ΕΡΓΑΣΤΗΡΙΟ ΔΥΝΑΜΙΚΗΣ ΜΗΧΑΝΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ - ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΑΡΙΣΤΟΤΕΛΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΟΝΙΚΗΣ, 54124 Θεσσαλονίκη
Διευθυντής: Καθηγητής Σ. Νατσιάβας

ΥΛΟΠΟΙΗΣΗ ΣΥΝ-ΠΡΟΣΟΜΟΙΩΣΗΣ ΔΥΝΑΜΙΚΟΥ ΣΥΣΤΗΜΑΤΟΣ ΜΕΣΩ ΤΟΥ ΠΡΟΤΥΠΟΥ FMI: ΕΦΑΡΜΟΓΗ ΣΕ ΠΡΟΒΛΗΜΑ ΑΝΑΠΗΔΟΥΜΕΝΗΣ ΣΦΑΙΡΑΣ

ΒΑΡΔΟΥΝΙΩΤΗΣ ΒΑΣΙΛΕΙΟΣ
ΑΕΜ: 5822

Επιβλέπων Καθηγητής: Νατσιάβας Σωτήριος
Συνεπιβλέπων: Καραουλάνης Φώτιος, Δρ. Πολ. Μηχ.

Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης
Θεσσαλονίκη, Μάρτιος 2020

Περίληψη

Η συν-προσομοίωση αποτελεί μια σύγχρονη και συνεχώς εξελισσόμενη τεχνική προσομοίωσης, με εφαρμογές σε προβλήματα μηχανικής, τα οποία εμπλέκουν υποσυστήματα που απαιτούν τη χρήση διαφορετικών λογισμικών επίλυσης. Την τελευταία δεκαετία έχει αποκτήσει ιδιαίτερη δημοτικότητα και επιτελούνται διαρκώς προσπάθειες για τη βελτίωσή της. Αποτέλεσμα τέτοιων προσπαθειών αποτελεί το πρότυπο FMI, μια δομή που ορίζει έναν κοινό και καθολικό τρόπο επικοινωνίας μεταξύ διαφορετικών υποσυστημάτων ενός σύνθετου προβλήματος, με στόχο τη συν-προσομοίωσή τους. Σκοπός της παρούσας διπλωματικής εργασίας είναι η ανάδειξη των δυνατοτήτων της συν-προσομοίωσης, χρησιμοποιώντας το πρότυπο FMI στην επίλυση ενός δυναμικού προβλήματος αναπηδούμενης σφαίρας. Πιο συγκεκριμένα, παρουσιάζονται αρχικά τα διάφορα εργαλεία που είναι σήμερα διαθέσιμα για την κατάστρωση των μοντέλων προς συν-προσομοίωση. Έπειτα εξετάζεται η συν-προσομοίωση των υποσυστημάτων του προβλήματος, φέροντας σε συνεργασία ένα εμπορικό λογισμικό με έναν ανεξάρτητο και προσαρμοσμένο προγραμματιστικά επιλυτή, υλοποιημένο σε γλώσσα *C++*. Τέλος παρουσιάζονται τα αποτελέσματα της συν-προσομοίωσης, αναδεικνύοντας τις δυνατότητές της, τα πλεονεκτήματα και τις αδυναμίες της.

Abstract

Co-simulation is a modern and constantly evolving simulation technique, with applications to engineering problems involving coupled subsystems, that require the use of solvers of different engineering disciplines. The technique has gained popularity during the last decade and efforts are being made for its improvement. A result of such efforts is the FMI standard, an interface that defines a common and universal way of communication amongst various subsystems of a complex problem, with the aim of co-simulating them. The objective of this thesis is to highlight the potential of co-simulation, using the FMI standard in solving a bouncing ball dynamic problem. In particular, various tools for the preparation of models for co-simulation are presented firstly. Afterwards, the co-simulation of the problem's subsystems is examined, involving the cooperation of commercial software with an independent and custom solver, implemented in *C++*. Finally, the results of the co-simulation are presented, showcasing its advantages, strengths and weaknesses.

Ευχαριστίες

Με την περάτωση της παρούσας διπλωματικής εργασίας, θα ήθελα να ευχαριστήσω θερμά όσους συνέβαλαν στην πραγμάτωσή της.

Ειδικότερα, θα ήθελα αρχικά να ευχαριστήσω τον επιβλέποντα Καθηγητή κ. Σωτήριο Νατσιάβα, για την εμπιστοσύνη που έδειξε στο πρόσωπό μου αναθέτοντάς μου τη συγκεκριμένη εργασία. Η παρουσία του και η διδασκαλία των μαθημάτων του ήταν καταλυτική στην εκπαίδευσή μου ως μηχανικό και την επαφή μου με τα πεδία της Θεωρητικής και Υπολογιστικής Μηχανικής.

Έπειτα, οφείλω να ευχαριστήσω θερμά τον Δρ. Φ. Καραουλάνη, συνεπιβλέποντα της διπλωματικής μου εργασίας. Κατά τη διάρκεια εκπόνησής της έδειξε αμέριστο ενδιαφέρον, παρείχε πολύτιμη καθοδήγηση και κατέστησε εφικτή την ολοκλήρωσή της. Μέσα από ουσιώδεις συζητήσεις και υποδείξεις συνέβαλε στην όξυνση της κριτικής μου ικανότητας και διαμόρφωση της αντίληψής μου ως μηχανικό, ενώ παράλληλα μου μετέδωσε απαραίτητες γνώσεις στα πεδία της Υπολογιστικής Μηχανικής και του προγραμματισμού.

Ευχαριστίες οφείλω να αποδώσω και στους υποψήφιους διδάκτορες και τα μέλη του Εργαστηρίου Δυναμικής Μηχανών, κ. Π. Πασσά, κ. Ι. Ντινόπουλο και κ. Δ. Λιάλιο για τις συμβουλές, την υποστήριξη και το ευχάριστο κλίμα που παρείχαν.

Τέλος, θα ήθελα να εκφράσω την αμέριστη ευγνωμοσύνη μου στην οικογένεια και τους φίλους μου, που στήριξαν τις προσπάθειές μου καθ' όλη τη διάρκεια των φοιτητικών μου χρόνων.

Βαρδουνιώτης Βασίλειος

Περιεχόμενα

Κατάλογος σχημάτων

Λίστα αλγορίθμων

Λίστα καταγραφών

1 Εισαγωγή

Η προσομοίωση ως έννοια σήμερα είναι συνυφασμένη με το μηχανολόγο μηχανικό. Αποτελεί αναπόσπαστο κομμάτι κάθε ολοκληρωμένης μελέτης, λ.χ. μιας κατασκευής ή ενός μηχανισμού. Με αυτήν είναι δυνατό να υπολογιστεί και να προβλεφθεί η συμπεριφορά του εξεταζόμενου αντικειμένου υπό διάφορες συνθήκες και επομένως το αποτέλεσμα της μελέτης να είναι ακριβές και αξιόπιστο.

Για την επίτευξη ρεαλιστικών αποτελεσμάτων στην προσομοίωση απαιτούνται σύνθετα και μεγάλα μαθηματικά μοντέλα. Αυτά, εξ' αιτίας των πολλών παραμέτρων και βαθμών ελευθερίας που περιέχουν, καταλήγουν να περιορίζουν το μηχανικό στη μελέτη ενός προβλήματος, καθώς εισάγουν αυξημένη πολυπλοκότητα και σημαντικό χρόνο επίλυσης. Οι σημερινοί εμπορικοί υπολογιστές έχουν πεπερασμένη υπολογιστική ισχύ, οπότε και είναι απαραίτητη η αναζήτηση εναλλακτικών μεθόδων και τεχνικών προσομοίωσης.

Ο μηχανικός έχει στη διάθεσή του πληθώρα εμπορικών πακέτων λογισμικού, τα οποία είναι εξειδικευμένα στην επίλυση ενός συγκεκριμένου τύπου προβλήματος. Τα προσομοιώματα μηχανισμών επιλύονται ως δυναμικά συστήματα πολλαπλών σωμάτων (MultiBody Dynamics, "MBD"), τα κοκκώδη υλικά με επαναλαμβανόμενο γεωμετρικό μοτίβο ως μοντέλο διακριτών σωμάτων (Discrete Element Method, "DEM") [1], ενώ τα ρευστά μπορούν να προσομοιωθούν πλέον και αυτά με επαναλαμβανόμενο γεωμετρικό μοτίβο ως μοντέλο διακριτών σωματιδίων (Smoothed Particle Hydrodynamics, "SPH"). Είτε χρησιμοποιούνται κλασικές μέθοδοι προσομοίωσης είτε νέες, επιδιώκεται πάντα η μέγιστη δυνατή ακρίβεια με στόχο τη βέλτιστη απεικόνιση της πραγματικότητας.

Η επίλυση πολυδιάστατων προβλημάτων από διαφορετικές επιστημονικές σκοπιές οδηγεί σε πολλαπλά μαθηματικά μοντέλα. Η μελέτη και σχεδίαση λ.χ. ενός βαρέως οχήματος μεταφοράς υλικού απαιτεί την επίλυση πολυάριθμων επιμέρους προβλημάτων που προσεγγίζονται από διαφορετικά μαθηματικά προσομοιώματα. Ο μηχανισμός εκτροπής της καρότσας προσεγγίζεται ως μοντέλο πολλαπλών σωμάτων, ενώ το μεταφερόμενο υλικό προσεγγίζεται ως μοντέλο διακριτών σωμάτων.

Έτσι, γεννάται η ανάγκη της προσομοίωσης και επίλυσης όλων των επιμέρους υπο-προβλημάτων γρήγορα, αποδοτικά και με τη μέγιστη δυνατή ακρίβεια. Για να επιτευχθεί αυτό, απαιτείται ουσιαστικά η επίλυση των διαφορετικής φύσεως υπο-μοντέλων με τον αντίστοιχο επιλυτή, ταυτόχρονα. Το κενό αυτό έρχεται να γεφυρώσει η σύγχρονη τεχνική της συν-προσομοίωσης.

2 Σύντομη βιβλιογραφική ανασκόπηση

Με τον όρο «συν-προσομοίωση (αγγλ. co-simulation)» εννοείται ένα θεωρητικό και τεχνικό σύνολο με σκοπό να επιτευχθεί καθολική προσομοίωση ενός συζευγμένου συστήματος, μέσω της σύνθεσης των επιμέρους προσομοιωτών [2]. Είναι στην ουσία ένας τρόπος επικοινωνίας μεταξύ διαφορετικών επιλυτών με στόχο να λύνουν το ίδιο πρόβλημα ταυτόχρονα, από διαφορετική σκοπιά ο καθένας. Το γενικό πρόβλημα είναι αποσυζευγμένο σε επιμέρους υπο-προβλήματα και ο εκάστοτε επιλυτής επιλύει το αντίστοιχο για το οποίο είναι υλοποιημένος.

Η συν-προσομοίωση αποτελεί μια σύγχρονη, πολλά υποσχόμενη τεχνική μοντελοποίησης προβλημάτων, που πρόσφατα άρχισε να κεντρίζει το ενδιαφέρον των μηχανικών. Αυτό γίνεται εμφανές από την πληθώρα εργαλείων που υποστηρίζουν σήμερα το πρότυπο FMI (βλ. ενότητα 3.1) και παρουσιάζονται στην επίσημη ιστοσελίδα[1] του προτύπου.

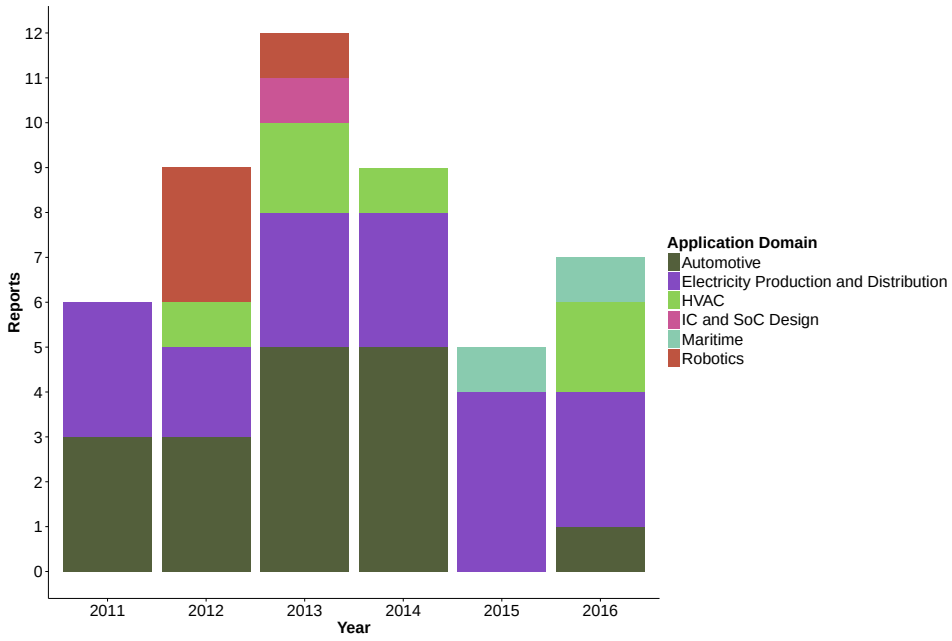
Ανέκαθεν γίνονταν προσπάθειες για τη συνεργασία και τη διεπαφή μεταξύ υπαρχόντων λογισμικών, αλλά με μη μεθοδικό τρόπο και μη γενικεύσιμη διαδικασία σε πολλαπλούς τύπους προβλημάτων. Την τελευταία δεκαετία, με τη δημιουργία και τη διάδοση του προτύπου FMI, προτάθηκε ένας αναλυτικός τρόπος διασύνδεσης δυναμικών μοντέλων και εκτέλεσης συν-προσομοίωσης. Έκτοτε, ολοένα και περισσότεροι μελετητές έχουν εφαρμόσει τη συν-προσομοίωση σε σύνθετα προβλήματα διαφόρων πεδίων της μηχανικής [2].

Σχετική έρευνα των *Gomes et al.* [2] δείχνει πως ανάμεσα σε περίπου 30 μοντέλα από δημοσιεύσεις παρατηρούνται επιλύσεις που περιλαμβάνουν τη συν-προσομοίωση κυρίως δύο μόνο υποσυστημάτων, γεγονός που αναμένεται να αλλάξει τα επόμενα χρόνια. Οι μέχρι σήμερα υλοποιήσεις υποδεικνύουν τις ολοένα και αυξανόμενες απαιτήσεις σε χαρακτηριστικά της συν-προσομοίωσης, καθώς και σε ακρίβεια, πιστότητα και ταχύτητα. Στο σχήμα 1 παρουσιάζονται διάφορα πεδία εφαρμογών συν-προσομοίωσης των πρόσφατων ετών που επιβεβαιώνουν την εξάπλωση της μεθόδου και τις εκτεταμένες δυνατότητές της.

Ορισμένοι μελετητές έχουν στρέψει το ενδιαφέρον τους στην επέκταση των δυνατοτήτων του προτύπου FMI, παρέχοντας εργαλεία και δυνατότητες με στόχο την ακριβέστερη συν-προσομοίωση. Συγκεκριμένα, οι *F. Cremona et al.* [3], προτείνουν προσθήκες στο πρότυπο προκειμένου να υποστηρίζει την εκτέλεση «υβριδικής συν-προσομοίωσης». Με τον όρο «υβριδική συν-προσομοίωση» εννοείται η ταυτόχρονη ύπαρξη ασυνεχών σημάτων και γεγονότων συνεχούς, διακριτού και μηδενικού χρόνου κατά

[1] https://fmi-standard.org/tools/

τη διάρκεια του χρόνου της συν-προσομοίωσης. Η παρούσα έκδοση του προτύπου FMI (2.0) δεν υποστηρίζει αυτές τις δυνατότητες και προτείνουν μια διαφορετική υλοποίηση για το μοντέλο του χρόνου που εφαρμόζει το πρότυπο.



Σχήμα 1: Πεδία εφαρμογών συν-προσομοίωσης της τελευταίας δεκαετίας (από [2]).

Η συν-προσομοίωση όμως σαν τεχνική εξαπλώνεται και σε κλάδους πέραν της μηχανικής, με προσπάθειες να γίνονται για τη δημιουργία δομών και προτύπων παρόμοιας λογικής με του FMI. Οι *F. Gaisbauer et al.* [4], θέλοντας να εφαρμόσουν τη συν-προσομοίωση στο πεδίο της ανθρώπινης προσομοίωσης και κινησιολογίας, προτείνουν τη δημιουργία του προτύπου «Motion Model Interface» (MMI), βασισμένο στο πρότυπο FMI. Μέσω αυτού θα είναι εφικτή η συν-προσομοίωση σύνθετων κινήσεων του ανθρώπινου σώματος, διαχωρισμένες σε προσομοιώματα απλούστερων κινήσεων. Ο συγκεκριμένος τύπος προσομοιώσεων βρίσκει απήχηση στους τομείς υγείας, διασκέδασης, ήχου και εικόνας.

3 Διαθέσιμα εργαλεία

Στόχος του συγκεκριμένου κεφαλαίου είναι να παρουσιάσει συγκεντρωμένα ορισμένα από τα εργαλεία που έχει στη διάθεσή του ο μηχανικός προκειμένου να καταστρώσει το μοντέλο ενός προβλήματος, όντας κατάλληλο για συν-προσομοίωση. Τα συγκεκριμένα εργαλεία αποτέλεσαν σημείο αναφοράς για την εκπόνηση της παρούσας εργασίας. Έπειτα από την παρουσίασή τους γίνεται εμφανές το γεγονός ότι η προετοιμασία ενός μοντέλου για χρήση σε συν-προσομοίωση περιλαμβάνει αρκετά στάδια και εμπλέκει μια ποικιλία διαφορετικών εργαλείων λογισμικού.

