Université du Québec
Institut national de la recherche scientifique
Énergie, Matériaux et Télécommunications

Inverse design of optical signal processing systems

Par

Benjamin MacLellan

Mémoire présenté pour l'obtention du grade de
Maîtrise en sciences, M.Sc.
de l'énergie et des matériaux

Jury d'évaluation

Examinateur externe	Prof. Pablo Bianucci Concordia University
Examinateur interne	Prof. Kulbir Ghuman INRS-EMT
Directeur de recherche	Prof. Roberto Morandotti INRS-EMT

Acknowledgments

Firstly, I would like to thank my supervisor, Prof. Roberto Morandotti, for his continued support throughout my years at INRS, for giving me the freedom to pursue my curiosity wherever it led, and for the many opportunities to grow and learn here at INRS. I would also like to thank Prof. José Azaña for lending his expertise, time, and support through the development of this project. To my friends, colleagues, and mentors in the Ultrafast Optical Processing and Nonlinear Photonics Groups at INRS, each of whom has impacted me and the works contain in these pages in their own way. In particular, to Piotr Roztocki for his friendship and mentorship. Our many discussions about physics, science, music, and more have helped me grow, and have helped crystallize the work contained in these pages from a vague idea to reality. To Bennet Fischer, Dr. Mario Chemnitz, Dr. Yoann Jestin, Dr. Luis Romero Cortés, Dr. Stefania Sciara, Nicolas Perron, Robin Helsten, Dr. Michael Kues, and Dr. James van Howe for selflessly sharing with me their advice, support, expertise, and passion for pursuing challenging problems. I have been incredibly lucky to be surrounded by such a group of talented and hard-working people, and the invaluable lessons I have learned from each of you will always remain with me. Thanks to Julie Belleville and Kaleb Ruscitti, for their hard work in helping bring these ideas to life and working with me to develop these software tools for optical design. I would also like to thank the Natural Sciences and Engineering Research Council of Canada and INRS for their financial support. And finally, to my parents and family, Sarah, Brian, Robyn, and Kayla, who have supported me with unconditional belief and love through all these years, and to Emily, who has been my best friend and partner through all the highs and lows of this adventure. This thesis, on the form and function of optical systems, is dedicated to Robin Hopper.

Abstract

Inverse design methods – algorithmic techniques for discovering new systems based on desired functional characteristics – are powerful computational techniques utilized in a wide array of scientific and engineering disciplines. In optics, inverse design has accelerated the development of new, high-performing optical lens systems, nanophotonic components, and quantum optics experimental setups, among others. This work develops novel, inverse design techniques for automatically designing new optical systems, specifically towards optical signal processing setups. In this approach, optical systems are represented as a computational graph, comprised of parameterized conventional optical components. Automatic differentiation provides an efficient and flexible method to optimize system parameters and analyze their sensitivities to perturbations. New optical systems, which have not been explicitly programmed, are automatically produced by iteratively changing the graph topologies via an evolutionary algorithm. The highly general and flexible methodology developed in this work is demonstrated to re-discover a variety of known optical systems, proving its effectiveness and highlighting its potential to expedite the discovery and design of entirely new photonic systems for a broad range of applications.

keywords: *inverse design; optical signal processing; ultrafast optics; automatic differentiation; evolutionary algorithms; gradient descent; Hessian analysis; graph topology optimization*

Résumé

Les méthodes de conception inverses - techniques algorithmiques pour découvrir de nouveaux systèmes basés sur les caractéristiques fonctionnelles souhaitées - sont des techniques informatiques puissantes utilisées dans un large éventail de disciplines scientifiques et d'ingénierie. En optique, la conception inverse a accéléré le développement de nouveaux systèmes de lentilles optiques hautes performances, de composants nanophotoniques et de configurations expérimentales d'optique quantique, entre autres. Ce travail développe de nouvelles techniques de conception inverses pour concevoir automatiquement de nouveaux systèmes optiques, en particulier pour des configurations de traitement optique ultra-rapides. Dans cette approche, les systèmes optiques sont représentés sous la forme d'un graphe de calcul, composé de composants optiques paramétrés. La différenciation automatique fournit une méthode efficace et flexible pour optimiser les paramètres du système et analyser leurs sensibilités aux perturbations. De nouveaux systèmes optiques, qui n'ont pas été explicitement programmés, sont automatiquement produits en changeant itérativement les topologies de graphe via un algorithme évolutif. Les outils algorithmiques ciblent la conception de systèmes de traitement optique ultrarapides et il est montré qu'ils redécouvrent une variété de techniques optiques connues et non triviales avec une application dans un grand nombre de technologies.

mots clés: *conception inverse; traitement des signaux optiques; optique ultra-rapide; modélisation du système optique; différenciation automatique; algorithmes évolutifs; descente gradient; analyse de Hesse; optimisation de la topologie du graphe*

Synopsis

Conception inverse de systèmes de traitement de signaux optiques

La conception de nouveaux systèmes optiques a étayé chaque étape du progrès scientifique dans le domaine de la photonique et a conduit à d'innombrables nouvelles technologies [1–6]. Les scientifiques et les ingénieurs exploitent régulièrement divers phénomènes et principes optiques pour construire des outils et des technologies toujours plus performants basés sur la lumière : des télécommunications à la détection, en passant par le divertissement et bien plus encore. Ces systèmes optiques sont conçus, construits et testés pour répondre à des fonctionnalités spécifiques et souhaitées, qu'il s'agisse de composants nanophotoniques pour le routage optique à haut rendement [1], de systèmes de lentilles pour les systèmes d'imagerie [3] ou d'expériences pour tester la compréhension fondamentale de la mécanique quantique [7]. Cependant, la conception de systèmes optiques complexes peut prendre beaucoup de temps, être peu intuitive, nécessiter des connaissances spécialisées ainsi qu'une grande expertise. En général, ce processus nécessite qu'un concepteur propose et analyse chaque solution possible, que ce soit de manière expérimentale, analytique ou par simulation. Ces méthodes, appelées conception directe, sont restrictives et nécessitent un concepteur pour tester chaque solution candidate. Elles sont aussi limitées par l'intuition et l'expertise du concepteur. Ces limitations peuvent restreindre la portée et l'étendue de nouvelles solutions et peuvent prendre beaucoup de temps pour converger vers des conceptions efficaces.

La conception inverse, cependant, atténue certains de ces défis et accélère le développement de nouveaux systèmes. La conception inverse fait référence à une vaste catégorie de techniques et de méthodes permettant de trouver de nouvelles solutions algorithmiques pour résoudre un problème. Elle est largement utilisée dans l'ingénierie automobile et aérospatiale [8–10], l'électronique [11], l'architecture [12, 13], le traitement biologique [14], la synthèse chimique [15], et au-delà. Les méthodes de conception inverse visent à paramétrer les systèmes possibles et à utiliser des ressources et des algorithmes de calcul pour explorer rapidement et intelligemment de nouvelles solutions afin d'évaluer celles qui pourraient être prometteuses. En pratique, le concepteur spécifie un comportement et des performances cibles, puis permet à l'algorithme d'explorer et d'évaluer de nombreuses nouvelles solutions possibles, en indiquant à l'utilisateur final le ou les meilleurs résultats. Les défis de la conception inverse sont différents de ceux de la conception directe, car la difficulté consiste plutôt à créer des algorithmes et des modèles mathématiques suffisamment adaptés et généraux pour trouver des solutions utiles.

En optique, la conception inverse est utilisée dans un certain nombre de sous-disciplines avec beaucoup de succès. Elle est avant tout utilisée dans la conception de systèmes de lentilles pour l'imagerie, l'astronomie et le divertissement [16–20]. Ces démonstrations ont été les premières dans le domaine et sont les plus répandues de l'utilisation de la conception inverse en optique. Plus récemment, la conception de dispositifs nanophotoniques et de métamatériaux de haute performance pour les télécommunications et la détection a été améliorée grâce à l'adoption de telles techniques algorithmiques [1, 21–25]. Enfin, des concepts connexes ont également été démontrés dans la conception de nouvelles expériences d'en optique quantique pour produire des états de lumière complexes et intriqué [5, 7, 26]..

Les méthodes de conception inverse, cependant, devraient trouver d'autres applications et notamment dans la conception de systèmes de traitement des signaux optiques ultrarapides. Alors que l'optimisation des paramètres a été utilisée dans la conception des lasers [27], les télécommunications [28], l'optique ultrarapide [29] et d'autres domaines connexes, la conception inverse au niveau du système à proprement dit n'a pas encore été largement explorée. Lors de la conception d'un système optique, il faut tenir compte de trois éléments principaux : les composants optiques qui constituent le système, la topologie du système (c-à-d. l'ordre et la direction de la propagation optique), et les paramètres opérationnels de ces composants.

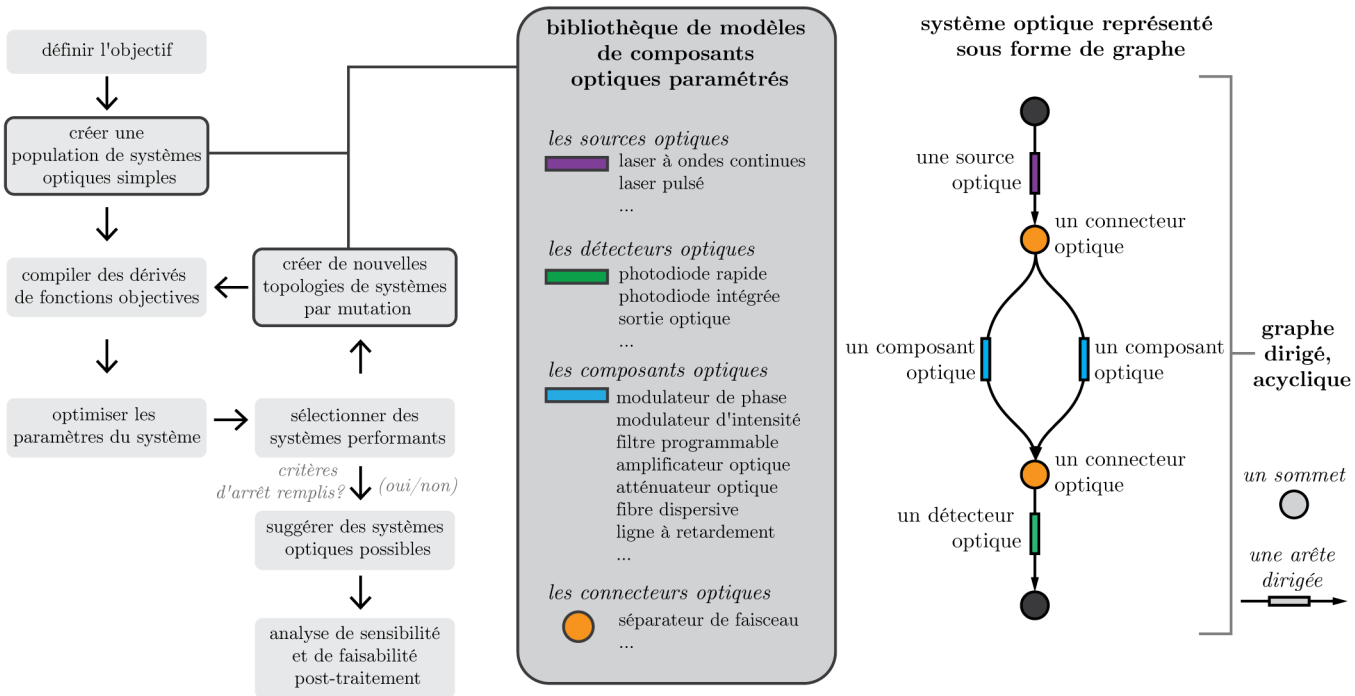


Figure S.1 – Vue d'ensemble de la conception inverse des systèmes de traitement optique ultrarapide. **(à gauche)** Organigramme de l'algorithme d'optimisation de la topologie, qui construit et évalue itérativement de nouveaux systèmes optiques, en utilisant **(au centre)** une bibliothèque de modèles de composants optiques. **(à droite)** Les systèmes optiques sont représentés par des graphes acycliques dirigés, où les nœuds (cercles orange) représentent les interconnexions optiques et les arêtes dirigés (flèches de connexion) représentent les éléments optiques qui génèrent, détectent ou manipulent le champ optique dans un seul chemin de propagation.

Premièrement, dans le traitement optique ultrarapide des signaux et dans d'autres domaines connexes, un ensemble commun de composants et d'effets optiques est utilisé pour de nombreuses technologies différentes. Il s'agit, par exemple, de la dispersion, de la non-linéarité, de l'interférence, de la perte et du gain - chacun dépendant des degrés de liberté du champ optique (temporel, spectral, polarisation, etc.). Diverses combinaisons de ces éléments optiques supportent d'innombrables technologies et produits commerciaux, et sont réalisées par des composants optiques tels que : sources optiques, modulateurs, fibres dispersives, amplificateurs, filtres spectraux, photodiodes, etc. Comme ces phénomènes optiques ne commutent généralement pas entre eux, l'ordre dans lequel ils agissent sur un champ électromagnétique lors de la propagation modifie considérablement le rendement global d'un système (voir la figure 1.2 du texte principal). Ainsi, *la topologie* du système optique permet de nombreuses fonctionnalités différentes avec les mêmes composants optiques. La topologie d'un système est l'aménagement des composants, ou la manière dont ceux-ci sont reliés entre eux par des chemins de propagation (par exemple, des guides d'ondes, des cordons de fibres optiques ou des liaisons en espace libre). Les systèmes peuvent être en séries, c'est-à-dire qu'ils se composent d'un seul chemin spatial dont les composants sont connectés les uns après les autres. Ils peuvent également contenir des chemins de propagation optiques parallèles, qui donnent lieu à des effets d'interférence de la part de circuits optiques de type interférométrique. Les systèmes peuvent aller de topologies relativement simples contenant quelques composants connectés en série, à des systèmes complexes composés de centaines de chemins de propagation optique parallèles [30]. La dernière considération à prendre en compte dans la conception des systèmes optiques sont *les paramètres des composants*, qui dictent comment chaque composant manipule la lumière pendant la propagation. Ces paramètres sont les valeurs réglables sur un composant optique donné qui définissent la fonctionnalité de celui-ci, et qui comprennent, par exemple, la profondeur de modulation d'un modulateur, la longueur d'une fibre dispersive, ou la valeur de gain d'un amplificateur optique. En résumé, la conception inverse des configurations optiques au niveau du système nécessite la prise en compte de ces trois éléments : les composants optiques, la topologie du système et les paramètres des composants.

L'objectif principal de ce travail est de développer des outils de calcul pour la conception inverse de systèmes de traitement optique ultrarapide. Pour accomplir cette tâche, j'ai d'abord développé une bibliothèque de modèles de composants optiques compacts et paramétrés tels que les sources, les modulateurs, les éléments dispersifs, les amplificateurs, etc. (voir annexe A.4). Ensuite, une méthode de test de la sensibilité des systèmes aux perturbations a été mise en œuvre, elle utilise la différenciation automatique pour déterminer la dépendance des performances du système à chaque composant et paramètre. Enfin, deux routines d'optimisation sont utilisées pour trouver les systèmes optiques qui fonctionnent le mieux pour la tâche souhaitée - la première pour rechercher dans les topologies des systèmes et la seconde pour optimiser les paramètres des composants. Ces routines d'optimisation sont couplées, de sorte que pour chaque topologie proposée, un ensemble optimal de paramètres opérationnels est également trouvé. Ces outils, pris ensemble, constituent une nouvelle méthode pour la conception de systèmes optiques pour une grande variété d'objectifs

souhaités et sont appelés `ASOPE`, ou *Recherche automatisée pour les expériences de traitement optique* (Automated Search for Optical Processing Experiments). Dans ce travail, je me concentre sur les systèmes qui manipulent les degrés de liberté temps-fréquence des composants à base de fibres, cependant, les méthodes sont extensibles à d'autres degrés de liberté et à des plateformes photoniques.

`ASOPE`: Recherche automatisée pour les expériences de traitement optique

Dans la section suivante, j'expose les méthodes de représentation, d'analyse et d'optimisation développées dans l'`ASOPE` et appliqué à la conception inverse des systèmes optique. Le logiciel a été développé dans le langage de programmation Python.

Représentation et simulation des systèmes

Le premier niveau de représentation est à l'échelle d'un composant optique individuel. Les composants optiques fondamentaux sont représentés par un modèle compact et paramétré qui décrit son effet sur le champ optique (soit dans le domaine temporel, $\Psi(t)$, soit dans le domaine fréquentiel, $\tilde{\Psi}(\omega)$). Chaque modèle agit soit sur un seul chemin spatial (par exemple, modulateurs, filtres, amplificateurs, etc.), soit sur des cartes entre différents chemins spatiaux (par exemple, séparateurs et multiplexeurs). Pour un composant qui agit sur un seul chemin spatial, la fonction de transfert est écrite sous la forme,

$$\Psi_{\text{out}} = f_{c_i}(\Psi_{\text{in}}, \mathbf{x}_{c_i}) \tag{S.1}$$

où les paramètres, $\mathbf{x}_{c_i}$, sur un composant optique donné, c_i, sont tous considérés comme des valeurs scalaires limitées et continues et représentent des paramètres expérimentaux accordables. Pour les composants ayant plusieurs ports d'entrée/sortie (c'est-à-dire un système contenant des séparateurs et/ou des multiplexeurs), un formalisme matriciel des fonctions de transfert doit être adopté. Pour les composants avec N chemins d'entrée et M chemins de sortie,

$$\Psi_{\text{out},\ell} = \sum_{k=1}^{N} f_{c_i,(k,\ell)}(\Psi_{\text{in},k}, \mathbf{x}_{c_i}) \tag{S.2}$$

$$\{k \in \mathbb{Z} \mid 1 \leq k \leq N\}, \qquad \{\ell \in \mathbb{Z} \mid 1 \leq \ell \leq M\}$$

Pour plus de détails sur les modèles de composants exacts utilisés dans l'`ASOPE`, voir l'annexe A.4, qui comprend une source laser à onde continue, une source laser à mode bloqué, des modulateurs de phase, des modulateurs d'intensité, des filtres programmables, des lignes à retard, des amplificateurs optiques, des atténuateurs optiques, des séparateurs et des photodiodes.

Le deuxième niveau de représentation modélise un système optique complet sous forme de graphe de calcul. Un graphe, $G = (V, E)$, comporte un ensemble de sommets/nœuds $\{V\}$, et un ensemble d'arêtes, $\{E\}$, qui relient ces nœuds. Les graphes de calcul est une méthode compacte de description de fonctions complexes, utilisée dans une variété de domaines, y compris les réseaux neuronaux artificiels et les systèmes de flux. Ces nœuds et ces arêtes représentent les entrées et les sorties de fonctions élémentaires qui sont mises en correspondance les unes avec les autres afin de produire des fonctions plus complexes. En outre, les graphes sont des représentations utiles dans un certain nombre d'approches de conception inverse, y compris l'apprentissage automatique, la chimie, la conception de réseaux et l'optique quantique [7, 15, 31]. Plus précisément, dans l'`ASOPE`, le graphe de calcul est un type précis de graphe : un *multi-graphe dirigé, acyclique*, ou DAG. Dans l'`ASOPE`, les nœuds et les arêtes représentent tous deux les composants optiques du système et le flux de lumière à travers le système (voir la figure S.1 ci-dessus et la figure 3.2 dans le texte principal). Le codage utilisé dans l'ASOPE pour représenter un système optique est le suivant :

Arêtes du graphe: Elles représentent les composants qui génèrent, manipulent ou détectent un champ optique dans un mode spatial unique ; et comprennent les sources optiques, les détecteurs ou les éléments de contrôle optique (par exemple, les modulateurs, les filtres, les amplificateurs, les atténuateurs, les lignes à retard, etc.) La direction des arêtes spécifie la direction de la propagation optique.

Nœuds du graphe: Ils représentent les interconnexions entre les éléments optiques, qui cartographient les trajets spatiaux entrants et sortants ; ils comprennent les séparateurs, les coupleurs et les multiplexeurs.

Les sommets de type source et puit: Ils sont utilisés au début et à la fin (terminaux) d'un graphe de système. Ce sont des points d'ancrage pour définir la direction de la propagation, c'est-à-dire que la propagation optique à travers le système est définie par l'ordre topologique du graphe, des sommets de type source aux sommets de type puit.

Le système optique global, composé d'un ensemble de N composants, c_i, $\{i \in \mathbb{Z} \mid 1 \leq i \leq N\}$, est ensuite paramétré par $\mathbf{x}_G$, qui englobe les paramètres de chaque composant optique, $\mathbf{x}_i$ (c-à-d. $\mathbf{x}_G = [\mathbf{x}_{c_1}, \mathbf{x}_{c_2}, \ldots, \mathbf{x}_{c_N}]^{\mathrm{T}}$). Avec un ensemble complet de composants, une topologie de graphe de système et un ensemble de paramètres, la fonction de propagation globale du système entier est simulée en composant les fonctions de transfert des éléments optiques dans l'ordre donné par la topologie du graphe. Par exemple,

$$f_G = f_{c_N} \circ f_{c_{N-1}} \circ \cdots \circ f_{c_1}(\Psi, \mathbf{x}) \tag{S.3}$$

Pour un système optique plus complexe qui contient plusieurs chemins de propagation spatiale (c'est-à-dire un système contenant des séparateurs et/ou des multiplexeurs), il faut adopter un formalisme matriciel de la composition des fonctions - le nombre d'arêtes entrants ou sortants vers

un sommet indiquant les dimensions de la matrice. Voir la figure 3.2 et les équations 3.4-3.6 pour des exemples concrets. Cette approche générale permet de décrire et de modéliser une grande variété de systèmes optiques, et d'ajouter facilement de nouveaux types de composants à la bibliothèque de modèles pour d'autres cas d'utilisation. .

Afin de permettre une analyse efficace et automatique de ces systèmes optiques, les modèles de composants et les modèles de graphes de systèmes sont développés pour permettre une différenciation automatique. Celle-ci est utilisée pour l'analyse automatique, appelée "analyse de sensibilité", qui estime l'effet de chaque paramètre x_j ou de chaque ensemble de paramètres de composants $\mathbf{x}_{c_i}$ sur les performances du système, ou bien les deux. De même, la différenciation automatique permet d'optimiser les paramètres par le biais d'une solution basée sur des gradients, comme la descente de gradient ou les routines de type Newton [16, 32]. Pour le langage de programmation Python, le logiciel `autograd` offre un ensemble convivial pour la différenciation automatique des modèles écrits avec des fonctions Python, `numpy` et `scipy`, et peut suivre les dérivés par le biais de fonctions à valeur complexe telles que FFT et IFFT. Cela permet d'écrire des modèles de composants optiques dans le domaine où ils sont les plus simples et les plus directs [33] - par exemple, les éléments dispersifs sont modélisés dans le domaine fréquentiel, les éléments modulateurs variant dans le temps dans le domaine temporel, et les dérivés peuvent être suivis par la fonction de propagation.

Évaluation et analyse du système

L'évaluation d'un graphe de système optique permet d'estimer la performance d'un système par rapport à l'objectif souhaité. Grâce à la différenciation automatique, la définition des fonctions de coût est très souple : elle peut être définie dans le domaine temporel ou fréquentiel, utiliser des fonctions complexes et être rapidement prototypée.

La principale exigence est qu'il renvoie une valeur de coût scalaire unique, $F_G(\mathbf{x}) \in \mathbb{R}$, et utilise des fonctions supportées par `autograd`. Cette fonction de coût est ensuite minimisée par rapport aux paramètres du graphe, ce qui correspond à une meilleure performance du système. Une fonction de coût utile compare le signal optique (ou électronique) d'un élément spécifique de la configuration à un signal optique (ou électronique) souhaité par le biais d'une fonction de similarité. Plus précisément, nous utilisons la somme des carrés, ou ℓ^2-norme, sur la fenêtre temporelle de la simulation, T,

$$F_G(\mathbf{x}) = \int_T |I_{\text{target}} - I_{\text{generated}}|^2 \, \mathrm{d}t \tag{S.4}$$

où $I(t) = |\Psi|^2$ est l'intensité optique dépendant du temps. Cette intégrale de chevauchement, $F(G, \mathbf{x}) \in \mathbb{R}$, mesure la similarité d'un champ optique généré, dépendant du temps, avec une intensité cible. L'équation S.4 peut être appliquée de manière similaire à un signal de tension dépendant du temps mesuré au niveau de l'un des éléments de la photodiode. Un deuxième exemple de fonction de coût est la sensibilité du système aux perturbations - par exemple, la sensibilité de

la puissance optique moyenne à un déphasage dans un chemin spatial,

$$F_G(\mathbf{x}) = -\left|\frac{\partial P_{\text{avg}}}{\partial x_\varphi}\right|_{\mathbf{x}} \tag{S.5}$$

où P_{avg} est la puissance optique moyenne et x_φ est la phase d'un élément de déphasage dans le système. Cette méthode n'est accessible que par l'utilisation de la différenciation automatique et pourrait être utile pour la recherche de systèmes très stables aux perturbations ou très sensibles (par exemple, la détection optique).

De plus, l'analyse de la façon dont les changements des paramètres des composants d'un système optique affectent la fonction objective est bénéfique pour acquérir une compréhension plus approfondie du système et guider la réalisation du système en laboratoire. Dans le cadre de l'`ASOPE`, comme les topologies des systèmes évoluent, des méthodes automatiques d'analyse de sensibilité sont nécessaires. À cette fin, on utilise une analyse du système basée sur le gradient qui utilise le vecteur de gradient, $\nabla F_G(\mathbf{x})$, et la matrice hessienne, $\mathbf{H}_G(\mathbf{x})$, calculée par différenciation automatique. En particulier, l'analyse basée sur la matrice hessienne permet de comprendre comment les paramètres de conception du système sont couplés, c-à-d. comment deux paramètres influencent ensemble les performances du système – ce qui ne peut être réalisé avec le seul gradient ou d'autres techniques d'analyse de sensibilité.

La matrice hessienne, $\mathbf{H}$, est la dérivée de second ordre d'une fonction $F : \mathbb{R}^n \to \mathbb{R}$ et est une matrice réelle, carrée et symétrique dont les indices sont définis comme $H_{i,j} = \partial^2 F/\partial x_i \partial x_j$. Elle peut être utilisée pour extraire deux informations utiles sur le système optique étudié : *les paramètres courbes/directions* et *les principales courbes/directions* [34–36]. Les courbures des paramètres sont les éléments diagonaux de $\mathbf{H}$, ou H_{ij}, $\forall\, i = j$. Inversement, les directions et les courbures principales sont respectivement les vecteurs propres et les valeurs propres de la matrice hessienne, satisfaisant à l'équation suivante:

$$\mathbf{H}\mathbf{v}_i = \lambda_i \mathbf{v}_i \tag{S.6}$$

Alors que les courbures des paramètres indiquent la sensibilité des paramètres pris individuellement, les courbures principales et les éléments hors diagonale indiquent la sensibilité des paramètres combinés. Si l'ampleur de la courbure du paramètre (principal) est faible, le déplacement dans la direction du paramètre (principal) associé est très stable, et *vice versa*. En outre, les signes des courbures principales peuvent indiquer si la fonction sur $\mathbf{x}^*$ est un minimum, un maximum ou un point de selle (points qui peuvent causer des problèmes de convergence dans une variété de routines d'optimisation des paramètres [16, 32]). Pour comparer les paramètres avec des unités variables, la matrice hessienne est mise à l'échelle comme dans l'équation 3.10 de sorte que tous les éléments de la matrice H_{ij} ont les mêmes unités physiques (c-à-d. qu'ils ont les mêmes unités que F). La figure S.2 montre l'analyse de sensibilité d'un exemple de système (c-à-d. un système permettant de manipuler la période de répétition d'un laser pulsé). La figure S.2d en particulier montre la puis-

sance de l'analyse de sensibilité basée sur la matrice hessienne, car les principales courbes/directions indiquent un couplage entre les paramètres qui ne peut être obtenu à partir des seules courbes des paramètres (figure S.2c). Plus précisément, comme $\lambda_4 \approx 0$, la direction principale $\mathbf{v}_4$ est très stable - et correspond à l'allongement de la première fibre et au rétrécissement de la seconde fibre (ce qui a un impact global nul sur les performances du système).

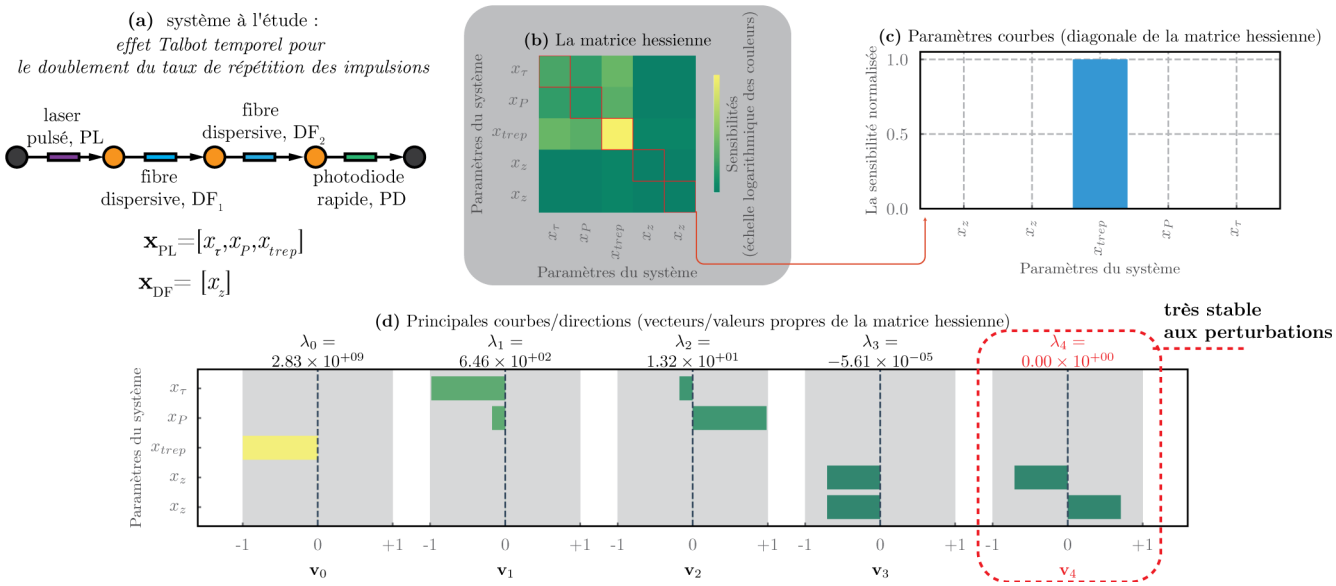


Figure S.2 – Analyse automatique de la sensibilité des systèmes optiques. **(a)** Exemple de système de traitement des signaux optiques, qui multiplie la fréquence de répétition d'un train d'impulsions laser par l'effet Talbot temporel [6]. Deux fibres sont incluses pour démontrer comment l'analyse automatique de sensibilité peut être utilisée pour comprendre les processus physiques sous-jacents et les considérations dans le système – par exemple, quels composants du système sont les plus influents sur la performance du système et/ou quels composants peuvent être potentiellement redondants. L'analyse de sensibilité automatique comprend le calcul **(b)** de la matrice hessienne, **(c)** des courbes paramètre et **(d)** des courbes/directions principales.

Si ces méthodes d'analyse de sensibilité permettent de comprendre comment la fonction objective change en fonction des perturbations, des imperfections ou de l'incertitude des paramètres, elles n'englobent pas toutes les imperfections qui existent dans les systèmes optiques réels. En particulier, elle ne peut pas englober le bruit stochastique provenant de sources optiques, électroniques et mécaniques - par exemple, l'émission stimulée amplifiée se produisant dans un amplificateur optique peut avoir des effets néfastes dans une variété de systèmes optiques, tels que les systèmes de télécommunications [37]. Dans le cadre de simulations plus réalistes, le bruit est ajouté au champ optique simulé lorsqu'il se propage dans le(s) système(s). Une contribution générale au bruit est modélisée sous la forme d'un bruit gaussien additif (Additive Gaussian Noise - AGN) sur les composants les plus applicables, c-à-d. les éléments actifs, y compris les sources laser, les amplificateurs et les photodiodes. Chaque modèle de composant possède un modèle de bruit connecté, qui simule les caractéristiques du bruit de chaque composant par le biais d'un bruit additif à bande limitée, tel que le bruit rose, le bruit passe-bas, etc. Une analyse de sensibilité et une optimisation peuvent être effectuées en présence de ce bruit. En outre, le bruit peut être activé/désactivé pour tous les composants afin de comparer la propagation du bruit à la propagation idéale, ce qui pourrait

permettre d'optimiser la tolérance au bruit des systèmes optiques par des méthodes basées sur le gradient.

Optimisation

Comme les paramètres des composants et la topologie du système utilisent différentes représentations abstraites - les paramètres en tant que vecteur de scalaires réels délimités et le système optique en tant que graphe dirigé - l'optimisation globale est divisée en deux algorithmes distincts. Une routine d'optimisation imbriquée est développée ; la routine externe change et optimise la structure du graphe en ajoutant, échangeant et supprimant des sommets et des arêtes, et la routine interne trouve les paramètres optimaux sur tous les composants du graphe donné. En travaillant en tandem, ces algorithmes imbriqués découvrent des systèmes optiques qui fonctionnent idéalement pour une tâche définie par l'utilisateur (par exemple, trouver des systèmes optiques qui peuvent produire un signal optique ciblé). Le chapitre 4 démontre l'application de cette routine d'optimisation pour la conception de systèmes optiques pour une variété de tâches.

Optimisation des paramètres

Il existe des outils d'optimisation omniprésents pour optimiser une fonction $F(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$. Il s'agit notamment de la descente de gradient, des algorithmes génétiques, de l'optimisation par essaims particulaires, de la méthode de Newton, etc. (comme indiqué au section 2.3). À partir d'un graphe de système G et d'un vecteur de paramètres $\mathbf{x}$, les fonctions de propagation et d'évaluation sont compilées, ainsi que les dérivées successives par différenciation automatique. La fonction d'évaluation, $F_G(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$, peut alors être considérée comme une fonction de boîte noire. Cette fonction, ainsi que le vecteur de gradient, $\nabla F_G(\mathbf{x})$, peuvent être fournis à une variété de routines d'optimisation standard – y compris les algorithmes basés sur le gradient et les algorithmes évolutionnaires/distribués pour rechercher les points optimaux dans l'espace des paramètres. Les routines d'optimisation des paramètres testées en vue d'une utilisation dans l'`ASOPE` comprennent : un algorithme génétique [38], une variante de l'algorithme ADAM personnalisée pour s'adapter aux paramètres limités [16, 39], l'algorithme L-BFGS [40, 41], et une implémentation de type PSO [42, 43] (voir la figure 4.4 et le tableau 4.2). Une approche hybride a finalement été adoptée comme méthode d'optimisation des paramètres par défaut. Tout d'abord, une optimisation "grossière" est réalisée par un algorithme génétique, suivie d'une optimisation "fine" par la méthode L-BFGS. C'est une stratégie intéressante pour l'optimisation des systèmes physiques, car l'algorithme évolutif initial explore mieux l'espace de conception complet et une méthode de gradient ultérieure peut alors converger vers le minimum local le plus proche [44]. Empiriquement, cette approche hybride d'un algorithme génétique suivi d'une minimisation locale L-BFGS a montré les meilleures performances au cours du développement. Après l'optimisation, le post-traitement et l'analyse sont

effectués en utilisant, par exemple, l'analyse de la matrice hessienne. La figure S.3 montre les étapes de l'optimisation et de l'analyse d'une topologie de système optique fixe.

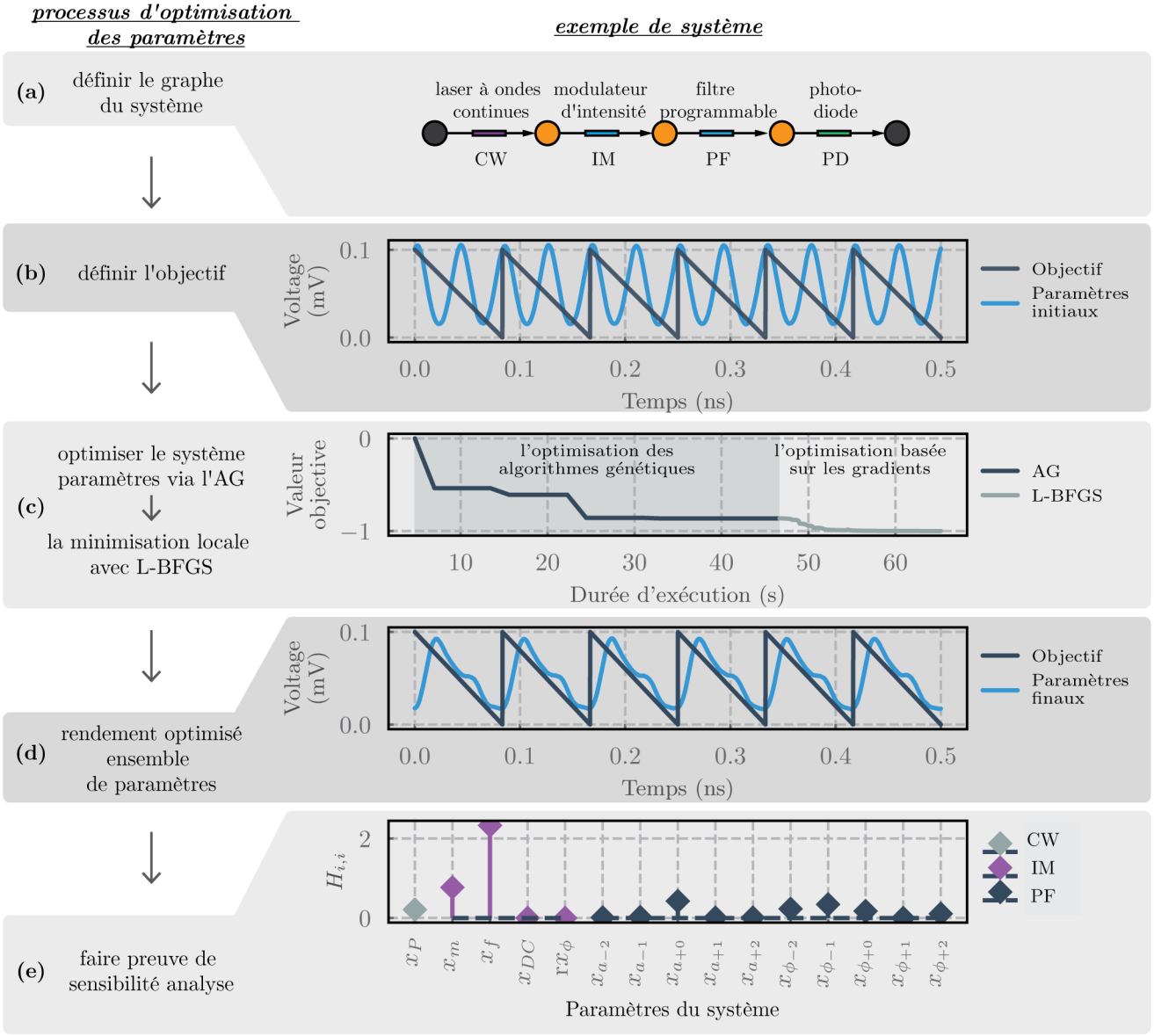


Figure S.3 – Procédure d'optimisation des paramètres sur une topologie de graphe fixe. **(a)** Exemple de système optique, qui génère des formes d'ondes électroniques arbitraires par des manipulations optiques et une conversion optique-électronique. La fonction de coût de ce système est la ℓ^2-norme entre le signal optique cible (forme d'onde en dents de scie; ligne solide, bleu foncé) et le signal optique généré (ligne solide, bleu clair). **(b)** Un ensemble initial de paramètres de départ a des performances médiocres, qui sont **(c)** optimisés par des routines d'optimisation AG et L-BFGS pour produire un ensemble de paramètres de composants très performants **(d)**. **(e)** Analyse de sensibilité automatique sur le système, évaluée aux paramètres optimisés.

