LIVRE BLANC

L'approche Model-Based Design pour les systèmes de contrôle spatiaux

MathWorks®

Imaginez que votre équipe développe le système d'alimentation d'un satellite. Ce système intègre une combinaison d'éléments physiques (par exemple, batterie, panneaux solaires), une logique de contrôle et des conditions externes (par exemple, température, rayonnements). Avant de commencer le design, il convient de répondre à quelques questions clés, par exemple :

- Comment dimensionner les batteries ?
- Que se passe-t-il si les exigences changent ?
- Comment optimiser le design pour garantir les performances souhaitées ?
- Comment tester le design de manière approfondie tout en minimisant les risques ?

Que vous développiez des commandes pour un système de vol, un robot industriel, une éolienne, une machine de production, un véhicule autonome, un excavateur ou un servomoteur électrique, si votre équipe écrit manuellement le code et utilise la capture des exigences à travers des documents, la seule façon de répondre à ces questions consiste à appliquer la méthode essai et erreur ou à réaliser des tests sur un prototype physique. Et si une seule exigence change, le système entier devra être recodé et recompilé, retardant ainsi le projet de plusieurs jours, voire plusieurs semaines.

En utilisant l'approche Model-Based Design avec MATLAB® et Simulink® au lieu d'avoir un code manuel ou des documents, vous créez un modèle de système qui englobe le modèle physique, les algorithmes de contrôle et l'environnement. Vous pouvez simuler le modèle à n'importe quel stade pour obtenir une vue instantanée du comportement du système et pour tester différents scénarios et analyses de compromis, sans risque, sans délai et sans devoir recourir à du hardware onéreux.

Ce livre blanc présente l'approche Model-Based Design et propose des astuces et des bonnes pratiques pour bien démarrer. En s'appuyant sur des exemples réels, il montre comment les équipes dans les différentes industries ont adopté l'approche Model-Based Design pour réduire le temps de développement, minimiser les problèmes d'intégration des composants et délivrer des produits de meilleure qualité.

Introduction à l'approche Model-Based Design

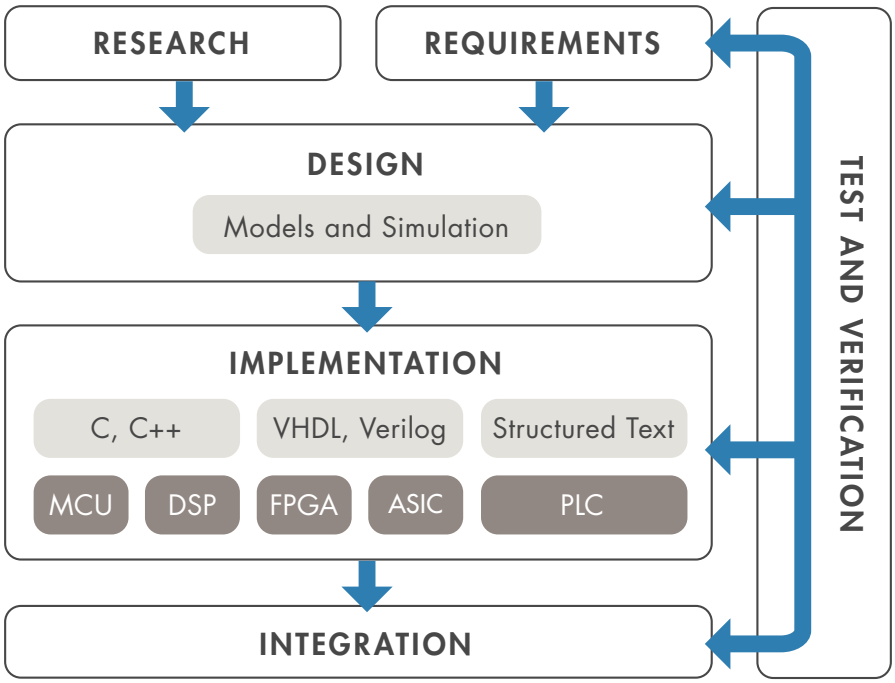
La meilleure façon de comprendre l'approche Model-Based Design consiste à la voir en action :

> Une équipe d'ingénieurs en aérospatiale entreprend de concevoir le système de guidage, de navigation et de contrôle d'un satellite. Parce qu'elle utilise l'approche Model-Based Design, elle commence par créer un modèle d'architecture à partir des exigences système ; dans ce cas, il s'agit du modèle de satellite lui-même. Un modèle de simulation/design est ensuite dérivé. Ce modèle basse fidélité de haut niveau inclut des parties du logiciel de contrôle qui sera embarqué dans le satellite, plus le système physique et l'environnement opérationnel.
>
> L'équipe effectue les tests initiaux du système et de l'intégration en simulant ce modèle de haut niveau dans le cadre de différents scénarios afin de vérifier que le système est correctement représenté et qu'il répond bien aux signaux d'entrée.
>
> Elle ajoute des détails au modèle, en testant et en vérifiant de manière continue le comportement au niveau du système par rapport aux spécifications. Si le système est volumineux et complexe, les ingénieurs peuvent développer et tester des composants individuels de manière indépendante, tout en continuant à les tester dans le cadre d'une simulation de système complet.

Au final, ils créent un modèle détaillé du système et de l'environnement dans lequel il fonctionne. Ce modèle capture les connaissances accumulées concernant le système (la propriété intellectuelle ou IP). Les ingénieurs génèrent du code automatiquement à partir du modèle des algorithmes de contrôle à des fins de test et de vérification du logiciel. Après avoir réalisé les tests Hardware-in-the-Loop, ils téléchargent le code généré sur du hardware de production afin de le tester dans un système final réel.

Comme le montre ce scénario, l'approche Model-Based Design utilise les mêmes éléments que les workflows de développement traditionnels, mais avec deux différences essentielles :

- La plupart des étapes chronophages ou sujettes aux erreurs dans le workflow (par exemple, la génération de code) sont automatisées.
- Un modèle de système se trouve au cœur du développement, de la capture des exigences au design, à l'implémentation et aux tests.



