, entre 0 et 5000Hz à gauche et entre 0 et 400000 Hz à droite, pour $f_m = 220Hz$.

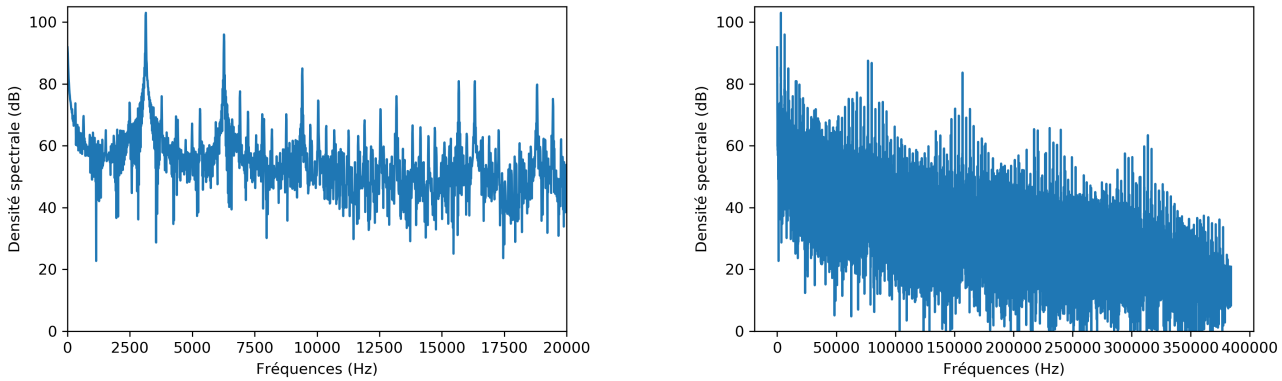

FIGURE 35 – Idem figure 34 pour $f_m = 3136Hz$.

5.1.3 Simulation de l'ensemble mélangeur+préamplificateur

La netlist et le code de simulation du préamplificateur sont donnés en annexe D. L'ensemble est simulé en connectant les deux SHP par un transformateur de rapport 1/3. La figure 36 montre le spectre du signal obtenu en sortie pour une tension d'entrée $V_e = \frac{sin(2\pi f_p t)+sin(2\pi f t)}{2}$ en entrée du mélangeur avec $f_p = 80kHz$ et $f = f_p - f_m$, avec $f_m = 220Hz$. On constate que le spectre obtenu est sensiblement le même que celui issu du mélangeur seul pour cette fréquence. Pour les paramètres que nous avons estimés, l'impédance de charge est en effet assez faible ; le préamplificateur fonctionne alors en classe A et n'apporte pas de non-linéarités supplémentaires. Il s'agit soit d'un choix délibéré de Maurice Martenot, soit d'une sous-estimation des paramètres de charge de notre part. Des mesures effectuées directement sur le circuit de l'Onde 169 permettraient de trancher.

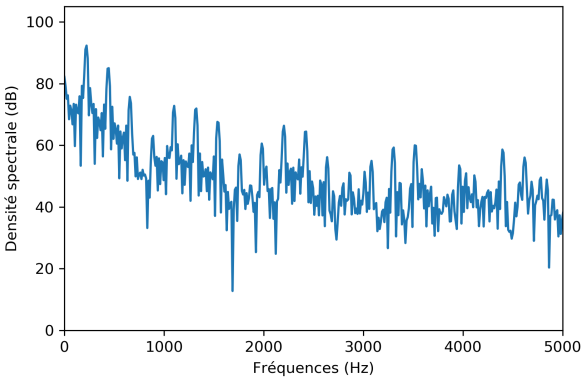

FIGURE 36 – Spectre du signal simulé en sortie de l'ensemble mélangeur+préamplificateur, entre 0 et 5000 Hz, pour $f_m = 220Hz$.

