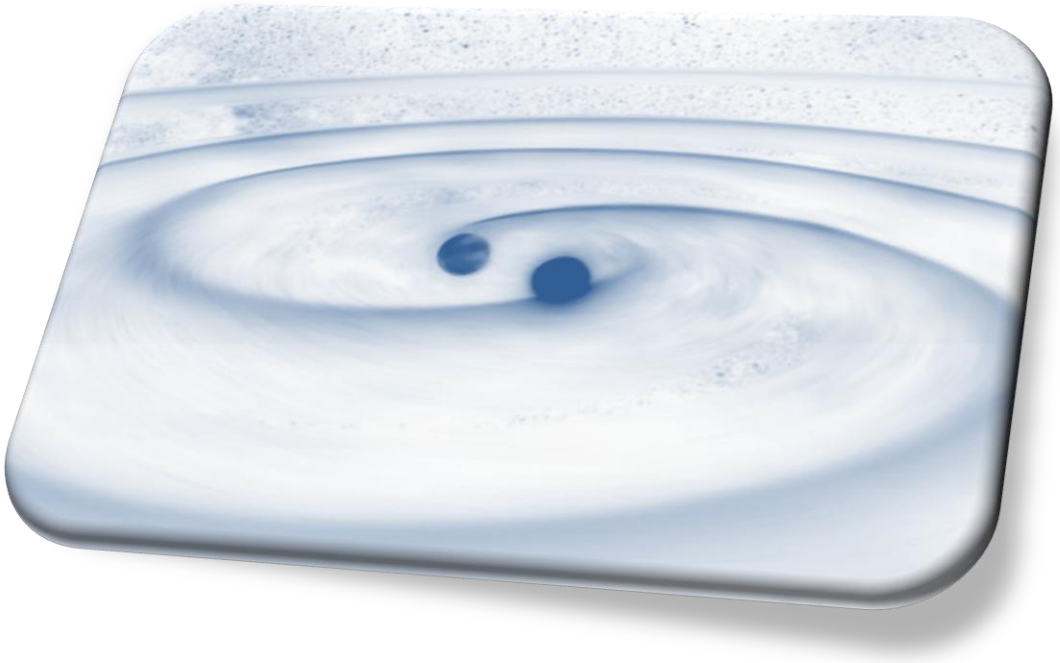
Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης
Τμήμα Φυσικής – Τομέας Αστροφυσικής, Αστρονομίας & Μηχανικής
Πρόγραμμα Μεταπτυχιακών Σπουδών Υπολογιστικής Φυσικής



Διπλωματική Εργασία

Προσομοιώσεις Διπλών Συστημάτων Συμπαγών Αστέρων στην Ελλειψοειδή Προσέγγιση

Βουκάντζης Δημήτρης

Μάρτιος 2007

Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης
Τμήμα Φυσικής – Τομέας Αστροφυσικής, Αστρονομίας & Μηχανικής
Πρόγραμμα Μεταπτυχιακών Σπουδών Υπολογιστικής Φυσικής

Διπλωματική Εργασία στα πλαίσια του
Π.Μ.Σ. Υπολογιστικής Φυσικής

Προσομοιώσεις Διπλών Συστημάτων Συμπαγών Αστέρων στην Ελλειψοειδή Προσέγγιση

Βουκάντζης Δημήτρης
Επιβλέπων: Στεργιούλας Νικόλαος

Μάρτιος 2007

Περίληψη

Η κατάρρευση διπλών συστημάτων αποτελούμενα από αστέρες νετρονίων και μελανές οπές αναμένεται να αποτελέσει μια από τις σημαντικότερες πηγές βαρυτικής ακτινοβολίας, ανιχνεύσιμη από τα συμβολόμετρα LIGO, VIRGO, TAMA και GEO600. Αντικείμενο αυτής της διπλωματικής εργασίας αποτελεί η μελέτη της εξέλιξης τέτοιων συστημάτων καθώς και η εκπομπή βαρυτικής ακτινοβολίας από αυτά. Λαμβάνοντας υπόψη την πρώτη μετανευτώνεια διόρθωση της τροχιακής κίνησης και αντιμετωπίζοντας τους αστέρες νετρονίων ως Riemann-S ελλειψοειδή επιλύουμε αριθμητικά τις διαφορικές εξισώσεις κίνησης για διάφορες τιμές των φυσικών παραμέτρων του συστήματος (πολυτροτικός δείκτης, λόγος μαζών και ακτίνα του αστέρα). Τέλος, υπολογίζουμε τις κυματομορφές της βαρυτικής ακτινοβολίας των συστημάτων αυτών και προχωρούμε σε ανάλυση των συχνοτήτων από τις οποίες συνίστανται.

Abstract

Black hole – Neutron star binary systems are one of the most promising sources of gravitational radiation for detection by laser interferometric detectors, such as LIGO, VIRGO, TAMA and GEO600. In this thesis we study the evolution and emission of gravitational radiation by these binaries. Using the first post-Newtonian correction of the orbital motion and modeling the neutron stars as compressible ellipsoids, obeying a polytropic equation of state, we study the evolution of these binaries for different values of several physical parameters (polytropic index, mass ratio and neutron star radius). Furthermore, we calculate the gravitational waveforms and study the modes of which they consist.

Ευχαριστίες

Η πτυχιακή αυτή εργασία εκπονήθηκε κατά το χρονικό διάστημα Δεκέμβριος 2005 - Μάρτιος 2007. Όντας πλέον ολοκληρωμένη θα ήθελα να ευχαριστήσω θερμά τον επιβλέπων καθηγητή κ. Στεργιούλα Νικόλαο για την εξαιρετική συνεργασία. Επίσης, οφείλω να ευχαριστήσω τη Dörte Hansen και τον Prof. Gerhard Schäfer από το πανεπιστήμιο της Jena (Γερμανία) τόσο για τη συνεργασία τους, όσο και για τη φιλοξενία τους κατά το διάστημα Μάιος – Ιούνιος 2006.

Περιεχόμενα

ΠΙΝΑΚΑΣ ΣΧΗΜΑΤΩΝ

ΠΙΝΑΚΑΣ ΠΛΑΙΣΙΩΝ

ΕΙΣΑΓΩΓΗ

Μια από τις σπουδαιότερες προβλέψεις της Γενικής Θεωρίας της Σχετικότητας είναι η εκπομπή βαρυτικής ακτινοβολίας από επιταχυνόμενες μάζες. Παρόλο που μέχρι σήμερα δεν έχει γίνει άμεση παρατήρηση της ακτινοβολίας αυτής, υπάρχουν ισχυρές ενδείξεις για την ύπαρξή της από έμμεσες παρατηρήσεις, όπως για παράδειγμα αυτή του πάλσαρ PSR1913+16 (με ακρίβεια καλύτερη από 1% στην εξέλιξη της φάσης). Τα αμέσως επόμενα χρόνια αναμένεται να γίνουν και οι πρώτες άμεσες παρατηρήσεις βαρυτικών κυμάτων, τόσο από τους επίγειους ανιχνευτές LIGO, VIRGO, TAMA, GEO, όσο και από τον διαστημικό ανιχνευτή LISA, που αναμένεται να υλοποιηθεί την επόμενη δεκαετία.

Η άμεση παρατήρηση της βαρυτικής ακτινοβολίας αναμένεται να ανοίξει ένα από τα τελευταία «παράθυρα» παρατήρησης του Σύμπαντος, που παραμένουν ακόμη κλειστά. Το γεγονός αυτό, σε συνδυασμό με τη φύση και τις ιδιότητες της βαρυτικής ακτινοβολίας, είναι πολλά υποσχόμενο. Πιο συγκεκριμένα:

- Η εξαιρετικά ασθενής αλληλεπίδραση των βαρυτικών κυμάτων με την ύλη έχει ως αποτέλεσμα την μετάδοση του κύματος από τη πηγή στον παρατηρητή χωρίς σχεδόν καμιά παραμόρφωση.
- Τα βαρυτικά κύματα δημιουργούνται από τη κίνηση της πηγής ως σύνολο και όχι από επιμέρους κινήσεις των ατόμων όπως τα ηλεκτρομαγνητικά κύματα. Ως εκ τούτου μεταφέρουν ένα τελείως διαφορετικό είδος πληροφορίας σε σχέση μ' αυτή που παρέχουν τα ηλεκτρομαγνητικά κύματα.
- Η βαρυτική ακτινοβολία θα αποτελέσει τον μόνο άμεσο τρόπο παρατήρησης κάποιων αστρικών αντικειμένων, όπως π.χ. οι μελανές οπές.
- Τέλος, η βαρυτική ακτινοβολία είναι σε θέση να δώσει πληροφορίες για τη κατάσταση του Σύμπαντος, μόλις 10^{-24} δευτερόλεπτα μετά τη δημιουργία του.

Τα παραπάνω στοιχεία καθιστούν τη βαρυτική ακτινοβολία μια εξαιρετική πηγή για την άντληση νέων πληροφοριών γύρω από γνωστά προβλήματα της αστροφυσικής, ενώ ταυτόχρονα θα οδηγήσει σε νέες ανακαλύψεις.

Η ανίχνευση των βαρυτικών κυμάτων, από τους σημερινούς ανιχνευτές, προϋποθέτει τη γνώση της αναμενόμενης κυματομορφής με εξαιρετική ακρίβεια. Οι κυματομορφές αυτές μπορούν να υπολογιστούν με την αριθμητική επίλυση των εξισώσεων της γενικής θεωρίας της σχετικότητας, κάτι που είναι εξαιρετικά δύσκολο και δαπανηρό. Η απάντηση στο πρόβλημα αυτό έρχεται συνήθως μέσα από απλούστερα μοντέλα που εμπεριέχουν κάποιες βασικές ιδιότητες των

εξισώσεων της σχετικότητας (π.χ. μετανευτώνειες προσεγγίσεις). Αυτή θα είναι η μέθοδος που θα ακολουθήσουμε στην εργασία αυτή.

Πιο συγκεκριμένα, θα ασχοληθούμε με διπλά συστήματα αποτελούμενα από μελανές οπές ή/και αστέρες νετρονίων. Οι εξισώσεις περιέχουν την πρώτη μετανευτώνεια διόρθωση στη τροχιακή κίνηση, ενώ οι αστέρες νετρονίων θα αντιμετωπιστούν ως Riemann-S ελλειψοειδή. Τα συστήματα αυτά έχουν ενδιαφέρον, γιατί αναμένεται να αποτελέσουν μια από τις πρώτες πηγές βαρυτικής ακτινοβολίας που πρόκειται να ανιχνευτούν.

Το κείμενο που ακολουθεί αποτελείται από τέσσερα κεφάλαια. Στο πρώτο παρουσιάζουμε κάποια στοιχεία για τη βαρυτική ακτινοβολία, την ανίχνευσή της και τις πηγές από τις οποίες αναμένεται να προέλθει. Στο δεύτερο κεφάλαιο παρουσιάζουμε τις λεπτομέρειες του μοντέλου που θα χρησιμοποιήσουμε (Riemann-S ελλειψοειδή, μετανευτώνειες προσεγγίσεις, κυματομορφές βαρυτικής ακτινοβολίας). Στο τρίτο κεφάλαιο περιγράφουμε τις αριθμητικές τεχνικές που χρησιμοποιήθηκαν, ενώ στο τέταρτο κεφάλαιο παρουσιάζουμε τα αποτελέσματα. Τέλος, στο παράρτημα παραθέτουμε τον κώδικα του προγράμματος.

ΚΕΦΑΛΑΙΟ 1^{Ο} "ΜΕΡΙΚΑ ΣΤΟΙΧΕΙΑ ΓΙΑ ΤΗ ΒΑΡΥΤΙΚΗ ΑΚΤΙΝΟΒΟΛΙΑ"

Στο κεφάλαιο αυτό θα περιγράψουμε εν συντομία μερικά στοιχεία που αφορούν τη φύση και την ανίχνευση των βαρυτικών κυμάτων. Συγκεκριμένα, στη πρώτη ενότητα θα δούμε τις προβλέψεις και προσεγγίσεις την Γενικής Θεωρίας της Σχετικότητας για τη βαρυτική ακτινοβολία. Στη δεύτερη ενότητα θα δούμε τις σημαντικότερες πηγές από τις οποίες αναμένεται να έχουμε μετρήσιμη βαρυτική ακτινοβολία. Τέλος, στην τρίτη ενότητα θα δούμε τις δυσκολίες που ενέχει η παρατήρηση βαρυτικών κυμάτων και θα παρουσιάσουμε μερικά στοιχεία για τους ανιχνευτές που έχουν κατασκευαστεί ή αναμένεται να κατασκευαστούν. Εκτός και αν αναφερθεί διαφορετικά, σ' όλο το κεφάλαιο θα θεωρήσουμε $G = c = 1$.

1.1 Η ΒΑΡΥΤΙΚΗ ΑΚΤΙΝΟΒΟΛΙΑ ΣΤΗ ΣΧΕΤΙΚΟΤΗΤΑ

Στην ενότητα αυτή παρουσιάζουμε τις προβλέψεις της Γενικής Θεωρίας της Σχετικότητας για τη βαρυτική ακτινοβολία. Για περισσότερες λεπτομέρειες δείτε τις εργασίες ανασκόπησης [Sathyaprakash, 2005] και [Andersson & Kokkotas, 2003], καθώς και το βιβλίο «Εισαγωγή στη Γενική Θεωρία της Σχετικότητας» (Σπύρου Ν.Κ., Εκδόσεις Γαρταγάνη, 1985).

Στη νευτώνεια θεωρία της βαρύτητας το δυναμικό ικανοποιεί την εξίσωση Poisson

$$\nabla^2\varphi(t,\boldsymbol{x}) = 4\pi\rho(t,\boldsymbol{x}) \tag{1.1}$$

Στην εξίσωση αυτή δεν υπάρχει ρητή έκφραση του χρόνου τόσο στο αριστερό, όσο και στο δεξί μέλος, μ' αποτέλεσμα να μην υπάρχουν φαινόμενα υστέρησης, δηλαδή οποιαδήποτε αλλαγή στην κατανομή της πυκνότητας επιφέρει μια ακαριαία αλλαγή στο δυναμικό. Αντιθέτως, στη θεωρία της σχετικότητας οι συνιστώσες της μετρικής αποτελούν τα βαρυτικά δυναμικά και ικανοποιούν μια κυματική εξίσωση. Συνεπώς, θα υπάρχουν φαινόμενα υστέρησης. Τα παραπάνω γίνονται εμφανή στις γραμμικοποιημένες εξισώσεις Einstein.

Αν υποθέσουμε ότι το βαρυτικό πεδίο είναι ασθενές, τότε η μετρική του χωρόχρονου, $g_{\alpha\beta}$, μπορεί να θεωρηθεί πως θα διαφέρει λίγο από τη μετρική Minkowski, $\eta_{\alpha\beta} = Diag(-1,1,1,1)$. Έτσι, μπορούμε να γράψουμε τον μετρικό τανυστή ως

$$g_{\alpha\beta} = \eta_{\alpha\beta} + h_{\alpha\beta}\,, \quad \left|h_{\alpha\beta}\right| \ll 1 \tag{1.2}$$

Ο τανυστής $h_{\alpha\beta}$ λέγεται διαταραχή της μετρικής και εκφράζει την απόκλιση της μετρικής από τον επίπεδο χωρόχρονο. Εφόσον έχουμε υποθέσει πως το βαρυτικό πεδίο είναι ασθενές, οι εξισώσεις

πεδίου για τη διαταραχή της μετρικής μπορούν να γραμμικοποιηθούν, καταλήγοντας στις παρακάτω κυματικές εξισώσεις για την διαταραχή $h_{\alpha\beta}$

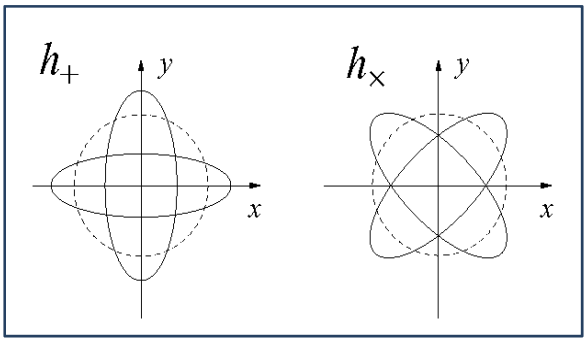
$$\bar{h}_{\alpha\beta} = 16\pi T_{\alpha\beta} \tag{1.3}$$

όπου συμβολίζουμε με $2\bar{h}_{\alpha\beta} \equiv 2h_{\alpha\beta} - \eta_{\alpha\beta}h^{\mu}_{\mu}$, με $\Box \equiv \eta^{\alpha\beta}\partial_{\alpha}\partial_{\beta}$ τον τελεστή του d' Alembert και τέλος με $T_{\alpha\beta}$ τον τανυστή ενέργειας-ορμής. Οι εξισώσεις (1.3) έχουν λύσεις της μορφής

$$\bar{h}_{\alpha\beta}(t,\boldsymbol{x}) = 4\int \frac{T_{\alpha\beta}(t-|\boldsymbol{x}-\boldsymbol{x}'|,\boldsymbol{x}')\,d^3x'}{|\boldsymbol{x}-\boldsymbol{x}'|} \tag{1.4}$$

Σύμφωνα με την εξίσωση (1.4), η διαταραχή της μετρικής σ' ένα σημείο $\boldsymbol{x}$ κάποια χρονική στιγμή t θα καθορίζεται από την πηγή $T_{\alpha\beta}$ με μία χρονική καθυστέρηση $t-|\boldsymbol{x}-\boldsymbol{x}'|$. Επομένως, διαταραχές της πηγής, $T_{\alpha\beta}$, θα μεταφέρονται με πεπερασμένη ταχύτητα στον χωροχρόνο και όχι ακαριαία όπως προέβλεπε η Νευτώνεια βαρύτητα.

Η διαταραχή της μετρικής, $\bar{h}_{\alpha\beta}$, αποτελεί μια πρώτη προσέγγιση της περιγραφής των βαρυτικών κυμάτων στα πλαίσια της γραμμικοποιημένης θεωρίας της βαρύτητας (quadrupole formula). Για μια πηγή που εκπέμπει βαρυτική ακτινοβολία κατά τη z-διεύθυνση, παρόλο που ο τανυστής $\bar{h}_{\alpha\beta}$ έχει 10 συνιστώσες, στην πραγματικότητα υπάρχουν μόνο δύο βαθμοί ελευθερίας. Οι δύο διαφορετικές πολώσεις του βαρυτικού κύματος αναφέρονται συνήθως ως h_{+} και $h_{\times}$, από τις παραμορφώσεις που επιφέρουν σε κάποιο αντικείμενο κατά τη διέλευση του βαρυτικού κύματος μέσα από αυτό. Στο Σχήμα 1.1 αποσαφηνίζεται η δράση των δύο πολώσεων της βαρυτικής ακτινοβολίας.