Optimisation de la topologie

Le dernier pilier central du travail présenté est lié à l'optimisation des topologies des systèmes optiques. Ici, des algorithmes heuristiques sont utilisés pour construire et améliorer itérativement des graphes de systèmes optiques, en explorant de nouvelles configurations. La méthode utilise un algorithme évolutif pour faire évoluer les systèmes optiques, en optimisant et analysant les paramètres, tout en regroupant les systèmes haute performance pour les proposer à l'utilisateur. Une population de graphes mute et change de manière itérative, les graphes les plus performants étant plus susceptibles de rester dans la population et/ou de produire de nouveaux systèmes. Pour chaque graphe, les fonctions de propagation et d'évaluation sont compilées et les paramètres sont optimisés comme décrit ci-dessus. À chaque génération, des graphes très performants sont sélectionnés pour produire la population suivante et subissent une mutation stochastique pour produire de nouveaux systèmes (c-à-d. que leur topologie change par l'ajout, la suppression ou la modification des composants optiques). Ce processus est répété jusqu'à ce qu'un critère d'arrêt soit rempli, et l'`ASOPE` renvoie une famille de systèmes optiques les plus performants pour un examen plus approfondi. Une vue de haut niveau du processus est fournie dans la figure S.1 (voir également la figure 3.9).

La principale méthode utilisée par l'`ASOPE` pour générer de nouveaux systèmes optiques est l'utilisation d'opérateurs de mutation de graphes. Ces opérateurs de mutation, illustrés à la figure S.4a, sont:

L'ajout d'arête Ajoute une seule nouvelle arête entre deux sommets existants. Physiquement, il s'agit de l'ajout d'un nouveau chemin spatial en parallèle avec les chemins spatiaux existants (par exemple, un dispositif interférométrique).

L'ajout d'arête et de sommet Ajoute à la fois un nouveau composant et un connecteur dans un chemin spatial courant. Physiquement, cela équivaut à ajouter un nouveau composant en série dans un chemin spatial existant.

La suppression d'arête Supprime une arête (entre les sommets u et v) du système. Aucun sommet n'est supprimé, sauf s'il n'y a pas d'autre chemin entre u et v (c-à-d. si le graphe n'est pas connecté), auquel cas u et v sont regroupés en un seul sommet.

L'échange de modèle Échange un composant pour un autre composant du même type. Par exemple, une source optique peut être échangée contre une autre (par exemple, un laser à onde continue contre un laser pulsé), un élément à trajet unique peut être échangé contre un autre élément (par exemple, un modulateur de phase contre une fibre dispersive), etc. Toutefois, les modèles ne peuvent pas être échangés contre des éléments d'un type différent (par exemple, un modulateur ne peut pas être échangé contre un détecteur).

Ces opérateurs de mutation, dénommés $M_i = M_{\text{ajoutArete}}$, $M_{\text{ajoutAreteSommet}}$, $M_{\text{suppressionArete}}$, and $M_{\text{echangeModele}}$, peuvent chacun être appliqués de multiples façons distinctes sur le même graphe.

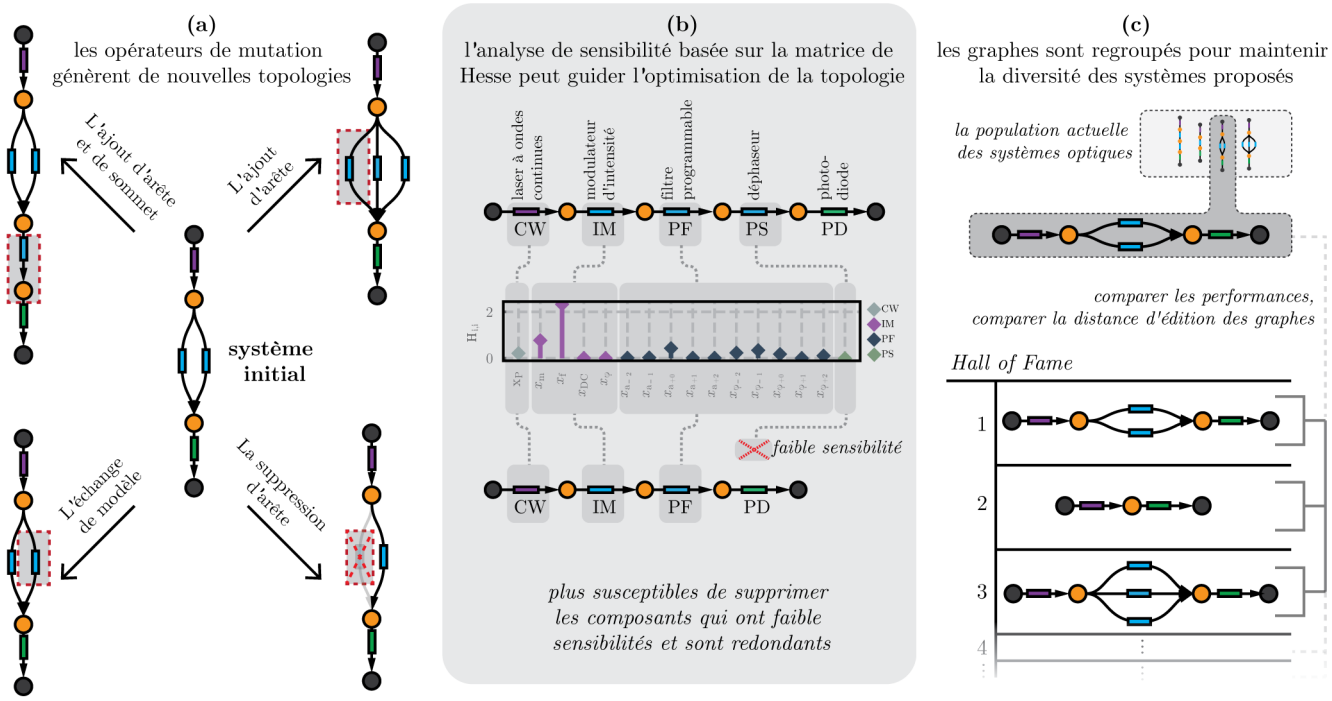


Figure S.4 – Méthodes d'optimisation de la topologie des graphes de systèmes optiques. **(a)** Les opérateurs de mutation permettent à l'ASOPE de produire de nouveaux graphes de systèmes optiques. Chaque opérateur fait correspondre un graphe à un nouveau graphe, ils sont sélectionnés et appliqués stochastiquement pendant l'optimisation de la topologie. **(b)** L'analyse automatique de sensibilité peut être utilisée pour guider l'optimisation de la topologie vers un système plus simple et plus compact. Les composants ayant moins d'influence sur les performances (i.e. une faible sensibilité) ont plus de chances d'être éliminés. **(c)** Le regroupement des graphes de systèmes permet de conserver une diversité de types de systèmes à proposer à l'utilisateur. Chaque graphe est comparé au "Hall of Fame" en termes de performances et de similarité avec d'autres systèmes, en utilisant la métrique de similarité de la distance d'édition des graphes.

$M_{i,j}$ désigne donc uniquement l'opérateur de mutation et l'emplacement dans un graphe où il sera mis en œuvre. Chacun de ces opérateurs de mutation fait correspondre un graphe actuel à un nouveau graphe; $M_{i,j} : G \to G'$. À chaque nouvelle génération de l'algorithme d'optimisation de la topologie, une nouvelle population de graphes est générée en appliquant ces opérateurs de mutation de manière probabiliste, c'est-à-dire que chaque opérateur $M_{i,j}$ a une probabilité $\Pr(M_{i,j})$ d'être choisie et appliquée (une seule mutation est appliquée à la fois).

La manière dont ces probabilités sont calculées est fondamentalement importante pour la convergence et la performance de l'`ASOPE`. Les méthodes les plus simples, mais naïves, de fixation des probabilités sont faites pour :

i) Avoir une probabilité de sélection uniforme pour toutes les mutations possibles, de sorte que,

$$\Pr(M_{i_1,j_1}) = \Pr(M_{i_2,j_2}),\ \forall i_1, j_1, i_2, j_2$$

ii) Avoir une probabilité de sélection constante pour chaque type de mutation unique, de sorte que

$$\Pr(M_i) = \text{constante}$$

Cependant, des méthodes plus sophistiquées peuvent améliorer la convergence et trouver des systèmes optiques plus performants. À cette fin, nous développons une méthode adaptative, guidée par la matrice hessienne, pour fixer les probabilités de mutation $\Pr(M_{i,j})$, où les composants qui ont peu d'impact (tel que mesuré par l'analyse de sensibilité de la matrice hessienne) sur la valeur objective, ont plus de chances d'être éliminés (voir figure S.4b). Cette méthode est décrite plus en détail à la section 3.3.2 et à l'annexe A.2.

À mesure que l'optimisation progresse, des systèmes optiques très performants sont stockés dans *le Hall of Fame* (HoF, [45]). En regroupant les graphes, de multiples types de systèmes distincts qui utilisent des composants et des techniques optiques différents sont maintenus grâce à l'optimisation de la topologie. L'objectif est de maintenir la diversité des systèmes finaux proposés à l'utilisateur. Un graphe n'est ajouté à la HoF que si, et seulement si, il a un meilleur score qu'un membre actuel de la HoF et qu'il n'y a pas déjà de graphes plus performants et très similaires dans la HoF. L'ASOPE compare les graphes via la *distance d'édition des graphes* (GED), comme pseudo-mesure de similarité entre deux graphes [46, 47], et utilise un seuil de similarité pour maintenir différents types de configuration. Ce seuil vise à garantir que des topologies de graphes très similaires ne "dépassent" pas la distance d'édition du graphe et qu'un ensemble de systèmes divers soit renvoyé à l'utilisateur à la fin de l'optimisation de la topologie.

Le logiciel `ASOPE` a été développé pour tirer profit de l'aptitude inhérente de l'algorithme à un traitement parallèle. Au cours de l'optimisation de la topologie, différents systèmes sont générés, optimisés et analysés sur différents cœurs d'unités centrales. La méthode passe de manière transparente du traitement à un seul cœur (pour les tests et le débogage), au multicœur (tel qu'utilisé pour toutes les expériences de calcul présentées ici), à un cluster de calcul haute performance (pour les travaux futurs, voir l'annexe A.3).

Systèmes conçus par l'`ASOPE`

Ici, l'algorithme `ASOPE` décrit ci-dessus est appliqué à trois tâches de traitement du signal optique : la manipulation de la période d'impulsion laser à verrouillage de mode, la génération d'une forme d'onde électronique arbitraire à grande vitesse et la détection des changements de phase. Cela démontre son large domaine d'application et sa capacité à reproduire des techniques optiques connues dans une variété de tâches de traitement du signal optique.

Multiplication/division de la période d'impulsion du laser

La manipulation de la période d'impulsion des lasers pulsés est une technique courante et importante dans les systèmes optiques ultrarapides [48–52]. Les méthodes de manipulation de la fréquence de répétition dépendent des caractéristiques de l'impulsion d'entrée : notamment la période d'impulsion d'entrée ($x_{t_{\text{rep}}}$), la puissance optique (x_P) et la largeur d'impulsion (x_τ). Les méthodes comprennent, par exemple: l'entrelacement des trains d'impulsions qui ont été divisés et retardés par des interféromètres [50], l'utilisation des effets Talbot temporels et spectraux [6, 49], la mise en forme de l'amplitude ou de la phase spectrale [48, 51, 52] (ou bien les deux) et la découpe des impulsions.

En utilisant l'algorithme `ASOPE`, nous visons la multiplication et la division du temps de répétition des impulsions pour un laser pulsé à entrée fixe. Tout au long de l'optimisation de la topologie, un seul laser pulsé est fixé comme source optique (i.e qu'il n'est pas retiré ou échangé par l'optimisation de la topologie). Le laser pulsé d'entrée a des paramètres fixes de largeur d'impulsion, de temps de répétition et de puissance de crête (on suppose que les impulsions sont de forme gaussienne). L'objectif des systèmes optiques est de manipuler le train d'impulsions de telle sorte que le champ optique à la sortie soit également un train d'impulsions avec un temps de répétition différent (un multiple scalaire de l'entrée). Considérons la multiplication ou la division d'un temps de répétition des lasers par un facteur de p/q, où $p, q \in \mathbb{Z}$ (de telle sorte que lorsque $p > q$ le temps de répétition augmente et lorsque $p < q$ il diminue). La figure S.5a montre trois systèmes optiques proposés par l'`ASOPE` pour doubler le temps de répétition des impulsions du laser d'entrée. L'`ASOPE` utilise de façon intéressante un certain nombre de techniques optiques connues pour résoudre ce problème. Sur la figure S.5a.1, le système utilise la mise en forme ligne par ligne du spectre optique avec un filtre programmable. Les systèmes présentés sur les figures S.5a.2 et a.3 utilisent respectivement l'effet Talbot temporel [6] et l'entrelacement de trains d'impulsions séparées et retardées. Une analyse plus approfondie de ces systèmes, ainsi que des configurations plus complexes et peu intuitives suggérées par l'ASOPE qui ne sont pas présentées ici, sont nécessaires pour déterminer comment la conception de ces systèmes se compare aux techniques des systèmes de pointe actuels. Voir la section 4.1.1 et la figure 4.1 dans le texte principal pour d'autres exemples de systèmes suggérés pour cette tâche.

Génération de formes d'ondes arbitraires en radiofréquence

La génération de formes d'ondes électroniques arbitraires à grande vitesse est d'une importance fondamentale dans un grand nombre de domaines [53–57]. Une méthode utilise les largeurs de bande et les mécanismes de contrôle de l'optique pour produire des formes d'onde arbitraires par voie optique, suivie d'une conversion optique-électronique (via une photodiode). Ces méthodes utilisent à la fois des entrées optiques à ondes continues et pulsées, manipulant le champ optique soit dans le domaine spectral via des filtres et la dispersion [56, 57], soit dans le domaine temporel avec des retards [54, 55], soit un mélange des deux. Pour un examen de ces méthodes, voir Références. [54] ou [58].

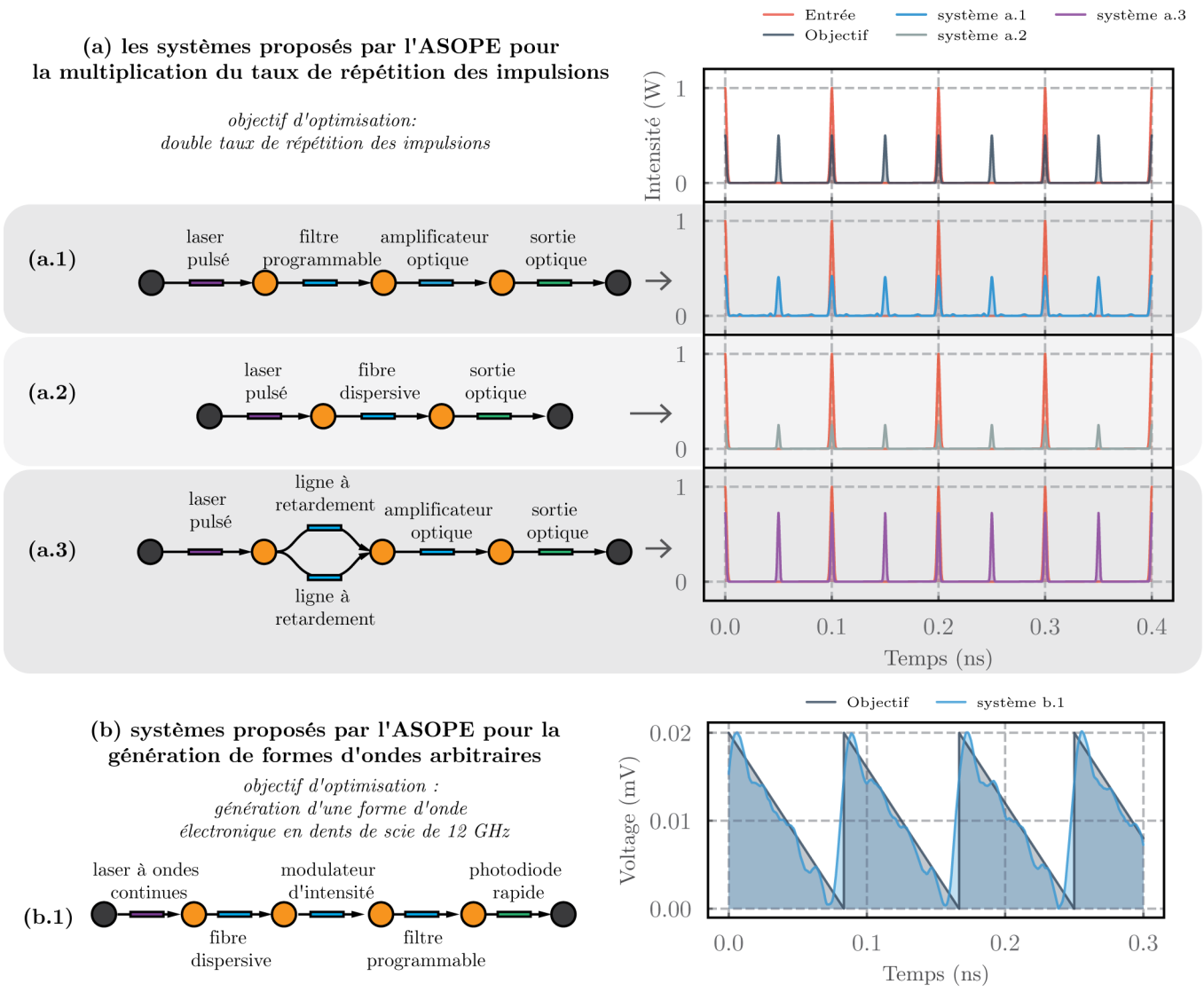


Figure S.5 – Les systèmes proposés par l'ASOPE pour **(a)** la manipulation de la fréquence de répétition des lasers pulsés et **(b)** la génération de formes d'ondes arbitraires. **(a.1-a.3)** Trois systèmes optiques différents renvoyés par l'ASOPE pour l'objectif cible de doubler la fréquence de répétition des impulsions laser pulsé, en utilisant divers techniques de traitements de signaux optiques connus. Les sorties optiques associées (à droite) montrent que chaque système atteint l'objectif visé, mais des différences de puissance optique sont observées. Les paramètres optimisés des systèmes ne sont pas indiqués ici par souci de concision. **(b.1)** Un exemple de système optique suggéré par l'ASOPE pour la génération d'une forme d'onde électronique en dents de scie de 12 GHz. Voir les figures 4.1 à 4.3 pour d'autres exemples.

En appliquant l'ASOPE au problème de la suggestion de configurations pour la génération de formes d'onde arbitraires assistée par la photonique, nous recherchons des systèmes optiques qui produisent les formes d'onde électroniques souhaitées après conversion optique-électronique (i.e en mesurant le signal de tension temporel simulé au niveau de la photodiode). La fonction objectif utilisée est la ℓ^2-norme entre le signal électronique généré et le signal cible (avec compensation des différences de phase globales entre le signal cible et le signal généré). Une forme d'onde cible est

définie dans le domaine RF par sa forme (par exemple, carrée, en scie, pulsée, binaire, etc.), son taux de répétition et son amplitude. Dans le texte principal, nous visons la génération de formes d'onde arbitraires à grande vitesse allant de 10 GHz à 40 GHz. Ici, la figure S.5b montre un exemple de système suggéré par l'ASOPE après optimisation de la topologie pour générer une forme d'onde électronique en dents de scie de 12 GHz. Ce système utilise une entrée laser à onde continue et génère de nouvelles composantes spectrales par modulation. Ces nouvelles composantes spectrales sont ensuite manipulées en amplitude et en phase via le filtre programmable qui, lorsqu'il tombe sur une photodiode à grande vitesse, battent ensemble et produisent une forme d'onde très similaire à la forme d'onde idéale. Voir la figure 4.2 pour d'autres démonstrations de cette tâche de conception, en particulier pour les systèmes proposés. Une fois la topologie optimale du système déterminée, l'utilisateur peut rapidement tester l'étendue des formes d'onde qui peuvent être produites avec le système en relançant l'optimisation des paramètres sur différentes formes d'onde cibles.

Il est difficile de comprendre les mécanismes importants et les contributions des composantes au fonctionnement global d'un système. Dans les méthodes de conception prospective, où un concepteur invoque une nouvelle conception et en étudie manuellement les propriétés, le concepteur a une compréhension plus approfondie des processus sous-jacents à l'œuvre. À l'opposé, dans la conception inverse, lorsqu'un algorithme "génère" une nouvelle conception, les processus importants (et intéressants) ne sont pas toujours clairs au départ. Là encore, c'est là que les techniques d'analyse, telles que l'analyse hessienne, peuvent être utiles - les interactions entre les éléments optiques et l'importance relative des composants peuvent être estimées (voir section 4.2.2).

Détection sensible à la phase

Les expériences optiques sensibles à la phase sont à la base d'innombrables découvertes fondamentales [59, 60] et de technologies optiques [61, 62]. Les plus courantes sont les interféromètres, tels que les dispositifs de Michelson ou de Mach-Zehnder, qui utilisent l'interférence optique de deux trajectoires spatiales pour détecter de légères variations de la phase relative entre les trajectoires. Comme simple démonstration de la souplesse de définition des fonctions de coût par différenciation automatique, nous ciblons la conception de systèmes optiques sensibles à la phase. Un élément déphaseur (PS) est placé dans le système initial, et ne peut être retiré ou échangé pendant l'optimisation de la topologie, mais de nouveaux éléments de système peuvent être ajoutés autour de cet élément déphaseur. La fonction de coût est proportionnelle à la sensibilité de la puissance optique de sortie à ce déphasage, $F_G = -|\partial P_{\text{avg}}/\partial x_\varphi|$, comme défini dans l'équation S.5. Le système proposé par l'`ASOPE`, illustré à la figure 4.3 dans le texte principal, est une systeme interférométrique de type Mach-Zehnder. Les paramètres de ce système suivent également les techniques expérimentales courantes pour optimiser la sensibilité de phase dans les interféromètres : la puissance optique du laser à onde continue est à sa puissance maximale et l'atténuateur optique dans le second bras de l'interféromètre a une perte égale à la perte de l'élément de déphasage. Cette démonstration de la

preuve de concept pourrait être étendue à la recherche de systèmes optiques ayant des propriétés telles qu'une grande sensibilité, ou à la recherche de systèmes très stables aux perturbations.

Conclusions

En conclusion, j'ai développé et démontré des outils algorithmiques pour la conception inverse de systèmes de traitement de signaux optiques, et j'ai montré que ces méthodes peuvent reproduire des techniques optiques connues. Cette suite de conception devrait permettre de concevoir rapidement des systèmes optiques nouveaux, divers et performants. Dans l'algorithme démontré, les systèmes optiques sont représentés par des graphes de calcul, les arêtes et les sommets représentant des composants optiques paramétrés. Pour produire de nouveaux systèmes optiques, les opérateurs de mutation de graphe ajoutent/suppriment/changent de façon itérative les composants dans le graphe ; où la différenciation automatique et les algorithmes heuristiques sont utilisés pour optimiser les paramètres des composants et pour analyser leur sensibilité. Enfin, un ensemble topologique de systèmes optiques diverses et performantes est mis à jour tout au long de la recherche et renvoyé à l'utilisateur pour une analyse plus approfondie.

Actuellement, les modèles utilisés dans l'`ASOPE` sont limités dans la mesure où les interactions optiques non linéaires ne sont pas prises en compte et où les boucles de système ne sont pas autorisées. En particulier, les composants non-linéaire et dispersif qui nécessitent des modèles plus sophistiqués pour être simulés avec précisionne font pas partie de la bibliothèque de composants (par exemple, les fibres hautement non linéaires). Pour disposer d'un modèle précis de ces composants, il faudrait, par exemple, utiliser la méthode de Fourier à pas fractionné [63], ce qui augmenterait les ressources de calcul nécessaires à la convergence de l'ASOPE. De même, la simulation de boucles optiques nécessite des représentations plus complexes d'un système optique, ou des méthodes plus complexes pour calculer le résultat de la propagation [64, 65]. Les travaux futurs porteront sur la simulation de ces composants et boucles complexes, en déterminant comment les modéliser efficacement et comment allouer de manière optimale les ressources de calcul.

Les améliorations futures prometteuses de l'algorithme et du logiciel développé comprennent, par exemple : l'utilisation de méthodes de différenciation automatique avancées, la mise en œuvre de nouveaux concepts d'optimisation topologique [31], et le développement de nouvelles méthodes pour guider le changement topologique à partir, par exemple, de dérivés de fonctions de coût. Tout d'abord, des projets à source ouverte, principalement dans le domaine de l'apprentissage automatique, fournissent la fonctionnalité de simulation de modèles et de calcul de dérivés sur des unités de coprocesseurs (par exemple, GPU ou TPU), qui peuvent tirer parti d'un parallélisme de données énorme pour un calcul efficace (voir l'annexe A.1 pour une discussion plus approfondie sur les améliorations possibles des méthodes de différenciation automatique). Cela pourrait permettre une accélération significative de la durée de calcul et la possibilité d'optimiser et d'analyser des systèmes plus complexes. Deuxièmement, les concepts d'autres implémentations d'optimisation des

graphes pourraient être adaptés pour être utilisés dans le contexte de l'ASOPE. En particulier, les concepts de la neuro-évolution (trouver des architectures de réseaux neuronaux artificiels optimales) tels que la spéciation et les croisements de graphes devraient être explorés et adaptés [31]. Enfin, d'autres méthodes d'utilisation de l'analyse de sensibilité pour informer l'optimisation topologique, telles que les probabilités de mutation guidées par la matrice hessienne, devraient être explorées. J'ai présenté une méthode pour informer de l'élimination des composants optiques via les sensibilités des paramètres, cependant, d'autres liens et mesures devraient être étudiés. Par exemple, la matrice hessienne indique les composants qui sont étroitement couplés entre eux et dans quelle mesure ils influencent les performances du système; ainsi, les paires de composants hautement couplés et très sensibles pourraient être mises en évidence et avoir plus de chances de rester dans l'évolution du système.

L'utilisation de la conception inverse au niveau du système pourrait trouver de nombreuses possibilités d'utilisation future. Trois de ces orientations qui présentent un intérêt immédiat sont les suivantes: cibler de nouveaux objectifs, y compris des degrés de liberté optiques nouveaux/différents et étudier la perspective d'une co-conception optique et électronique. L'utilisation principale de l'ASOPE est de l'appliquer à de nouveaux problèmes et tâches. Grâce à l'algorithme, à la bibliothèque de composants, aux routines d'optimisation des paramètres et de la topologie, à l'analyse de sensibilité et aux outils de visualisation développés, les nouveaux objectifs et les fonctionnalités souhaitées peuvent être rapidement prototypés. En particulier, l'optimisation des mesures basées sur le bruit (par exemple, signal-à-bruit, gamme dynamique) pourrait trouver des applications dans les télécommunications et la détection, où le bruit peut être particulièrement préjudiciable. Toutefois, il convient d'étudier plus profondément la manière dont l'injection de bruit dans les simulations peut affecter la convergence d'optimisation et l'analyse de sensibilité. D'autres mesures basées sur des considérations logistiques plutôt que sur le comportement du système, telles que son coût, le poids mécanique ou la stabilité, pourraient être incluses. Ces paramètres pourraient même, potentiellement, être utilisés comme objectifs de conception secondaires dans l'optimisation multi-objectifs. Une deuxième orientation des travaux futurs consisterait à ajouter différents degrés de liberté optique, tels que la polarisation, et à utiliser des modèles de composants intégrés ou en espace libre. Comme l'ensemble ASOPE a été construit dans une structure modulaire pour permettre des ajouts simples et flexibles de nouvelles fonctions et/ou composants objectifs (voir annexe A.3), la topologie et les techniques d'optimisation des paramètres peuvent être facilement étendues à ces nouveaux types de systèmes optiques. Une troisième direction pour les travaux futurs implique la co-conception de systèmes optiques et électroniques. Cela pourrait avoir un impact sur la conception de la photonique intégrée, où les circuits électroniques et les circuits de guides d'ondes optiques sont fabriqués de manière monolithique sur une seule puce, les signaux électroniques et optiques se propageant conjointement. Grâce à des travaux plus approfondis, les méthodes de conception inverse présentées ici pourraient être utilisées pour découvrir de nouvelles conceptions de puces photoniques intégrées pour une variété d'utilisations cibles.

En résumé, les travaux présentés dans cette thèse démontrent une nouvelle méthode de conception des systèmes optiques. J'ai développé un certain nombre d'outils avec des implémentations logicielles, y compris l'analyse de sensibilité, l'optimisation des paramètres et l'optimisation de la topologie, qui sont facilement utilisables pour trouver des implémentations non intuitives et innovantes de systèmes optiques futurs.

Table of contents

List of figures

List of tables

Notation and abbreviations

Abbreviations

AD	Automatic differentiation
BFGS	Broyden Fletcher Goldfarb Shanno algorithm
BS	Beam-splitter
CMA	Covariance matrix adaptation algorithm
CW	Continuous wave laser
DF	Dispersive fiber
DL	Delay line
EDFA	Erbium-doped fiber amplifier
FFT	Fast-Fourier transform
GA	Genetic algorithm
GD	Gradient descent
GED	Graph Edit Distance
IFFT	Inverse Fast-Fourier transform
IM	Intensity modulator
PD	Photodiode
PL	Pulsed laser
PM	Phase modulator
PSO	Particle swarm optimization
WS	Waveshaper

Mathematic Notation

$\mathbb{R}$	Real numbers
$\mathbb{Z}$	Integers
$\mathbb{C}$	Complex numbers
a	Scalar
$\mathbf{a}$	Vector
$\mathbf{A}$	Matrix
$G(V, E)$	Graph
$\mathcal{U}$	Uniform distribution
$\mathcal{N}$	Normal distribution
$\mathcal{F}$	Fourier transfrom
$\mathcal{F}^{-1}$	Inverse Fourier transform

Physics Symbols

ω	Optical angular frequency
c	Speed of light in vacuum
Ψ	Optical field, time-domain
$\tilde{\Psi}$	Optical field, frequency-domain

CHAPTER 1

Introduction and motivation

The design of novel optical systems has underpinned every step of scientific progress in the field of optics, and has led to countless new technologies [1–6]. Scientists and engineers routinely harness various optical phenomena and principles to build every-improving tools and technologies based on light; from telecommunications, to sensing, entertainment, and beyond. These optical systems are designed, built, and tested to meet specific, desired functionalities – be it nanophotonic components for high-efficiency optical routing [1], lens systems for imaging setups [3], or experiments to test the fundamental understanding of quantum mechanics [7]. However, designing complex optical systems can be time-consuming, be unintuitive, and require expert-level domain knowledge. In general, this process requires a designer to propose and analyze each possible solution, either experimentally, analytically, via simulation, or through a combination of all of these approaches. Such methods, termed *forward design*, are, however, restrictive – they require an experienced designer to test each candidate solution, are constricted by the intuition and domain knowledge of the designer, and, thus, are often limited to experts. These limitations can restrict the scope and breadth of new solutions – even preventing us from ever solving the most counterintuitive problems – and can be time-consuming in converging on effective designs.

Inverse design, however, mitigates some of these challenges and accelerates the development of new systems. Inverse design refers to a broad class of techniques and methods for algorithmically finding new solutions to address a problem. It is used widely in automotive and aerospace engineering [8–10], electronics [11], architecture [12, 13], biological processing [14], chemical synthesis [15], and beyond. Also known as generative, parametric, or algorithmic design, the exact tools and techniques may differ between disciplines in terminology and implementation – but common principles and algorithms span across fields of use. While some concepts of inverse design have existed for hundreds of years, the field as it exists today began in the second half of the twentieth century with the advent of digital computing and numerical optimization [1]. Steady improvements in computational resources, algorithms, and infrastructures have provided the necessary support for algorithmic design to proliferate across disciplines. Inverse design techniques are an attractive method as digital computers can often test solutions much faster than a human and they are not limited to the (potentially restrictive) intuition of a human designer. Inverse design methods aim

to parameterize possible systems and use computational resources and algorithms to quickly and intelligently explore and evaluate promising new solutions. In forward design the onus is on the designer to envision and test any candidate solution; while in inverse design the onus is instead on the designer to specify how to numerically evaluate the performance of a solution and build an algorithm which can automatically produce new systems. The designer specifies a target behaviour and performance, and then allows the algorithm to do the 'heavy lifting' of exploring and evaluating many new possible solutions – reporting back to the end-user the best candidate(s). The challenges in inverse design are different than in forward design, as the difficulty shifts instead to how to create algorithms and mathematical models which are sufficiently adept and general to find useful solutions. This includes how to parameterize and represent a design, how to evaluate its quality and performance, and how to then change the design in order to improve said performance. These considerations – which I term representation, evaluation, perturbation, and optimization – are the pillars of an inverse design approach and are highly problem-dependent and domain-specific (see Table 1.1).

Inverse design is used in numerous fields, and has been steadily expanding in its use since the advent of digital computing. In structural and mechanical engineering, inverse design methods are routinely used to determine optimal structures for industrial and consumer products, for example in additive manufacturing (3D printing) [8], structures with desired radar signatures and low noise [9], as well as complex geometries and high-performance building structures [10]. In architecture, generative design began with the creation of efficient hospital layouts [12] in the 1970's, with these techniques steadily progressing and now available in major architecture design software packages [13], allowing designers and architects to take advantage of computational resources to find optimal building layouts, structures, and shapes with minimal human input. In chemistry, computational approaches to inverse design are ushering in a new era of chemical design tools, providing novel resources to conceptualize prospective drugs, synthetic routes to organic compounds, and optimization of photovoltaics and batteries [15]. In particular, as the manufacturing capabilities of a discipline mature, the methods of design grow in parallel to include more sophisticated, computer-aided design approaches. A common trait of these inverse design methods is to iteratively combine simple elements to build and evaluate new and complex systems; for example, engineering complex molecules from atoms in inverse chemical design [15], or creating meshes of different material types in additive manufacturing [8]. Combining these building blocks gives rise to complex interactions and, thus, new functionalities or emergent behaviour. Inverse design, therefore, aims to efficiently explore these complex interactions and automatically discover new systems and designs.

In optics, inverse design is utilized in a number of sub-disciplines to great success. First and foremost, it is used in creating lens systems for imaging, astronomy, and entertainment [16–20]. These demonstrations are the earliest and most-widespread utilization of inverse design within optics. More recently, the realization of high-performance nanophotonic and metamaterial devices for telecommunications and sensing [1, 21–25] has been improved with the adoption of such algorith-

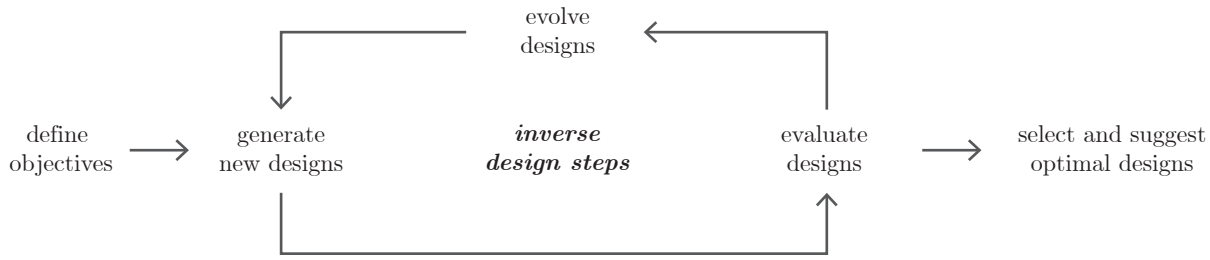


Figure 1.1 – Basic steps of inverse design. First, the target functionality is defined which provides in the form of an objective function. Then, new designs are iteratively generated, evaluated, and evolved until some stopping criterion is met and the final, optimal designs are returned to the user for further analysis and consideration.

Table 1.1 – Pillars of inverse design.

Representation	**Evaluation**	**Perturbation**	**Optimization**
provide a model of a solution	estimate performance of a solution	understand performance of similar solutions	find high-performing solutions
$\mathbf{x}$	$F(\mathbf{x})$	$F(\mathbf{x}+\delta\mathbf{x})$	$\min_{\mathbf{x}} F(\mathbf{x})$

mic techniques. Additionally, related concepts have also been demonstrated in the discovery of new quantum optics experiments to produce complex, entangled states of light [5, 7, 26].

Inverse design methods, however, stand to find further application in the design of ultrafast optical processing systems. While parameter optimization has been used in laser engineering [27], telecommunications [28], ultrafast optics [29], and other related fields, inverse design on the *system-level* in such optical disciplines has not yet been widely explored. When designing an optical system for a specific application, one must consider three main things:

- The optical components which comprise the system
- The system topology, or order and direction of optical propagation
- The operational parameters of these components

Firstly, in ultrafast optical processing and related fields, a common set of optical components and effects are used towards numerous different technologies. These include, for example, components that rely on chromatic dispersion, nonlinearity, interference, loss, and gain – each dependent on the degrees-of-freedom of the optical field (temporal, spectral, polarization, etc.). Various combinations of these optical elements underpin countless technologies and commercial products, and are realized through photonic components such as: optical sources, modulators, optical fibers, amplifiers, spectral filters, photodiodes, and so on. As these optical phenomena (e.g., dispersion, modulation, interference, absorption, etc.) do not generally commute with one another, the order in which they act on a propagating electromagnetic field significantly alters a system's overall output. This is demonstrated by the simple setups in Fig. 1.2 – the components between the top and bottom

setups are the same, except the ordering of the modulator and the optical spectrum shaper are switched. The top system first applies a time-dependent phase modulation, followed by frequency-dependent phase transfer function, while in the bottom system these operations are reversed. As seen in the system outputs (right column), these discrete topological changes result in a drastically different output optical field.

As such, the *topology* of the optical system allows many different functionalities with the same optical components. With the term topology, we refer to the ordering of the components in time and space, or in other words, how the components are connected together via propagation paths (e.g., waveguides, optical fiber patch-cords, or free-space links). Systems can be serial, comprised of a single spatial path with components connected one-after-another. They may also contain parallel optical propagation paths, which give rise to interference effects from interferometric-type optical circuits. Systems can range from relatively simple topologies containing a few components connected serially, to complex systems comprised of hundreds of parallel optical propagation paths giving rise to desired interference effects [30].

The final consideration in designing optical systems are the *component parameters*, which dictate how each component manipulates light during propagation. These parameters are the tunable values on a given optical component that define the operational regime of that element, e.g., the modulation depth of a modulator, fiber length of a dispersive fiber, or gain value of an optical amplifier. These component parameters must be selected based on the desired functionality of the system, and complex interactions between these parameters can make understanding and selecting these parameters challenging. Further, some of these component parameters require a great deal of precision when building the system, as perturbations to the parameters could significantly change performance. It is beneficial and often necessary, therefore, to optimize the performance of the system towards the desired functionality, as well as understand its sensitivity to slight changes in the design parameters. Therefore, inverse design of optical setups on the system level requires the consideration of all three of these elements; the optical components, the topology of the system, and the component parameters.