3.1 Το πρότυπο FMI

Το FMI (Functional Mock-up Interface) αποτελεί ένα πρότυπο μέσω του οποίου ορίζεται μια «διεπαφή (αγγλ. interface)». Η διεπαφή αυτή υλοποιείται σε ένα εκτελέσιμο αρχείο συμπιεσμένης μορφής με ονομασία FMU (Functional Mock-up Unit). Ένα FMU περιγράφει ένα οποιοδήποτε σύστημα με τις εξισώσεις που το ορίζουν και μπορεί εν δυνάμει να μην περιέχει το δικό του κατάλληλο επιλυτή, που σε αυτή την περίπτωση απαιτείται η αριθμητική του ολοκλήρωση μέσω του περιβάλλοντος προσομοίωσης [5]. Τα FMU επιτρέπουν την εύκολη και γρήγορη αρχικοποίηση ενός συστήματος σε ένα περιβάλλον προσομοίωσης και συνήθως χρησιμοποιούνται σε συνδυασμό με άλλα FMU. Το πρότυπο ορίζει δύο περιπτώσεις διεπαφής ανάλογα με τη χρήση: το FMI for Model Exchange («ανταλλαγή» μοντέλων) και το FMI for Co-Simulation (συν-προσομοίωση).

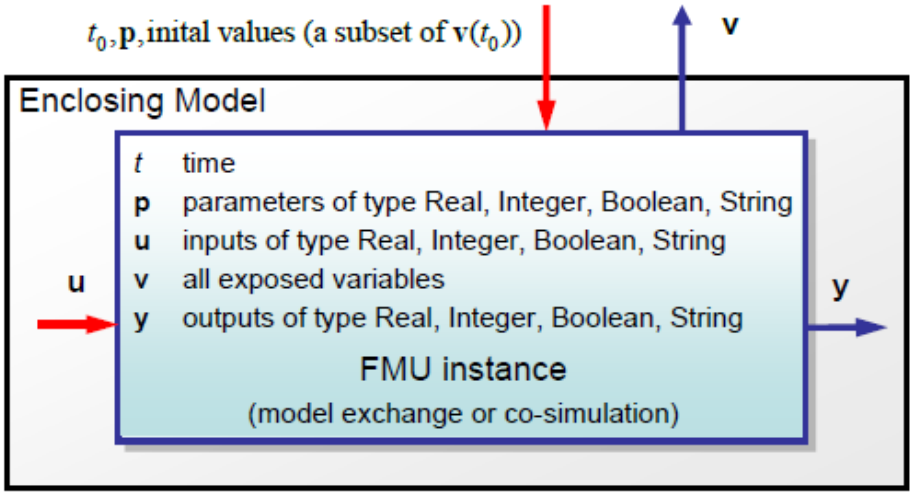
Μέσω του Model Exchange δημιουργείται η διεπαφή τύπου FMU-ME. Με αυτό τον τρόπο ορίζονται μοντέλα δυναμικών συστημάτων που περιγράφονται από διαφορικές, αλγεβρικές και διακριτού χρόνου εξισώσεις, τα οποία έχουν σκοπό να ολοκληρωθούν χρονικά σε περιβάλλοντα προσομοίωσης είτε σε «ενσωματωμένα περιβάλλοντα ελέγχου (αγγλ. embedded control systems)» Επιδέχονται άμεσα (explicit) και έμμεσα (implicit) σχήματα ολοκλήρωσης, σταθερού ή μεταβλητού χρονικού βήματος [5].

Μέσω του Co-Simulation δημιουργείται η διεπαφή τύπου FMU-CS. Με αυτό τον τρόπο ορίζονται μοντέλα δυναμικών συστημάτων τα οποία όμως σε αυτή την περίπτωση έχουν στόχο τη σύζευξή τους με άλλα μοντέλα για την εκτέλεση συν-προσομοίωσης. Ένα μοντέλο τέτοιου τύπου περιλαμβάνει το δικό του επιλυτή, μαζί με την περιγραφή του μέσω εξισώσεων, όλα συμπιεσμένα μέσα στο εκτελέσιμο αρχείο. Ο σκοπός των FMU-CS είναι να υπολογίζουν την λύση μοντέλων και συστημάτων που εξαρτώνται από το χρόνο, τα οποία περιέχουν υποσυστήματα που είναι «συζευγμένα (αγγλ. coupled)» μεταξύ τους. Τα υποσυστήματά τους μπορούν να είναι

χρονικά συνεχή και να περιγράφονται από διαφορικές-αλγεβρικές εξισώσεις (DAE), είτε να είναι χρονικά διακριτά και να περιγράφονται από εξισώσεις διαφορών, όπως σε συστήματα ψηφιακών ελεγκτών [5].

Σε μια σχηματική απεικόνιση συζευγμένου συστήματος, τα υποσυστήματα αναπαρίστανται ως αυτόνομα «μπλοκ» με εσωτερικές μεταβλητές κατάστασης $\mathbf{x}(t)$ και συνδέονται με τα υπόλοιπα υποσυστήματα μέσω των μεταβλητών εισόδου (inputs) $\mathbf{u}(t)$ και των μεταβλητών εξόδου (outputs) $\mathbf{y}(t)$. Κατά τη διάρκεια της προσομοίωσης, το κάθε υποσύστημα ολοκληρώνεται ανεξάρτητα από τα υπόλοιπα με το δικό του χρονικό βήμα, ενώ η επικοινωνία μεταξύ τους μέσω των μεταβλητών εισόδου και εξόδου γίνεται σε συγκεκριμένα χρονικά σημεία επικοινωνίας (communication points) [5].

Οι διεπαφές και των δύο τύπων που ορίζει το πρότυπο έχουν και ορισμένα κοινά χαρακτηριστικά. Χρησιμοποιούν το FMI Application Programming Interface (API), δηλαδή ένα προγραμματιστικό περιβάλλον διεπαφής, γραμμένο σε γλώσσα προγραμματισμού *C++*, μέσω του οποίου ο χρήστης έχει πρόσβαση σε βιβλιοθήκες και συναρτήσεις απαραίτητες για τον ορισμό των μοντέλων και των εξισώσεων που τα διέπουν. Επίσης, τα FMU περιγράφονται από μια κοινή δομή, στη μορφή ενός αρχείου τύπου *.xml*. Εκεί αναγράφονται όλες οι μεταβλητές, ο τύπος τους, και οι ιδιότητες του μοντέλου με έναν συστηματοποιημένο και κοινό τρόπο [6].



Σχήμα 2: Σχηματική απεικόνιση του γενικού FMU. Ροή δεδομένων εισόδου και εξόδου κατά τη διάρκεια της συν-προσομοίωσης (από [5]).

Εν συντομία, ένα FMU περιγράφει ένα σύστημα και τις εξισώσεις που το διέπουν (με ή χωρίς αυτόνομο επιλυτή) και διανέμεται σε συμπιεσμένη

μορφή τύπου *.zip*. Περιέχει το αρχείο *.xml* που περιγράφει το σύστημα και τις μεταβλητές του, καθώς και τα εκτελέσιμα αρχεία δυαδικού κώδικα και τις δυναμικές προγραμματιστικές βιβλιοθήκες, απαραίτητα για την προσομοίωση.

3.2 Η γλώσσα προγραμματισμού Modelica

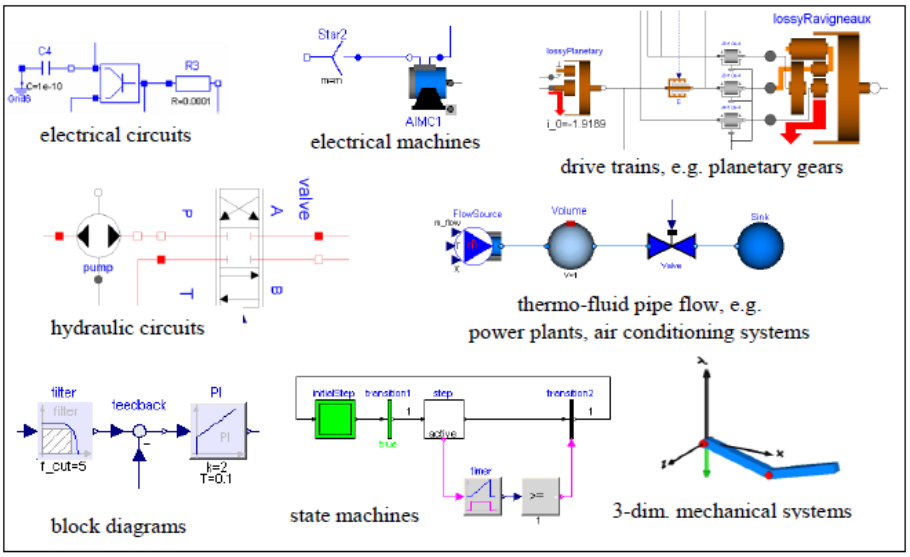
Η Modelica είναι μια ελεύθερα διαθέσιμη αντικειμενοστραφής γλώσσα προγραμματισμού, κατάλληλη για τη μοντελοποίηση μεγάλων, σύνθετων και ετερογενών συστημάτων. Ενδείκνυται για προβλήματα που απαιτούν ολιστική προσέγγιση, λ.χ. μηχατρονικής, ρομποτικής, αεροδιαστημικής και αυτοκινητοβιομηχανίας, που περιλαμβάνουν πολλά υποσυστήματα και αυτόματο έλεγχο. Τα μοντέλα μέσω της γλώσσας περιγράφονται από διαφορικές, αλγεβρικές και διακριτές εξισώσεις [7].

Από τη σκοπιά ενός χρήστη, τα μοντέλα αναπαρίστανται από σχηματικά διαγράμματα, όπως φαίνεται στο σχήμα 3. Τα διάφορα υποσυστήματα είναι μεταξύ τους συνδεδεμένα με «διασυνδέσεις (αγγλ. connectors)», που επιτρέπουν τη μεταφορά δεδομένων και ενεργοποιούν δυνατότητες αλληλεπίδρασης λ.χ. σήμα εισόδου [7]. Όπως είναι εμφανές, πρόκειται για μια γλώσσα «υψηλού επιπέδου (αγγλ. high-level)», που σημαίνει ότι ο ορισμός ενός μοντέλου γίνεται με αφηρημένο και σχηματικό τρόπο, χωρίς ο χρήστης να πρέπει να μεταφράσει ο ίδιος το εικονικό μοντέλο του σε μαθηματικό προσομοίωμα με εξισώσεις. Το προγραμματιστικό περιβάλλον που χρησιμοποιείται, εκμεταλλευόμενο τις δυνατότητες της Modelica, μετατρέπει αυτόματα τον αφηρημένο ορισμό των μοντέλων σε εξισώσεις και συνεπώς σε κώδικα. Φυσικά υπάρχει και η δυνατότητα απ'ευθείας επεξεργασίας εξισώσεων σε κώδικα Modelica [8, 9], δυνατότητα η οποία χρησιμοποιήθηκε στα πλαίσια αυτής της εργασίας, λόγω της μεγαλύτερης προγραμματιστικής ευελιξίας που προσφέρει.

Προκειμένου να γίνει χρήση της γλώσσας για την επίλυση οποιουδήποτε προβλήματος απαιτείται η ύπαρξη ενός προγραμματιστικού περιβάλλοντος μοντελοποίησης. Με ένα τέτοιο περιβάλλον ο χρήστης έχει τη δυνατότητα να ορίσει το μοντέλο ενός μαθηματικού προσομοιώματος μέσω γραφικού περιβάλλοντος είτε με τη χρήση διαγραμμάτων ροής. Το μοντέλο έπειτα μεταφράζεται αυτόματα σε μορφή τέτοια ώστε να μπορεί να γίνει η προσομοίωσή του, από συμβολική μορφή σε σύστημα εξισώσεων. Τέλος, η προσομοίωση εκτελείται με τυπικές μεθόδους αριθμητικής ολοκλήρωσης και τα αποτελέσματα οπτικοποιούνται [10].

Τέτοια προγραμματιστικά περιβάλλοντα είναι διαθέσιμα είτε σε μορφή εμπορικού πακέτου είτε σε δωρεάν διανομές λογισμικού «ανοιχτού κώ-

δικα (αγγλ. open-source)». Στις επόμενες υποενότητες παρουσιάζονται συνοπτικά ορισμένες από τις διαθέσιμες επιλογές, καθώς και η υλοποίηση που επιλέχθηκε για την εκπόνηση της παρούσας εργασίας.



Σχήμα 3: Σχηματική απεικόνιση μοντέλων διαφορετικών επιστημονικών πεδίων μέσω της γλώσσας Modelica (από [7]).

3.2.1 JModelica

Η JModelica.org είναι μια πλατφόρμα λογισμικού ανοιχτού κώδικα, βασισμένη στο πρότυπο FMI και τη γλώσσα Modelica, σχεδιασμένη για προσομοίωση, βελτιστοποίηση και ανάλυση σύνθετων δυναμικών συστημάτων [11]. Είναι συμβατή με μια πληθώρα προγραμματιστικών βιβλιοθηκών Modelica, συμπεριλαμβανόμενης και της διαδεδομένης "Modelica Standard Library" (MSL). Η πλατφόρμα είναι προϊόν έρευνας του Τμήματος Αυτομάτου Ελέγχου του πανεπιστημίου του Lund [12], ενώ σήμερα συνεχίζει να συντηρείται και να αναπτύσσεται από την Modelon AB σε συνεργασία με το πανεπιστήμιο.

Με τη JModelica.org είναι εφικτή [13]:

- Η μοντελοποίηση συστημάτων με τη χρήση της γλώσσας Modelica
- Η επίλυση σύνθετων προβλημάτων προσομοίωσης και βελτιστοποίησης με χρήση σύγχρονων τεχνικών και αλγορίθμων αριθμητικής επίλυσης

- Η αυτοματοποίηση της εργασίας του χρήστη μέσω περιβάλλοντος «γλώσσας σεναρίου (αγγλ. scripting language)» Python
- Η οπτικοποίηση αποτελεσμάτων.

Πιο συγκεκριμένα, η πλατφόρμα βασίζεται στη γλώσσα προγραμματισμού Python για τη διεπαφή με το χρήστη, προσφέροντας εργαλεία ανάπτυξης εξειδικευμένων εφαρμογών και πρωτότυπων αλγορίθμων ολοκλήρωσης. Διαθέτει τις μαθηματικές βιβλιοθήκες Numpy και Scipy, απαραίτητες για την υποστήριξη αριθμητικών πράξεων γραμμικής άλγεβρας και διανυσμάτων. Υποστηρίζεται επίσης το πρότυπο FMI, με χρήση των FMU αρχείων, όπως περιγράφηκαν στην ενότητα 3.1. Αυτό γίνεται με τη χρήση της βιβλιοθήκης PyFMI, η οποία καθιστά εφικτή τη χρήση μοντέλων που έχουν μεταφραστεί σε FMU, καθώς και την προσομοίωσή τους, τη ρύθμιση των παραμέτρων τους και την οπτικοποίηση των αποτελεσμάτων τους [11].

Στα πλαίσια της συγκεκριμένης εργασίας, επιλέχθηκε η JModelica.org ως πλατφόρμα προσομοίωσης, λόγω της εκτενούς ευελιξίας που προσφέρει στο χρήστη. Έγινε χρήση της βιβλιοθήκης PyFMI κατά την προσομοίωση του προβλήματος, εξ' αιτίας της ευκολίας που προσφέρει στη ρύθμιση των μοντέλων αλλά και στην υλοποίηση του κώδικα επίλυσης.

3.2.2 OpenModelica

Το OpenModelica είναι ένα ελεύθερα διαθέσιμο περιβάλλον προσομοίωσης μοντέλων. Βασίζεται στη γλώσσα Modelica και χρησιμοποιείται για ακαδημαϊκή και βιομηχανική χρήση. Αναπτύσσεται και υποστηρίζεται από το Πανεπιστήμιο του Linköping και το Open Source Modelica Consortium (OSMC), το οποίο είναι ένα σύνολο επιχειρήσεων, πανεπιστημίων, ινστιτούτων και μεμονωμένων ατόμων [10].

Ο στόχος του OpenModelica είναι να παρέχει ένα πλήρες περιβάλλον προσομοίωσης μοντέλων, δωρεάν διανεμημένο και βασισμένο σε ελεύθερο λογισμικό που αποτελεί προϊόν έρευνας, διδασκαλίας και βιομηχανικής χρήσης [14]. Στην τωρινή του μορφή, μπορεί να υποστηρίξει σχεδόν το σύνολο των λειτουργιών της γλώσσας Modelica, συμπεριλαμβανομένων εξισώσεων, αλγορίθμων, διαχείρισης χρονικών γεγονότων και πακέτων βιβλιοθηκών [10]. Περιλαμβάνει επίσης εργαλεία για τον εύκολο και γρήγορο ορισμό μοντέλων και συστημάτων, είτε σε σχηματική είτε σε μορφή κώδικα, καθώς και οπτικοποίηση αποτελεσμάτων. Το κυρίως εργαλείο του περιβάλλοντος είναι ο μεταφραστής OpenModelica Compiler (OMC), που μεταφράζει τα συστήματα που δημιουργούνται στη γλώσσα Modelica, σε εκτελέσιμα αρχεία δυαδικού κώδικα προς προσομοίωση.

3.2.3 Dymola

Το DYMOLA (Dynamic Modeling Laboratory) είναι ένα εμπορικό λογισμικό που παρέχεται από την εταιρεία Dassault Systemes. Αποτελεί ένα από τα δημοφιλέστερα λογισμικά πακέτα στο χώρο της προσομοίωσης, χάρη στην ποικιλία εργαλείων που προσφέρει, αλλά και λόγω της ιστορίας του. Σχεδιάστηκε αρχικά από τον Hilding Elmqvist το 1978 και συνεχίστηκε από την εταιρεία που ο ίδιος δημιούργησε το 1992, Dynamism AB. Η γλώσσα Modelica αποτέλεσε ένα παρα-προϊόν του Dymola και ξεκίνησε έκτοτε να αναπτύσσεται παράλληλα. Το 2006 η εταιρεία αγοράστηκε από την Dassault Systemes και το Dymola συνεχίστηκε να αναπτύσσεται ξεχωριστά από τη Modelica [15].