Σχήμα 1. 1. Οι δύο πολώσεις ενός βαρυτικού κύματος, h_+ και $h_\times$, και η παραμόρφωση που προκαλούν σ' ένα κυκλικό σώμα για εκπομπή κατά τη z-διεύθυνση (Σχήμα 2, Andersson & Kokkotas, 2003).

Ένα βαρυτικό κύμα που δρα στο επίπεδο x-y μετατρέπει το κυκλικό αντικείμενο σε έλλειψη. Η h_{+} συνιστώσα δρα κατά μήκος των αξόνων x και y, ενώ η $h_{\times}$ συνιστώσα δρα κατά μήκος των διχοτόμων των αξόνων x και y.

Αν στις λύσεις (1.4) επιβάλλουμε τέτοιες συνθήκες, ώστε η λύση στην περιοχή μακριά από την πηγή (far-zone solution) να είναι ένα εξερχόμενο κύμα, τότε καταλήγουμε στις παρακάτω σχέσεις για το πλάτος των βαρυτικών κυμάτων και τη φωτεινότητα της πηγής

$$h_{jk} = \frac{2}{r}\ddot{\mathfrak{T}}_{jk}(t-\boldsymbol{r}) \tag{1.5}$$

Όπου το σύμβολο « ˙ » αντιστοιχεί στη χρονική παράγωγο και με $\mathfrak{I}_{jk}$ συμβολίζεται ο ανηγμένος τανυστής της τετραπολικής ροπής της πηγής, που δίνεται από τη σχέση

$$\mathfrak{I}_{jk} = \int \rho \left(x_j x_k - \frac{1}{3} r^2 \delta_{jk} \right) d^3x \tag{1.6}$$

Η δε φωτεινότητα είναι

$$\mathcal{L} = \frac{1}{5} \langle \dddot{\mathfrak{I}}_{jk} \dddot{\mathfrak{I}}^{jk} \rangle \tag{1.7}$$

Η συχνότητα των βαρυτικών κυμάτων, καθορίζεται είτε από κάποια κυρίαρχη κίνηση που εκτελεί η πηγή (π.χ. η περιστροφή για έναν πάλσαρ), ή συνδέεται με τις φυσικές συχνότητες ιδιοταλάντωσης ενός σώματος. Μια απλή εκτίμηση για τη βασική συχνότητα ιδιοταλάντωσης δίνεται από τη σχέση

$$f_0 = \sqrt{\bar{\rho}/4\pi} \tag{1.8}$$

όπου $\bar{\rho} = 3M/4\pi R^3$ είναι η μέση πυκνότητα της μάζας-ενέργειας της πηγής.

Σε συμπαγή διπλά συστήματα, οι βασικές συχνότητες ιδιοταλάντωσης μπορεί είναι της ίδιας τάξης μεγέθους με τη συχνότητα περιφοράς λίγο πριν τη σύγκρουση των δύο αστέρων.

1.2 ΠΗΓΕΣ ΒΑΡΥΤΙΚΗΣ ΑΚΤΙΝΟΒΟΛΙΑΣ

Μέχρι να γίνουν άμεσες παρατηρήσεις βαρυτικών κυμάτων, η κατανόηση που υπάρχει για τις σημαντικότερες πηγές βαρυτικής ακτινοβολίας θα βασίζεται σε εκτιμήσεις. Η ανιχνευσιμότητα μιας πηγής καθορίζεται τόσο από την απόλυτη λαμπρότητα της πηγής, όσο και από την απόσταση στην οποία βρίσκεται. Ωστόσο, καθοριστική είναι η σημασία της χωρικής πυκνότητας των πηγών, έτσι ώστε η συχνότητα παρατήρησης να είναι ικανοποιητική. Παρακάτω, θα περιγράψουμε εν συντομία αυτές που αναμένεται να αποτελέσουν τις σημαντικότερες πηγές βαρυτικής ακτινοβολίας. Για περισσότερες λεπτομέρειες σχετικά με πηγές βαρυτικής ακτινοβολίας δείτε [Schutz, 1999] και [Andersson & Kokkotas, 2003], καθώς και στις αναφορές των δημοσιεύσεων αυτών. Στο Σχήμα 1.2 παρουσιάζονται συνοπτικά οι πηγές βαρυτικής ακτινοβολίας στις οποίες θα αναφερθούμε (διάγραμμα ακτίνας – μάζας).

ΠΕΡΙΣΤΡΕΦΟΜΕΝΟΙ ΑΣΤΕΡΕΣ ΝΕΤΡΟΝΙΩΝ

Αναμένεται πως ένας αστέρας νετρονίων ακτίνας R με συχνότητα περιστροφής $f \sim 300Hz$ και μια προεξοχή στην επιφάνειά του μάζας, m, θα εκπέμπει ακτινοβολία πλάτους (προσέγγιση τετραπολικής τάξης)

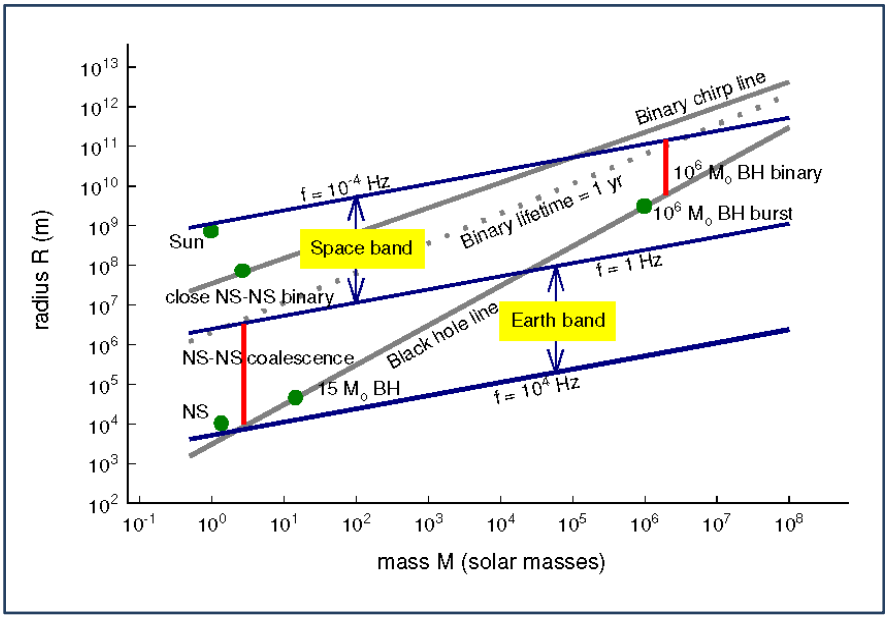
$$h \sim 2(2\pi f)^2 m/r \tag{1.9}$$

ενώ η φωτεινότητά του θα δίνεται προσεγγιστικά από την σχέση

$$L \sim (1/5)(2\pi f)^6 m^2 R^4 \tag{1.10}$$

Θεωρείται ότι ο μηχανισμός επιβράδυνσης της περιστροφής ενός τέτοιου αστέρα νετρονίων, δεν μπορεί να είναι αποκλειστικά η εκπομπή βαρυτικής ακτινοβολίας (δεδομένου ότι η μάζα της προεξοχής είναι περίπου $m \sim 10^{-5} M_\odot$). Παρόλο αυτά ενδέχεται να εκπέμπουν σημαντικά ποσά

βαρυτικής ακτινοβολίας, ικανής να ανιχνευτεί από τους πρώτους ανιχνευτές. Ιδιαίτερο ενδιαφέρον παρουσιάζουν οι περιστρεφόμενοι αστέρες νετρονίων που ανήκουν σε διπλά συστήματα χαμηλής μάζας, που εκπέμπουν ακτινοβολία X (Low-Mass X-ray Binaries, LMXBs). Τα συστήματα αυτά ενδεχομένως να αποτελέσουν μακροχρόνιες πηγές βαρυτικής ακτινοβολίας.



Σχήμα 1. 2. Οι σημαντικότερες πηγές βαρυτικής ακτινοβολίας, διάγραμμα ακτίνας – μάζας (Σχήμα 2 από Schutz, 1999).

ΔΙΠΛΑ ΣΥΣΤΗΜΑΤΑ ΑΣΤΕΡΩΝ

Τα διπλά συστήματα αστέρων αποτελούν μια ακόμη ενδεχόμενη πηγή βαρυτικής ακτινοβολίας. Το πλάτος των βαρυτικών κυμάτων από αυτές τις πηγές (για σύστημα αστέρων με ίδια μάζα, M, και ακτίνα, R, σε κυκλική τροχιά) αναμένεται να είναι κατά προσέγγιση

$$h \sim M^2/(2rR) \tag{1.11}$$

Ενώ η φωτεινότητα αυτής θα είναι

$$L \sim \frac{1}{80}\left(\frac{M}{R}\right)^5 \tag{1.12}$$

Με απλούς υπολογισμούς εύκολα μπορούμε να διαπιστώσουμε ότι η ενέργεια που χάνουν τα συστήματα αυτά μέσω εκπομπής βαρυτικής ακτινοβολίας είναι πολύ περισσότερη σε σχέση μ' αυτή που χάνουν με εκπομπή ηλεκτρομαγνητικής ακτινοβολίας. Η απώλεια αυτή έχει ως αποτέλεσμα τη συρρίκνωση της τροχιάς τους και την αύξηση της συχνότητας περιφοράς. Η χρονική κλίμακα στην οποία συμβαίνει το φαινόμενο εξαρτάται από τα χαρακτηριστικά του συστήματος (συγκεκριμένα, από την ποσότητα $\mathcal{M} = \mu^{3/5}(M_1 + M_2)^{2/5}$, όπου μ η ανηγμένη μάζα του συστήματος) και ο προσδιορισμός της οδηγεί στον καθορισμό των μαζών του συστήματος. Επιπλέον, ο καθορισμός του πλάτους μπορεί οδηγήσει στον προσδιορισμό της απόστασης του συστήματος.

Όσο τα συστήματα αυτά βρίσκονται σε πρώιμα στάδια της ζωής τους, θα εκπέμπουν βαρυτική ακτινοβολία σε χαμηλές συχνότητες η οποία δεν μπορεί να ανιχνευτεί από τους επίγειους ανιχνευτές. Ωστόσο, πριν την οριστική κατάρρευσή τους, τα συστήματα αυτά εισέρχονται στο επίγειο παράθυρο παρατήρησης. Έτσι, π.χ. για διπλά συστήματα αποτελούμενα από αστέρες νετρονίων, που βρίσκονται στην τελευταία φάση της ζωής τους, αναμένεται αύξηση της συχνότητας περιφοράς από τα $10Hz$ στα $\sim 1kHz$ για τους τελευταίους 10^4 κύκλους. Ο ρυθμός παρατήρησης τέτοιων γεγονότων αναμένεται να είναι έως 3 το χρόνο με την πρώτη γενιά ανιχνευτών, ενώ αναμένεται να αυξηθεί σημαντικά με τη δεύτερη γενιά ανιχνευτών.

ΚΑΝΟΝΙΚΟΙ ΤΡΟΠΟΙ ΤΑΛΑΝΤΩΣΗΣ ΑΣΤΕΡΩΝ ΝΕΤΡΟΝΙΩΝ

Μια ακόμη ενδιαφέρουσα πηγή βαρυτικής ακτινοβολίας αναμένεται να αποτελέσουν οι κανονικοί τρόποι ταλάντωσης των αστέρων νετρονίων. Μέχρι σήμερα είναι γνωστοί αρκετοί τέτοιοι μη ακτινικοί τρόποι δόνησης (f-, g-, p-, w-, r-modes), οι οποίοι είναι σε θέση να αποτελέσουν πηγή βαρυτικής ακτινοβολίας. Η καταγραφή τέτοιας πληροφορίας θα ήταν εξαιρετικά χρήσιμη για τον προσδιορισμό της δομής του αστέρα νετρονίων και της καταστατικής του εξίσωσης. Η συχνότητα των πηγών αυτών αναμένεται να είναι στη περιοχή $0.5 - 10\ kHz$ και την απαιτούμενη ευαισθησία στις συχνότητες αυτές θα έχουν οι ανιχνευτές δεύτερης ή τρίτης γενιάς.

ΒΑΡΥΤΙΚΗ ΚΑΤΑΡΡΕΥΣΗ

Η βαρυτική κατάρρευση που προηγείται της δημιουργίας ενός αστέρα νετρονίων οδηγεί σε μια έκρηξη υπερκαινοφανούς (supernova). Πιθανολογείται πως το σήμα ενός τέτοιου γεγονότος θα είναι μικρής διάρκειας με συχνότητα ανάμεσα στα $100Hz$ και στα $10kHz$. Αν η ενέργεια που εκπέμπεται κατά την έκρηξη είναι μεγαλύτερη από $0.01M_{\odot}$, τότε οι ανιχνευτές δεύτερης γενιάς θα είναι σε θέση να ανιχνεύσουν τα κύματα μιας τέτοιας έκρηξης στο τοπικό σμήνος γαλαξιών (σμήνος της Παρθένου).

ΑΣΤΡΙΚΕΣ ΜΕΛΑΝΕΣ ΟΠΕΣ

Είναι γενικά αποδεκτό πως υπάρχουν δύο κατηγορίες μελανών οπών: οι αστρικές μελανές οπές και οι γαλαξιακές μελανές οπές. Στην πρώτη κατηγορία ανήκουν εκείνες που έχουν μάζες $\sim 10M_{\odot}$ και προέρχονται από την εξέλιξη των αστέρων, ενώ στην δεύτερη κατηγορία ανήκουν οι μελανές οπές που βρίσκονται στο κέντρο των γαλαξιών και έχουν μάζες μεταξύ $10^6 M_{\odot}$ και $10^{10} M_{\odot}$, επιπλέον υπάρχουν ισχυρές ενδείξεις για την ύπαρξη μελανών οπών ενδιάμεσης μάζας σε γαλαξιακά σμήνη. Όλες κατηγορίες μελανών οπών εκπέμπουν βαρυτική ακτινοβολία όταν βρίσκονται σε διπλά συστήματα, που το ένα ή και τα δύο μέλη τους είναι μελανές οπές. Αν και ο σχηματισμός μελανής οπής είναι γενικά σπανιότερος από το σχηματισμό αστέρα νετρονίων, διπλά συστήματα με μία ή δύο μελανές οπές αναμένεται να υπάρχουν σε αριθμό συγκρίσιμο με αυτό αυτόν διπλών συστημάτων αστέρων νετρονίων.

Η μεγάλη μάζα των μελανών οπών είναι σε θέση να καταστήσει την εκπεμπόμενη από το διπλά σύστημα βαρυτική ακτινοβολία παρατηρήσιμα σε πολύ μεγάλες αποστάσεις. Έτσι, αν το πλήθος των συστημάτων αυτών είναι τόσο, όσο εκτιμάται, τότε αποτελούν ενδεχομένως τις καλύτερες προς ανίχνευση πηγές από τους ανιχνευτές πρώτης γενιάς.

ΓΑΛΑΞΙΑΚΕΣ ΜΕΛΑΝΕΣ ΟΠΕΣ

Μια ακόμη πηγή βαρυτικής ακτινοβολίας αναμένεται να αποτελέσουν οι γαλαξιακές μελανές οπές. Αυτές εντοπίζονται στο κέντρο των περισσότερων γαλαξιών και μπορούν να αποτελέσουν πηγή βαρυτικών κυμάτων, είτε περιφερόμενες η μια γύρω από κάποια άλλη συνιστώντας ένα διπλό σύστημα μεγαλύτερης κλίμακας σε σχέση με αυτά που αναφέρθηκαν παραπάνω, είτε κατά το σχηματισμό τους. Και στις δύο περίπτωσεις η συχνότητα των βαρυτικών κυμάτων αναμένεται να είναι μικρή αλλά το πλάτος μεγάλο. Τα παραπάνω χαρακτηριστικά καθιστούν τα αντικείμενα αυτά από τους πρώτους στόχους του διαστημικού ανιχνευτή, LISA, που αναμένεται να υλοποιηθεί την επόμενη δεκαετία.

ΒΑΡΥΤΙΚΑ ΚΥΜΑΤΑ ΑΠΟ ΤΗΝ ΜΕΓΑΛΗ ΕΚΡΗΞΗ

Μια από τις σημαντικότερες μετρήσεις που αναμένεται να κάνουν οι ανιχνευτές βαρυτικών κυμάτων είναι η βαρυτική ακτινοβολία από την μεγάλη έκρηξη. Σε αντίθεση με τη μικροκυματική ακτινοβολία υποβάθρου, που αντικατοπτρίζει την εικόνα του Σύμπαντος 10^5 χρόνια μετά τη δημιουργία του, και μελέτες πάνω στη νουκλεοσύνθεση, που δίνουν την εικόνα του Σύμπαντος 3 λεπτά μετά τη μεγάλη έκρηξη, η βαρυτική ακτινοβολία περιέχει πληροφορίες για την εικόνα του Σύμπαντος $10^{-24} sec$ μετά τη μεγάλη έκρηξη. Έτσι, αναμένονται πολύτιμες πληροφορίες που θα επιβεβαιώσουν ή θα απορρίψουν τις προβλέψεις των σημερινών θεωριών.

Ωστόσο, αν και αναμένεται η ακτινοβολία αυτή να καλύπτει όλο το φάσμα, το εξαιρετικά μικρό πλάτος της, την καθιστά πολύ δύσκολα ανιχνεύσιμη, αφού στα περισσότερα μήκη κύματος θα καλύπτεται από θόρυβο (εξαιτίας του ανθρώπινου παράγοντα ή από άλλες πηγές βαρυτικής ακτινοβολίας). Θεωρείται πως το καλύτερο παράθυρο ανίχνευσης θα είναι αυτό κάτω των $10 \mu Hz$, που θα είναι σε θέση να μελετηθεί από μελλοντικούς διαστημικούς ανιχνευτές.

1.3 ΑΝΙΧΝΕΥΣΗ ΒΑΡΥΤΙΚΗΣ ΑΚΤΙΝΟΒΟΛΙΑΣ – ΑΝΙΧΝΕΥΤΕΣ

Στην ενότητα αυτή θα αναφερθούμε στις δυσκολίες που ενέχει η παρατήρηση βαρυτικών κυμάτων και θα παρουσιάσουμε μερικά στοιχεία για τους ανιχνευτές που έχουν κατασκευαστεί ή αναμένεται να κατασκευαστούν. Για περισσότερες λεπτομέρειες ο αναγνώστης παραπέμπεται στις εργασίες [Schutz, 1999], [Sathyaprakash, 2005] και [Grishchuk et al., 2001].

