CHAPITRE 2

Concevoir par la computation : notions, systèmes et enjeux

Aurélie de Boissieu[1]

Le terme computationnel renvoie, au sens large, à ce qui relève de la computation : un anglicisme couramment employé pour désigner les capacités de calcul et de traitement de données propres à l'informatique. Les approches dites computationnelles connaissent depuis plusieurs années un essor marqué, certains diront même une révolution, dans tous les secteurs de notre quotidien, professionnels et personnels. C'est le cas également en architecture et dans la construction en général : l'usage des outils numériques s'y diversifie, depuis des pratiques expérimentales encore relativement niches jusqu'à des modes de production devenus courant.

La technologie est souvent décrite comme une prothèse[2], physique ou cognitive, permettant à l'humain de dépasser certaines limites : un *Pharmakon*, à la fois remède et poison[3]. Dans cette perspective, il devient indispensable de s'interroger sur ses spécificités pour comprendre ses effets. Dans ce chapitre, nous nous intéressons en particulier aux algorithmes développés et mobilisés dans ces pratiques de conception architecturale dites computationnelles.

La recherche et l'expérimentation en conception architecturale et urbaine computationnelle émergent dès le début des années 1960, à travers notamment les travaux d'Ivan Sutherland, Yona Friedman et Nicholas Negroponte comme le montre bien le chapitre 1. Toutefois, la généralisation des outils numériques dédiés à l'industrie de la construction n'intervient qu'à la fin des années 1980, avec la démocratisation de la CAO (avec des outils de dessin technique

1 Laboratoire de culture numérique en architecture, Université de Liège
aurelie.deboissieu@uliege.be

2 Alombert, A. 2025. *De la bêtise artificielle*. Allia

3 Stiegler, B. 2014. *Nouvelle critique de l'anthropologie – Rêves, cinémas, cerveaux*. Séminaire Pharmakon. IRI

comme AUTOCAD), puis avec l'essor d'autres technologies (modélisation 3D, rendu, gestion documentaire, fabrication numérique). Aujourd'hui, les innovations se succèdent à un rythme soutenu. Mais lorsque l'on parle d'« assisté par ordinateur », de « numérique » ou de « computationnel », que distingue-t-on exactement ? Et comment définir la conception computationnelle en architecture, en particulier au regard des relativement récentes intelligences artificielles génératives ? Voilà les questions que nous abordons dans ce chapitre.

1. Assisté par ordinateur, automatisé, numérisé ou computationnel ?

Les pratiques de conception computationnelle sont hétérogènes, mais elles partagent un socle commun : elles s'appuient à la fois sur la puissance de calcul des ordinateurs et sur les modes de pensée qui en découlent. L'ordinateur rend manipulables des géométries et des opérations qui, sans lui, seraient trop fastidieuses ou trop lourdes à soutenir dans la durée. Surtout, au-delà de l'automatisation de processus existants, la capacité à traiter de grandes masses de données transforme les manières de raisonner, de représenter et de concevoir[4].

Dès 1969, Nicolas Negroponte[5] propose une distinction importante : d'un côté, l'« assisté par ordinateur » renvoie à des processus fondamentalement manuels, simplement soutenus par des outils informatiques ; de l'autre, l'« informatisé » ou le « numérisé » désigne des processus nouveaux, rendus possibles par l'ordinateur. Dans les années 2000 l'architecte Kostas Terzidis[6] propose ensuite d'aller plus loin en réservant le terme *computationnel* aux usages qui relèvent non seulement d'une technique (l'ordinateur comme outil), mais d'un paradigme de pensée (la computation). Finalement, les pratiques numériques peuvent être pensées sur un spectre, allant de la « simple » automatisation de pratiques existantes, à un changement de mode de pensée marqué par la computation (figure 1).

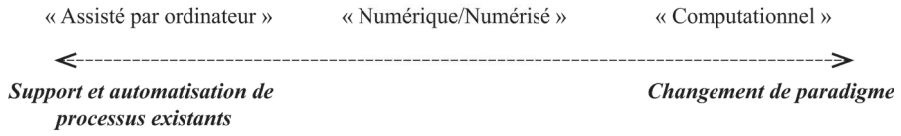

Figure 1 : « D'assisté par ordinateur » à « computationnel ».

Cette distinction suggère qu'adopter la conception computationnelle requiert d'abord un changement de mentalité, et non une simple transposition numérique d'une pratique préexistante : « les algorithmes sont utilisés non pas pour améliorer les conceptions architecturales, mais pour les concevoir elles-mêmes »[7].