Πρόκειται για ένα πλήρες εργαλείο με ολοκληρωμένο περιβάλλον προσομοίωσης, κατάλληλο για σύνθετα και πολυδιάστατα μοντέλα εφαρμογών αεροναυπηγικής, ρομποτικής, αυτοκινητοβιομηχανίας και διεργασιών. Με αυτό είναι εφικτή η δημιουργία, η επίλυση, ο έλεγχος και η οπτικοποίηση αποτελεσμάτων σύνθετων προβλημάτων. Το λογισμικό υποστηρίζει την τελευταία υλοποίηση της γλώσσας Modelica και τη βιβλιοθήκη Modelica Standard Library (MSL), καθώς και το πρότυπο FMI [16].

3.3 Ο κώδικας MotionSolve

Ο κώδικας MotionSolve είναι ένας κώδικας επίλυσης προβλημάτων δυναμικής πολλαπλών σωμάτων (αγγλ. Multi Body Dynamics, MBD) και είναι υλοποιημένος μέσω του εμπορικού λογισμικού Motionview της εταιρείας Altair. Ο ρόλος του είναι να επιλύει συστήματα διαφορικών - αλγεβρικών εξισώσεων (αγγλ. Differential Algebraic Equations, DAE), τα οποία εμφανίζονται από τη συστηματική διατύπωση εξισώσεων κίνησης σε συστήματα πολλαπλών σωμάτων. Η διατύπωση αυτή γίνεται με βάση τη θεωρία του Lagrange, κατά την οποία προκύπτουν κανονικές διαφορικές εξισώσεις από τις εξισώσεις κίνησης, καθώς και αλγεβρικές εξισώσεις από τους περιορισμούς (γεωμετρικούς και κινηματικούς) [17].

Ένα σύστημα DAE χαρακτηρίζεται από τον βαθμό του, ο οποίος ισούται με τον αριθμό των παραγωγίσεων ως προς την ανεξάρτητη μεταβλητή που απαιτούνται προκειμένου να μετατραπεί σε σύστημα κανονικών διαφορικών εξισώσεων (ODE). Για ένα σύστημα βαθμού 3 (DAE Index 3) το MotionSolve διαθέτει τον ολοκληρωτή DSTIFF, κατάλληλος να επιλύσει ένα σύστημα της μορφής $\mathbf{G}(t, \mathbf{y}, \dot{\mathbf{y}}) = \mathbf{0}$. Ο DSTIFF είναι βασισμένος στον ελεύθερα διαθέσιμο επιλυτή DASPK και περιλαμβάνει ένα συνδυασμό από μεθόδους Backward Differentiation Formulas (BDF) [18].

Πιο συγκεκριμένα, οι εξισώσεις του εν λόγω συστήματος (DAE Index

3) σε μορφή Euler-Lagrange είναι οι εξής:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} + \sum_{j=1}^{M} \lambda_j \frac{\partial \Phi_j}{\partial q_i} - \sum_{k=1}^{Na} F_k \frac{\partial v_k}{\partial q_i} = 0 \quad , \qquad i = 1, 2, ..., N \tag{1}$$

$$\Phi_j(\mathbf{q}, t) = 0 \quad , \qquad j = 1, 2, ..., M \ , \tag{2}$$

όπου:

- $\mathbf{q} = [q_1, q_2, ..., q_N]^T$ οι N γενικευμένες συντεταγμένες
- $\mathbf{\Phi} = [\Phi_1(\mathbf{q}, t), \Phi_2(\mathbf{q}, t), ..., \Phi_M(\mathbf{q}, t)]^T$ οι M ολόνομοι δεσμοί κίνησης που προκύπτουν από τους περιορισμούς
- $\boldsymbol{\lambda} = [\lambda_1, \lambda_2, ..., \lambda_N]^T$ οι M πολλαπλασιαστές Lagrange
- $\mathbf{F} = [F_1, F_2, ..., F_N a]^T$ οι Na γενικευμένες εξωτερικές δυνάμεις
- $T(\mathbf{q}, \dot{\mathbf{q}})$ η κινητική ενέργεια του συστήματος
- $V(\mathbf{q})$ η δυναμική ενέργεια του συστήματος
- $L(\mathbf{q}, \dot{\mathbf{q}}) = T(\mathbf{q}, \dot{\mathbf{q}}) - V(\mathbf{q})$ η εξίσωση Lagrange.

Το σύστημα εξισώσεων για να πάρει πιο συμπαγή μορφή, υποβιβάζεται κατά μία τάξη με την ανάθεση $\mathbf{u} = \dot{\mathbf{q}}$ (4) και ξαναγράφεται σε μητρωική μορφή ως εξής:

$$\begin{cases} \mathrm{M}\dot{\mathbf{u}} + \mathbf{\Phi}^T \boldsymbol{\lambda} - \mathbf{f}(\mathbf{q}, \mathbf{u}) = \mathbf{0} & (3) \\ \mathbf{u} - \dot{\mathbf{q}} = \mathbf{0} & (4) \\ \mathbf{\Phi}(\mathbf{q}, t) = \mathbf{0} \ . & (5) \end{cases}$$

Οι εξισώσεις (3) εκφράζουν το δεύτερο νόμο του Νεύτωνα. Ο πρώτος όρος αντιπροσωπεύει τις αδρανειακές δυνάμεις, ο δεύτερος όρος τις δυνάμεις των δεσμών και ο τρίτος όρος τις γενικευμένες εξωτερικές δυνάμεις. Οι εξισώσεις (5) εκφράζουν τους αλγεβρικούς περιορισμούς που προκύπτουν από την επιβολή των δεσμών. Από τις N γενικευμένες συντεταγμένες οι M δεν είναι γραμμικά ανεξάρτητες, λόγω των δεσμών. Έτσι, το σύστημα διαθέτει $p = \mathrm{N} - \mathrm{M}$ βαθμούς ελευθερίας.

Θέτοντας $\mathbf{y} = [\mathbf{u}^T, \mathbf{q}^T, \boldsymbol{\lambda}^T]^T$ στις εξισώσεις (3), (4) και (5), προκύπτει η συμπαγής μορφή που περιγράφηκε προηγουμένως:

$$\mathbf{G}(\mathbf{y}, \dot{\mathbf{y}}, t) = \mathbf{0} \ . \tag{6}$$

Παραγωγίζοντας τις εξισώσεις (3) μία φορά και τις (5) δύο φορές, το σύστημα διαφορικών - αλγεβρικών εξισώσεων (DAE) (6) μετατρέπεται σε σύστημα κανονικών διαφορικών εξισώσεων (ODE) υπό τη μορφή:

$$\dot{\mathbf{y}} = \mathbf{g}(\mathbf{y}, t) \ . \tag{7}$$

Συνολικά πραγματοποιήθηκαν τρεις παραγωγίσεις, οπότε το σύστημα έχει βαθμό (index) 3. Όσον αφορά τη διαδικασία επίλυσης, αυτή διακρίνεται σε δύο στάδια - την πρόβλεψη και τη διόρθωση. Στο πρώτο στάδιο, πραγματοποιείται μια πρόβλεψη για τις αρχικές συνθήκες της επόμενης χρονικής στιγμής. Η πρόβλεψη αυτή γίνεται μέσω πολυωνυμικής παρεμβολής, με πολυώνυμο το οποίο περνά από τις προηγουμένως υπολογισθείσες τιμές. Δεν επιλύονται οι εξισώσεις κίνησης σε αυτό το στάδιο, παρά μόνο γίνεται η πρόβλεψη. Οι συντελεστές που ορίζουν το πολυώνυμο παρεμβολής υπολογίζονται μέσω της μεθόδου διηρημένων διαφορών (Newton Divided Differences). Σύμφωνα με τη μέθοδο αυτή, για την πρόβλεψη της χρονικής στιγμής t_{n+1} το πολυώνυμο έχει την εξής μορφή:

$$y^i_{n+1} = y^i_n + (t - t_n) \cdot \left[y^i_n, y^i_{n-1}\right] + (t - t_n) \cdot (t - t_{n-1}) \cdot \left[y^i_n, y^i_{n-1}, y^i_{n-2}\right] + \cdots \tag{8}$$

$$i = 1, 2, ..., 2N + M \ ,$$

με:

$$\left[y^i_n, y^i_{n-1}\right] = \frac{y^i_{n-1} - y^i_n}{t_{n-1} - t_n}, \quad \left[y^i_n, y^i_{n-1}, y^i_{n-2}\right] = \frac{\left[y^i_{n-1}, y^i_{n-2}\right] - \left[y^i_n, y^i_{n-1}\right]}{t_{n-2} - t_n} \ . \tag{9}$$

Στο δεύτερο στάδιο, πραγματοποιείται η διόρθωση της πρόβλεψης ή αλλιώς των αρχικών συνθηκών του πρώτου σταδίου. Ο ολοκληρωτής DSTIFF χρησιμοποιεί τη μέθοδο Newton-Raphson για να διασφαλίσει ότι η διόρθωση της πρόβλεψης ικανοποιεί τις εξισώσεις κίνησης. Αυτό το πετυχαίνει μετατρέποντας το σύστημα DAE (6) σε ένα σύστημα μη γραμμικών αλγεβρικών εξισώσεων διαφορών. Ο μετασχηματισμός αυτός πραγματοποιείται με αντικατάσταση της (10) στην (6). Η εξίσωση αυτή είναι το σχήμα ολοκλήρωσης και αποτελεί μια έμμεση μέθοδο πολλαπλού βήματος [19]:

$$y^i_{n+1} = \sum_{j=1}^{k} \alpha_j \cdot y^i_{n-j+1} + h \cdot \beta_0 \cdot \mathbf{g}^i \left(y^i_{n+1}, t_{n+1}\right) \ , \tag{10}$$

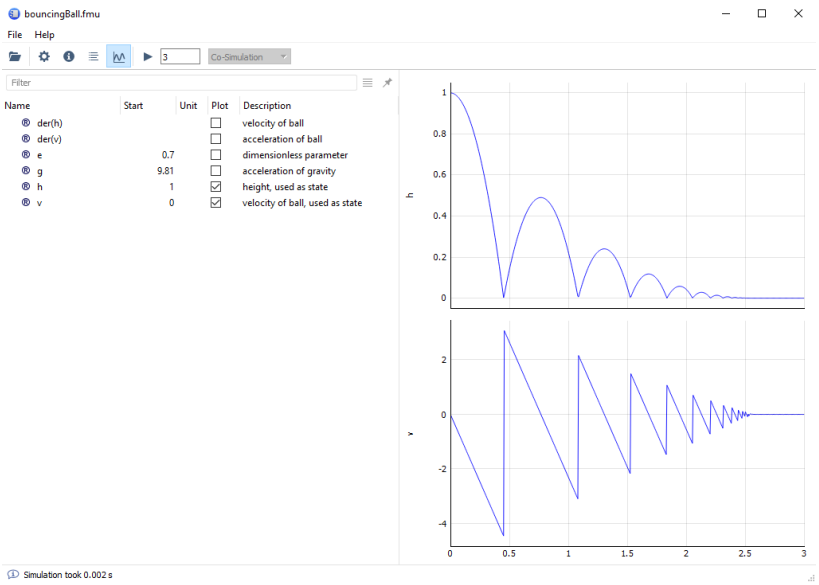
όπου:

- h το χρονικό βήμα
- k ο βαθμός του ολοκληρωτή
- α_j, β_0 σταθερές του ολοκληρωτή, εξαρτώμενες από τη χρησιμοποιούμενη μέθοδο.

3.4 Το εργαλείο FMU Simulator

Το FMU Simulator είναι ένα «αυτόνομο (αγγλ. standalone)» πρόγραμμα της εταιρείας 3DS-Dassault Systemes που επιτρέπει στο χρήστη την επαλήθευση, επισκόπηση και προσομοίωση των FMU. Υποστηρίζει την πρόσφατη έκδοση του προτύπου FMI και για Model Exchange (FMU-ME) και για Co-simulation (FMU-CS), ενώ επιτρέπει και τη χρήση απλών αρχείων για εισαγωγή μεταβλητών εισόδου και εξόδου (input-output) στα FMU [20].

Με το FMU Simulator είναι δυνατή η γρήγορη επαλήθευση των FMU που δημιουργεί ο χρήστης. Το πρόγραμμα προσφέρει έναν εύκολο τρόπο προσομοίωσής τους, καθώς και αυτόματη παραγωγή οπτικών διαγραμμάτων. Αξίζει να σημειωθεί ότι το εκάστοτε FMU προσομοιώνεται αυτόνομα μόνο του, χωρίς κάποια δυνατότητα συν-προσομοίωσης. Ο χρήστης μπορεί εποπτικά να ελέγξει το FMU όσον αφορά τον ορισμό των μεταβλητών του, ενώ είναι δυνατή και η απεικόνιση των τιμών της κάθε μεταβλητής κατά τη διάρκεια του συνολικού χρόνου προσομοίωσης (σχήμα 4).



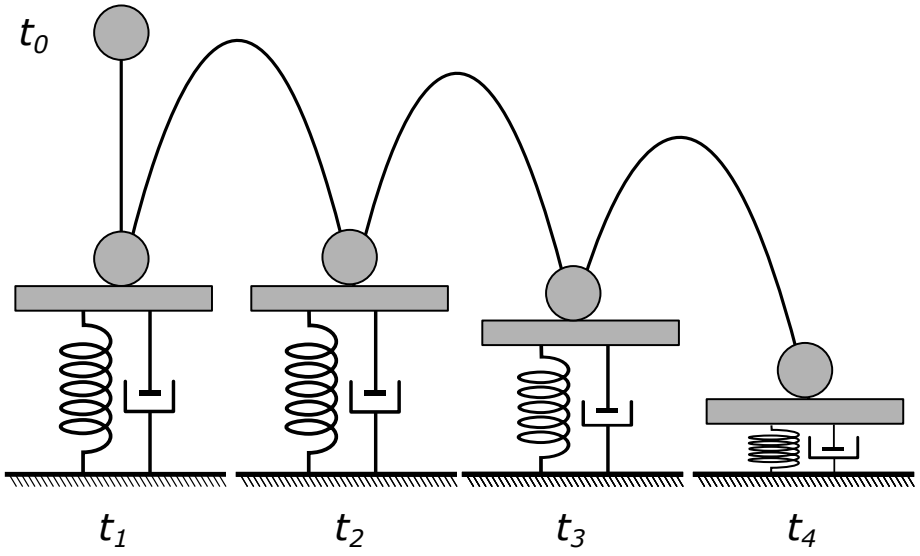
Σχήμα 4: Αυτοτελής προσομοίωση και οπτικοποίηση αποτελεσμάτων των FMU μέσω του FMU Simulator.

Στα πλαίσια της συγκεκριμένης εργασίας, έγινε χρήση του εργαλείου κατά τη διάρκεια της εκπόνησης του κώδικα αυτόματης παραγωγής των FMU. Το FMU Simulator αναγράφει όλες τις μεταβλητές που ορίζονται στο αρχείο *.xml* ενός FMU και εκτελεί την αυτόνομη προσομοίωσή του, διαδικασία που επιτρέπει την εύρεση λαθών και προβλημάτων.

4 Το δυναμικό πρόβλημα

Το πρόβλημα που εξετάζεται αποτελείται από ένα δυναμικό σύστημα μιας σφαίρας που κινείται κατακόρυφα υπό την επίδραση της βαρύτητας και μιας οριζόντιας επιφάνειας συνδεδεμένης μέσω ελατηρίου και αποσβεστήρα με το έδαφος. Η σφαίρα προσκρούει στην οριζόντια επιφάνεια και αναπηδά μέχρις ότου να ισορροπήσει πάνω της και το φαινόμενο ολοκληρώνεται με την απόσβεση της ταλάντωσης της επιφάνειας.

Πρόκειται για ένα σύστημα που συνδυάζει δύο επιμέρους υπο-προβλήματα: την επίλυση της ελεύθερης πτώσης της σφαίρας υπό την επίδραση της βαρύτητας και την επίλυση της επαφής που συμβαίνει μεταξύ της σφαίρας και της ταλαντούμενης επιφάνειας. Το σύστημα διαιρείται στα δυο προαναφερθέντα υποσυστήματα, το καθένα εκ των οποίων θα αντιμετωπιστεί ξεχωριστά.



Σχήμα 5: Σχηματική απεικόνιση του δυναμικού προβλήματος της αναπηδούμενης σφαίρας.

4.1 Προσομοίωση

Το προς εξέταση πρόβλημα μπορεί να προσομοιωθεί εξ ολοκλήρου σαν ένα πρόβλημα δυναμικής πολλαπλών σωμάτων, το οποίο θα παρουσιαστεί και στη συνέχεια. Είτε πρόκειται όμως για απλή προσομοίωση, είτε για συν-προσομοίωση, σημαντικό παράγοντα του προβλήματος αποτελεί ο τρόπος με τον οποίο θα προσομοιωθεί η επαφή κατά την κρούση των δύο σωμάτων, αντικείμενο που συνεχώς μελετάται και εξελίσσεται από

τους μηχανικούς. Ο τρόπος με τον οποίο θα μοντελοποιηθεί η επαφή θα πρέπει να είναι κοινός ανάμεσα στους δυο τρόπους επίλυσης, αλλά και κοινός μεταξύ των δυο διαφορετικών υποσυστημάτων στην περίπτωση της συν-προσομοίωσης.

4.1.1 Προσομοίωμα επαφής σε MBD

Για τον υπολογισμό μιας επαφής μεταξύ σωμάτων απαιτείται η γεωμετρική απεικόνισή τους. Στο Motionview αυτό είναι εφικτό μέσω της ανάθεσης βασικών γεωμετρικών σχημάτων πάνω στα δημιουργημένα σώματα του μοντέλου. Το λογισμικό αναγνωρίζει τους κλειστούς όγκους που ορίζονται από τα γραφικά σχήματα και χρησιμοποιεί το μοντέλο επαφής κατά Hertz, προκειμένου να οριστεί η γεωμετρική επαφή τους [18].