The main focus of this work is to develop computational tools for the inverse design of ultrafast optical processing systems. To accomplish this task, first, a library of component models is developed, with compact, parameterized models of optical elements such as sources, modulators, dispersive elements, amplifiers, etc. (see Appendix A.4). Next, a method for testing the sensitivity of systems to perturbations is implemented, which uses automatic differentiation to determine the dependence of the system performance on each component and parameter. Finally, two optimization routines are used to find optical systems that perform well on the desired task – the first to search through system topologies and the second to optimize the component parameters. These optimization routines are coupled, such that for each proposed topology, an optimal set of operational parameters is also found. These tools, taken together, comprise a new method for the design of optical systems for a broad variety of desired goals. In this work, I focus on systems

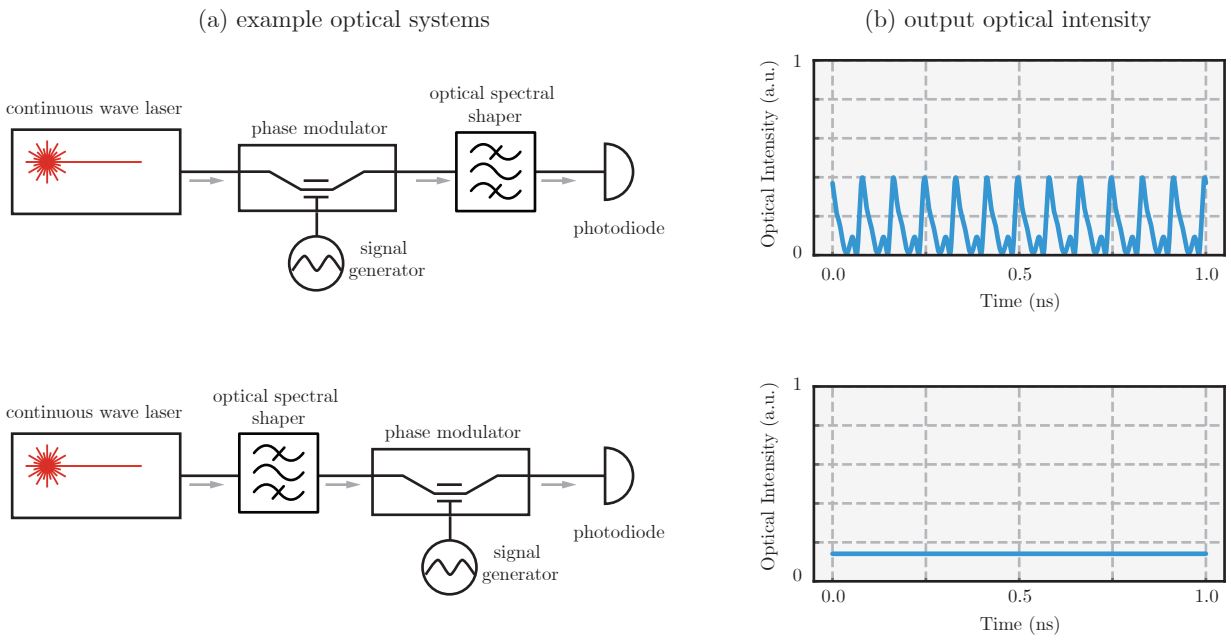


Figure 1.2 – Example of non-commutativity between optical components. Between the top row and the bottom row, the order of the phase modulator and optical spectral shaper is exchanged, with the other components, all operating parameters, and overall topology of the optical system unchanged. The resultant optical fields (right column) between the two setups, however, have vastly different temporal shapes (and different spectra). This change can be thought of as the non-commutativity of the transfer function of the optical components.

which manipulate the time-frequency degrees-of-freedom in fiber-based components, however, the methods are extendable to other degrees-of-freedom (e.g. polarization, angular momentum, etc.) and photonic platforms (e.g. integrated optics, free-space optics, etc.).

Structure of this thesis

The presented thesis is structured as follows:

Chapter 1: Discusses the motivation and objectives for the presented work on the inverse design of ultrafast optical systems.

Chapter 2: Reviews common techniques and tools used in inverse design approaches across disciplines, followed by a review of inverse design in optics. To compare between disparate fields of study, I present them in the context of four main elements used in inverse design: representation, evaluation, perturbation, and optimization (see Table 1.1). These four pillars will be used to frame techniques between fields, as well as contextualize the novel contributions of this work.

Chapter 3: Presents the main, novel work of this project – algorithmic tools for the inverse design of ultrafast optical systems, dubbed *Automated Search for Optical Processing Experiments*,

or `ASOPE`. The algorithm enables the design of optical systems with desired functionality by iteratively building, simulating, and evaluating systems. This includes representing optical systems as computational graphs, automatically differentiating the system with respect to design parameters, and using topology optimization to explore new optical systems.

Chapter 4: Demonstrates `ASOPE` for the design of different optical systems and tests the performance of various algorithms and techniques. The inverse design of three types of optical systems is presented: multiplication/division of pulse repetition rates, arbitrary waveform generation, and phase-sensitive systems. Sensitivity analysis, parameter optimization, and topology optimization techniques are then benchmarked and compared.

Chapter 5: Discusses the main conclusions of the work and future research directions.

Appendix A: Contains further discussions on automatic differentiation and sensitivity analysis, as well as a brief white-paper overview of the `ASOPE` software package (Appendix A.3) and descriptions of the optical component models currently developed (Appendix A.4).

CHAPTER 2

Review of inverse design methods

Inverse design methods share similar techniques across their broad and disparate use-cases. Common methods may include: mathematically and numerically formulating potential solutions to an inverse design problem, performing sensitivity analysis to understand cause-effect relationships, and using optimization routines to search for the best-performing designs. This Chapter provides a review of these tools, with the aim of using a broad and general formulation, discussing shared underlying concepts, and comparing the relative advantages of related tools.

2.1 Representation and evaluation

Inverse design is necessarily a computational approach. This means that in order to use an inverse method towards a certain problem, the system and its desired performance must first be formulated mathematically and simulated numerically. The chosen representation, or parameterization, encompasses the key details of a candidate solution, which can then be simulated to determine design performance. Possible designs must thus be represented via an abstract mathematical object that can be optimized, for example a scalar value, a vector, a matrix, or a graph. As inverse design is used to create new physical systems – e.g., a building layout, a chemical structure, an optical component – there are often natural choices of representation to use. Throughout this thesis, the representation of a system is denoted as $\mathbf{x}$, and the set of all possible values of $\mathbf{x}$ is termed the *solution* or *design space*. The representation $\mathbf{x}$ often corresponds to a vector, $\mathbf{x} \in \mathbb{R}^n$, in many problems, but more complex mathematic objects are often used (e.g. a graph, as used in `ASOPE` and discussed in Chapter 3). It is the goal of inverse design to explore this solution space to find high-performing regions.

Choosing a function to evaluate a solution can be challenging, yet is foundationally important. Evaluation is the process of simulating how a physical system will behave with a set of parameters $\mathbf{x}$, and providing a quantitative estimate of how well the solution will perform on the desired task. This, again, is entirely dependent on the field and problem being studied. Throughout this document, the evaluation of a candidate solution $\mathbf{x}$ is denoted as $F(\mathbf{x})$ and (as is generally true in inverse

design and optimization) a solution is evaluated to a scalar value, $F(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$. If the desired functionality is not well encapsulated by the evaluation function, then optimizing it will provide no benefit.

Evaluation functions could, for example, be a physical quantity (e.g. optical collection efficiency or aberrations, as in lens design [19]) or a similarity metric between a target output and a generated output (e.g. fidelity between a targeted quantum state and a generated quantum state [5, 7], or between a targeted and a generated temporal envelope of an optical field, see Sec. 3.1.2 and 4.1). Objective functions may also be heuristic, which provide an approximate cost and are based off of intuition of the designer and work well with the optimization routine. In these cases, the absolute values may not convey information, but rather the relative scores between solutions are useful in indicating which perform better.

The set of all possible objective values is termed the *objective space*; thus $F(\mathbf{x})$ maps from the solution space to the objective space. Evaluation functions are referred by two standards in literature, depending on whether the author aims to maximize or minimize the objective value. When a large value of $F(\mathbf{x})$ indicates a 'good' solution, it is referred to as a fitness function or figure-of-merit. Conversely, when a small value of $F(\mathbf{x})$ indicates a 'good' solution it is known as a cost function or loss function. In the work outlined in this thesis, $F(\mathbf{x})$ is standardized to be a cost function (i.e., minimizing $F(\mathbf{x})$ improves the design), and referred to as an objective or cost function.

In the context of optics, the choice of representation and evaluation is related to which physical formalism used, e.g., ray optics, wave optics, quantum optics, or approximating Maxwell's equations to simulate the electromagnetic field. Different formalisms are better suited to different contexts and problems. When designing a lens system, for example, ray tracing is most commonly used [20]. Alternatively, when optimizing nanophotonic structures, a much more detailed simulation is required and Maxwell's equations must be solved using, for example, finite-difference time-domain techniques [66]. Generally, there is a trade-off between the accuracy of a simulation and the required computation time and/or resources, and different representations fall into different regimes of this trade-off. Using the example of optics formalisms, while solving Maxwell's equations is the most accurate method, it is also computationally prohibitive for many applications and instead a more compact model, such as ray optics, is used – which may have more approximations and assumptions in exchange for significantly reduced computational requirements. Within each of these formalisms, decisions must be made as to the complexity of the simulation and which physical effects must be considered. As a further example, in Ref. [15], eight different representations of a chemical molecule are discussed – each with benefits and drawbacks in the context of inverse design. Some representations restrict the solution space in such a way that physically realizable solutions (i.e. real molecules) are not accessible with a representation choice, or there is large degeneracy in the solutions (many different solutions map to the same physical system, due to e.g. symmetry). Similar challenges, constraints, and advantages exist in each use-case of inverse methods when choosing a

representation. As such, the choice of representing a physical system with an abstract mathematic structure and numerical simulation is the first step in developing an inverse design approach.

2.2 Perturbation

Perturbation, or sensitivity analysis, estimates how small changes to the parameters may effect the design's performance. Sensitivity analysis seeks to explore the local function landscape around a one set of design parameters, $\mathbf{x}^*$. This can indicate if $\mathbf{x}^*$ is a local extreme point (i.e., minimum, maximum, or saddle point), as well as how quickly the performance changes as the parameters are perturbed from $\mathbf{x}^*$. Sensitivity analysis indicates which parameters are the most important, can guide the optimization routine, and can provide insight to the designer about parameter tolerances and considerations for the experimental realization of a system. For example, via sensitivity analysis one can determine that perturbing one parameter, x_i, heavily impacts the performance of a design and, as a consequence, in a laboratory environment this parameter must be precisely controlled and/or be kept stable during operation. This places bounds on the fabrication tolerances of the systems and the repeatability of the fabrication process, as highly sensitive parameters must be well-controlled to that ensure the experimentally realized design will perform well.

Broadly, there are two main classes of sensitivity analysis: *sampling methods*, which use a probabilistic approach to sample the function around a center point and calculate statistical quantities (such as the moments of a function), and *gradient-based methods*, which use first- and higher-order derivative terms to estimate how quickly the objective function changes along different parameter axes [67, 68]. A key goal of sensitivity analysis is to quantitatively measure the effect each parameter x_i has on the objective function $F(\mathbf{x})$, and to order the parameters according to their relative impact. Exact quantitative values of sensitivity will vary between methods, however, they should ideally provide similar characterization of the most important parameters (as in Fig. 2.1, bottom row).

2.2.1 Sampling methods

Sampling methods, or Monte Carlo techniques, for sensitivity analysis encompass a number of techniques which treat $\mathbf{x}$ as a random variable. A set of sample solutions, $\{\mathbf{x}^{(j)}\}$, are drawn from a distribution, $\Pr(\mathbf{x})$, and the variance and/or other statistical quantities are computed from the set of evaluations, $\{F(\mathbf{x}^{(j)})\}$. These techniques are attractive as they treat $F(\mathbf{x})$ as a black-box function – not requiring any further information about $F(\mathbf{x})$, such as its gradient. The parameters of $\mathbf{x}$ are considered as independent random variables, with a probability distribution,

$$\Pr(\mathbf{x}) = \Pr(x_1)\Pr(x_2)\Pr(x_3)\dots\Pr(x_n) \tag{2.1}$$

For each individual parameter, the choice of probability function $\Pr(x_i)$ is guided by domain knowledge or by fitting experimental data. These probability distributions are most commonly a uniform distribution within an upper and lower bound, $\mathcal{U}(b_{\text{lower}}, b_{\text{upper}})$, or normal distribution $\mathcal{N}(\mu, \sigma)$, with

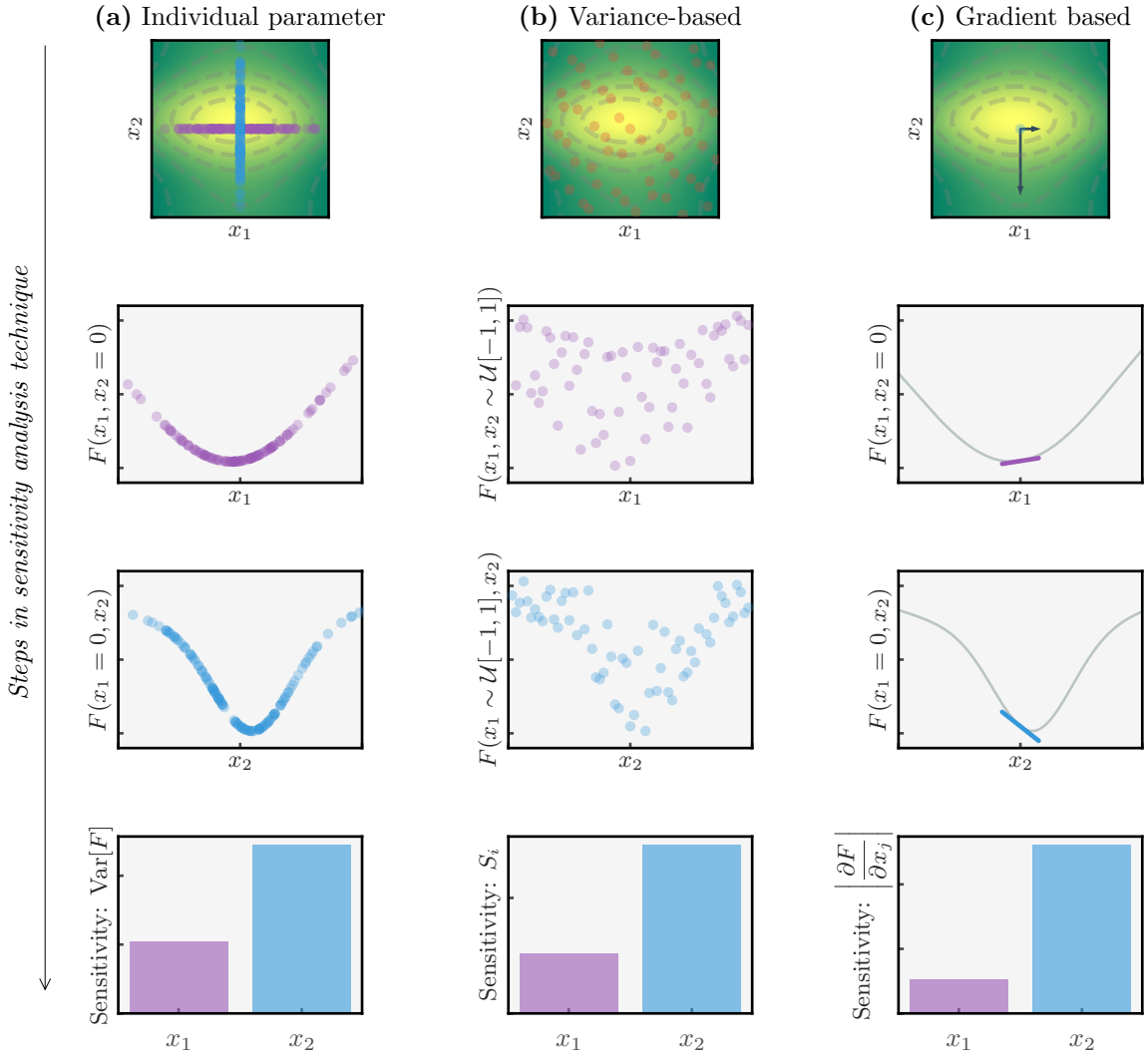


Figure 2.1 – Examples of sensitivity analysis techniques. **(a)** Sequential Monte Carlo sensitivity analysis around the central point of $\mathbf{x}^* = [0, 0]$, with the sensitivity of each parameter calculated with Eqn. 2.2. **(b)** Variance-based analysis using first-order variance indices (Sobol indices) as described in [68]. $\sim \mathcal{U}$ refers to drawing the parameter from a uniform distribution. **(c)** Gradient-based sensitivity analysis, where black arrows (top row) and solid blue/purple lines (middle rows) are the gradients at the point $\mathbf{x}^* = (0, 0)$. The bottom row of a-c demonstrate that all three methods, while exact quantities vary, provide a similar characterization of the sensitivity of $F(\mathbf{x})$ on x_1 and x_2. Code for these plots can be found here.

width σ and mean μ. These bounds or widths can represent physical limitations in manufacturing to better understand, for example, design tolerances.

The most straightforward method for estimating the impact that each parameter has is by changing them one at a time (Fig. 2.1a). To estimate the sensitivity of the function $F(\mathbf{x})$ around a point $\mathbf{x}^*$ we sample the function many times while only changing a single parameter. We denote $\mathbf{x}^*_{x_i \sim \Pr(x_i)}$ to be the vector equal to $\mathbf{x}^*$, except in the parameter x_i which is instead a random variable drawn from the distribution $\Pr(x_i)$. The variance of the evaluations,

$$\mathrm{Var}[F(\mathbf{x}^*_{x_i \sim \Pr(x_i)})] \tag{2.2}$$

is estimated by sampling many times and provides an estimate of the sensitivity in the parameter x_i. This method, while simple and easy to implement, does not account for any interaction of parameters and may require many calculations of F to generate accurate statistics.

More sophisticated sampling methods and sensitivity estimators that are widely used include the *first-order sensitivity indices* or Sobol indices [68]. These, similarly, consider the effect of each individual parameter, but average over the effects of other parameters and are not sampled only on the parameter axes, but rather use a quasi-random sequence of parameter sets (using, e.g., Sobol sequences, Halton sequences, or Latin hypercube sampling). Fig. 2.1b demonstrates the first-order sensitivity indices of x_1 and x_2, following the approach in [68]. Sampling methods which can, instead, indicate interactions between parameters include, e.g., the *total effect indices*. Ref. [68] provides a detailed overview of these techniques. Advantageously, these methods treat F as a black box function, which is useful when no further information about the objective function, such as its gradient, is available.

2.2.2 Gradient-based methods

Gradient-based methods of sensitivity analysis use derivatives of the objective function $F(\mathbf{x})$ at a central point, $\mathbf{x}^*$, to estimate the effect of each parameter x_i. Around a point $\mathbf{x}^*$, the function can be approximated by taking a truncated Taylor series. For a multivariate function $F(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$, the second-order approximation [69] is

$$F(\mathbf{x}^* + \delta\mathbf{x}) \approx F(\mathbf{x}^*) + \delta\mathbf{x}^{\mathrm{T}}[\nabla F(\mathbf{x}^*)] + \delta\mathbf{x}^{\mathrm{T}}[\mathbf{H}(\mathbf{x}^*)]\delta\mathbf{x} \tag{2.3}$$

where $\delta\mathbf{x}$ is a small deviation, ∇F is the gradient vector of F, and $\mathbf{H}$ is the *Hessian matrix* of size $n \times n$ – given as,

$$\nabla_i F = \frac{\partial F}{\partial x_i}, \qquad H_{i,j} = \frac{\partial^2 F}{\partial x_i \partial x_j} \tag{2.4}$$

This means that perturbing the parameters by a small amount, $\delta\mathbf{x}$, from the central point $\mathbf{x}^*$ will result in a change in $F(\mathbf{x})$ of approximately (to the second order) $\delta\mathbf{x}^{\mathrm{T}}[\nabla F(\mathbf{x}^*)] + \delta\mathbf{x}^{\mathrm{T}}[H(\mathbf{x}^*)]\delta\mathbf{x}$ – which can therefore serve as a measure of sensitivity. The gradient terms $\nabla_i F$ are often called the 'sensitivities' and measure the effect each parameter x_i has on the output (Fig. 2.1c) – but does not include the effect that any interactions between parameters may have. However, the Hessian, being a higher-order term, does contain information about the interactions between parameters. Therefore the Hessian terms $H_{i,j}$, indicate how parameters x_i and x_j may –in combination– influence $F(\mathbf{x})$. In very high-dimensional problems, efficiently calculating the Hessian can be computationally challenging, in which case sensitivity analysis may consider only the first-order derivative or smaller submatrices of the Hessian.

The central requirement of using gradient-based sensitivity analysis methods is the ability to calculate $\nabla F(\mathbf{x})$, and (if desired or required) higher order derivatives as well. This necessitates that the parameters are treated as continuous and the function is differentiable at $\mathbf{x}^*$. Computing the derivatives can be done via analytic differentiation (i.e. using a pen and paper), numerical differentiation (i.e. using finite difference approximation), symbolic differentiation (i.e. using computer-algebra), or *automatic differentiation* [70]. Here, we focus on automatic differentiation due to its efficiency, flexibility, and ease-of-use.

Automatic differentiation

Automatic differentiation (c.f. algorithmic differentiation and computational differentiation) is a family of techniques which first appeared in the 1960's to calculate the derivatives of a function expressed as a computer program [71]. Automatic differentiation is, in essence, the continued application of the chain rule in calculus. When a function $\mathbf{y} = F(\mathbf{x})$ is implemented as a computer program (a list of instructions), the calculation is composed of elementary operations: such as addition, multiplication, exponentiation, and trigonometric functions. Each of these elementary operations has a known derivative, which is also described by elementary operations. By composing these known derivatives together, it is possible to automatically compile a computer program which calculates the derivative of the function. Implementations of automatic differentiation are available in many programming languages and allow for derivatives to be calculated with minimal programming overhead. Fig. 2.2 outlines the concept of automatic differentiation with an example function $F : \mathbb{R} \to \mathbb{R}$. In the more general case of multiple input and output variables, the automatic differentiation uses Jacobian vector products to trace the function derivative through the calculation [72].

Automatic differentiation has two modes of operation: forward mode and reverse mode. Forward mode fixes the dependent variable(s), $\mathbf{x}$, and traverses the graph in the forward direction, while reverse mode fixes the dependent variable(s), $F(\mathbf{x})$, and traverses the computational graph backwards (see Fig. 2.2). While these two methods will give the same evaluation, they differ in the

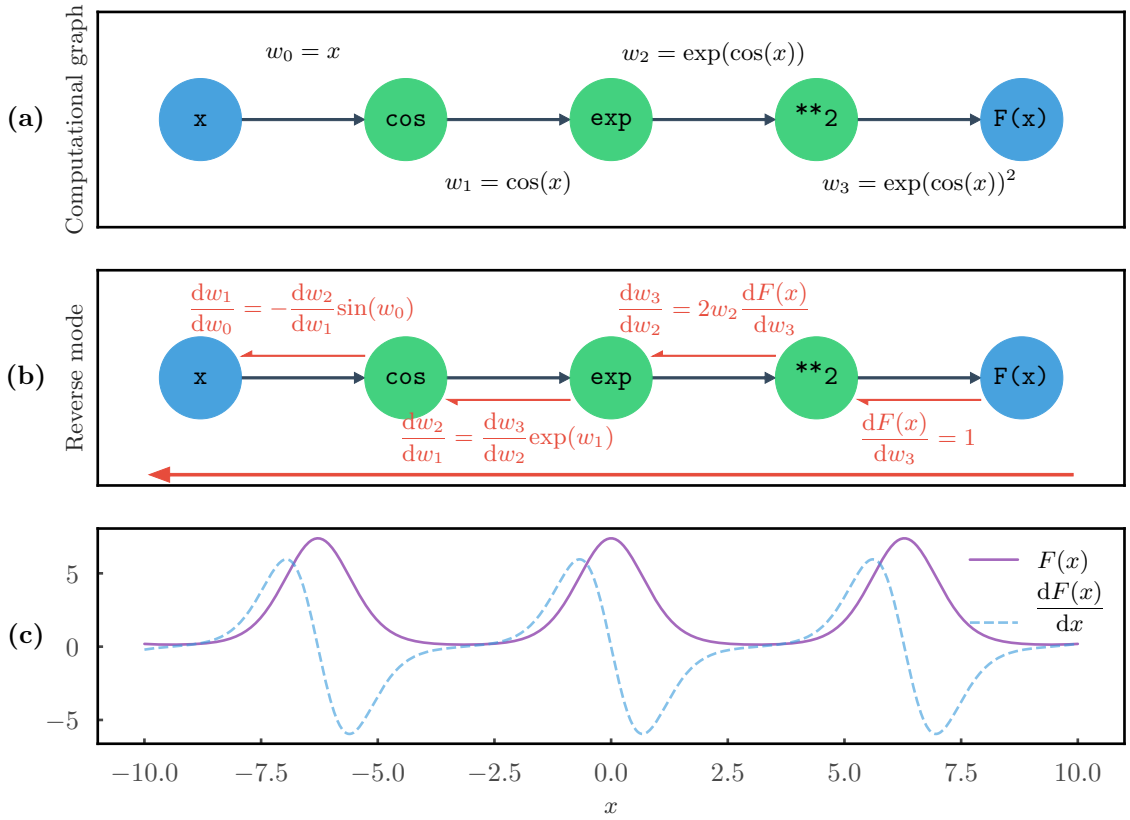


Figure 2.2 – Automatic differentiation of an example function $y = F(x) = \exp(\cos(x))^2$. **(a)** Computational graph of the function, where the blue circles are the input and output scalar values and green circle are the elementary operations which compose the function. **(b)** Reverse mode automatic differentiation, with derivatives accumulated moving from right-to-left along the computational graph. **(c)** Plot of $F(x)$, along with its derivative calculated using automatic differentiation. Continued, higher-order derivatives are trivial to compute by the successive application of automatic differentiation. Code for these plots can be found here.

memory and time requirements: for a function $\mathbf{y} = F(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}^m$, if $n \ll m$ (i.e. less input variables than output variables) then forward mode differentiation is preferred, while for $n \gg m$ (i.e. more input variables than output variables) the reverse mode differentiation is preferred [70]. As optimization problems have $n \gg m$, reverse mode differentiation is generally used in this setting (using reverse mode differentiation to calculate the derivative of a cost function and minimizing). Because the derivatives are themselves composed of elementary functions, higher-order derivatives can continually be calculated with practically no overhead in programming.

Automatic differentiation is a powerful tool for design methods as systems can be modeled quickly and easily, and then compute derivatives for optimization and analysis with minimal programming overhead. It is capable of efficiently evaluating the derivatives to machine precision. This allows for fast prototyping of complex system where other differentiation methods, e.g., numeric, analytic, or symbolic, may become prohibitively costly or unstable. For example, in the field of machine learning, reverse-mode automatic differentiation is used to train deep neural networks, hugely complex functions comprised of millions of parameters and elementary functions. No other

method for calculating gradients of such networks is realistic, as the complexity of the functions is prohibitive. A number of automatic differentiation tools exist in a number of computing languages, including FORTRAN, C, Java, Matlab, and Python. In the machine learning community, Python has become an important and essential language, where a number of actively-developed tools for automatic differentiation exist – these include `autograd`, PyTorch, TensorFlow, JAX, among others [73–75]. See Appendix A.1 for further discussion.

2.3 Optimization

The final pillar of inverse design is the optimization routines used to search for the highest-performing solutions. A naive method to do so is via a brute force search of the function $F(\mathbf{x})$ – visiting all possible values of $\mathbf{x}$ or a sufficiently dense grid of solutions. However, this becomes computationally intractable very quickly, as the number of evaluations of $F(\mathbf{x})$ will increase exponentially with the number of free parameters; for example, an optimization problem with 50 parameters and only 10 points per parameter would require 10^{50} function evaluations to cover the entire search grid (if each sequential calculation of F required 1 ms, the required running time would correspond to many times the age of the universe). In contrast, an optimization routine is a guided exploration of the solution space, aiming to find better performing designs via successive iterations – resulting in significantly fewer required evaluations to find a high-performance solution. For a cost function, as in this work, optimizing a function aims to find local or global minimum point(s) under a set of constraints and bounds, or

$$\begin{aligned} \underset{\mathbf{x}}{\text{minimize}} \quad & F(\mathbf{x}) \\ \text{subject to} \quad & g_k(\mathbf{x}) = 0, \quad h_k(\mathbf{x}) \leq 0 \end{aligned} \tag{2.5}$$

where F is the evaluation function, $\mathbf{x}$ is a parameter vector, g_k is a set of equality constraints, and h_k is a set of inequality constraints. These equality and inequality constraints can enforce relationships between design parameters or bound their values. Bounded optimization is common when designing physical systems, as the physical parameters must fall within some experimentally accessible range. Constraints enforce necessary relationships between parameters, e.g., the bounds on x_i being dependent on the value of x_j, where $i \neq j$. These constraints may, for example, ensure that energy or momentum is conserved when changing the design parameters (see Appendix A.4). Constrained optimization problems can be transformed into an unconstrained problem using methods such as substitution or Lagrange multipliers [76].

A parameter set corresponds to a local minimum of $F(\mathbf{x})$ is denoted as $\mathbf{x}_{\text{min}}$ – such that $F(\mathbf{x}_{\text{min}}) \leq F(\mathbf{x}_{\text{min}} + \delta\mathbf{x})$ for any small deviation $\delta\mathbf{x}$. The global minimum of $F(\mathbf{x})$ is the point $\mathbf{x}_{\text{min}}$ where $F(\mathbf{x}_{\text{min}}) \leq F(\mathbf{x}) \ \forall \ \mathbf{x}$. In inverse design tasks, there is often no guarantee of finding a globally-optimal solution; instead, these algorithms often converge to local minima. Meta-heuristic

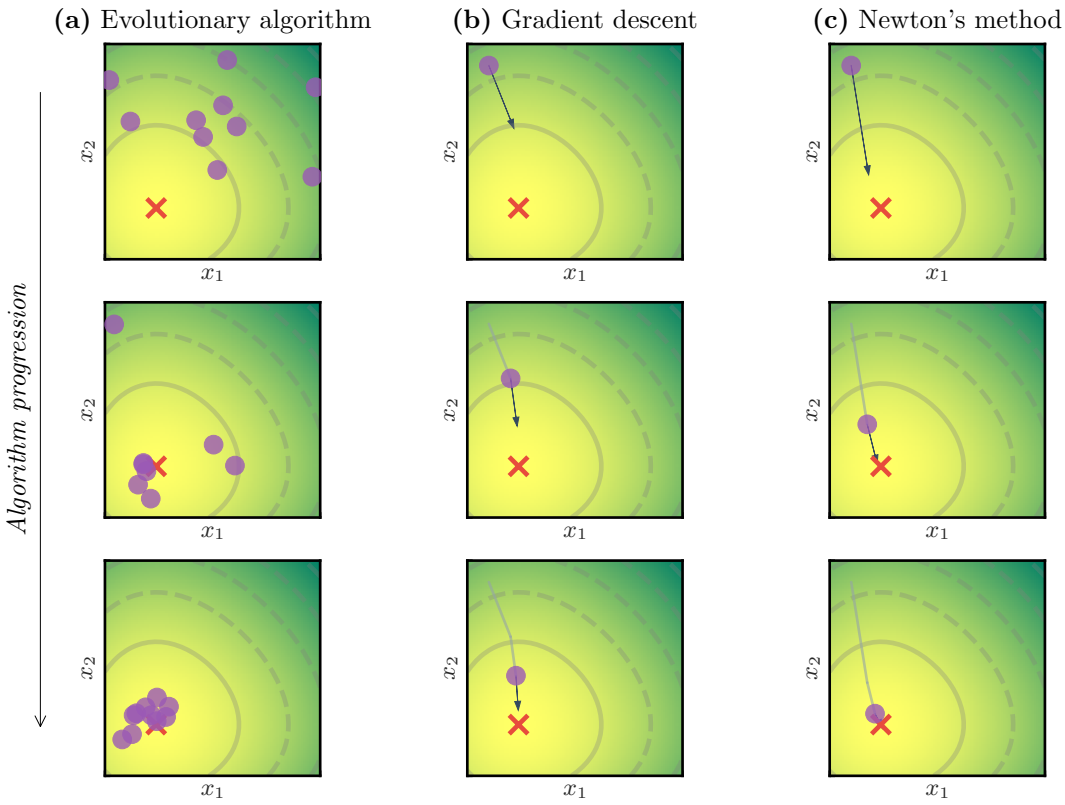


Figure 2.3 – Examples of optimization routines applied to a black-box function $F(x_1, x_2) : \mathbb{R}^2 \to \mathbb{R}$, with a local minimum denoted by the red cross markers and candidate solutions represented by purple circle markers. **(a)** Snapshots of an evolutionary algorithm (particle swarm algorithm) as it searches for a function minimum. As the algorithm progresses (top-to-bottom), it can be seen that the tested solutions (purple markers) 'swarm' to the minimum point. **(b)** Successive steps in a first-order gradient descent algorithm navigating to the minimum, with the current solution (purple marker) moving in the direction of steepest descent (black arrows, as evidenced by each arrow being perpendicular to the contour lines). **(c)** Successive steps of Newton's optimization method, which uses both the first- and second-order derivatives to update the current solution (purple marker) closer to the minimum point. While the gradient descent and Newton's method start at the same point in the top row, the update direction differs – this is due the second-order considerations of Newtons's method. The code for these plots, which contains simple implementations of a particle swarm optimization, gradient descent, and Newton's method can be found here.

optimization routines aim to find sufficiently good solutions in intractably large search spaces, yet the optimized parameters may, in fact, be a local minimum [3, 38]. Other optimization routines (e.g. gradient descent) are specifically capable of iteratively moving towards the nearest local minimum [16].

Many algorithms exist for numerically optimizing a function, but the choice depends on the specific problem being considered. Considerations informing this choice include,

1. Are the parameters continuous, discrete, or a combination of both?

2. Is the function differentiable (via, e.g., analytic or automatic differentiation)? Does it have a continuous second derivative?

3. Are the parameters bounded, as is often the case when optimizing a physical system?

4. Do the parameters need to satisfy constraints/relationships between themselves (boundary conditions, conservation laws, etc.)

In the context of inverse design, two main classes of algorithms are used: evolutionary algorithms [38] and gradient-based algorithms [40]. Evolutionary algorithms are beneficial as they do not require a gradient and can often well-approximate a global minimum solution. However, their performance may scale poorly with the number of parameters to be optimized. In turn, gradient-based methods converge to a local minimum from a starting location and can be computationally efficient for large number of free parameters. However, they require the ability to compute the gradient (which is not always possible, e.g. when using discrete parameters) and the performance of the algorithm can be highly dependent on the initial optimization parameters.

2.3.1 Evolutionary methods

Evolutionary (or swarm) optimization methods are algorithms inspired by natural phenomena, first appearing in the literature in the second half of the 20th century [38]. These include genetic algorithms, simulated annealing, memetic algorithms, particle swarm algorithms, ant colony optimization, artificial bee colony, grey wolf optimization, cuckoo search algorithm, covariance matrix adaptation evolution strategies, and many more [42]. These algorithms iteratively update a population (a set) of possible designs/parameter sets, using heuristic methods to guide it towards better performing designs. These algorithms use a population (a set) of candidate solutions which share information about their evaluation score between each other to iteratively guide individual solutions in the population toward better performing areas in the design space. While each evolutionary algorithm has its own unique aspects, each follows the general process:

1. Create an initial population of designs (a set of parameters, $\{\mathbf{x}^{(j)}\}$). These are randomly selected from a probability distribution or given *a priori*.

2. Evaluate the performance of each design in the population, $\{F(\mathbf{x}^{(j)})\}$.

3. Select high-performing designs and evolve them to generate new designs. Heuristic algorithms are categorized by the rules and methods which update the population. For example, this update rule corresponds to mutation/crossover/selection operators in genetic algorithms, velocity/position updates in particle swarm algorithms, changing the covariance matrix in the covariance matrix adaptation algorithms, etc.

4. Iterate Steps 2 and 3 until a stopping criterion is reached. This could include reaching a maximum number of iterations, reaching a certain objective value, or plateauing of the performance in several consecutive populations.

5. Return the best performing solution(s).

Commonly, these methods have a stochastic element to them which balances between trying new parameter sets or improving on already tested parameters [38]. This is often discussed as a trade-off between *exploration* and *exploitation*, and is an important consideration when building optimization routines. This trade-off is often controlled via *hyper-parameters*, constant values which will define how the populations changes during optimization.

Genetic algorithms, perhaps the most common and longest standing evolutionary algorithm, were first introduced in the in the 1960's, but gained popularity for numerical optimization in the 1990's [77]. Genetic algorithms update the population at each successive step via three main operations; selection, cross-over, and mutation. Selection determines which designs in a population are kept and evolved for the next generation, and examples include *tournament selection* and *roulette selection*. Cross-over mixes the parameters of two solutions together, example include: one-point, two-point, or blend cross-over. Mutation randomly changes some subset of the solution's parameters, and example include: uniform, Gaussian, or bit-flip mutations.

Evolutionary algorithms and related methods have a number of advantages. First, they do not require information about the gradient of the function in order to optimize solutions. Second, they are well-suited to parallel-computing infrastructures. For this, solutions in a population can be evaluated on distributed CPU (central processing unit) cores or nodes, with one CPU core/node acting as a control, sending out the tasks to workers to evaluate designs in parallel. Ideally, parallelizing an evolutionary algorithm would provide a linear speed-up in computation time (i.e., for X CPUs, the optimization is X times faster). However, handling the distribution of these calculations adds extra computational overhead, which takes time – meaning that there are diminishing returns for adding more worker CPUs [78]. Finally, evolutionary procedures have the advantage of optimizing multiple objectives simultaneously. Multi-objective optimization problems can, in some scenarios, be simplified to a single scalar optimization by framing the problem as $F(\mathbf{x}) = a_1F_1(\mathbf{x})+a_2F_2(\mathbf{x})+\dots$, where $F_i(\mathbf{x})$ is each individual objective to be optimized and $F(\mathbf{x})$ is their weighted sum with weights a_i. This requires *a priori* knowledge of how to weights different objectives, which is often not available. Alternatively, multi-objective optimization algorithms can provide a return a set of solutions that perform well on multiple cost functions – for example the non-dominated sorting genetic algorithm, or NSGA-II [79], which approximates the *Pareto front* [80]. This then allows the designer to explore a number of possible solution and decide, *a posteriori*, how to balance the multiple objectives.

2.3.2 Gradient-based methods

In contrast to evolutionary methods, which treat $F(\mathbf{x})$ as a black-box, gradient-based methods use information about the derivatives of the objective function to iteratively move towards a local minimum. Just as in evolutionary algorithms, the difference between different gradient-based optimization routines is mainly the update rule at each successive step. As introduced in Section 2.2.2, for $F(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$ which is continuous, the first- and second-order derivatives (gradient vector and Hessian matrix) are respectively,

$$\nabla_i F(\mathbf{x}) = \left.\frac{\partial F}{\partial x_i}\right|_{\mathbf{x}} \qquad\qquad H_{i,j}(\mathbf{x}) = \left.\frac{\partial^2 F}{\partial x_i \partial x_j}\right|_{\mathbf{x}} \tag{2.6}$$

A gradient-descent algorithm follows the steps of:

1. Start with an initial candidate solution, $\mathbf{x}_0$. This could be randomly selected, given from *a priori* knowledge, or seeded from a different optimization algorithm.
2. Update the candidate solution, $\mathbf{x}_m$, to a lower-cost solution by moving in the direction of the negative gradient (the direction of steepest descent). The general update rule is $\mathbf{x}_{m+1} = \mathbf{x}_m - \alpha_m \nabla F(\mathbf{x}_m)$, where α_m is known as the learning rate at step m, however variants of gradient descent use more sophisticated update rules.
3. Repeat Step 2 until a stopping criterion is met; such as the number of iterations, the cost value reaches a certain point, or the function evaluations plateau (the gradient vanishes).