4 Denning, P. J. & Tedre, M. 2019. *Computational thinking*. MIT Press

5 Negroponte, N. 1969. *Towards a humanism through Machines*. in Computational design thinking (eds. Menges, A. & Ahlquist, S.) AD Readers

6 Terzidis, K. 2006. *Algorithmic Architecture*. Architectural Press

7 *ibid*

Dans ce cadre, les technologies et la pensée computationnelles, dites aussi parfois algorithmiques, servent à définir et à manipuler simultanément des flux de données et des géométries de projet. Le concepteur formule des séries d'instructions, des règles et des relations sous forme d'algorithmes, afin de contrôler la conception, d'en faciliter l'analyse, la modification et, le cas échéant, la génération. Ces capacités ouvrent des perspectives nouvelles pour le secteur de la construction[8].

Afin de préciser ce que recouvrent ces pratiques, nous nous attarderons ici sur trois déclinaisons de ces modes de pensée fréquemment mobilisées dans la littérature : la conception paramétrique, la conception générative et la conception algorithmique. Comme le soulignent les chercheurs et architectes Menges et Ahlquist[9], la compréhension du fonctionnement des systèmes numériques est un enjeu central de la conception computationnelle. Les définitions proposées ci-dessous visent donc à caractériser ces trois types de systèmes (paramétrique, génératifs et algorithmiques), en gardant à l'esprit que leurs frontières restent parfois discutées dans l'industrie comme dans la littérature académique[10].

2. Conception paramétrique

Un système paramétrique peut être décrit comme une association de règles et de contraintes explicites, organisées selon un ordre défini. Des entrées (*inputs*) sont introduites dans le système ; les règles et contraintes sont alors résolues par propagation de ces entrées à travers le système, produisant des sorties (*outputs*) cohérentes (figure 2a). L'ensemble constitué par : la plage d'entrées possibles (les paramètres), les règles (mathématiques, géométriques, etc.) et les relations entre ces éléments forme le système paramétrique (figure 2b). À système identique, des entrées différentes produisent des sorties différentes. Les relations entre paramètres, règles et sorties sont définies explicitement avant l'exécution : elles instaurent des contraintes et des dépendances entre éléments.

Un point technique souvent déterminant est que, dans un système paramétrique au sens strict, la propagation des dépendances est unidirectionnelle : il n'y a pas de boucles possibles, et le modèle est dit acyclique. Malgré cette limitation, des systèmes complexes émergent de la combinaison de règles et de paramètres relativement simples.

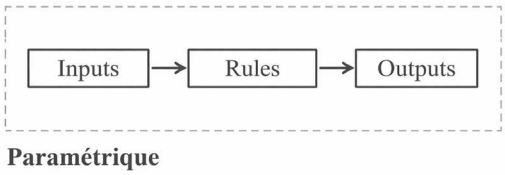

Figure 2a : Entrées, règles et sorties dans un système paramétrique : dépendances principales.

8 Carpo, M. 2017. *The second digital turn: design beyond intelligence*. MIT Press
Shelden, D. 2020. *The disruptors : Technology-driven architect-entrepreneurs*. Wiley

9 Menges, A. & Ahlquist, S. 2011. *Computational Design Thinking*. Wiley

10 Caetano, I., Santos, L. & Leitão, A. 2020. *Computational design in architecture: Defining parametric, generative, and algorithmic design*. Frontiers of Architectural Research

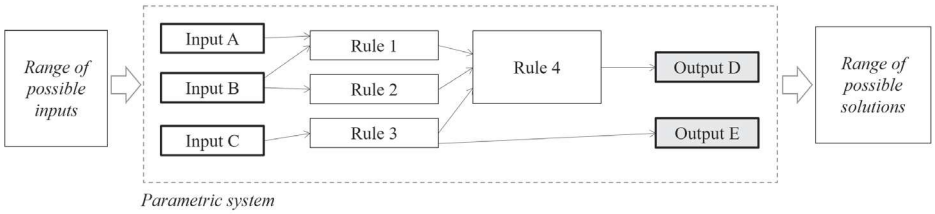

Figure 2b : Entrées, règles et sorties dans un système paramétrique : systèmes complexes par combinaisons simples de règles et de paramètres.

Dans la conception paramétrique, le concepteur travaille à deux niveaux étroitement articulés : le premier à la définition du système (règles, paramètres, dépendances, vus précédemment) et le second à l'exploration des résultats produits par ce système, appelés *instances*. Lors de la conception d'un algorithme paramétrique le processus progresse par itérations : l'exploration d'*instances* rétroagit sur la définition du système, et inversement (fig. 3). Il arrive que des algorithmes paramétriques soient réappropriés par d'autres concepteur que celui ou celle qui l'a développé. Quoi qu'il en soit, la créativité du concepteur intervient à ces deux niveaux, dans la formulation du système comme dans l'interprétation et la sélection des instances pertinentes.