ΑΝΙΧΝΕΥΣΗ ΒΑΡΥΤΙΚΗΣ ΑΚΤΙΝΟΒΟΛΙΑΣ

Το επίπεδο θορύβου των επίγειων ανιχνευτών βαρυτικών κυμάτων, αναμένεται να είναι το ίδιο ή και μεγαλύτερο από το σήμα που θα πρέπει να ανιχνεύσουν. Τέτοια σήματα ωστόσο, μπορούν να ανιχνευτούν αν είναι γνωστή η αναμενόμενη κυματομορφή με αρκετή ακρίβεια. Η τεχνική αυτή, της εξαγωγής μέσα από ένα σύνολο δεδομένων μιας γνωστής κυματομορφής, λέγεται matched filtering.

Σύμφωνα με τη τεχνική αυτή η έξοδος του ανιχνευτή πολλαπλασιάζεται με μία συνάρτηση χρόνου (πρότυπο) και τα αποτελέσματα ολοκληρώνονται. Αν μέσα στα δεδομένα υπάρχει κάποιο σήμα, η έξοδος του φίλτρου θα είναι υψηλότερη απ' ότι η αντίστοιχη για καθαρό θόρυβο. Ένα απλό παράδειγμα ενός τέτοιου φίλτρου είναι ο μετασχηματισμός Fourier που είναι το κατάλληλο φίλτρο για σήμα σταθερής συχνότητας.

Η τεχνική mached filtering είναι αρκετά απαιτητική από υπολογιστικής άποψης. Οι λόγοι γι' αυτό είναι οι εξής:

- Ο χρόνος άφιξης ενός σήματος μικρής διάρκειας είναι εν γένει άγνωστος, επομένως η φόρμα θα πρέπει να πολλαπλασιάζεται με τη ροή των δεδομένων, θεωρώντας πως κάθε καινούρια μέτρηση θα μπορούσε να αποτελέσει την έναρξη του σήματος.
- Η κυματομορφή η οποία αναμένεται από κάποια πηγή συνήθως εξαρτάται από ορισμένα χαρακτηριστικά της, π.χ. η κυματομορφή ενός διπλού συστήματος εξαρτάται από τη μάζα $\mathcal{M} = \mu^{\frac{3}{5}}(M_1 + M_2)^{2/5}$. Ως εκ τούτου θα πρέπει να ελεγχθούν πολλές κυματομορφές κάθε μία για διαφορετικές παραμέτρους και διαφορετικές τιμές των παραμέτρων αυτών.
- Τέλος, η τεχνική αυτή λειτουργεί ικανοποιητικά όσο το πρότυπο παραμένει σε σύμπτωση φάσης με το σήμα για το σύνολο των δεδομένων. Κάτι τέτοιο απαιτεί γνώση της κυματομορφής με εξαιρετική ακρίβεια, ιδιαίτερα για μεγάλης διάρκειας σήματα.

Η ανιχνευσιμότητα ενός σήματος εξαρτάται από ενεργό πλάτος του, h_{eff}, το οποίο καθορίζεται ως εξής

$$h_{eff} = h\sqrt{N_{cycles}} \tag{1.13}$$

όπου N_{cycles} είναι ο αριθμός κύκλων της κυματομορφής που έχουν εναρμονιστεί με την φόρμα, π.χ. για έναν αστέρα νετρονίων που εκπέμπει βαρυτική ακτινοβολία λόγω κάποιας προεξοχής στην επιφάνειά του, το ενεργό πλάτος θα είναι $h_{eff} = h_{bump}\sqrt{2fT_{obs}}$, όπου T_{obs} ο χρόνος παρατήρησης. Για να ανιχνευτεί η ακτινοβολία θα έπρεπε να παρατηρείται ο αστέρας αυτός για περίπου 4 μήνες.

Συνοψίζοντας, θα μπορούσαμε να πούμε πως η ευαισθησία των ανιχνευτών περιορίζεται, όχι μόνο από τεχνολογικής άποψης, αλλά και από τη διάρκεια της παρατήρησης, την ακρίβεια στη γνώση της αναμενόμενης κυματομορφής και την απαιτούμενη υπολογιστική ισχύ για την ανάλυση των δεδομένων.

ΑΝΙΧΝΕΥΤΕΣ ΒΑΡΥΤΙΚΗΣ ΑΚΤΙΝΟΒΟΛΙΑΣ

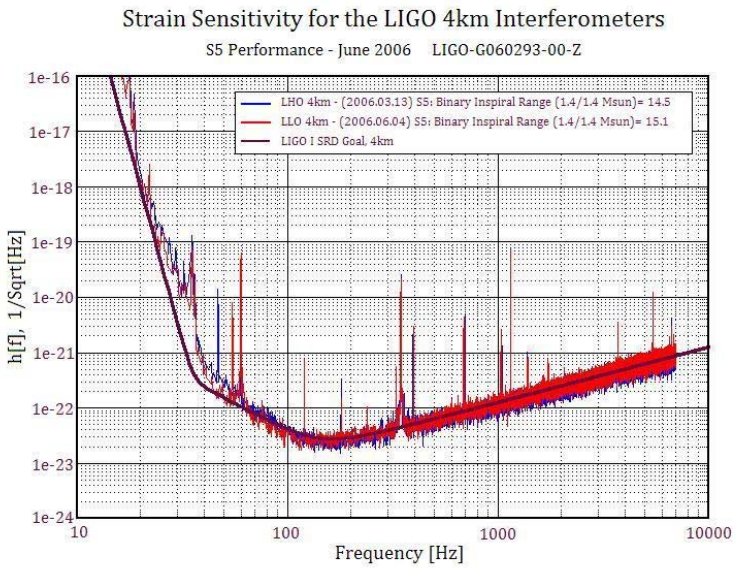
Υπάρχουν δύο τύποι ανιχνευτών βαρυτικών κυμάτων, που είναι σε λειτουργία: i) οι ράβδοι συντονισμού και ii) τα συμβολόμετρα λέιζερ. Γενικά, η ευαισθησία ενός ανιχνευτή καθορίζεται από τη μικρότερη δυνατή παραμόρφωση (προκαλούμενη από μια αστρονομική πηγή) που μπορεί να διακρίνει σε σχέση με το θόρυβο υποβάθρου και μετριέται σε $Hz^{-1/2}$ (strain sensitivity).

Επίγειοι Ανιχνευτές (Συμβολόμετρα)

Τα επίγεια συμβολόμετρα είναι ευαίσθητα στην περιοχή των 10-1000 Hz. Αποτελούνται από δύο βραχίονες κάθετους μεταξύ τους, καθένας από τους οποίους διατρέχεται από μια ακτίνα λέιζερ (συμβολόμετρο Michelson). Η μέτρηση κάποιας παραμόρφωσης εκφράζεται με διαφορά φάσης μεταξύ των δύο δεσμών.

Η ευαισθησία των οργάνων αυτών περιορίζεται στις μικρές συχνότητες (10-40 Hz) από τον ανθρώπινο παράγοντα και σεισμικές διαταραχές, στις μέσες συχνότητες (40-300 Hz) από θερμικό θόρυβο στις οπτικές διατάξεις και στις διατάξεις αιώρησης και τέλος στις υψηλές συχνότητες (>300 Hz) από θόρυβο ανάκλασης φωτονίων (photon shot noise).

Αυτή την εποχή λειτουργούν έξι τέτοιοι ανιχνευτές. Ο LIGO (Laser Interferometer Gravitational-Wave Observatory, ΗΠΑ) αποτελεί ένα δίκτυο τριών ανιχνευτών, δύο με βραχίονες μήκους 4km, και έναν με μήκος 2km. Ο VIRGO (Γαλλία-Ιταλία) με βραχίονες μήκους 3km, Ο GEO600 (Γερμανία-Βρετανία) με βραχίονες μήκους 600m, Ο TAMA (Ιαπωνία) με βραχίονες μήκους 100m και τέλος μια δοκιμαστική κατασκευή στην Αυστραλία με βραχίονες 80m και προοπτική να γίνουν 1km. Οι ανιχνευτές αυτοί αποτελούν την πρώτη γενιά, ενώ οι περισσότεροι αναμένεται να αναβαθμιστούν στις αρχές της επόμενης δεκαετίας και να αποτελέσουν τη δεύτερη γενιά ανιχνευτών, με αυξημένη ευαισθησία κατά έναν παράγοντα 10-15.

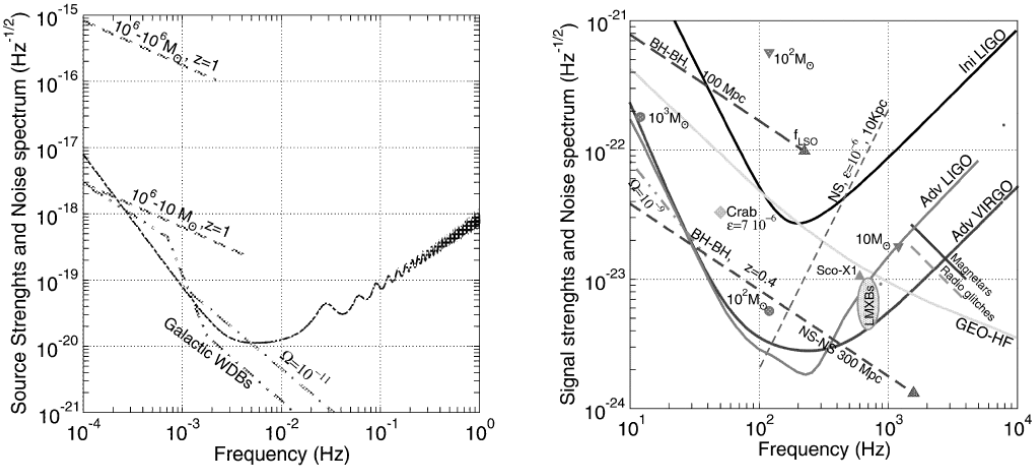


Σχήμα 1. 3. Σύγκριση εκτιμώμενης (συμπαγής γραμμή) και πραγματικής ευαισθησίας του LIGO κατά τη διάρκεια μιας δοκιμαστικής μέτρησης (Ιούνιος 2006). Το LIGO έχει επιτύχει τους κατασκευαστικούς του στόχους (πηγή: http://www.ligo-wa.caltech.edu).

Στο Σχήμα 1.3 παρουσιάζεται μια σύγκριση της εκτιμώμενης και της πραγματικής ευαισθησίας του LIGO. Είναι εμφανές ότι ο ανιχνευτής αυτός έχει επιτύχει τους κατασκευαστικούς του στόχους.

Διαστημικοί Ανιχνευτές (Συμβολόμετρα)

Η αδυναμία παρατήρησης των επίγειων ανιχνευτών σε χαμηλές συχνότητες μπορεί να ξεπεραστεί με την κατασκευή ενός διαστημικού ανιχνευτή. Τέτοιος αναμένεται να είναι ο LISA (Laser Interferometer Space Antenna) που αποτελεί συνεργασία των NASA και ESA και προβλέπεται να ολοκληρωθεί στα μέσα της επόμενης δεκαετίας. Ο ανιχνευτής θα αποτελείται από 3 δορυφόρους σε ηλιοκεντρική τροχιά και σε σχηματισμό ισόπλευρου τριγώνου, πλευράς $5 \times 10^6 Km$. Στο Σχήμα 1.4 παρουσιάζονται συνοπτικά οι εκτιμώμενες δυνατότητες των μελλοντικών ανιχνευτών με ορισμένες πηγές βαρυτικής ακτινοβολίας που θα είναι σε θέση να ανιχνεύσουν.



Σχήμα 1. 4. Αριστερά: Η εκτιμώμενη ευαισθησία του διαστημικού ανιχνευτή (LISA) σε σύγκριση με την ένταση της ακτινοβολίας των πρώτων στόχων του. Δεξιά: Η εκτιμώμενη ευαισθησία των ανιχνευτών LIGO (initial & advanved), VIRGO (advanced) και GEOHF (advanced) σε σύγκριση με τη βαρυτική ακτινοβολία ορισμένων πηγών (Σχήμα 1 από Sathyaprakash, 2005).

Για πιο εκτεταμένη ανάλυση σχετικά με τεχνικές ανίχνευσης βαρυτικών κυμάτων και τους ανιχνευτές βαρυτικής ακτινοβολίας παραπέμπουμε τον αναγνώστη στην εργασία [Grishchuk et al., 2001].

ΚΕΦΑΛΑΙΟ 2^{Ο} "ΔΙΠΛΑ ΣΥΣΤΗΜΑΤΑ"

Στο κεφάλαιο αυτό θα περιγράψουμε το μοντέλο που χρησιμοποιούμε για να προσεγγίσουμε τα διπλά συστήματα αποτελούμενα από μελανές οπές και αστέρες νετρονίων. Πιο συγκεκριμένα η πρώτη ενότητα είναι αφιερωμένη στα ελλειψοειδή Riemann-S, με τα οποία θα προσεγγίσουμε τη δομή των αστέρων νετρονίων. Στη δεύτερη ενότητα θα περιγράψουμε την πρώτη μετανευτώνεια προσέγγιση που κάνουμε για την τροχιακή κίνηση. Στην τρίτη ενότητα θα υπολογίσουμε τις δυναμικές εξισώσεις ενός διπλού συστήματος, χωρίς να λάβουμε υπόψη την εκπομπή βαρυτικής ακτινοβολίας, ενώ στην τέταρτη ενότητα θα εισάγουμε στις εξισώσεις την εκπομπή βαρυτικής ακτινοβολίας. Τέλος, στη πέμπτη ενότητα θα αναφερθούμε στις κυματομορφές των βαρυτικών κυμάτων που θα υπολογίσουμε από τα διπλά συστήματα.

2.1 RIEMANN-S ΕΛΛΕΙΨΟΕΙΔΗ

Τα ελλειψοειδή τύπου Riemann-S, αποτελούν μια ελλειψοειδή μορφή ισορροπίας για ένα Νευτώνειο ρευστό με ιδιοβαρύτητα. Αν και αναλυτική λύση υπάρχει μόνο στην περίπτωση που το ρευστό είναι ασυμπίεστο, αποδεικνύεται πως ακόμη και όταν διέπονται από μια καταστατική εξίσωση της μορφής $P = K\rho^{1+1/n}$, μπορούν να θεωρηθούν προσεγγιστικά ως μορφές ισορροπίας. Τα ελλειψοειδή αυτά θα αποτελέσουν το μοντέλο για τον αστέρα νετρονίων, στα συστήματα που θα μελετήσουμε. Για περισσότερες πληροφορίες σχετικά με τα Riemann-S ελλειψοειδή παραπέμπουμε τον αναγνώστη στις εργασίες [Lai et al., 1993, 1994]. Άλλοι τρόποι προσέγγισης της αστρικής δομής έχουν γίνει στις δημοσιεύσεις [Spyrou & Kokkotas, 1994] και [Kokkotas & Schäfer, 1995].

ΥΠΟΘΕΣΕΙΣ

Όπως αναφέρθηκε και παραπάνω, η πρώτη υπόθεση που κάνουμε είναι, ότι το ρευστό διέπεται από μια καταστατική εξίσωση της μορφής

$$P = K\rho^{1+1/n} \tag{2.1}$$

όπου P η πίεση, ρ η πυκνότητα και n ο πολυτροπικός δείκτης. Επιπλέον, υποθέτουμε πως οι επιφάνειες σταθερής πυκνότητας στο εσωτερικό του αστέρα αποτελούν όμοια ελλειψοειδή. Με την προϋπόθεση αυτή, η γεωμετρία του αστέρα καθορίζεται πλήρως από τους τρεις κύριους άξονες της εξωτερικής επιφάνειας του ελλειψοειδούς. Ακόμη, θεωρούμε ότι η κατανομή της πυκνότητας, $\rho(m)$ στο εσωτερικό του αστέρα, όπου m η μάζα στο εσωτερικό μιας ισόπυκνης επιφάνειας, είναι ίδια με

την αντίστοιχη ενός σφαιρικού πολύτροπου του ίδιου όγκου. Στο σύστημα αναφοράς που κινείται μαζί με το κέντρο μάζας του αστέρα, το πεδίο ταχυτήτων του ρευστού θεωρείται ως γραμμική υπέρθεση των εξής τριών συνιστωσών: i) μια άκαμπτη περιστροφή της ελλειψοειδούς μορφής, ii) μια εσωτερική κίνηση του ρευστού με ομοιόμορφο στροβιλισμό και iii) συστολές ή διαστολές των τριών αξόνων του ελλειψοειδούς.

Υπό αυτές τις προϋποθέσεις, ο αριθμός των εσωτερικών βαθμών ελευθερίας για κάθε αστέρα είναι εννιά: οι τρεις κύριοι άξονες του ελλειψοειδούς, a_1, a_2, a_3, οι τρεις συνιστώσες της γωνιακής ταχύτητας, $\Omega_1, \Omega_2, \Omega_3$ και οι τρεις συνιστώσες του διανύσματος στροβιλότητας, $\zeta_1, \zeta_2, \zeta_3$. Για την περίπτωση των Riemann-S ελλειψοειδών τόσο το διάνυσμα της γωνιακής ταχύτητας, όσο και της στροβιλότητας είναι παράλληλα με έναν από τους κύριους άξονες (επιλέγουμε να είναι παράλληλα με τον z-άξονα, που αντιστοιχεί στον a_3 του ελλειψοειδούς). Με τους παραπάνω περιορισμούς, οι εξισώσεις Euler που περιγράφουν τη κίνηση ιδανικού ρευστού καθίστανται συνήθεις διαφορικές εξισώσεις για τις πέντε μεταβλητές που προσδιορίζουν πλήρως το σύστημα, δηλαδή τους τρεις άξονες a_1, a_2, a_3, τη γωνιακή ταχύτητα, $\boldsymbol{\Omega} = \Omega \boldsymbol{e}_3$ και τη γωνιακή συχνότητα της εσωτερικής κίνησης του ρευστού, Λ.