5.2 Temps réel avec implémentation dans JUCE

JUCE est un environnement de travail C++ open-source et multi-plateformes dédié au développement d'applications audio. Il comporte à la fois sa propre IDE (interface de développement) proposant des modèles pour chaque type d'application - standalone, plugin, mobile etc - appelée Projucer, ainsi qu'une importante bibliothèque de fonctions de traitement et d'interfaçage graphique. Sa modularité en fait un standard reconnu et exploité dans l'industrie. PyPHS permet de générer un objet C++ représentant le SHP et les équations associées, importable dans JUCE. On crée donc un système simplifié constitué des étages mélangeur et préamplificateur, que l'on met en cascade avec un générateur de somme de sinus directement écrit dans JUCE. Le résultat est un plugin VST, compatible avec la plupart des logiciels audio, dans lequel on peut contrôler la fréquence de jeu et le volume de sortie à l'aide de réglettes, et qui reproduit le son en sortie du préamplificateur de l'Onde 169 en temps réel. Nous obtenons ainsi un instrument virtuel jouable, fac-simile de notre modèle du circuit de l'Onde 169 (figure 37).

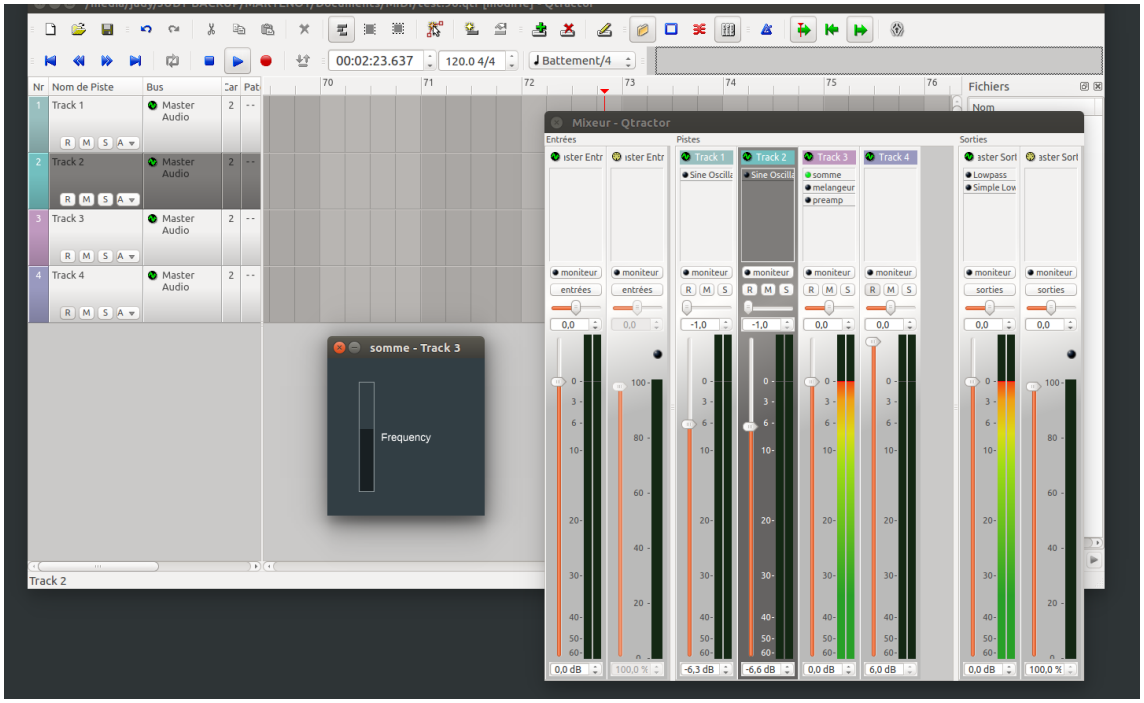

FIGURE 37 – Capture d'écran de l'utilisation du plugin temps-réel dans le logiciel Qtractor.

Conclusion et perspectives

Lors de ce stage, nous avons étudié en profondeur le circuit de l'Onde 169 et le comportement de ses composants. Cette étude nous a permis d'en proposer une modélisation par Systèmes Hamiltoniens à Ports et nous avons pu réaliser plusieurs simulations en temps différé comme en temps réel grâce à la bibliothèque open-source PyPHS. Nous avons contribué à cette bibliothèque en proposant un algorithme de résolution des conflits de réalisabilité applicable aux composants mono-variables non linéaires, travail ayant fait l'objet d'un article pour présentation à la 145e Convention de l'AES.