Workflow de l'approche Model-Based Design.

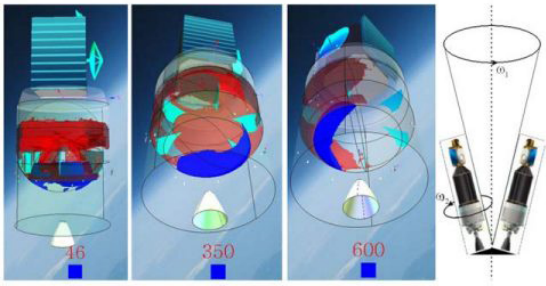
Ingénierie système, capture et gestion des exigences

Dans un workflow traditionnel, où les exigences sont capturées dans des documents, le transfert peut générer des erreurs et des retards. Souvent, les ingénieurs qui créent les documents de design ou les exigences ne sont pas ceux qui conçoivent le système. Les exigences peuvent être mal transmises, ce qui montre une absence de communication claire et cohérente entre les deux équipes.

Dans le cadre de l'approche Model-Based Design, vous créez, analysez et gérez les exigences dans votre modèle Simulink. Vous pouvez créer des exigences textuelles enrichies avec des attributs personnalisés, puis les lier à des designs, du code et des tests. Il est également possible d'importer et de synchroniser les exigences à partir de sources externes telles que des outils de gestion des exigences. Lorsqu'une exigence liée au design est modifiée, vous recevez une notification automatique. Ainsi, vous pouvez identifier la partie du design ou du test directement concernée par la modification, et prendre les mesures appropriées pour la traiter. Vous pouvez définir, analyser et spécifier des architectures et des compositions pour les systèmes et les composants logiciels.

En outre, les ingénieurs système peuvent utiliser MATLAB et Simulink pour réaliser des analyses dynamiques. Ils utilisent des modèles exécutables d'engins spatiaux et de systèmes au sol multi-domaines pour la validation et la vérification des exigences, et sont donc en mesure de communiquer des informations concernant le comportement et les performances au niveau du système qu'une analyse statique, à elle seule, est incapable de fournir. Cette approche leur permet également de tracer les exigences à partir des spécifications de haut niveau, de surveiller leur implémentation détaillée dans le design et d'assurer leur suivi dans le code source généré automatiquement. Ils peuvent mapper les exigences à des cas de tests et mesurer automatiquement leur couverture à mesure que les cas de test sont exécutés.

Étude de cas : *ESA et Airbus Defence and Space*



Mouvement d'ergols dans les étages supérieurs en rotation à 46, 350 et 600 secondes. La distribution après 350 secondes devient inégale.

« L'approche Model-Based Design nous a permis de créer un cadre pour concevoir des contrôleurs de vol avec des algorithmes de pointe, robustes de design de contrôle, créer des modèles physiques multi-domaines, mettre au point le design grâce à l'optimisation et générer le code pour les tests en mode PIL sur le hardware cible, le tout dans environnement unique. »

— Hans Strauch, Airbus D&S

Lorsqu'un lanceur de l'Agence spatiale européenne (ESA), tel qu'Ariane 5 ou Vega, met en orbite la charge utile de son satellite, le système de contrôle d'attitude (ACS) prend le contrôle, oriente la charge utile et commande la séparation de l'étage supérieur du lanceur. Outre l'orientation du satellite, l'ACS doit identifier et gérer les problèmes liés au processus de séparation, au ballottement d'ergols et à de nombreux dysfonctionnements potentiels du hardware.

L'ESA et Airbus souhaitaient simuler des défaillances de séparation avec un modèle physique afin de tester la capacité du contrôleur à détecter les défaillances et à mener des actions correctives. Ils devaient également simuler le ballottement d'ergols, les fuites dans les conduites, les vannes bloquées et toute une série d'autres défaillances. Ils voulaient par ailleurs exécuter des optimisations afin d'identifier les pires performances du système en cas de défaillance.

Les ingénieurs ont cherché à tester leurs algorithmes de contrôle sur le hardware de l'ordinateur de vol le plus tôt possible dans le développement. Les algorithmes de contrôle étant de plus en plus complexes, ils repoussent les limites des performances des processeurs et des autres ressources informatiques. Les ingénieurs devaient vérifier les performances des algorithmes et l'utilisation des ressources sur un ordinateur de vol représentatif pendant le design du contrôleur, au moment où il serait le plus facile de remédier aux problèmes.

Les ingénieurs de l'ESA et d'Airbus ont utilisé l'approche Model-Based Design avec MATLAB, Simulink et une combinaison d'outils de génération de code, de modélisation physique, de design de la logique de contrôle, ainsi que de vérification et validation pour créer l'USACDF (Upper Stage Attitude Control and Design Framework). Ce framework, qui permet la simulation en boucle fermée et la vérification des algorithmes de contrôle avec des modèles physiques, est utilisé pour créer des démonstrateurs de concepts complexes d'opérations de mission de services en orbites.

Design

Dans une approche traditionnelle, chaque idée de design doit être codée et testée sur un prototype physique. Par conséquent, seul un nombre limité d'idées et de scénarios de design peuvent être explorés, car chaque itération de test augmente le coût et le temps de développement. Dans le domaine spatial, les choix de design sont particulièrement essentiels pour garantir que le système physique utilisé pendant la mission est adapté.

Dans le cadre de l'approche Model-Based Design, le nombre d'idées pouvant être explorées est quasiment illimité. Les exigences, les composants système, la propriété intellectuelle (IP) et les scénarios de test sont tous capturés dans votre modèle. Le fait de pouvoir simuler le modèle permet alors d'étudier les problèmes et les questions liés au design bien avant de construire un hardware onéreux. Vous pouvez rapidement évaluer plusieurs idées de design, explorer les compromis et voir comment chaque modification du design impacte le système.

Étude de cas : *Tessella*

Présentation artistique de Solar Orbiter.