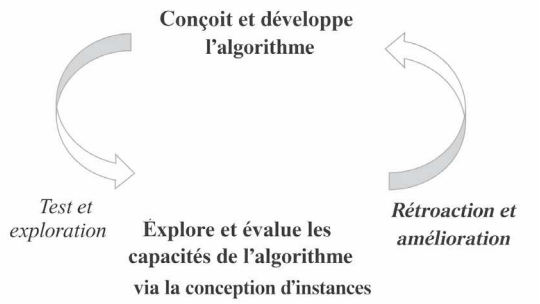

Figure 3 : De la conception d'un système paramétrique à l'exploration d'exemples, et inversement.

Un exemple caractéristique de conception paramétrique est la gare de Waterloo à Londres, conçue par Grimshaw au début des années 1990. Contrainte par un site urbain dense et étroit, la largeur de la gare se réduit quand on s'éloigne des quais (voir figure 4a). Le choix de l'architecte a été de créer une continuité visuelle de la couverture de la gare, malgré ses changements de dimensions. Ici, c'est la logique de la charpente métallique qui assure cette cohérence, sous la forme d'une variabilité continue, rendue possible par une conception paramétrique du système de poutres treillis (figure 4b).

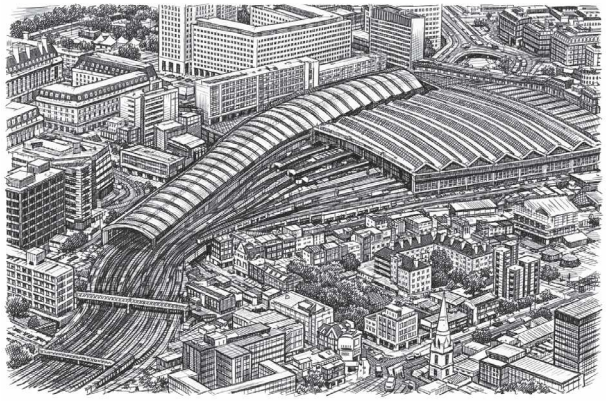

Figure 4a : Gare de Waterloo : vue extérieure. Sur un site dense au centre de Londres, la portée du toit s'élargit vers l'entrée piétonne de la gare.

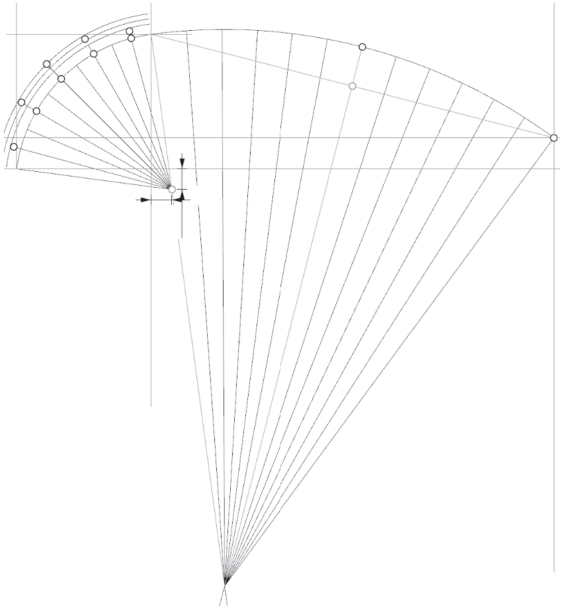

Figure 4b : Esquisse géométrique de la charpente de la gare de Waterloo (Aurélie de Boissieu d'après Shane Burger).

Les relations géométriques (contraintes et dépendances) formalisent deux arcs asymétriques (figures 4). À partir des limites du site, le modèle paramétrique permet non seulement de générer et d'explorer rapidement la géométrie de la couverture[11] et de contrôler des variantes, mais aussi de faciliter une fabrication rationalisée et réglementée des poutres. Dans cette gare, chaque poutre est de dimension différente tout en répondant aux mêmes principes géométriques (figures 5).

11 Burger, S. 2011. *Natural and intuitive. http://shaneburger.com/2011/08/designingdesign/*

Figure 5 : Gare de Waterloo : vue intérieure.

On voit bien dans cet exemple comment on est passé de la conception d'une géométrie singulière à la conception d'un système contrôlant la géométrie et le flux de travail de production. Cette articulation entre conception et construction constitue un thème récurrent des pratiques computationnelles (voir le chapitre 4 à ce sujet).

Un des outils de modélisation paramétrique les plus courant est Grasshopper (ses conditions de production sont largement discutées dans le chapitre 11).

Finalement, parce que les contraintes y sont unidirectionnelles, les systèmes paramétriques sont relativement faciles à anticiper et à maîtriser. Cette propriété rend l'approche plus accessible que d'autres, mais elle la limite également : certaines intentions de conception ne se laissent pas aisément exprimer dans des structures acycliques.