Extensions to this first-order gradient descent have also been developed, and make use of more sophisticated update rules. These include stochastic gradient descent methods such as RMSProp, AdaGrad, and ADAM [16, 39, 81], which use historical gradient evaluations, decaying values of α_m, and/or different values of α_m for each parameter. These variants of gradient descent are foundational to deep-learning architectures, as they have lower resource requirements and can be efficiently parallelized with co-processors such as graphical processing units (GPUs) or tensor processing units (TPUs).

Another class of gradient-based methods uses the function gradient with variants of the Newton method for root-finding. Newton's method is an iterative technique for finding the roots (i.e. zero-crossings) of a generic function, $y(\mathbf{x}) = 0$, using the gradient $\nabla y(\mathbf{x})$ to do so. Given that the minimum points of the objective function $F(\mathbf{x})$ occur when $\nabla F(\mathbf{x}) = 0$, by setting $y(\mathbf{x}) = \nabla F(\mathbf{x})$, Newton's method can be used to optimize the objective function. Towards this end, Newton's method uses the second-order derivative (Hessian) to converge on a local minimum, $\mathbf{x}_{\min}$. Compared to the first-order gradient-descent methods discussed above, Newton's method can take a more direct route to a local minimum, as it uses the curvature of the function [82, 83]. For each successive

iteration of Newton's method, the candidate is instead updated by $\mathbf{x}_{m+1} = \mathbf{x}_m - \nabla F(\mathbf{x}_m)\,[H(\mathbf{x}_m)]^{-1}$. For some problems, it can be computationally expensive to directly calculate the Hessian and its inverse, so an approximation is made using the gradient and objective function evaluations from previous iterations. Variants of Newton's method include the Broyden-Fletcher-Goldfard-Shanno (BFGS) algorithm (and related routines such as the limited-memory approximation – the L-BFGS), the Davidon-Fletcher-Powell update, and symmetric rank-one update rule [40]. These methods are generally superior to solely relying on first-order gradients when the number of parameters is relatively small (tens to hundreds). Beyond this, however, Newton's method can become computationally prohibitive and perform poorly [83]. It is for this reason that deep learning, which can have millions of parameters, uses first-order gradient descent algorithms such as ADAM.

2.4 Inverse design in optics

Inverse design techniques in optics hold both historical importance and state-of-the-art significance across sub-disciplines. For photonics, the process of inverse design involves parameterizing the optical device/system design, simulating the optical propagation, evaluating the design performance, and optimizing for high-performance solutions. These techniques have been used most prominently for three sub-fields: lens design, nanophotonics, and quantum optics. Inverse design in each of these fields shares some commonalities in optimization routines and sensitivity analyses as outlined in Sec. 2.2 and 2.3, but differ in their formalisms, model complexities, optical degrees-of-freedom, and desired functionalities. This Chapter reviews these three uses of inverse design in optics, discussing the choices of representation, perturbation, and optimization for each.

2.4.1 Lens system optimization

Arguably the most widespread use of inverse design in optics is in the design of lens systems, studied since the early 1990s [17, 18] and continuing to this day [16, 19, 20, 84]. Lens design aims to create optical systems which have high optical performance (e.g. proper focal length, low aberration, low reflection) limited by manufacturing and environmental constraints (e.g. low footprint and cost) [2]. The parameters $\mathbf{x}$ which represent a candidate solution may include the lens curvature, distance between lenses, or lens material [19]. More complicated parameterizations are also used, allowing greater flexibility in the shapes of lenses the algorithm has access too. This include, for example, freeform lenses [19], which increases the number of design parameters and optimization challenges, but can produce systems with superior performance to simpler parameterizations [84]. A candidate system is evaluated using the ray optics tracing formalism, with objective values extracted from the image formed following propagation through the system.

Lens system design methods use both evolutionary algorithms and gradient-based routines. Early research into lens systems optimization began in the second-half of 20th century, first used gradient-descent algorithms, often targeting the minimization of lens abberations via a damped least-squares objective function [85, 86]. However, in the 1990's, the advent of global optimization techniques like evolutionary algorithms gained in popularity. These algorithms (mainly simulated annealing and genetic algorithms at the time), were often better suited to finding global optima and explored a larger region of the parameter space. This has worked very effectively for finding good lens designs for both small- and large-scale problems [3, 87]. Another benefit of using genetic algorithms in lens design was the ability to permute the order of optical elements. Thus, from a set of simple or well-known lens system topologies, new ones can be generated by the computer and tested for performance automatically [86].

On the other had, 'differential ray tracing' is widely used for optimizing lens systems with gradient-descent [88]. These techniques have, in the past, used analytic or numerical differentiation, which has been shown to be inadequate in some cases [89]. More recently, automatic differentiation has been applied to differentiate the cost function of a lens design with respect to the design parameters [20]. The application of automatic differentiation in Ref [20] improves the computation time, and importantly, the designer time – as the user does not need to manually derive any gradients.

Sensitivity analysis is often accomplished with Monte Carlo sampling methods and is available in a number of commercial software packages, such as ZEMAX.

2.4.2 Nanophotonics design

Inverse design is widely used to optimize nano- and micro-scale optical component designs [1, 22–24, 66, 90, 91]. Often, these methods target photonic devices which achieve better performance or require a smaller footprint than traditional photonic designs. The design of nanophotonic and metamaterial optical devices including microcavities [91], photonic crystal fibers [92], and integrated couplers [23, 90], among others, have been demonstrated via inverse design techniques. Here, the devices are represented as a parameterized dielectric structure, constrained by the fabrication capabilities of the foundry, and designs are simulated and evaluated via various beam propagation methods, such as finite-difference time-domain (FDTD) methods [93].

The earliest demonstrations of inverse design for nanophotonics began in the 1990's. In Ref. [22] Spühler *et al.* used a genetic algorithm to design a fiber-to-chip coupler with 2dB better efficiency compared to previous techniques, and the authors of Ref. [24] employ gradient-descent optimization on a symmetric structure of two dielectrics to increase the width of a photonic bandgap (demonstrating a 34% increase from the starting structure). Evolutionary algorithms are often used in problems with fewer parameters [21, 22], however, as the number of design parameters

increases, evolutionary methods become intractable and gradient-based techniques are required. For example, in recent shape optimization work [1, 23, 66], the parameterization of the device is the dielectric material on a mesh grid, and as such, the number of grid points can grow very large as resolution or photonic chip size increases. In order to optimize such a device, the gradient must be accessible, stable, and computationally-efficient. Numerical differentiation is impossible in this situation due to both the number of parameters and the computation time required for each forward problem. Fortunately, the derivative of the cost function in this situation can be recast as solving the adjoint problem [94]. This allows the derivative to be computed via only one additional simulation of the device – an approach that scales well with the number of parameters and requiring less programming overhead [23]. This adjoint method is a foundationally important technique for inverse design problems which require solving partial differential equations – such as Maxwell's equations for photonics [23], Fourier's equation for heat transfer [95], or fluid dynamics for mechanical shape optimization [96]. It can be formulated from the theory of Lagrange multipliers and falls under the category of analytic differentiation discussed in Sec. 2.2.2. See Refs. [97] and [23] for a discussion on the adjoint method in nanophotonic design in theory and practice.

Automatic differentiation has also been recently demonstrated for solving Maxwell's equations in photonic devices [25, 98], demonstrating the automatic differentiation of coupling efficiency, spectral power, and intensity distribution with respect to dielectric material properties. It should be noted that this employs the forward mode automatic differentiation which, for a general $F(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}^m$, is well suited to situations where $n \ll m$ – this is however not the case for many nanophotonic inverse design problems where $m = 1$ and n is very large.

In some cases of sensitivity analysis for nanophotonic structures, a simple 'under-etch' and 'over-etch' (representing the tolerances of the foundry's fabrication process) are tested to determine an expected loss of performance when produced [66]. When the gradient vector is available, i.e. when using the adjoint method or automatic differentiation, the first-order derivative is used for sensitivity analysis. More complex analysis of sensitivity is demonstrated in Ref. [99], which uses principal component analysis to determine the most impactful parameters and fully explore the lower-dimensional design space along the principal components. Commercial solutions for nanophotonic design, such as Lumerical, have recently developed inverse design packages [66] which have been demonstrated for the design of compact, low-loss waveguides [90]. Similarly, the COMSOL Multiphysics Suite provides functionality for calculating design sensitivities via the adjoint method [100].

2.4.3 Quantum optics system design

In recent years, inverse design has been used to design new quantum optics experiments [5, 7, 26, 101]. The first demonstration of such an algorithm, named Melvin, was in 2016 by Krenn *et al.*, and illustrated the ability to design an optical setup which produces high-dimensional Green-

berger–Horne–Zeilinger quantum states (a useful state in quantum communication, metrology, and fundamental investigations) [102]. This was the first system proposal to create such states [5]. Based on this, follow-up algorithms explored the design-space of quantum optics experiments with search and reinforcement learning algorithms [101] which continued to suggest novel systems for various complex quantum states. Most recently, Krenn *et al.* have demonstrated a new algorithm (named Theseus) which uses a novel and abstract representation of quantum experiments as a weighted graph [26, 103] and boasts impressive results, capable of finding new systems for a number of important, complex quantum states.

These algorithms are parameterized by both real, continuous numbers [7], but also by discrete objects, graphs, which require discrete topology optimization [5]. As such, the optimization procedure can no longer use gradient-based methods or most evolutionary algorithms in their general form – requiring different techniques. The first demonstrations used a simple, random topology search algorithm – adding new components to the experiment, testing the figure-of-merit (the quantum fidelity), attempting to simplify the setup, and iterating until a sufficiently high-performing system was discovered [5]. As well, small systems which performed well were saved as individual building blocks which the algorithm could re-use. This process improved convergence speeds and often discovered known sub-systems which have been used in the field for years [5]. The most recent demonstration in Ref. [7] uses both graph topology optimization in combination with a gradient-based optimization routine (BFGS algorithm) to minimize the continuous, real graph weights (which correspond to probability amplitudes of the quantum state). The topology optimization deletes edges in the graph when they are deemed unimportant, realized through the addition of a lasso (ℓ^1) regularization term in the cost function.

This inverse design of quantum experiments shares similarity to some of the advanced lens system design methods in that, through the optimization of a discrete structure (order of lenses, topology of quantum experiment), new functionalities can be discovered. The benefit of these methods is that the human designer does not need to have *a priori* knowledge about a good initial design, but rather, this initial design is discovered through an iterative process. Ideally, this lowers the amount of high-level domain-knowledge prerequisite to quantum system design, allowing to find novel, unintuitive designs. This is especially important in problems, such as quantum optics, where conceptual and intuitive understanding of the full problem space is challenging or impossible due to the complexity of the system.

2.4.4 Ultrafast optics

Finally, optimization of optical system parameters is used throughout ultrafast optics, laser design, optical signal processing, telecommunications, and other related fields. However, these are largely individual demonstrations; broader connections between the concepts, tools, and techniques of these research areas are lacking. In general, these works use parameter optimization routines as described

in Chapter 2.3 on a fixed optical system topology, particularly gradient-free optimization, as deriving the gradient of such systems is often challenging. A handful of examples of these include: the design of lasers [27, 104] and other optical sources [105], noise mitigation over optical channels [29], pulse shaping [106], all-optical processing [107], and networking applications [108]. Developing common techniques and tools for use in these fields remains an outstanding need, which is the main motivation of this work. Here, computational inverse design methods are developed and demonstrated for the design of ultrafast optical processing systems. The methods and software presented in Chapters 3 and 4 provide a platform for rapid simulation, analysis, and optimization of optical systems, along with automatically exploring new optical system via topology optimization. While the results of this work focuses specifically on fiber-based systems in the time-frequency domains, the concepts and software can readily be extended to new problems.

CHAPTER 3

Methods: Automated Search for Optical Processing Experiments (ASOPE)

As outlined in preceding Chapters, inverse design has been a foundational tool for designing novel, high-performing optical systems and devices. In this Chapter, I introduce and demonstrate a novel method for the inverse design of optical systems, specifically targeting ultrafast, fiber-based systems. In this approach, optical systems are represented as computational graphs, with vertices and edges representing different types of optical elements provided in a user-defined library. Vertices represent optical components which map between spatial propagation paths (e.g., beam splitters) and edges represent optical elements that alter the optical field within a single propagation path (e.g., laser sources, modulators, amplifiers, filters, and detection elements). The system graph, comprising of an experimental topology and parameter set, is then simulated and evaluated. Using automatic differentiation, derivatives of the evaluation function are compiled which enable gradient-based optimization and sensitivity analysis. Finally, through the iterative evolution of the graph structure (i.e. topology optimization) and parameter optimization, new optical systems are generated, evaluate, and optimized – searching for new functionalities and emergent behavior. Fig. 3.1 provides the basic process flow of the algorithm. The inverse design algorithm *Automated Search for Optical Processing Experiments*, or `ASOPE`, was developed in the Python programming language.

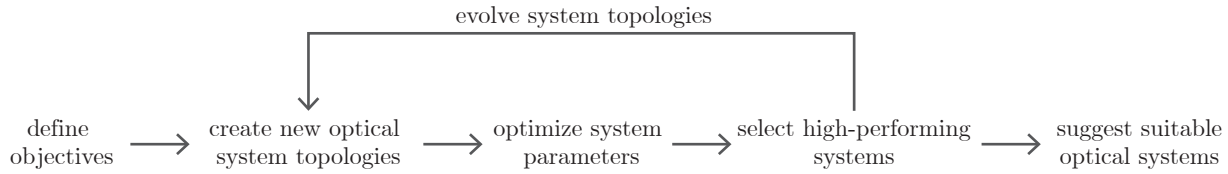


Figure 3.1 – Overview of inverse design procedure for optical systems.

3.1 Representation and evaluation of optical systems

3.1.1 Parameter representation

The first level of representation is on the scale of an individual optical component. The fundamental "building block" components are represented with a compact, parameterized model which describes its effect on the optical field (either in the time-domain, $\Psi(t)$, or in the frequency domain, $\tilde{\Psi}(\omega)$). Each compact model of a component is a time- or frequency-dependent transfer function, $f_{v_i}(t)$ or $f_i(\omega)$, parameterized by a vector $\mathbf{x}_i$. We write these transfer functions as,

$$\Psi_{\text{out}} = f_i(\Psi_{\text{in}}, \mathbf{x}_i) \tag{3.1}$$

Each model acts on either a single spatial path (i.e. modulators, filters, amplifiers, etc.) or maps between different spatial paths (i.e. splitters and multiplexers). The parameters, $\mathbf{x}_{v_i}$ on a given optical component are all considered as bounded, continuous values and represent tunable experimental parameters. For example, a modulator (e.g., phase, amplitude) may be parameterized by the modulation frequency, modulation depth, and/or bias.

For a more complex optical system which contains multiple spatial propagation paths (i.e. a system containing splitters and/or multiplexers), a matrix formalism of composing functions must be adopted – with the number of incoming or outgoing edges to a vertex indicating the matrix dimensions (see Eqn. 3.6) or, more generally for components with N input paths and M output paths,

$$\Psi_{\text{out},\ell} = \sum_{k=1}^{N} f_{i,(k,\ell)}(\Psi_{\text{in},k}, \mathbf{x}_i) \tag{3.2}$$

$$\{k \in \mathbb{Z} \mid 1 \leq k \leq N\},$$
$$\{\ell \in \mathbb{Z} \mid 1 \leq \ell \leq M\}$$

Appendix A.4 discusses the component models developed and used in `ASOPE`. The compact models of each component are similar to the scattering matrices or block-diagrams elements used in photonic circuit simulation engines (e.g., Lumerical INTERCONNECT, Synopsis) to model system level performance of an optical schematic [109].

While experimentally, some parameters may technically be discrete (e.g., due to bit-depth resolution of a programmable filter), all parameters are treated as continuous in the algorithm in order to use gradient-based methods for optimization and sensitivity analysis. Each parameter x_i of a model is defined by an upper and lower bound (b_{upper} and b_{lower}, respectively) and an uncertainty σ_i reflecting its experimental tolerance or imprecision, these values informed by domain knowledge

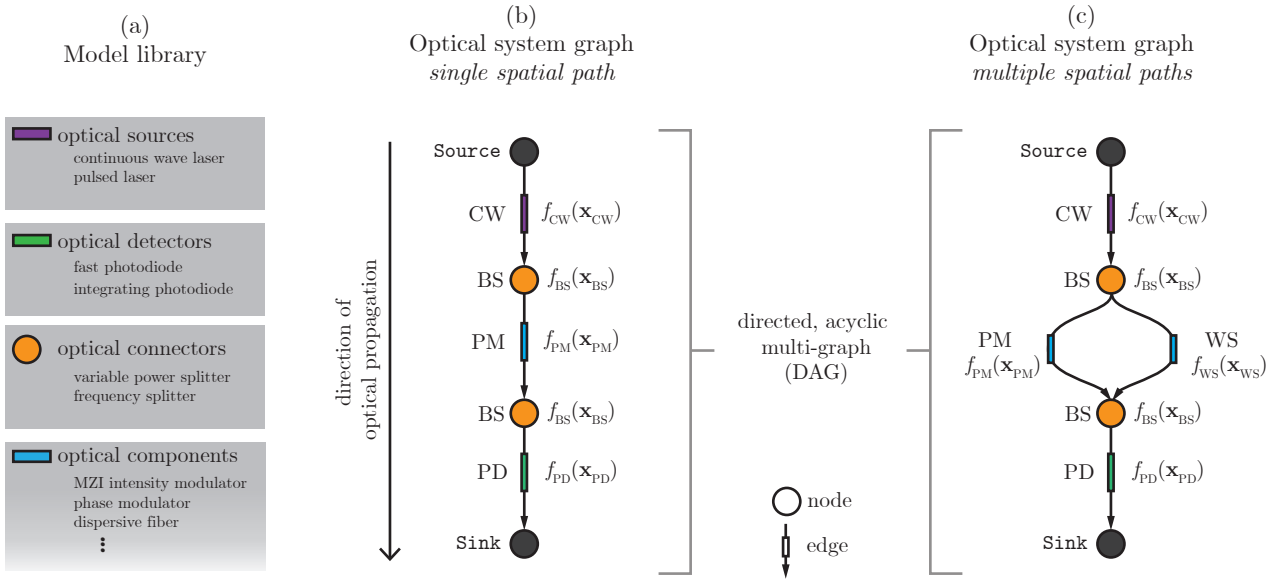


Figure 3.2 – Example optical systems represented as computational graphs. (a) A serial optical system using only one spatial path. The resulting propagation function is written in Eqn. 3.4. (b) A more complex optical system that includes multiple, parallel spatial propagation paths. The resulting propagation function is presented in Eqn. 3.6.

or experimental results (i.e. can be extracted from a component datasheet provided by optical component manufacturers). Currently implemented components include continuous wave lasers, mode-locked lasers, optical amplifiers, attenuators, dispersive fibers, optical spectrum shapers, phase and intensity modulators, as well as bandwidth-limited photodiodes; all are differentiable with respect to their design parameters (see Appendix A.4 for more details).

3.1.2 Graph representation

The second representation level models a complete optical system as a computational graph. A graph, $G = (V, E)$, has a set of vertices/nodes $\{V\}$, and a set of edges $\{E\}$ that connect these nodes. These collections of nodes and edges represent computations that map inputs to outputs. Graphs are useful representations in a number of inverse design approaches, including machine learning, chemistry, network design, and quantum optics [7, 15, 31], and each domain has unique problems and constraints which must be encoded into the graph data. Similarly in `ASOPE`, unique constraints which arise for optical systems must be implemented in the graph representation. In `ASOPE`, the nodes and edges both represent optical components in the system and the flow of light through the system (see Fig. 3.2). First, the edges are *directed*, which indicates the direction of optical propagation. Second, the graph is *acyclic*, to disallow optical loops in the system (see Chapter 5 for further discussion). Finally, the graphs are *multi-graphs*, which enables multiple edges between the same two nodes, allowing interferometric setups. Thus, an optical system is represented as a *directed, acyclic multi-graph*, or DAG. The encoding used in `ASOPE` to represent an optical system as a DAG is as follows:

Graph edges: These represent components which generate, manipulate, or detect an optical field within a single spatial mode. These include optical sources (e.g., continuous wave or pulsed laser sources), detectors (e.g., photodiodes), or optical control elements (e.g., modulators, filters, amplifiers, attenuators, delay-lines, etc.). The edge direction specifies the direction of optical propagation.

Graph vertices: These represent interconnects between optical elements, which map incoming spatial paths to outgoing spatial paths (Eqn. 3.2). The incoming/outgoing spatial paths are represented by the directed edges which end/start at the vertex, respectively. These include splitters, couplers, and multiplexers.

`Source` and `Sink` vertices: These are used at the start and end (terminals) of a system graph. They are 'anchor' points to define the direction of propagation, i.e. the optical propagation through the system is defined by the graph's topological order, from the `Source` vertices to the `Sink` vertices.

The overall optical system, composed of a set of N components, v_i, $\{i \in \mathbb{Z} \mid 1 \leq i \leq N\}$, is then parameterized by $\mathbf{x}_G$, which encompasses the parameters on every optical component, $\mathbf{x}_i$ (i.e. $\mathbf{x}_G = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N]^{\mathrm{T}}$).

With a graph, complete with a set of components, topology, and parameters, the overall propagation function of the whole system is then simulated. This is accomplished by successively simulating the effect that each element has on the optical field, in the order set by the graph topology (a DAG has, by definition, a unique graph topological order). For small systems, this can be written out as the composition of the component model functions,

$$f_G = f_{v_N} \circ f_{v_{N-1}} \circ \cdots \circ f_{v_1}(\Psi, \mathbf{x}) \tag{3.3}$$

For example, for the simple point-to-point system in Fig. 3.2b, it can be written as

$$f_G = f_{\mathrm{PD}} \circ f_{\mathrm{BS}} \circ f_{\mathrm{PM}} \circ f_{\mathrm{BS}} \circ f_{\mathrm{CW}}(\Psi, \mathbf{x}) \tag{3.4}$$

For a more complex optical system which contains multiple spatial propagation paths (i.e. a system containing splitters and/or multiplexers), a matrix formalism of function composition must be adopted – with the number of incoming or outgoing edges to a vertex indicating the matrix dimensions (see Eqn. 3.6). For the system depicted in Fig. 3.2b, the propagation function can be written as,

$$f_G = f_{\mathrm{PD}} \circ \begin{pmatrix} f_{\mathrm{BS2}_a} & f_{\mathrm{BS2}_b} \end{pmatrix} \circ \begin{pmatrix} 0 & 0 \\ 0 & f_{\mathrm{WS}} \end{pmatrix} \circ \begin{pmatrix} f_{\mathrm{PM}} & 0 \\ 0 & 0 \end{pmatrix} \circ \begin{pmatrix} f_{\mathrm{BS1}_a} \\ f_{\mathrm{BS1}_b} \end{pmatrix} \circ f_{\mathrm{CW}} \tag{3.5}$$

Expanding this out, it can also be written as,

$$= f_{\mathrm{PD}} \circ (f_{\mathrm{BS2}_a} \circ f_{\mathrm{PM}} \circ f_{\mathrm{BS1}_a} \circ f_{\mathrm{CW}} + f_{\mathrm{BS2}_b} \circ f_{\mathrm{WS}} \circ f_{\mathrm{BS1}_b} \circ f_{\mathrm{CW}}) \tag{3.6}$$

However, as optical systems scale up in size, writing the propagation functions becomes unwieldy and bulky, and performing analysis by-hand can become tedious or impossible. The graph representation thus serves as a compact method to define and simulate larger, more complex optical systems.

Further, any possible optical system that `ASOPE` generates and suggests must be physically realizable in a lab. This includes, for example, ensuring at least one optical source is included in the setup and that components have a valid number of connected spatial paths (e.g. a modulator should have a single input and single output, if more output paths are needed, a splitting element must be inserted). To track these constraints, the optical components are divided into four types:

Optical sources: Optical sources such as continuous wave lasers, pulsed lasers, or an optical data stream. These are represented on the graph edges, $\{E\}$.

Optical detectors: Represents measurement devices such as a photodiode, optical spectrum analyzer, or optical transmission channel. These are represented on the graph edges, $\{E\}$.

Single-path optical components: Represents optical components which manipulate the optical field in a single propagation path such as modulators, filters, dispersive elements, attenuators, and amplifiers. These are represented on the graph edges, $\{E\}$.

Spatial path connectors: Represents components which couple/split optical fields from/into multiple propagation paths, such as beam-splitters or multiplexers. These are modeled generally to have any number of input/output ports. These are represented on the graph vertices, $\{V\}$.

Importantly for inverse design, the algorithm must be able to create new solutions without them being explicitly programmed. A set of graph mutation (or evolution) rules change the graph topology and optical components to generate new optical systems without explicitly defining them. So, a user can either explicitly define an optical system by its components and topology, or use these rules to automatically produce new systems. These mutations include adding a new edge and/or vertex, removing an edge and/or vertex, or swapping the optical component models on existing edges/vertices (see Sec. 3.3.2 and Fig. 3.7). These rules, in essence, map from one DAG to a new DAG – ensuring that the newly generated system graph is valid (i.e. it is an optical system that could be realized experimentally).

As the optical systems continually change, automatic differentiation is necessary to compute derivatives for gradient-based optimization and sensitivity analysis. For the Python programming language, `autograd` offers a simple, user-friendly framework, largely because it differentiates functions written with the standard numerical and scientific computing libraries, i.e., `numpy` and `scipy`.

This allows one to translate from the mathematical description of optical propagation to a differentiable simulation in a compact and intuitive way. Importantly, complex-valued functions such as the FFT and IFFT, are differentiable with the base library. This allows optical component models to be written in the domain where they are simplest and most direct [33] – e.g., dispersive elements are modeled in the frequency domain, time-varying modulator elements in the time domain, and derivatives can be traced through the propagation function. As such, each optical component model, f_{v_i}, and cost function, $F(G, \mathbf{x})$, are developed to be differentiable with `autograd`, opening the door for efficient optimization and sensitivity analysis. Appendix A.4 details the component models, discusses some considerations when automatically differentiating the optical propagation, and compares tools for scaling up the simulations using GPU processing.

This graph representation of optical systems can encompass a wide variety of optical technologies and systems that used in fields such as telecommunications and optical signal processing. Current limitations of this representation are that it does not consider optical loops, backward propagation, or nonlinear elements that require more sophisticated simulation methods (e.g., the split step Fourier method [63]). Finally, this thesis focuses on the time-frequency degree-of-freedom and fiber-based optical components, however the representation and the following computational methods could be adapted to other optical degrees-of-freedom.

Evaluation of a system

The evaluation of an optical system graph estimates how well the system performs towards the desired objective. With automatic differentiation, defining cost functions is very flexible – it can be defined in the time-domain or frequency-domain, and use many complex functions. The main requirement is that it returns a single scalar cost value, $F(G, \mathbf{x}) \in \mathbb{R}$ and uses functions supported by `autograd`. Automatic differentiation also allows rapid prototyping and testing of new cost functions. This cost function is then minimized with respect to the graph parameters, corresponding to better system performance.

One useful cost function compares the optical (or electronic) signal at a specific element in the setup to a desired optical (or electronic) signal via a similarity function. While a number of similarity metrics are available (e.g. cosine similarity, ℓ^1-norm, ℓ^p-norm), we use ℓ^2-norm (i.e. sum-of-squares) between the target and generated optical field, over the simulation time window, T,

$$F(G, \mathbf{x}) = \int_T |I_{\text{target}} - I_{\text{generated}}|^2 \, \mathrm{d}t \tag{3.7}$$

where $I(t) = |\Psi|^2$ is the time-dependent intensity. This overlap integral, $F(G, \mathbf{x}) \in \mathbb{R}$, measures the similarity of a generated, time-dependent optical field intensity is to a target intensity. Eqn. 3.7 can be similarly applied to a time-dependent voltage signal measured at one of a photodiode element (as the optical-to-electronic conversion is not a linear function and has a finite bandwidth,

the two are not equivalent). For example, in Sec. 4.1.1, the cost function is calculated based on the difference in optical intensity between generated and target optical field, while Sec. 4.1.2 instead compares the electronic signal produced at the photodiode to a target electronic waveform.

A second type of cost function is the sensitivity of the system to perturbations – for example, the sensitivity of the average optical power to a phase shift in one spatial path. For example, in Sec. 4.1.3, the cost function is of the form:

$$F(G, \mathbf{x}) = -\left| \frac{\partial P_{\text{avg}}}{\partial x_\varphi} \right|_{\mathbf{x}} \tag{3.8}$$

where P_{avg} is the average optical power and x_φ is the phase on a phase shifting element in the system. This is only accessible via the use of automatic differentiation, and could be useful for searching for systems that are are highly stable to perturbations or highly sensitive (e.g., optical sensing). This searches for optical systems which are highly phase-sensitive, and as demonstrated in Sec. 4.1.3, returns common interferometric optical systems.

3.2 Perturbation

For a fixed system graph topology, perturbation techniques –or sensitivity analysis– provides insight about the local objective function landscape around a set of parameters $\mathbf{x}^*$. Performing sensitivity analysis on a system and set of parameters can be useful in a number of ways. It can: provide information about which components are most sensitive to perturbations, allow quantitative comparison between the stability of two different parameter sets or system topologies, and indicate some of the physics underlying the system.

3.2.1 Hessian sensitivity analysis

While the first-order gradient vector is called the 'sensitivity' in some inverse design problems, this is limited for two main reasons: it only indicates sensitivities of parameters in isolation and also provides no information about the objective function landscape at an extrema point. At any minimum, maximum, or saddle point, the gradient vector is $\nabla_i F(\mathbf{x}^*) = \mathbf{0}$, so no indication of sensitivity is available at an optimal point using only first-order derivative methods. Sensitivity analysis using the Hessian matrix is a second-order gradient-based method and characterizes not only the sensitivity of the objective function to each individual parameter, but also parameters in combination.

The Hessian matrix $\mathbf{H}(\mathbf{x}^*)$ is the second-order derivative of a function $F : \mathbb{R}^n \to \mathbb{R}$, and is a real, square, and symmetric matrix. Each element, $H_{i,j} = \partial^2 F / \partial x_i \partial x_j$, is the second-order partial

derivative on parameters x_i and x_j, evaluated at $\mathbf{x}^*$. It can be used to extract two useful pieces of information: the *parameter curvatures/directions* and the *principal curvatures/directions* [34–36]. The parameter directions are the directions in the high-dimensional parameter space along each parameter axis, and the principal curvatures are the second-order derivative. In other words, the parameter curvatures are the diagonal elements of $\mathbf{H}$. Conversely, the principal directions and curvatures are the eigenvectors and eigenvalues, respectively, of the Hessian matrix, satisfying the following equation:

$$\mathbf{H}(\mathbf{x}^*)\mathbf{v}_i = \lambda_i \mathbf{v}_i \tag{3.9}$$

As $\mathbf{H}$ is always real and symmetric, it is diagonalizable via eigendecomposition. While the parameter curvatures indicate the sensitivity of parameters in isolation, the principal curvatures and off-diagonal elements indicate the sensitivity of parameter in combination with other. This diagonalization of $\mathbf{H}(\mathbf{x}^*)$ can be seen as a linear basis transformation such that there are no mixed second-order partial derivatives.

If $|\lambda_i| \approx 0$, moving in the direction of $\mathbf{v}_i$ is likely highly stable (though not guaranteed, as this is only a second-order approximation of cost function). Alternatively, if $|\lambda_i|$ is large, then moving in the direction of $\mathbf{v}_i$ is highly unstable and the objective value changes rapidly. Additionally, the signs of the principal curvatures can indicate if the function at $\mathbf{x}^*$ is a minimum, maximum, or saddle-point. Such analysis of $\mathbf{H}(\mathbf{x}^*)$ can be useful, as saddle-points are known to plague high-dimensional problem spaces and can cause issues with some optimization algorithms (e.g., SGD, L-BFGS [16, 32]).

Visualizing and understanding the Hessian is challenging. In a parameter space with a dimension of two, plotting the function landscape can be done directly, along with the parameter and principal directions/curvatures, as in Fig. 3.3. Here, the bottom row presents the principal directions and curvatures of $\mathbf{H}$, which indicate directions of highest and lowest sensitivity. The arrows are now rotated (bottom row) along the function curvature and their lengths are proportional to λ_i – this matches with our visual understanding of function landscape from the heatmap. The same information can be displayed by plotting the Hessian matrix, parameter curvatures, and principal curvatures directly, as in Fig. 3.4. Figs. 3.3 and 3.4 convey the same information via two different methods, however the latter can easily scale with the number of parameters. See Appendix A.2 for further examples of the Hessian analysis on a high-dimensional test function.

An issue arises when moving from test functions to cost functions of an optical system – as parameters will have a variety of units (e.g., a modulator driving frequency parameter range from MHz to GHz, while the parameter representing an applied phase shift is in radians between 0 and 2π rad). To accurately compare relative sensitivities, then, the Hessian must be scaled by a factor to ensure the indices are all the same unit. Each parameter, x_i, is assigned an uncertainty, σ_i, based off data sheets, experimental data, or informed by domain knowledge – which can represent

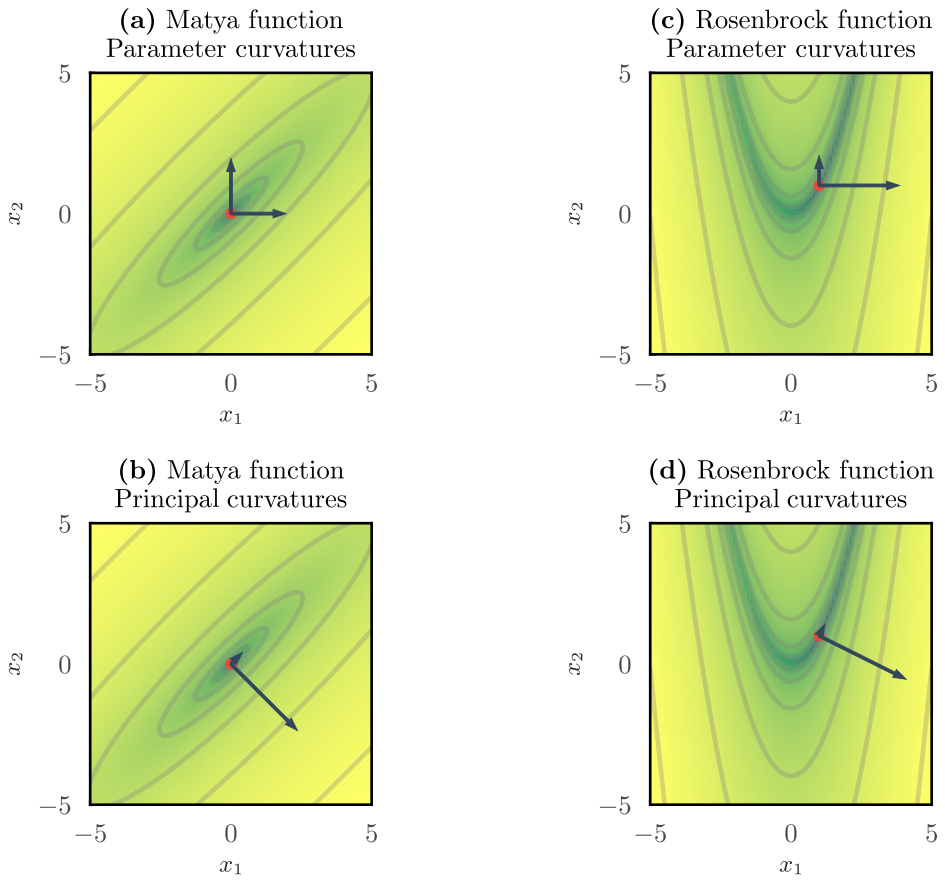


Figure 3.3 – Geometric interpretation of the Hessian analysis. The parameter curvatures and principal curvatures for two common test functions, Matya's function (left) and Rosenbrock's function (right). Red markers indicate global minimum points. **a, c)** Parameter curvatures for x_1 and x_2, with the arrow length proportional to $H_{1,1}$ and $H_{2,2}$. The principal directions and curvatures correspond to the eigenvectors and eigenvalues, respectively, of $\mathbf{H}$ at each global minimum point. These provide a measure of sensitivity for each parameter in isolation. **b, d)** Principal directions, $\mathbf{v}_i$, with the arrows in the direction of $\mathbf{v}_1$ and $\mathbf{v}_2$ and scaled to λ_1, λ_2, respectively. These provide a measure of sensitivity when parameter are perturbed together. Code can be found here.

manufacturing tolerances (e,g. fiber splice length precision) or experimental uncertainties (e.g., timing or frequency jitter). To accurately compare the effect each parameter has on F, the Hessian matrix is scaled by these uncertainty values, σ_i,

$$H_{i,j}^{(\text{scaled})} = (\sigma_i \sigma_j) \frac{\partial^2 F}{\partial x_i \partial x_j} \tag{3.10}$$

which gives each element of $\mathbf{H}^{(\text{scaled})}$ the same units as F. While the exact values of $H_{i,j}^{(\text{scaled})}$ and λ_i may not be entirely intuitive, extreme values of these can be indicative of certain physical behaviours in the system. If, for example, $\lambda_i = 0$, this indicates that the direction $\mathbf{v}_i$ is highly stable and can have useful consequences, such as picking out redundancy or unique physical effects (see Sec. 4.2.2 for a concrete example). Also, by grouping the Hessian elements by the optical

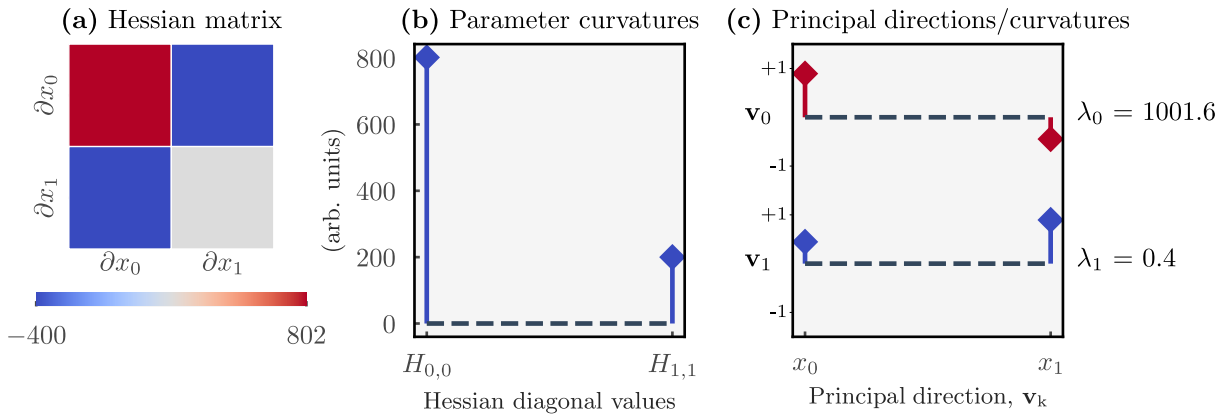


Figure 3.4 – Hessian analysis visualization for the two-dimensional Rosenbrock function. The same information is encoded as in Fig. 3.3c-d, but in a way which can be scaled to dimensions $n > 2$. (a) The Hessian matrix of the two-dimensional Rosenbrock function as a color heatmap. (b) Parameter curvatures of the function (diagonal elements of the Hessian matrix). From this we can determine that the function is more sensitive to perturbation in x_1 than x_2. Equivalent to Fig. 3.3c. (c) Principal directions and curvatures of the function (eigenvectors and eigenvalues of the Hessian matrix). From this analysis we can determine that the function is relatively stable in the direction of $\mathbf{v}_1$, but highly sensitive in the direction of $\mathbf{v}_0$. Equivalent to Fig. 3.3d. Code can be found here.

components to which they correspond (as in Fig. 4.6), one can estimate which optical components are most important to tightly control when building the system in the lab.