« Nous avons vu les avantages de l'approche Model-Based Design sur plusieurs projets précédents. Sur ce projet, MATLAB et Simulink nous ont permis de créer une spécification détaillée qui a minimisé l'écart entre les algorithmes prototypes que nous avons développés, réglés et testés dans Simulink et l'implémentation logicielle finale. »

— Andrew Pollard, Tessella

La mission Solar Orbiter du programme Cosmic Vision de l'Agence spatiale européenne est destinée à répondre à des questions fondamentales sur le fonctionnement du système solaire et les origines de l'univers.
Le sous-système de contrôle d'attitude et d'orbite (AOCS) sera chargé de maintenir l'engin spatial et son bouclier solaire orientés vers le soleil et de conserver une attitude précise afin d'optimiser l'exactitude des instruments.

L'AOCS doit constamment ajuster l'attitude de l'engin spatial Solar Orbiter afin que le bouclier solaire assure une protection maximale lorsque le vaisseau passe près du soleil. Pour des raisons de sécurité, l'AOCS ne peut permettre à aucun moment à l'engin spatial de se dépointer de plus de 6,5 degrés par rapport au soleil, même après une défaillance. Pendant les observations scientifiques, la stabilité du pointage doit être de quelques dixièmes de seconde d'arc.

En plus de répondre à ces exigences, l'AOCS devait tenir compte des couples perturbateurs dus à la pression de radiation solaire, au gradient de gravité et aux forces aérodynamiques.

La structure physique de l'engin spatial a compliqué le design de l'AOCS. Le bouclier solaire a contribué à une distribution inhabituelle de la masse qui a rendu la stabilité difficile. De plus, les nombreux appendices flexibles, y compris les panneaux solaires, ont rendu l'ensemble de la structure sensible à la résonance.

Les ingénieurs de Tessella ont utilisé l'approche Model-Based Design pour concevoir, modéliser, simuler et effectuer le réglage préliminaire des algorithmes, et prouver leur aptitude au codage et à la vérification formels. En utilisant Simulink, l'équipe a modélisé les systèmes d'actionnement de l'engin spatial, notamment ses quatre roues de réaction et ses propulseurs à propulsion chimique. Afin de fournir un contrôle précis des propulseurs, l'équipe a développé et modélisé un algorithme de commande d'actionneur utilisant la modulation de largeur d'impulsion.

Systèmes d'alimentation

Les ingénieurs en systèmes d'alimentation utilisent MATLAB et Simulink afin de réaliser des simulations pour l'analyse du profil d'alimentation de la mission, prédire les impacts du vieillissement des batteries sur le système et réaliser le design détaillé des composants électriques, tels que les convertisseurs DC-DC.

Ils peuvent rapidement modéliser des composants et systèmes électriques, tels que des panneaux solaires et des régulateurs de tension, à l'aide de blocs prédéfinis ; ils peuvent également créer des blocs personnalisés là où le design le nécessite. Les ingénieurs peuvent ensuite simuler le modèle pour résoudre les systèmes d'équations complexes sous-jacents, sans avoir à écrire de code de bas niveau et visualiser immédiatement les résultats. Ils peuvent aussi inclure des effets thermiques et d'attitude dans leurs modèles pour effectuer des simulations multi-domaines au sein d'un environnement unique.

Étude de cas : *Lockheed Martin*

Engin spatial Orion de la NASA.

« Avec Simscape Electrical, nous avons créé un modèle de système d'alimentation intégré faisant le lien entre les domaines électrique et thermique, ce qui nous permet d'avoir une vue d'ensemble lors de nos simulations au niveau mission. Si nous devons modéliser les moteurs qui font tourner les panneaux solaires, nous disposons alors également de la capacité d'intégrer ces composants mécaniques. »

— Hector Hernandez, Lockheed Martin

Les ingénieurs de Lockheed Martin ont développé un modèle de système d'alimentation pour le programme Constellation de la NASA en utilisant Microsoft® Excel® et Visual Basic® for Applications (VBA). Cependant, lorsque les ingénieurs ont essayé d'examiner de plus en plus de cas de test, le modèle ralentissait et cessait de fonctionner parfois. Comme il s'agissait d'un modèle d'alimentation à un seul nœud, l'équipe ne pouvait pas l'utiliser pour évaluer les chutes de tension dans l'ensemble du système ou pour évaluer les exigences de qualité de l'alimentation.

Les ingénieurs de Lockheed Martin ont utilisé Simulink et Simscape Electrical pour modéliser et simuler le système d'alimentation de l'engin spatial Orion. Pour calculer une courbe courant-tension (IV) précise pour le panneau solaire, le modèle prend en compte le nombre de cellules par aile, les angles de pointage du panneau solaire, l'ombrage et les effets thermiques tels que l'énergie solaire, l'infrarouge et l'albédo planétaire. Les facteurs de dégradation des performances, comme le rayonnement, la contamination et les collisions avec des micrométéorites et des débris orbitaux ont également été pris en compte.

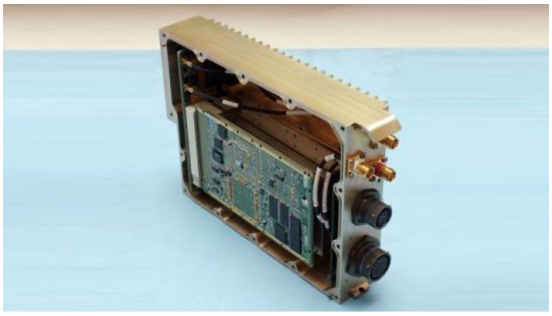
Le modèle de la batterie lithium-ion tient compte des effets d'hystérésis de la batterie, de la température et de son état de charge. L'équipe a également créé un bloc de charge électrique personnalisé afin de simuler les charges variables générées par les systèmes de propulsion, de guidage et autres de l'engin spatial, qui s'allument et s'éteignent tout au long de la mission. Le modèle de système d'alimentation intégré a permis à l'équipe d'effectuer des simulations pour divers profils de mission. Elle a surveillé les changements de mission et de charge pour évaluer l'impact de l'alimentation au cours du développement, et a simulé des scénarios de défaillance, notamment des pannes de batterie, des interrupteurs bloqués et des pannes de panneaux solaires pendant les phases d'ascension et d'orbite. Les résultats ont été utilisés pour guider le développement des opportunités de mission et des protocoles pour les cas de défaillance.