La conception générative, que l'on aborde maintenant, gagne en flexibilité par rapport à l'approche paramétrique, mais au prix d'une complexité accrue.

3. Conception générative

Comme la conception paramétrique, la conception générative est une approche fondée sur des règles capables de produire une diversité de résultats. Toutefois, les méthodes algorithmiques mobilisées peuvent être plus variées et ne sont pas nécessairement contraintes par l'acyclicité. L'objectif de l'approche générative est de définir un *espace de problèmes* à l'aide d'ensembles de règles et d'instructions, ainsi que des domaines d'entrée *(inputs)* clairement spécifiés. À partir de cette formulation, les systèmes génératifs produisent une gamme de résultats, appelé un *espace de solutions*.

Cet *espace de solutions* peut ensuite être évalué, classé et exploré au regard de critères explicites, dits *objectifs* (voir également à ce sujet le chapitre 3). Il devient alors possible de rechercher des solutions qui optimisent un ou plusieurs objectifs visés (figure 6). Dans de nombreux cas, l'intérêt principal de cette approche réside dans la gestion d'objectifs multiples, parfois contradictoires : l'optimisation permet ainsi de rendre plus lisibles certains compromis structurels, énergétiques, économiques ou formels, et d'éclairer les effets de ces objectifs concurrents sur le processus génératif.

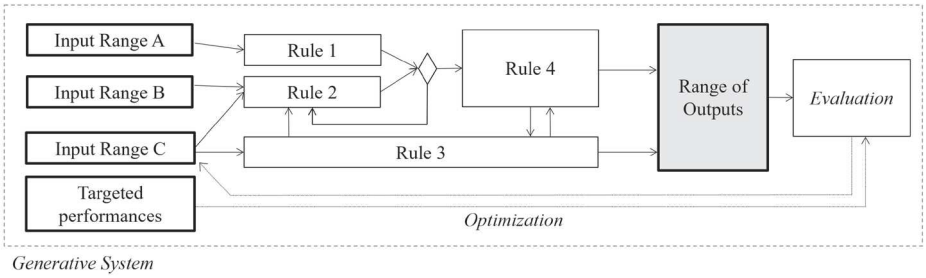

Figure 6 : L'approche générative.

Au début des années 2000, l'agence EZCT[12] a développé un intéressant projet expérimental de chaise à partir de cette approche. L'algorithme développé part d'une matrice de voxels (l'équivalent de pixel en trois dimensions) correspondant aux dimensions globales d'une chaise (le « domaine de définition » voir figures 7). À chaque itération, l'algorithme ajoute ou retire de la matière, puis évalue les performances de la solution produite (figures 7c), notamment au regard d'une stabilité structurelle testée sous charges multiples (figures 7a), ainsi que du poids et du volume de l'instance de chaise produite. L'équilibre entre ces objectifs (par exemple : une chaise plus légère ou plus robuste) est réglé par l'algorithme. Les solutions les plus performantes à une itération sont sélectionnées et réintroduites dans le système pour améliorer l'itération suivante, selon une logique d'algorithme génétique (voir au sujet des algorithmes génétiques le chapitre 3 section 3.1). La répétition de ces boucles de rétroaction permet d'optimiser progressivement la génération globale et de constituer un espace de solutions « performant », à partir duquel certaines instances peuvent être sélectionnées par l'architecte selon des critères qui lui seront propres, puis fabriquées (figures 7d).

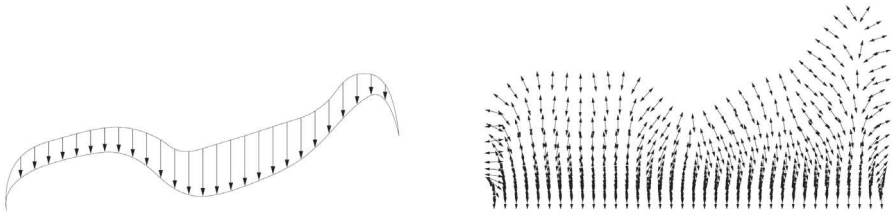

Figure 7a : EZCT : études structurelles préliminaires pour une chaise structurelle, stratégie de charges multiples (crédits : Philippe Morel).