Les extensions théoriques possibles du travail effectué lors de ce stage consisteraient à généraliser la méthode formelle de résolution de conflits aux composants multi-variables, afin de l'appliquer notamment aux composants variables dans le temps. Plus généralement, il serait intéressant de modéliser les effets du vieillissement des composants dans un circuit électronique par SHP en intégrant des phénomènes d'endommagement irréversibles, ces phénomènes influant sur le son. Par ailleurs, simuler des circuits non linéaires hautes fréquences implique de travailler à des fréquences d'échantillonnage très élevées, ce qui augmente considérablement le temps de calcul. Explorer des méthodes alternatives à l'échantillonnage, comme le calcul de trajectoires paramétrées par exemple, se révélerait sans doute fructueux. Concernant la modélisation de l'instrument en soi, elle demande à être raffinée à plusieurs niveaux. Des mesures sur l'instrument et ses composants permettraient de confirmer la pertinence des modèles utilisés et les résultats des simulations de façon réellement quantitative, au-delà de l'aspect qualitatif du son obtenu. Enfin, pour parachever le réalisme sonore, une modélisation d'un diffuseur semble indispensable. Pour le contrôle de l'instrument virtuel, le marché propose à ce jour plusieurs modèles de claviers continus qui permettraient de combiner en une seule interface l'aspect "clavier" et l'aspect "ruban" ainsi que l'expressivité.

Remerciements

Je tiens à remercier tous les membres de l'équipe S3AM pour leur accueil chaleureux, leur implication et leur disponibilité, en particulier mes encadrants Thomas Hélie et David Roze. Un grand merci également à Henri Boutin, Arnaud Recher, Charles Picasso et Raphaël Sallé de Chou, dont les interventions efficaces et bienveillantes m'ont été précieuses. Merci bien sûr à Antoine Falaize pour sa générosité et sa réactivité. Merci à Thierry Maniguet et Stéphane Vaiedelich d'avoir ouvert les réserves du Musée, véritable caverne d'Ali-Baba. Enfin, bravo à mes camarades stagiaires de l'IRCAM qui ont contribué à faire de ce stage une expérience conviviale, stimulante et enrichissante.

Références

[1] Tung-Sol. 6f5 typical operating conditions and characteristics. https://frank.pocnet.net/sheets/127/6/6F5.pdf, 1942.

[2] Tung-Sol. 6c5 typical operating conditions and characteristics. https://frank.pocnet.net/sheets/127/6/6C5.pdf, 1939.

[3] RCA. 2a3 average plate characteristics. https://frank.pocnet.net/sheets/049/2/2A3.pdf, 1933.

[4] Thierry Courrier. Analyse de fonctionnement onde 169. unpublished document, Musée de la musique, 2012.

[5] John S Belrose. Reginald aubrey fessenden and the birth of wireless telephony. *IEEE Antennas and Propagation Magazine*, 44(2) :38–47, 2002.

[6] L. Quartier, T. Meurisse, J. Colmars, J. Frelat, and S. Vaiedelich. Intensity Key of the Ondes Martenot : An Early Mechanical Haptic Device. *Acta Acustica united with Acustica*, 101(2) :421–428, 2015.

[7] Julian Parker and Stefan Bilbao. Spring reverberation : A physical perspective. In *Proceedings of the 12th International Conference on Digital Audio Effects (DAFx'09)*, pages 416–421, 2009.

[8] Damien Bouvier, Thomas Hélie, and David Roze. Nonlinear homogeneous order separation for volterra series identification. In *20th International Conference on Digital Audio Effects*, 2017.

[9] Tristan Lebrun, Damien Bouvier, Thomas Hélie, and David Roze. Estimation de paramètres d'un haut-parleur électrodynamique non linéaire. In *14ème Congrès Français d'Acoustique*, 2018.

[10] Lee De Forest. Device for amplifying feeble electrical currents., January 15 1907. US Patent 841,387.

[11] E Leon Chaffee et al. Theory of thermionic vacuum tubes. *Fundamentals. Amplifiers. Detectors. New York & London, mcGraw Hill*, 8, 1933.

[12] Norman Koren. Improved vacuum tube models for spice simulations. *Glass Audio*, 8(5) :18–27, 1996.