3.2.2 Noise analysis

Noise arises from myriad sources, including optical, electronic, and mechanical. While the sensitivity analysis methods outlined above are suitable for understanding how the objective function changes with perturbation, imperfections, or uncertainty in parameters, it does not encompass all nonidealities that exist in real optical systems. The parameter uncertainties, σ_i, reflect systematic errors in the system parameters, but cannot encompass stochastic noise processes. Towards more realistic simulations, noise is added to the simulated optical field as it propagates and both sensitivity analysis and optimization can be performed in the presence of this noise (see Fig. 3.5). For example, amplified stimulated emission (ASE) occurring in an optical amplifier can have detrimental effects in a variety of optical system, such as telecommunications systems [37].

In `ASOPE`, a general noise contribution is modeled as additive Gaussian noise (AGN) on the components where most applicable, i.e. active elements including laser sources, amplifiers, and photodiodes. Each component model has a connected noise model, which simulates the noise characteristics of each component via band-limited additive noise, such as pink noise, low-pass noise, etc. Noise can be toggled on/off for all components to perform comparison between noisy propagation and ideal.

Complex cost functions could potentially be designed based on noise quantities. For example, simulating the system with and without AGN injections and performing a calculation on the two signals (to extract e.g. signal-to-noise ratios, dynamic ranges, etc.) may enable the design of systems with desired noise tolerances. In order to still differentiate the objective function with automatic differentiation, noise is treated as a randomly drawn, constant vector. A similar approach to stochastic gradient descent for the training of artificial neural networks is adopted, where periodically in the optimization process the noise is re-sampled [16].

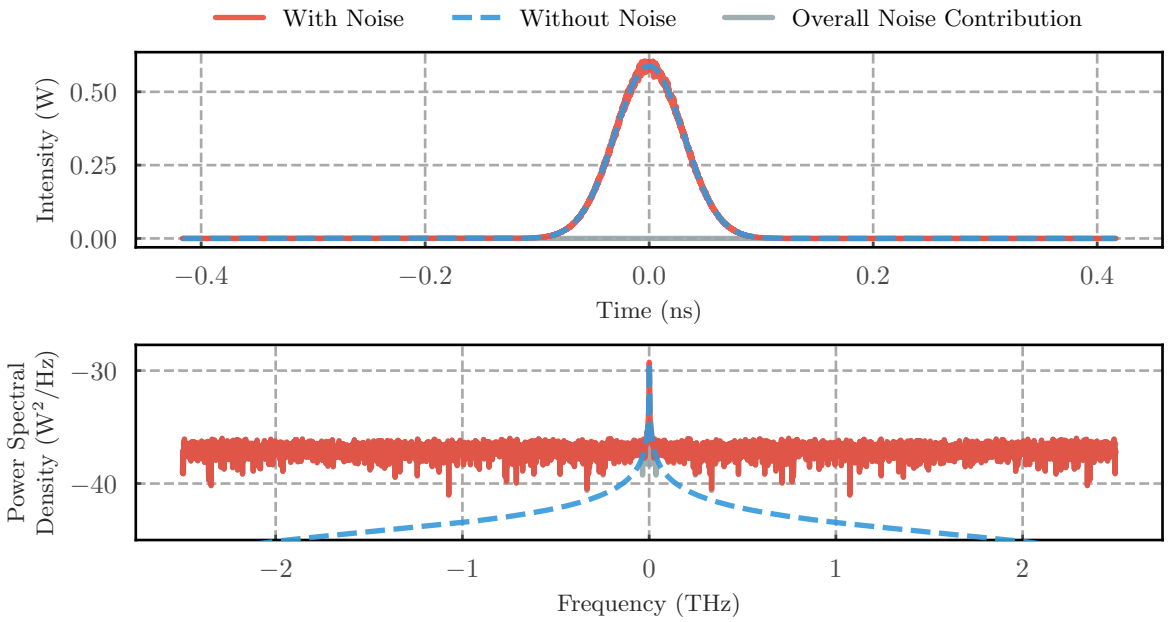


Figure 3.5 – Example of Additive Gaussian noise in optical propagation simulations.

3.3 Optimization

As the component parameters and system topology use different abstract representations – the parameters as a vector of real, bounded values and the optical system as a graph– the optimization routines employed are treated separately and the overall optimization is split in two. A nested optimization routine is developed; with the outer routine changing and optimizing the graph structure by adding, swapping, and removing vertices and edges, and the inner routine finding optimal parameters on all components for the given graph. Working in tandem, these nested algorithms discover optical systems which perform well on a user-defined task (e.g., find optical systems which can produce a targeted optical signal). Chapter 4 demonstrates the application of this optimization routine for the design of optical systems for a variety of tasks.

3.3.1 Parameter optimization

There are ubiquitous optimization tools for optimizing a function $F(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$. These include gradient descent, particle swarm optimization, Newton's method, etc., as discussed in Section 2.3. From a system graph G and parameter vector $\mathbf{x}$, the propagation and evaluation functions are compiled, along with successive derivatives via automatic differentiation. The evaluation function, $F(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$, can be considered a black-box function. This, along with the gradient vector, $F'(\mathbf{x}) = \nabla F(\mathbf{x})$, can be provided to a variety of standard optimization routines – including gradient-based (e.g., L-BFGS, SGD, ADAM, etc.) and evolutionary/swarm algorithms (e.g., GA, PSO, CMA, etc.) to search for optimal points in the parameter space. Parameter optimization routines tested towards use in `ASOPE` include: a custom genetic algorithm [38], a variant of the ADAM algorithm customized to accommodate bounded parameters [16, 39], the L-BFGS algorithm [40, 41], and a PSO implementation [42, 43]. Fig. 4.4 and Table 4.2 compares these algorithms on an example system and target.

A hybrid approach was ultimately adopted as the default parameter optimization method. First, a 'coarse' optimization is accomplished by a genetic algorithm, followed by a 'fine' optimization via the L-BFGS method. This is an attractive strategy in optimizing physical systems, as the initial evolutionary algorithm better explores the full design space and a subsequent gradient method can then converge to the closest local minimum [44]. Empirically, this hybrid approach of a GA followed by L-BFGS local minimization showed the best performance during development. In the custom GA, the operations to alter the population from one generation to the next are one-point cross-over, uniform mutation, and tournament selection. The L-BFGS algorithm uses the gradient vector compiled via automatic differentiation. Fig. 3.6 succinctly outlines the algorithm flow for hybrid parameter optimization for a set graph topology. Fig. 3.6 gives an example of parameter optimization on a fixed graph topology.

3.3.2 Topology optimization

The final and most central pillar of the presented work is the optimization of optical system topologies. Here, heuristic algorithms are used to iteratively build and improve optical system graphs, exploring new optical setups. The method uses an evolutionary algorithm, to evolve and analyze optical systems, while clustering high-performance systems to suggest back to the user. Similar uses of topology optimization have been demonstrated in quantum optics [5, 7], machine learning [31], and molecular design [110, 111] – related concepts and references are discussed where relevant. The results of this work focus on the design of ultra-fast optical systems in the fiber optic platform, see Sections 4.1.2-4.1.3. Please note that, while related, topology optimization has different connotations in nanophotonic device and lens system design, and generally refers to the shape or topology of one device [1], whereas here, it refers to the graph topology.

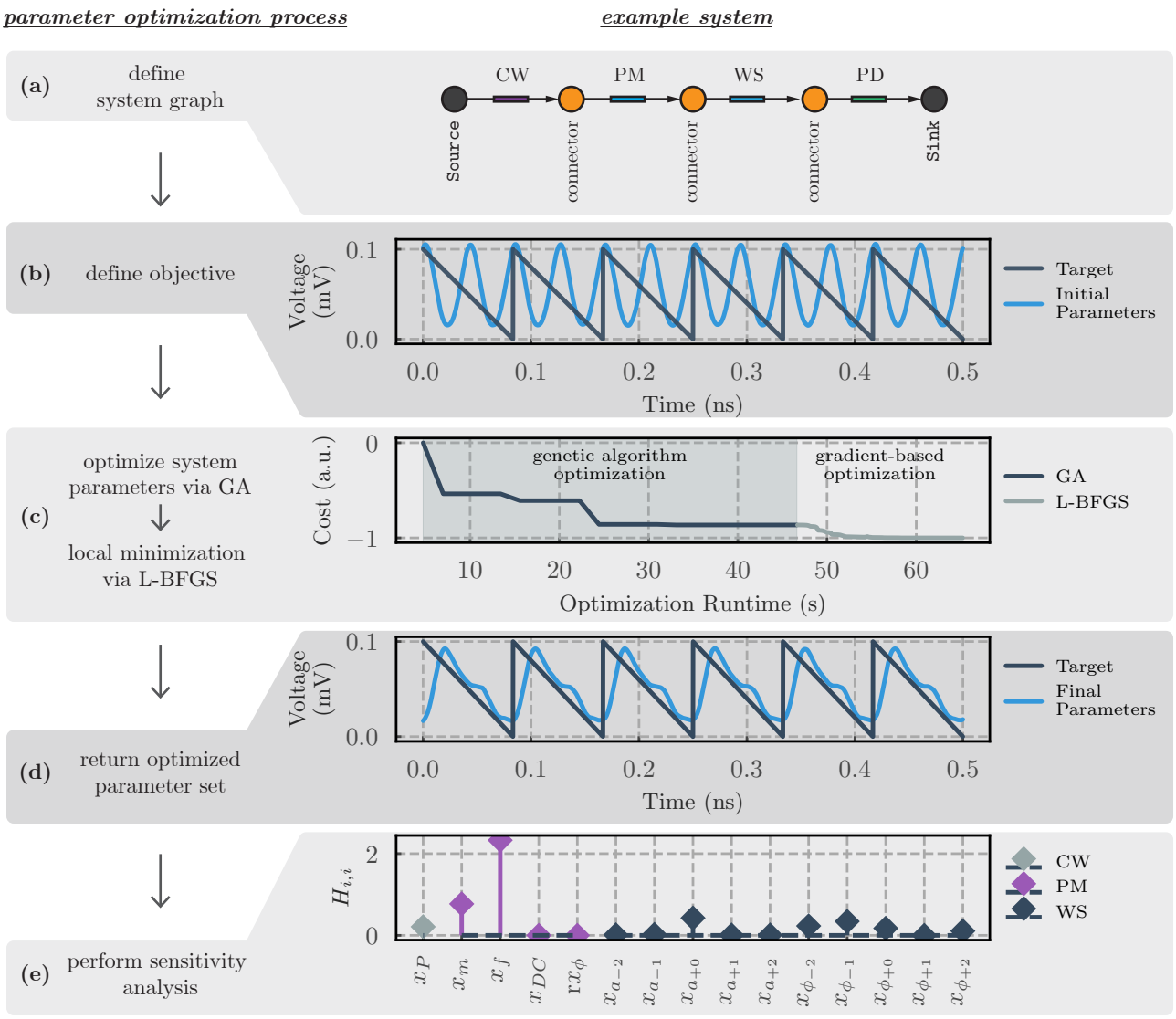


Figure 3.6 – Parameter optimization procedure on a fixed graph topology. **(a)** The fixed system graph used, which generates arbitrary waveforms. The cost function for this system is the ℓ^2-norm similarity between the target optical signal (sawtooth waveform; solid, dark blue line) and the generated optical signal (solid, light blue line). **(b)** An initial starting set of parameters have poor performance, which are **(c)** optimized via a GA and L-BFGS optimization routines to produce a **(d)** high-performing set of component parameters. **(e)** Sensitivity analysis on the final parameter set provides insight into which parameters most effect the performance.

The topology optimization procedure is an evolutionary algorithm. A population of optical system graphs iteratively mutate and change, optimizing the parameters along the way, with high-performing graphs more likely to remain and/or produce new systems. Each graph has the propagation and evaluation function compiled and the parameters optimized as described in Sec. 3.3.1. After the parameters on each graph have been optimized, a new population is created – and the process is repeated. Each generation, high-performing graphs are selected to produce the next population and are stochastically mutated (i.e. their topologies changed, optical components are added/removed) to produce new systems. This process iterates until a stopping criteria is met, and

`ASOPE` returns a family of the top performing optical systems The overall algorithmic flow of the topology optimization is outlined in Fig. 3.9.

Mutation operators

The main method for `ASOPE` to generate new optical systems is the graph mutation operators. These operators take a directed acyclic graph, and return a new directed acyclic graph, slightly altered. Only valid mutations which result in a physically valid optical system can be applied. These mutation operators, demonstrated in Fig. 3.7, are:

Add Edge Adds a single new edge between two existing vertices. In physical terms, this is the addition of a new spatial path in parallel with existing spatial paths (e.g. an interferometric setup).

Add Node & Edge Adds both a new component and connector into a current spatial path. Physically, this is equivalent to adding a new component serial into an existing spatial path.

Remove Edge Removes an edge (between vertices u and v) from the system. No vertices are removed, unless there is no other path between u and v (i.e. the graph is unconnected) in which case u and v are collapsed into a single vertex.

Swap Model Swaps a component for a different component of the same type. For example, an optical source can swapped for a different optical source (e.g. continuous wave laser to pulsed laser), a single-path element can be swapped for a different single-path element (e.g. phase modulator to dispersive fiber), etc. However, models cannot be swapped for elements of a different type (e.g. a modulator cannot be swapped for a detector).

These mutation operators are denoted as, $M_i = M_{\text{AddEdge}}$, $M_{\text{AddNodeEdge}}$, $M_{\text{RemoveEdge}}$, and $M_{\text{SwapModel}}$, respectively. These mutation operators also require a choice of where in the graph they should be applied, e.g. which edge to remove, in between which nodes to add an edge, etc. Thus, the same mutation operator could be applied in a number of different locations on a single graph, which we denote as $M_{i,j}$ – where i is the operator (AddEdge, AddNodeEdge, RemoveEdge, SwapModel) and j specifies where in the graph it will be applied. Each of these mutation operators maps a graph to a new graph; $M_{i,j} : G \to G'$. In Fig. 3.8, these operators are sequentially applied to demonstrate how new, and varied, optical systems are produced. These graph mutation operators are based on those used in evolving neural network architectures [31], but adapted and extended to be used with optical system graphs. Fig. 3.8 shows how these mutation operators, applied sequentially, produce new and varied optical systems.

At each new generation of the topology optimization algorithm, a new population of graphs is generated by applying these mutation operators probabilistically, i.e. such that each operator $M_{i,j}$

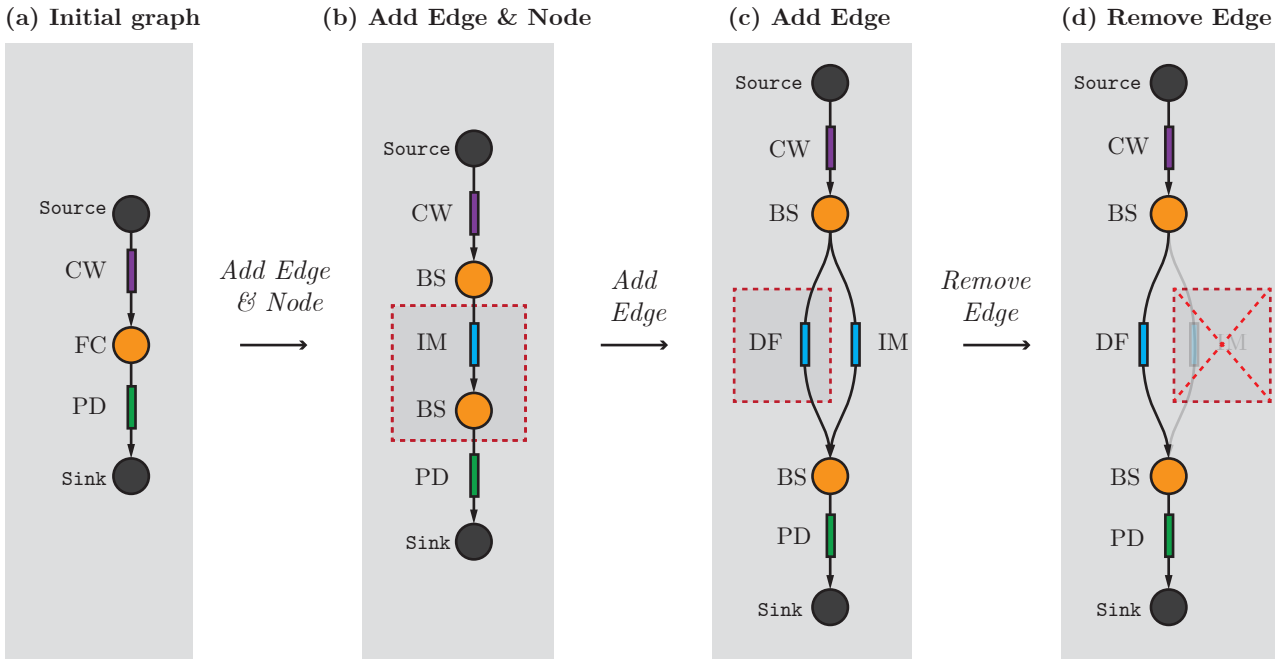


Figure 3.7 – Mutation operators to produce new optical systems. CW: continuous wave laser, IM: intensity modulator, DF: dispersive fiber, PD: photodiode, BS: beam-splitter (note that when a BS element has one incoming edge and one outgoing edge, they are equivalent to a simple connector and do not alter the propagating optical field).

has a probability of $\Pr(M_{i,j})$ of being chosen and applied (only one mutation is applied at a time). Defining these probabilities is important to the convergence and performance of the algorithm. Three such methods are outlined here:

Naive Implementation: The first, naive definition of the mutation probabilities is to simply allow all valid mutation operators to have an equal probability of being applied, which is kept constant throughout the evolution process,

$$\Pr(M_{i,j}) = \frac{1}{|\{M_{i,j}\}|} \tag{3.11}$$

where $\{M_{i,j}\}$ is the set of all valid operators. However, this may skew the trajectory of the optimization, as a mutation operator, M_i, that can be applied on a graph in more ways, is more likely to be drawn. For example, there are often many ways for a new edge to be added, but less ways for them to be removed – which could lead to a rapid increase in the size of the graphs.

Constant Probabilities: The second method, instead, sets a constant probability for each type of mutation, $\Pr(M_i)$, such that there is a constant probability of applying each of AddEdge, AddNodeEdge, RemoveEdge, SwapModel. Thus,

$$\Pr(M_{i,j}) = \frac{\Pr(M_i)}{|\{M_{i|j}\}|} \tag{3.12}$$

where $\{M_{i|j}\}$ is the set of all valid mutation operators for M_i. By balancing the probability of increasing and decreasing the graph size, the system graphs are less likely to continually increase in size and complexity.

Adaptive, Hessian-guided Probabilities: The final, more sophisticated approach for setting the probabilities of different mutations throughout optimization uses: i) adaptive probabilities and ii) Hessian-based sensitivity analysis to remove components that do not effect the system performance. Adaptive probabilities change throughout the topology optimization. They first expand the size of the graphs, and as the topology optimization proceeds, contract them. By trading-off between high probabilities for AddNode and AddEdgeNode mutations, to high probability of RemoveEdge mutation, the trajectory of graph sizes and optimization can be altered. This provides a mechanism to start with a broader exploration of the system design space, and then later, further optimize known solutions by reducing their complexity. This is conceptually similar to adaptive step sizes [112, 113] or an epsilon-greedy approach [114] in other algorithms. Next, the Hessian-based sensitivity analysis guides which components are removed from the setup, ideally without significantly reducing performance, by adjusting the liklihood of removing them based on their sensitivities. See Appendix A.2 for further discussion on this method. Other demonstrations of topological optimization have used a simplification step to remove components by iteratively removing elements and re-evaluating [5]. In this context, however, this require re-optimizing the system parameters each time, which is undesired.

See Sec. 4.2.3 for a comparison of these methods.

Selection and elitism

After the current population has been evaluated, high-performing system graphs are selected to produce the next generation. Elitism, a part of selection, copies the top few performing graphs, unchanged, over to the next generation. This ensures that high-performing systems are kept and can continually be improved upon. Empirically (see Fig. 4.8), keeping at least a small portion of the best-performing systems improves the convergence and final scores of the topology optimization. For each generation, `ASOPE` keeps the top $\sim 5\%$ of system graphs unchanged into the new generation, as an empirically chosen hyperparameter (see Fig. 4.8).

The difference between elitism and selection, here, is that elitism is deterministic: the top N performing graphs are always kept. Selection, on the other hand, makes a stochastic decision on which graphs to keep. A number of common selection methods exist, many of which scale the liklihood of selecting a graph with its score [38, 115]. For topology optimization in `ASOPE`, tournament selection is implemented as the default method in `ASOPE`; this selects high-performing

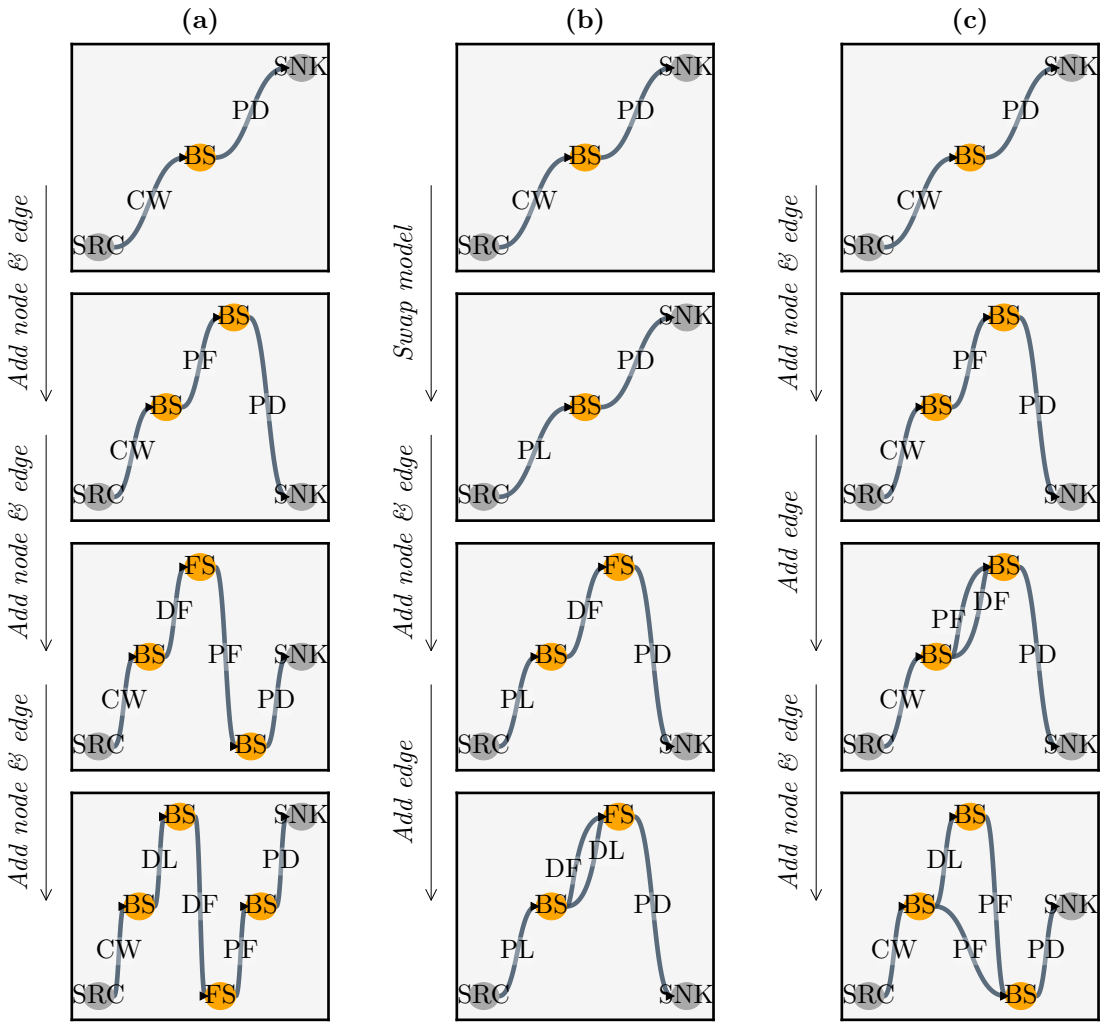


Figure 3.8 – Exploration of system graph topology via mutation operators. **(a-c)** Starting with the same system (top row), a different sequence of mutation operators are applied to generate new experimental setups, which can then be automatically simulated, analyzed, and optimized. PL: pulsed laser, CW: continuous wave laser, DF: dispersive fiber, DL: delay line, PF: programmable filter, BS: beam splitter, FS: frequency splitter, PD: photodiode, SRC/SNK: `Source` and `Sink` nodes, respectively.

graphs with higher probability, but it does not depend on the absolute scores of the graphs, which makes it more robust to a variety of cost functions.

Hall of Fame

A number of distinct, diverse optical systems may perform well on the desired objective – where each could use different optical components and techniques. To suggest a number of different possible systems, the *Hall of Fame* (HoF, [45]), stores the best-performing optical systems explored so far and clusters graphs that are similar to maintain diversity.

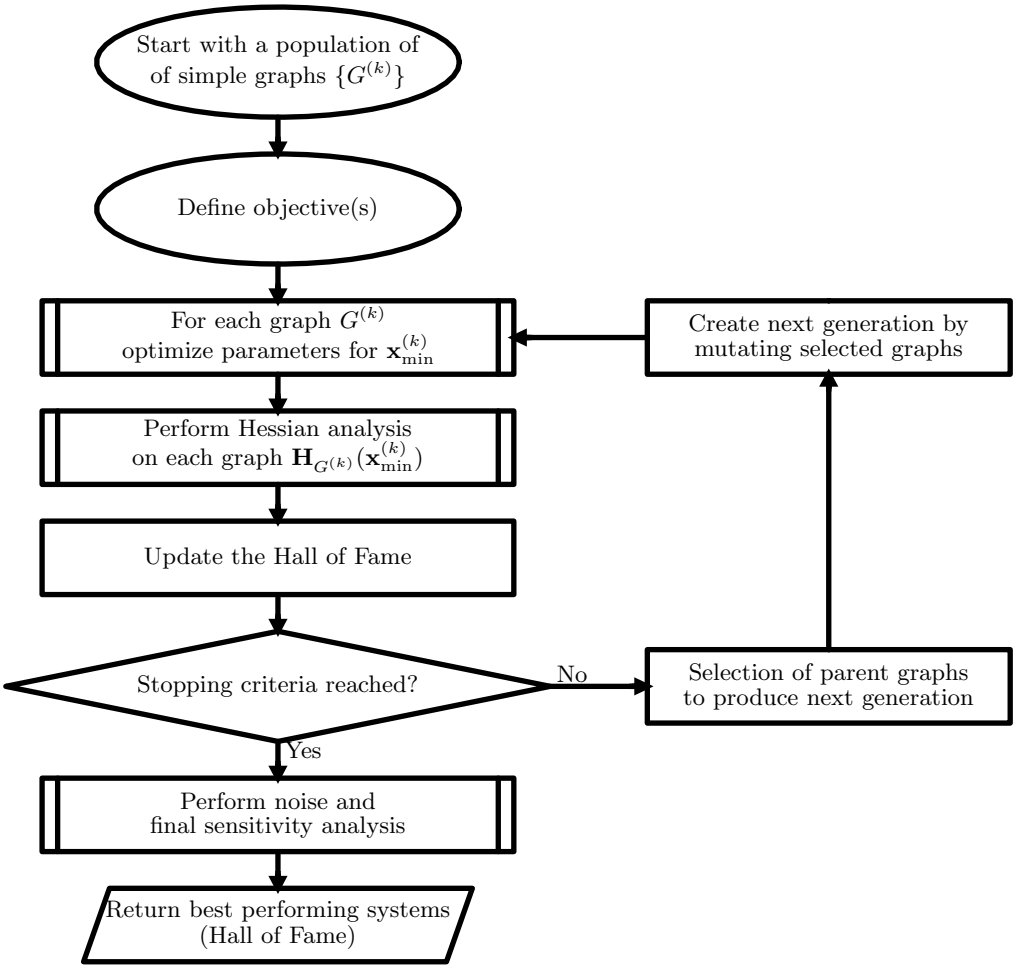


Figure 3.9 – Flowchart of topology optimization routine.

After each generation, the current population of system graphs is compared against the HoF, which is updated with new, better-performing systems. A graph is only added to the HoF if, and only if, it has a better score than a current member of the HoF. Specifically, it is added to the k-th position of the HoF if it scores better than the current k-th member of the HoF. To ensure the same system (or highly similar systems) do not dominate throughout the optimization, a candidate solution is only added to the HoF if it is sufficiently dissimilar to current members. `ASOPE` uses the *Graph Edit Distance* (GED), a pseudo measure of similarity between two graphs, G_1 and G_2, which calculates the cost to transform G_1 into G_2 using elementary transformations [46, 47]. These transformations are, indeed, similar to the mutation operators described above. We set a threshold such that no two graphs in the Hall of Fame can be within an GED threshold away from each other. This threshold is arbitrarily set to ensure that highly similar graph topologies do not 'overtake' the HoF and remove any diversity in the solutions proposed.

Parallel optimization

Topology optimization is inherently parallel and thus well-suited for distributed computation. For each generation of the topology optimization, each optical system graph is optimized and evaluated on separate CPU cores. `ASOPE` is built to seamlessly scale from single-core, through multi-core, to cluster computations [116]. The parallel nature provides an approximately linear speed-up in the optimization time with the number of cores. However, overhead in distributing processes among the worker CPUs will cause the speed-up to be sub-linear. The more important deviation from a linear speed-up is the difference in the optimization time for each graph – as the time it takes to optimize a single topology can vary depending on the parameters which must be optimized (small graphs or graphs with simple elements will be optimized faster than more complex setups). Optimization runs were accomplished using a cloud computing infrastructure, where each computational experiment presented in Chapter 3 was performing on a 16 CPU machine with 16 GB of RAM [117].

CHAPTER 4

Results and Discussion

In this Chapter, results and benchmarking of the `ASOPE` algorithm for inverse design of optical systems are presented. First, I target the design of systems for three optical signal processing tasks: the manipulation of mode-locked laser pulse period, the generation of high-speed arbitrary electronic waveform, and the detection of phase changes. `ASOPE` is capable of finding known, non-trivial systems which generate desired optical fields, and is able to discover multiple techniques in hours of runtime. Second, the sensitivity analysis and optimization routines (both topology and parameter) detailed in Chapter 3 are benchmarked and compared.

4.1 Demonstration of ASOPE on known photonics applications

4.1.1 Division and multiplication of pulse repetition times

The manipulation of pulsed laser repetition rate, i.e. the time interval between consecutive pulses emitted by the laser, is a common and important technique in ultrafast optical systems [48–52, 118]. Methods for manipulating the repetition time are dependent on the input pulse characteristics: including the input repetition time itself ($x_{t_{\text{rep}}}$), optical power (x_P), and pulse width (x_τ). Methods include: interleaving pulse trains which have been split and delayed via interferometers [50], using temporal and spectral Talbot effects [6, 49, 118], shaping the spectral amplitude and/or phase [48, 51, 52], and pulse picking [119].

Utilizing the `ASOPE` algorithm, we target the multiplication and division of the pulse repetition time for a fixed input pulsed laser. Throughout the topology optimization, a single pulsed laser is fixed as the optical source (i.e. it is not removed or swapped through the topology optimization). The input pulsed laser has fixed parameters of the pulse width, repetition time, and peak power (Gaussian-shaped pulses are assumed). The target of the optical systems is to manipulate the pulse train such that the output optical field is also a pulse train of pulses identical to those emitted by the laser, but with a different, user-defined repetition time (an arbitrary multiple of the input). Consider the multiplication or division of a lasers repetition time by a factor of p/q, where $p, q \in \mathbb{Z}$

(such that when $p > q$ the repetition time increases and when $p < q$ it decreases). In Fig. 4.1, three example systems are presented – with input/target pulse parameters as in Table. 4.1. The evaluation function is the ℓ^2-norm between the generated pulse train intensity and the target pulse train intensity. The scaling factor p/q is arbitrary and user-defined, and scales both the pulse

Table 4.1 – Relationship between input and target pulse train parameters.

Input Pulse Train Parameters	Target Pulse Train Parameters
$x_{t_{\text{rep}}}$	$\left(\frac{p}{q}\right) x_{t_{\text{rep}}}$
x_P	$\left(\frac{p}{q}\right) x_P$
x_τ	x_τ

period ($x_{t_{\text{rep}}}$) and the peak pulse power (x_P) as we assume an ideally loss-less manipulation which conserves energy between the input and output pulse trains.

In Fig. 4.1, three different input/target pairs are provided to the topology optimization for pulse repetition rate manipulation. The algorithm returns and suggests optical systems which, when analyzed, interestingly make use of a number of the known techniques listed above. First, Fig. 4.1 top row, uses an optical delay-line element (DL) in parallel (i.e. in an interferometric setup) such that the initial pulse train is split, delayed by a time-delay of $x_{t_{\text{rep}}}/2$, and the two pulse pulse trains are then recombined (temporally interleaved). Such methods are widely utilized for manipulating a pulse train, akin to concepts in [50]. Fig. 4.1, middle row, uses the temporal Talbot effect, utilizing the chromatic dispersion of a dispersive fiber over a set length to create a sub-image of the initial pulse train [49]. Fig. 4.1, bottom row, uses a combination of the temporal and spectral Talbot effect, as presented in [6].

A problem which can arise during topology optimization is the introduction of components which are unnecessary, but do not necessarily reduce performance. For example, in Fig. 4.1, bottom row, it is known that manipulation of a pulse train can be accomplished with one dispersive element and one phase modulation element – as such, the first WaveShaper (WS) element is not strictly necessary. The parameters on the elements may be such that it could not have been removed without re-optimizing the parameters. Thus, naively removing such elements without re-optimizing the parameters is likely to result in poor performance. In cases such as this, sensitivity analysis tools such the Hessian matrix, can be useful for understanding the impact of removing a component may have and help guide the topology optimization towards simpler setups.

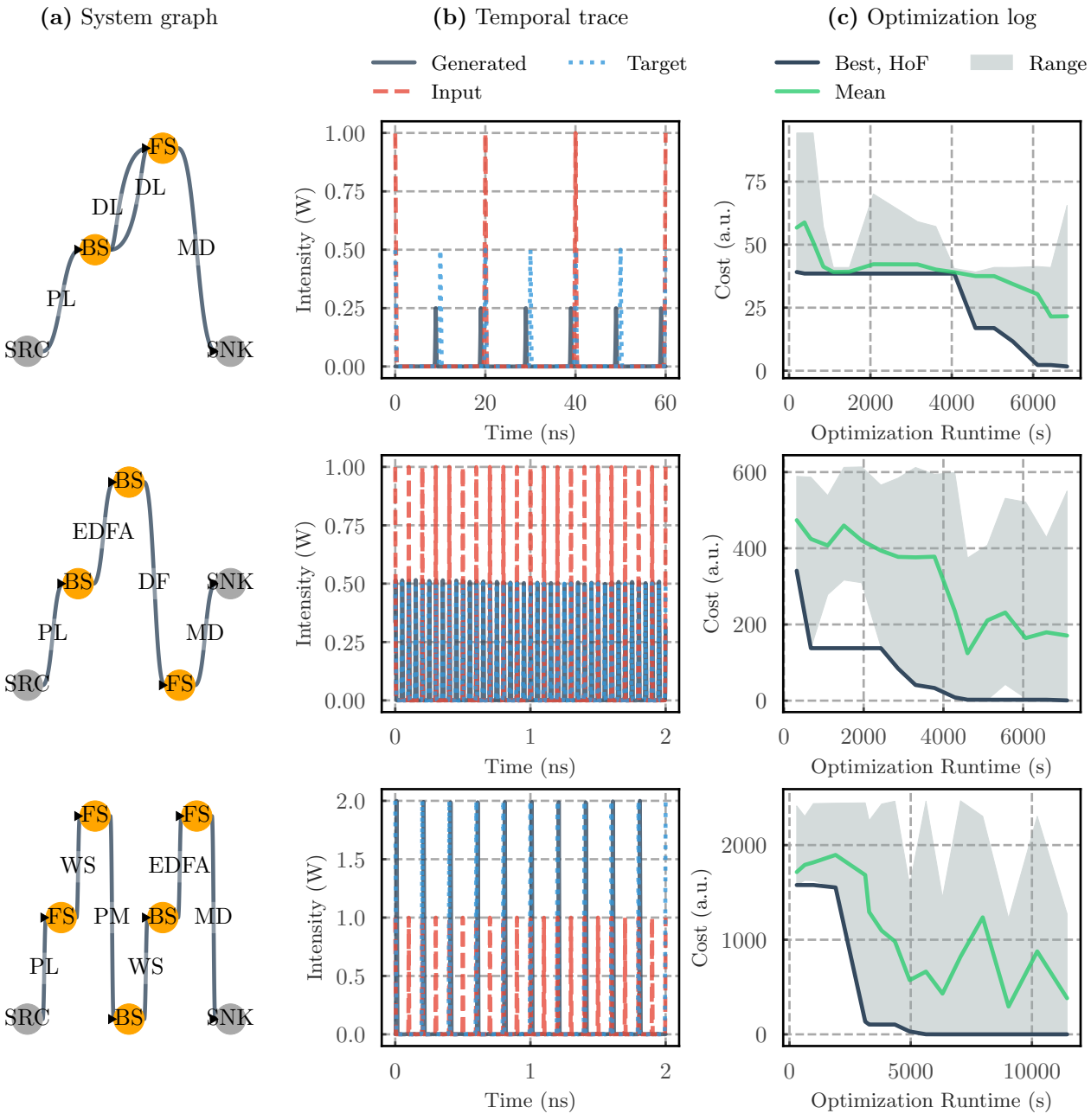


Figure 4.1 – Optical systems which multiply or divide the repetition rate of a pulsed laser are targeted, using the cost function of the form in Eqn. 3.7. The suggested systems from the HoF utilize a number of known optical techniques. The simulation parameters are as follows: Top row: $p = 1, q = 2$, input pulse parameters $x_\tau = 1.0$ns, $x_{\text{rep}} = 20$ns. middle row: $p = 1, q = 3$, input pulse parameters $x_\tau = 3$ps, $x_{\text{rep}} = 0.1$ns. Bottom row: $p = 2, q = 1$, input pulse parameters $x_\tau = 3$ps, $x_{\text{rep}} = 0.1$ns. PL: pulsed laser, DF: dispersive fiber, DL: delay line, EDFA: erbium-doped optical amplifier, BS: beam splitter, IM: intensity modulator, PM: phase modulator, WS: Waveshaper, MD: measurement device (calculates the intensity of the incident optical field), SRC/SNK: `Source` and `Sink` nodes, respectively.

4.1.2 Photonic-assisted radio-frequency arbitrary waveform generation

The generation of high-speed, arbitrary electronic waveforms is of fundamental importance in a broad variety of fields [53–57]. One method utilizes the bandwidths and control mechanisms of optics to produce arbitrary waveforms optically, followed by optical-to-electronic conversion (via a photodiode). Methods use both continuous and pulsed optical inputs, manipulating the optical field either in the spectral domain via filters and dispersion [56, 57], in the time-domain with delays [54, 55], or a mixture of both. For a review of such methods, see [54] or [58].

Applying `ASOPE` to the problem of suggesting setups for photonic-assisted arbitrary waveform generation, we search for optical systems which produce the desired electronic waveforms after optical-to-electronic conversion (i.e. measuring the simulated temporal voltage signal at the photodiode). The objective function employed is the ℓ^2-norm between the generated electronic signal and the target signal (with any global phase differences between the target and generated signals compensated for). A target waveform is defined in the RF domain by its shape (e.g., square, saw, pulsed, bit pattern, etc.), pattern repetition rate, and peak-to-peak voltage amplitude. In the following computational runs, we target the generation of arbitrary high-speed waveforms for speeds ranging from 10 GHz to 40 GHz as a proof-of-concept.