Στο συγκεκριμένο πρόβλημα, η σφαίρα και το ορθογώνιο παραλληλεπίπεδο είναι τα γεωμετρικά σχήματα που αναπαριστούν τα σώματα που λαμβάνουν μέρος στην επαφή. Δεν απαιτείται κάποια σύνθετη γεωμετρική αναπαράσταση και αυτό προσδίδει ακρίβεια στην επαφή και συγκεκριμένα στη μέτρηση της διείσδυσης του ενός σώματος στο άλλο. Ο αλγόριθμος analytical contact μπορεί εύκολα να εντοπίσει την επαφή και να υπολογίσει το βάθος και το ρυθμό διείσδυσης, αλλά και την κάθετη δύναμη που εμφανίζεται μεταξύ των σωμάτων.

Το μοντέλο επαφής που χρησιμοποιείται είναι το *Impact*, σύμφωνα με το οποίο η κάθετη δύναμη επαφής αναλύεται σε μια συνιστώσα δύναμη ελατηρίου και μια συνιστώσα δύναμη απόσβεσης. Το μέτρο της υπολογίζεται από την εξίσωση:

$$F_{contact} = F_{spring} + F_{damping} = \begin{cases} K \cdot z^{exp} - STEP(z, -d_{max}, c, 0, 0) \cdot \frac{dz}{dt} & , z > 0 \\ 0 & , z \leq 0 \end{cases}, \tag{11}$$

όπου:

- K η στιβαρότητα της επαφής
- z το βάθος διείσδυσης
- exp ο εκθέτης της δύναμης ελατηρίου
- d_{max} το βάθος διείσδυσης στο οποίο εφαρμόζεται η μέγιστη δύναμη απόσβεσης
- c ο συντελεστής απόσβεσης.

Το βάθος διείσδυσης z και ο ρυθμός διείσδυσης $\frac{dz}{dt}$ είναι συναρτήσεις του χρόνου υπολογίζονται από τις ακόλουθες εξισώσεις. Τα πρόσημα τον όρων της εξίσωσης (13) είναι αντίθετα από το αναμενόμενο, για λόγους συμβατότητας με τα πρόσημα της εξίσωσης (11).

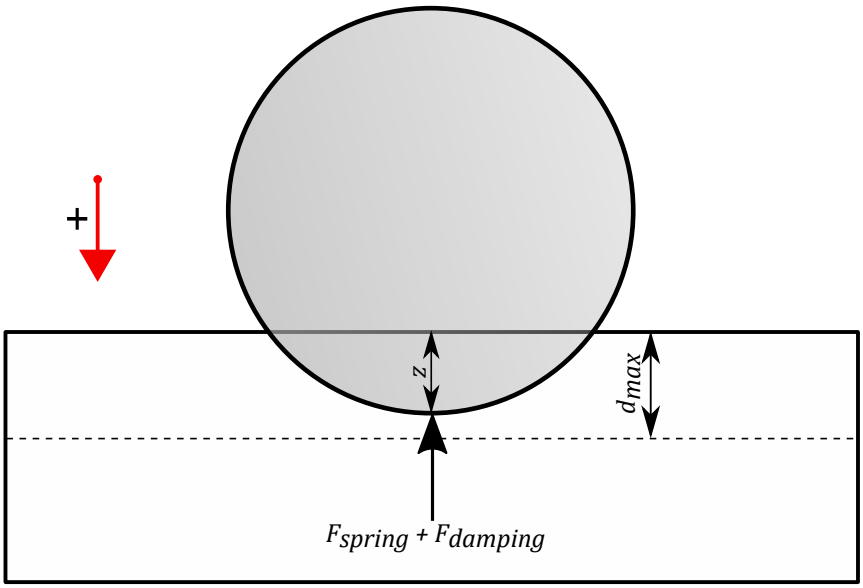
$$\begin{cases} z(t) = r - (y_{ball}(t) - y_{surface}(t)) & (12) \\ \dfrac{dz}{dt}(t) = \dot{y}_{ball} - \dot{y}_{surface} & (13) \end{cases}$$

Ως *STEP* ορίζεται μια βηματική συνάρτηση η οποία για τη δεδομένη διείσδυση επαφής z λαμβάνει τιμή από $(-d_{max}, c)$ έως $(0, 0)$. Με αυτό τον τρόπο, ανάλογα με τη διείσδυση, ελέγχεται η συνιστώσα δύναμη απόσβεσης της επαφής, παίρνοντας ομαλά τιμές μεταξύ των ορίων αυτών. Το μέτρο της υπολογίζεται από την εξίσωση:

$$STEP(x, x_0, y_0, x_1, y_1) = \begin{cases} y_0 & , x \leq x_0 \\ y_0 + \Delta y \tilde{x}^2(3 - 2\tilde{x}) & , x_0 < x < x_1 \\ y_1 & , x \geq x_1 \end{cases}, \quad (14)$$

όπου:

- $\Delta y = (y_1 - y_0)$
- $\tilde{x} = \dfrac{x - x_0}{x_1 - x_0}$.



Σχήμα 6: Σχηματική απεικόνιση του μοντέλου επαφής ανάμεσα σε σφαίρα και επιφάνεια.

4.2 Συν-προσομοίωση

Στη συν-προσομοίωση το κάθε υποσύστημα λειτουργεί ξεχωριστά, δηλαδή οι εξισώσεις του καθενός επιλύονται με το εκάστοτε βήμα ολοκλήρωσης. Ο τρόπος με τον οποίο αυτά συνδέονται είναι μέσω ενός κοινού χρονικού βήματος επικοινωνίας H.

Κάθε χρονικό διάστημα H πραγματοποιείται η επικοινωνία, δηλαδή η ανταλλαγή πληροφοριών μεταξύ των υποσυστημάτων. Το κάθε ένα θα πρέπει να είναι σε θέση να δώσει στο άλλο την εσωτερική του κατάσταση X_i μέσω των μεταβλητών εξόδου Y_i. Έπειτα, κάθε προσομοιωτής προχωράει κατά το εσωτερικό του βήμα, μέχρι να συναντήσει το επόμενο βήμα επικοινωνίας. Κατά το διάστημα αυτό γίνεται χρήση συναρτήσεων παρεμβολής προκειμένου να προσεγγιστεί η συμπεριφορά του άλλου υποσυστήματος στο ίδιο διάστημα, χρησιμοποιώντας την πληροφορία που λήφθηκε από τις εισόδους U_i στο προηγούμενο σημείο επικοινωνίας. Τα παραπάνω συνοψίζονται στον ορισμό ενός συστήματος προσομοίωσης συνεχούς χρόνου S_i, που αποτελεί δηλαδή τα περιεχόμενα ενός FMU [2]:

$$\begin{cases} S_i = \langle X_i, U_i, Y_i, \delta_i, \lambda_i, x_i(0), \phi_{U_i} \rangle \\ \delta_i : \Re \times X_i \times U_i \to X_i \\ \lambda_i : \Re \times X_i \times U_i \to Y_i \quad \text{ή} \quad \Re \times X_i \to Y_i \\ x_i(0) \in X_i \\ \phi_{U_i} : \Re \times U_i \times ... \times U_i \to U_i \end{cases}, \tag{15}$$

όπου:

- X_i οι μεταβλητές του υποσυστήματος, στον $\Re^n$
- U_i οι μεταβλητές εισόδου του υποσυστήματος, στον $\Re^m$
- Y_i οι μεταβλητές εξόδου του υποσυστήματος, στον $\Re^p$
- $x_i(0)$ οι αρχικές συνθήκες
- δ_i ένας ενεργοποιητής για τον υπολογισμό της συμπεριφοράς από τη χρονική στιγμή t έως την $t + H$, χρησιμοποιώντας τη συνάρτηση παρεμβολής ϕ_{U_i}
- $\lambda_i(t, x_i(t), u_i(t)) = y_i(t)$ ή $\lambda_i(t, x_i(t)) = y_i(t)$ η συνάρτηση εξόδου.

4.2.1 Αλγόριθμος συν-προσομοίωσης

Για να πραγματοποιηθεί η συν-προσομοίωση πρέπει να υπάρξει ένας αλγόριθμος «ενορχήστρωσης» των ενεργειών, που αποκαλειται και αλγόριθμος "master". Ο ρόλος του είναι να εκκινήσει την επίλυση, να φέρει σε επικοινωνία τα FMU μεταξύ τους και να καθοδηγήσει την ανταλλαγή δεδομένων. Τα FMU μπορούν να επικοινωνήσουν μόνο με τον master και υπό το συγκεκριμένο πλαίσιο, αποκαλούνται και "slaves". Ο master μπορεί να είναι κάποιο αυτόνομο λογισμικό ή ακόμα και κομμάτι κώδικα που εμπεριέχεται μέσα σε κάποιο από τα FMU που προσομοιώνονται.

Το πρότυπο FMI δεν ορίζει κάποιο συγκεκριμένο αλγόριθμο master, με στόχο να υπάρχει ευελιξία στις ενέργειες που μπορεί να εκτελέσει. Για παράδειγμα, ο πιο κοινός αλγόριθμος master σταματάει τη συν-προσομοίωση όλων των slave στα σημεία επικοινωνίας t_i, συλλέγει τα outputs $y(t_i)$, υπολογίζει τα inputs $u(t_i)$ που αντιστοιχούν στον κάθε slave και τους τα διανέμει κι έπειτα συνεχίζει τη συν-προσομοίωση μέχρι το επόμενο σημείο επικοινωνίας $t_i \rightarrow t_{i+1} = t_i + H$. Οι πιο απλοί αλγόριθμοι προσεγγίζουν τη συμπεριφορά των slave μέχρι το επόμενο σημείο επικοινωνίας t_{i+1} κρατώντας σταθερές τις τιμές των input u από το προηγούμενο σημείο επικοινωνίας t_i.

Το πόσο πολύπλοκος μπορεί να γίνει ένας αλγόριθμος master εξαρτάται από τα χαρακτηριστικά των FMU που ελέγχει, τα οποία όπως προαναφέρθηκε εμπεριέχονται μέσα στο αρχείο *.xml* που συνοδεύει το FMU και το περιγράφει. Τυπικά παραδείγματα είναι:

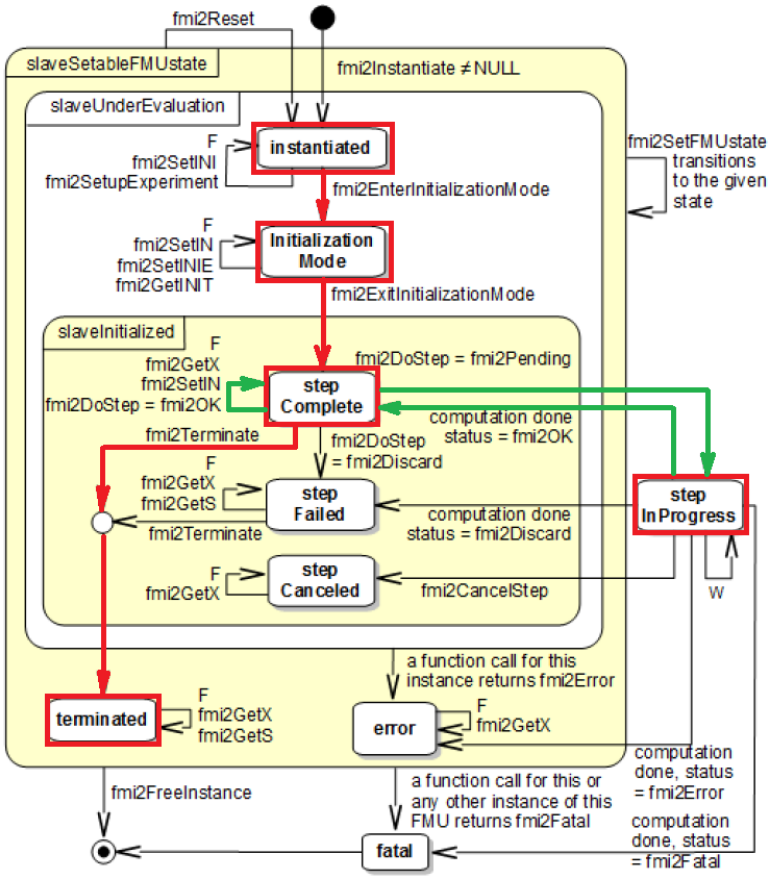
- η ικανότητα χειρισμού μεταβλητού βήματος επικοινωνίας H
- η δυνατότητα επανάληψης βήματος επικοινωνίας που απορρίφθηκε
- η ικανότητα να υπολογίζει μερικές χρονικές παραγώγους των output u_i για τη δημιουργία. συναρτήσεων παρεμβολής.

Η ροή των ενεργειών που υποδεικνύει ο αλγόριθμος συνοψίζεται σε 4 φάσεις, κατά τη διάρκεια των οποίων ο master μπορεί να υλοποιήσει συγκεκριμένες εντολές:

1. Instantiation - Το FMU «πραγματώνεται», δηλαδή προετοιμάζεται για τη συν-προσομοίωση και σε αυτή τη φάση ορίζονται οποιεσδήποτε αρχικές μεταβλητές
2. Initialization - Το FMU «αρχικοποιείται», δηλαδή ενεργοποιούνται οι δίαυλοι επικοινωνίας και επαληθεύονται οι μεταβλητές input και output

3. Step mode - Η συν-προσομοίωση ξεκινάει και οι slaves εκτελούν τον υπολογισμό για το εσωτερικό τους βήμα, υπολογίζουν τα output και φτάνοντας στο επόμενο σημείο επικοινωνίας εκτελούν την ανταλλαγή δεδομένων. Αυτή είναι η κύρια φάση της συν-προσομοίωσης κατά την οποία επαναλαμβάνεται η παραπάνω διαδικασία μέχρι την ολοκλήρωση του συνολικού χρόνου

4. Termination - Με το πέρας του τελικού χρόνου ολοκληρώνεται η συν-προσομοίωση και οι slaves τερματίζουν τους υπολογισμούς τους και «καταστρέφονται».



Σχήμα 7: Βρόχος συν-προσομοίωσης (προσαρμογή από [5]). Με πράσινες γραμμές συμβολίζεται ο βρόχος μεταξύ σημείων επικοινωνίας και με κόκκινες συμβολίζεται η κύρια ροή ενεργειών.

Οι παραπάνω φάσεις γίνονται εμφανείς και στο πληρέστερο σχήμα 7, το οποίο απεικονίζει την κατάσταση την οποία διαχειρίζεται ο master αλγόριθμος κατά τη διάρκεια της συν-προσομοίωσης. Με πράσινες γραμμές

συμβολίζεται ο βρόχος ανάμεσα στα σημεία επικοινωνίας, τον υπολογισμό input και output των slave και την ανταλλαγή πληροφοριών μεταξύ τους. Επιπλέον περιπτώσεις εμφανίζονται όταν προκύπτουν σφάλματα, όπως η αποτυχία βήματος και η απόρριψη βήματος [5].

4.2.2 Επιλογή αλγορίθμου master

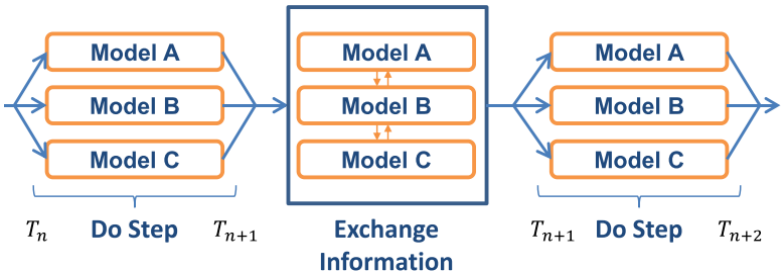
Όπως είναι εμφανές, η επιλογή του αλγορίθμου master που θα εκτελέσει τη συν-προσομοίωση είναι μείζονος σημασίας για την ακρίβεια των αποτελεσμάτων. Στην προηγούμενη ενότητα περιγράφηκε συνοπτικά ο πιο γενικός αλγόριθμος παράλληλης εκτέλεσης των υποσυστημάτων. Η προσέγγιση αυτή συναντάται στη βιβλιογραφία με τον όρο *"Jacobi-like"* [21]. Τα βήματα εκτέλεσής του σε μορφή ψευδοκώδικα είναι τα εξής:

Αλγόριθμος 1: Αλγόριθμος Master - συν-προσομοίωση παράλληλης εκτέλεσης υποσυστημάτων (*"Jacobi-like"*).

Δεδομένα: $j = 1, ..., M$ υποσυστήματα
$u = f(y)$ οι συνδέσεις τους
H το βήμα επικοινωνίας

Όσο $t_i \leq t_{final}$ **επανάλαβε**:
  Για $j = 1$ ***έως*** M **επανάλαβε**:
    Θέσε τα inputs του j μοντέλου $u^{[j]}(t_i)$
    Εκτέλεσε χρονικό βήμα για το j μοντέλο μέχρι το επόμενο σημείο επικοινωνίας $t_i \rightarrow t_{i+1}$ (Υπολόγισε $u^{[j]}(t_{i+1})$)
  τέλος
  Για $j = 1$ ***έως*** M **επανάλαβε**:
    Συγκέντρωσε τα outputs του j μοντέλου $y^{[j]}(t_{i+1})$
  τέλος
  Εκτέλεσε την ανταλλαγή inputs για όλα τα μοντέλα:
  $u(t_{i+1}) = f(y(t_{i+1}))$
  $t_i += H$
τελος

Στα πλαίσια της παρούσας εργασίας έγινε χρήση της JModelica.org ως πλατφόρμα προσομοίωσης και συγκεκριμένα της βιβλιοθήκης της PyFMI, όπως προαναφέρθηκε και στην υποενότητα 3.2.1. Η PyFMI αποτελεί μια συλλογή από αλγορίθμους και προγραμματιστικά εργαλεία που αποσκοπούν στην εύκολη χρήση και συν-προσομοίωση των FMU. Είναι πλήρως συμβατή με το πρότυπο FMI, ενώ η διεπαφή με το χρήστη γίνεται μέσω γλώσσας προγραμματισμού Python.