Systèmes de communications

Les ingénieurs en systèmes de communications utilisent MATLAB et Simulink comme environnement de design commun pour développer, analyser et implémenter les systèmes de communications des véhicules spatiaux. Ils peuvent utiliser MATLAB et Simulink pour prototyper les éléments de la chaîne du signal, tels que les éléments numériques, RF et d'antenne. Les travaux de plusieurs équipes peuvent ensuite être combinés en un modèle exécutable au niveau système.

Les ingénieurs peuvent rapidement explorer les imperfections au niveau système et étudier différents scénarios de simulation difficiles à produire en laboratoire. À mesure que le design évolue, les ingénieurs peuvent automatiquement générer du code C pour les processeurs embarqués ou du code HDL pour les FPGA.

Étude de cas : *BAE Systems*



Carte personnalisée utilisée dans le workflow de design traditionnel.

« Il a fallu 645 heures à un ingénieur ayant des années d'expérience en codage VHDL pour coder manuellement une forme d'onde SDR entièrement fonctionnelle à partir de notre flux de design traditionnel. Un deuxième ingénieur, moins expérimenté, a eu besoin d'à peine 46 heures pour terminer ce même projet en ayant recours à Simulink et Xilinx System Generator. »

— Dr David Haessig, BAE Systems

À la pointe de la technologie SDR depuis de nombreuses années, BAE Systems a toujours utilisé un flux de design reposant sur le codage manuel de FPGA dans VHDL®. Cependant, l'entreprise a récemment saisi l'opportunité d'évaluer cette démarche par rapport à l'approche Model-Based Design en utilisant les outils MathWorks et Xilinx®. Alors qu'ils exécutaient deux projets de développement de formes d'onde SDR en parallèle, les ingénieurs de BAE Systems ont constaté que Simulink® et Xilinx System Generator permettaient de considérablement réduire les temps de développement.

BAE Systems a été chargé de concevoir une forme d'onde pour les communications par satellite conforme à la norme militaire (MIL-STD-188-165A) en vue d'une implémentation dans une radio C4ISR (Command, Control, Communications, Computers, Intelligence, Surveillance, and Reconnaissance). L'entreprise cherchait par ailleurs à évaluer un nouveau flux de design destiné à réduire ses temps de développement.

En collaboration avec Xilinx, BAE Systems a appliqué l'approche Model-Based Design à l'aide de Simulink et de Xilinx System Generator, et a été capable de concevoir et déployer une forme d'onde SDR MIL-STD-188 dix fois plus rapidement qu'avec son approche de codage manuel. Compte tenu du succès de l'effort initial sur ce projet, BAE Systems a lancé un effort conjoint avec MathWorks, Virginia Tech, Xilinx et Zeligsoft pour améliorer la portabilité des formes d'onde. Ce groupe développe aujourd'hui une interface permettant au code généré par Simulink Coder™ ou Xilinx System Generator d'être directement incorporé dans des radios SCA (Software Communications Architecture).

Guidage, navigation et contrôle

Grâce à MATLAB et Simulink, les ingénieurs peuvent tester leurs algorithmes de contrôle en utilisant des modèles physiques avant l'implémentation, ce qui leur permet de réaliser des designs complexes sans recourir à des prototypes onéreux. Ils peuvent concevoir plusieurs configurations physiques, comme l'architecture commune d'un bus pour le design d'un satellite. Dans un environnement unique, les ingénieurs travaillent sur :

- La création et le partage de modèles GNC
- L'intégration et la simulation des effets au niveau système des modifications du design des systèmes mécaniques et de contrôle
- La réutilisation automatique du code généré pour l'avionique et des cas de test
- L'intégration de nouveaux designs aux designs et outils existants

Étude de cas : *Systèmes spatiaux Lockheed Martin*

Le télescope spatial IRIS.

Le télescope IRIS (Interface Region Imaging Spectrograph) est actuellement en orbite autour de la Terre, où il capture des spectres ultraviolets et des images haute résolution du soleil. Ces images aideront les scientifiques à mieux comprendre les flux d'énergie et de plasma dans les niveaux les plus bas de l'atmosphère solaire.

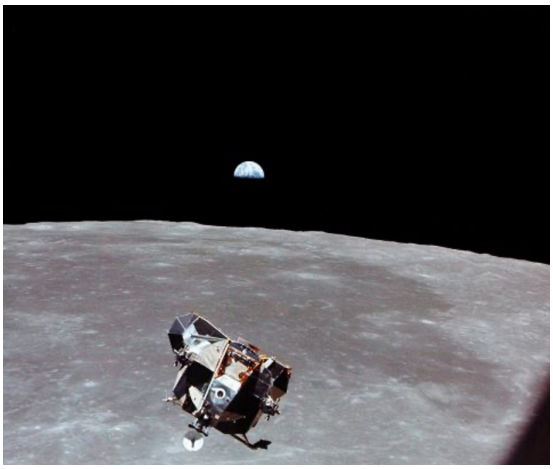
> « Une équipe de quatre ingénieurs a conçu, intégré et testé le système GNC en seulement 23 mois. Nous avons pu gagner en efficacité en utilisant les mêmes outils pour l'analyse et le développement du code. Cette méthode nous a permis de générer 20 000 lignes de code exemptes de défauts. Pour nous, c'est un argument de poids en faveur de l'approche Model-Based Design. »
>
> — *Vincentz Knagenhjelm, ingénieur GNC, Lockheed Martin Space Systems*

Pour des projets similaires réalisés par le passé, les ingénieurs de Lockheed Martin ont élaboré des documents détaillés relatifs au design d'algorithmes, dont certains comptaient plus de 1000 pages. Les programmeurs rédigeaient ensuite leur code à la main, selon leur interprétation de ces documents. L'ensemble du processus était lent, et des défauts étaient parfois introduits lors du codage manuel. Ne disposant que de 23 mois pour assurer le design, l'intégration et les tests du logiciel, l'équipe en charge du projet a été contrainte d'accélérer considérablement le processus de livraison du logiciel.