[13] W Marshall Leach Jr. Spice models for vacuum-tube amplifiers. *Journal of the Audio Engineering Society*, 43(3) :117–126, 1995.

[14] Ivan Cohen and Thomas Hélie. Measures and parameter estimation of triodes, for the real-time simulation of a multi-stage guitar preamplifier. In *129th Convention of Audio Engineering Society*, San Francisco, United States, November 2010.

[15] Kristjan Dempwolf and Udo Zölzer. A physically-motivated triode model for circuit simulations. In *14th international conference on Digital Audio Effects DAFx*, volume 11, 2011.

[16] Christian Maennel. Théorie des oscillateurs électroniques. http://physique-maennel.pagesperso-orange.fr/wa_files/8_20Th_C3_A9orie_20des_20oscillateurs_20v4.pdf.

[17] Antoine Falaize and Thomas Hélie. PyPHS : Passive modeling and simulation in python, 2016. Python package, Web page.

[18] Alfred Fettweis. Wave digital filters : Theory and practice. *Proceedings of the IEEE*, 74(2) :270–327, 1986.

[19] David T Yeh, Jonathan S Abel, and Julius O Smith. Automated physical modeling of nonlinear audio circuits for real-time audio effects—part i : Theoretical development. *IEEE transactions on audio, speech, and language processing*, 18(4) :728–737, 2010.

[20] B. M. Maschke, A. J. Van der Schaft, and P. Breedveld. An intrinsic hamiltonian formulation of net- work dynamics : Non-standard poisson structures and gyrators. *Journal of the Franklin institute*, pages 923–966, 1992.

[21] Vincent Duindam, Alessandro Macchelli, Stefano Stramigioli, and Herman Bruyninckx. *Modeling and control of complex physical systems : the port-Hamiltonian approach*. Springer Science & Business Media, 2009.

[22] Arjan van der Schaft, Dimitri Jeltsema, et al. Port-hamiltonian systems theory : An introductory overview. *Foundations and Trends® in Systems and Control*, 1(2-3) :173–378, 2014.

[23] T Hélie, A Falaize, and N Lopes. 1 systèmes hamiltoniens à ports.

[24] Antoine Falaize and Thomas Hélie. Passive guaranteed simulation of analog audio circuits : A port-hamiltonian approach. *Applied Sciences*, 6(10) :273, 2016.

[25] Antoine Falaize. *Modélisation, simulation, génération de code et correction de systèmes multi-physiques audios : Approche par réseau de composants et formulation Hamiltonienne à Ports*. PhD thesis, Université Pierre & Marie Curie-Paris 6, 2016.

[26] Roger W Brockett. *Finite dimensional linear systems*, volume 74. SIAM, 2015.

[27] AJ Van der Schaft. A realization procedure for systems of nonlinear higher-order differential equations. *IFAC Proceedings Volumes*, 20(5) :85–90, 1987.

[28] Ivan Cohen. *Modélisation, analyse et identification de circuits non linéaires : Application aux amplificateurs guitare à lampes pour la simulation en temps réel*. PhD thesis, Université Pierre & Marie Curie-Paris 6, 2012.