Σχήμα 8: Παράλληλη εκτέλεση συν-προσομοίωσης (από [21]).

Ο παραπάνω αλγόριθμος είναι ήδη υλοποιημένος και έτοιμος για χρήση από την PyFMI. Όπως αναφέρθηκε, η βιβλιοθήκη είναι ανοικτού κώδικα και όλα τα αρχεία «πηγαίου κώδικα (αγγλ. source files)» μπορούν να βρεθούν ελεύθερα στο διαδίκτυο[2]. Στην πράξη, ο πλήρης αλγόριθμος Master της PyFMI διαθέτει μια πληθώρα επιλογών που τροποποιούν τις βασικές εντολές του και προσφέρουν περισσότερο έλεγχο στις παραμέτρους της συν-προσομοίωσης. Τέτοιες επιλογές είναι:

Πίνακας 1: Επιλογές συν-προσομοίωσης του αλγορίθμου master.

Επιλογή	**Περιγραφή**
step size	Βήμα επικοινωνίας όταν επιλέγεται να είναι σταθερό
extrapolation order	Βαθμός παρεμβολής των inputs στο διάστημα του βήματος επικοινωνίας
execution	Παράλληλη ή σειριακή εκτέλεση
smooth coupling	Εξομάλυνση της παρεμβολής των inputs εάν *extrapolation order* > 0
error controlled	Δυνατότητα μεταβολής του βήματος επικοινωνίας του αλγορίθμου
linear correction	Διόρθωση των outputs κατά την ανταλλαγή τους
atol	Απόλυτη ακρίβεια σφάλματος στην περίπτωση *error controlled*
rtol	Σχετική ακρίβεια σφάλματος στην περίπτωση *error controlled*
num threads	Αριθμός πυρήνων στην παράλληλη επίλυση
result handling	Μορφή εκτύπωσης αποτελεσμάτων
maxh	Μέγιστο βήμα επικοινωνίας στην περίπτωση *error controlled*

[2]https://github.com/modelon-community/PyFMI/blob/master/src/pyfmi/master.pyx

Όπως είναι εμφανές, ορισμένες από τις επιλογές υποδεικνύουν και έναν διαφορετικό τρόπο επίλυσης σε αντίθεση με το γενικό αλγόριθμο που παρουσιάστηκε προηγουμένως. Με την επιλογή *error controlled*, ο αλγόριθμος εκτελεί τη συν-προσομοίωση στο χρόνο με μεταβλητό βήμα επικοινωνίας και συνεχίζει στην επόμενη στιγμή μόνο εάν η εκτίμηση του σφάλματος για αυτήν βρίσκεται κάτω από κάποιο όριο. Αυτό είναι εφικτό με μια παραλλαγή της παρεμβολής Richardson, σύμφωνα με την οποία, διαπιστώνεται αν το επιλεγμένο βήμα H θα οδηγήσει σε μεγάλο ή μικρό σφάλμα. Ο αλγόριθμος εκτελεί δύο συνεχόμενα βήματα στο χρόνο, με $H = \frac{H}{2}$, υπολογίζοντας τα inputs και outputs στο τέλος κάθε βήματος. Έπειτα ο αλγόριθμος εκτελεί τον υπολογισμό για το ίδιο χρονικό διάστημα με το πλήρες βήμα H και συγκρίνει το σφάλμα των outputs που προκύπτει ανάμεσα στις δυο περιπτώσεις, προκειμένου να απορρίψει ή όχι το χρονικό βήμα [21, 22].

Κατά τη συν-προσομοίωση του δυναμικού προβλήματος της παρούσας εργασίας δεν έγινε χρήση των παραπάνω επιλογών, αφού κρίθηκε άσκοπο να εισαχθούν επιπλέον μεταβλητές που θα επηρεάσουν το αποτέλεσμα. Ο στόχος ήταν να παραμείνουν οι παράμετροι της συν-προσομοίωσης σε ελέγξιμο επίπεδο, μέσω της χρήσης του βασικού αλγορίθμου, προκειμένου να αναδειχθεί η δυνατότητα εφαρμογής της και η εγκυρότητά της. Έτσι, ο αλγόριθμος master επιλέχθηκε να λειτουργεί με σταθερό βήμα επικοινωνίας και σειριακή εκτέλεση. Ειδικά, οι επιλογές του αλγορίθμου που χρησιμοποιήθηκαν είναι οι εξής:

- *execution*: 'serial'
- *error controlled*: False
- *step size* (**H**): $\mathbf{1e-4}$ [**sec**]
- *linear correction*: False
- *extrapolation order*: 0
- *result handling*: 'file'.

4.3 Διαίρεση μοντέλου

Το πρόβλημα χωρίζεται στα επιμέρους υποσυστήματα με τη δημιουργία ενός FMU για το καθένα. Τα δύο FMU θα πρέπει να είναι κατάλληλα διαμορφωμένα προκειμένου να είναι εφικτή η επικοινωνία και η ανταλλαγή δεδομένων μεταξύ τους, όπως περιγράφηκε στην προηγούμενη υποενότητα.

4.3.1 FMU αναπηδούμενης σφαίρας

Το πρώτο υποσύστημα του προβλήματος περιλαμβάνει τη σφαίρα και την κίνησή της. Για τη δημιουργία του FMU είναι απαραίτητη η γνώση της σχετικής θέσης των δυο σωμάτων μεταξύ τους, καθώς και της δύναμης επαφής.

Η κίνηση της σφαίρας μπορεί να προσδιοριστεί και αναλυτικά, εφόσον επιδρά πάνω της μόνο η δύναμη της βαρύτητας και η δύναμη της επαφής, όταν αυτή εμφανίζεται.

Από την εφαρμογή του δεύτερου νόμου του Νεύτωνα στη σφαίρα που εκτελεί ελεύθερη πτώση λαμβάνουμε:

$$\sum \mathbf{F} = m\ddot{\mathbf{r}} \Rightarrow F_c - mg = m\ddot{y} \Rightarrow \ddot{y} = -g + \frac{F_c}{m} \ . \tag{16}$$

Η παραπάνω αποτελεί μια κανονική διαφορική εξίσωση και επιλύεται αριθμητικά με τη μέθοδο *Euler*. Χρησιμοποιώντας τις αρχικές συνθήκες $\dot{y}(0) = v_0$ και $y(0) = y_0$ προκύπτουν οι εξισώσεις ταχύτητας και θέσης του κέντρου μάζας της σφαίρας, λαμβάνοντάς τες τη χρονική στιγμή $n+1$:

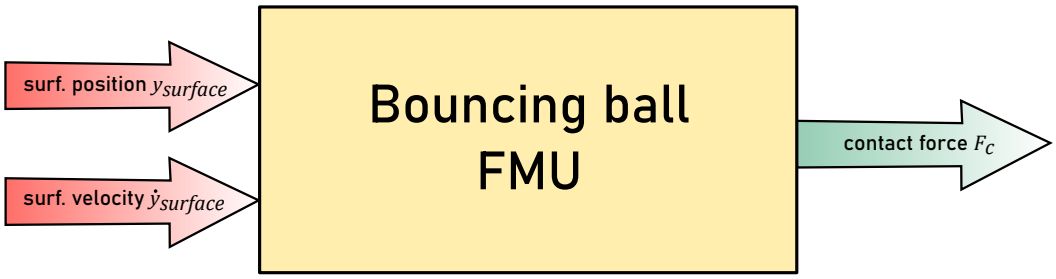
$$\begin{cases} \dot{y}_{n+1} = \dot{y}_n + h \cdot \left(-g + \dfrac{F_{c,n}}{m} \right), \quad n = 0, 1, 2, ... & (17) \\ y_{n+1} = y_n + h \cdot \left[\dot{y}_n + h \cdot \left(-g + \dfrac{F_{c,n}}{m} \right) \right], \quad n = 0, 1, 2, ... \ , & (18) \end{cases}$$

όπου h το χρονικό βήμα ολοκλήρωσης και F_c η δύναμη επαφής τη συγκεκριμένη χρονική στιγμή, όπως αυτή ορίστηκε από την (11).

Αφού είναι γνωστές οι εξισώσεις κίνησης, σειρά έχει η δημιουργία του FMU που θα εμπεριέχει το παραπάνω υποσύστημα του προβλήματος. Αρχικά, για τη δημιουργία του συγκεκριμένου FMU, εκπονήθηκε κώδικας σε γλώσσα προγραμματισμού *C++* με βάση τον οποίο ορίζεται το δυναμικό σύστημα με τις εξισώσεις που πρέπει να επιλυθούν σύμφωνα με το πρότυπο FMI.

Μεταξύ διαδοχικών στιγμών επικοινωνίας κατά τη διάρκεια της συνπροσομοίωσης, το «FMU αναπηδούμενης σφαίρας» (αγγλ. bouncing ball FMU) υπολογίζει την τροχιά της σφαίρας μέσω των εξισώσεων (17) και (18) με εσωτερικό χρονικό βήμα επίλυσης που επιλέχθηκε να είναι μικρότερο από το ολικό βήμα επικοινωνίας, δηλαδή $\mathbf{h = 10^{-7}}$ [**sec**]. Υπενθυμίζεται πως στο διάστημα μεταξύ των στιγμών επικοινωνίας, ο επιλυτής λειτουργεί ανεξάρτητα και χρησιμοποιεί το δικό του χρονικό βήμα και το δικό του σχήμα ολοκλήρωσης. Αυτό σημαίνει πως είναι δυνατή η επίλυση των εξισώσεων κίνησης της σφαίρας με επαρκώς μικρό βήμα, με στόχο την καλύτερη ακρίβεια της λύσης.

Τα **inputs** του FMU είναι η κατακόρυφη *θέση* και *ταχύτητα της επιφάνειας* σε κάθε χρονικό βήμα επικοινωνίας, απαραίτητα για να υπολογιστούν η διείσδυση και ο ρυθμός διείσδυσης της σφαίρας στην επιφάνεια και συνεπώς η δύναμη επαφής, όπως ορίστηκε από την εξίσωση (11). Το **output** του FMU είναι η υπολογισθείσα *δύναμη επαφής* για τη δεδομένη χρονική στιγμή, η οποία είναι απαραίτητη για τον υπολογισμό της θέσης και ταχύτητας της επιφάνειας μέσω του motionsolve FMU, το οποίο θα περιγραφεί στη συνέχεια.



Σχήμα 9: Μεταβλητές εισόδου και εξόδου *(Inputs & Outputs)* του bouncing ball FMU.

Ο κώδικας δημιουργίας του FMU παράγει, όπως ορίζεται από το πρότυπο FMI, ένα αρχείο δυναμικής βιβλιοθήκης δυαδικής μορφής τύπου *.dll*, το οποίο περιέχει τις εξισώσεις του συστήματος, τον ορισμό των μεταβλητών του και την υλοποίηση των συναρτήσεων του προτύπου. Οι συναρτήσεις αυτές καθιστούν ικανό το χειρισμό του FMU από τον master αλγόριθμο στα στάδια της συν-προσομοίωσης που αναφέρθηκαν στην υποενότητα 4.2.1. Στο Παράρτημα Α΄ παρουσιάζεται ένα απόσπασμα του κώδικα παραγωγής FMU. Η συνάρτηση `FmiDoStep` στην Καταγραφή 3 εκτελεί τη χρονική ολοκλήρωση των εξισώσεων κίνησης ανάμεσα στα σημεία επικοινωνίας της συν-προσομοίωσης.

Παράλληλα δημιουργείται και το αρχείο *"Model description"*, δηλαδή το αρχείο που περιέχει μια συνοπτική περιγραφή του μοντέλου και των μεταβλητών του σε μορφή *.xml*, όπως ορίζεται από το πρότυπο [5]. Η κάθε μεταβλητή συνοδεύεται από:

- *name*: όνομα της μεταβλητής
- *description*: βοηθητική περιγραφή της μεταβλητής
- *casuality*: υποδηλώνει αν είναι παράμετρος, input, output κ.ά.
- *variability*: υποδηλώνει αν η τιμή της μεταβάλλεται, είναι σταθερή, συνεχής, διακριτή κ.ά.

- *initial*: υποδηλώνει αν έχει αρχική τιμή σταθερή, υπολογιζόμενη, κ.ά..

Επίσης περιέχεται και η δομή του μοντέλου (*model structure*), δηλαδή ποιες μεταβλητές είναι αρχικά άγνωστες, ποιες αποτελούν παραγώγους άλλων, ποιες είναι inputs και ποιες outputs. Στο bouncing ball FMU ορίστηκαν οι μεταβλητές που αναγράφονται στον πίνακα 2. Στην Καταγραφή 4 παρουσιάζεται το πλήρες αρχείο *ModelDescription.xml* που χρησιμοποιήθηκε για τη δημιουργία του FMU.

Στη συνέχεια δημιουργείται το FMU, το οποίο δεν αποτελεί παρά ένα αρχείο συμπιεσμένης μορφής *.zip* με τροποποιημένη κατάληξη *.fmu*.

Πίνακας 2: Ορισμός μεταβλητών στο αρχείο *modelDescription.xml* του bouncing ball FMU.

Name	**Casuality**	**Variability**	**Initial**	**Description**
h	local	continuous	exact	vertical position of sphere
v	local	continuous	exact	vertical velocity of sphere
g	parameter	fixed	exact	acceleration of gravity
inp	input	-	-	input (vertical position of floor)
out	output	-	-	output (F_c)

4.3.2 FMU ταλαντούμενης επιφάνειας

Το δεύτερο υποσύστημα του προβλήματος περιλαμβάνει την επιφάνεια που αναπηδά η σφαίρα, το ελατήριο και τον αποσβεστήρα στα οποία είναι προσαρτημένη η επιφάνεια και το έδαφος. Για τη δημιουργία του FMU είναι απαραίτητη η γνώση της δύναμης επαφής F_c σε κάθε χρονική στιγμή, πληροφορία που θα έρχεται από το bouncing ball FMU κατά τη διάρκεια της συν-προσομοίωσης.

Το «FMU ταλαντούμενης επιφάνειας» (αγγλ. motionsolve FMU) δεν περιλαμβάνει το σώμα της σφαίρας, αλλά μόνο τη δύναμη επαφής που προκύπτει από την αναπήδηση της σφαίρας, η οποία προκαλεί και την ταλάντωση της επιφάνειας. Η δύναμη εφαρμόζεται στο κέντρο μάζας της επιφάνειας, ενώ επίσης προστίθεται και ένας δεσμός κίνησης που περιορίζει την κίνησή της στον κατακόρυφο άξονα, υπό τη μορφή του "*Translational Joint*" του Motionsolve.

Για να είναι δυνατή η επικοινωνία του FMU κατά τη συν-προσομοίωση δημιουργούνται οι μεταβλητές input και output υπό τη μορφή του "*Solver Variable*" του Motionsolve. Για την **input** εισάγεται ως εξίσωση της μεταβλητής η δύναμη στην κατακόρυφη διεύθυνση (άξονας Z στο μοντέλο) που

εφαρμόζεται στο κέντρο μάζας της επιφάνειας, δηλαδή:

$$sv_{input} = F_z^{cm}(t) \ .$$

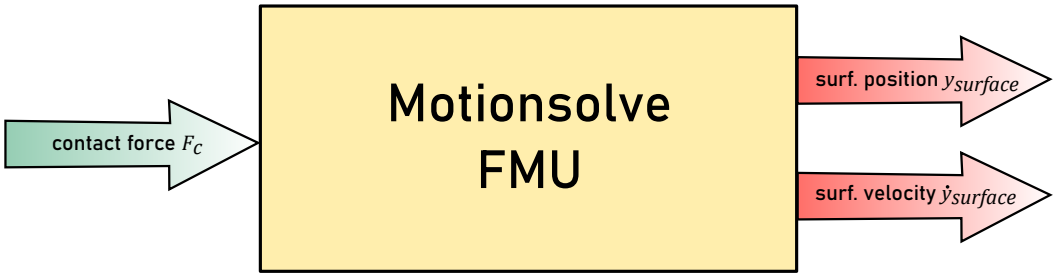
Για την πρώτη μεταβλητή **output** εισάγεται ως εξίσωση μεταβλητής η κατακόρυφη συντεταγμένη του κέντρου μάζας της επιφάνειας ως προς το αδρανειακό σύστημα συντεταγμένων του εδάφους δηλαδή:

$$sv_{output1} = z_{cm}(t) + \frac{width}{2} \ .$$

Σε αυτήν προστίθεται το μισό πάχος της επιφάνειας, προκειμένου να υπολογίζεται σωστά η δύναμη επαφής από το bouncing ball FMU κατά τη συν-προσομοίωση. Για τη δεύτερη μεταβλητή **output** εισάγεται ως εξίσωση μεταβλητής αντίστοιχα η κατακόρυφη ταχύτητα του κέντρου μάζας της επιφάνειας, δηλαδή:

$$sv_{output2} = \dot{z}_{cm}(t) \ .$$

Επίσης ορίζονται και δύο *"Solver Array"*, δηλαδή λίστες μεταβλητών που περιέχουν τις solver variables για το input και το output. Για τη μεταβλητή sv_{input} ορίζεται ένα solver array τύπου *Plant Input* και για τη μεταβλητή sv_{output} τύπου *Plant Output*. Με αυτά υποδηλώνεται ότι οι μεταβλητές sv_{input} και sv_{output} αποτελούν τις μεταβλητές εισόδου u_i και εξόδου y_i του συστήματος.



Σχήμα 10: Μεταβλητές εισόδου και εξόδου *(Inputs & Outputs)* του motionsolve FMU.

Έπειτα, για την υλοποίηση του FMU, το MotionView διαθέτει την επιλογή *"Export model as FMU"*, το οποίο δημιουργεί και εξάγει απευθείας τα αρχεία *.dll* και *ModelDescription.xml* και τα συμπιέζει στην επιθυμητή μορφή *.fmu*, επιταχύνοντας τη διαδικασία.