12 Morel, P., Agid, F. & Feringa, J. 2004. Studies on Optimization : *Computational chair design using genetic algorithms. http://transnatural.org/wpcontent/uploads/2011/09/EZCT_Booklet-Screen.pdf*

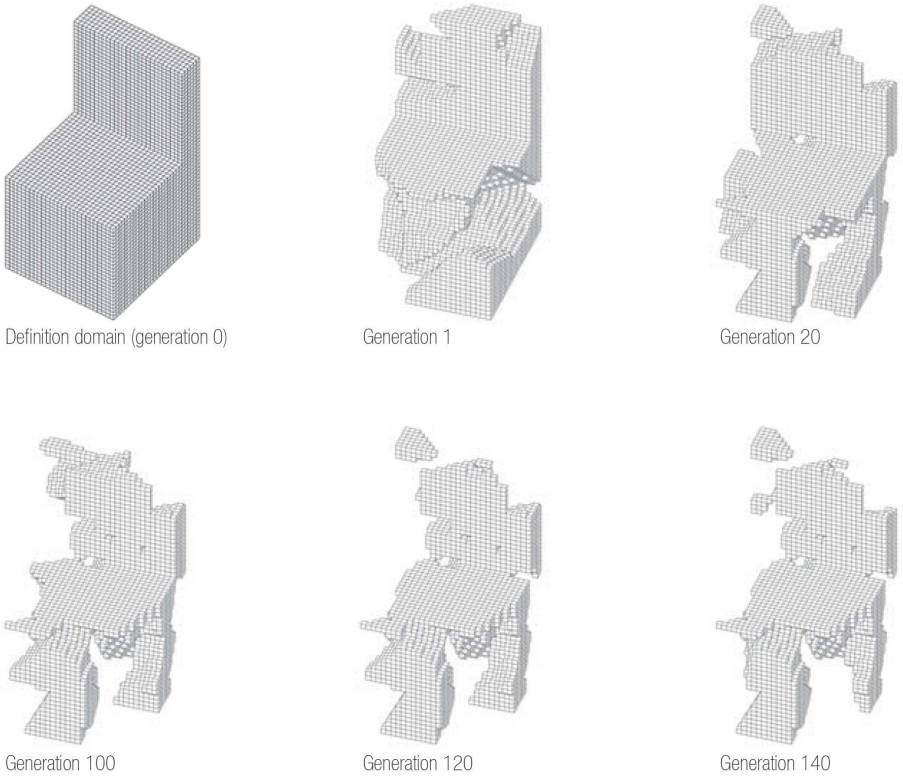

Figure 7b : EZCT : chaise computationnelle, extraits de géométries issues de l'espace de solutions (crédits : Philippe Morel).

CHAIR	VOXELS	DIMENSIONS (PxLxH)	VOLUME	WEIGHT 1 *	WEIGHT 2 *	STABILITY
Test1-860	4714	52 cm x 54 cm x 100 cm	0.0377 m3	18.85 kg	22.62 kg	FINE
Test2-700	4420	54 cm x 50 cm x 96 cm	0.0353 m3	17.65 kg	21.18 kg	FINE
Test3-840	4789	52 cm x 40 cm x 100 cm	0.0383 m3	19.15 kg	22.98 kg	LOW
Test4-640	4456	60 cm x 46 cm x 98 cm	0.0356 m3	17.8 kg	21.36 kg	FINE
Test5-620	2953	54 cm x 44 cm x 94 cm	0.0236 m3	11.8 kg	14.16 kg	LOW
Test6-680	4412	50 cm x 60 cm x 96 cm	0.0331 m3	16.55 kg	19.86 kg	FINE
Test7-680	4158	56 cm x 54 cm x 92 cm	0.0332 m3	16.6 kg	19.92 kg	LOW
Test8-700	2885	60 cm x 46 cm x 92 cm	0.0230 m3	11.5 kg	13.8 kg	LOW
Test9-720	2334	50 cm x 36 cm x 92 cm	0.0186 m3	9.3 kg	11.16 kg	LOW
Test10-760	3920	50 cm x 44 cm x 92 cm	0.0186 m3	15.65 kg	18.78 kg	LOW

Figure 7c : EZCT : chaise computationnelle, extraits d'évaluations de l'espace de solutions (crédits : Philippe Morel).