Fig. 4.2 demonstrates three suggested systems for the generation of a 12 GHz square, 12 GHz sawtooth, and 36 GHz sawtooth electronic waveforms. These systems, along with others in the HoF of the optimization runs, use a variety of techniques for generating radio-frequency waveforms. However, distinguishing between them is more challenging than as in Sec. 4.1.1. Each system generally uses the process of generating new spectral components from a continuous wave laser via time-dependent modulation, and then manipulating the phases and amplitudes of the spectral components such that interference creates the desired waveform envelopes.

Understanding the important mechanisms and contributions of components to the overall operation of a system is challenging. In forward design methods, where a designer invokes a new design and manually investigates the properties, the designer has a deeper understanding of the underlying processes at work. On the opposite side in inverse design, when an algorithm 'spits-out' a new design, the important (and interesting) processes are not always initially clear. Again, this is where analysis techniques, such as the Hessian analysis outlined in Sec. 3.2, can be useful – interactions between optical elements and relative importance of components can be estimated (see Sec. 4.2.2).

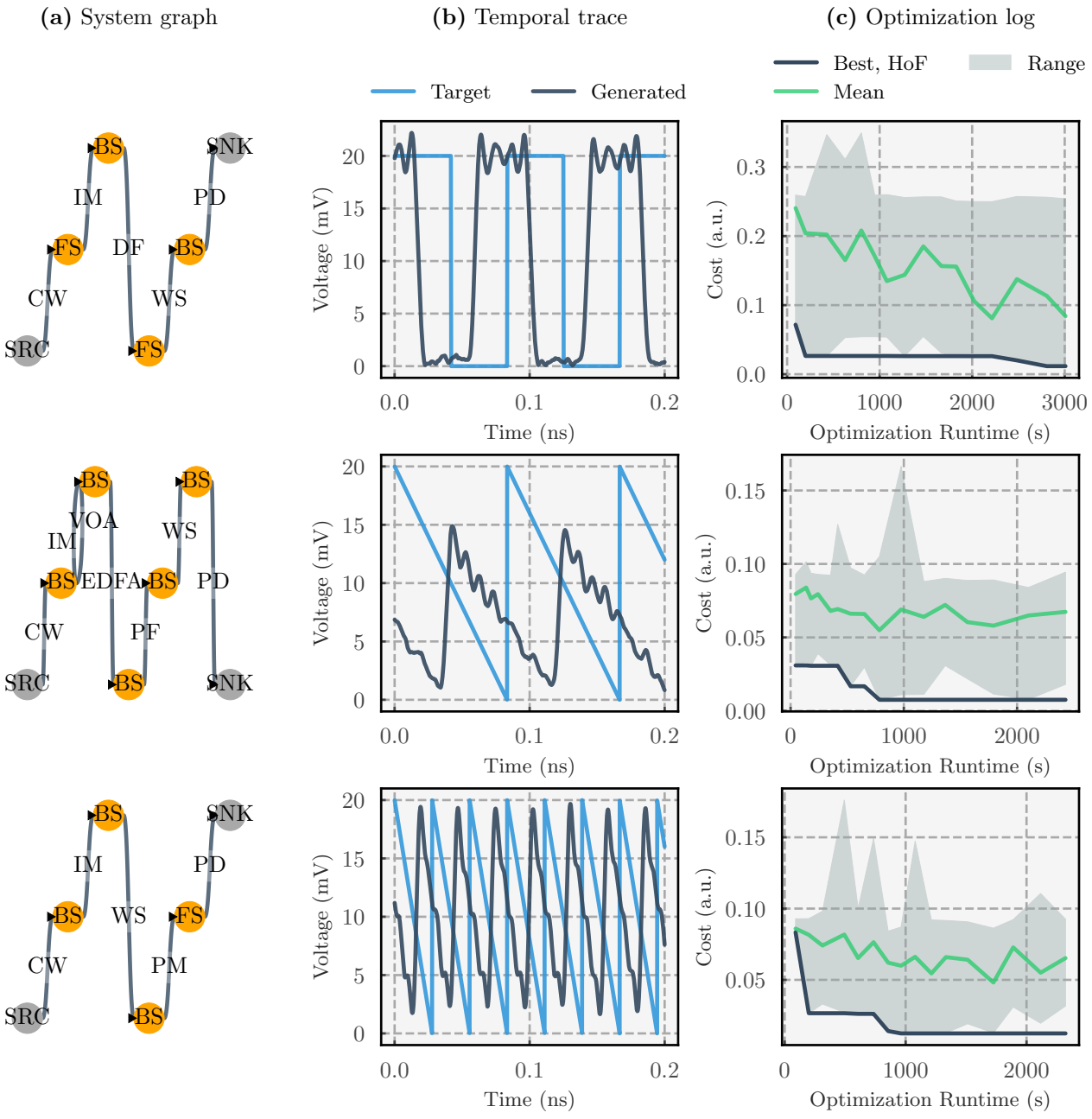
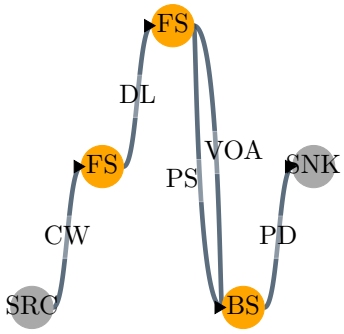


Figure 4.2 – Targeting optical systems for photonic-assisted arbitrary waveform generation. The cost function is of the form in Eqn. 3.7. The targeted waveforms are as follows Top row: 12 GHz square waveform. Middle row: 12 GHz sawtooth waveform. Bottom row: 36 GHz sawtooth waveform. CW: continuous wave laser, DF: dispersive fiber, DL: delay line, EDFA: erbium-doped optical amplifier, BS: beam splitter, IM: intensity modulator, PM: phase modulator, WS: Waveshaper, PF: programmable filter, VOA: variable optical attenuator, PD: high-speed photodiode.

4.1.3 Phase-sensitive systems

Phase-sensitive optical experiments are a foundational underpinning of countless fundamental discoveries [59, 60] and optical technologies [61, 62]. The most common of these are interferometers,

(a) System graph



(b) Optimization log

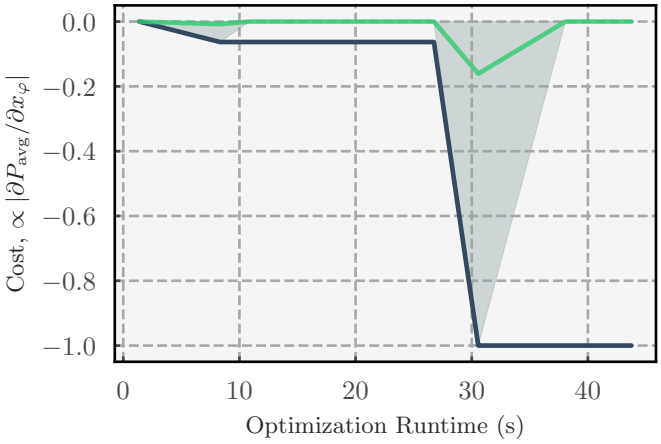


Figure 4.3 – Inverse design of phase-sensitive optical systems, which targets systems which maximize the sensitivity of average optical power to a phase perturbation applied at the phase shifter (PS). The algorithm returns Mach-Zehnder interferometric systems that match common experimental methods for increasing sensitivity such as maximizing input optical power, balancing loss between arms, and setting the central phase shift to $\pi/2$. CW: continuous wave laser, DL: delay line, VOA: variable optical attenuator, PD: photodiode, BS: beam splitter.

such as the Michelson or Mach-Zehnder setups, which use the optical interference of two spatial paths to detect slight variations in the relative phase between the paths. To demonstrate the flexibility in defining cost functions via automatic differentiation, we target the design of phase-sensitive optical systems. A phase-shifting element (PS) is placed in the initial system, and cannot be removed or swapped during topology optimization. The cost function is proportional to the sensitivity of the output optical power to this phase shift, $F = -|\partial P_{\text{avg}}/\partial x_{\varphi}|$, as defined as in Eqn. 3.8. The suggested system, shown in Fig. 4.3, is a Mach-Zehnder-type interferometric setup. The parameters in this system also follow common experimental techniques to optimize phase sensitivity in interferometers: the continuous wave laser optical power is at its maximum power and the optical attenuator in the second arm of the interferometer has loss equal to that from the phase shifting element. This proof-of-concept demonstration could be extended to search for optical systems that have properties such as high sensitivity, or *vice versa*, and search for systems which are highly stable to perturbations.

4.2 Comparing and benchmarking of tools

4.2.1 Comparison of parameter optimization tools

Fig. 4.4 compares the runtime and convergence of a variety of parameter optimization routines, including ADAM, L-BFGS, PSO, GA, as well as hybrid algorithms (e.g., L-BFGS+GA, ADAM+GA, and L-BFGS+PSO). The optical system and evaluation function used is the same as in Fig. 3.2, the generation of a 12 GHz sawtooth electronic waveform.

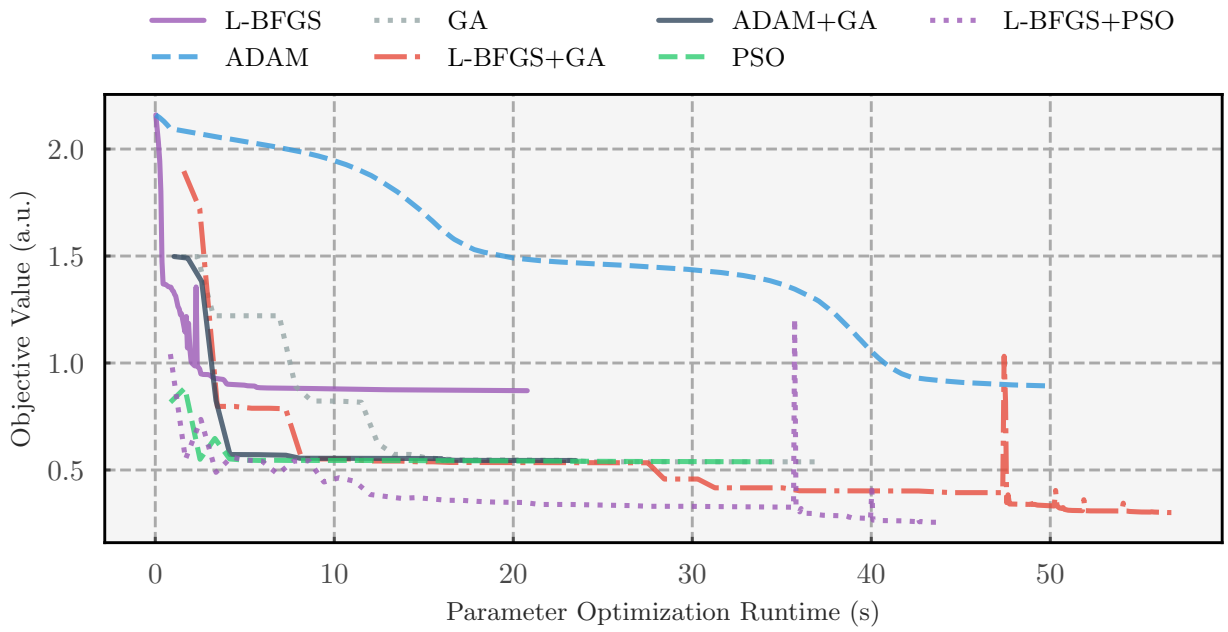


Figure 4.4 – Comparison of parameter optimization routines for a single optimization run (topology used is the same as in Fig. 3.2a). In general, hybrid algorithms have the best performance, but require more computational time. Further tuning of hyper-parameters could improve convergence of various algorithms.

Stochastic gradient descent algorithms (e.g., RMSProp, ADAM) are popular for the training of neural networks as they perform well on tasks with thousands to millions of parameters. L-BFGS, on the other hand, often performs better on optimization problem with tens to hundreds of parameters as it can converge quicker. However, both of these routines are sensitive to the initial starting points. We can see that for repeated optimization runs with different initial guesses these algorithms have a larger variance in final score compared to 'global' search routines such as a GA or PSO. In Fig. 4.4, we see that the L-BFGS algorithm converges very quickly but has high variance in final score (Table 4.2). This is a motivation for using a hybrid optimization scheme. For the tested problems, hybrid algorithms perform the best – with a lower average final cost value and less variability in the final scores, though they require more computational time.

Table 4.2 – Comparison of parameter optimization routines for a fixed system topology. Mean and standard deviations are from 10 optimization runs with different random seeds and starting points.

Parameter Optimization Routine	Type of Optimization Routine	Runtime (s) (mean ± s.d.)	Score (a.u.) (mean ± s.d.)
L-BFGS	gradient-based	13.33 ± 9.32	0.67 ± 0.15
ADAM	gradient-based	50.68 ± 0.53	0.72 ± 0.16
GA	evolutionary	37.69 ± 0.46	0.50 ± 0.08
L-BFGS+GA	hybrid	52.99 ± 3.59	0.47 ± 0.08
ADAM+GA	hybrid	23.72 ± 0.35	0.55 ± 0.01
PSO	evolutionary	34.86 ± 0.36	0.39 ± 0.13
L-BFGS+PSO	hybrid	42.09 ± 3.53	0.40 ± 0.15

4.2.2 Comparison of sensitivity analysis tools

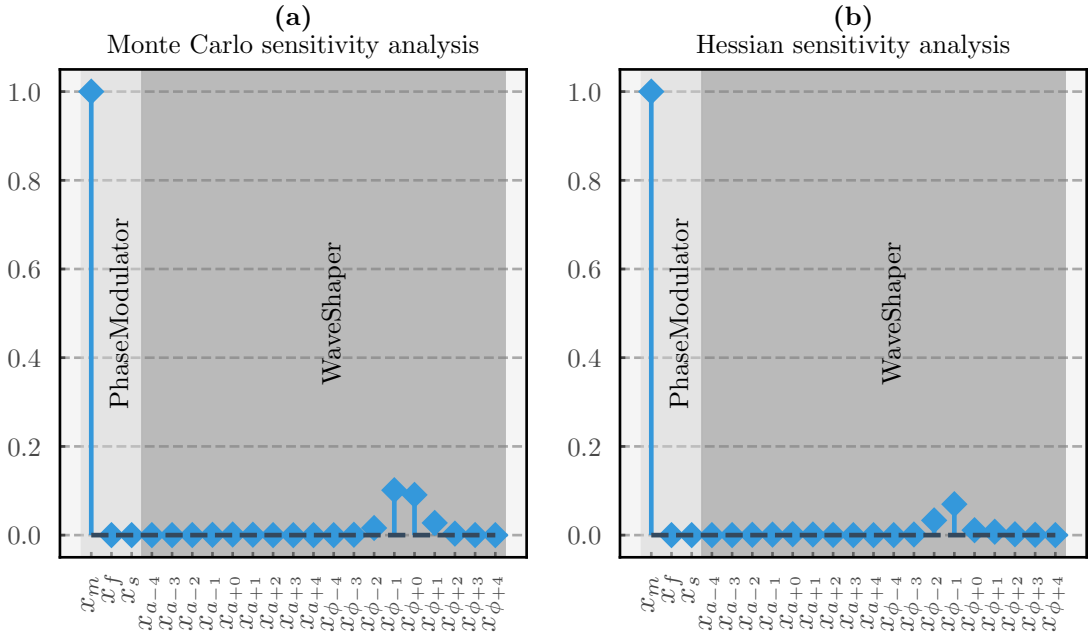


Figure 4.5 – Comparison of sampling method and Hessian-based sensitivity analysis (topology and parameters used are as in Fig. 3.6)a-b. For **(b)**, the stem plot values are the diagonal of the Hessian matrix (i.e. $\partial^2 F/\partial x_i^2|_{\mathbf{x}_{\min}}$). Similar relative sensitivities are found between the two methods.

Fig. 4.5 demonstrates a comparison between a Monte Carlo sensitivity analysis and Hessian sensitivity analysis on a fixed topology and optimal parameters (the same system as in Fig. 3.2). The stem lines in Fig. 4.5a are the individual sensitivities calculated via a Monte Carlo method (Eqn. 2.2), while the stem lines in Fig. 4.5b are the diagonal values, H_{ii}, of the Hessian matrix $\mathbf{H}(\mathbf{x}_{\min})$. There is close agreement between the two methods for relative sensitivities. The two techniques will not return identical analyses, as they make different assumptions and use significantly different computational methods. As the Monte Carlo method is only first-order, combinatorial effects and sensitivities cannot be estimated, while they can with Hessian analysis.

The Hessian analysis of two system graphs is shown in Fig. 4.6 as a color heatmap for the values of H_{ij} (parameters are grouped by the optical component they belong to in order to reduce text in the plot). The full Hessian analysis of an optical system that has tens to hundreds of parameters, would be challenging to visualize. As such, full analyses of the matrices in Fig. 4.6 are not detailed here. However, the interactive dashboard (see Appendix A.3) can be used dynamically plot, explore, and understand the topology, parameters, score, etc. of large, complex optical system graphs, including their principal components and curvatures. The Hessian matrix can contain

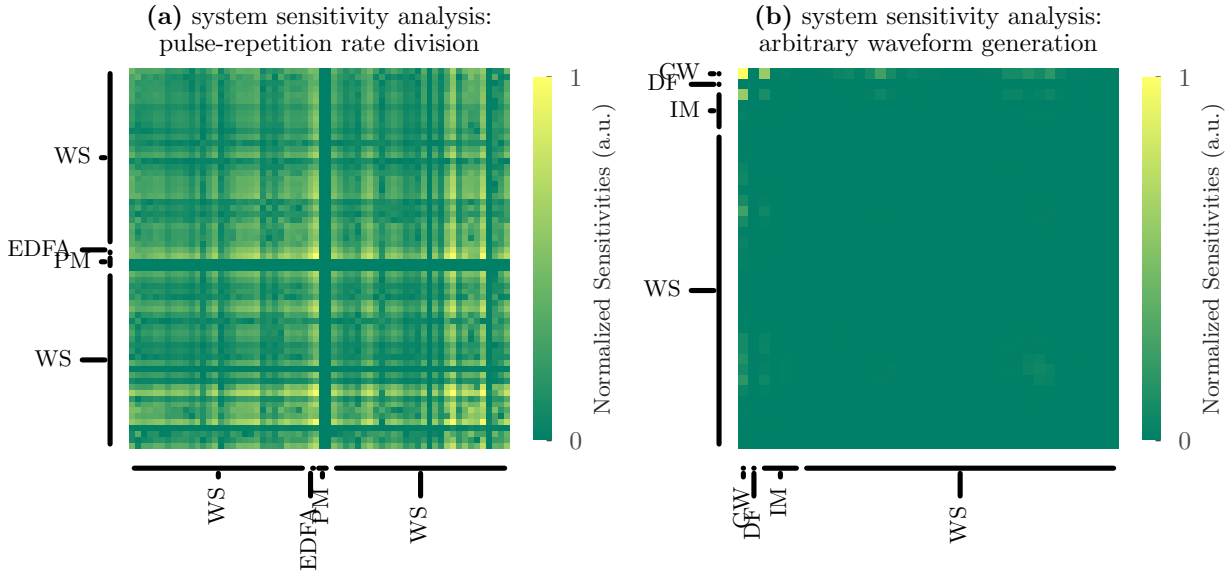


Figure 4.6 – Examples of Hessian matrices of optical systems. **(a)** Hessisan matrix for the system presented in the bottom row of Fig. 4.1. **(b)** Hessian matrix for the system presented in the top row of Fig. 4.2. The large values of the off-diagonal elements (particularly for (a)) demonstrate the complex couplings between design parameters which could make intuitive understanding and design of such systems challenging.

interesting and useful information about the operation of the system under study. As discussed in Sec. 3.2, the Hessian matrix indicates if a point is a local minima, maxima, or saddle-point – i.e. the shape of the objective function landscape at a single point. The utility of understanding the function landscape goes beyond this, however, and one can match patterns in the Hessian matrices to experimental understanding and intuition.

Take, for example, the optical system in Fig. 4.7a (where the evaluation function is the ℓ^2-norm between a target pulse train and generated pulse train, as in Sec. 4.1.1). The system is composed of a pulsed laser and two, back-to-back dispersive fibers (identical types of fibers, both parameterized by the fiber lengths, x_z). This system uses chromatic dispersion to broaden the pulse, eventually forming sub-images via the temporal Talbot effect (i.e. a frequency-dependent, quadratic phase shift). From the diagonal elements of the Hessian matrix (Fig. 4.7b), we can immediately see that the pulse repetition time has a large influence on the evaluation function. Analytic, closed-form equations modeling this phenomena agree, supporting that the temporal Talbot effect has a strong dependence on the pulse repetition time [49, 118, 120].

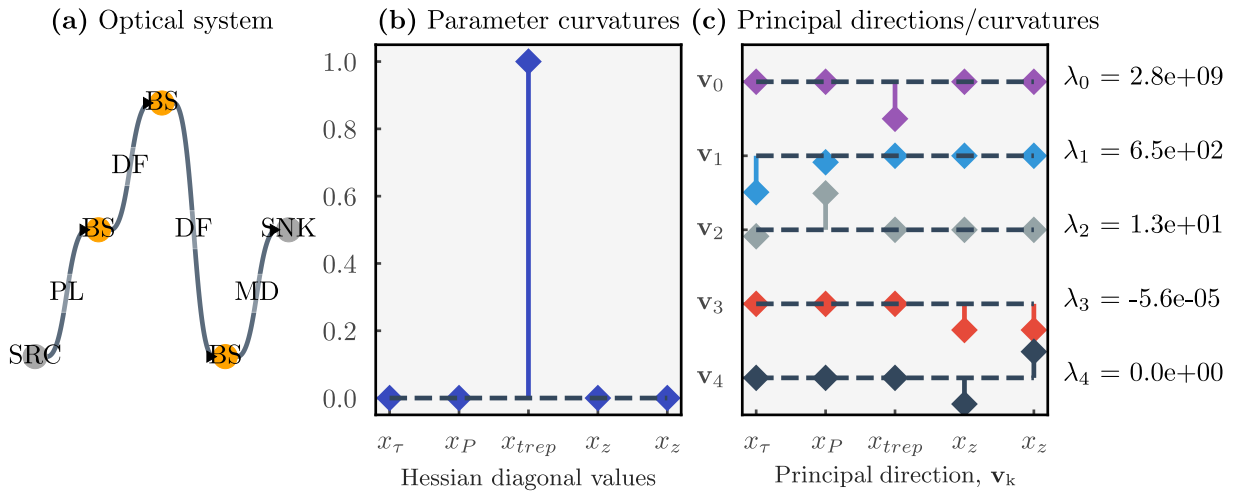


Figure 4.7 – Hessian analysis of an optical system for pulse repetition rate manipulation. **a)** The optical system, G, under consideration, comprised of a pulsed laser and two, back-to-back dispersive fiber elements. **b)** The normalized parameter curvatures of the evaluation function for the system. **c)** The principal directions and curvatures of the evaluation function. Patterns in the principal (and parameter) curvatures/directions, which indicate directions and combinations of parameters that are highly sensitive or stable. In particular, perturbing the parameters in the direction of $\mathbf{v}_4$ is highly stable and has no effect on the system operation.

Further understanding of this system can be gained from looking at the principal directions and curvatures (via eigendecomposition). For example, principal curvatures that are small, $|\lambda_i| \approx 0$, indicate that the associated principal directions is highly stable. This piece of information from the Hessian analysis can help us infer properties about the optical system. Consider the principal curvature and direction, λ_4 and $\mathbf{v}_4$, in Fig. 4.7c. As $\lambda_4 = 0$, changing the parameters in the direction, $\mathbf{v}_4$, has no impact on the cost function – and this direction is highly stable. Experimentally, this means that when the length of the first fiber is increased by a length δx_z and the second fiber length is decreased by the same length, $-\delta x_z$, the objective score remains constant. This matches experimental intuition: with two back-to-back, identical optical fibers, if one is extended and one contracted, the overall length, and therefore effect on the light, remains constant. This is one example of how the Hessian matrix can provide meaningful information about to the system sensitivities and how component interact.

4.2.3 Comparison of topology optimization tools

Determining the effect that different topology optimization algorithms and hyperparameters have on convergence is essential. First, including elitism in the selection process demonstrates an improvement in convergence (see Fig. 4.8). Three different percentages of elitism in the topology optimization were tested: 0%, 10%, and 30%. Fig. 4.8 clearly indicates that elitism improves convergence – keeping the top 10% (blue lines) or 30% (grey lines) best-performing graphs tends to drive the optimization to overall better solutions compared to no elitism (purple lines). This is

because, with elitism, high-performing graphs can continue to be evolved in beneficial ways; without elitism, high-performing graphs may not be selected or may be mutated to a higher-cost setup. One downside of elitism is that it can crowd out other designs; slight variations of the graph may fill the population or HoF, such that the diversity of optical systems being optimized and evaluated is reduced.

Next, the topology optimization convergence is compared for different methods of determining the mutation operator probabilities. Fig. 4.9 presents the optimization convergence curve using the three mutation methods outlined in Sec. 3.3.2. Each line is the average of 5 runs. This

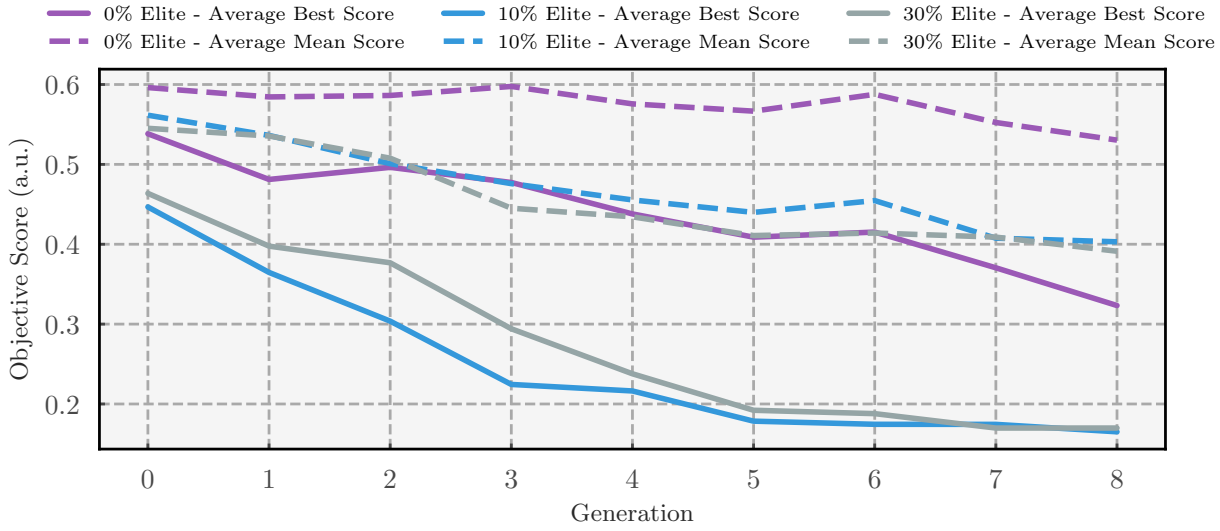


Figure 4.8 – Comparison of elitism in topology optimization, with three levels of graphs copied to the next generation. An improvement in optimization convergence is observed when a non-zero level of elitism is implemented.

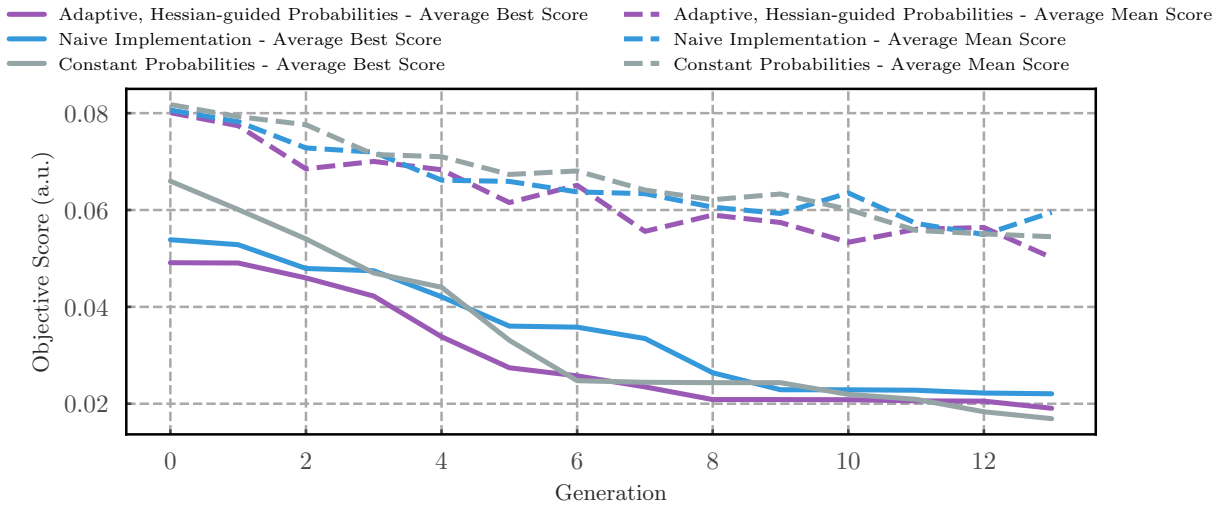


Figure 4.9 – Comparison of three different evolver methods, outlined in Sec. 3.3.2.

demonstrates minimal differences in performance between evolvers. However, as the problems scale in complexity, such that the optical systems include more components and spatial paths, methods such as the adaptive, Hessian-guided approach may demonstrate greater advantages over more naive implementations.

CHAPTER 5

Conclusions and Perspectives

In this work, I have developed and demonstrated algorithmic tools for the inverse design of ultra-fast optical systems. In my implementation, optical systems are represented by computational graphs, with the edges and vertices representing parameterized optical components. To produce new optical systems, graph mutation operators iteratively add/remove/change components in the graph. For each new optical system, automatic differentiation and heuristic algorithms are used to optimize component parameters and to estimate their sensitivities. A set of diverse, high-performing optical system topologies is updated throughout the search and at the end, suggested to the user for further analysis.

Using these tools and concepts, I demonstrate the design of optical systems towards three types of use-cases: pulse repetition rate manipulation, arbitrary waveform generation, and phase-sensitive detection. These illustrate `ASOPE`'s high degree of flexibility when it comes to defining cost functions. In particular, these include objectives defined in both the electronic and optical domains, as well as cost functions that include derivative terms. The optical systems suggested by `ASOPE` tend to use known, yet nontrivial optical methods/techniques. Further, I show that gradient-based sensitivity analysis, efficiently computed in `ASOPE` using automatic differentiation, can inform both experimental intuition and topological optimization methods. Using Hessian analysis, an understanding of relative parameter sensitivities and how parameters interact can help guide experimental implementations of suggested systems, as well as provide an efficient method to guide discrete topology changes.

Even without topology optimization, the simulation, sensitivity analysis, and parameter optimization tools in `ASOPE` could find use in forward-design methodologies. These could be useful to quickly simulate, optimize, and analyze a variety of fixed system setups. For instance, for the case of arbitrary waveform generation, repeatedly optimizing a system for the generation of different waveforms compiles an operational parameter library that can be referenced to quickly change output waveforms. Further, the Hessian-based analysis can find use in the risk-assessment, fault-analysis, or loss-budgeting of optical systems.

The models used in ASOPE are currently limited in that nonlinear optical interactions are not considered and that system looping is not permitted. In particular, highly nonlinear components, which require more sophisticated models to accurately simulate, are not featured in the component library. To have an accurate model of such components would require, for example, the Split-Step Fourier method [63], which would increase the computational resources required for ASOPE's convergence. Similarly, simulating optical loops requires more complex representations of an optical system, or more complex methods to calculate the result of the propagation [64, 65]. Future work will target the simulation of such complex components and loops, determining how to model them efficiently and how to optimally allocate computational resources. Finally, while the systems discussed in this thesis utilize a single optical source and detector, setups involving multiple could potentially be targeted by introducing new graph mutation operators.

Promising future improvements to the algorithm and developed software include: using advanced automatic differentiation methods, implementing new topological optimization concepts [31], and developing further ways to guide topological change from, e.g., cost function derivatives. Firstly, open-source projects, mainly in the field of machine learning, provide the functionality for simulating models and calculating derivatives on co-processing units (e.g., GPUs or TPUs), which can take advantage of huge data parallelism for efficient computation (see Appendix A.1 for further discussion on possible improvements to the automatic differentiation methods). This could allow for significant speed-ups and the ability to optimize and analyze more complex systems. Secondly, concepts from other graph optimization implementations could be adapted for use in the context of ASOPE. Specifically, concepts from neuro-evolution (finding optimal artificial neural network architectures) such as speciation and graph cross-overs should be further explored and adapted [31]. Finally, further methods of using sensitivity analysis to inform topological optimization, such as Hessian-guided mutation probabilities, should be explored. I have presented one method for informing the removal of optical components via the Hessian matrix, however, further linkages and metrics should be investigated. For example, the Hessian matrix indicates components that are tightly-coupled together and how much they influence system performance; so, highly-coupled, highly-sensitive pairs of component could be emphasized and be made more likely to remain in the system. This would combine gradient-based optimization methods with discrete optimization, which could find use in other inverse design use-cases which also use graph-based representations.

As demonstrated through this work, inverse design methods are capable of discovering known photonic systems through the iterative generation, testing, and analysis of new system topologies. Using the developed method, a variety of known optical systems are rediscovered through the computational search method – some of which have been first demonstrated relatively recently, and likely required years of development by domain experts. Using the presented methods, these systems, along with a family of other setups, are found and returned to the user in only hours of computing time. With these successful proofs-of-principle and the flexibility afforded via the general graph representation and automatic differentiation of system/element models, the method can be

immediately extended to a broad variety of new problems, disciplines, and technologies. As concrete examples for future work I identify three possible directions: targeting new objectives, including new/different optical degrees of freedom, and investigating the prospect of optical and electronic co-design. With the central algorithm, component libraries, optimization routines, and analysis techniques developed, new objectives and desired functionalities can immediately be tested and targeted. As one example, optimizing noise-based metrics (e.g., signal-to-noise, dynamic range) could find applications in telecommunications and sensing, where noise can be particularly detrimental. Cost functions based on derivatives (as in Sec. 4.1.3) are uniquely enabled through automatic differentiation and could find many uses in searching for highly sensitivity or highly stable photonic systems. Further metrics based on logistical considerations rather than system behaviour, such as system cost, mechanical weight, or stability, could also be included (and could potentially be used as a secondary design goals in multi-objective optimization). A second direction for future work is to consider new/different optical degrees-of-freedom, such as polarization or angular momentum, as well as consider integrated or free-space optical models. As the `ASOPE` package was built in a modular structure to allow for flexible and simple additions of new objective functions and/or components (see Appendix A.3), the topology and parameter optimization techniques can be readily extended to these new types of optical systems. A third direction for future work involves the co-design of optical and electronic systems. This could find impact in the design of integrated photonics, where in most cases it is highly desirable (or even necessary) to have electronic circuits and optical waveguide circuits monolithically fabricated onto a single chip, with electronic and optical signals operating in conjunction within the same circuit. Further, integrated optical circuits are often built from a small, specific library of components offered by the fabrication foundry – which is thus well suited to the `ASOPE` algorithm.

In summary, the work presented in this thesis demonstrates a novel design method for optical systems. I have developed an inverse system design method, along with the associated software tools for sensitivity analysis, parameter optimization, and topology optimization, which has been shown to rediscover a variety of known photonic systems. The results reported in this work suggest that the presented method has the potential to greatly benefit and accelerate the design of photonics systems across a broad range of optical technologies.

Candidate's Works

Journal Publications

J.1 **MacLellan B.***, Roztocki P.*, Belleville J., Romero Cortés L., Fischer B., Ruscitti K., Azaña J., Morandotti R., *Inverse design of photonic signal processing systems*, (in preparation), 2021.

J.2 Roztocki P.*, **MacLellan B.***, Islam M., Reimer C., Fischer B., Sciara S., Helsten R., Jestin Y., Cino A., Chu S.T., Little B.E., Moss D.J., Kues M., Morandotti R., *Arbitrary phase access for stable fiber interferometers*, Laser and Photonics Reviews (in press), 2021.

J.3 Zhang Y., Kues M., Roztocki P., Reimer C., Fischer B., **MacLellan B.**, Bisianov A., Peschel U., Little B.E., Chu S.T., Moss D.J., Caspani L., Moradotti R., *Induced photon correlations through the overlap of two four-wave mixing processes in integrated cavities*, Laser and Photonics Reviews, 14(9), (2020).

J.4 **MacLellan B.***, Roztocki P.*, Kues M., Reimer C., Romero Cortés L., Zhang Y., Sciara S., Wetzel B., Cino A., Chu S.T., Little B.E., Moss D.J., Caspani L., Azaña J., Morandotti R., *Generation and coherent control of pulsed quantum frequency combs*, Journal of Visualized Experiments, 136, (2018).

Conferences

C.1 **MacLellan B.**, Roztocki P., Islam M., Reimer C., Fischer B., Sciara S., Helsten R., Jestin Y., Cino A., Chu S.T., Little B.E., Moss D.J., Kues M., Morandotti R., "Fiber Interferometers for Time-Domain Quantum Photonics," *CLEO US*, May 2021.

C.2 **MacLellan B.**, Roztocki P., Islam M., Reimer C., Fischer B., Sciara S., Helsten R., Jestin Y., Cino A., Chu S.T., Little B.E., Moss D.J., Kues M., Morandotti R., "Arbitrary phase access for stable fiber interferometers," *Montreal Photonics Networking Event*, October 2020. **(winner of Industrial Choice Award)**

C.3 **MacLellan B.**, Roztocki P., Sciara S., Reimer C., Romero Cortés L., Cino A., Little B.E., Moss D.J., Caspani L., Munro W., Azaña J., Kues M., Morandotti R., "Scalability of high-dimensional quantum operations in the spectral domain," *Photonics North Conference*, May 2020.

C.4 Roztocki P., Zhang Y., Kues M., Reimer C., Fischer B., **MacLellan B.**, Bisianov A., Peschel U., Little B.E., Chu S.T., Moss D.J., Caspani L., Morandotti R., "Induced photon correlations by the superposition of two four-wave mixing processes on a photonic chip," *Photonics North Conference*, May 2020.

C.5 Sciara S., Roztocki P., Reimer C., **MacLellan B.**, Fischer B., Romero Cortés L., Moss D.J., Caspani L., Munro W.J., Azaña J., Kues M., Morandotti R., "On-chip quantum frequency combs for the generation of complex entangled photon states" *Photonics North Conference*, May 2020.

C.6 Roztocki P., **MacLellan B.**, Islam M., Reimer C., Fischer B., Sciara S., Helsten R., Jestin Y., Cino A., Chu S.T., Little B.E., Moss D.J., Kues M., Morandotti R., "Phase retrieval in fiber-based interferometers," *Advanced Photonics Congress*, June 2020.

C.7 Roztocki P., Kues M., Zhang Y., Kues M., Reimer C., Fischer B., **MacLellan B.**, Bisianov A., Peschel U., Little B.E., Chu S.T., Moss D.J., Caspani L., Moradotti R., "Induced photon correlations by the superposition of two four-wave mixing processes on a photonic chip", *Advanced Photonics Congress*, June 2020.

C.8 Zhang Y., Kues M., Roztocki P., Reimer C., Fischer B., **MacLellan B.**, Caspani L., Little B.E., Chu S.T., Moss D.J., Moradotti R., "Single-photon induced correlation with integrated quantum frequency combs", *European Quantum Electronics Conference*, June 2019.