5 Αριθμητική επίλυση - Αποτελέσματα

Προκειμένου να αξιολογηθεί η αποτελεσματικότητα της συν - προσομοίωσης του δυναμικού προβλήματος που παρουσιάστηκε προηγουμένως καταστρώθηκαν τρία διαφορετικά μοντέλα. Αρχικά παρουσιάζεται ένα παράδειγμα της αυτοτελούς χρήσης του bouncing ball FMU, χρησιμοποιώντας εξωτερικό σήμα input για την απόκριση της επιφάνειας που προσκρούει η σφαίρα, απ' ευθείας με τη γλώσσα Modelica. Έπειτα παρουσιάζεται η αυτοτελής χρήση του motionsolve FMU, πάλι με εξωτερικό σήμα input για τη δύναμη που εφαρμόζεται στην επιφάνεια. Τέλος, παρουσιάζεται η συν-προσομοίωση και των δύο FMU, συγκρίνοντας τα αποτελέσματα με την προσομοίωση με MBD.

Η γεωμετρία του προβλήματος χαρακτηρίζεται από τα μεγέθη του πίνακα 3. Οι τιμές αναλογούν σε μία κανονική μπάλα του μπάσκετ που αφήνεται να πέσει ελεύθερα από ύψος h και προσκρούει σε απαραμόρφωτο έδαφος.

Πίνακας 3: Δεδομένα προβλήματος.

$h(0) =$	$5\ [m]$	ύψος εκκίνησης της σφαίρας
$v(0) =$	$0\ [m/s]$	ταχύτητα εκκίνησης της σφαίρας
$g =$	$9.81\ [m/s^2]$	επιτάχυνση της βαρύτητας
$m =$	$0.62\ [kg]$	μάζα της σφαίρας
$r =$	$0.12\ [m]$	ακτίνα της σφαίρας
$K =$	$1e+5\ [N/m]$	στιβαρότητα δύναμης ελατηρίου επαφής
$exp =$	2.1	εκθέτης δύναμης ελατηρίου επαφής
$d_{max} =$	$0.08\ [m]$	βάθος διείσδυσης που εφαρμόζεται μέγιστη δύναμη απόσβεσης επαφής
$c =$	10	συντελεστής απόσβεσης δύναμης επαφής
$length =$	$1\ [m]$	μήκος πλευράς τετράγωνης επιφάνειας
$width =$	$0.1\ [m]$	πάχος τετράγωνης επιφάνειας
$M =$	$10\ [kg]$	μάζα επιφάνειας
$k =$	$2.0836e+4\ [N/m]$	στιβαρότητα ελατηρίου επιφάνειας
$C =$	$25\ [Ns/m]$	αποσβεστήρας επιφάνειας

5.1 Επαλήθευση bouncing ball FMU

Στο πρώτο μοντέλο γίνεται χρήση του bouncing ball FMU το οποίο δημιουργείται όπως περιγράφηκε στην υποενότητα 4.3.1. Προκειμένου να εξακριβωθεί η ικανότητά του να λειτουργήσει αποτελεσματικά στη

συν-προσομοίωση με το motionsolve FMU, δοκιμάζεται αρχικά σε συν-προσομοίωση με ένα απλό μοντέλο σήματος.

Το μοντέλο σήματος περιέχει μια συνάρτηση η οποία αντιπροσωπεύει τη θέση της επιφάνειας του πλήρους δυναμικού προβλήματος. Αποτελεί δηλαδή ένα απλουστευμένο motionsolve FMU, το οποίο όμως περιέχει μόνο μια συνάρτηση. Δεδομένης της απλής υπόστασής του, η δημιουργία του είναι εφικτή μέσω της γλώσσας προγραμματισμού Modelica. Σημειώνεται πως για το συγκεκριμένο παράδειγμα, η επιφάνεια επιλέχθηκε να παραμένει σταθερή για λόγους απλούστευσης του προβλήματος. Το κύριο δυναμικό φαινόμενο της αναπήδησης της σφαίρας στην επιφάνεια συνεχίζει να υφίσταται.

Δημιουργείται ένα αρχείο μορφής *.mo* στο οποίο ορίζεται η συνάρτηση που δίνει το σήμα εξόδου του FMU, όπως επίσης ορίζονται και οι συνδέσεις (connections). Εδώ γίνεται εμφανής η ευχέρεια που παρέχει η γλώσσα Modelica, με τις λέξεις-κλειδιά που επιτρέπουν απ' ευθείας τον ορισμό των παραπάνω εννοιών [8].

Καταγραφή 1: Κώδικας μοντέλου απόκρισης επιφάνειας σε γλώσσα *Modelica*, *"Surface.mo"*

```
model Surface "Surface"

    connector InputConnector
        input Real value;
    end InputConnector;
    InputConnector ci;

    connector OutputConnector
        output Real value;
    end OutputConnector;
    OutputConnector co;

    connector OutputConnector
        output Real value;
    end OutputConnector;
    OutputConnector co2;

equation
    co.value = 0.0;
    co2.value = 0.0;
end Surface;
```

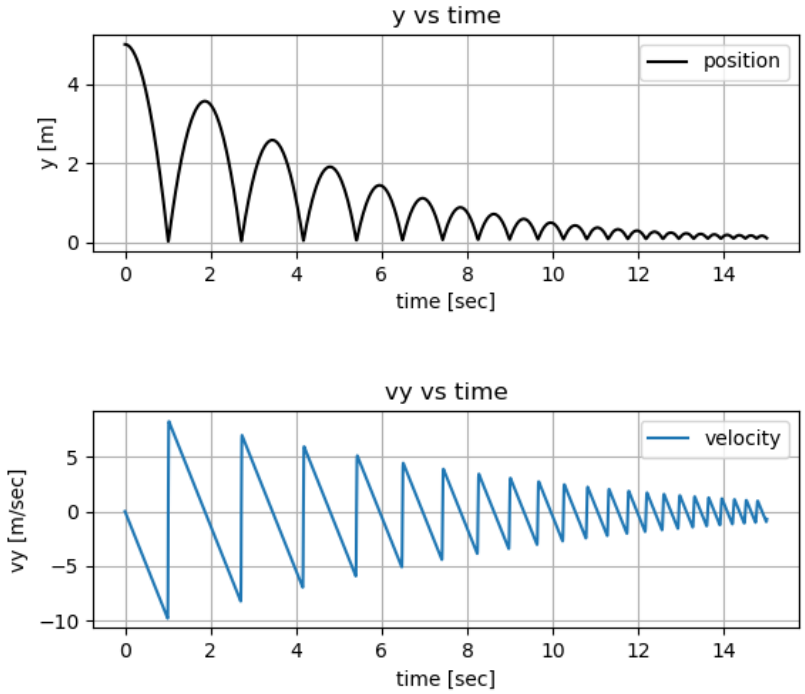
Η μεταβλητές εξόδου co και $co2$ έχουν σταθερή τιμή και ίση με το 0 και αντιπροσωπεύουν τη σταθερή, στη συγκεκριμένη επίλυση, επιφάνεια με μηδενική ταχύτητα. Το μοντέλο αυτό έπειτα διαμορφώνεται αυτόματα ως

FMU για συν-προσομοίωση (με την ονομασία “surface FMU”), πάλι μέσω της βιβλιοθήκης PyFMI, με την εντολή `compile_fmu` ως εξής:

```
compile_fmu('Surface', 'Surface.mo', compile_to="Surface.fmu",
            target='cs')
```

5.1.1 Αριθμητική επίλυση

Η συν-προσομοίωση που θα εκτελεστεί μεταξύ του surface FMU και του bouncing ball FMU θα πρέπει να έχει ως σημείο αναφοράς μια επίλυση του ίδιου προβλήματος με αριθμητικό τρόπο. Για το σκοπό αυτό εκπονήθηκε κώδικας σε γλώσσα *Python* ο οποίος επιλύει το πρόβλημα της σφαίρας που εκτελεί ελεύθερη πτώση από ηρεμία και προσκρούει σε σταθερό και απαραμόρφωτο έδαφος. Τα δεδομένα του προβλήματος παραμένουν ίδια, όπως ορίστηκαν στον πίνακα 3. Τα αποτελέσματα για τελικό χρόνο $t_{final} = 15\ [sec]$ και βήμα $h = 1e-5\ [sec]$ είναι τα εξής:



Σχήμα 11: Αριθμητική επίλυση αναφοράς για τη συν-προσομοίωση της αναπηδούμενης σφαίρας σε ακίνητο έδαφος.

5.1.2 Συν-προσομοίωση

Για την εκτέλεση της συν-προσομοίωσης με τη βιβλιοθήκη PyFMI, εισάγονται αρχικά τα δύο FMU με την εντολή `load_fmu` ως εξής:

```
m1 = load_fmu('bouncingball.fmu')
m2 = load_fmu('Surface.fmu')
```

Έπειτα ορίζονται οι συνδέσεις των μεταβλητών εισόδου και εξόδου τους και το συζευγμένο πλέον σύστημα ως εξής:

```
models = [m1, m2]
connections = [
    (m2, "co.value", m1, "inp"),
    (m2, "co2.value", m1, "inp2"),
    (m1, "out", m2, "ci.value")
]
coupled = Master(models, connections)
```

Με τη χρήση της συνάρτησης `Master()` υποδηλώνεται ότι το συζευγμένο σύστημα θα συν-προσομοιωθεί χρησιμοποιώντας τον αλγόριθμο master που ορίστηκε στην υποενότητα 4.2.2. Σε αυτό το σημείο ορίζονται και οι επιλογές του αλγορίθμου (options) που επισημάνθηκαν στην ίδια υποενότητα. Τέλος, εκτελείται η συν-προσομοίωση με την εντολή `simulate()`.

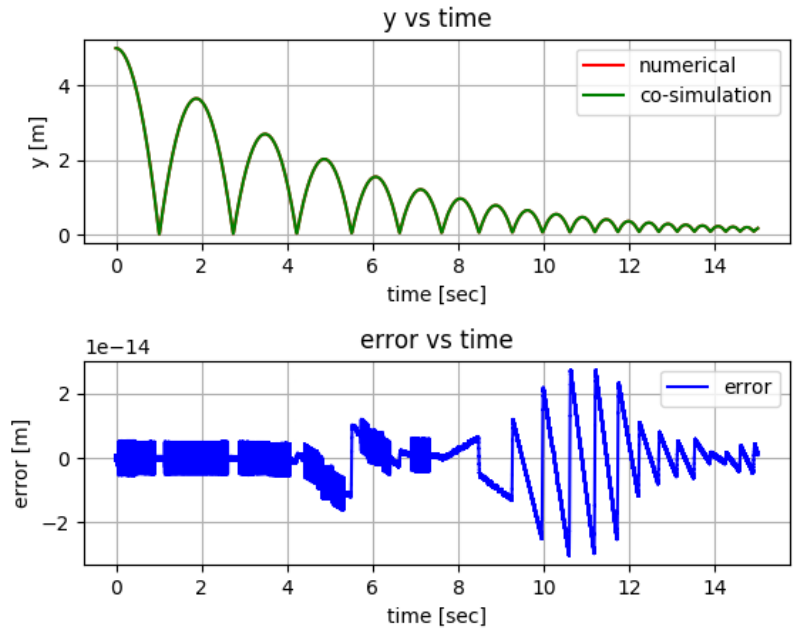
```
opts = coupled.simulate_options()
opts['step_size'] = 0.00001
opts['execution'] = 'serial'
opts['error_controlled'] = False
opts['linear_correction'] = False
opts['extrapolation_order'] = 0
opts['result_handling'] = 'file'
result = coupled.simulate(final_time=15., options=opts)
```

Τα παραπάνω αποτελούν αποσπάσματα του κώδικα συν-προσομοίωσης σε γλώσσα *Python* και παρουσιάζονται συνοπτικά στην Καταγραφή 5.

Τα αποτελέσματα στη συνέχεια συγκρίνονται αναφορικά με αυτά της αναλυτικής προσομοίωσης της προηγούμενης υποενότητας και παρουσιάζονται στο σχήμα 12. Πραγματοποιείται συγκεκριμένα σύγκριση της καμπύλης της κατακόρυφης θέσης της σφαίρας συναρτήσει του χρόνου. Το σφάλμα σε κάθε χρονική στιγμή προκύπτει ως η αφαίρεση των τιμών των δύο καμπυλών, δηλαδή:

$$error = y_{numerical} - y_{cosimulation} \ . \tag{19}$$

Παρατηρούμε πως οι δυο καμπύλες συμπίπτουν απόλυτα και το σφάλμα ανέρχεται σε τιμές της τάξης του $1e-14$ και κάτω.



Σχήμα 12: Συν-προσομοίωση bouncing ball FMU με surface FMU.

5.2 Επαλήθευση motionsolve FMU

Με ανάλογο τρόπο με το bouncing ball FMU θα γίνει και η επαλήθευση του motionsolve FMU. Σε αυτό το παράδειγμα, το motionsolve FMU θα δοκιμαστεί σε συν-προσομοίωση με ένα αντίστοιχο μοντέλο σήματος, το οποίο θα αντιπροσωπεύει τη δύναμη επαφής F_c που εμφανίζεται στο πλήρες δυναμικό πρόβλημα. Το μοντέλο σήματος δημιουργείται πάλι με τη γλώσσα Modelica και περιέχει μία μόνο συνάρτηση εξόδου, που θα αποτελέσει την είσοδο για το motionsolve FMU, καθώς και τις συνδέσεις.

Καταγραφή 2: Κώδικας μοντέλου δύναμης επαφής σε γλώσσα *Modelica*, "*Force.mo*"

```
model Force "Force"

    connector InputConnector
        input Real value;
    end InputConnector;
```

```
    InputConnector ci;

    connector InputConnector
        input Real value;
    end InputConnector;
    InputConnector ci2;

    connector OutputConnector
        output Real value;
    end OutputConnector;
    OutputConnector co;

equation
    co.value = 200 * cos(2 * Modelica.Constants.pi * time);
end Force;
```

Η μεταβλητή εξόδου co έχει συνημιτονοειδή μορφή η οποία εκλέγεται αυθαίρετα και αντιπροσωπεύει μια δύναμη που μεταβάλλεται με το χρόνο. Με αυτό τον τρόπο το μαθηματικό μοντέλο μπορεί να επιλυθεί και αναλυτικά, το οποίο θα αποτελέσει το σημείο αναφοράς με βάση το οποίο θα συγκριθούν τα αποτελέσματα της συγκεκριμένης συν-προσομοίωσης. Το FMU του μοντέλου (με την ονομασία "force FMU") δημιουργείται αντίστοιχα με την εντολή `compile_fmu` ως εξής:

```
compile_fmu('Force', 'Force.mo', compile_to="Force.fmu",
            target='cs')
```

5.2.1 Αναλυτική επίλυση

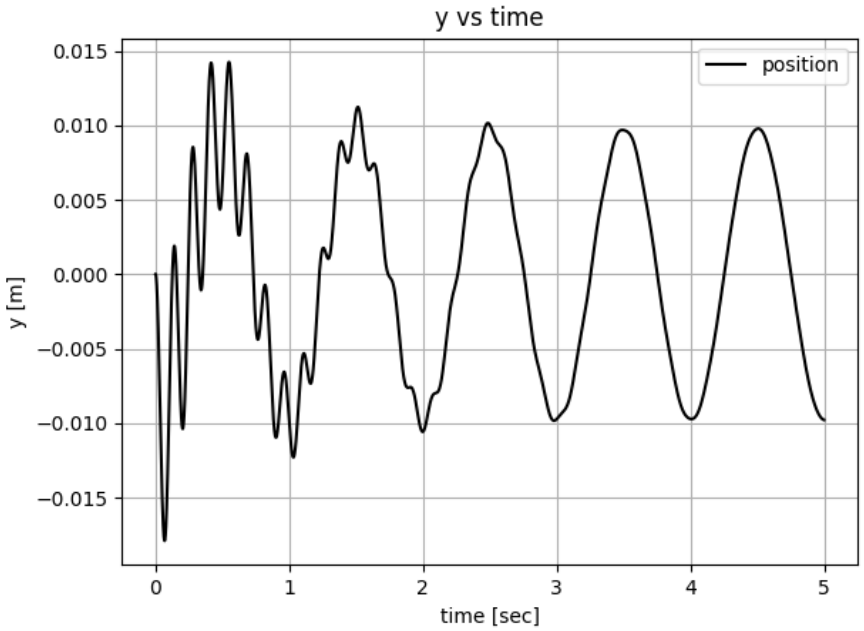
Η συν-προσομοίωση του συγκεκριμένου παραδείγματος ανάμεσα στο motionsolve FMU και το force FMU δεν είναι τίποτε άλλο παρά ένα κλασικό πρόβλημα γραμμικού ταλαντωτή με αρμονική διέγερση με εξίσωση κίνησης $m\ddot{x} + c\dot{x} + kx = f(t)$ και εξίσωση διέγερσης $f(t) = \hat{f}cos(\Omega t) = -200cos(2\pi t)$. Η απόκριση του ταλαντωτή με υποκρίσιμη απόσβεση στην αρμονική διέγερση είναι [19]:

$$x(t) = Ae^{-\delta t}cos\left(\omega_d t - \theta\right) + \hat{x}cos\left(\Omega t - \phi\right) \ , \tag{20}$$

όπου οι σταθερές A και θ υπολογίζονται από τις αρχικές συνθήκες και οι σταθερές $\hat{x}$ και ϕ από τις σχέσεις

$$\hat{x} = \frac{\hat{f}}{\sqrt{\left(k - m\Omega^2\right)^2 + \left(c\Omega\right)^2}} \quad \text{και} \quad tan\phi = \frac{c\Omega}{k - m\Omega^2} \ , 0 \leq \phi < \pi \ . \tag{21}$$

Από τα δεδομένα του πίνακα 3 προκύπτει ότι $\zeta = c/(2\sqrt{km}) \approx 0.0274$, δηλαδή η απόσβεση είναι υποκρίσιμη.