Les ingénieurs de Lockheed Martin sont parvenus à réduire le temps de développement du logiciel de guidage, de navigation et de contrôle (GNC) du télescope IRIS en adoptant l'approche Model-Based Design. Avec MATLAB et Simulink, ils ont pu développer un modèle de base du système de contrôle pour analyser les performances de pointage et la précision avec laquelle l'engin spatial peut être réorienté.

Ils ont ensuite vérifié le design initial du logiciel de GNC en exécutant des simulations en boucle fermée sur le modèle physique et en menant une analyse de la couverture du modèle sur les simulations à l'aide de Simulink Coverage™. Les ingénieurs se sont servis de Embedded Coder pour générer du code C pour chaque composant, auquel ils ont ajouté une petite quantité de code créé à la main pour assurer l'interopérabilité avec un microprocesseur Moog Broad Reach Engineering résistant aux radiations et son logiciel d'exécution. Enfin, grâce à une interface utilisateur personnalisée MATLAB, l'équipe a expérimenté une variété de cas de test Simulink pour chaque unité du logiciel de vol GNC.

Étude de cas : *L'alunissage de Apollo 11 : le design d'engins spatiaux d'hier à aujourd'hui*



Source de l'image : NASA.

Apollo 11, transportant les astronautes Neil Armstrong et Buzz Aldrin, a atterri sur la Lune il y a plus de 50 ans. Afin de commémorer cet événement historique et de mettre en lumière les programmes qui travaillent actuellement sur les prochains alunissages, nous revenons sur le récit de Richard J. Gran consacré au design du pilote automatique numérique du module lunaire, publié en 1999. Dans son article, Dr Richard décrit la méthode qu'il utilisait dans les années 1960 et la compare à la manière dont l'approche Model-Based Design avec MATLAB et Simulink pourrait être utilisée pour concevoir des systèmes de GNC en 1999. Au cours des 20 années qui se sont écoulées depuis la publication de cet article, l'approche Model-Based Design a considérablement évolué. Pour illustrer cette évolution, nous exposons comment MATLAB et Simulink peuvent être appliqués au design de systèmes GNC aujourd'hui.

Lisez cet article pour en savoir plus, et découvrez le modèle de système Simulink que Richard Gran a développé en revisitant le pilote automatique numérique du module lunaire.

Traitement d'images et de données et Computer Vision

Les systèmes de détection basée sur la vision permettent d'augmenter les niveaux d'autonomie et de précision de la navigation pour les missions spatiales. Grâce à la navigation relative et optique de haute précision, les systèmes de détection basés sur la vision jouent un rôle essentiel dans les domaines suivants :

- Opérations de proximité et de rendez-vous (Rendezvous and Proximity Operations ou RPO)
- Entrée, descente et atterrissage (Entry, Descent, and Landing ou EDL)
- Exploration robotique du système solaire

Les exigences toujours plus ambitieuses des missions d'aujourd'hui, ainsi qu'un secteur privé énergique et innovant, motivent une forte augmentation de la recherche sur les techniques de détection et de perception basées sur la vision qui utilisent l'intelligence artificielle avec le Machine Learning.

Consultez ce livre blanc pour en savoir plus sur :

- La détection basée sur la vision facilite l'autonomie des engins spatiaux
- Comment l'essor du Machine Learning dans l'industrie aérospatiale influence l'intelligence artificielle (IA) des engins spatiaux
- Comment vous pouvez utiliser les routines MATLAB et Simulink pour vous concentrer sur le design de plus haut niveau

Étude de cas : *Cornell University*

Dispositif d'analyse acoustique utilisé par le Bioacoustics Research Program (BRP) pour recueillir des données sur les grandes baleines à fanons et autres mammifères marins. Source de la photo : Dimitri Ponirakis.

« La fonction de calcul haute performance de MATLAB nous permet de traiter de larges volumes de données jamais analysés auparavant. Nos découvertes nous éclairent sur l'impact des activités humaines sur la santé des écosystèmes et nous permettent de prendre des décisions responsables sur les actions de l'Homme sur terre et dans les océans. »

— Dr Christopher Clark, Cornell University

Depuis plus de 30 ans, les scientifiques étudient les populations animales locales en enregistrant les sons émis dans les océans, les jungles, les forêts et autres environnements naturels. Ils exploitent ces résultats pour évaluer les effets du bruit d'origine humaine sur les milieux naturels, surveiller les populations animales menacées et étudier la communication entre animaux. Les systèmes de surveillance acoustique passive enregistrent les sons en continu, ce qui représente des téraoctets de données. Bien souvent, les scientifiques sont dans l'incapacité de traiter ne serait-ce qu'1 % de ces données, car ils ne disposent pas des algorithmes avancés et de la capacité de traitement nécessaires.

Ils sont également confrontés à la variabilité des sons entre les individus d'une même espèce. Les données bruitées et la variabilité entraînent une augmentation du nombre de faux positifs et négatifs, réduisant ainsi la précision des algorithmes de détection. Le traitement des centaines de téraoctets de données recueillies par BRP représente un autre défi. Un projet classique exige le traitement de plusieurs années de données acoustiques brutes (jusqu'à 10 To) enregistrées sur plusieurs canaux. Chaque canal peut capturer des centaines de millions d'événements, c'est-à-dire des sons qui se détachent lorsque les données sont visualisées sous forme de spectrogramme. Les algorithmes testés sur de petits échantillons de haute qualité sont souvent beaucoup moins précis lorsqu'ils sont appliqués à des jeux de données plus importants et plus bruités.