Patents

P.1 **MacLellan B.**, Roztocki P., Van Howe J., Romero Cortés L., Fischer B., Jestin Y., Azaña J., Morandotti R., *Method for automated design of optical systems*, Provisional patent, 2020.

P.2 Fischer B., Chemnitz M., **MacLellan B.**, Roztocki P., Azaña J., Morandotti R., *System and method for arbitrary optical waveform generation*, Provisional patent, 2020. (in preparation)

P.3 Fischer B., Roztocki P., Kues M., Chemnitz M., Rimoldi C., **MacLellan B.**, Cortés L.R., Azaña J., Jestin Y., Morandotti R., *System and method for optical information processing with a reconfigurable nonlinear optical network*, Provisional patent, 2020.

References

1. Molesky, S. *et al.* Inverse design in nanophotonics. *Nature Photonics* **12,** 659–670. ISSN: 17494893. http://dx.doi.org/10.1038/s41566-018-0246-9%20http://web.stanford.edu/group/nqp/jv%7B%5C_%7Dfiles/papers/molesky2018inverse.pdf (2018).
2. Thöniss, T., Adams, G. & Gerhard, C. Optical System design. *Optik & Photonik* **4,** 30–33 (2009).
3. Höschel, K. & Lakshminarayanan, V. Genetic algorithms for lens design: a review. *Journal of Optics* **48,** 134–144. ISSN: 09746900. https://link.springer.com/content/pdf/10.1007/s12596-018-0497-3.pdf (2019).
4. Dory, C. *et al.* Inverse-designed diamond photonics. *Nature Communications* **10,** 1–8. ISSN: 20411723. arXiv: 1812.02287. https://arxiv.org/pdf/1812.02287.pdf (2019).
5. Krenn, M., Malik, M., Fickler, R., Lapkiewicz, R. & Zeilinger, A. Automated Search for new Quantum Experiments. *Physical Review Letters* **116,** 1–5. ISSN: 10797114. arXiv: 1509.02749 (2016).
6. Maram, R., Van Howe, J., Li, M. & Azaña, J. Noiseless intensity amplification of repetitive signals by coherent addition using the temporal Talbot effect. *Nature Communications* **5.** ISSN: 20411723 (2014).
7. Krenn, M., Kottmann, J., Tischler, N. & Aspuru-Guzik, A. Conceptual understanding through efficient inverse-design of quantum optical experiments, 1–14. arXiv: 2005.06443. http://arxiv.org/abs/2005.06443 (2020).
8. Plocher, J. & Panesar, A. Review on design and structural optimisation in additive manufacturing: Towards next-generation lightweight structures. *Materials and Design* **183.** ISSN: 18734197 (2019).
9. Dulikravich, G. S. Special Issue on Inverse Design and Optimization in Engineering: Introduction. *Applied Mechanics Reviews* **41,** 216–216. ISSN: 0003-6900. https://asmedigitalcollection.asme.org/appliedmechanicsreviews/article/41/6/216/395285/Special-Issue-on-Inverse-Design-and-Optimization (June 1988).
10. Rolvink, A., van de Straat, R. & Coenders, J. Parametric Structural Design and beyond. *International Journal of Architectural Computing* **8,** 319–336. ISSN: 1478-0771 (2010).
11. Harris, S. P. & Ifeachor, E. C. Automatic design of frequency sampling filters by hybrid genetic algorithm techniques. *IEEE Transactions on Signal Processing* **46,** 3304–3314. ISSN: 1053587X (1998).
12. Delon, G. L. A methodology for total hospital design. *Health Services Research* **5,** 210–23. ISSN: 0017-9124. http://www.ncbi.nlm.nih.gov/pubmed/5494263%20http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC1067497 (1970).

13. Nagy, D. & Villaggi, L. *Generative Design for Architectural Space Planning* 2020. https://www.autodesk.com/autodesk-university/article/Generative-Design-Architectural-Space-Planning-2020 (2020).
14. HamediRad, M. *et al.* Towards a fully automated algorithm driven platform for biosystems design. *Nature Communications* **10,** 1–10. ISSN: 20411723. http://dx.doi.org/10.1038/s41467-019-13189-z (2019).
15. Sanchez-Lengeling, B. & Aspuru-Guzik, A. Inverse molecular design using machine learning:Generative models for matter engineering. *Science* **361,** 360–365. ISSN: 10959203 (2018).
16. Ruder, S. An overview of gradient descent optimization algorithms, 1–14. arXiv: 1609.04747. http://arxiv.org/abs/1609.04747 (Sept. 2016).
17. Chen, X. & Yamamoto, K. An experiment in genetic optimization in lens design. *Journal of Modern Optics* **44,** 1693–1702. ISSN: 13623044 (1997).
18. Betensky, E. I. Postmodern lens design. *Optical Engineering* **32,** 1750. ISSN: 00913286 (1993).
19. Yabe, A. & Yabe, A. *Optimization in Lens Design* ISBN: 9781510619838 (2018).
20. Volatier, J.-B., Menduiña-Fernández, Á. & Erhard, M. Generalization of differential ray tracing by automatic differentiation of computational graphs. *Journal of the Optical Society of America A* **34,** 1146. ISSN: 1084-7529 (2017).
21. Jafar-Zanjani, S., Inampudi, S. & Mosallaei, H. Adaptive Genetic Algorithm for Optical Metasurfaces Design. *Scientific Reports* **8,** 1–16. ISSN: 20452322 (2018).
22. Spühler, M. M. *et al.* A very short planar silica spot-size converter using a nonperiodic segmented waveguide. *Journal of Lightwave Technology* **16,** 1680–1685. ISSN: 07338724 (1998).
23. Su, L. *et al.* Nanophotonic inverse design with SPINS: Software architecture and practical considerations. *Applied Physics Reviews* **7.** ISSN: 19319401. arXiv: 1910.04829 (2020).
24. Cox, S. J. . & Dobson, D. C. . Maximizing Band Gaps in Two-Dimensional Photonic Crystals. *SIAM Journal on Applied Mathematics* **59,** 2108–2120 (1999).
25. Hughes, T. W., Williamson, I. A. D., Minkov, M. & Fan, S. Forward-Mode Differentiation of Maxwell's Equations. arXiv: 1908.10507. http://arxiv.org/abs/1908.10507 (2019).
26. Krenn, M., Gu, X. & Zeilinger, A. Quantum Experiments and Graphs: Multiparty States as Coherent Superpositions of Perfect Matchings. *Physical Review Letters* **119,** 1–6. ISSN: 10797114. arXiv: 1705.06646 (2017).
27. Tsai, C.-M. & Fang, Y.-C. A study of optical design and optimization of laser optics. *Optical Modeling and Performance Predictions VI* **8840,** 88400O. ISSN: 0277786X (2013).
28. Coelho, L. D., Gaete, O. & Hanik, N. An algorithm for global optimization of optical communication systems. *AEU - International Journal of Electronics and Communications* **63,** 541–550. ISSN: 14348411. https://linkinghub.elsevier.com/retrieve/pii/S1434841109001186 (July 2009).
29. Harish, A. V. & Nilsson, J. Optimization of phase modulation with arbitrary waveform generators for optical spectral control and suppression of stimulated Brillouin scattering. *Optics Express* **23,** 6988. ISSN: 1094-4087 (2015).
30. Wang, J., Sciarrino, F., Laing, A. & Thompson, M. G. Integrated photonic quantum technologies. *Nature Photonics* **14,** 273–284. ISSN: 1749-4885. http://www.nature.com/articles/s41566-019-0532-1 (May 2020).

31. Stanley, K. O. & Miikkulainen, R. Evolving neural networks through augmenting topologies. *Evolutionary Computation* **10,** 99–127. http://mitpress.mit.edu/e-mail (2002).
32. Dauphin, Y. N. *et al.* Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. *Advances in Neural Information Processing Systems* **4,** 2933–2941. ISSN: 10495258. arXiv: 1406.2572. https://arxiv.org/pdf/1406.2572.pdf (2014).
33. Saleh, B. E. A. & Teich, M. C. *Fundamentals of Photonics , 2nd Edition* 2007.
34. Cook, R. D. Principal Hessian Directions Revisited. *Journal of the American Statistical Association* **93,** 84. ISSN: 01621459. https://www.jstor.org/stable/2669605?origin=crossref (Mar. 1998).
35. Li, K.-C. On Principal Hessian Directions for Data Visualization and Dimension Reduction: Another Application of Stein's Lemma. *Journal of the American Statistical Association* **87,** 1025. ISSN: 01621459. https://www.jstor.org/stable/2290640?origin=crossref (Dec. 1992).
36. Larocca, M., Calzetta, E. A. & Wisniacki, D. A. Exploiting Landscape Geometry to Enhance Quantum Optimal Control. arXiv: 1911.07105. http://arxiv.org/abs/1911.07105%20http://dx.doi.org/10.1103/PhysRevA.101.023410 (Nov. 2019).
37. Islam, M. Raman amplifiers for telecommunications. *IEEE Journal of Selected Topics in Quantum Electronics* **8,** 548–559. ISSN: 1077-260X. http://ieeexplore.ieee.org/document/1016358/ (May 2002).
38. Simon, D. *Evolutionary Optimization Algorithms* 1st. ISBN: 978-0-470-93741-9 (Wiley, New Jersey, 2013).
39. Kingma, D. P. & Ba, J. Adam: A Method for Stochastic Optimization. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings,* 1–15. arXiv: 1412.6980. http://arxiv.org/abs/1412.6980 (Dec. 2014).
40. Nocedal, J. & Wright, S. *Numerical Optimization* ISBN: 978-0-387-30303-1. http://link.springer.com/10.1007/978-0-387-40065-5 (Springer New York, 2006).
41. Virtanen, P. *et al.* SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods* (2020).
42. Ab Wahab, M. N., Nefti-Meziani, S. & Atyabi, A. A comprehensive review of swarm optimization algorithms. *PLoS ONE* **10,** 1–36. ISSN: 19326203. https://journals.plos.org/plosone/article/file?id=10.1371/journal.pone.0122827%7B%5C&%7Dtype=printable (2015).
43. Miranda, L. J. V., Moser, A. & Cronin, S. K. *PySwarms: A Research Toolkit for Particle Swarm Optimization in Python* 2017. https://pyswarms.readthedocs.io/en/latest/.
44. Grosan, C. & Abraham, A. in *Hybrid Evolutionary Algorithms* 1–17 (2007). http://link.springer.com/10.1007/978-3-540-73297-6%7B%5C_%7D1.
45. Fortin, F.-A., De Rainville, F.-M., Gardner, M.-A. G., Parizeau, M. & Gagné, C. DEAP: Evolutionary Algorithms Made Easy. *J. Mach. Learn. Res.* **13,** 2171–2175. ISSN: 1532-4435 (July 2012).
46. Gao, X., Xiao, B., Tao, D. & Li, X. A survey of graph edit distance. *Pattern Analysis and Applications* **13,** 113–129. ISSN: 14337541 (2010).
47. Hagberg, A. A., Schult, D. A. & Swart, P. J. *Exploring Network Structure, Dynamics, and Function using NetworkX* in *Proceedings of the 7th Python in Science Conference* (eds Varoquaux, G., Vaught, T. & Millman, J.) (Pasadena, CA USA, 2008), 11–15.

48. Yiannopoulos, K. *et al.* Pulse repetition frequency multiplication with spectral selection in Fabry-Perot filters. *IEEE Journal of Quantum Electronics* **40,** 157–165. ISSN: 00189197 (2004).

49. Maram, R., Romero Cortés, L., Van Howe, J. & Azaña, J. Energy-Preserving Arbitrary Repetition-Rate Control of Periodic Pulse Trains Using Temporal Talbot Effects. *Journal of Lightwave Technology* **35,** 658–668. ISSN: 07338724 (2017).

50. Haboucha, A. *et al.* Optical-fiber pulse rate multiplier for ultralow phase-noise signal generation. *Optics Letters* **36,** 3654. ISSN: 0146-9592. `https://www.osapublishing.org/abstract.cfm?URI=ol-36-18-3654` (Sept. 2011).

51. Caraquitena, J., Jiang, Z., Leaird, D. E. & Weiner, A. M. Simultaneous repetition-rate multiplication and envelope control based on periodic phase-only and phase-mostly line-by-line pulse shaping. *Journal of the Optical Society of America B* **24,** 3034. ISSN: 0740-3224 (2007).

52. Pudo, D. & Chen, L. R. Tunable passive all-optical pulse repetition rate multiplier using fiber Bragg gratings. *Journal of Lightwave Technology* **23,** 1729–1733. ISSN: 07338724 (2005).

53. Dunn, A. M., Hofmann, O. S., Waters, B. & Witchel, E. *Cloaking malware with the trusted platform module* 2011.

54. Yao, J. Photonic generation of microwave arbitrary waveforms. *Optics Communications* **284,** 3723–3736. ISSN: 00304018. `http://dx.doi.org/10.1016/j.optcom.2011.02.069` (2011).

55. Bazargani, H. P. & Azaña, J. Optical pulse shaping based on discrete space-totime mapping in cascaded co-directional couplers. *Optics Express* **23,** 23450–23461. ISSN: 10944087 (2015).

56. Chen, L. R. Photonic generation of chirped microwave and millimeter wave pulses based on optical spectral shaping and wavelength-to-time mapping in silicon photonics. *Optics Communications* **373,** 70–81. ISSN: 00304018. `http://dx.doi.org/10.1016/j.optcom.2015.04.023` (2016).

57. Rashidinejad, A. & Weiner, A. M. Photonic radio-frequency arbitrary waveform generation with maximal time-bandwidth product capability. *Journal of Lightwave Technology* **32,** 3383–3393. ISSN: 07338724 (2014).

58. Bui, L. A. A Review of Photonic Generation of Arbitrary Microwave Waveforms. *Progress In Electromagnetics Research B* **75,** 1–12 (2017).

59. Michelson, A. A. & Morley, E. W. On the relative motion of the Earth and the luminiferous ether. *American Journal of Science* **s3-34,** 333–345. ISSN: 0002-9599. `http://www.ajsonline.org/cgi/doi/10.2475/ajs.s3-34.203.333` (Nov. 1887).

60. Hong, C. K., Ou, Z. Y. & Mandel, L. Measurement of subpicosecond time intervals between two photons by interference. *Physical Review Letters.* ISSN: 00319007 (1987).

61. Brendel, J., Gisin, N., Tittel, W. & Zbinden, H. Pulsed energy-time entangled twin-photon source for quantum communication. *Physical Review Letters* **82,** 2594. ISSN: 10797114 (1999).

62. Drexler, W. *et al.* Ultrahigh-resolution ophthalmic optical coherence tomography. *Nature Medicine* **7,** 502–506. ISSN: 10788956 (2001).

63. Dudley, J. M., Genty, G. & Coen, S. Supercontinuum generation in photonic crystal fiber. *Reviews of Modern Physics* **78,** 1135–1184. ISSN: 0034-6861. `https://link.aps.org/doi/10.1103/RevModPhys.78.1135` (Oct. 2006).

64. Agha, I. H., Okawachi, Y. & Gaeta, A. L. Theoretical and experimental investigation of broadband cascaded four-wave mixing in high-Q microspheres. *Optics Express* **17,** 16209. ISSN: 1094-4087 (2009).

65. Zhang, L. *et al.* Generation of two-cycle pulses and octave-spanning frequency combs in a dispersion-flattened micro-resonator. *Optics Letters* **38,** 5122. ISSN: 0146-9592 (2013).
66. Lalau-Keraly, C. M. *LumOpt - Continuous adjoint optimization wrapper for Lumerical* 2016. `https://github.com/chriskeraly/lumopt`.
67. Kleijnen, J. P. *Sensitivity analysis and related analyses: A review of some statistical techniques* **1-4,** 111–142. ISBN: 0094965970881 (1997).
68. Saltelli, A. *et al.* Variance based sensitivity analysis of model output. Design and estimator for the total sensitivity index. *Computer Physics Communications* **181,** 259–270. ISSN: 00104655 (2010).
69. Pressley, A. *Elementary Differential Geometry* 2nd ed. (Springer, 2001).
70. Güne¸, A., Baydin, G., Pearlmutter, B. A. & Siskind, J. M. Automatic Differentiation in Machine Learning: a Survey. *Journal of Machine Learning Research* **18,** 1–43. `https://jmlr.csail.mit.edu/papers/volume18/17-468/17-468.pdf` (2018).
71. Wengert, R. E. A simple automatic derivative evaluation program. *Communications of the ACM* **7,** 463–464. ISSN: 15577317. `https://www.cs.princeton.edu/courses/archive/fall19/cos597C/files/wengert1964.pdf` (1964).
72. Griewank, A. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation* Frontiers (SIAM, Philadelphia, PA, 2000).
73. Bücker, M. & Hovland, P. *Automatic Differentiation* 2020. `http://www.autodiff.org/`.
74. Bradbury, J. *et al. JAX: Composable transformation of Python and Numpy programs* 2018. `http://github.com/google/jax`.
75. Maclaurin, D., Duvenaud, D., Johnson, M. & Townsend, J. *Autograd* 2016. `https://github.com/HIPS/autograd`.
76. Bertsekas, D. P. *Constrained Optimization and Lagrange Multiplier Methods* ISBN: 9780120934805. `https://linkinghub.elsevier.com/retrieve/pii/C20130103662` (Elsevier, 1982).
77. Fogel, D. B. *Evolutionary Computation* ISBN: 9780470544600. `http://ieeexplore.ieee.org/xpl/bkabstractplus.jsp?bkn=5263042` (IEEE, 1998).
78. Sudholt, D. in *Springer Handbook of Computational Intelligence* 929–959 (Springer Berlin Heidelberg, Berlin, Heidelberg, 2015). `http://link.springer.com/10.1007/978-3-662-43505-2%7B%5C_%7D46`.
79. Deb, K., Pratap, A., Agarwal, S. & Meyarivan, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation* **6,** 182–197. ISSN: 1089778X (2002).
80. Gunantara, N. & Ai, Q. A review of multi-objective optimization: Methods and its applications. *Cogent Engineering* **5,** 1502242. ISSN: 2331-1916. `https://www.tandfonline.com/doi/full/10.1080/23311916.2018.1502242` (Jan. 2018).
81. Duchi, J., Hazan, E. & Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *COLT 2010 - The 23rd Conference on Learning Theory,* 257–269. `https://stanford.edu/%7B~%7Djduchi/projects/DuchiHaSi10%7B%5C_%7Dcolt.pdf` (2010).
82. Andrychowicz, M. *et al.* Learning to learn by gradient descent by gradient descent. *Advances in Neural Information Processing Systems,* 3988–3996. ISSN: 10495258. arXiv: `1606.04474` (2016).

83. Le, Q. V. *et al. On Optimization Methods for Deep Learning* in *Proceedings of the 28th International Conference on International Conference on Machine Learning* (Omnipress, Madison, WI, USA, 2011), 265–272. ISBN: 9781450306195.

84. Yang, T., Jin, G.-F. & Zhu, J. Automated design of freeform imaging systems. *Light: Science & Applications* **6,** e17081–e17081. ISSN: 2047-7538. http://www.nature.com/articles/lsa201781 (Oct. 2017).

85. Brixner, B. Lens design and local minima. *Applied Optics* **20,** 384. ISSN: 0003-6935 (1981).

86. Shafer, D. Global optimization in optical design. *Computers in physics* **8,** 188–195. ISSN: 08941866 (1994).

87. Ono, I., Kobayashi, S. & Yoshida, K. Global and multi-objective optimization for lens design by real-coded genetic algorithms. *International Optical Design Conference 1998* **3482,** 110. ISSN: 1996756X (1998).

88. Shi, R. & Kross, J. *Differential ray tracing for optical design* in *Conference Proceeds of SPIE - 3737* (Aug. 1999), 149. http://proceedings.spiedigitallibrary.org/proceeding.aspx?doi=10.1117/12.360003.

89. Dain Lin, P. Determination of first-order derivative matrix of wavefront aberration with respect to system variables. *Applied Optics* **51,** 486. ISSN: 1559-128X. https://www.osapublishing.org/abstract.cfm?URI=ao-51-4-486 (Feb. 2012).

90. Yu, Z. & Sun, X. Inverse-Designed Photonic Jumpers with Ultracompact Size and Ultralow Loss. *Journal of Lightwave Technology* **8724,** 1–1. ISSN: 0733-8724. https://ieeexplore.ieee.org/document/9153889/ (2020).

91. Lin, Z., Lončar, M. & Rodriguez, A. W. Topology optimization of multi-track ring resonators and 2D microcavities for nonlinear frequency conversion. *Optics Letters* **42,** 2818. ISSN: 0146-9592. arXiv: 1701.05628. https://www.osapublishing.org/abstract.cfm?URI=ol-42-14-2818 (July 2017).

92. Frei, W. R., Tortorelli, D. A. & Johnson, H. T. Geometry projection method for optimizing photonic nanostructures. *Optics Letters* **32,** 77. ISSN: 0146-9592 (2007).

93. Deng, Y. & Korvink, J. G. Topology optimization for three-dimensional electromagnetic waves using an edge element-based finite-element method. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* **472.** ISSN: 14712946 (2016).

94. Lalau-Keraly, C. M., Bhargava, S., Miller, O. D. & Yablonovitch, E. Adjoint shape optimization applied to electromagnetic design. *Optics Express* **21,** 21693. ISSN: 1094-4087 (2013).

95. Jarny, Y., Ozisik, M. N. & Bardon, J. P. A general optimization method using adjoint equation for solving multidimensional inverse heat conduction. *International Journal of Heat and Mass Transfer* **34,** 2911–2919. ISSN: 00179310 (1991).

96. Jameson, A. & Jameson, A. Aerodynamic Shape Optimization Using the Adjoint Method. *Vki Lecture Series on Aerodynamic Drag Prediction and Reduction, Von Karman Institute of Fluid Dynamics, Rhode St Genese,* 3–7. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.13.6065 (2003).

97. Hughes, T. W., Minkov, M., Williamson, I. A. & Fan, S. Adjoint Method and Inverse Design for Nonlinear Nanophotonic Devices. *ACS Photonics* **5,** 4781–4787. ISSN: 23304022. arXiv: 1811.01255 (2018).

98. Hughes, T. W., Williamson, I. A. D., Minkov, M. & Fan, S. *Ceviche* 2019. https://pypi.org/project/ceviche/.

99. Melati, D. *et al.* Mapping the global design space of nanophotonic components using machine learning pattern recognition. *Nature Communications* **10,** 1–9. ISSN: 20411723. arXiv: 1811.01048. http://dx.doi.org/10.1038/s41467-019-12698-1 (2019).

100. Frei, W. *Computing Design Sensitivities in COMSOL Multiphysics* 2015. https://www.comsol.com/blogs/computing-design-sensitivities-in-comsol-multiphysics/.

101. Melnikov, A. A. *et al.* Active learning machine learns to create new quantum experiments. *Proceedings of the National Academy of Sciences of the United States of America* **115,** 1221–1226. ISSN: 10916490. arXiv: 1706.00868 (2018).

102. Erhard, M., Malik, M., Krenn, M. & Zeilinger, A. Experimental Greenberger–Horne–Zeilinger entanglement beyond qubits. *Nature Photonics* **12,** 759–764. ISSN: 17494893. http://dx.doi.org/10.1038/s41566-018-0257-6 (2018).

103. Gu, X., Erhard, M., Zeilinger, A. & Krenn, M. Quantum experiments and graphs II: Quantum interference, computation, and state generation. *Proceedings of the National Academy of Sciences of the United States of America* **116,** 4147–4155. ISSN: 10916490. arXiv: 1803.10736 (2019).

104. Isenor, G. A Global Optimization Approach to Laser Design. *Optimization and Engineering* **4,** 177–196. ISSN: 1389-4420 (2003).

105. Arteaga-Sierra, F. R. *et al.* Supercontinuum optimization for dual-soliton based light sources using genetic algorithms in a grid platform. *Optics Express* **22,** 23686. ISSN: 1094-4087 (2014).

106. Yang, L., Lavrinenko, A. V., Hvam, J. M. & Sigmund, O. Design of one-dimensional optical pulse-shaping filters by time-domain topology optimization. *Applied Physics Letters* **95,** 2007–2010. ISSN: 00036951 (2009).

107. Scholtz, L., Ladanyi, L. & Mullerova, J. Optimization of all-optical signal processing via nonlinear fiber Bragg gratings. *International Conference on Transparent Optical Networks* **2016-Augus,** 2–5. ISSN: 21627339 (2016).

108. Lukens, J. M. *et al.* All-Optical Frequency Processor for Networking Applications. *Journal of Lightwave Technology* **38,** 1678–1687. ISSN: 15582213. arXiv: 1904.08511 (2020).

109. *LUMERICAL INTERCONNECT Transient Sample/Block Mode Simulator* https://support.lumerical.com/hc/en-us/articles/360034399174-INTERCONNECT-Transient-Sample-Block-Mode-TSM-TBM-Simulator (2020).

110. Jin, W., Barzilay, R. & Jaakkola, T. *Junction Tree Variational Autoencoder for Molecular Graph Generation* in *Proceedings of the 35th International Conference on Machine Learning* (eds Dy, J. & Krause, A.) **80** (PMLR, Stockholmsmässan, Stockholm Sweden, 2018), 2323–2332. http://proceedings.mlr.press/v80/jin18a.html.

111. Jin, W., Yang, K., Barzilay, R. & Jaakkola, T. Learning Multimodal Graph-to-Graph Translation for Molecular Optimization. arXiv: 1812.01070. http://arxiv.org/abs/1812.01070 (Dec. 2018).

112. Kerr, C. C., Dura-Bernal, S., Smolinski, T. G., Chadderdon, G. L. & Wilson, D. P. Optimization by Adaptive Stochastic Descent. *PLOS ONE* **13** (ed Kaderali, L.) e0192944. ISSN: 1932-6203. https://dx.plos.org/10.1371/journal.pone.0192944 (Mar. 2018).

113. Azad, S. K. & Hasançebi, O. An elitist self-adaptive step-size search for structural design optimization. *Applied Soft Computing* **19,** 226–235. ISSN: 15684946. https://linkinghub.elsevier.com/retrieve/pii/S1568494614000878 (June 2014).

114. dos Santos Mignon, A. & de Azevedo da Rocha, R. L. An Adaptive Implementation of ϵ-Greedy in Reinforcement Learning. *Procedia Computer Science* **109,** 1146–1151. ISSN: 18770509. https://linkinghub.elsevier.com/retrieve/pii/S1877050917311134 (2017).

115. Carvalho, D. B., Bittencourt, J. C. N. & Maia, T. D. M. The Simple Genetic Algorithm Performance: A Comparative Study on the Operators Combination. *Proceedings of INFOCOMP 2011,* 20–24. https://pdfs.semanticscholar.org/b74b/ee32cf09fe83f2cb16c273363327c19bbc25.pdf (2011).

116. Moritz, P. *et al. Ray: A Distributed Framework for Emerging AI Applications* Dec. 2017. arXiv: 1712.05889. http://arxiv.org/abs/1712.05889.

117. *Google Cloud Documentation* 2020. https://cloud.google.com/docs.

118. Romero Cortés, L., Maram, R., Guillet de Chatellus, H. & Azaña, J. Arbitrary Energy-Preserving Control of Optical Pulse Trains and Frequency Combs through Generalized Talbot Effects. *Laser & Photonics Reviews* **13,** 1900176. ISSN: 1863-8880. https://onlinelibrary.wiley.com/doi/abs/10.1002/lpor.201900176 (Dec. 2019).

119. Balzer, J. *et al.* Modelocked semiconductor laser system with pulse picking for variable repetition rate. *Electronics Letters* **47,** 1387–1388 (2011).

120. Romero Cortés, L. *Generalized Talbot Effect: Theory and application to advanced optical wave processing* PhD thesis (L'Institut National de la Recherche Scientifique, 2018), 1–275. ISBN: 0819406732.

121. Bradbury, J. *et al. {JAX}: composable transformations of {P}ython+{N}um{P}y programs* 2018. http://github.com/google/jax.

122. Rosenbrock, H. H. An Automatic Method for Finding the Greatest or Least Value of a Function. *The Computer Journal* **3,** 175–184. ISSN: 0010-4620. https://academic.oup.com/comjnl/article-lookup/doi/10.1093/comjnl/3.3.175 (Mar. 1960).

123. Kok, S. & Sandrock, C. Locating and characterizing the stationary points of the extended Rosenbrock function. *Evolutionary Computation* **17,** 437–453. ISSN: 10636560 (2009).

124. Harris, C. R. *et al.* Array programming with {NumPy}. *Nature* **585,** 357–362. https://doi.org/10.1038/s41586-020-2649-2 (Sept. 2020).

125. Bokeh Development Team. *Bokeh: Python library for interactive visualization* (2020). https://bokeh.org/.

126. Hansen, N., Akimoto, Y. & Baudis, P. *CMA-ES Python Package: pycma* Zenodo. Feb. 2019. https://doi.org/10.5281/zenodo.2559634.

APPENDIX A

Appendix

A.1 Automatic differentiation

Automatic differentiation is a set of techniques for computing the derivative(s) of a function written as a computer program. It has many advantages over other differentiation techniques, such as numeric, analytic, or symbolic differentiation, as it is efficient in the number of input variables, reduces programming overhead for complex functions, and computes the derivatives to machine precision [70, 72]. To illustrate automatic differentiation, the basic concept is introduced using a function, $y = F(x) : \mathbb{R} \to \mathbb{R}$, which takes one independent variable and returns one dependent variable. To demonstrate the basics of automatic differentiation, take the function $F(x)$ to be a composite function composed of three elementary operations, $f_i,\ i = 1, 2, 3$. Here, $\circ$ represents function composition (the output of one function in the input to the next). We write $F(x)$ as,

$$F(x) = f_3(f_2(f_1(x))) = f_3 \circ f_2 \circ f_1(x) \tag{A.1}$$

To break up the function, we introduce intermediate variables, w_i to store the outputs of each individual composing function, f_i, as:

$$w_1 = f_1(w_0), \qquad w_2 = f_2(w_1), \qquad w_3 = f_3(w_2) \tag{A.2}$$

and these intermediate variables are related to the input parameters and output value as:

$$x = w_0, \qquad F(x) = w_3 \tag{A.3}$$

With this setup, $F(x)$ may be written by expanding out the w_i variables as,

$$F(x) = w_3 = f_3(w_2) = f_3(f_2(w_1)) = f_3(f_2(f_1(w_0))) = f_3(f_2(f_1(x))) \tag{A.4}$$

It is helpful to visualize this function as a computational graph as in Fig. 2.2a. To now calculate the derivative of $F(x)$ with respect to x, it can be written in terms of the the intermediary variables

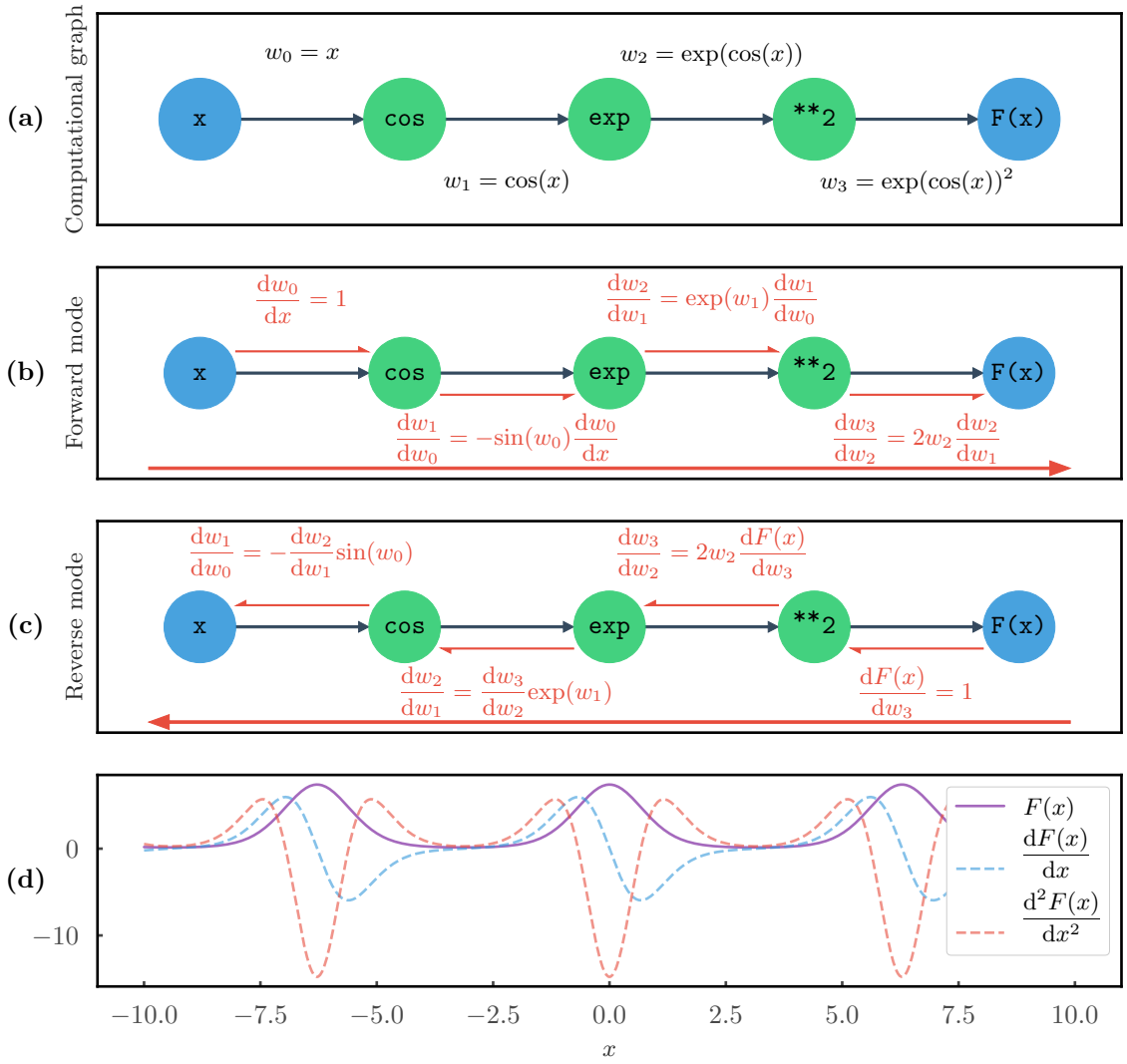


Figure A.1 – Automatic differentiation of an example function $y = F(x) = \exp(\cos(x))^2$. **(a)** Computational graph of the function, where the blue circles are the input and output scalar values and green circle are the elementary operations which compose the function. **(b)** Forward mode automatic differentiation, with derivatives accumulated moving from left-to-right along the computational graph. **(c)** Reverse mode automatic differentiation, with derivatives accumulated moving from right-to-left along the computational graph. **(d)** Plot of $F(x)$, along with its first- and second- order derivatives calculated using automatic differentiation. Continued, higher-order derivatives are trivial to compute by the successive application of automatic differentiation. Code for these plots can be found here.

w_i and their derivatives as,

$$\frac{\mathrm{d}F(x)}{\mathrm{d}x} = \frac{\mathrm{d}F(x)}{\mathrm{d}w_3} \times \frac{\mathrm{d}w_3}{\mathrm{d}w_2} \times \frac{\mathrm{d}w_2}{\mathrm{d}w_1} \times \frac{\mathrm{d}w_1}{\mathrm{d}w_0} \times \frac{\mathrm{d}w_0}{\mathrm{d}x} \tag{A.5}$$

This example can easily be generalized to more complicated functions $F(x) = f_n \circ \cdots \circ f_1(x)$, where $w_i = f_i(w_{i-1})$. It also generalizes to multivariate functions with the scalar derivatives replaced by Jacobian matrices and scalar multiplication by matrix multiplication [72].

Automatic differentiation has two modes of operation: forward mode and reverse mode. These refer to the order that the intermediary variables w_i and their derivatives are calculated and accumulated. Forward mode fixes the dependent variable(s), $\mathbf{x}$, and traverses the graph in the forward direction computing the derivative with $\frac{\mathrm{d}w_i}{\mathrm{d}x} = \frac{\mathrm{d}w_i}{\mathrm{d}w_{i-1}}\frac{\mathrm{d}w_{i-1}}{\mathrm{d}x}$. Reverse mode fixes the dependent variable(s), $F(\mathbf{x})$, and traverses the computational graph backwards: $\frac{\mathrm{d}y}{\mathrm{d}w_i} = \frac{\mathrm{d}y}{\mathrm{d}w_{i+1}}\frac{\mathrm{d}w_{i+1}}{\mathrm{d}w_i}$. These opposite modes are depicted in Fig. 2.2 with the example function $F(x) = \exp(\cos(x))^2$. In this example, both methods provide the same derivative evaluation of $\frac{\mathrm{d}F(x)}{\mathrm{d}x} = -2\sin(x)\exp(2\cos(x))$. While both methods evaluate the same derivative function, they differ in the memory and computation time required: for a function $\mathbf{y} = F(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}^m$, if $n \ll m$ (i.e. less input variables than output) then forward mode differentiation is preferred, while for $n \gg m$ (i.e. more input variables than output) the reverse mode differentiation is preferred. As optimization problems have $n \gg m$, reverse mode differentiation is generally used in this setting (using reverse mode differentiation to calculate the derivative of a cost function and minimizing). Because the derivatives are themselves composed of elementary functions, higher-order derivatives can be continually calculated with practically no overhead in programming.

Future work for photonic system automatic differentiation

The main benefit of using `autograd` in `ASOPE` compared to popular tools in Python, e.g. TensorFlow and PyTorch, is that it wraps around the standard scientific/numeric computing packages of Python, i.e., `numpy` and `scipy`. As such, models can be rapidly prototyped and developed using simple programming syntax, and yet still benefit from the efficient numerical calculations of `numpy`. One can, in some instances, even take optical models and simulations that had not originally been intended to be differentiable, and with relatively modest effort use `autograd` to make them differentiable. Yet, for more computationally-demanding simulations and optimization, it may be beneficial to scale to using GPU resources – something that is not supported by `autograd` but is supported by, e.g., TensorFlow and PyTorch. A related project, JAX, has extended the capabilities of `autograd` to support GPU and TPU calculations, along with other useful tools such as composable function transformations [121]. Other potential benefits of employing JAX in the future is the ability to accelerate CPU computations using the automatic vectorization – for which initial tests indicated a $\sim 3\times$ improvement in simulation time. This could provide a significant speed-up while also still using multi-CPU parallel computing.