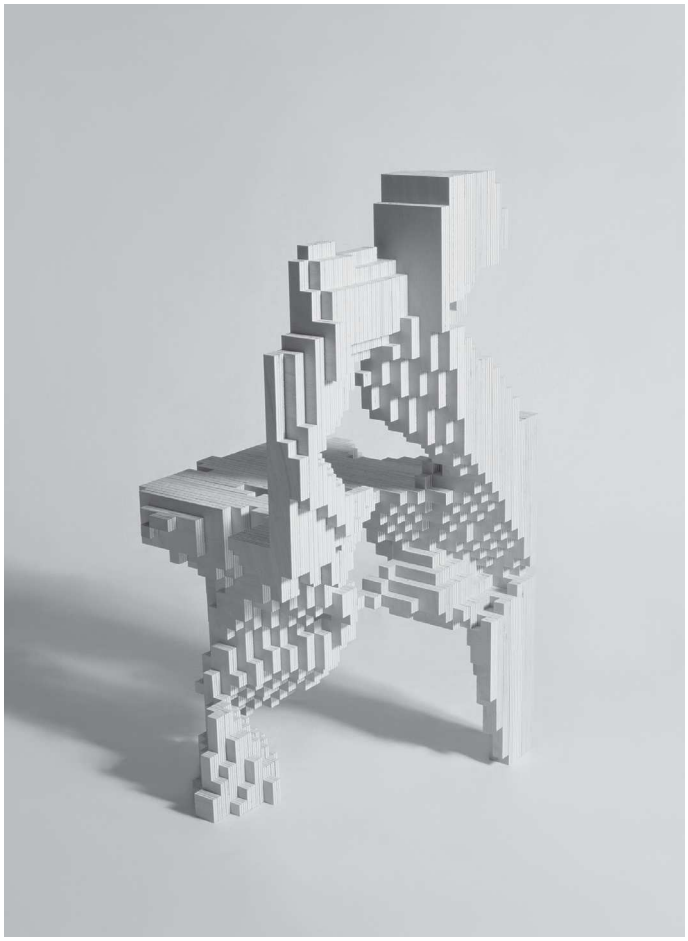

Figure 7d : Modèle de chaise « T1-M 860 » (instance fabriquée)
(crédits : Philippe Morel).

4. Conception algorithmique

La conception algorithmique est fréquemment citée dans la littérature sur la conception computationnelle, bien que son périmètre recoupe souvent celui des approches paramétriques et génératives. Elle se caractérise généralement par un engagement plus direct avec les méthodes et la pensée computationnelles, au-delà de l'usage de l'ordinateur comme simple outil de représentation formelle. Kostas Terzidis, déjà cité précédemment, souligne notamment que la conception algorithmique peut permettre au concepteur d'« aller au-delà du commun et du prévisible ». Plus largement, elle offre une marge de liberté accrue dans l'équilibre délicat entre intention et émergence.

Dans tout algorithme (qu'il soit paramétrique ou génératif), l'association de règles simples peut produire une complexité telle que les résultats deviennent difficilement anticipables :

c'est ce que l'on désigne par émergence. Cette notion est courante en systémique, la science des systèmes, où le tout est considéré comme supérieur à la somme des parties. Cette propriété prend une importance particulière lorsque les algorithmes traitent des volumes massifs de données *(big data)* ou exploitent des espaces de recherche très vastes. Il devient donc difficile au cerveau humain d'anticiper certains résultats d'algorithme : l'émergence devient alors imprévisible. C'est ici que ce situe les pratiques dites algorithmiques dans la littérature.

À cet égard, la chaise computationnelle d'EZCT (figures 7) peut être interprétée comme un cas frontière, relevant à la fois de la conception générative et de la conception algorithmique : la stratégie de conception fait une place importante à l'émergence, conduisant à des instances inattendues. Ce type d'exemple rappelle que la caractérisation d'une approche dans un sous-ensemble unique n'est pas toujours évidente, et que des chevauchements sont fréquents.

On peut alors faire ressortir deux caractéristiques saillantes de la pensée computationnelle : premièrement, un changement de paradigme vers la computation (axe vertical dans la figure 8) et deuxièmement l'apparition de phénomènes d'émergence, où les résultats sont partiellement imprévisibles (axe horizontal). Ces caractéristiques distinguent la conception computationnelle de la CAO au sens classique, mais aussi de l'usage d'objets paramétriques préexistants ou de l'automatisation de flux de travail strictement définis, pour lesquels les résultats restent largement anticipables. Elles permettent enfin de situer relativement les approches paramétriques, génératives et algorithmiques (figure 8).