Annexes

A Module de résolution des conflits de réalisabilité

```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
"""
Created on Fry May 25 15:23:49 2018
by @author: Najnudel
"""
import warnings
from pyphs.dictionary import pwl
from pyphs.dictionary.pwl.tools import data_generator
import numpy as np
import os


def PWL(X, Y, Xin):
    n = len(X)
    if n < 2:
        raise ValueError('X must contain at least 2 elements')
    if len(Y) != n:
        raise ValueError('Y and X must have same length')
    if not all(Y[i] < Y[i+1] and X[i] < X[i+1] for i in range(n-1)):
        raise ValueError('PWL must be increasing')
    if not 0 in X or Y[np.where(X==0)] != 0:
        raise ValueError('PWL of zero must be zero')
    else:
        Yout = []
        for x in Xin:
            Xtemp = np.append(X,x)
            Xtemp.sort()
            i, = np.where(Xtemp==x)
            if len(i) > 1:
                y = Y[i[0]]
            else :
                if i == 0:
                    y = (Y[1]-Y[0])/(Xtemp[2]-Xtemp[1])*(x-Xtemp[1]) + Y[0]
                elif i == n:
                    y = (Y[n-1]-Y[n-2])/(Xtemp[n-1]-Xtemp[n-2])*(x-Xtemp[n-2]) + Y[n-2]
                else:
                    y =  (Y[i]-Y[i-1])/(Xtemp[i+1]-Xtemp[i-1])*(x-Xtemp[i-1]) + Y[i-1]
            Yout = np.append(Yout,y)
        Yout.sort()
        return Yout


def PWL_inv(X, Y):
    n = len(X)
    if n < 2:
        raise ValueError('X must contain at least 2 elements')
    if len(Y) != n:
        raise ValueError('Y and X must have same length')
    if not all(Y[i] < Y[i+1] and X[i] < X[i+1] for i in range(n-1)):
        raise ValueError('PWL must be increasing')
    if not 0 in X or Y[np.where(X==0)] != 0:
        raise ValueError('PWL of zero must be zero')
    else:
```

```
        return Y, X


def PWL_comp(X, Y1, Y2, Z):
    Zout = PWL(Y2, Z, Y1)
    return X, Zout


def PWL_sum(XX, YY, tol=2):
    eps = np.finfo(float).eps
    n = len(XX)
    Xout = []
    Yout = []
    for i in range(n):
        Ytemp = list(YY[i])
        for k in range(n):
            if k != i:
                Ytemp += PWL(XX[k], YY[k], XX[i])
        Yout = np.append(Yout, [y for y in Ytemp if not any(y/(el+eps) - 1 <= tol*eps for el in Yo
        Yout.sort()
        Xout = np.append(Xout, [x for x in XX[i] if not any(x/(el+eps) - 1 <= tol*eps for el in Xo
        Xout.sort()
    return Xout, Yout


def PWL_integ(X, Y):
    p = X[1]-X[0]  #step is supposedly constant
    Yout = []
    z, = np.where(X==0)
    z = z[0]
    for i in range(len(X)):
        if z < i:
            integ = p*(Y[z+1:i].sum() + Y[i]/2)  #trapezoidal rule
        else:
            integ = -p*(Y[i+1:z].sum() + Y[i]/2)
        Yout = np.append(Yout, integ)
    return X, Yout


def PWL_Heq(XX, YY, tol=2):
    n = len(XX)
    XXo = []
    YYo = []
    Xq, Yq = PWL_sum(YY, XX, tol)
    for i in range(n):
        Xint, Yint = PWL_integ(XX[i], YY[i])
        Xint, Yint = PWL_comp(YY[i], XX[i], Xint, Yint)
        XXo.append(Xint)
        YYo.append(Yint)
    X, Y = PWL_sum(XXo, YYo, tol)
    Xout, Yout = PWL_comp(Yq, Xq, X, Y)
    return Xout, Yout


def get_key(dic, value):
    for k, v in dic.items():
            if v == value:
                return k
```

```
def isStorage(edge):
    _, _, dic = edge
    return (dic['type'] == 'storage')


def initialize_Heq():
    XX, YY = [], []
    label = 'Heq_'
    keys = []
    nodeslist = []
    return XX, YY, label, keys, nodeslist


def replace_Heq_par(graph, keys, nodes, path, label):
    """
        Replace all parallel storages within a parallel graph with
        equivalent storage

        Parameters
        ---------

        graph : Graph
            Graph object to be modified

        keys : list
            list of edges keys to be removed

        nodes : tuple
            tuple of edges nodes to be removed

        path : str
            path to file containing pwl values of equivalent storage

        label : str
            label of the equivalent storage
        """
    for key in keys:
        graph.remove_edge(*nodes, key)
    dic = {'file':path, 'integ':False, 'ctrl':'f'}
    Heq = pwl.Storage(label, nodes, **dic)
    graph += Heq
    warnings.warn('Replacing parallel storage with equivalent storage ' + label)


def replace_Heq_ser(graph, keys, nodeslist, path, label, firstnode, lastnode):
    """
        Replace all serial storages within a serial graph with
        equivalent storage

        Parameters
        ---------

        graph : Graph
            Graph object to be modified

        keys : list
```

```
            list of edges keys to be removed

        nodeslist : list
            list of edges nodes to be removed

        path : str
            path to file containing pwl values of equivalent storage

        label : str
            label of the equivalent storage

        firstnode : str
            first node of the equivalent storage

        lastnode : str
            last node of the equivalent storage
        """
    for key, nodes in zip(keys, nodeslist):
        graph.remove_edge(*nodes, key)
    dic = {'file':path, 'integ':False, 'ctrl':'e'}
    Heq = pwl.Storage(label, (firstnode, lastnode), **dic)
    graph += Heq
    warnings.warn('Replacing serial storages with equivalent storage ' + label)


def graph_analysis_serial(graph):
    """
        Walk through a serial graph and perform replace_Heq_ser wherever possible
    """
    edges = graph.edgeslist
    XX, YY, label, keys, nodeslist = initialize_Heq()
    nodesbin = []
    for edge in edges:
        node1, node2, dic = edge
        nodes = node1, node2
        data = graph.get_edge_data(*nodes)
        key = get_key(data, dic)
        if isStorage(edge) and dic['ctrl'] == 'e':
            keys.append(key)
            nodeslist.append(nodes)
            path = dic['file']
            vals = np.vstack(map(np.array, data_generator(path)))
            x_vals = vals[0, :]
            h_vals = vals[1, :]
            XX.append(x_vals)
            YY.append(h_vals)
            label += str(dic['label'])
            if len(keys) == 1:
                firstnode = node1
            else:
                nodesbin.append(node1)
            lastnode = node2
            if edges.index(edge) == len(edges)-1 and len(keys) > 1:
                Xeq, Yeq = PWL_Heq(XX, YY)
                path = os.path.join(os.getcwd(), label + '_data.pwl')
                np.savetxt(path, np.vstack((Xeq, Yeq)))
                replace_Heq_ser(graph, keys, nodeslist, path, label, firstnode, lastnode)
                XX, YY, label, keys, nodeslist = initialize_Heq()
```

```
        elif len(keys) > 1:
                Xeq, Yeq = PWL_Heq(XX, YY)
                path = os.path.join(os.getcwd(), label + '_data.pwl')
                np.savetxt(path, np.vstack((Xeq, Yeq)))
                replace_Heq_ser(graph, keys, nodeslist, path, label, firstnode, lastnode)
                XX, YY, label, keys, nodeslist = initialize_Heq()
        else:
            XX, YY, label, keys, nodeslist = initialize_Heq()
    graph.remove_nodes_from(nodesbin)


def graph_analysis_parallel(graph):
    """
        Walk through a parallel graph and perform replace_Heq_par wherever possible
    """
    edges = graph.edgeslist
    XX, YY = [], []
    label = 'Heq_'
    keys = []
    nodes = edges[0][0], edges[0][1]
    for edge in edges:
        _, _, dic = edge
        data = graph.get_edge_data(*nodes)
        key = get_key(data, dic)
        if isStorage(edge) and dic['ctrl'] == 'f':
            keys.append(key)
            path = dic['file']
            vals = np.vstack(map(np.array, data_generator(path)))
            x_vals = vals[0, :]
            h_vals = vals[1, :]
            XX.append(x_vals)
            YY.append(h_vals)
            label += str(dic['label'])
    if len(keys) > 1:
        Xeq, Yeq = PWL_Heq(XX, YY)
        path = os.path.join(os.getcwd(), label + '_data.pwl')
        np.savetxt(path, np.vstack((Xeq, Yeq)))
        replace_Heq_par(graph, keys, nodes, path, label)


def graph_eq(splitgraph):
    """
        Walk through a split graph (call Graph.sp_split method first) and perform
        replace_Heq_par and replace_Heq_ser wherever possible
    """
    edges = splitgraph.edgeslist
    for edge in edges:
        if edge[2]['type'] == 'graph':
            subgraph = edge[2]['graph']
            graph_eq(subgraph)
    if edges[0][1] == edges[1][1]:
        graph_analysis_parallel(splitgraph)
    else:
        graph_analysis_serial(splitgraph)
```