Les data scientists de BRP ont utilisé MATLAB pour développer un logiciel de calcul haute performance (HPC) dédié au traitement automatique des données acoustiques. La première étape des projets de détection et classification consiste à collecter des enregistrements audio de l'animal qu'ils souhaitent écouter, des enregistrements du bruit de fond dans l'environnement de cet animal, et des fichiers MAT de données acoustiques archivées. À l'aide de MATLAB, ils développent de nouveaux algorithmes ou affinent les algorithmes existants pour détecter dans les données archivées des séquences audio similaires à celles du catalogue d'extraits. L'équipe BRP a développé une interface MATLAB qui permet aux chercheurs de spécifier les algorithmes, les jeux de données et le nombre de processeurs à utiliser. Outre les algorithmes de détection et de classification, BRP utilise MATLAB pour l'analyse du bruit et la modélisation acoustique, dans laquelle les effets de dispersion temporelle et fréquentielle des environnements marins ou terrestres sont enregistrés et simulés.

Génération de code

Dans un workflow traditionnel, le code embarqué doit être écrit à la main à partir des modèles des systèmes ou en partant de zéro. Les ingénieurs software écrivent des algorithmes de contrôle en s'appuyant sur les spécifications écrites par les ingénieurs en systèmes de contrôle. Chacune des étapes de ce processus (écriture des spécifications, codage manuel des algorithmes et débuggage du code manuel) peut être à la fois chronophage et sujette aux erreurs.

Avec l'approche Model-Based Design, au lieu d'écrire des milliers de lignes de code à la main, vous générez le code directement à partir de votre modèle, et celui-ci sert de passerelle entre les ingénieurs software et les ingénieurs en systèmes de contrôle. Le code généré peut être utilisé pour le prototypage rapide ou la production.

Le prototypage rapide offre un moyen rapide et peu coûteux de tester des algorithmes sur du hardware en temps réel et de réaliser des itérations de design en quelques minutes au lieu de plusieurs semaines auparavant. Vous pouvez utiliser du hardware de prototypage ou votre ECU de production. Associé aux modèles de design, ce hardware de prototypage rapide vous permet de réaliser des tests Hardware-in-the-Loop et d'autres activités de test et de vérification afin de valider les designs hardware et logiciel avant la production.

La génération de code de production convertit votre modèle en code réel qui sera implémenté sur le système de production embarqué. Le code généré peut être optimisé pour des architectures de processeurs spécifiques et intégré à du code manuel existant.

Étude de cas : *Swedish Space Corporation*

Représentation artistique de SMART-1 se dirigeant vers la Lune.

« Nous avons réussi à développer l'AOCS de la sonde SMART-1 dans un délai très court et avec un budget très restreint. Les outils MathWorks de simulation et de génération de code pour l'avionique ont joué un rôle clé dans cette réussite et serviront de base aux futurs programmes de satellites, tels que Prisma. »

— Per Bodin, Swedish Space Corporation

Swedish Space Corporation (SSC) a développé le système de contrôle d'attitude et d'orbite (AOCS) de la sonde SMART-1 (Small Missions for Advanced Research and Technology) en utilisant du code pour l'avionique automatiquement généré. L'AOCS oriente l'engin spatial afin d'optimiser la poussée vectorielle, le pointage des instruments scientifiques et l'illumination des panneaux solaires. Il contrôle également l'alignement du vecteur de poussée de la propulsion électrique à l'intérieur du vaisseau pendant les phases de transfert et de descente lunaire, tout en fournissant un système avancé de détection, d'isolation et de reprise après panne (FDIR).

SSC devait développer un AOCS en respectant un profil de mission low-cost, des normes de développement logiciel strictes et un cycle de développement de moins de deux ans. De plus, étant donné que l'AOCS devait fonctionner dans un environnement spatial difficile, marqué par des radiations intenses, une gravité très basse et d'autres effets ne pouvant pas être testés en laboratoire ou sur Terre, SCC devait impérativement pouvoir compter sur des capacités de test rigoureuses afin de s'assurer que le système fonctionne correctement en vol.

SCC a mis en place un nouveau processus de développement basé sur les outils de MathWorks pour l'approche Model-Based Design afin de modéliser, simuler, générer automatiquement du code et tester le logiciel AOCS embarqué. Les ingénieurs de l'entreprise ont développé des modèles de simulation précis capables de prédire le comportement du système et de créer des cas de test exhaustifs pour le système et le logiciel, en conformité avec la norme de développement logiciel PSS-05 de l'Agence spatiale européenne. SCC a également procédé à des tests du système logiciel sur un environnement de simulation hardware temps réel et a analysé les résultats avec MATLAB. La vérification de l'AOCS au niveau système s'est faite à l'aide de l'engin spatial intégré. Ces tests comprenaient des essais en boucle ouverte et fermée au centre ESTEC (European Space Research and Technology Centre) aux Pays-Bas.

Tests et vérification

Dans un workflow de développement traditionnel, les tests et la vérification interviennent généralement tard dans le processus, rendant ainsi difficile l'identification et la correction des erreurs introduites pendant les phases de design et de codage.

Dans l'approche Model-Based Design, les tests et la vérification se produisent tout au long du cycle de développement, en commençant par la modélisation des exigences et des spécifications et en continuant par le design, la génération de code et l'intégration. Vous pouvez créer des exigences dans votre modèle et assurer leur traçabilité jusqu'au design, aux tests et au code. Il existe des méthodes formelles qui permettent de prouver que votre design répond aux exigences. Vous pouvez produire des rapports et des artefacts et certifier votre logiciel pour des normes de sécurité fonctionnelle telles que ***NPR 7150.2*** (NASA Software Engineering Requirements) et ***ECSS-E-40*** (European Cooperation for Space Standardization, logiciels d'ingénierie spatiale).

Étude de cas : *OHB*

Simulation de stratégies GNC pour le vol en formation autonome avancée.

« Ces modèles sont devenus des modèles de vol complets, que nous avons vérifiés dans des simulations en boucle fermée avec un modèle physique Simulink. À partir de là, générer le code pour l'avionique a pu être réalisé en un clic. »

— Ron Noteborn, OHB

Les missions spatiales planifiées reposent souvent sur le vol en formation autonome, qui consiste pour un engin spatial à s'approcher ou à voler à côté d'un autre. Le projet Prisma, mené par OHB AG (OHB) en collaboration avec les agences spatiales française et allemande ainsi que l'Université technique du Danemark, teste et valide les stratégies GNC pour les vols autonomes avancés en formation.