ΕΝΕΡΓΕΙΑ ΚΑΙ ΣΥΝΑΡΤΗΣΗ LAGRANGE

Αν συμβολίσουμε με $\{\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3\}$ τη μοναδιαία βάση κατά μήκος των στιγμιαίων διευθύνσεων των τριών κύριων αξόνων του ελλειψοειδούς, που περιστρέφεται κατά τον άξονα $\boldsymbol{e}_3$, τότε το πεδίο ταχυτήτων στο περιστρεφόμενο σύστημα αναφοράς θα έχει ως εξής

$$\boldsymbol{u}_{rot} = Q_1 x_2 \boldsymbol{e}_1 + Q_2 x_1 \boldsymbol{e}_2 \tag{2.2}$$

Οι ποσότητες Q_1, Q_2 δίνονται από τις παρακάτω σχέσεις

$$Q_1 = -\frac{a_1^2}{a_1^2 + a_2^2}\zeta = +\frac{a_1}{a_2}\Lambda \tag{2.3}$$

$$Q_2 = +\frac{a_2^2}{a_1^2 + a_2^2}\zeta = -\frac{a_2}{a_1}\Lambda \tag{2.4}$$

Η δε στροβιλότητα, ζ, στο περιστρεφόμενο σύστημα αναφοράς θα δίνεται από τη σχέση

$$\zeta = (\boldsymbol{\nabla} \times \boldsymbol{u}_{rot}) \cdot \boldsymbol{e}_3 = -\frac{a_1^2 + a_2^2}{a_1 a_2}\Lambda \tag{2.5}$$

Στο δε αδρανειακό σύστημα αναφοράς το πεδίο ταχυτήτων, για έναν αστέρα που δεν είναι σε ισορροπία θα έχει ως εξής

$$\boldsymbol{u} = \boldsymbol{u}_s + \boldsymbol{u}_e \tag{2.6}$$

Ο όρος $\boldsymbol{u}_s$, αφορά το πεδίο ταχυτήτων ενός ελλειψοειδούς σε ισορροπία, στο αδρανειακό σύστημα αναφοράς και συνδέεται με το αντίστοιχο στο περιστρεφόμενο σύστημα αναφοράς με τη σχέση

$$\boldsymbol{u}_s = \boldsymbol{u}_{rot} + \boldsymbol{\Omega} \times \boldsymbol{x} \tag{2.7}$$

Αντικαθιστώντας την (2.2) στη παραπάνω σχέση και λαμβάνοντας υπόψη τις (2.3) και (2.4) καταλήγουμε για το $\boldsymbol{u}_s$στη σχέση

$$\boldsymbol{u}_s = \left(\frac{a_1}{a_2}\Lambda - \Omega\right) x_2 \boldsymbol{e}_1 + \left(-\frac{a_2}{a_1}\Lambda + \Omega\right) x_1 \boldsymbol{e}_2 \tag{2.8}$$

Ο δε όρος $\boldsymbol{u}_e$, αφορά το πεδίο ταχυτήτων αστέρων, που δεν είναι σε ισορροπία και περιγράφει τις ταλαντώσεις κατά μήκος των τριών αξόνων του ελλειψοειδούς. Δίνεται από τη σχέση

$$\boldsymbol{u}_e = \frac{\dot{a}_1}{a_1} x_1 \boldsymbol{e}_1 + \frac{\dot{a}_2}{a_2} x_2 \boldsymbol{e}_2 + \frac{\dot{a}_3}{a_3} x_3 \boldsymbol{e}_3 \tag{2.9}$$

Από τα παραπάνω, εύκολα μπορεί να υπολογιστεί η κινητική ενέργεια του αστέρα. Θα είναι

$$T = \int \frac{1}{2} \rho \boldsymbol{u}^2 d^3x = T_s + T_e \tag{2.10}$$

Η ενέργεια T_s είναι η κινητική ενέργεια λόγω της περιστροφής του αστέρα και της εσωτερικής κυκλοφορίας του ρευστού και υπολογίζεται πως είναι

$$T_s = \frac{1}{2} I(\Lambda^2 + \Omega^2) - \frac{2}{5} \kappa_n M a_1 a_2 \Lambda \Omega \tag{2.11}$$

όπου I είναι η ροπή αδράνειας

$$I = \frac{1}{5} \kappa_n M (a_1^2 + a_2^2) \tag{2.12}$$

M η μάζα του αστέρα και κ_n είναι μια αδιάστατη σταθερή που εξαρτάται αποκλειστικά από τον πολυτροπικό δείκτη, n, της καταστατικής εξίσωσης του αστέρα και υπολογίζεται από τη σχέση

$$\kappa_n \equiv \frac{5}{3} \frac{\int_0^{\xi_1} \theta^n \xi^4 d\xi}{\xi_1^4 |\theta_1'|} \tag{2.13}$$

όπου θ και ξ είναι οι μεταβλητές Lane-Emden, για το αντίστοιχο σφαιρικό μοντέλο.

Η δε ενέργεια T_e είναι η κινητική ενέργεια που σχετίζεται με τις συστολές και διαστολές της επιφάνειας του ελλειψοειδούς και υπολογίζεται πως είναι

$$T_e = \frac{1}{10} \kappa_n M (\dot{a}_1^2 + \dot{a}_2^2 + \dot{a}_3^2) \tag{2.14}$$

Έτσι, από τις σχέσεις (2.10), (2.11) και (2.14) έχουμε τη συνολική έκφραση της κινητικής ενέργειας, η οποία θα είναι

$$T = \frac{1}{2} I(\Lambda^2 + \Omega^2) - \frac{2}{5} \kappa_n M a_1 a_2 \Lambda \Omega + \frac{1}{10} \kappa_n M (\dot{a}_1^2 + \dot{a}_2^2 + \dot{a}_3^2) \tag{2.15}$$

Η συνολική εσωτερική ενέργεια του ρευστού είναι

$$U = \int \frac{nP}{\rho} dm = k_1 K \rho_c^{1/n} M \tag{2.16}$$

όπου ρ_cείναι η κεντρική πυκνότητα του ελλειψοειδούς (ίση με εκείνη ενός σφαιρικού πολύτροπου με την ίδια μάζα και τον ίδιο όγκο) και k_1 μια σταθερή που εξαρτάται από τον πολυτροπικό δείκτη και δίνεται από τη σχέση

$$k_1 = \frac{n(n+1)}{5-n} \xi_1 |\theta_1'| \tag{2.17}$$

Επιπλέον, η βαρυτική δυναμική ενέργεια σύνδεσης θα δίνεται από τη σχέση

$$W = -\frac{3}{5-n}\frac{GM^2}{R}f \tag{2.18}$$

όπου $R \equiv (a_1 a_2 a_3)^{1/3}$ η μέση ακτίνα του ελλειψοειδούς και

$$f = \frac{\mathcal{G}}{2R^2} \ , \qquad \mathcal{G} = A_1 a_1^2 + A_2 a_2^2 + A_3 a_3^2 \tag{2.19}$$

όπου οι όροι A_i (με $i = 1,2,3$) είναι τα ολοκληρώματα

$$A_i \equiv a_1 a_2 a_3 \int_0^\infty \frac{du}{\Delta(a_i^2+u)} \ , \quad \Delta^2 = (a_1^2+u)(a_2^2+u)(a_3^2+u) \tag{2.20}$$

Τέλος, η συνάρτηση Lagrange, που καθορίζει τη δυναμική του ελλειψοειδούς θα είναι

$$L(q_i, \dot{q}_i) = T - U - W \tag{2.21}$$

Οι γενικευμένες συντεταγμένες θα είναι $\{q_i\} = \{a_1, a_2, a_3, \gamma, \psi\}$ και οι γενικευμένες ταχύτητες θα είναι $\{\dot{q}_i\} = \{\dot{a}_1, \dot{a}_2, \dot{a}_3, \ \Omega, \Lambda\}$. Οι γωνίες γ και ψ έχουν οριστεί έτσι, ώστε $\dot{\gamma} = \Omega$ και $\dot{\psi} = \Lambda$. Έτσι, αντικαθιστώντας τις σχέσεις (2.15), (2.16) και (2.18) στην (2.21) θα καταλήξουμε στην παρακάτω έκφραση για τη συνάρτηση Lagrange

$$L = \frac{1}{2}I(\Lambda^2+\Omega^2) - \frac{2}{5}\kappa_n M a_1 a_2 \Lambda\Omega + \frac{1}{10}\kappa_n M(\dot{a}_1^2+\dot{a}_2^2+\dot{a}_3^2) - k_1 K \rho_c^{1/n} M + \frac{3}{5-n}\frac{GM^2}{2R^3}\mathcal{G} \tag{2.22}$$

ΠΕΡΙΟΡΙΣΜΟΙ ΣΤΑ RIEMANN-S ΕΛΛΕΙΨΟΕΙΔΗ

Προφανώς, τα ελλειψοειδή τύπου Riemann-S δεν αποτελούν την κατάλληλη προσέγγιση για περιπτώσεις όπου έχουμε αστρικές συγκρούσεις που είναι σε θέση να προκαλέσουν την πλήρη διαταραχή του αστέρα. Ακόμη και σε πολύ πιο ήπια γεγονότα δυναμικής εξέλιξης, το εν λόγω μοντέλο παραβλέπει μια σειρά παραμέτρων που ενδέχεται να αποδειχτούν μεγάλης σημασίας για τη μελέτη του φαινομένου. Μια τέτοια περίπτωση είναι οι μικρές μη γραμμικές ταλαντώσεις του αστέρα. Για $n = 0$ μόνο οι f-τρόποι δόνησης υπάρχουν. Αντιθέτως, όταν $n \neq 0$ υπάρχουν πολλοί περισσότεροι τρόποι δόνησης, που δεν είναι διαταραχές της ελλειψοειδούς μορφής του αστέρα. Οι τρόποι αυτοί διαφεύγουν πλήρως του συγκεκριμένου μοντέλου. Στην περίπτωση του ασυμπίεστου ρευστού ($n = 0$), μόνο οι $l = 2$ τετραπολικοί f-τρόποι δόνησης μπορούν να περιγραφούν, οι οποίοι, αν και κυρίαρχοι σε πολλές περιπτώσεις διαταραχής των αστέρων, δεν αποτελούν την πλήρη περιγραφή του φαινομένου.

Με βάση τα παραπάνω, η μελέτη των διπλών συστημάτων που θα ακολουθήσει θα περιορίζεται αρκετά πριν τη τελική σύγκρουση των συνοδών. Το ουσιαστικό πλεονέκτημα του παραπάνω μοντέλου συνίσταται στο ότι δίνει βασικές πληροφορίες για τη δομή του αστέρα, ενώ ταυτόχρονα είναι αρκετά απλό (συνήθεις διαφορικές εξισώσεις) ώστε να μην απαιτεί μεγάλη υπολογιστική ισχύ.

2.2 ΠΡΩΤΗ ΜΕΤΑΝΕΥΤΩΝΕΙΑ ΔΙΟΡΘΩΣΗ ΤΗΣ ΤΡΟΧΙΑΚΗΣ ΚΙΝΗΣΗΣ

Η δυσκολία της επίλυσης των εξισώσεων της γενικής θεωρίας της σχετικότητας ακόμη και για τα πιο απλά συστήματα, όπως το πρόβλημα των δύο σωμάτων, οδηγεί συνήθως σε προσεγγιστικές μεθόδους, όπως είναι οι μετανευτώνειες προσεγγίσεις. Την τεχνική αυτή θα ακολουθήσουμε στα διπλά συστήματα που θέλουμε να μελετήσουμε, ενσωματώνοντας στις εξισώσεις κίνησης την πρώτη μετανευτώνεια προσέγγιση της τροχιακής κίνησης. Παρακάτω, θα περιγράψουμε εν συντομία τον τρόπο με τον οποίο καταλήγουμε στη διορθωμένη, σε πρώτη μετανευτώνεια προσέγγιση, συνάρτηση Lagrange, για ένα σύστημα δύο σημειακών μαζών M_1 και M_2.

Η συνάρτηση Lagrange ενός τέτοιου συστήματος μπορεί να γραφεί στη μορφή

$$L = L_N + L_{PN} \tag{2.23}$$

όπου L_N είναι η Νευτώνεια συνεισφορά και L_{PN} η μετανευτώνεια. Εξ ορισμού θα είναι

$$\frac{L_{PN}}{L_N} \sim \frac{1}{c^2} \tag{2.24}$$

Ο υπολογισμός της παραπάνω συνάρτησης Lagrange μπορεί να γίνει από μια αρχή μεταβολών και παρουσιάζεται συνοπτικά παρακάτω, ακολουθώντας κατά βάση την εργασία [Richardson & Kelly, 1988].

Θα θεωρήσουμε αρχικά την κίνηση ενός δοκιμαστικού σωματιδίου (test particle) μάζας m_2, σ' ένα μετανευτώνειο βαρυτικό πεδίο που δημιουργείται από το σώμα M_1. Η συνάρτηση Lagrange, L_2, του m_2 προκύπτει απο την αντίστοιχη της εξίσωσης (2.23) από τη σχέση

$$L_2 = \lim_{M_2 \to 0} L/M_2 \tag{2.25}$$

Παρόμοια, αν η μάζα M_1 είχε επιλεγεί ως δοκιμαστικό σωματίδιο η συνάρτηση Lagrange, L_1, θα ήταν

$$L_1 = \lim_{M_1 \to 0} L/M_1 \tag{2.26}$$

Η L_2 θα υπολογιστεί από τη παρακάτω αρχή μεταβολών

$$\delta \int_{s_1}^{s_2} ds = 0 \tag{2.27}$$

όπου με ds συμβολίζουμε το

$$ds = \left(g_{\mu\nu} dx^\mu dx^\nu\right)^{1/2} \tag{2.28}$$

και με $g_{\mu\nu}$ το μετρικό τανυστή[1]. Αν υποθέσουμε ότι η βαρυτικό πεδίο είναι ασθενές, τότε ο μετρικός τανυστής μπορεί να γραφεί ως

$$g_{\mu\nu} = \eta_{\mu\nu} + h_{\mu\nu} \tag{2.29}$$

όπου $\eta_{\mu\nu}$ είναι ο μετρικός τανυστής του Minkowski, που περιγράφει τον επίπεδο χωροχρόνο, ενώ ο $h_{\mu\nu}$ είναι η διαταραχή της μετρικής $g_{\mu\nu}$ από τον επίπεδο χωροχρόνο και έχει τις ιδιότητες

$$\lim_{r \to \infty} h_{\mu\nu} = 0 \quad \text{και} \quad \lim_{c \to \infty} h_{\mu\nu} = 0 \tag{2.30}$$

[1] Οι ελληνικοί δείκτες κυμαίνονται από 0 ως 3, ενώ οι λατινικοί από 1ως 3.

όπου $r \equiv (x^i x^i)^{1/2}$. Η δεύτερη από τις ιδιότητες (2.30) επιτρέπει να εκφραστεί ο $h_{\mu\nu}$ ως άθροισμα δυνάμεων του $1/c$.

Συνδέοντας τη συντεταγμένη x^0 με το χρόνο μέσω της σχέσης $x^0 = ct$, η εξίσωση (2.27) μπορεί να πάρει τη μορφή

$$\delta \int_{x_1^0}^{x_2^0} \left(\frac{ds}{dx^0}\right) dx^0 = 0 \tag{2.31}$$

ή

$$\delta \int_{x_1^0}^{x_2^0} \left(g_{\mu\nu} \frac{dx^\mu}{dx^0} \frac{dx^\nu}{dx^0}\right)^{1/2} dx^0 = 0 \tag{2.32}$$

Το παραπάνω ολοκλήρωμα συνιστά τη δράση του συστήματος, ενώ η υπό ολοκλήρωση συνάρτηση αποτελεί τη συνάρτηση Lagrange. Αν αλλάξουμε τη μεταβλητή ολοκλήρωσης στην (2.31) θα έχουμε τη μορφή

$$\delta \int_{t_1}^{t_2} L_2 dt = 0 \tag{2.33}$$

με

$$L_2 = \left(g_{00} + 2\frac{g_{0i}}{c}\frac{dx^i}{dt} + \frac{g_{ij}}{c^2}\frac{dx^i}{dt}\frac{dx^j}{dt}\right)^{1/2} \tag{2.34}$$

Από τη σχέση (2.29) προκύπτουν οι ακόλουθες

$$g_{00} = 1 + h_{00} \tag{2.35α}$$

$$g_{0i} = h_{0i} \tag{2.35β}$$

$$g_{ij} = -\delta_{ij} + h_{ij} \tag{2.35γ}$$

Για να υπάρχει συνέπεια, μεταξύ της γενικής θεωρίας της σχετικότητας και της Νευτώνειας θεωρίας, οι συνιστώσες του $h_{\mu\nu}$, όταν αναπτυχθούν σε αντίστροφες δυνάμεις της c οφείλουν να έχουν τη παρακάτω μορφή

$$h_{00} = \frac{h_{00}^{(2)}}{c^2} + \frac{h_{00}^{(4)}}{c^4} + \cdots \tag{2.36α}$$

$$h_{0i} = \frac{h_{0i}^{(3)}}{c^3} + \frac{h_{0i}^{(5)}}{c^5} + \cdots \tag{2.36β}$$

$$h_{ij} = \frac{h_{ij}^{(2)}}{c^2} + \frac{h_{ij}^{(4)}}{c^4} + \cdots \tag{2.36γ}$$

Κατά συνέπεια, η συνάρτηση Lagrange L_2, μετά την αντικατάσταση των σχέσεων (2.36) και κανονικοποιώντας ως προς c^2, θα έχει ως εξής

$$L_2 = \frac{1}{2}\dot{x}^i\dot{x}^i - \frac{1}{2}h_{00}^{(2)} - \frac{1}{2c^2}\left[h_{00}^{(4)} + 2h_{0i}^{(3)}\dot{x}^i + h_{ij}^{(2)}\dot{x}^i\dot{x}^j - \frac{1}{4}(\dot{x}^i\dot{x}^i)^2 - \frac{1}{4}\left(h_{00}^{(2)}\right)^2 + \frac{1}{2}h_{00}^{(2)}\dot{x}^i\dot{x}^i\right] + O\left(\frac{1}{c^4}\right) \quad (2.37)$$

Οι συνιστώσες του αναπτύγματος $h_{\mu\nu}^{(k)}$ καθορίζονται από τις εξισώσεις πεδίου της σχετικότητας. Αν συμβολίσουμε με $\boldsymbol{r}_1$ και $\boldsymbol{r}_2$ τα διανύσματα θέσης (με συντεταγμένες $\{x_1^i\}$ και $\{x_2^i\}$ αντίστοιχα) των μαζών M_1 και m_2 αντίστοιχα, τότε θα έχουμε

$$h_{00}^{(2)} = -\frac{2GM_1}{r} \quad (2.38\alpha)$$

$$h_{00}^{(4)} = 2\left(\frac{GM_1}{r}\right)^2 - \frac{3GM_1\dot{\boldsymbol{r}}_1\cdot\dot{\boldsymbol{r}}_2}{r} \quad (2.38\beta)$$

$$h_{0i}^{(3)} = \frac{7GM_1\dot{x}_1^i}{2r} + \frac{GM_1(\dot{\boldsymbol{r}}_1\cdot\boldsymbol{r})y_1^i}{2r^3} \quad (2.38\gamma)$$

$$h_{ij}^{(2)} = -\frac{\delta_{ij}GM_1}{r} \quad (2.38\delta)$$

όπου, $r \equiv |\boldsymbol{r}_2 - \boldsymbol{r}_1|$ είναι η απόσταση μεταξύ των δύο σωμάτων και το αντίστοιχο διάνυσμα $\boldsymbol{r}$ έχει συντεταγμένες $\{y^i\}$. Με αντικατάσταση των (2.38) στην (2.37) η συνάρτηση Lagrange θα γραφεί ως εξής

$$L_2 = \frac{1}{2}\dot{\boldsymbol{r}}_2\cdot\dot{\boldsymbol{r}}_2 + \frac{GM_1}{r} + \frac{1}{c^2}\left[\frac{1}{8}(\dot{\boldsymbol{r}}_2\cdot\dot{\boldsymbol{r}}_2)^2 - \frac{1}{2}\left(\frac{GM_1}{r}\right)^2 + \frac{3GM_1}{2r}(\dot{\boldsymbol{r}}_1\cdot\dot{\boldsymbol{r}}_1 + \dot{\boldsymbol{r}}_2\cdot\dot{\boldsymbol{r}}_2) - \frac{7GM_1}{2r}\dot{\boldsymbol{r}}_1\cdot\dot{\boldsymbol{r}}_2 - \frac{GM_1}{2r^3}(\dot{\boldsymbol{r}}_1\cdot\boldsymbol{r})(\dot{\boldsymbol{r}}_2\cdot\boldsymbol{r})\right] \quad (2.39)$$