B Netlist et code de simulation pour l'oscillateur fixe

```
electronics.source Vb ('N1', '#'): type=voltage;
electronics.source Vg1 ('N3', 'N5'): type=voltage;
electronics.source Vg2 ('N5', '#'): type=voltage;
electronics.source Io1 ('N1', 'N2'): type=current;
electronics.source Io2 ('N1', 'N2'): type=current;
electronics.capacitor C8 ('N1', 'N2'): C=('C8', 470e-12);
electronics.inductor L ('N1', 'N2'): L=('L', 1);
electronics.capacitor C3 ('N4', '#'): C=('C3', 0.22e-06);
electronics.resistor R2 ('N4', '#'): R=('R2', 7.5e03);
electronics.triode T ('N4', 'N2', 'N3'): mu=('mu', 98.); Ex=('Ex', 1.6); Kg=('Kg', 2614.);
                                         Kp=('Kp', 905.); Kvb=('Kvb', 1.87);
                                         Vcp=('Vcp', 0.5); Va=('Va', 0.33); Rgk=('Rgk', 1300.);
```

```
# -*- coding: utf-8 -*-

import os
import math as m
from pyphs import Netlist, signalgenerator

path = os.path.join(os.getcwd(), 'porteuse_boucle.net')
netlist = Netlist(path)
graph = netlist.to_graph()
core = graph.to_core()

label = 'porteuse'
T = 1
config = {'fs': 768e03, 'path': os.path.join(os.getcwd(), label)}
nt = T*int(config['fs'])

F = 80e03
gm = 1.6e-03
mu = core.subs[core.symbols('mu')]
C_value = core.subs[core.symbols('C8')]
L_value = 1/(C_value*(2*m.pi*F)**2)
subs = {core.symbols('L'): L_value}
core.subs.update(subs)
alpha = 1/mu
beta = 4*m.pi*F*C_value/gm

indices = (1, 3)
M = alpha + beta/2000
core.add_connector(indices, M)
core.connect()

simu = core.to_simulation(config = config)
config_isig = {'which': 'const',
'tsig': T,
'fs': config['fs'],
'A': 90
}
isig = signalgenerator(**config_isig)

config_vgsig = {'which': 'noise',
'tsig': T,
'fs': config['fs'],
'A': 1e-3
}
```

```
vgsig = signalgenerator(**config_vgsig)

def u():
    for Vb_, Vg_ in zip(isig(), vgsig()):
        yield (0, Vb_, Vg_)

simu.init(u = u(), nt = nt)
simu.process()
```