Les ingénieurs d'OHB ont adopté l'approche Model-Based Design pour développer des algorithmes GNC, exécuter des simulations temps réel en boucle fermée au niveau système et générer le code pour l'avionique destiné aux deux satellites de Prisma, Mango et Tango.

OHB a divisé le design du GNC en opérations de vol en formation, de rendez-vous et de proximité. L'entreprise a ensuite testé et analysé des idées d'algorithmes dans MATLAB avant de les modéliser pour les vérifier dans une simulation en boucle fermée.

Les ingénieurs ont généré du code à partir de leurs modèles GNC et physiques, déployé le code des modèles physiques vers Simulink Real-Time™ et compilé le code GNC pour le processeur cible embarqué LEON2. OHB a ensuite effectué des tests Hardware-in-the-Loop (HIL) du système Simulink Real-Time et du contrôleur LEON2 afin de vérifier le fonctionnement temps réel des algorithmes.

Pour bien démarrer

Même si votre équipe perçoit les avantages du passage à l'approche Model-Based Design, elle peut également s'inquiéter des risques et des difficultés (d'ordre organisationnel, logistique et technique) qui peuvent en découler. Cette section répond aux questions les plus fréquemment posées par les équipes d'ingénierie qui envisagent d'adopter l'approche Model-Based Design. Elle propose des astuces et des bonnes pratiques qui ont aidé bon nombre de ces équipes à gérer la transition.

Q. En quoi les ingénieurs sont-ils affectés par l'adoption de l'approche Model-Based Design ?

R. L'approche Model-Based Design ne remplace pas l'expertise d'ingénierie en design de systèmes de contrôle et d'architecture logicielle. Avec cette approche, les rôles des ingénieurs en contrôle passent de l'établissement d'exigences sur papier à l'établissement d'exigences exécutables sous forme de modèles et de code. Les ingénieurs software passent moins de temps à coder le logiciel applicatif et davantage à modéliser l'architecture, à coder le système d'exploitation, le pilote du périphérique et d'autres logiciels de plateforme, et à effectuer l'intégration des systèmes. Les ingénieurs en contrôle et en software influencent le design au niveau système dès les premières étapes du processus de développement.

Q. Qu'advient-il de notre code existant ?

R. Il peut faire partie du design ; votre modèle de système peut contenir à la fois des composants intrinsèquement modélisés et des composants existants. Cela signifie que vous pouvez intégrer progressivement les composants existants tout en continuant à effectuer la simulation, la vérification et la génération de code du système.

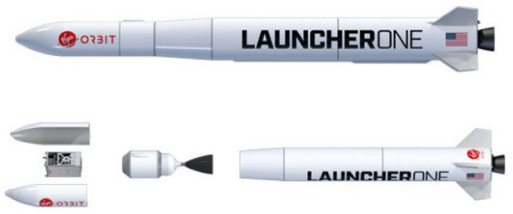
Q. Existe-t-il une méthode recommandée dans l'adoption de l'approche Model-Based Design ?

R. Essayer de nouvelles approches et de nouveaux outils de design implique toujours un risque. Les équipes, qui ont réussi, ont atténué ce risque en introduisant progressivement l'approche Model-Based Design, en prenant des mesures ciblées qui font avancer le projet sans le ralentir. Peu importe leur taille, les entreprises commencent souvent à adopter l'approche Model-Based Design au sein de petits groupes. Elles commencent généralement par un seul projet pour obtenir un succès rapide et s'appuient sur cette première réussite. Après avoir acquis de l'expérience, elles déploient cette approche au niveau du département de sorte que les modèles soient désormais un élément indispensable au développement de tous les systèmes embarqués du groupe.

Les quatre bonnes pratiques suivantes ont bien fonctionné pour de nombreuses équipes :

- **Tester sur une petite partie du projet.** Un bon point de départ consiste à sélectionner une nouvelle partie du système embarqué, à créer un modèle du comportement du logiciel et à générer du code à partir du modèle. Un membre de l'équipe peut apporter ce petit changement avec un investissement minimal dans l'apprentissage de nouveaux outils et techniques. Vous pouvez utiliser les résultats pour démontrer certains des avantages clés de l'approche Model-Based Design :
  - Possibilité de générer automatiquement du code de haute qualité.
  - Le code correspond au comportement du modèle.
  - En simulant un modèle, vous pouvez éliminer les bugs dans les algorithmes beaucoup plus simplement et en obtenant davantage d'informations pertinentes qu'en testant le code C sur un desktop.
- **Tirer parti de vos premières réussites en matière de modélisation en ajoutant la simulation au niveau système.** Comme l'ont montré les précédentes sections de ce document, vous pouvez utiliser la simulation de système pour valider les exigences, étudier les questions de design et effectuer des tests et vérifications dès le début du processus de développement. Le modèle de système ne doit pas forcément être de haute fidélité ; il doit simplement être suffisamment détaillé pour s'assurer que les bonnes unités sont utilisées pour les signaux d'interfaçage, que ces derniers sont connectés aux bons canaux, et que le comportement dynamique du système est capturé. Les résultats de la simulation vous donnent une première idée du futur comportement du système physique et du contrôleur.
- **Utiliser des modèles pour résoudre des problèmes de design spécifiques.** Votre équipe peut obtenir des avantages ciblés sans même avoir à développer des modèles à l'échelle réelle du système physique, de l'environnement et de l'algorithme. Supposons par exemple que votre équipe doive sélectionner des paramètres pour un solénoïde utilisé pour l'activation. Elle peut développer un modèle simple qui trace un « volume de contrôle » conceptuel autour du solénoïde, y compris ce qui le commande et ce sur quoi il agit. L'équipe peut tester différentes conditions de fonctionnement extrêmes et dériver les paramètres de base sans avoir à dériver les équations. Ce modèle peut être stocké pour être utilisé dans le cadre d'un autre problème de design ou d'un futur projet.
- **Commencer avec les éléments fondamentaux de l'approche Model-Based Design.** Les avantages immédiats du Model-Based Design incluent la possibilité de créer des modèles de composants et de systèmes, d'utiliser des simulations pour tester et valider des designs, et de générer automatiquement du code C à des fins de prototypage et de test. Par la suite, vous pouvez envisager d'utiliser des outils et pratiques avancés et de mettre en place des directives de modélisation, la vérification automatique de la conformité, la traçabilité des exigences et l'automatisation de la compilation du logiciel.