Η (2.39) είναι η έκφραση της συνάρτησης Lagrange για τη κίνηση ενός δοκιμαστικού σωματιδίου στο βαρυτικό πεδίο της M_1. Αλλάζοντας τους δείκτες θα πάρουμε την έκφραση της συνάρτησης Lagrange, L_1, για την κίνηση ενός δοκιμαστικού σωματιδίου στο βαρυτικό πεδίο της μάζας M_2. Η συνολική συνάρτηση Lagrange, εξαιτίας μη γραμμικής σύζευξης δε θα είναι απλή υπέρθεση των L_1 και L_2. Ωστόσο, θα είναι συμμετρική ως προς τις δύο μάζες, τις συντεταγμένες τους και τις ταχύτητές τους. Πιο συγκεκριμένα θα είναι αναλλοίωτη στον μετασχηματισμό

$$M_1 r_1 \rightleftharpoons M_2 r_2 \quad (2.40)$$

Επιπλέον, οι κυρίαρχοι όροι της συνάρτησης Lagrange θα είναι οι Νευτώνειοι. Με βάση τα παραπάνω και λαμβάνοντας υπόψη και τις σχέσεις (2.25) και (2.26) καταλήγουμε στην παρακάτω μορφή για τη συνάρτηση Lagrange του συστήματος

$$L = \frac{1}{2}M_1\dot{\boldsymbol{r}}_1\cdot\dot{\boldsymbol{r}}_1 + \frac{1}{2}M_2\dot{\boldsymbol{r}}_2\cdot\dot{\boldsymbol{r}}_2 + \frac{GM_1M_2}{r} + \frac{1}{8c^2}[M_1(\dot{\boldsymbol{r}}_1\cdot\dot{\boldsymbol{r}}_1)^2 + M_2(\dot{\boldsymbol{r}}_2\cdot\dot{\boldsymbol{r}}_2)^2] + \frac{GM_1M_2}{2rc^2}\left[3(\dot{\boldsymbol{r}}_1\cdot\dot{\boldsymbol{r}}_1 + \dot{\boldsymbol{r}}_2\cdot\dot{\boldsymbol{r}}_2) - 7\dot{\boldsymbol{r}}_1\cdot\dot{\boldsymbol{r}}_2 - \frac{1}{r^2}(\dot{\boldsymbol{r}}_1\cdot\boldsymbol{r})(\dot{\boldsymbol{r}}_1\cdot\boldsymbol{r}) - \frac{G}{r}(M_1+M_2)\right] \quad (2.41)$$

Η παραπάνω συνάρτηση Lagrange μπορεί να εκφραστεί και συναρτήσει των μεταβλητών της σχετικής κίνησης των δύο μαζών. Η μορφή της (μετά από κανονικοποίηση με $(M_1+M_2)/M_1M_2$) θα είναι

$$L_{1PN}=\frac{1}{2}\dot{\boldsymbol{r}}\cdot\dot{\boldsymbol{r}}+\frac{G(M_1+M_2)}{r}+\frac{1-3\nu}{8c^2}(\dot{\boldsymbol{r}}\cdot\dot{\boldsymbol{r}})^2-\frac{G^2(M_1+M_2)^2}{2r^2c^2}$$
$$+\frac{G(M_1+M_2)}{2rc^2}\left[(3+\nu)\dot{\boldsymbol{r}}\cdot\dot{\boldsymbol{r}}+\frac{\nu}{r^2}(\boldsymbol{r}\cdot\dot{\boldsymbol{r}})^2\right]+O\left(\frac{1}{c^4}\right) \tag{2.42}$$

όπου θέσαμε $\nu=M_1M_2/(M_1+M_2)^2$. Από τη παραπάνω συνάρτηση Lagrange, εύκολα μπορούμε να υπολογίσουμε τη συνάρτηση Hamilton, μ' ένα μετασχηματισμό Legendre

$$H_{1PN}=\frac{1}{2}\boldsymbol{p}\cdot\boldsymbol{p}-\frac{G(M_1+M_2)}{r}-\frac{1-3\nu}{8c^2}(\boldsymbol{p}\cdot\boldsymbol{p})^2+\frac{G^2(M_1+M_2)^2}{2r^2c^2}$$
$$-\frac{G(M_1+M_2)}{2rc^2}\left[(3+\nu)\boldsymbol{p}\cdot\boldsymbol{p}+\frac{\nu}{r^2}(\boldsymbol{r}\cdot\boldsymbol{p})^2\right]+\mathcal{O}\left(\frac{1}{c^4}\right) \tag{2.43}$$

2.3 ΔΥΝΑΜΙΚΕΣ ΕΞΙΣΩΣΕΙΣ ΓΙΑ ΔΙΠΛΑ ΣΥΣΤΗΜΑΤΑ

Για τον προσδιορισμό των εξισώσεων κίνησης ενός διπλού συστήματος θα πρέπει να καθορίσουμε τη συνάρτηση Lagrange. Οι εξισώσεις κίνησης θα προκύψουν από τις εξισώσεις Euler-Lagrange

$$\frac{d}{dt}\left(\frac{\partial L}{\partial\dot{q}_i}\right)=\frac{\partial L}{\partial q_i} \tag{2.44}$$

Παρακάτω θα υπολογίσουμε τη συνάρτηση Lagrange ενός διπλού συστήματος αποτελούμενο από αστέρες νετρονίων (η πιο γενική περίπτωση). Από τις εξισώσεις που θα προκύψουν εύκολα μπορούμε να υπολογίσουμε και τις αντίστοιχες, είτε για διπλά συστήματα αποτελούμενα από αστέρα νετρονίων και μελανή οπή, είτε αποτελούμενα από δύο μελανές οπές.

Η ΣΥΝΟΛΙΚΗ ΣΥΝΑΡΤΗΣΗ LAGRANGE

Γενικά, η συνάρτηση Lagrange ενός τέτοιου συστήματος θα είναι

$$L=L_{orb}+L_S+L_{S'} \tag{2.45}$$

όπου L_{orb} η συνεισφορά των τροχιακών όρων και L_S, L_S η συνεισφορά των αστρικών βαθμών ελευθερίας του αστέρα μάζας M και M' αντίστοιχα. Οι δύο τελευταίοι όροι δίνονται από τη σχέση (2.22). Συγκεκριμένα, θα είναι

$$L_S=\frac{1}{2}I(\Lambda^2+\Omega^2)-\frac{2}{5}\kappa_nMa_1a_2\Lambda\Omega+\frac{1}{10}\kappa_nM(\dot{a}_1^2+\dot{a}_2^2+\dot{a}_3^2)$$
$$-k_1K\rho_c^{1/n}M+\frac{3}{5-n}\frac{GM^2}{2R^3} \tag{2.46α}$$

$$L_{S'} = \frac{1}{2} I' \left(\Lambda'^2 + \Omega'^2 \right) - \frac{2}{5} \kappa'_n M' a'_1 a'_2 \Lambda' \Omega' + \frac{1}{10} \kappa'_n M' (\dot{a}'^2_1 + \dot{a}'^2_2 + \dot{a}'^2_3)$$

$$- k'_1 K'^{\rho'^{\frac{1}{n'}}_c} M' + \frac{3}{5-n'} \frac{GM'^2}{2R'^3} \qquad (2.46\beta)$$

Ο δε τροχιακός όρος μπορεί να γραφεί με την ακόλουθη μορφή

$$L_{orb} = L_{1PN} - W_q \qquad (2.47)$$

όπου L_{1PN} είναι η πρώτη μετανευτώνεια διόρθωση της συνάρτησης Lagrange (σχέση 2.42), την οποία γράφουμε παρακάτω έχοντας υπολογίσει τα εσωτερικά γινόμενα και συμβολίζοντας με M_{tot} τη συνολική μάζα του συστήματος

$$L_{orb} = \frac{GM_{tot}}{r} + \frac{1}{2} M_{tot} \left(\dot{r}^2 + r^2 \dot{\phi}^2 \right)^2 + \frac{(1-3\nu) M_{tot}}{8c^2} \left(\dot{r}^4 + 2r^2 \dot{r}^2 \dot{\phi}^2 + r^4 \dot{\phi}^4 \right)$$

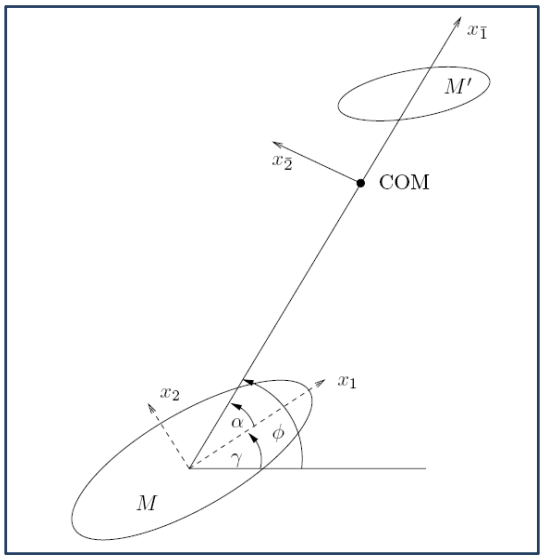
$$+ \frac{GM_{tot}{}^2}{c^2} \left[- \frac{G(M_1 + M_2)}{2r^2} + \frac{(3+2\nu) \dot{r}^2 + (3+\nu) r^2 \dot{\phi}^2}{2r} \right] \qquad (2.48)$$

ενώ W_q είναι τετραπολικός όρος του δυναμικού αλληλεπίδρασης μεταξύ των δύο μαζών και δίνεται από τη σχέση

$$W_q = - \frac{GMM' \kappa_n}{10 r^3} [a_1^2 (3 \cos^2 \alpha - 1) + a_2^2 (3 \sin^2 \alpha - 1) - a_3^2]$$

$$- \frac{GMM' \kappa'_n}{10 r^3} \left[a'^2_1 (3 \cos^2 \alpha' - 1) + a'^2_2 (3 \sin^2 \alpha' - 1) - a'^2_3 \right] \qquad (2.49)$$

όπου με α και α' συμβολίζονται οι γωνίες που σχηματίζουν οι άξονες a_1 και a'_1 αντίστοιχα με την ευθεία που ενώνει τα κέντρα των δύο αστέρων (Σχήμα 2.1).



Σχήμα 2. 1. Οι άξονες x_i του περιστρεφόμενου συστήματος αναφοράς σχηματίζουν γωνία ϕ με το αδρανειακά σύστημα. Οι στικτές γραμμές συμβολίζουν το σύστημα αναφοράς που είναι πακτωμένο στον αστέρα. Το COM αποτελεί το κέντρο μάζας του συστήματος.

Έτσι, αντικαθιστώντας στη (2.45) τις L_S και $L_{S'}$ από τις σχέσεις (2.46α) και (2.46β), και την L_{orb}, με βάση τη (2.47), από τις (2.48) και (2.49), καταλήγουμε στη συνολική συνάρτηση Lagrange για ένα διπλό σύστημα αποτελούμενο από δύο αστέρες νετρονίων [Hansen et al., 2007]

$$L = \frac{GM_{tot}}{r} + \frac{1}{2}M_{tot}\left(\dot{r}^2 + r^2\dot{\phi}^2\right)^2 + \frac{GM_{tot}{}^2}{c^2}\left[-\frac{G(M_1+M_2)}{2r^2} + \frac{(3+2\nu)\dot{r}^2 + (3+\nu)r^2\dot{\phi}^2}{2r}\right]$$

$$+\frac{(1-3\nu)M_{tot}}{8c^2}\left(\dot{r}^4 + 2r^2\dot{r}^2\dot{\phi}^2 + r^4\dot{\phi}^4\right) + \frac{GMM'\kappa_n}{10r^3}\left[a_1^2(3\cos^2\alpha - 1) + a_2^2(3\sin^2\alpha - 1) - a_3^2\right]$$

$$+\frac{GMM'\kappa'_n}{10r^3}\left[a'_1{}^2(3\cos^2\alpha' - 1) + a'_2{}^2(3\sin^2\alpha' - 1) - a'_3{}^2\right]$$

$$+\frac{1}{2}I(\Lambda^2 + \Omega^2) - \frac{2}{5}\kappa_n M a_1 a_2 \Lambda\Omega + \frac{1}{10}\kappa_n M(\dot{a}_1^2 + \dot{a}_2^2 + \dot{a}_3^2) - k_1 K\rho_c^{1/n}M + \frac{3}{5-n}\frac{GM^2}{2R^3}$$

$$+\frac{1}{2}I'\left(\Lambda'^2 + \Omega'^2\right) - \frac{2}{5}\kappa'_n M' a'_1 a'_2 \Lambda'\Omega' + \frac{1}{10}\kappa'_n M'(\dot{a}'_1{}^2 + \dot{a}'_2{}^2 + \dot{a}'_3{}^2) - k'_1 K'\rho_c'^{\frac{1}{n'}}M' + \frac{3}{5-n'}\frac{GM'^2}{2R'^3} \tag{2.50}$$

ΔΙΑΦΟΡΙΚΕΣ ΕΞΙΣΩΣΕΙΣ ΓΙΑ ΣΥΣΤΗΜΑ ΔΥΟ ΑΣΤΕΡΩΝ ΝΕΤΡΟΝΙΩΝ (NS-NS)

Η συνάρτηση Lagrange της σχέσης (2.50) περιέχει τις ακόλουθες γενικευμένες συντεταγμένες $\{q_i\} = \{a_1, a_2, a_3, \gamma, \psi, a'_1, a'_2, a'_3, \gamma', \psi', r, \phi\}$ καθώς και τις γενικευμένες ταχύτητες $\{\dot{q}_i\} = \{\dot{a}_1, \dot{a}_2, \dot{a}_3, \Omega, \Lambda, \dot{a}'_1, \dot{a}'_2, \dot{a}'_3, \Omega', \Lambda', \dot{r}, \dot{\phi}\}$. Έτσι, από τις εξισώσεις Euler-Lagrange (εξ. 2.44) θα προκύψουν οι διαφορικές εξισώσεις που περιγράφουν την εξέλιξη του συστήματος. Μετά την εκτέλεση των πράξεων καταλήγουμε στις παρακάτω εξισώσεις

$$\ddot{a}_1 = a_1(\Omega^2 + \Lambda^2) - 2a_2\Omega\Lambda - \frac{2\pi}{q_n}a_1 A_1\bar{\rho} + \left(\frac{5k_1}{n\kappa_n}\frac{P_c}{\rho_c}\right)\frac{1}{a_1} + \frac{M'a_1}{r^3}(3\cos^2\alpha - 1) \tag{2.51}$$

$$\ddot{a}_2 = a_2(\Omega^2 + \Lambda^2) - 2a_1\Omega\Lambda - \frac{2\pi}{q_n}a_2 A_2\bar{\rho} + \left(\frac{5k_1}{n\kappa_n}\frac{P_c}{\rho_c}\right)\frac{1}{a_2} + \frac{M'a_2}{r^3}(3\sin^2\alpha - 1) \tag{2.52}$$

$$\ddot{a}_3 = -\frac{2\pi}{q_n}a_3 A_3\bar{\rho} + \left(\frac{5k_1}{n\kappa_n}\frac{P_c}{\rho_c}\right)\frac{1}{a_3} + \frac{M'a_3}{r^3} \tag{2.53}$$

$$\dot{\Omega} = \left(\frac{a_2}{a_1} - \frac{a_1}{a_2}\right)^{-1}\left[2\left(\frac{\Omega}{a_2} + \frac{\Lambda}{a_1}\right)\dot{a}_1 - 2\left(\frac{\Omega}{a_1} + \frac{\Lambda}{a_2}\right)\dot{a}_2 - \frac{3M'}{2r^3}\left(\frac{a_1}{a_2} + \frac{a_2}{a_1}\right)\sin 2\alpha\right] \tag{2.54}$$

$$\dot{\Lambda} = \left(\frac{a_2}{a_1} - \frac{a_1}{a_2}\right)^{-1}\left[2\left(\frac{\Omega}{a_1} + \frac{\Lambda}{a_2}\right)\dot{a}_1 - 2\left(\frac{\Omega}{a_2} + \frac{\Lambda}{a_1}\right)\dot{a}_2 - \frac{3M'}{2r^3}\sin 2\alpha\right] \tag{2.55}$$