C Netlist et code de simulation pour le mélangeur

```
electronics.source Vf ('N5', 'N4'): type=voltage;
electronics.source Vbm ('N1', '#'): type=voltage;
electronics.source Iom ('N1', 'N2'): type=current;
electronics.resistor R4 ('N5', 'N3'): R=('R4', 1e06);
electronics.capacitor C21 ('N5', 'N3'): C=('C21', 200e-12);
electronics.capacitor C9 ('N2', 'N1'): C=('C9', 1e-09);
electronics.inductor L ('N2', 'N1'): L=('L', 1);
electronics.capacitor C4 ('N4', '#'): C=('C4', 0.22e-06);
electronics.resistor R3 ('N4', '#'): R=('R3', 1e03);
electronics.triode T ('N4', 'N2', 'N3'): mu=('mu', 20.); Ex=('Ex', 1.5); Kg=('Kg', 2837.);
                                         Kp=('Kp', 138.); Kvb=('Kvb', 89.); Vcp=('Vcp', 0.8);
                                         Va=('Va', 0.33); Rgk=('Rgk', 1300.);
```

```
# -*- coding: utf-8 -*-

import os
from pyphs import Netlist, signalgenerator
import math as m

path = os.path.join(os.getcwd(), 'melangeur.net')
netlist = Netlist(path)
graph = netlist.to_graph()
core = graph.to_core()
label = 'melangeur'
T = 1
config = {'fs': 768e3, 'path': os.path.join(os.getcwd(), label)}
nt = T*int(config['fs'])

F1 = 80000
F2 = F1-220
fc = 500
tau = 0.0002
C21_value = core.subs[core.symbols('C21')]
R4_value = tau/C21_value
C9_value = core.subs[core.symbols('C9')]
L_value = 1/(C9_value*(2*m.pi*fc)**2)
subs = {core.symbols('L'): L_value, core.symbols('R4'): R4_value}
core.subs.update(subs)

simu = core.to_simulation(config = config)
config_fsig1 = {'which': 'sin',
'tsig': T,
'fs': config['fs'],
'f0': F1
}
fsig1 = signalgenerator(**config_fsig1)
```

```
config_fsig2 = {'which': 'sin',
'tsig': T,
'fs': config['fs'],
'f0': F2
}
fsig2 = signalgenerator(**config_fsig2)

config_isig = {'which': 'const',
'tsig': T,
'fs': config['fs'],
'A': 100
}
isig = signalgenerator(**config_isig)

def u():
    for Vb_, Vf1_, Vf2_ in zip(isig(), fsig1(), fsig2()):
        yield (0, Vf1_/2+Vf2_/2, Vb_)

simu.init(u = u(), nt = nt)
simu.process()
```