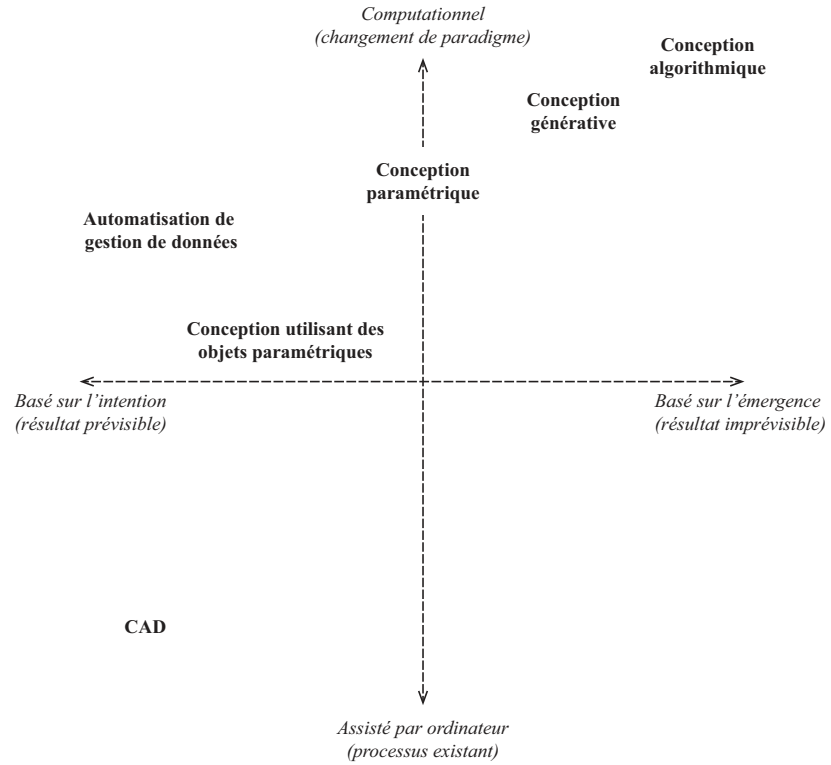

Figure 8 : Conception paramétrique, générative et algorithmique : vers des pratiques de pensée computationnelle et d'émergence.

5. Conception computationnelle, programmation visuelle et IAG

On notera que ces différentes approches computationnelles, paramétriques, génératives et algorithmiques ne dépendent pas d'outils spécifiques, mais désignent bien des approches, des « architectures » de systèmes algorithmiques.

Les premières tentatives d'intégration de la computation en architecture, dans les années 1960, supposaient une maîtrise importante de l'informatique. L'outillage du concepteur s'est d'abord constitué par transfert depuis d'autres secteurs (en particulier l'aéronautique ou le design industriel) avant de se spécialiser progressivement, d'abord vers des outils destinés aux artistes, puis vers des environnements conçus pour les besoins de l'industrie de la construction.

Cette situation a profondément évolué. Les outils de programmation visuelle abaissent aujourd'hui le seuil d'entrée : ils permettent de formaliser des paramètres et des règles sous forme de composants, et d'exprimer leurs relations via des connexions. Le résultat prend le plus souvent la forme d'un script (ou graphe de calcul), autrement dit d'un graphe de dépendances entre opérations (figure 9). Les relations de dépendances entre les fonctions y sont lisibles explicitement. Ces environnements proposent généralement un ensemble de fonctions prêtes à l'emploi, tout en restant extensibles au moyen de librairies, de plugins ou de développements spécifiques.

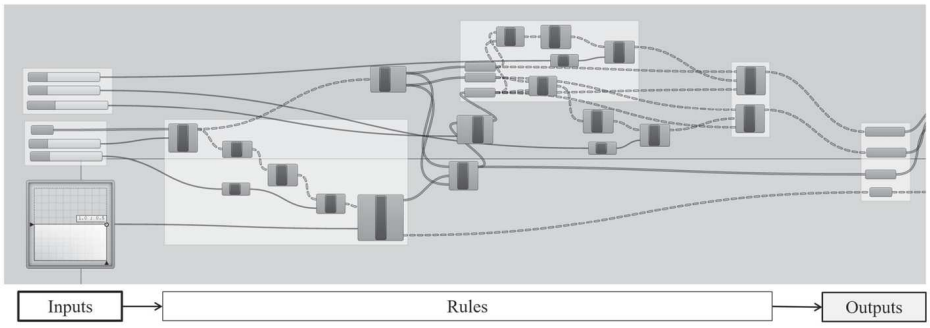

Figure 9 : Exemple de script développé avec l'outil Grasshopper.

De tels outils soutiennent une grande diversité de pratiques de conception computationnelle, y compris celles distinguées dans ce chapitre : paramétrique, générative et algorithmique. Ils se prêtent particulièrement bien à la conception paramétrique, dont les dépendances unidirectionnelles sont plus aisées à représenter et à exécuter dans un graphe. En revanche, les logiques de boucles, d'itérations et de rétroaction, plus caractéristiques des approches génératives ou algorithmiques, deviennent rapidement difficiles à manipuler « par câblage ». La programmation textuelle demeure souvent plus adaptée lorsqu'il s'agit d'implémenter ces mécanismes de manière robuste.

L'intelligence artificielle générative est aujourd'hui une technologie qui évolue à très grande vitesse et qui se retrouve dans de nombreux outils. Ainsi des librairies d'algorithmes relevant de l'intelligence artificielle symbolique (comme les algorithmes génétiques) ou connexionniste (comme les réseaux de neurones) sont depuis plusieurs années incluses dans des outils

existants comme Grasshopper. Et de nouveaux outils apparaissent : des outils dédiés à la génération d'architectures comme archistar.ai, ou encore à la génération de modèles paramétriques ou génératifs eux-mêmes comme Raven[13] ou certaines fonctionnalités d'Hypar[14].