Αντίστοιχα για τις μεταβλητές $a_1', a_2', a_3', \Omega'$ και Λ', οι εξισώσεις θα προκύψουν με εναλλαγή τονούμενων και μη τονούμενων ποσοτήτων στις (2.51) ως (2.55). Τέλος, για τις τροχιακές μεταβλητές ισχύει

$$\ddot{r} = r\dot{\phi}^2 - \frac{GM_{tot}}{r^2} - \frac{3GM_{tot}\kappa_n}{10r^4}[a_1^2(3\cos^2\alpha - 1) + a_2^2(3\sin^2\alpha - 1) - a_3^2]$$

$$-\frac{3GM_{tot}\kappa'_n}{10r^4}\left[{a_1'}^2(3\cos^2\alpha' - 1) + {a'}_2^2(3\sin^2\alpha' - 1) - {a'}_3^2\right] - \frac{GM_{tot}}{c^2}(3\nu + 1)\dot{\phi}^2$$

$$+\frac{GM_{tot}}{20r^2c^2}\Big\{(60 - 70\nu)\dot{r}^2 - \frac{3}{2}(3\nu - 1)\dot{\phi}^2[\kappa_n(a_1^2 + a_2^2 - 2a_3^2) + \kappa'_n\left({a'}_1^2 + {a'}_2^2 - 2{a'}_3^2\right)$$
$$+ 3\kappa_n(a_1^2 - a_2^2)\cos 2\alpha + 3\kappa'_n\left({a'}_1^2 - {a'}_2^2\right)\cos 2\alpha']\Big\}$$

$$-\frac{GM_{tot}}{10c^2r^3}\{-20GM_{tot}(\nu + 2) + 3(3\nu - 1)\dot{r}\dot{\phi}\left[\kappa_n(a_1^2 - a_2^2)\sin 2\alpha + \kappa'_n\left({a'}_1^2 - {a'}_2^2\right)\sin 2\alpha'\right]\}$$

$$-\frac{GM_{tot}}{c^2}\frac{9(3\nu - 1)\dot{r}^2}{40r^4}\left[\kappa_n(a_1^2 + a_2^2 - 2a_3^2) + \kappa'_n\left({a'}_1^2 + {a'}_2^2 - 2{a'}_3^2\right) + 3\kappa_n(a_1^2 - a_2^2)\cos 2\alpha\right.$$
$$\left. + 3\kappa'_n\left({a'}_1^2 - {a'}_2^2\right)\cos 2\alpha'\right]$$

$$+\frac{G^2M_{tot}^2}{c^2}\frac{3(3 + 2\nu)}{20r^5}\left[\kappa_n(a_1^2 + a_2^2 - 2a_3^2) + \kappa'_n\left({a'}_1^2 + {a'}_2^2 - 2{a'}_3^2\right) + 3\kappa_n(a_1^2 - a_2^2)\cos 2\alpha\right.$$
$$\left. + 3\kappa'_n\left({a'}_1^2 - {a'}_2^2\right)\cos 2\alpha'\right] \tag{2.56}$$

$$\ddot{\phi} = -\frac{2\dot{r}\dot{\phi}}{r} - \frac{3GM_{tot}\kappa_n}{10r^5}(a_1^2 - a_2^2)\sin 2\alpha - \frac{3GM_{tot}\kappa'_n}{10r^5}\left({a'}_1^2 - {a'}_2^2\right)\sin 2\alpha'$$

$$-\frac{GM_{tot}}{c^2}\frac{2(\nu - 2)\dot{r}\dot{\phi}}{r^2} - \frac{GM_{tot}}{c^2}\frac{9(3\nu - 1)\dot{\phi}^2}{20r^3}\left[\kappa_n(a_1^2 - a_2^2)\sin 2\alpha + \kappa'_n\left({a'}_1^2 - {a'}_2^2\right)\sin 2\alpha'\right]$$

$$-\frac{GM_{tot}}{c^2}\frac{3(3\nu - 1)\dot{r}\dot{\phi}}{20r^4}\left[\kappa_n(a_1^2 + a_2^2 - 2a_3^2) + \kappa'_n\left({a'}_1^2 + {a'}_2^2 - 2{a'}_3^2\right) + 3\kappa_n(a_1^2 - a_2^2)\cos 2\alpha\right.$$
$$\left. + 3\kappa'_n\left({a'}_1^2 - {a'}_2^2\right)\cos 2\alpha'\right]$$

$$-\frac{GM_{tot}}{c^2}\frac{3(3\nu - 1)\dot{r}^2}{20r^5}\left[\kappa_n(a_1^2 - a_2^2)\sin 2\alpha + \kappa'_n\left({a'}_1^2 - {a'}_2^2\right)\sin 2\alpha'\right]$$

$$+\frac{G^2\,M_{tot}^2}{c^2}\frac{3(3 + \nu)}{10r^6}\left[\kappa_n(a_1^2 - a_2^2)\sin 2\alpha + \kappa'_n\left({a'}_1^2 - {a'}_2^2\right)\sin 2\alpha'\right] \tag{2.57}$$

όπου $q_n \equiv \kappa_n(1 - n/5)$, P_c είναι η κεντρική πίεση, ρ_c η κεντρική πυκνότητα και $\bar{\rho} = 3M/4\pi R^3$ η μέση πυκνότητα του αστέρα.

ΔΙΑΦΟΡΙΚΕΣ ΕΞΙΣΩΣΕΙΣ ΓΙΑ ΣΥΣΤΗΜΑ ΜΕΛΑΝΗΣ ΟΠΗΣ - ΑΣΤΕΡΑ ΝΕΤΡΟΝΙΩΝ (BH-NS)

Από τις παραπάνω εξισώσεις (συνολικά δώδεκα) εύκολα προκύπτουν οι αντίστοιχες για διπλό σύστημα αστέρα νετρονίων - μελανής οπής, θέτοντας $\kappa'_n, a'_1, a'_2, a'_3 \to 0$ (οπότε ο αστέρας μάζας M' θα είναι μελανή οπή). Έτσι, θα καταλήξουμε σ' ένα σύστημα επτά διαφορικών εξισώσεων από τις οποίες οι πρώτες πέντε θα είναι οι (2.51) ως και (2.55), ενώ οι δύο τελευταίες θα έχουν ως εξής

$$\ddot{r} = r\dot{\phi}^2 - \frac{GM_{tot}}{r^2} - \frac{3GM_{tot}\kappa_n}{10r^4}[a_1^2(3\cos^2\alpha - 1) + a_2^2(3\sin^2\alpha - 1) - a_3^2] - \frac{GM_{tot}}{c^2}(3\nu + 1)\dot{\phi}^2$$
$$+ \frac{GM_{tot}}{20r^2c^2}\left\{(60 - 70\nu)\dot{r}^2 - \frac{3}{2}(3\nu - 1)\dot{\phi}^2[\kappa_n(a_1^2 + a_2^2 - 2a_3^2) + 3\kappa_n(a_1^2 - a_2^2)\cos 2\alpha]\right\}$$
$$- \frac{GM_{tot}}{10c^2r^3}\{-20GM_{tot}(\nu + 2) + 3(3\nu - 1)\dot{r}\dot{\phi}[\kappa_n(a_1^2 - a_2^2)\sin 2\alpha]\}$$
$$- \frac{GM_{tot}}{c^2}\frac{9(3\nu - 1)\dot{r}^2}{40r^4}[\kappa_n(a_1^2 + a_2^2 - 2a_3^2) + 3\kappa_n(a_1^2 - a_2^2)\cos 2\alpha]$$
$$+ \frac{G^2M_{tot}^2}{c^2}\frac{3(3 + 2\nu)}{20r^5}[\kappa_n(a_1^2 + a_2^2 - 2a_3^2) + 3\kappa_n(a_1^2 - a_2^2)\cos 2\alpha] \quad (2.58)$$

$$\ddot{\phi} = -\frac{2\dot{r}\dot{\phi}}{r} - \frac{3GM_{tot}\kappa_n}{10r^5}(a_1^2 - a_2^2) - \frac{GM_{tot}}{c^2}\frac{2(\nu - 2)\dot{r}\dot{\phi}}{r^2} - \frac{GM_{tot}}{c^2}\frac{9(3\nu - 1)\dot{\phi}^2}{20r^3}[\kappa_n(a_1^2 - a_2^2)\sin 2\alpha]$$
$$- \frac{GM_{tot}}{c^2}\frac{3(3\nu - 1)\dot{r}\dot{\phi}}{20r^4}[\kappa_n(a_1^2 + a_2^2 - 2a_3^2) + 3\kappa_n(a_1^2 - a_2^2)\cos 2\alpha]$$
$$- \frac{GM_{tot}}{c^2}\frac{3(3\nu - 1)\dot{r}^2}{20r^5}[\kappa_n(a_1^2 - a_2^2)\sin 2\alpha] + \frac{G^2\,M_{tot}^2}{c^2}\frac{3(3 + \nu)}{10r^6}[\kappa_n(a_1^2 - a_2^2)\sin 2\alpha] \quad (2.59)$$

ΔΙΑΦΟΡΙΚΕΣ ΕΞΙΣΩΣΕΙΣ ΓΙΑ ΣΥΣΤΗΜΑ ΔΥΟ ΜΕΛΑΝΩΝ ΟΠΩΝ (BH-BH)

Οι διαφορικές εξισώσεις για διπλό σύστημα δύο μελανών οπών, θα προκύψουν από τις αντίστοιχες ενός BH-NS μέσω της προσέγγισης $\kappa_n, a_1, a_2, a_3 \to 0$. Πρόκειται για τις ακόλουθες δύο εξισώσεις

$$\ddot{r} = r\dot{\phi}^2 - \frac{GM_{tot}}{r^2} - \frac{GM_{tot}}{c^2}(3\nu + 1)\dot{\phi}^2 + \frac{GM_{tot}(6 - 7\nu)\dot{r}^2}{2r^2c^2} + \frac{2G^2M_{tot}^2(\nu + 2)}{c^2r^3} \quad (2.60)$$

$$\ddot{\phi} = -\frac{2\dot{r}\dot{\phi}}{r} - \frac{GM_{tot}}{c^2}\frac{2(\nu - 2)\dot{r}\dot{\phi}}{r^2} \quad (2.61)$$

Ο ΟΡΟΣ ΤΗΣ ΠΙΕΣΗΣ

Στις παραπάνω εξισώσεις συναντάμε τον όρο $5k_1P_c/n\kappa_n\rho_c$, ο οποίος είναι βολικότερο (για την αριθμητική επίλυση των εξισώσεων) να αντικατασταθεί από κάποια έκφραση της μάζας, M, του αστέρα και της ακτίνας R_0, που αντιστοιχεί σε σφαιρικό πολύτροπο ίδιας μάζας μ' αυτή του αστέρα.

Γράφοντας την καταστατική εξίσωση του αστέρα στη μορφή $P_c/\rho_c = K\rho_c^{1/n} \propto R^{-3/n}$, ο παραπάνω όρος θα γραφεί ως εξής

$$\frac{5k_1P_c}{n\kappa_n\rho_c} = C_nR^{-3/n} \tag{2.62}$$

Όπου C_n είναι μια σταθερή αναλογίας που χρειάζεται να προσδιοριστεί. Θεωρώντας ένα μη περιστρεφόμενο σφαιρικό πολύτροπο σε ισορροπία οι εξισώσεις (2.51) ως και (2.53), με την προσθήκη της (2.62), γίνονται

$$-\frac{2\pi G}{q_n}R_0A_1\bar{\rho}_0 + C_nR_0^{-3/n}\frac{1}{R_0} = 0 \tag{2.63}$$

και εφόσον πρόκειται για σφαιρικό πολύτροπο (οπότε $A_1 = 2/3$), θα έχουμε $C_n = \frac{GM}{q_n}R_0^{3/(n-1)}$, οπότε ο όρος της πίεσης θα έχει ως εξής

$$\frac{5k_1P_c}{n\kappa_n\rho_c} = \frac{GM}{q_nR_0}\left(\frac{R_0}{R}\right)^{3/n} \tag{2.64}$$

Η σχέση (2.64) ισχύει για τη περίπτωση όπου $n \neq 0$.

2.4 ΕΚΠΟΜΠΗ ΒΑΡΥΤΙΚΗΣ ΑΚΤΙΝΟΒΟΛΙΑΣ ΑΠΟ ΔΙΠΛΑ ΣΥΣΤΗΜΑΤΑ

Σ' αντίθεση με τη Νευτώνεια προσέγγιση, στην οποία τα διπλά συστήματα είναι συντηρητικά, στη γενική θεωρία της σχετικότητας τέτοια συστήματα χάνουν ενέργεια εξαιτίας της εκπομπής βαρυτικής ακτινοβολίας. Η απώλεια αυτή μπορεί να προσεγγιστεί από ένα δυναμικό ανάδρασης (radiation reaction potential), Φ_{reac}, το οποίο δίνεται από τη σχέση

$$\Phi_{reac} = \frac{G}{5c^5}\mathfrak{I}^{(5)}_{\bar{a}\bar{b}}x_{\bar{a}}x_{\bar{b}} \tag{2.65}$$

όπου $\mathfrak{I}^{(5)}_{\bar{a}\bar{b}}$ είναι η πέμπτη χρονική παράγωγος του συμμετρικού, μηδενικού ίχνους τανυστή τετραπολικής μάζας (STF mass quadrupole tensor) στο περιστρεφόμενο σύστημα αναφοράς. Ο παραπάνω όρος θα επιφέρει μια τροποποίηση στη συνάρτηση Lagrange του συστήματος και θα οδηγήσει σε ένα νέο σύστημα διαφορικών εξισώσεων. Η διαδικασία αυτή είναι ισοδύναμη με το να παραμείνει αμετάβλητη η συνάρτηση Lagrange (εξ. 2.50) και να τροποποιηθούν κατάλληλα οι εξισώσεις Euler-Lagrange [Lai & Shapiro, 1995]. Έτσι, μπορούμε να γράψουμε

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) = \frac{\partial L}{\partial q_i} + \mathcal{F}_{q_i} \tag{2.66}$$

όπου $\mathcal{F}_{q_i}$ είναι οι γενικευμένες δυνάμεις που δίνονται από τη σχέση

$$\mathcal{F}_{q_i} = \frac{\partial \mathcal{W}}{\partial q_i} \tag{2.67}$$

Με $\mathcal{W}$ συμβολίζουμε τον ρυθμό απώλειας ενέργειας, ο οποίος με τη σειρά του είναι

$$\mathcal{W} = -\int \boldsymbol{v} \cdot \nabla \Phi_{reac} \rho dV \tag{2.68}$$

Η συνεισφορά του αστέρα μάζας M στον παραπάνω ρυθμό απώλειας, προκύπτει αν θεωρήσουμε ότι το πεδίο ταχυτήτων του αστέρα μπορεί να γραφεί ως $\boldsymbol{v} = \boldsymbol{u} + \boldsymbol{u}_{orb}$, όπου

$$\boldsymbol{u} = \left(\frac{a_1}{a_2}\Lambda - \Omega\right) x_2 \boldsymbol{e}_1 + \left(-\frac{a_2}{a_1}\Lambda + \Omega\right) x_1 \boldsymbol{e}_2 + \frac{\dot{a}_1}{a_1} x_1 \boldsymbol{e}_1 + \frac{\dot{a}_2}{a_2} x_2 \boldsymbol{e}_2 + \frac{\dot{a}_3}{a_3} x_3 \boldsymbol{e}_3 \tag{2.69}$$

η ταχύτητα ενός στοιχειώδους όγκου του ρευστού σε σχέση με το κέντρο του αστέρα και

$$\boldsymbol{u}_{orb} = -\dot{r}_M \boldsymbol{e}_{\bar{1}} - r_M \dot{\phi} \boldsymbol{e}_{\bar{2}} \tag{2.70}$$

η τροχιακή ταχύτητα του κέντρου μάζας του αστέρα. Τα μοναδιαία διανύσματα $\boldsymbol{e}_a$ είναι πακτωμένα στον αστέρα, ενώ τα $\boldsymbol{e}_{\bar{a}}$ καθορίζουν το περιστρεφόμενο σύστημα αναφοράς. Τα δύο αυτά συστήματα αναφοράς συνδέονται με τις σχέσεις

$$x_{\bar{1}} = x_1 \cos\alpha + x_2 \sin\alpha - r_M \ , \ \ x_{\bar{2}} = -x_1 \sin\alpha + x_2 \cos\alpha \quad \text{και} \quad x_{\bar{3}} = x_3 \tag{2.71}$$

Σύμφωνα με τα παραπάνω, η συνεισφορά του αστέρα μάζας M στο συνολικό ρυθμό απώλειας ενέργειας θα είναι

$$\begin{aligned}
\mathcal{W}_M = &-\frac{2G}{5c}\frac{\kappa_n M}{5}\left[a_1\dot{a}_1\left(\mathfrak{T}^{(5)}_{\bar{1}\bar{1}}\cos^2\alpha + \mathfrak{T}^{(5)}_{\bar{2}\bar{2}}\sin^2\alpha - \mathfrak{T}^{(5)}_{\bar{1}\bar{2}}\sin 2\alpha\right)\right. \\
&+a_2\dot{a}_2\left(\mathfrak{T}^{(5)}_{\bar{1}\bar{1}}\sin^2\alpha + \mathfrak{T}^{(5)}_{\bar{2}\bar{2}}\cos^2\alpha + \mathfrak{T}^{(5)}_{\bar{1}\bar{2}}\sin 2\alpha\right) \\
&\left.+\Omega(a_1^2 - a_2^2)\left(\mathfrak{T}^{(5)}_{\bar{1}\bar{2}}\cos 2\alpha + \frac{1}{2}\left(\mathfrak{T}^{(5)}_{\bar{1}\bar{1}} - \mathfrak{T}^{(5)}_{\bar{2}\bar{2}}\right)\sin 2\alpha\right) + a_3\dot{a}_3\mathfrak{T}^{(5)}_{\bar{3}\bar{3}}\right] \\
&-\frac{2GM}{5c^5}\left[r_M\dot{r}_M\mathfrak{T}^{(5)}_{\bar{1}\bar{1}} + r_M^2\dot{\phi}\mathfrak{T}^{(5)}_{\bar{1}\bar{2}}\right]
\end{aligned} \tag{2.72}$$

Η αντίστοιχη συνεισφορά του αστέρα μάζας M' θα προκύψει αντικαθιστώντας στη σχέση (2.72) τις τονούμενες με μη τονούμενες μεταβλητές και αντιστρόφως. Τέλος, προσθέτοντας τους δύο αυτούς όρους καταλήγουμε στην παρακάτω έκφραση για τη συνολική απώλεια ενέργειας

$$\begin{aligned}
\mathcal{W} = &-\frac{2G}{5c}\frac{\kappa_n M}{5}\left[a_1\dot{a}_1\left(\mathfrak{T}^{(5)}_{\bar{1}\bar{1}}\cos^2\alpha + \mathfrak{T}^{(5)}_{\bar{2}\bar{2}}\sin^2\alpha - \mathfrak{T}^{(5)}_{\bar{1}\bar{2}}\sin 2\alpha\right)\right. \\
&+a_2\dot{a}_2\left(\mathfrak{T}^{(5)}_{\bar{1}\bar{1}}\sin^2\alpha + \mathfrak{T}^{(5)}_{\bar{2}\bar{2}}\cos^2\alpha + \mathfrak{T}^{(5)}_{\bar{1}\bar{2}}\sin 2\alpha\right) \\
&\left.+\Omega(a_1^2 - a_2^2)\left(\mathfrak{T}^{(5)}_{\bar{1}\bar{2}}\cos 2\alpha + \frac{1}{2}\left(\mathfrak{T}^{(5)}_{\bar{1}\bar{1}} - \mathfrak{T}^{(5)}_{\bar{2}\bar{2}}\right)\sin 2\alpha\right) + a_3\dot{a}_3\mathfrak{T}^{(5)}_{\bar{3}\bar{3}}\right] \\
&-\frac{2G}{5c}\frac{\kappa'_n M'}{5}\left[a'_1\dot{a}'_1\left(\mathfrak{T}^{(5)}_{\bar{1}\bar{1}}\cos^2\alpha' + \mathfrak{T}^{(5)}_{\bar{2}\bar{2}}\sin^2\alpha' - \mathfrak{T}^{(5)}_{\bar{1}\bar{2}}\sin 2\alpha'\right)\right. \\
&+a'_2\dot{a}'_2\left(\mathfrak{T}^{(5)}_{\bar{1}\bar{1}}\sin^2\alpha' + \mathfrak{T}^{(5)}_{\bar{2}\bar{2}}\cos^2\alpha' + \mathfrak{T}^{(5)}_{\bar{1}\bar{2}}\sin 2\alpha'\right) \\
&\left.+\Omega\left(a'^2_1 - a'^2_2\right)\left(\mathfrak{T}^{(5)}_{\bar{1}\bar{2}}\cos 2\alpha' + \frac{1}{2}\left(\mathfrak{T}^{(5)}_{\bar{1}\bar{1}} - \mathfrak{T}^{(5)}_{\bar{2}\bar{2}}\right)\sin 2\alpha'\right) + a'_3\dot{a}'_3\mathfrak{T}^{(5)}_{\bar{3}\bar{3}}\right] \\
&-\frac{2GM}{5c^5}\left[r\dot{r}\mathfrak{T}^{(5)}_{\bar{1}\bar{1}} + r^2\dot{\phi}\mathfrak{T}^{(5)}_{\bar{1}\bar{2}}\right]
\end{aligned} \tag{2.73}$$