D Netlist et code de simulation pour l'ensemble mélangeur-préamplificateur

```
electronics.source Vgp ('N3', '#'): type=voltage;
electronics.source Iog ('N3', '#'): type=current;
electronics.source Vbp ('N1', '#'): type=voltage;
electronics.source Iop ('N1', 'N2'): type=current;
electronics.resistor Rp ('N1', 'N2'): R=('Rp', 10e03);
electronics.inductor L2 ('N1', 'N2'): L=('L2', 79);
electronics.capacitor C5 ('N4', '#'): C=('C5', 0.277e-06);
electronics.resistor R5 ('N4', '#'): R=('R5', 1e03);
electronics.triode Tp ('N4', 'N2', 'N3'): mu=('mu', 20.); Ex=('Ex', 1.5); Kg=('Kg', 2837.);
                                          Kp=('Kp', 138.); Kvb=('Kvb', 89.); Vcp=('Vcp', 0.8);
                                          Va=('Va', 0.33); Rgk=('Rgk', 1300.);
```

```
# -*- coding: utf-8 -*-

import os
import math as m
from pyphs import Netlist, Core, signalgenerator

pathm = os.path.join(os.getcwd(), 'melangeur.net')
netlistm = Netlist(pathm)
graphm = netlistm.to_graph()
corem = graphm.to_core()

pathp = os.path.join(os.getcwd(), 'preamp.net')
netlistp = Netlist(pathp)
graphp = netlistp.to_graph()
corep = graphp.to_core()

# params melangeur
F1 = 80000
F2 = F1-220
fc = 500
tau = 0.0002
C21_value = corem.subs[corem.symbols('C21')]
```

```
R4_value = tau/C21_value
C9_value = corem.subs[corem.symbols('C9')]
L_value = 1/(C9_value*(2*m.pi*fc)**2)
subs = {corem.symbols('L'): L_value, corem.symbols('R4'): R4_value}
corem.subs.update(subs)

# connexion melangeur
label = 'ondes'
core = Core(label)
core += corem + corep

# connexion preamp
Mp = 1/3
indicesp = (0, 5)
core.add_connector(indicesp, Mp)
core.connect()

T = 1
config = {'fs': 768e03, 'path': os.path.join(os.getcwd(), label)}
nt = T*int(config['fs'])
simu = core.to_simulation(config = config)

config_isigm = {'which': 'const',
'tsig': T,
'fs': config['fs'],
'A': 100
}
isigm = signalgenerator(**config_isigm)

config_isigp = {'which': 'const',
'tsig': T,
'fs': config['fs'],
'A': 180
}
isigp = signalgenerator(**config_isigp)

config_fsig1 = {'which': 'sin',
'tsig': T,
'fs': config['fs'],
'f0': F1
}
fsig1 = signalgenerator(**config_fsig1)

config_fsig2 = {'which': 'sin',
'tsig': T,
'fs': config['fs'],
'f0': F2
}
fsig2 = signalgenerator(**config_fsig2)

def u():
    for Vf1, Vf2, Vbm_, Vbp_ in zip(fsig1(), fsig2(), isigm(), isigp()):
        yield (Vf1/2+Vf2/2, Vbm_, 0, 0, Vbp_)

simu.init(u = u(), nt = nt)
simu.process()
```