Étude de cas : *Virgin Orbit*



Le véhicule LauncherOne de Virgin Orbit assemblé (en haut), avec vue éclatée montrant la coiffe, la charge utile, ainsi que le premier et le second étages (en bas).

« MATLAB et Simulink nous ont permis d'économiser environ 90 % des coûts par rapport à l'autre solution que nous avions envisagée. Ils nous ont également offert la souplesse de codage nécessaire pour développer nos propres modules et pleinement appréhender les hypothèses formulées, ce qui est essentiel pour communiquer les résultats aux autres équipes. »

— Patrick Harvey, Virgin Orbit

LauncherOne est le lanceur spatial à deux étages de Virgin Orbit conçu pour envoyer de petits satellites en orbite terrestre basse. Afin de réduire les coûts et de bénéficier d'une plus grande flexibilité en matière de lieux de lancement, LauncherOne est conçu pour être largué d'un avion porteur 747-400 en vol. Chaque mission comportera plusieurs événements de séparation décisifs, notamment la séparation du LauncherOne de son avion porteur, le premier étage du second, la coiffe du second étage et la charge utile du satellite du second étage.

Lorsque le design structurel du LauncherOne était encore en cours de développement, l'équipe a dû prendre en compte un certain nombre d'inconnues dans l'analyse des événements de séparation, notamment les propriétés physiques de chaque composant ainsi que les forces et les caractéristiques de synchronisation des poussoirs pneumatiques et à ressort utilisés pour initier les séparations. L'équipe a dû réaliser des milliers de simulations de Monte Carlo en faisant varier les valeurs de ces facteurs incertains afin de déterminer si une combinaison spécifique de paramètres risquait de provoquer une collision.

Les ingénieurs de Virgin Orbit ont modélisé et simulé les événements de séparation de l'étage et de la charge utile du LauncherOne avec Simulink et Simscape Multibody, en utilisant Parallel Computing Toolbox™ pour exécuter les simulations en parallèle sur des processeurs multicœurs. En travaillant dans Simulink avec Simscape Multibody, l'équipe en charge du projet a conçu un modèle préliminaire composé de formes 3D basiques, telles que des sphères, des cônes et des cylindres. Cette même équipe travaille actuellement à la simulation de l'événement de séparation par un système à air, lequel intégrera un modèle des forces et effets aérodynamiques. Elle perfectionne par ailleurs son modèle sur la base des résultats des essais au sol du hardware de vol en vue du lancement inaugural de l'engin spatial.

Résumé

Un modèle de système se trouve au cœur du développement, de la capture des exigences au design, à l'implémentation et aux tests. C'est l'essence de l'approche Model-Based Design. Avec ce modèle de système, vous pouvez :

- Lier les designs directement aux exigences
- Collaborer dans un environnement de design partagé
- Simuler différents scénarios de simulation
- Faire des choix judicieux en matière de design basés sur la simulation
- Optimiser les performances au niveau système
- Générer automatiquement le code du logiciel embarqué, les rapports et la documentation
- Détecter les erreurs plus tôt en procédant plus tôt aux tests

Outils pour l'approche Model-Based Design

Produits fondamentaux

MATLAB

Analyser des données, développer des algorithmes et créer des modèles mathématiques

Simulink

Modéliser et simuler des systèmes embarqués

Capture et gestion des exigences

Simulink Requirements

Créer, gérer et tracer des exigences dans les modèles, le code généré et les cas de test

System Composer

Concevoir et analyser des architectures systèmes et logicielles

Design

Simulink Control Design

Linéariser des modèles et concevoir des systèmes de contrôle

Stateflow

Modéliser et simuler la logique décisionnelle avec des machines à états et des diagrammes de flux

Simscape

Modéliser et simuler des systèmes physiques multi-domaines

Satellite Communications Toolbox

Simuler, analyser et tester des liaisons et des systèmes de communications par satellite

Génération de code

Simulink Coder

Générer du code C et C++ à partir de modèles Simulink et Stateflow

Embedded Coder

Générer du code C et C++ optimisé pour systèmes embarqués

HDL Coder

Générer du code VHDL et Verilog pour les designs FPGA et ASIC

Tests et vérification

Simulink Test

Développer, gérer et exécuter des tests basés sur la simulation

Simulink Check

Vérifier la conformité aux directives de style et aux normes de modélisation

Simulink Coverage

Mesurer la couverture de test dans les modèles et le code généré

Simulink Real-Time

Créer, exécuter et tester des applications temps réel

Produits Polyspace

Prouver l'absence d'erreurs run-time critiques

En savoir plus

mathworks.com propose tout un ensemble de ressources pour vous aider à progresser rapidement dans l'approche Model-Based Design. Nous vous recommandons de commencer par celles-ci :

Présentation

MATLAB et Simulink pour les systèmes spatiaux

Tutoriels interactifs

MATLAB Onramp

Signal Processing Onramp

Image Processing Onramp

Machine Learning Onramp

Deep Learning Onramp

Simulink Onramp

Stateflow Onramp

Simscape Onramp

Control Design Onramp

Reinforcement Learning Onramp

Webinars

Simulink pour les nouveaux utilisateurs (36:05)

L'approche Model-Based Design pour les systèmes de contrôle (54:59)

Accélérer le rythme et étendre la portée du design des systèmes de contrôle (51:03)

Impact de la digitalisation et de l'IA sur l'ingénierie des systèmes spatiaux (1:23:50)

Formations sur site ou autoformations

Fondamentaux MATLAB

Simulink pour la modélisation de systèmes et d'algorithmes

Design de systèmes de contrôle avec MATLAB et Simulink

Ressources additionnelles

Services de consulting