Με αντικατάσταση της παραπάνω σχέσης στη (2.67) και μετά από πράξεις, θα καταλήξουμε στις παρακάτω εκφράσεις για τις γενικευμένες δυνάμεις

$$\mathcal{F}_{a_1} = -\frac{2G}{5c^5}\frac{\kappa_n M}{5}\left[\mathfrak{I}^{(5)}_{\bar{1}\bar{1}}\cos^2\alpha + \mathfrak{I}^{(5)}_{\bar{2}\bar{2}}\sin^2\alpha - \mathfrak{I}^{(5)}_{\bar{1}\bar{2}}\sin 2\alpha\right]a_1 \tag{2.74}$$

$$\mathcal{F}_{a_2} = -\frac{2G}{5c^5}\frac{\kappa_n M}{5}\left[\mathfrak{I}^{(5)}_{\bar{1}\bar{1}}\sin^2\alpha + \mathfrak{I}^{(5)}_{\bar{2}\bar{2}}\cos^2\alpha + \mathfrak{I}^{(5)}_{\bar{1}\bar{2}}\sin 2\alpha\right]a_2 \tag{2.75}$$

$$\mathcal{F}_{a_3} = -\frac{2G}{5c^5}\frac{\kappa_n M}{5}\mathfrak{I}^{(5)}_{33}a_3 \tag{2.76}$$

$$\mathcal{F}_{\gamma} = -\frac{2G}{5c^5}\frac{\kappa_n M}{5}(a_1^2 - a_2^2)\left[\mathfrak{I}^{(5)}_{\bar{1}\bar{2}}\cos 2\alpha + \frac{1}{2}\left(\mathfrak{I}^{(5)}_{\bar{1}\bar{1}} - \mathfrak{I}^{(5)}_{\bar{2}\bar{2}}\right)\sin 2\alpha\right] \tag{2.77}$$

$$\mathcal{F}_{\psi} = 0 \tag{2.78}$$

$$\mathcal{F}_{r} = -\frac{2G\mu}{5c^5}\mathfrak{I}^{(5)}_{\bar{1}\bar{1}}r \tag{2.79}$$

$$\mathcal{F}_{\phi} = -\frac{2G\mu}{5c^5}\mathfrak{I}^{(5)}_{\bar{1}\bar{2}}r^2 \tag{2.80}$$

Σύμφωνα με τις τροποποιημένες εξισώσεις Euler-Lagrange, εύκολα μπορούν να υπολογιστούν οι εξισώσεις εξέλιξης. Γενικά οι τροποποιήσεις στις εξισώσεις θα έχουν ως εξής

$$\ddot{a}_1 = [\dots] - \frac{2G}{5c^5}\left[\mathfrak{I}^{(5)}_{\bar{1}\bar{1}}\cos^2\alpha + \mathfrak{I}^{(5)}_{\bar{2}\bar{2}}\sin^2\alpha - \mathfrak{I}^{(5)}_{\bar{1}\bar{2}}\sin 2\alpha\right]a_1 \tag{2.81}$$

$$\ddot{a}_2 = [\dots] - \frac{2G}{5c^5}\left[\mathfrak{I}^{(5)}_{\bar{1}\bar{1}}\sin^2\alpha + \mathfrak{I}^{(5)}_{\bar{2}\bar{2}}\cos^2\alpha + \mathfrak{I}^{(5)}_{\bar{1}\bar{2}}\sin 2\alpha\right]a_2 \tag{2.82}$$

$$\ddot{a}_3 = [\dots] - \frac{2G}{5c^5}\mathfrak{I}^{(5)}_{33}a_3 \tag{2.83}$$

$$\dot{\Omega} = \left(\frac{a_2}{a_1} - \frac{a_1}{a_2}\right)^{-1}\left\{[\dots] + \frac{2G}{5c^5}\left[\mathfrak{I}^{(5)}_{\bar{1}\bar{2}}\cos 2\alpha + \frac{1}{2}\left(\mathfrak{I}^{(5)}_{\bar{1}\bar{1}} - \mathfrak{I}^{(5)}_{\bar{2}\bar{2}}\right)\sin 2\alpha\right]\left(\frac{a_1}{a_2} + \frac{a_2}{a_1}\right)\right\} \tag{2.84}$$

$$\dot{\Lambda} = \left(\frac{a_2}{a_1} - \frac{a_1}{a_2}\right)^{-1}\left\{[\dots] + \frac{4G}{5c^5}\left[\mathfrak{I}^{(5)}_{\bar{1}\bar{2}}\cos 2\alpha + \frac{1}{2}\left(\mathfrak{I}^{(5)}_{\bar{1}\bar{1}} - \mathfrak{I}^{(5)}_{\bar{2}\bar{2}}\right)\sin 2\alpha\right]\right\} \tag{2.85}$$

$$\ddot{r} = [\dots] - \frac{2G}{5c^5}\mathfrak{I}^{(5)}_{\bar{1}\bar{1}}r \tag{2.86}$$

$$\ddot{\phi} = [\dots] - \frac{2G}{5c^5}\mathfrak{I}^{(5)}_{\bar{1}\bar{2}} \tag{2.87}$$

όπου με $[\dots]$ συμβολίζουμε το κομμάτι των εξισώσεων[2] που υπολογίστηκε στην ενότητα 2.3.

Οι πέμπτες χρονικές παράγωγοι του $\mathfrak{I}_{ij}$ συνίστανται από την τροχιακή συνεισφορά, $\mathfrak{I}^{(5),orb}_{ij}$ και από τη συνεισφορά των αστρικών βαθμών ελευθερίας, $\mathfrak{I}^{(5),star}_{ij}$. Το άθροισμα των δύο αυτών όρων συνιστά τις $\mathfrak{I}^{(5)}_{ij}$ [Hansen, 2006]. Παρακάτω υπολογίζουμε τους επιμέρους όρους των παραγώγων.

[2] Εκτός του όρου $\left(\frac{a_2}{a_1} - \frac{a_1}{a_2}\right)^{-1}$ για τις εξισώσεις των Ω και Λ.

ΣΥΝΕΙΣΦΟΡΑ ΤΩΝ ΑΣΤΡΙΚΩΝ ΒΑΘΜΩΝ ΕΛΕΥΘΕΡΙΑΣ ΣΤΙΣ $\mathfrak{T}_{ij}^{(5)}$

Η σχέση που δίνει τις πέμπτες χρονικές παραγώγους του τανυστή τετραπολικής μάζας είναι η ακόλουθη

$$\mathfrak{T}_{\bar{a}\bar{b}}^{(5)} = T_{\bar{a}\alpha}(\phi)T_{\bar{b}\beta}(\phi)\frac{d^5}{dt^5}\left[T_{\alpha i}^{\dagger}(\gamma)T_{\beta j}^{\dagger}(\gamma)\mathfrak{T}_{ij}^{(BF.)}\right]$$

$$= \sum_{m=0}^{5}\binom{5}{m}\left[\frac{d^{5-m}}{dt^{5-m}}\mathfrak{T}_{ij}^{(BF)}\right]\sum_{p=0}^{m}\binom{m}{p}\left[T_{\bar{a}\alpha}(\phi)\frac{d^{m-p}}{dt^{m-p}}T_{\alpha i}^{\dagger}(\gamma)\right]\left[T_{\bar{b}\beta}(\phi)\frac{d^{p}}{dt^{p}}T_{\beta j}^{\dagger}(\gamma)\right]$$

$$= \sum_{m=0}^{5}\sum_{p=0}^{m}\binom{5}{m}\binom{m}{p}\left[\frac{d^{5-m}}{dt^{5-m}}\mathfrak{T}_{ij}^{(BF)}\right]R_{\bar{a}i}^{m-p}R_{\bar{b}j}^{p} \tag{2.88}$$

όπου ο δείκτης (BF) δηλώνει το πακτωμένο σύστημα αναφοράς. Επιπλέον,

$$R_{\bar{a}i}^{p} = T_{\bar{a}\alpha}(\phi)\frac{d^p}{dt^p}T_{\alpha i}^{\dagger}(\gamma) \tag{2.89}$$

και $T_{ab}(\phi)$ είναι ο πίνακας μετασχηματισμού

$$T_{ab}(\phi) = \begin{pmatrix} \cos\phi & \sin\phi & 0 \\ -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{2.90}$$

Η σχέση (2.66) οδηγεί σε αρκετά πολύπλοκες εκφράσεις για τον τανυστή $\mathfrak{T}_{\bar{a}\bar{b}}^{(5)}$. Ωστόσο, μπορεί να απλοποιηθεί σημαντικά αν εφαρμόσουμε την ημιστατική προσέγγιση (qausi-static approximation). Σύμφωνα μ' αυτή, όλες οι εσωτερικές ταχύτητες και επιταχύνσεις μπορούν να θεωρηθούν μικρές και επομένως μπορούμε να αγνοήσουμε όλους τους όρους τάξης $\mathcal{O}(\ddot{a}_i)$ και $\mathcal{O}(\ddot{\Omega})$ και να κρατήσουμε μόνο τους γραμμικούς όρους ως προς $\dot{a}_i$ και $\dot{\Omega}$. Θεωρώντας ακόμη ότι $|\dot{a}_i| \ll |\Omega a_i|$ η σχέση (2.66) θα απλοποιηθεί στην παρακάτω έκφραση

$$\mathfrak{T}_{\bar{a}\bar{b}}^{(5)} = \mathfrak{T}_{ij}\sum_{p=0}^{5}\binom{5}{p}R_{\bar{a}i}^{p}R_{\bar{b}j}^{5-p} + \dot{\mathfrak{T}}_{ij}\sum_{p=0}^{4}\binom{4}{p}R_{\bar{a}i}^{p}R_{\bar{b}j}^{4-p} \tag{2.91}$$

και με αλγεβρικές πράξεις θα καταλήξουμε στην παρακάτω έκφραση

$$\mathfrak{T}_{\bar{a}\bar{b}}^{(5),star} = (\mathfrak{T}_{11} - \mathfrak{T}_{22})\left[16\Omega^5\begin{pmatrix} \sin 2\alpha & \cos 2\alpha & 0 \\ \cos 2\alpha & -\sin 2\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} - 80\Omega^3\dot{\Omega}\begin{pmatrix} -\cos 2\alpha & \sin 2\alpha & 0 \\ \sin 2\alpha & \cos 2\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix}\right]$$

$$+40\Omega^4(\dot{\mathfrak{T}}_{11} - \dot{\mathfrak{T}}_{22})\begin{pmatrix} \cos 2\alpha & -\sin 2\alpha & 0 \\ -\sin 2\alpha & -\cos 2\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{2.92}$$

Η ΤΡΟΧΙΑΚΗ ΣΥΝΕΙΣΦΟΡΑ ΣΤΙΣ $\mathfrak{T}_{ij}^{(5)}$

Η ημιστατική προσέγγιση που εφαρμόσαμε στους εσωτερικούς βαθμούς ελευθερίας δε μπορεί να εφαρμοστεί και στις μεταβλητές της τροχιακής κίνησης. Για να υπολογίσουμε την τροχιακή

συνεισφορά στις $\mathfrak{T}_{\bar{\imath}\bar{\jmath}}^{(5)}$, υπολογίζουμε τις $\mathfrak{T}_{ij}^{(5)}$ στο αδρανειακό σύστημα αναφοράς, εισάγοντας τις Νευτώνειες εξισώσεις κίνησης

$$\ddot{r} = -\frac{GM}{r^2} + r\dot{\phi}^2 \ , \ \ddot{\phi} = -\frac{2\dot{r}\dot{\phi}}{r} \tag{2.93}$$

Μια πιο ακριβής προσέγγιση προκύπτει αν αντί των Νευτώνειων εξισώσεων αντικαταστήσουμε τις αντίστοιχες που εμπεριέχουν την μετανευτώνεια διόρθωση ή/και τους αστρικούς βαθμούς ελευθερίας. Ωστόσο, όλοι οι επιπλέον αυτοί όροι είναι μεγαλύτερης τάξης ως προς $1/r$ και επομένως δεν αναμένεται να τροποποιήσουν ιδιαίτερα το αποτέλεσμα.

Στο αδρανειακό σύστημα αναφοράς οι μη μηδενικές συνιστώσες του $\mathfrak{T}_{ij}^{(orb)}$ είναι οι εξής

$$\mathfrak{T}_{11}^{(orb)} = \frac{\mu r^2}{6}(1 + 3\cos 2\phi) \tag{2.94}$$

$$\mathfrak{T}_{22}^{(orb)} = \frac{\mu r^2}{6}(1 - 3\cos 2\phi) \tag{2.95}$$

$$\mathfrak{T}_{33}^{(orb)} = -\frac{\mu r^2}{3} \tag{2.96}$$

$$\mathfrak{T}_{12}^{(orb)} = \frac{\mu r^2}{2}\sin 2\phi \tag{2.97}$$

Εισάγοντας στις εξισώσεις (2.94) – (2.97) τις (2.93), καταλήγουμε στις εκφράσεις του $\mathfrak{T}_{\bar{\imath}\bar{\jmath}}^{(5)}$, εφόσον χρησιμοποιήσαμε το μετασχηματισμό

$$\mathfrak{T}_{\bar{a}\bar{b}}^{(5,orb)} = T_{\bar{a}\alpha}(\phi)T_{\bar{b}\beta}(\phi)\mathfrak{T}_{\alpha\beta}^{(5,orb)} \tag{2.98}$$

για να μεταβούμε από το αδρανειακό, στο περιστρεφόμενο σύστημα αναφοράς. Οι εκφράσεις της τροχιακής συνεισφοράς στις $\mathfrak{T}_{\bar{\imath}\bar{\jmath}}^{(5)}$ είναι οι ακόλουθες

$$\mathfrak{T}_{\bar{1}\bar{1}}^{(5,orb)} = -\frac{8GM_{tot}\mu}{3}\frac{\dot{r}}{r^4}\left[\frac{4GM_{tot}}{r} + 3\dot{r}^2 + 18r^2\dot{\phi}^2\right] \tag{2.99}$$

$$\mathfrak{T}_{\bar{2}\bar{2}}^{(5,orb)} = \frac{2GM_{tot}\mu}{3}\frac{\dot{r}}{r^4}\left[\frac{8GM_{tot}}{r} + 6\dot{r}^2 + 81r^2\dot{\phi}^2\right] \tag{2.100}$$

$$\mathfrak{T}_{\bar{3}\bar{3}}^{(5,orb)} = \frac{2GM_{tot}\mu}{3}\frac{\dot{r}}{r^4}\left[\frac{8GM_{tot}}{r} + 6\dot{r}^2 - 9r^2\dot{\phi}^2\right] \tag{2.101}$$

$$\mathfrak{T}_{\bar{1}\bar{2}}^{(5,orb)} = -\frac{4GM_{tot}\mu}{r^3}\dot{\phi}\left[\frac{2GM_{tot}}{r} + 9\dot{r}^2 - 9r^2\dot{\phi}^2\right] \tag{2.102}$$

Έτσι, η συνολική έκφραση για τις $\mathfrak{T}_{ij}^{(5)}$ θα είναι το άθροισμα

$$\mathfrak{T}_{ij}^{(5)} = \mathfrak{T}_{ij}^{(5),star} + \mathfrak{T}_{ij}^{(5),orb} \tag{2.103}$$

όπου οι επιμέρους όροι θα δίνονται από τις σχέσεις (2.92) και (2.99) – (2.102) αντίστοιχα.

2.5 ΚΥΜΑΤΟΜΟΡΦΕΣ ΒΑΡΥΤΙΚΗΣ ΑΚΤΙΝΟΒΟΛΙΑΣ

Η βαρυτική ακτινοβολία που εκπέμπεται από ένα απομονωμένο σύστημα μπορεί να γραφεί σαν άθροισμα πολυπολικών όρων, όπως φαίνεται στην παρακάτω σχέση

$$h_{jk}^{TT} = \frac{G}{Dc^4} P_{jkim}(\boldsymbol{n}) \sum_{l=2}^{\infty} \left[\left(\frac{1}{c}\right)^{l-2} \mathfrak{I}_{imA_{l-2}}^{(l)} \left(t - \frac{D}{c}\right) N_{A_l} + \left(\frac{1}{c}\right)^{l-1} \frac{8l}{(l+1)!} \varepsilon_{pq} \left(i\mathcal{J}_m^{(l)}\right)_{pA_{l-2}} (t - D/c) n_q N_{A_l} \right] \tag{2.104}$$

όπου $\mathfrak{I}_{A_l}$ και $\mathcal{J}_{A_l}$ είναι συμμετρικοί, μηδενικού ίχνος τανυστές της πολυπολικής μάζας και της ροπής αντίστοιχα σε μια Καρτεσιανή βάση, ενώ A_l είναι πολλαπλός δείκτης της μορφής $A_l = a_1 a_2 \dots a_l$, με $a_i = 1,2,3, n_a$ να είναι το διάνυσμα κατά τη διεύθυνση του παρατηρητή και $N_{A_l} = n_{a_1} n_{a_2} \dots n_{a_l}$. Τέλος, με D συμβολίζουμε την απόσταση από τον παρατηρητή, ενώ ο άνω δείκτης εντός παρενθέσεως είναι η παράγωγος αντίστοιχης τάξης. Η ποσότητα $P_{ijkm}(\boldsymbol{n})$ είναι ο τελεστής προβολής κατά τη διεύθυνση $\boldsymbol{D}$ και δίνεται από τη σχέση

$$P_{ijkm}(\boldsymbol{n}) = (\delta_{ik} - n_i n_k)(\delta_{jm} - n_j n_m) - \frac{1}{2}(\delta_{ij} - n_i n_j)(\delta_{km} - n_k n_m) \tag{2.105}$$

Για υπολογισμούς είναι βολικότερο να ορίσουμε τις αντίστοιχες ποσότητες των $\mathfrak{I}_{A_l}$ και $\mathcal{J}_{A_l}$ ως προς τον άξονα της στροφορμής. Οι ποσότητες αυτές συνδέονται με τις αντίστοιχες στο Καρτεσιανό σύστημα αναφοράς με τις σχέσεις

$$I^{lm}(t) = \frac{16\pi}{(2l+1)!!} \sqrt{\frac{(l+1)(l+2)}{2l(l-1)}} \mathfrak{I}_{A_l} Y_{\Lambda_l}^{lm*} \tag{2.106}$$

$$S^{lm}(t) = -\frac{32\pi l}{(l+1)(2l+1)!!} \sqrt{\frac{(l+1)(l+2)}{2l(l-1)}} \mathcal{J}_{A_l} Y_{\Lambda_l}^{lm*} \tag{2.107}$$

όπου για $m \geq 0$

$$Y_{A_l}^{lm} = (-1)^m (2l-1)!! \sqrt{\frac{2l+1}{4\pi(l-m)!\,(l+m)!}} \left(\delta_{\langle i_1}^1 + i\delta_{\langle i_1}^2\right) \dots \left(\delta_{\langle i_m}^1 + i\delta_{\langle i_m}^2\right) \delta_{i_{m+1}}^1 \dots \delta_{l\rangle}^3 \tag{2.108}$$

κι επιπλέον

$$I^{l-m} = (-1)^m I^{lm*} \tag{2.109}$$

Με βάση τα παραπάνω, η εκπομπή βαρυτικής ακτινοβολίας μπορεί να εκφραστεί από τη σχέση [Thorne, 1980, Junker & Schaefer, 1992]

$$h_{jk}^{TT} = \frac{G}{Dc^4} \sum_{l=2}^{\infty} \sum_{m=-l}^{l} \left[\left(\frac{1}{c}\right)^{l-2} (I^{lm})^{(l)} (t - D/c) T_{jk}^{E2,lm}(\Theta, \Phi) + \left(\frac{1}{c}\right)^{l-1} (S^{lm})^{(l)} (t - D/c) T_{jk}^{B2,lm}(\Theta, \Phi) \right] \tag{2.110}$$

όπου $T_{jk}^{E2,lm}$ και $T_{jk}^{B2,lm}$ οι σφαιρικές αρμονικές ηλεκτρικού και μαγνητικού τύπου αντίστοιχα του τανυστή του σπιν (pure-spin tensor).