Σχήμα 13: Απόκριση γραμμικού ταλαντωτή σε αρμονική διέγερση.

5.2.2 Συν-προσομοίωση

Όπως και στο προηγούμενο παράδειγμα, για την εκτέλεση της συν-προσομοίωσης εισάγονται αρχικά τα δύο FMU με την εντολή `load_fmu` ως εξής:

```
m1 = load_fmu('Motionsolve.fmu')
m2 = load_fmu('Force.fmu')
```

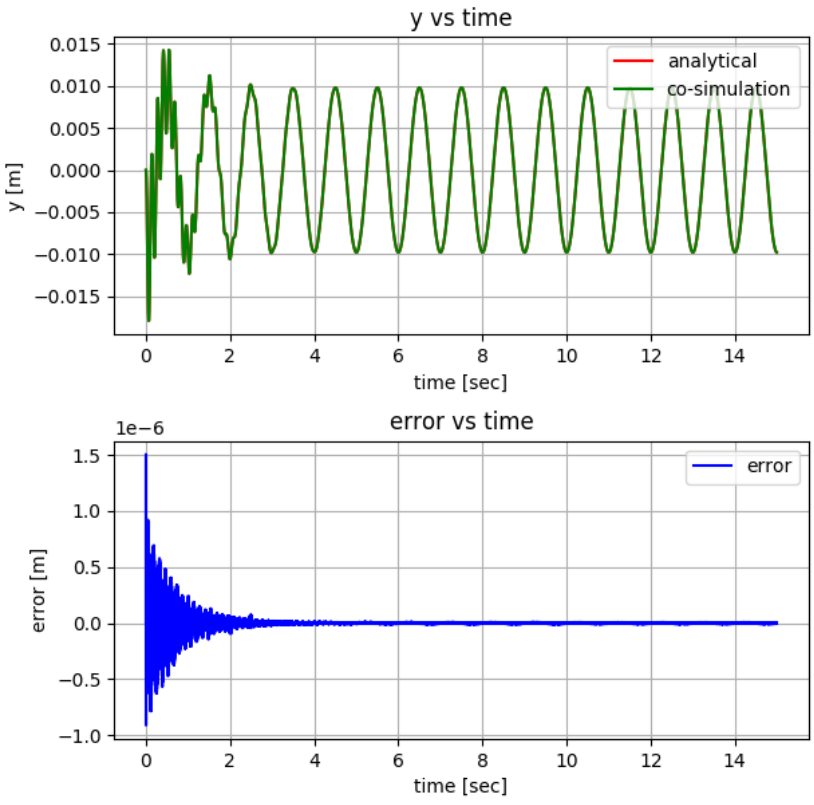
Έπειτα ορίζονται οι συνδέσεις των μεταβλητών εισόδου και εξόδου τους και το συζευγμένο πλέον σύστημα ως εξής:

```
models = [m1, m2]
connections = [
    (m2, "co.value", m1, "inp"),
    (m1, "out", m2, "ci.value"),
    (m1, "sv_output2", m2, "ci2.value")
]
coupled = Master(models, connections)
```

Στη συνέχεια ορίζονται οι επιλογές του αλγορίθμου master και εκτελείται η συν-προσομοίωση με την εντολή `simulate()`.

```
opts = coupled.simulate_options()
opts['step_size'] = 0.0001
opts['execution'] = 'serial'
opts['error_controlled'] = False
opts['linear_correction'] = False
opts['extrapolation_order'] = 0
opts['result_handling'] = 'file'
result = coupled.simulate(final_time=15., options=opts)
```

Αντιστοίχως, το αποτέλεσμα της συν-προσομοίωσης για την απόκριση της επιφάνειας συγκρίνεται με την αναλυτική απόκριση που υπολογίσθηκε στην προηγούμενη υποενότητα και παρουσιάζεται στο σχήμα 14. Για τη σύγκριση γίνεται και πάλι χρήση της σχέσης (19). Παρατηρείται και πάλι ότι οι δυο καμπύλες συμπίπτουν και το σφάλμα ανέρχεται σε τιμές της τάξης του $1.5e-6$ και κάτω.



Σχήμα 14: Συν-προσομοίωση motionsolve FMU με force FMU.

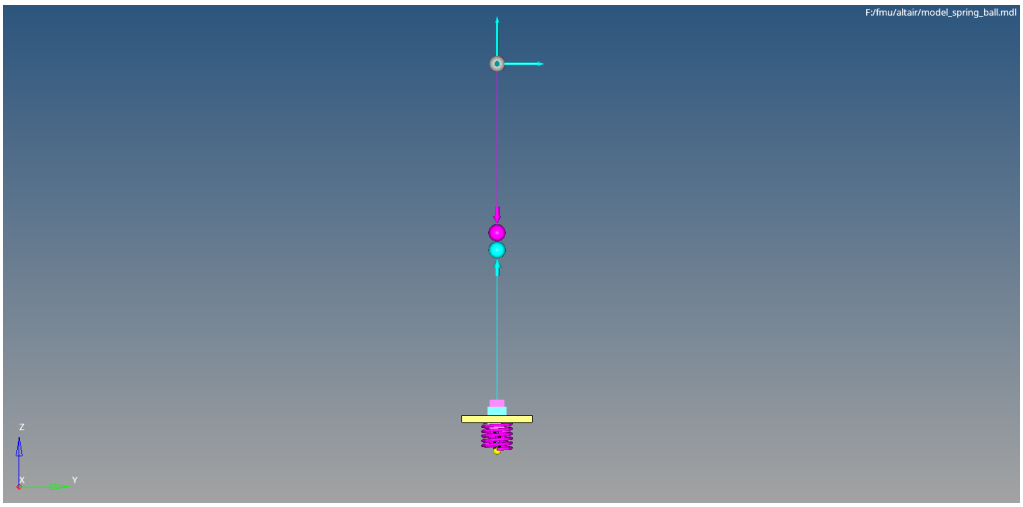
5.3 Συν-προσομοίωση πλήρους προβλήματος

Έχοντας δημιουργήσει και επαληθεύσει το bouncing ball FMU και το motionsolve FMU, σειρά έχει η συν-προσομοίωση του πλήρους δυναμικού προβλήματος, που αποτελεί άλλωστε και το αντικείμενο μελέτης της παρούσας εργασίας. Χρησιμοποιείται και εδώ η ίδια τεχνική συν-προσομοίωσης που ακολουθήθηκε στα προηγούμενα παραδείγματα.

Στη συγκεκριμένη περίπτωση όμως, δεν υπάρχει κάποια αναλυτική λύση του πλήρους προβλήματος με την οποία θα συγκριθεί το αποτέλεσμα της συν-προσομοίωσης, δηλαδή η τροχιά της σφαίρας και η απόκριση της επιφάνειας. Για το σκοπό αυτό, επιλέχθηκε να γίνει σύγκριση έχοντας ως αναφορά την επίλυση του προβλήματος εξ ολοκλήρου ως πρόβλημα δυναμικής πολλαπλών σωμάτων MBD.

5.3.1 Μοντέλο MBD

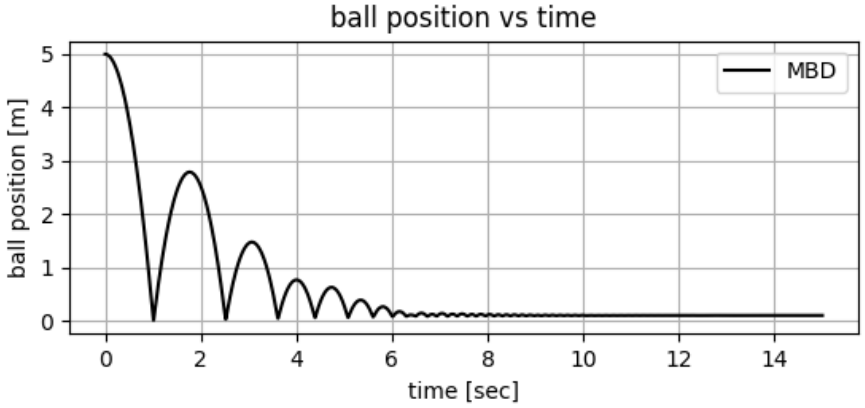
Το μοντέλο MBD δημιουργείται στο MotionView χρησιμοποιώντας τις παραμέτρους του προβλήματος που αναγράφονται στον πίνακα 3. Σημειώνεται ότι ο τύπος επαφής (*Impact*) είναι ο ίδιος που χρησιμοποιήθηκε στη δημιουργία του bouncing ball FMU, όπως και όλες οι υπόλοιπες χαρακτηριστικές διαστάσεις του προβλήματος, έτσι ώστε να υπάρξει συνέπεια μεταξύ των επιλύσεων.



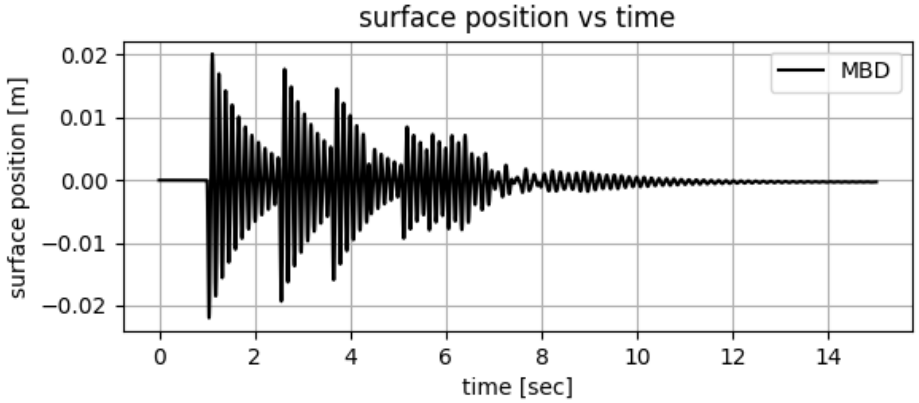
Σχήμα 15: Μοντέλο MBD πλήρους προβλήματος στο MotionView.

Στο MotionView το μοντέλο επιλύεται με τον κώδικα MotionSolve του οποίου η λειτουργία περιγράφηκε στην ενότητα 3.3. Ο επιλυτής *DSTIFF* δεν ολοκληρώνει με σταθερό χρονικό βήμα, αλλά προσαρμόζοντάς το μέσα

σε κάποιο εύρος που ορίζεται από το χρήστη. Έτσι, προκειμένου να υπάρχει άμεση συσχέτιση με τη συν-προσομοίωση που εκτελείται με χρονικό βήμα επικοινωνίας $\mathbf{H} = \mathbf{10^{-4}}$ [sec], τίθεται ένα άνω όριο του χρονικού βήματος της MBD επίλυσης ίσο με $\mathbf{h_{max}} = \mathbf{10^{-4}}$ [sec]. Στο σχήμα 16 και στο σχήμα 17 φαίνονται η τροχιά της σφαίρας και η απόκριση της επιφάνειας αντίστοιχα που προκύπτουν και θα αποτελέσουν τις καμπύλες αναφοράς για την εύρεση του σφάλματος της συν-προσομοίωσης.



Σχήμα 16: Τροχιά σφαίρας από αυτοτελή προσομοίωση MBD.



Σχήμα 17: Απόκριση επιφάνειας από αυτοτελή προσομοίωση MBD.

5.3.2 Συν-προσομοίωση

Ακολουθώντας τη λογική των προηγούμενων παραδειγμάτων, για την εκτέλεση της συν-προσομοίωσης εισάγονται αρχικά τα δύο FMU με την εντολή `load_fmu`, ορίζονται οι συνδέσεις των μεταβλητών εισόδου και εξόδου, καθώς και το συζευγμένο πλέον σύστημα ως εξής:

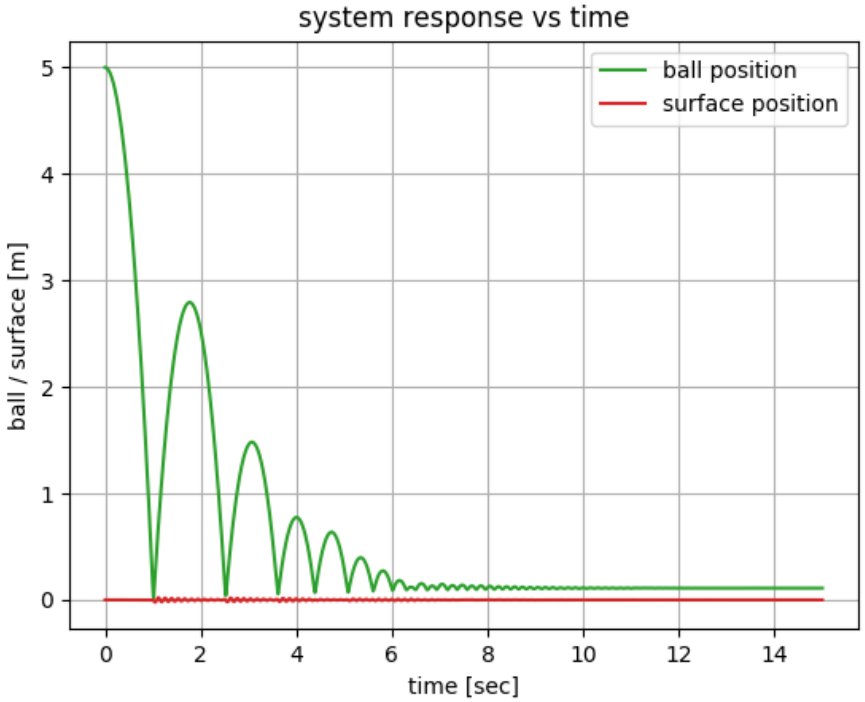
```
m1 = load_fmu('Motionsolve.fmu')
m2 = load_fmu('Force.fmu')

models = [m1, m2]
connections = [
    (m2, "co.value", m1, "inp"),
    (m1, "out", m2, "ci.value"),
    (m1, "sv_output2", m2, "inp2")
]
coupled = Master(models, connections)
```

Έπειτα ορίζονται οι επιλογές του αλγορίθμου master και εκτελείται η συν-προσομοίωση με την εντολή `simulate()`.

```
opts = coupled.simulate_options()
opts['step_size'] = 0.0001
opts['execution'] = 'serial'
opts['error_controlled'] = False
opts['linear_correction'] = False
opts['extrapolation_order'] = 0
opts['result_handling'] = 'file'
result = coupled.simulate(final_time=15., options=opts)
```

Το αποτέλεσμα της συν-προσομοίωσης σε αυτή την περίπτωση περιλαμβάνει τις καμπύλες και των δύο FMU, οι οποίες παρουσιάζονται σε κοινό διάγραμμα στο σχήμα 18, όπου φαίνεται συνολικά η συμπεριφορά του δυναμικού συστήματος.



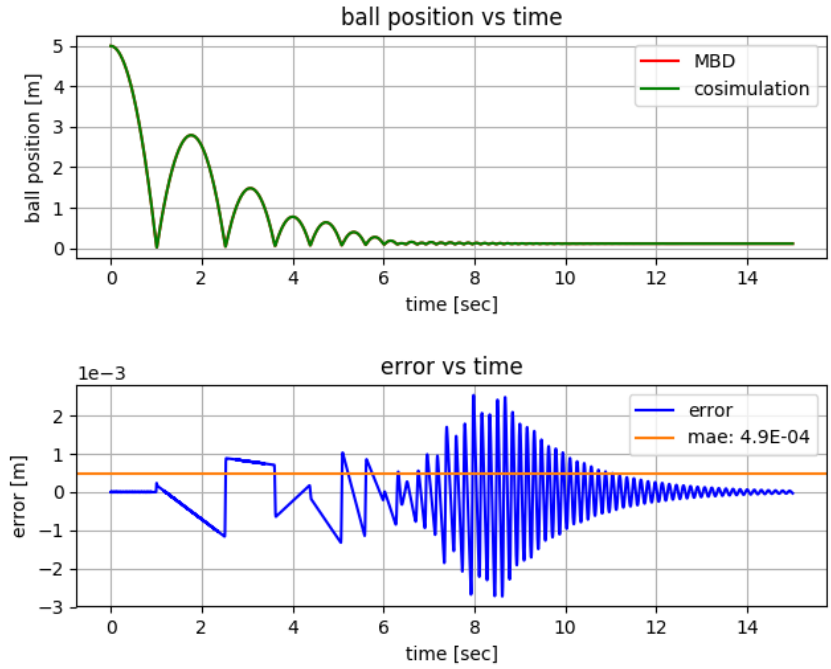
Σχήμα 18: Απόκριση δυναμικού συστήματος με συν-προσομοίωση.