A.2 Hessian analysis

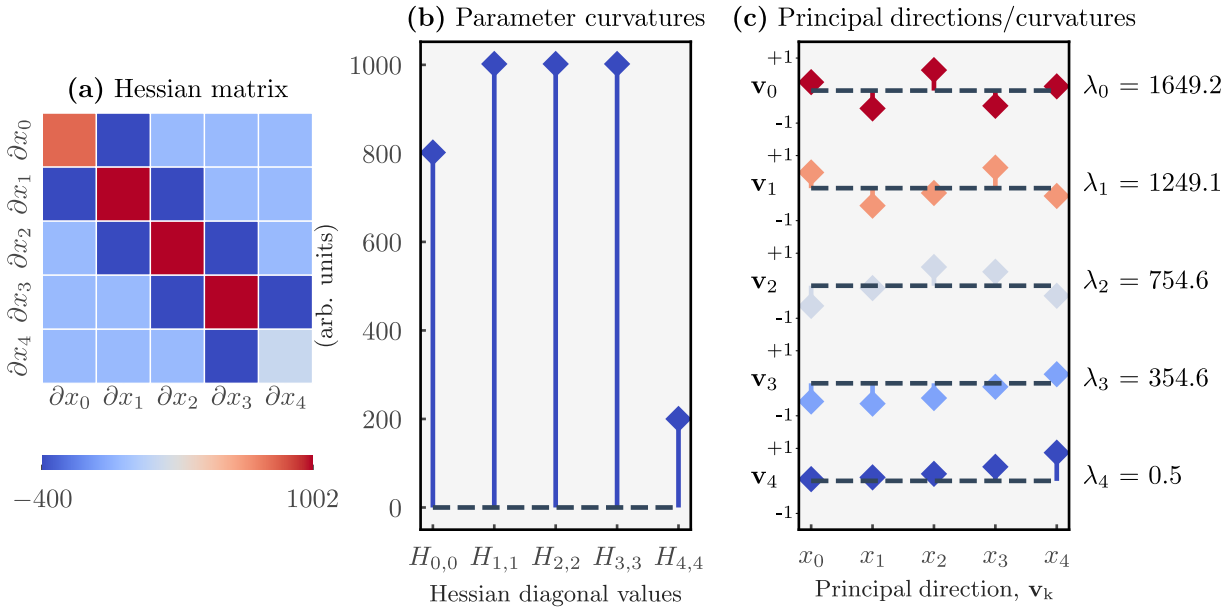


Figure A.2 – Hessian analysis of the 5-dimensional Rosenbrock function: **(a)** Hessian matrix, **(b)** Parameter curvatures, **(c)** Principal directions, $\mathbf{v}_i$, sorted by the associated principal curvatures, λ_i. Code can be found here.

The Hessian matrix $\mathbf{H}(\mathbf{x}^*)$ is the second-order derivative of a function $F : \mathbb{R}^n \to \mathbb{R}$, where each element $H_{i,j} = \partial^2 F / \partial x_i \partial x_j$ is the second-order partial derivative on parameters x_i and x_j, evaluated at $\mathbf{x}^*$. It is square $(n \times n)$, necessarily real (as we constrain $F(\mathbf{x})$ and $\mathbf{x}$ to be real), and symmetric (as the order in which partial derivatives are taken does not matter). The diagonal elements of $\mathbf{H}(\mathbf{x}^*)$, or $H_{i,i}$, are the *parameter curvatures* and express the sensitivity of $F(\mathbf{x}^*)$ with respect to x_i only. The off-diagonal elements $H_{i,j}$, $i \neq j$, however, indicate interaction terms between the parameters x_i and x_j. The eigenvector, $\mathbf{v}_i$, and eigenvalues, λ_i, of $\mathbf{H}$ are the *principal directions* and *principal curvatures*, respectively and satisfy,

$$\mathbf{H}(\mathbf{x}^*)\mathbf{v}_i = \lambda_i \mathbf{v}_i \tag{A.6}$$

This diagonalization of $\mathbf{H}(\mathbf{x}^*)$ can be seen as a linear basis transformation such that there are no mixed second-order partial derivatives. Checking the signs of the principal curvatures can indicate if the function at $\mathbf{x}^*$ is a minimum, maximum, or saddle-point. A positive eigenvalue indicates that, in the direction of its eigenvector, the function is concave-up, a negative eigenvalue indicates it is concave-down. Thus, if all eigenvalues are positive, the function at $\mathbf{x}^*$ is indeed a minimum; conversely, if all eigenvalues are negative, then $\mathbf{x}^*$ is a maximum; if there is a mix of positive and negative eigenvalues, $\mathbf{x}^*$ is a saddle point.

The Rosenbrock function is a n-dimensional function commonly used for testing optimization routines [122, 123]; it is defined as,

$$f(\mathbf{x}) = \sum_{i=1}^{n-1} 100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2 \tag{A.7}$$

and has global minimum of $(1, 1, \ldots, 1)$ for any dimension $n \geq 2$. The n-dimensional Rosenbrock function has one minimum point at,

$$\mathbf{x}_{\text{min}} = \underbrace{[1, \ldots, 1]}_{n \text{ times}} \tag{A.8}$$

The Matyas function is 2-dimensional test function defined as,

$$f(x_1, x_2) = 0.26(x_1^2 + x_2^2) - 0.48 x_1 x_2 \tag{A.9}$$

The function landscape and Hessian analysis of these two test functions are presented in Figs. 3.3 and 3.4. In Fig. A.2, all of the principal curvatures are positive, $\lambda_i \geq 0$, thus $F(\mathbf{x}_{\text{min}})$ is a local minima point, and not a maxima or saddle point. One can also infer that in the direction of $\mathbf{v}_4$, the function is highly stable.

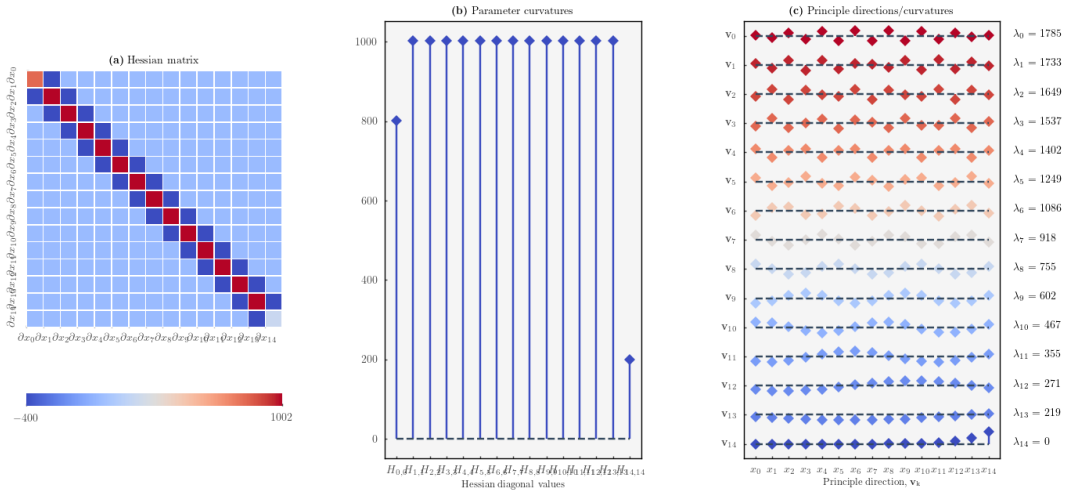


Figure A.3 – Hessian analysis of the 15-dimensional Rosenbrock function: **(a)** Hessian matrix, **(b)** Parameter curvatures, **(c)** Principal directions, $\mathbf{v}_i$, sorted by the associated principal curvatures, λ_i. Code can be found here.

In Chapter 3, one definition for a scaled Hessian matrix is (Eqn. 3.10),

$$H_{i,j}^{(\text{scaled})} = (\sigma_i \sigma_j) \frac{\partial F}{\partial x_i \partial x_j} \tag{A.10}$$

which gives each element of $\mathbf{H}^{(\text{scaled})}$ the same units as F. A secondary definition of a scaled Hessian is

$$H_{i,j}^{(\text{scaled})} = (x_i x_j)\frac{\partial F}{\partial x_i \partial x_j} \tag{A.11}$$

which removes requirement of having a second, user-defined value of σ_i – which may not be known.

Adaptive, Hessian-guided topology optimization

The method for using adaptive mutation probabilities based on sensitivity analysis aims to reduce the complexity of suggested optical systems by removing unnecessary optical elements. First, the probabilities of selecting a mutation operator, M_i, change at each iteration. The is conceptually similar to a greedy-epsilon [114] scheme in reinforcement learning or an adaptive step size in various other optimization algorithms [112, 113]. The update to these probabilities is:

$$\Pr(M_{\text{AddEdge},\,0}) = 0.25 \qquad \Pr(M_{\text{AddEdge},\,k}) = \Pr(M_{\text{AddEdge},\,0}) - \frac{0.25k}{N_{\text{gen}}} \tag{A.12}$$

$$\Pr(M_{\text{AddNodeEdge},\,0}) = 0.25 \qquad \Pr(M_{\text{AddNodeEdge},\,k}) = \Pr(M_{\text{AddNodeEdge},\,0}) - \frac{0.25k}{N_{\text{gen}}} \tag{A.13}$$

$$\Pr(M_{\text{SwapModel},\,0}) = 0.50 \qquad \Pr(M_{\text{SwapModel},\,k}) = \Pr(M_{\text{SwapModel},\,0}) \tag{A.14}$$

$$\Pr(M_{\text{RemoveEdge},\,0}) = 0.00 \qquad \Pr(M_{\text{RemoveEdge},\,k}) = \Pr(M_{\text{RemoveEdge},\,0}) + \frac{0.5k}{N_{\text{gen}}} \tag{A.15}$$

At the beginning of topology optimization, the AddEdge and AddNodeEdge mutations are more likely to be applied and, as such, the average size of the system graphs increases. This enables more exploration of the design space early in the evolution process. The aim is to explore the design space, increasing graph size, and find optical systems which may accomplish the desired functionality, but may contain extraneous components. Then, by trading-off between the probability of increasing the graph size to decreasing it, high-performing systems are further improved by removing unneeded optical elements. To aid this, the Hessian-based sensitivity analysis is used to estimate which optical components could be removed without impacting performance. An optical system, G, is parameterized by $\mathbf{x}_G = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N]^{\mathrm{T}}$, composed of the component parameters comprising the system. For the associated Hessian matrix, $\mathbf{H}_G(\mathbf{x})$, a 'sensitivity score' is assigned to each component model on a graph edge. For a graph with total parameter set, $\mathbf{x}_G \in \mathbb{R}^{N_G}$, containing a optical component model, C, with parameter set $\mathbf{x}_C \in \mathbb{R}^{N_C}$, the score is defined as:

$$S_C = \frac{\sum_{i \in \mathbf{x}_C} \sum_{j \in \mathbf{x}_G} H_{i,j}^{(\text{scaled})}}{N_G \times N_C} \tag{A.16}$$

This score is then used in scaling the likelihood of removing elements via a tournament-style method, such that low-sensitivity components are more likely to be removed, but is not done so determin-

istically. Further, the tournament-style method is more robust to a variety of objective functions where the order-of-magnitude of the cost value and sensitivity value may vary widely between uses.

A.3 Code structure

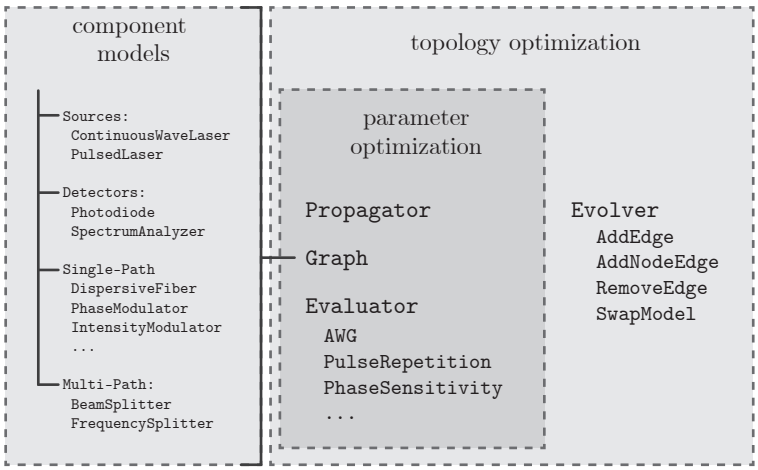


Figure A.4 – Software structure of `ASOPE`.

This Appendix provides a brief white-paper of the `ASOPE` software with code listings, detail about the software structure, and examples for using the optical design tools. Object-oriented programming techniques are used to encapsulate various levels of detail of the optical system.

```
vertices = {'source': SourceVertex(),
            'bs1': BeamSplitter(),
            'bs2': BeamSplitter(),
            'sink': SinkVertex()}
edges    = {('source', 'bs1'): ContinuousWaveLaser(),
            ('bs1', 'bs2'): IntensityModulator(),
            ('bs2', 'sink'): Photodiode()}
graph = Graph(vertices, edges)
```

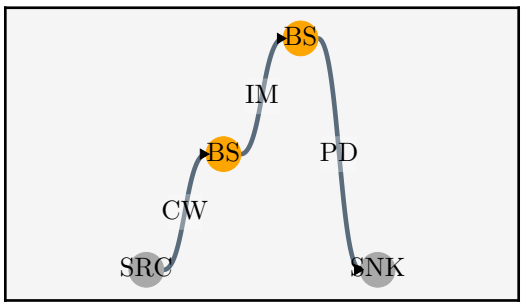


Figure A.5 – Example system.

The `Graph` class encapsulates the system graph, including the component models on the edges and vertices. Graphs are created by specifying the models on the vertices and the edges (edges also indicate a direction of propagation). New vertices can also be added after `Graph` is initialized or the `Graph` can randomly be changed with the mutation operators (i.e. `AddEdge, AddEdgeNode, RemoveEdge, SwapModel`). The `Propagator` class stores simulation data including the time window, time array, frequency array, central frequency, etc. The above code creates the system graph in Fig A.5.

```
propagator = Propagator(window_t=1e-9, n_samples = 2**14, central_wl=1.55e-6)
```

To simulate the system propagation, with the topology and parameters stored in the `Graph` instance,

```
graph.propagate(propagator)
```

The time-domain optical field (or electronic signal at e.g. photodiodes) can then be probed (for further analysis or plotting) as,

```
optical_field = graph.measure('sink')
```

Finally, the evaluation of a graph (both topology and parameter set) is done by an `Evaluator` class instance (each task is defined by a subclass of `Evaluator`). The overall function and its gradient and Hessian functions are compiled partial derivatives via automatic differentiation,

```
evaluator = Evaluator(propagator)  # or AWGEvaluator, PulseRepRateEvaluator, etc.
graph.initialize_func_grad_hess(propagator, evaluator)
```

With these initialized, parameters on a fixed `Graph` topology can be optimized via,

```
graph, xmin, cost, log = parameters_optimize(graph, method='L-BFGS+GA')
```

where the initial guesses are randomly drawn from a uniform distribution over the bounds. Functionality to manipulate the topology of a `Graph` class instance is applied with the `Evolver` class. This class determines valid mutation operators on a graph and encapsulates the topological changes. A topology optimization run requires an instance of the `Graph` class, `Propagator` class, `Evaluator` class, and `Evolver` class.

```
evolver = Evolver()
graph, score, log = topology_optimization(graph, propagator,
                                          evaluator, evolver,
                                          multiprocess=True)
```

Finally, optical and electronic noise (modeled as Additive Gaussian Noise) can be controlled for each type of model, each individual mode, or either turned on or off for all systems and computations with;

```
AdditiveNoise.noise_on = True  # turns all noise contributions on
AdditiveNoise.noise_on = False  # turns all noise contributions off
```

Sensitivity analysis is called as

```
hess, hess_eigenvalues, hess_eigenvectors = hessian_analysis(graph, x)
```

Interpreting suggested solutions

As `ASOPE` is built for the purpose of finding new and potentially intuitive optical systems, understanding the mechanisms and phenomena employed by a proposed system requires interactive analysis tools. In particular it is important and useful to investigate both the system topology, component parameters, and how they interact. To this end, the software includes an interactive dashboard interface for investigating a system graph. The dashboard allows the user to open saved

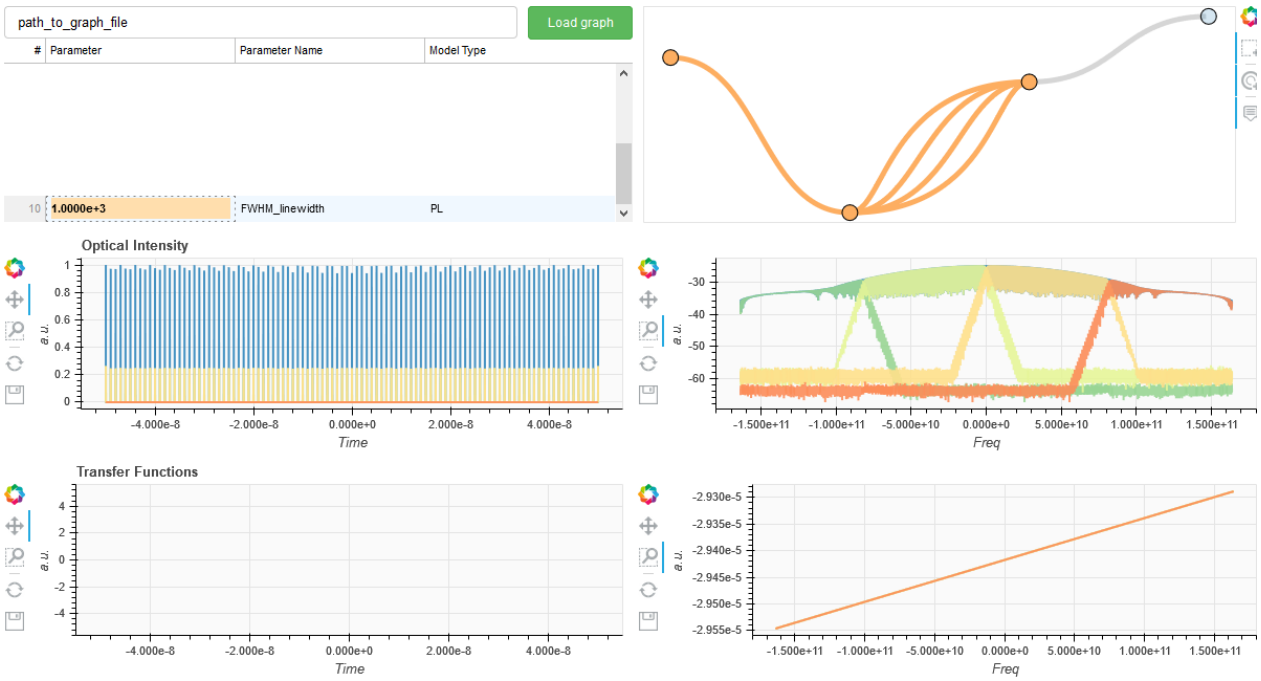


Figure A.6 – Interactive dashboard for exploring proposed systems. A user provides the path string to a directory containing a graph file (serialized object file) and related files. The graph is plotted in the top right, where the user may select different edges and nodes to further inspect – which plots the optical or electronic signal on the selected components and the transfer functions in the time and frequency domains (bottom). A separate panel allows the user to interactively investigate the Hessian matrix and its eigenanalysis.

system graph files (e.g. the Hall of Fame graphs saved during a topology optimization run, or a custom defined graph) and explore the graph topology (top right), the parameters on a given set of optical component models as text (top left), the time- and frequency-domain plots of the optical (or electronic) signal(s) and transfer functions, and the sensitivity analysis of the given solution (i.e. the Hessian matrix along with its analysis of principal directions/curvatures).

Defining new optical components and evaluation functions

The `ASOPE` software was developed such that adding new components to the library is straightforward. The new component is defined by constants (parameter bounds, parameter step sizes, parameter precision, etc.) and a function which acts on the optical field. The function only needs to be written in pure Python and `numpy` functions to be differentiable (test functions are available to ensure the derivative can accurately be calculated, comparing against, e.g., numerical differentiation). A boiler-plate class model is included for reference in the code-base, though following the structure and style of current component models is useful for understanding the programming restraints of `ASOPE` and `autograd`. The same is true for adding a new evaluation function in the form of a new `Evaluator` subclass. Considering more optical degrees-of-freedom would also be possible with minimal changes to the optimization software, and only the components and propagation simulation.

Open-source software tools used

The following is a list of the major software packages utilized in `ASOPE`:

`networkx` Used for graph manipulations. Optical system graphs are a sub-class of the `MultiDiGraph` class [47].

`autograd` Used for automatic differentiation of the optical system performance by tracking of user-defined models written with pure Python and `numpy` [124] functions. A related project, JAX, may be beneficial to use for future work, as it allows Just-In-Time (JIT) compilation of derivatives to be calculated on GPUs [121].

`ray` Used for seamless high-performance parallel processing [116]. Can quickly scale from single core computation (for testing and debugging purposes) to multi-core machines (as used for all experiments in Chapter 4.1) to even high-performance compute clusters.

`scipy` Parameter optimization routines and various digital signal processing tasks [41].

`bokeh` Used to build interactive dashboard for interpreting results and suggested optical systems [125].

I also test the usage of various optimization routines in Python for the parameter optimization. These include: `pyswarm` [43] for Particle Swarm Optimization, `pycma` [126] for the Covariance Matrix Algorithm, `deap` for early versions of the parameter Genetic Algorithm (and the inspiration of the Hall of Fame concept) [45], and `scipy.optimize` for L-BFGS algorithm optimization [41].

A.4 Component library

In this Appendix, the optical component models currently implemented in `ASOPE` are listed, along with considerations or compromises made for automatic differentiation. Models have both design parameters (i.e. tunable values which are optimized) and state parameter constants (i.e. constant values which remain fixed during optimization, but may be altered by the user based on, e.g., datasheets). This allows for matching the `ASOPE` library components to real, commercial components, if desired.

A.4.1 Simulation parameters

Optical simulations in `ASOPE` simulate the time-frequency degree-of-freedom of an optical field. The propagation function manipulates the time-varying complex envelope (i.e. carrier frequency is removed), denoted as $\Psi(t)$, with the tilde character denoting the frequency domain, $\tilde{\Psi}(\omega)$, as the optical spectrum. We assume each component and system is operating in a steady-state regime. The parameters vector (for a single component) is denoted by $\mathbf{x}$. For each component the transfer function is applied in the time-domain,

$$\Psi_{\text{out}} = f(\Psi_{\text{in}}(t), \mathbf{x}) \tag{A.17}$$

or the frequency-domain,

$$\tilde{\Psi}_{\text{out}} = f(\tilde{\Psi}_{\text{in}}(\omega), \mathbf{x}) \tag{A.18}$$

The complex optical field envelope is simulated from time $-T$ to T with M discrete samples. The simulations are axial approximations, single-mode, and do not consider polarization effects. Realistic loss per component is considered in the models.

A.4.2 Optical sources

Continuous wave laser

A continuous wave laser is parameterized by the optical power. Additive white Gaussian noise (AWGN) can be injected to model noise processes in the laser. Central frequency, linewidth, and optical noise are state constants.

$$\Psi_{\text{out}}(t, \mathbf{x}) = \sqrt{x_{\text{power}}} \tag{A.19}$$

Parameter	Description	Bounds	Units
x_{power}	average laser power	$[0.0, 0.1]$	W

Pulsed laser

A pulsed laser is parameterized by three values: the pulse repetition rate, the pulse width, and the peak power. A Gaussian pulse shape is used by default, though the option to switch to a hyperbolic secant or other common shapes is implemented. Central frequency and optical noise are state parameters. We assume timing-jitter and power instability is negligible.

$$\Psi_{\text{out}}(t, \mathbf{x}) = \sqrt{x_{\text{power}}} \exp\left(-4\log(2)\left(\frac{t \bmod x_{\text{rate}}}{x_{\text{width}}}\right)^2\right) \tag{A.20}$$

Parameter	Description	Bounds	Units
x_{P}	peak power	$[0.0, 1.0]$	W
$x_{\text{t}_{\text{rep}}}$	pulse repetition time (inverse of repetition rate)	$[10 \times 10^{-9}, 0.1 \times 10^{-9}]$	s
x_τ	pulse width (FWHM)	$[1 \times 10^{-12}, 100 \times 10^{-12}]$	s

The modulus function is not differentiable with automatic differentiation. A work-around for efficient differentiation with respect to the pulse repetition rate,

$$t_{\text{wrap}} = \frac{x_{\text{rate}}}{\pi}\arcsin\left(\sin\left(\frac{\pi t}{x_{\text{rate}}}\right)\right) \tag{A.21}$$

$$\Psi'(t, \mathbf{x}) = \sqrt{x_{\text{power}}} \exp\left(-\left(\frac{t_{\text{wrap}}}{x_{\text{width}}}\right)^2\right) \tag{A.22}$$

A.4.3 Single spatial path components

Dispersive fiber

Dispersive fiber is parameterized by the length. Dispersion up to second order (group velocity and chromatic dispersion) is modeled, along with loss per length. State parameters for dispersion constants are based on a Corning SMF-28 optical fiber.

$$\tilde{\Psi}_{\text{out}}(\omega, \mathbf{x}) = \tilde{\Psi}_{\text{in}} \exp\left(-ix_z(\beta_1(\omega - \omega_0) + \beta_2(\omega - \omega_0)^2/2) + \alpha x_z\right) \tag{A.23}$$

Parameter	Description	Bounds	Units
x_z	fiber length	$[0, 100]$	km

Variable optical attenuator

A variable optical attenuator is parameterized with the attenuation, in dB.

$$\Psi_{\text{out}}(t, \mathbf{x}) = \Psi_{\text{in}} 10^{x_\alpha/20} \tag{A.24}$$

Parameter	Description	Bounds	Units
x_α	optical attenuation	$[-60, 0]$	dB

Phase modulator

An electro-optic phase modulator is parameterized by the modulation frequency, modulation depth, and global phase shift. Single-tone RF driving is assumed.

$$\Psi_{\text{out}}(t, \mathbf{x}) = \Psi_{\text{in}} \exp(i x_m \cos(2\pi x_f t + x_s)) \tag{A.25}$$

Parameter	Description	Bounds	Units
x_m	modulation depth	$[0, 2\pi]$	radians
x_f	modulation frequency	$[1, 50]$	GHz
x_s	absolute phase shift	$[0, 2\pi]$	radians

Intensity modulator

An electro-optic intensity modulator (Mach-Zehnder modulator) has four free parameters: the modulation frequency, depth, global phase shift, and the bias term. Single-tone RF driving is assumed.

$$\Psi_{\text{out}}(t, \mathbf{x}) = \frac{\Psi_{\text{in}}}{2} + \frac{\Psi_{\text{in}}}{2} \exp(i x_m \cos(2\pi x_f t + x_s) + x_{\text{DC}}) \tag{A.26}$$

Parameter	Description	Bounds	Units
x_m	modulation depth	$[0, \pi]$	radians
x_f	modulation frequency	$[1, 50]$	GHz
x_s	absolute phase shift	$[0, 2\pi]$	radians
x_{DC}	bias	$[0, \pi]$	radians

Note that these parameters are not in voltages, but instead abstract and generalize directly to the phase experienced on the optical field. It would be necessary to back-track and find a modulator with proper specifications and calculate the driving voltages.

Waveshaper

The waveshaper is a programmable optical filter. Each pixel can apply an amplitude and phase change to the optical frequencies with the bandwidth of that pixel. There are a few hundred pixels in the device (depending on the model), so using every pixel as a free parameter can pose a challenge (as system could have many hundreds of parameters with many being redundant). The number of pixels are truncated to N around the central frequency. The parameters are the amplitude and phase setting for N center pixels. Each pixel covers a frequency bandwidth of $\delta\omega$. The transfer function is then modeled as a piece wise constant function.

$$\tilde{\Psi}_{\mathrm{out}}(\omega, \mathbf{x}) = A_{\mathrm{mask}}(\omega) \exp(i\Phi_{\mathrm{mask}}(\omega))\tilde{\Psi}_{\mathrm{in}} \tag{A.27}$$

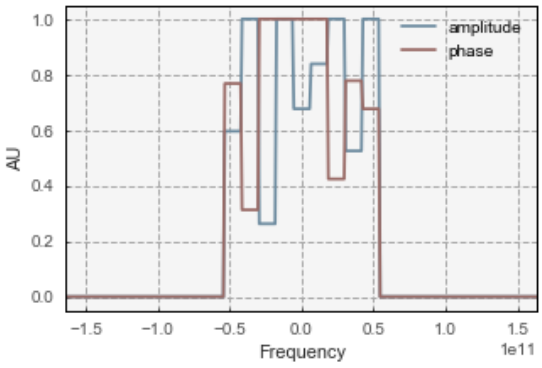
where the amplitude and phase masks, $A_{\mathrm{mask}}, \Phi_{\mathrm{mask}}$, are defined as,

$$A_{\mathrm{mask}}(\omega) = \begin{cases} x_{a_k} & \frac{(2k-1)\delta}{2} < \omega \leq \frac{(2k+1)\delta}{2} \quad | \quad k = -N, \ldots, 0, \ldots, N \\ 0 & \text{else} \end{cases} \tag{A.28}$$

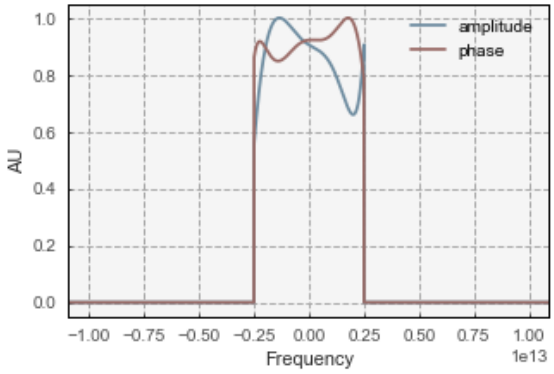
$$\Phi_{\mathrm{mask}}(\omega) = \begin{cases} x_{\varphi_k} & \frac{(2k-1)\delta}{2} < \omega \leq \frac{(2k+1)\delta}{2} \quad | \quad k = -N, \ldots, 0, \ldots, N \\ 0 & \text{else} \end{cases} \tag{A.29}$$

$\delta\omega$ is the pixel resolution of the Waveshaper (12 GHz), and $k = 0$ corresponds to the central wavelength bin.

Parameter	Description	Bounds	Units
$\{x_{a_k},\ k = -N, \ldots, 0, \ldots N\}$	k-th pixel amplitude filter	$[-60, 0]$	dB
$\{x_{\varphi_k},\ k = -N, \ldots, 0, \ldots N\}$	k-th pixel phase shift	$[0, 2\pi]$	rad



(a) Model 1 of the waveshaper, with individual pixels are the free parameters. Parameters are randomly chosen.



(b) Model 2 of the waveshaper, with the amplitude and phase masks calculated as a weighted sum of Bernstein polynomials. Parameters are randomly chosen.

Waveshaper - Second representation

This is physically the same device as the waveshaper above, but with a different parameterization. It models the device operation over the whole bandwidth, but uses fewer parameters the previous model. Instead of being parameterized by each pixel (amplitude and phase for each), a continuous function over the filters bandwidth is generated as a weighted sum of Bernstein polynomials (which form a complete polynomial basis). With this we can represent a large number of different amplitude and phase masks with only a few free parameters. The masks are restricted between 0 and 1 for amplitude, and 0 and 2π for phase.

$$\tilde{\Psi}_{\text{out}}(\omega, \mathbf{x}) = A_{\text{mask}}(\omega) \exp(i\Phi_{\text{mask}}(\omega))\tilde{\Psi}_{\text{in}} \tag{A.30}$$

The amplitude and phase masks are smooth functions over the bandwidth of the device.

$$A_{\text{mask}}(\omega) = \sum_{k=0}^{N} x_{a_k} B_{k,N}(\omega) \tag{A.31}$$

$$\Phi_{\text{mask}}(\omega) = \sum_{k=0}^{N} 2\pi x_{\varphi_k} B_{k,N}(\omega) \tag{A.32}$$

where $B_{k,N}$ is the k-th Bernstein polynomial of degree N. The Bernstein polynomials are non-negative over the range of $[0, 1]$, such that for any $\sum_{k=0}^{N} x_k B_{k,N}(\omega) \leq 1$ assuming that $x_k \leq 1,\ \forall k$.

Parameter	Description	Bounds	Units
$\{x_{a_k},\ k = 0, \ldots N\}$	k-th Berstein polynomial coefficient, amplitude	$[0, 1]$	-
$\{x_{\varphi_k}, k = 0, \ldots N\}$	k-th Berstein polynomial coefficient, phase	$[0, 1]$	-

Optical amplifier

The optical amplifier (such as an EDFA) amplifies incoming signals with a frequency and power dependent gain, and is parameterized by small signal gain, maximum output power, and gain flatness across a given wavelength interval. Additive Gaussian noise is injected to simulate amplified spontaneous emission noise (ASE). For an amplifier, the output frequency-domain optical field is scaled by a frequency-dependent gain factor.

$$\tilde{\Psi}_{\text{out}}(\omega, \mathbf{x}) = G(\omega, \Psi_{\text{in}}, \mathbf{x})\tilde{\Psi}_{\text{in}} \tag{A.33}$$

$$G(\omega, \Psi_{\text{in}}, \mathbf{x}) = G_0(\omega, \mathbf{x}) \left[1 + \frac{G_0(\omega, \mathbf{x})\bar{\Psi}_{\text{in}}}{x_{\text{Pmax}}}\right]^{-1} \tag{A.34}$$

where $\bar{\Psi}$ is the time-averaged optical power and x_{Pmax} is the maximum output optical power.

$$G_0(f, \mathbf{x}) = x_g \exp(-\omega^2/\beta) \tag{A.35}$$

$$\beta = \beta(x_{\text{flatness}}, x_{\lambda min}, x_{\lambda max}) \tag{A.36}$$

β is a scaling factor such that the expected gain flatness x_{flatness} is obtained across the operational wavelength range $[x_{\lambda_{\text{min}}}, x_{\lambda_{\text{max}}}]$. $\bar{\Psi}_{\text{in}}$ is the average input power.

Parameter	Description	Bounds	Units
x_g	peak small signal gain	$[0, 30]$	dB

The following are state parameters (constants), based on commercial datasheets,

Parameter	Description	Bounds	Units
x_g	peak small signal gain	$[0, 30]$	dB
x_{Pmax}	maximum output power	$[1 \times 10^{-7}, 1 \times 10^{1}]$	W
x_{flatness}	gain flatness across operation	$[0, 15]$	dB
x_F	noise factor	$[3, 10]$	dB
$x_{\lambda_{\text{min}}}$	minimum wavelength of operation	$[1530, 1600]$	nm
x_{λ_0}	peak wavelength of operation (max gain)	$[1535, 1612]$	nm
$x_{\lambda_{\text{max}}}$	maximum wavelength of operation	$[1540, 1625]$	nm

A.4.4 Multiple spatial path components

Components which map between spatial paths include couplers and splitters, and are designed for an arbitrary number of inputs and outputs. Unlike the other component models described to this point, splitting elements have equality constraints between parameters (such that momentum and energy are conserved from the input to the output of the component). In order to avoid constrained optimization, the constraints are instead adapted into the model of the components via substitution.

Variable power splitter

The $M \times N$-port power splitter is parameterized by $N-1$ splitting amplitudes, $x_{\alpha_k} \forall k \in [1, N-1]$, from which the variable power splitting ratios, w_k are defined as,

$$w_k = (1 - x_{\alpha_k}) \prod_{\ell=1}^{k-1} x_{\alpha_\ell} \quad \forall k \in [1, N] \tag{A.37}$$

and $\alpha_N \equiv 0$. This constrains energy to be conserved. Each w_k is between 0 and 1. The $M \times N$ scattering matrix is calculated as,

$$S_{m,n} = \sqrt{w_n} \exp\left[i\pi \left(\frac{n}{N-1} + \frac{m}{M-1}\right)\right] \tag{A.38}$$

such that it maps between incoming and outgoing spatial paths as,

$$\begin{pmatrix} \Psi_{1,\text{out}}(t) \\ \vdots \\ \Psi_{N,\text{out}}(t) \end{pmatrix} = \mathbf{S} \begin{pmatrix} \Psi_{1,\text{in}}(t) \\ \vdots \\ \Psi_{M,\text{in}}(t) \end{pmatrix} \tag{A.39}$$

Parameter	Description	Bounds	Units
x_{α_k}	k-th branching ratio	$(0, 1)$	-

We note that the phase shifts experienced between different input/output ports may be dependent on the type of splitter/coupler used. With this model, when $M = N = 1$ (i.e. one input, one output), the model is fixed and the optical field is unaffected (this corresponds to a simple connector, such as a fiber splice or fiber connector). This model assumes that all incoming paths are coupled together without loss, and further split based on the coupling ratio parameters w_k.

Frequency-dependent splitter

A frequency dependent splitter is parameterized by the cutoff frequencies of each output port (as a ratio of the simulated frequency range, $\Delta\omega$) with the frequency dependent filtering modeled by multiplied logistic functions. A separate band of the optical spectrum exits each output port. The $M \times N$-port frequency splitter is parameterized by $N-1$ values x_{α_k}, from which the cutoff frequencies are calculated for each output port

$$\omega_k = \Delta\omega \sum_{\ell=0}^{k} \left[(1 - x_{\alpha_\ell}) \prod_{m=1}^{\ell-1} x_{\alpha_m} \right] \quad \forall k \in [0, N] \tag{A.40}$$

with $x_{\alpha_0} \equiv 1$, $x_{\alpha_N} \equiv 0$, and $x_{\alpha_i}\ \forall i \neq 0, N$ as the tunable parameters. The (frequency-dependent) scattering matrix (with indices representing different spatial paths) is,

$$S_{m,n}(\omega) = \text{logistic}(-\omega_{n-1}) \times \text{logistic}(\omega_n) \qquad \forall n \in [1, N] \tag{A.41}$$

$$= \left(\frac{1}{1 + \exp[-p(\omega - \omega_{n-1})]} \right) \left(\frac{1}{1 + \exp[p(\omega - \omega_n)]} \right) \tag{A.42}$$

with p as a dimensionless constant which defines the 'sharpness' of the cutoff ($p \approx 300$). This scattering matrix maps incoming optical fields to output paths as,

$$\begin{pmatrix} \Psi_{1,\text{out}}(\omega) \\ \vdots \\ \Psi_{N,\text{out}}(\omega) \end{pmatrix} = \mathbf{S} \begin{pmatrix} \Psi_{1,\text{in}}(\omega) \\ \vdots \\ \Psi_{M,\text{in}}(\omega) \end{pmatrix} \tag{A.43}$$

The multiplication of two logistic function is used to approximate a pulse (rectangular) function in such a way which is differentiable via automatic differentiation.

Parameter	Description	Bounds	Units
x_{α_k}	branching cutoff	$(0, 1)$	-

A.4.5 Optical detectors

Photodiode

The photodiode is different than previous components, as it converts the time-varying optical field into a time-dependent electronic signal. The photodiode model used has no tunable parameters (i.e. none are optimized). The device is parameterized by its fixed responsivity, bandwidth, and maximum photocurrent. Electronic bandwidth is modeled as a n^{th}-order Butterworth filter. Both thermal noise and electronic shot noise can be added as AGN (also subject to the Butterworth filter).

The output voltage signal from the photodiode is a frequency-filtered RF signal proportional to the intensity of the input optical field.

$$I_{\text{out}}(t, \Psi_{\text{in}}, \mathbf{x}) = \mathcal{F}^{-1}[B_n \times \mathcal{F}[I_{\text{inf}}(t, \Psi, \mathbf{x})]] \tag{A.44}$$

where B_n is a n^{th} order Butterworth filter with bandwidth x_{bw}. The infinite-bandwidth current $I_{\text{inf}}(t, \Psi, \mathbf{x})$ is proportional to the square of the optical field – but also takes into account saturation of the photodiode at its maximum photocurrent value,

$$I_{\text{inf}}(t, \mathbf{x}) = \min(x_{\text{resp}}|\Psi_{\text{in}}(t)|^2,\ x_{\text{maxPC}}) \tag{A.45}$$

where Ψ_{out} is the output photocurrent.

In order to allow differentiation via reverse-mode automatic differentiation for the `min` function, the following approximation (which is effectively equivalent to Eqn. A.45 for large p) is used,

$$I_{\text{inf}}(t, \mathbf{x}) = x_{\text{resp}}|\Psi_{\text{in}}(t)|^2 \left(1 + \left|\frac{x_{\text{resp}}\,|\Psi_{\text{in}}(t)|^2}{x_{\text{maxPC}}}\right|^p\right)^{-1/p} \tag{A.46}$$

All the following parameters are state constants.

Parameter	**Description**	**Bounds**	**Units**
x_{bw}	bandwidth of photodiode	$[0.5, 100]$	GHz
x_{resp}	responsivity of photodiode	$[0, 1]$	A/W
x_{maxPC}	maximum output photocurrent	$[0, 0.1]$	A