À l'heure actuelle, les IAG sont extrêmement mis en avant lorsque que l'on aborde le sujet de la conception computationnelle. Ainsi, Philippe Morel, auteur via son bureau EZCT du projet discuté précédemment, et directeur du Master Science in Architectural Computation de la Bartlett School of Architecture, décrit notamment «un glissement des techniques algorithmiques classiques vers un recours massif au Deep Learning, aux modèles de diffusion et aux grands modèles de langage »[15] et souligne à la fois l'accessibilité actuelle de ces techniques et la rapidité avec laquelle elles se diffusent.

Conclusion

Ce chapitre a établi un cadre de définitions visant à clarifier ce que l'on entend par « assisté par ordinateur », « numérisé » et « computationnel », en soulignant que la conception computationnelle ne se réduit pas à l'adoption d'outils, mais engage un paradigme de pensée fondé sur la formulation de règles, la manipulation de données et l'exécution d'algorithmes. La distinction proposée entre conception paramétrique, générative et algorithmique permet de caractériser trois familles de systèmes : la première privilégie des dépendances explicites et acycliques, relativement maîtrisables ; la seconde organise un espace de problèmes et explore un espace de solutions au regard d'objectifs, souvent multiples ; la troisième met davantage l'accent sur la puissance combinatoire des règles et sur l'émergence, c'est-à-dire sur la possibilité de résultats partiellement imprévisibles.

Au-delà de cette typologie, l'enjeu principal est méthodologique : ces approches transforment ce que l'on peut définir, mesurer, faire varier et comparer dans un projet, en déplaçant une part de l'activité de conception vers la spécification du système (règles, données, critères) plutôt que vers la seule production d'une forme. C'est dans ce déplacement que se joue une partie de la portée, mais aussi des limites, de la conception computationnelle : la qualité des résultats dépend autant des hypothèses encodées que des procédures d'exploration et d'évaluation.

Enfin, l'irruption des intelligences artificielles génératives reconfigure ce paysage, à la fois sur le plan des pratiques et sur celui de la formation : elle accentue l'importance d'une compréhension critique des outils, de leurs modèles et de leurs conditions d'usage, afin d'éviter une délégation aveugle à des « boîtes noires ». Les distinctions établies ici offrent un vocabulaire et une grille de lecture pour situer ces technologies, et pour discuter, de manière rigoureuse, ce qu'elles automatisent effectivement, ce qu'elles rendent émergent, et ce qu'elles déplacent dans l'économie de la décision architecturale.

13 *https://www.raven.build/*

14 *https://hypar.io/*

15 "from classical algorithmic techniques towards massive uses of Deep Learning, diffusion models, [and] Large Language Models" Morel, Ph. 2025. *Doctoral consortium: Intelligence*. Conférence pour Digital Futures.

Références

ALOMBERT, A. 2025. *De la bêtise artificielle*. Paris : Éditions Allia.

CARPO, M. 2023. *Beyond Digital: Design and Automation at the End of Modernity*. Cambridge (MA) : MIT Press.

CHAILLOU, S. 2021. *L'intelligence artificielle au service de l'architecture*. Paris : Éditions du Moniteur.

DAVIS, D. 2013. *Modelled on Software Engineering: Flexible Parametric Models in the Practice of Architecture*. Thèse de doctorat, RMIT University.

DENNING, P. J. ; TEDRE, M. 2019. *Computational Thinking*. Cambridge (MA) : MIT Press.

DEUTSCH, R. 2019. *Superusers: Design Technology Specialists and the Future of Practice*. Londres : Routledge.

MENGES, A. ; AHLQUIST, S. 2011. *Computational Design Thinking*. Chichester : Wiley.

NEGROPONTE, N. 1969. « Towards a Humanism through Machines ». In MENGES, A. ; AHLQUIST, S. (éditeurs), *Computational Design Thinking*. Londres : AD Readers.

PICON, A. 2010. *Digital Culture in Architecture: An Introduction for the Design Professions*. Basel : Birkhäuser Architecture.

POTTMANN, H. ; ASPERL, A. ; HOFER, M. ; KILIAN, A. ; BENTLEY, D. 2007. *Architectural Geometry*. Exton (PA) : Bentley Institute Press.

SUSSKIND, R. E. ; SUSSKIND, D. 2015. *The Future of the Professions: How Technology Will Transform the Work of Human Experts*. Oxford : Oxford University Press.

SHELDEN, D. 2020. *The Disruptors: Technology-Driven Architect-Entrepreneurs*. Chichester : Wiley.

TERZIDIS, K. 2006. *Algorithmic Architecture*. Oxford : Architectural Press.