Στο σχήμα 19 παρουσιάζεται η σύγκριση της τροχιάς της σφαίρας όπως προέκυψε από τη συν-προσομοίωση σε σχέση με αυτήν από την αυτοτελή προσομοίωση με MBD. Για τη σύγκριση γίνεται και πάλι χρήση της σχέσης (19), ενώ προτείνεται και ένας τρόπος αξιολόγησης των σφαλμάτων, μέσω του υπολογισμού *Mean Absolute Error*, δηλαδή του μέσου απόλυτου σφάλματος των τιμών της χρονοϊστορίας. Το *MAE* αποτελεί το μέσο όρο των σφαλμάτων κάθε χρονικής στιγμής, και ορίζεται ως εξής:

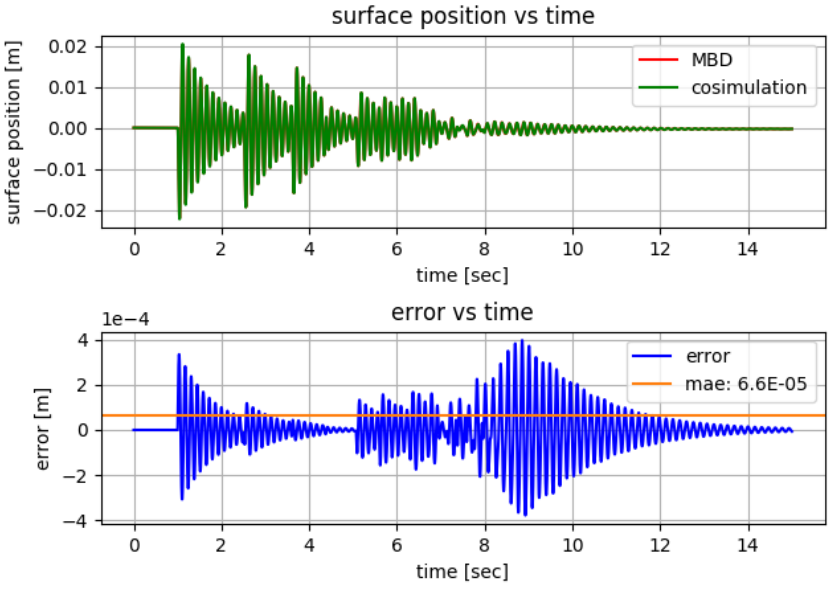
$$MAE = \frac{\sum_{i=1}^{n} |y_{i,MBD} - y_{i,cosim}|}{n} \ . \tag{22}$$

Για την καμπύλη τροχιάς της σφαίρας υπολογίστηκε ίσο με $\mathbf{MAE} = \mathbf{4.9e-4}$ $[m]$, με $n = 150000$ χρονικές στιγμές.

Αντίστοιχα, στο σχήμα 20 παρουσιάζεται η σύγκριση της απόκρισης της επιφάνειας ανάμεσα στη συν-προσομοίωση και την αυτοτελή προσομοίωση με MBD. Για τη συγκεκριμένη καμπύλη υπολογίστηκε μέσο απόλυτο σφάλμα ίσο με $\mathbf{MAE} = \mathbf{6.6e-5}$ $[m]$.



Σχήμα 19: Σφάλμα συν-προσομοίωσης - τροχιά σφαίρας.



Σχήμα 20: Σφάλμα συν-προσομοίωσης - απόκριση επιφάνειας.

6 Συμπεράσματα - Προτάσεις - Περιορισμοί

Η παρούσα διπλωματική εργασία είχε πολλαπλούς στόχους αναφορικά με τη σύγχρονη τεχνική της συν-προσομοίωσης. Αρχικά καταγράφηκαν τα εργαλεία που έχει στη διάθεσή του ο μηχανικός για να επιλύει σύνθετα προβλήματα με συν-προσομοίωση. Τέθηκαν οι βάσεις για την κατανόηση της τεχνικής και τη λογική με την οποία ένα πρόβλημα μπορεί να χωριστεί σε υπο-μοντέλα. Με το αριθμητικό παράδειγμα του δυναμικού συστήματος της σφαίρας που αναπηδά σε μη σταθερή επιφάνεια περιγράφηκε ένας τρόπος δημιουργίας των FMU, χωρίς απαραίτητα τη χρήση εμπορικού λογισμικού. Με την επίλυση του παραδείγματος αναδεικνύονται οι δυνατότητες της συν-προσομοίωσης, τα πλεονεκτήματα και οι αδυναμίες της.

Βασικός στόχος της εργασίας ήταν να αναδειχθεί ένας συστηματικός τρόπος επίλυσης προβλημάτων με συν-προσομοίωση, φέροντας σε επικοινωνία διαφορετικούς επιλυτές και λογισμικά. Όπως έγινε εμφανές στις προηγούμενες ενότητες, με τη χρήση του προτύπου FMI είναι δυνατή η δημιουργία εξειδικευμένων FMU που περιγράφουν υποσυστήματα ενός μοντέλου, τα οποία είναι ταυτόχρονα ικανά να χρησιμοποιηθούν από εμπορικά προγράμματα και να τεθούν σε συν-προσομοίωση. Για να συμβεί αυτό, καταστρώθηκαν οι εξισώσεις του μαθηματικού προσομοιώματος και υλοποιήθηκαν σε κώδικα γλώσσας *C++*, δημιουργώντας παράλληλα τις απαραίτητες μεταβλητές επικοινωνίας του FMU, με τρόπο που ορίζεται από το πρότυπο FMI. Η ίδια η συν-προσομοίωση εκτελέστηκε με τη χρήση του αλγορίθμου master παράλληλης επικοινωνίας της βιβλιοθήκης PyFMI, σε γλώσσα προγραμματισμού *Python*.

Από τα αποτελέσματα που παρουσιάστηκαν είναι εμφανές πως η συν-προσομοίωση βρίσκεται σε ικανοποιητικό στάδιο αναφορικά με τις κλασικές μεθόδους αυτοτελούς προσομοίωσης, όμως δεν έχει φτάσει ακόμα σε σημείο που να μπορέσει να τις αντικαταστήσει. Η μέθοδος ακόμα βρίσκεται υπό ανάπτυξη και η υπάρχουσα βιβλιογραφία δεν είναι ιδιαίτερα εκτενής την παρούσα χρονική στιγμή. Κατά τη διάρκεια εκπόνησης της εργασίας παρουσιάστηκαν δυσκολίες στη δημιουργία του bouncing ball FMU, αναφορικά με την ικανότητά του να ανταλλάσσει δεδομένα κατά τη διάρκεια εκτέλεσης της συν-προσομοίωσης. Στόχος ήταν να έχει τη δυνατότητα επικοινωνίας με διαφορετικά FMU, με εμπορικά λογισμικά συν-προσομοίωσης (λ.χ. Dymola, OpenModelica, JModelica) και με προγραμματιστικές βιβλιοθήκες (λ.χ. PyFMI). Για το σκοπό αυτό, το FMU δημιουργήθηκε ακολουθώντας το πρότυπο FMI, του οποίου το εγχειρίδιο δε στάθηκε επαρκώς ευνόητο, με αποτέλεσμα την διενέργεια χρονοβόρων δοκιμών.

Όπως συζητήθηκε και στην ενότητα 4.2, ο αλγόριθμος master που επιλέγεται είναι υπεύθυνος για την απρόσκοπτη επικοινωνία μεταξύ των FMU και την ορθή εκτέλεση της συν-προσομοίωσης. Ακόμη και ο αλγόριθμος της βιβλιοθήκης PyFMI που χρησιμοποιήθηκε στην επίλυση του παραδείγματος διαθέτει πληθώρα επιλογών. Δεδομένου ότι δεν έγινε εκτενής μελέτη των επιλογών που προσφέρονται στα πλαίσια αυτής της εργασίας, επαφίεται σαν πρόταση για περαιτέρω έρευνα, η εμβάθυνση στη βελτιστοποίηση των αλγορίθμων συν-προσομοίωσης. Μπορούμε να υποθέσουμε ότι η μέγιστη ακρίβεια είναι δυνατό να επιτευχθεί με την ανάπτυξη εξατομικευμένων master αλγορίθμων για το εκάστοτε πρόβλημα.

Επιπρόσθετα, ένα ακόμα πεδίο της συν-προσομοίωσης που χρήζει μελέτης είναι η ξεχωριστή ανάλυση ευστάθειας των υπο-μοντέλων και η μεθοδευμένη επικύρωση των αποτελεσμάτων (validation & benchmarking). Εφόσον το κάθε υποσύστημα επιλύεται αυτόνομα, με το δικό του εσωτερικό χρονικό βήμα, ανάμεσα στα σημεία επικοινωνίας της συν-προσομοίωσης, είναι αναμενόμενο ότι το ιδανικό εσωτερικό βήμα μπορεί να είναι διαφορετικό για το κάθε FMU προκειμένου να υπάρχει σύγκλιση και μέγιστη δυνατή ακρίβεια. Η παραπάνω ανάλυση θα επηρέαζε άμεσα και τον πραγματικό χρόνο εκτέλεσης, οπότε η συγκριτική αξιολόγηση των επιδόσεων της συν-προσομοίωσης στην επίλυση γνωστών αριθμητικών προβλημάτων μηχανικής θα ήταν απαραίτητη.

7 Βιβλιογραφία

[1] Δημήτριος Λιάλιος. Συν-προσομοίωση ιμάντων μεταφοράς και κοκκώδους υλικού. 2019.

[2] Cláudio Gomes, Casper Thule, David Broman, Peter Gorm Larsen, and Hans Vangheluwe. Co-simulation: State of the art. 2017.

[3] Fabio Cremona, Marten Lohstroh, David Broman, Edward A Lee, Michael Masin, Stavros Tripakis, and Edward A Lee. Hybrid co-simulation: it' s about time. *Software & Systems Modeling*, 2017.

[4] Felix Gaisbauer, Eva Lampen, Philipp Agethen, and Enrico Rukzio. Combining heterogeneous digital human simulations: presenting a novel co-simulation approach for incorporating different character animation technologies. *Visual Computer*, 2020.

[5] FMI Standard. FMI for ModelExchange and Co-Simulation v2.0. 2014.

[6] Jens Bastian, Christoph Clauß, Susann Wolf, and Peter Schneider. Master for co-simulation using fmi. *Proceedings from the 8th International Modelica Conference, Technical Univeristy, Dresden, Germany*, 63:115–120, 2011.

[7] Peter Fritzson and Vadim Engelson. Modelica A Unified Object-Oriented Language for Physical Systems Modeling Language Specification 3.2 Revision 2. 2017.

[8] Peter Fritzson. *Principles of Object-Oriented Modeling and Simulation with Modelica 2.1*. John Wiley & Sons, 2010.

[9] Daniel Kraus Cabo, Dr-Ing Georg Frey, and Fethi Belkhir. Integration of Modelica-based Models with the JModelica Platform. 2014.

[10] Modelica Association. Modelica Tools. `https://www.modelica.org/tools`.

[11] Modelon AB. JModelica.org User Guide. 2018.

[12] Johan Åkesson. Languages and Tools for Optimization of Large–Scale Systems. *Department of Automatic Control, Lund Institute of Technology, Lund University*, 2007.

[13] JModelica.org. JModelica.org. `https://jmodelica.org/`.

[14] OpenModelica. Welcome to OpenModelica. https://openmodelica.org/.

[15] Hilding Elmqvist. Modelica Evolution - From My Perspective. *Proceedings of the 10th International Modelica Conference, March 10-12, 2014, Lund, Sweden*, 96:17–26, 2014.

[16] 3DS Dassault Systemes. Dymola - Dassault Systèmes®. https://www.3ds.com/products-services/catia/products/dymola/.

[17] Σωτήριος Νατσιάβας. *Εφαρμοσμένη Δυναμική*. Ζήτη, Θεσσαλονίκη, 2η έκδ. edition, 1996.

[18] Altair. MotionSolve Reference Guide. 2019.

[19] Σωτήριος Νατσιάβας. *Ταλαντώσεις δυναμικών συστημάτων με μη γραμμικά χρακτηριστικά*. Ζήτη, Θεσσαλονίκη, 2000.

[20] 3DS Dassault Systemes. FMU Simulator. https://www.3ds.com/products-services/catia/products/dymola/free-downloads/fmu-simulator/.

[21] Christian Andersson. Methods and Tools for Co-Simulation of Dynamic Systems with the Functional Mock-up Interface. *Doctoral Theses in Mathematical Sciences, Lund Institute of Technology, Lund University*, 2016.

[22] Tom Schierz, Martin Arnold, and Christoph Clauss. Co-simulation with communication step size control in an FMI compatible master algorithm. *Proceedings of the 9th International Modelica Conference, September 3-5, 2012, Munich, Germany*, 76(2):205–214, 2012.

Α΄ ΠΑΡΑΡΤΗΜΑ

Καταγραφή 3: Υλοποίηση κώδικα της συνάρτησης *fmi2DoStep* του προτύπου FMI σε γλώσσα *C++*, συνάρτηση εκτέλεσης βήματος ανάμεσα σε σημεία επικοινωνίας της συν-προσομοίωσης.

```
fmi2Status fmi2DoStep(fmi2Component c,
                      fmi2Real currentCommunicationPoint,
                      fmi2Real communicationStepSize,
                      fmi2Boolean noSetFMUStatePriorToCurrentPoint)
{
    // ModelInstance reference at current time point
    ModelInstance *comp = (ModelInstance *)c;
    // Current time point of the co-simulation
    comp->time = currentCommunicationPoint;

    // Divide communication timestep into (n) segments
    int n = 100;
    double dt = communicationStepSize / n;

    // Inputs from MS FMU - values taken at time of communication
    double x0 = r(inp_);  // ground position
    double v0 = r(inp2_); // ground velocity

    double xn = r(h_); // ball position
    double vn = r(v_); // ball velocity

    double g = r(g_);
    double k = 1e5;
    double m = 0.62;
    double r = 0.12;

    // Start the internal loop for the Bouncing Ball FMU integrator
    // The internal timestep (dt) is used for additional (n) time
    // points between communication points of the co-simulation
    for (unsigned i = 0; i < n; i++)
    {
        double di = r - (xn - x0); // penetration depth
        double dzdt = vn - v0;     // penetration rate (velocity)
        // Contact detection & resolution
        if (di > 0.)
        {
            double f_spring = k * pow(di, 2.1);
            double f_damp = -step(xn, x0+r-0.08, 10, x0+r, 0)
                            * dzdt;
            double Fn = f_spring + f_damp;
        }
```

```
42          else
43          {
44              double Fn = 0.;
45          }
46
47          //** FORWARD EULER METHOD **//
48          xn = xn + dt * vn; // vn of previous time point inside loop
49          vn = vn + dt * (-g + Fn / m);
50      }
51
52      // Update ball velocity and position before the next
53      // communication time point
54      r(v_) = vn;
55      r(h_) = xn;
56
57      // Advance time to next communication point
58      comp->time += communicationStepSize;
59
60      // Output to MS FMU - value given at time of the next
61      // communication time point
62      r(out_) = Fn; // contact force
63
64      return fmi2OK;
65  }
```

Καταγραφή 4: Αρχείο περιγραφής μοντέλου *ModelDescription.xml* που απαιτείται από το πρότυπο FMI για τη δημιουργία του FMU.

```
1  <?xml version="1.0" encoding="ISO-8859-1"?>
2  <fmiModelDescription
3    fmiVersion="2.0"
4    modelName="bouncingBall"
5    guid="{8c4e810f-3df3-4a00-8276-176fa3c9f003}"
6    numberOfEventIndicators="0">
7
8  <CoSimulation
9    modelIdentifier="bouncingBall"
10   canHandleVariableCommunicationStepSize="true">
11   <SourceFiles>
12     <File name="bouncingBall.cc"/>
13   </SourceFiles>
14 </CoSimulation>
15
16 <LogCategories>
17   <Category name="logAll"/>
18   <Category name="logError"/>
19   <Category name="logFmiCall"/>
```

```
20    <Category name="logEvent"/>
21  </LogCategories>
22
23  <ModelVariables>
24    <ScalarVariable name="h" valueReference="0"
25                    description="height, used as state"
26                    causality="local" variability="continuous"
27                    initial="exact">
28      <Real start="1" reinit="true"/>
29    </ScalarVariable>
30    <ScalarVariable name="der(h)" valueReference="1"
31                    description="velocity of ball"
32                    causality="local" variability="continuous"
33                    initial="calculated">
34      <Real derivative="1"/>
35    </ScalarVariable>
36    <ScalarVariable name="v" valueReference="2"
37                    description="velocity of ball, used as state"
38                    causality="local" variability="continuous"
39                    initial="exact">
40      <Real start="0" reinit="true"/>
41    </ScalarVariable>
42    <ScalarVariable name="der(v)" valueReference="3"
43                    description="acceleration of ball"
44                    causality="local" variability="continuous"
45                    initial="calculated">
46      <Real derivative="3"/>
47    </ScalarVariable>
48    <ScalarVariable name="g" valueReference="4"
49                    description="acceleration of gravity"
50                    causality="parameter" variability="fixed"
51                    initial="exact">
52      <Real start="9.81"/>
53    </ScalarVariable>
54    <ScalarVariable name="e" valueReference="5"
55                    description="dimensionless parameter"
56                    causality="parameter" variability="tunable"
57                    initial="exact">
58      <Real start="0.7" min="0.5" max="1"/>
59    </ScalarVariable>
60    <ScalarVariable name="out" valueReference="6"
61                    description="output"
62                    causality="output">
63      <Real start="0."/>
64    </ScalarVariable>
65    <ScalarVariable name="inp" valueReference="7"
66                    description="ground position"
67                    causality="input">
68      <Real/>
```

```
  </ScalarVariable>
  <ScalarVariable name="inp2" valueReference="8"
                  description="ground velocity"
                causality="input">
    <Real/>
  </ScalarVariable>
</ModelVariables>

<ModelStructure>
  <Outputs>
    <Unknown index="7" />
  </Outputs>
  <Derivatives>
    <Unknown index="2" />
    <Unknown index="4" />
  </Derivatives>
  <InitialUnknowns>
    <Unknown index="2"/>
    <Unknown index="4"/>
  </InitialUnknowns>
</ModelStructure>

</fmiModelDescription>
```

Καταγραφή 5: Κώδικας συν-προσομοίωσης σε γλώσσα *Python* για την επαλήθευση του bouncing ball FMU.

```
from pymodelica import compile_fmu
from pyfmi import load_fmu, Master

compile_fmu('Surface', 'Surface.mo',
            compile_to="Surface.fmu", target='cs')

m1 = load_fmu('bouncingball.fmu')
m2 = load_fmu('Surface.fmu')

models = [m1, m2]
connections = [
    (m2, "co.value", m1, "inp"),
    (m2, "co2.value", m1, "inp2"),
    (m1, "out", m2, "ci.value")
]

coupled = Master(models, connections)

opts = coupled.simulate_options()
opts['step_size'] = 0.00001
```

```
21 opts['execution'] = 'serial'
22 opts['error_controlled'] = False
23 opts['linear_correction'] = False
24 opts['extrapolation_order'] = 0
25 opts['result_handling'] = 'file'
26
27 result = coupled.simulate(final_time=15.,options=opts)
```