Ο ΚΥΡΙΟΣ ΟΡΟΣ ΤΗΣ ΒΑΡΥΤΙΚΗΣ ΑΚΤΙΝΟΒΟΛΙΑΣ

Στην προσέγγιση που θα χρησιμοποιήσουμε για την προσομοίωση της βαρυτικής ακτινοβολίας από διπλά συστήματα, μόνο οι όροι για $l = 2$ λαμβάνονται υπόψη (προσέγγιση κύριου όρου). Στην προσέγγιση αυτή η εξίσωση (2.110) παίρνει την ακόλουθη μορφή

$$h_{jk}^{TT} = \frac{G}{Dc^4} \sum_{m=-2}^{2} \ddot{I}^{2m} T_{jk}^{E2,2m}(\Theta, \Phi) \tag{2.111}$$

Οι όροι $T_{jk}^{E2,2m}$ μπορούν να εκφραστούν ως εξής [Junker & Schaefer, 1992]

$$T^{E2,20} = \sqrt{\frac{15}{64\pi}} \sin^2\Theta \left(\hat{\boldsymbol{\Theta}}\otimes\hat{\boldsymbol{\Theta}} - \hat{\boldsymbol{\Phi}}\otimes\hat{\boldsymbol{\Phi}}\right) \tag{2.112}$$

$$T^{E2,2\pm2} = \sqrt{\frac{5}{128\pi}} e^{\pm 2i\Phi}\left[1 + \cos^2\Theta\left(\hat{\boldsymbol{\Theta}}\otimes\hat{\boldsymbol{\Theta}} - \hat{\boldsymbol{\Phi}}\otimes\hat{\boldsymbol{\Phi}}\right) \pm 2i\cos\Theta\left(\hat{\boldsymbol{\Theta}}\otimes\hat{\boldsymbol{\Phi}} - \hat{\boldsymbol{\Phi}}\otimes\hat{\boldsymbol{\Theta}}\right)\right] \tag{2.113}$$

Παρατηρούμε ότι οι δύο καταστάσεις πόλωσης της βαρυτικής ακτινοβολίας, h_+ και $h_\times$, είναι ανάλογες με $\left(\hat{\boldsymbol{\Theta}}\otimes\hat{\boldsymbol{\Theta}} - \hat{\boldsymbol{\Phi}}\otimes\hat{\boldsymbol{\Phi}}\right)$ και $\left(\hat{\boldsymbol{\Theta}}\otimes\hat{\boldsymbol{\Phi}} - \hat{\boldsymbol{\Phi}}\otimes\hat{\boldsymbol{\Theta}}\right)$ αντίστοιχα.

Με βάση τη σχέση (2.106) μπορούμε να συσχετίσουμε τις ποσότητες $\ddot{I}^{2m}$ με τις $\ddot{\mathfrak{T}}^{2m}$. Έτσι, προκύπτει

$$\ddot{I}^{2m} = \sqrt{3}\frac{16\pi}{15} \ddot{\mathfrak{T}}_{i_1 i_2} Y_{i_1 i_2}^{2m*} \tag{2.114}$$

ή αναλυτικά

$$\ddot{I}^{20} = 4\sqrt{\frac{3\pi}{5}} \ddot{\mathfrak{T}}_{33} \tag{2.115}$$

$$\ddot{I}^{21} = -\sqrt{\frac{32\pi}{5}}\left(\ddot{\mathfrak{T}}_{13} - i\ddot{\mathfrak{T}}_{23}\right) \tag{2.116}$$

$$\ddot{I}^{22} = \sqrt{\frac{8\pi}{5}}\left(\ddot{\mathfrak{T}}_{11} - \ddot{\mathfrak{T}}_{22} - 2i\ddot{\mathfrak{T}}_{12}\right) \tag{2.117}$$

Εφόσον η μόνη μη μηδενική συνιστώσα του τανυστή $\mathfrak{T}_{ij}$ είναι η $\mathfrak{T}_{12}$, από την (2.116) συμπεραίνουμε ότι $I^{21} = 0$, ενώ η I^{2-2} είναι ο μιγαδικός συζυγής της I^{22}. Με βάση τα παραπάνω μπορούμε να γράψουμε την (2.111) στην ακόλουθη μορφή

$$h_+ = \frac{G}{Dc^4}\left[\ddot{I}^{2-2}T_+^{E2,2-2} + \ddot{I}^{20}T_+^{E2,20} + \ddot{I}^{22}T_+^{E2,22}\right] \tag{2.118}$$

$$h_\times = \frac{G}{Dc^4}\left[\ddot{I}^{22}T_\times^{E2,22} + \left(\ddot{I}^{22}T_+^{E2,22}\right)^*\right] \tag{2.119}$$

όπου

$$T_+^{E2,22} = \sqrt{\frac{5}{128\pi}}e^{2i\Phi}(1+\cos^2\Theta) \tag{2.120}$$

$$T_\times^{E2,22} = \sqrt{\frac{5}{128\pi}}e^{2i\Phi}(2i\cos\Theta) \tag{2.121}$$

$$T_+^{E2,20} = \sqrt{\frac{15}{64\pi}}\sin^2\Theta \tag{2.122}$$

$$T_\times^{E2,20} = 0 \tag{2.123}$$

Εισάγοντας τις εκφράσεις των $\ddot{I}^{2m}$ και $T_{jk}^{E2,2m}$ στις (2.118) και (2.119) καταλήγουμε στις παρακάτω σχέσεις για τις δύο πολώσεις της βαρυτικής ακτινοβολίας

$$h_+ = \frac{G}{2Dc^4}\left\{(1+\cos^2\Theta)\left[\left(\ddot{\mathfrak{I}}_{11}-\ddot{\mathfrak{I}}_{22}\right)\cos 2\Phi + 2\ddot{\mathfrak{I}}_{12}\sin 2\Phi\right] + 3\ddot{\mathfrak{I}}_{33}\sin^2\Theta\right\} \tag{2.124}$$

$$h_\times = \frac{2G}{Dc^4}\cos\Theta\,\ddot{\mathfrak{I}}_{12} \tag{2.125}$$

Οι δεύτερες χρονικές παράγωγοι του τανυστή $\mathfrak{I}_{ij}$ θα υπολογιστούν παρόμοια με τις $\mathfrak{I}_{ij}^{(5)}$ όπως είδαμε στην ενότητα 2.4. Συγκεκριμένα, θα έχουμε την αστρική $\left(\ddot{\mathfrak{I}}_{ij}^{st}\right)$ και την τροχιακή $\left(\ddot{\mathfrak{I}}_{ij}^{orb}\right)$ συνεισφορά [Hansen, 2006], το άθροισμα των οποίων, σε Νευτώνεια προσέγγιση, συνιστά τις χρονικές παραγώγους των σχέσεων (2.124) και (2.125).

ΣΥΝΕΙΣΦΟΡΑ ΤΩΝ ΑΣΤΡΙΚΩΝ ΒΑΘΜΩΝ ΕΛΕΥΘΕΡΙΑΣ ΣΤΙΣ $\ddot{\mathfrak{I}}_{ij}$

Οι συνιστώσες του τανυστή $\mathfrak{I}_{ij}$ στις εκφράσεις των δύο πολώσεων της βαρυτικής ακτινοβολίας, αναφέρονται στο αδρανειακό σύστημα αναφοράς (Inertial System, IS). Ωστόσο, η μορφή του $\mathfrak{I}_{ij}$ είναι εξαιρετικά απλή στο πακτωμένο στον αστέρα σύστημα αναφοράς (Body-Fixed System, BF). Γενικά, ο τανυστής $\mathfrak{I}_{ij}^{(IS)}$ στο αδρανειακό σύστημα αναφοράς σχετίζεται με τον αντίστοιχο στο πακτωμένο σύστημα αναφοράς, $\mathfrak{I}_{ij}^{(BF)}$, μέσω του μετασχηματισμού

$$\mathfrak{I}_{ij}^{(IS)} = T_{i\alpha}^{\dagger}(\gamma)T_{j\beta}^{\dagger}(\gamma)\mathfrak{I}_{ij}^{(BF)} \tag{2.126}$$

όπου ο πίνακας του μετασχηματισμού, $T_{\alpha\beta}(\phi)$, δίνεται από τη σχέση (2.90). Αναλυτικά, οι παραπάνω σχέσεις θα έχουν ως εξής:

$$\mathfrak{T}_{11}^{(IS)} = \frac{1}{2}\left(\mathfrak{T}_{11}^{(BF)} - \mathfrak{T}_{22}^{(BF)}\right)\cos 2\gamma + \frac{1}{6}\left(\mathfrak{T}_{11}^{(BF)} + \mathfrak{T}_{22}^{(BF)} - 2\mathfrak{T}_{33}^{(BF)}\right) \quad (2.127)$$

$$\mathfrak{T}_{22}^{(IS)} = -\frac{1}{2}\left(\mathfrak{T}_{11}^{(BF)} - \mathfrak{T}_{22}^{(BF)}\right)\cos 2\gamma + \frac{1}{6}\left(\mathfrak{T}_{11}^{(BF)} + \mathfrak{T}_{22}^{(BF)} - 2\mathfrak{T}_{33}^{(BF)}\right) \quad (2.128)$$

$$\mathfrak{T}_{33}^{(IS)} = -\frac{1}{6}\left(\mathfrak{T}_{11}^{(BF)} + \mathfrak{T}_{22}^{(BF)} - 2\mathfrak{T}_{33}^{(BF)}\right) \quad (2.129)$$

$$\mathfrak{T}_{11}^{(IS)} = -\frac{1}{2}\left(\mathfrak{T}_{11}^{(BF)} - \mathfrak{T}_{22}^{(BF)}\right)\sin 2\gamma \quad (2.130)$$

Παραγωγίζοντας τις παραπάνω σχέσεις και ακολουθώντας την ημιστατική προσέγγιση, όπως και στην ενότητα 2.4 (οπότε $\ddot{\mathfrak{T}}_{ij}^{(BF)} \approx 0$), θα έχουμε τις παρακάτω σχέσεις

$$\ddot{\mathfrak{T}}_{11}^{(IS)} = -2\left(\mathfrak{T}_{11}^{(BF)} - \mathfrak{T}_{22}^{(BF)}\right)\Omega^2\cos 2\gamma - \left[2\left(\dot{\mathfrak{T}}_{11}^{(BF)} - \dot{\mathfrak{T}}_{22}^{(BF)}\right)\Omega + \left(\mathfrak{T}_{11}^{(BF)} - \mathfrak{T}_{22}^{(BF)}\right)\dot{\Omega}\right]\sin 2\gamma \quad (2.131)$$

$$\ddot{\mathfrak{T}}_{12}^{(IS)} = -2\left(\mathfrak{T}_{11}^{(BF)} - \mathfrak{T}_{22}^{(BF)}\right)\Omega^2\sin 2\gamma + \left[2\left(\dot{\mathfrak{T}}_{11}^{(BF)} - \dot{\mathfrak{T}}_{22}^{(BF)}\right)\Omega + \left(\mathfrak{T}_{11}^{(BF)} - \mathfrak{T}_{22}^{(BF)}\right)\dot{\Omega}\right]\cos 2\gamma \quad (2.132)$$

$$\ddot{\mathfrak{T}}_{22}^{(IS)} = -\ddot{\mathfrak{T}}_{11}^{(IS)} \quad (2.133)$$

$$\ddot{\mathfrak{T}}_{33}^{(IS)} = 0 \quad (2.134)$$

Οι σχέσεις αυτές αποτελούν την αστρική συνεισφορά στις δεύτερες χρονικές παραγώγους του $\mathfrak{T}_{ij}$. Η έκφραση του $\mathfrak{T}_{ij}$ στο πακτωμένο στον αστέρα σύστημα αναφοράς δίνεται από την παρακάτω σχέση

$$\mathfrak{T}_{ij}^{(BF)} = \frac{1}{5}\kappa_n M a_i^2 \delta_{ij} \quad (2.135)$$

ΤΡΟΧΙΑΚΗ ΣΥΝΕΙΣΦΟΡΑ ΣΤΙΣ $\ddot{\mathfrak{T}}_{ij}$

Οι συνιστώσες του $\mathfrak{T}_{ij}$ στο αδρανειακό σύστημα αναφοράς για ένα διπλό σύστημα δύο σημειακών μαζών δίνονται από τις σχέσεις (2.94) – (2.97). Εισάγοντας στις δεύτερες χρονικές παραγώγους αυτών τις Νευτώνειες εξισώσεις κίνησης για διπλό σύστημα μελανής οπής – αστέρα νετρονίων

$$\ddot{r} = r\dot{\phi}^2 - \frac{GM_{tot}}{r^2} - \frac{3GM_{tot}\kappa_n}{10r^4}\left[a_1^2(3\cos^2\alpha - 1) + a_2^2(3\sin^2\alpha - 1) - a_3^2\right] \quad (2.136)$$

$$\ddot{\phi} = -\frac{2\dot{r}\dot{\phi}}{r} - \frac{3GM_{tot}\kappa_n}{10r^5}(a_1^2 - a_2^2) \quad (2.137)$$

καταλήγουμε στις εκφράσεις για τη τροχιακή συνεισφορά στις $\ddot{\mathfrak{T}}_{ij}$

$$\ddot{\mathfrak{T}}_{11}^{orb} = \frac{\mu}{3}\left\{\dot{r}^2 + r^2\dot{\phi}^2 - \frac{GM_{tot}}{r} - \frac{3\kappa_n GM_{tot}}{10r^3}[a_1^2(3\cos^2\alpha - 1) + a_2^2(3\sin^2\alpha - 1) - a_3^2]\right\}$$

$$+\mu\cos 2\phi\left\{\dot{r}^2 - r^2\dot{\phi}^2 - \frac{GM_{tot}}{r} - \frac{3\kappa_n GM_{tot}}{10r^3}[a_1^2(3\cos^2\alpha - 1) + a_2^2(3\sin^2\alpha - 1) - a_3^2]\right\}$$

$$-\mu\sin 2\phi\left[2r\dot{r}\dot{\phi} - \frac{3\kappa_n GM_{tot}}{10r^3}(a_1^2 - a_2^2)\sin 2\alpha\right] \quad (2.138)$$

$$\ddot{\mathfrak{T}}_{12}^{orb} = \mu\cos 2\phi\left[2r\dot{r}\dot{\phi} - \frac{3\kappa_n GM_{tot}}{10r^3}(a_1^2 - a_2^2)\sin 2\alpha\right] \quad (2.139)$$

$$-\mu\sin 2\phi\left\{\dot{r}^2 - r^2\dot{\phi}^2 - \frac{GM_{tot}}{r} - \frac{3\kappa_n GM_{tot}}{10r^3}[a_1^2(3\cos^2\alpha - 1) + a_2^2(3\sin^2\alpha - 1) - a_3^2]\right\}$$

$$\ddot{\mathfrak{T}}_{22}^{orb} = \frac{\mu}{3}\left\{\dot{r}^2 + r^2\dot{\phi}^2 - \frac{GM_{tot}}{r} - \frac{3\kappa_n GM_{tot}}{10r^3}[a_1^2(3\cos^2\alpha - 1) + a_2^2(3\sin^2\alpha - 1) - a_3^2]\right\}$$

$$-\mu\cos 2\phi\left\{\dot{r}^2 - r^2\dot{\phi}^2 - \frac{GM_{tot}}{r} - \frac{3\kappa_n GM_{tot}}{10r^3}[a_1^2(3\cos^2\alpha - 1) + a_2^2(3\sin^2\alpha - 1) - a_3^2]\right\}$$

$$+\mu\sin 2\phi\left[2r\dot{r}\dot{\phi} - \frac{3\kappa_n GM_{tot}}{10r^3}(a_1^2 - a_2^2)\sin 2\alpha\right] \quad (2.140)$$

$$\ddot{\mathfrak{T}}_{33}^{orb} = -\frac{2\mu}{3}\left\{\dot{r}^2 + r^2\dot{\phi}^2 - \frac{GM_{tot}}{r} - \frac{3\kappa_n GM_{tot}}{10r^3}[a_1^2(3\cos^2\alpha - 1) + a_2^2(3\sin^2\alpha - 1) - a_3^2]\right\} \quad (2,141)$$

Με βάση τα παραπάνω, οι δεύτερες χρονικές παράγωγοι του $\mathfrak{T}_{ij}$ θα είναι το άθροισμα αστρικών και τροχιακών όρων, που δίνονται από τις σχέσεις (2.131) – (2.134) και (2.138) – (2.141), αντίστοιχα.

ΚΕΦΑΛΑΙΟ 3^{Ο} "ΠΡΟΣΟΜΟΙΩΣΗ ΤΩΝ ΔΙΠΛΩΝ ΣΥΣΤΗΜΑΤΩΝ"

Το πρόγραμμα προσομοίωσης είναι γραμμένο σε γλώσσα προγραμματισμού C++ και ο πηγαίος κώδικας παρατίθεται στο παράρτημα αυτής της εργασίας. Στην πρώτη ενότητα αυτού του κεφαλαίου θα αναφερθούμε στις αριθμητικές μεθόδους που χρησιμοποιήθηκαν, ενώ στη δεύτερη θα περιγράψουμε συνολικά τη δομή και λειτουργία του προγράμματος.

3.1 ΑΡΙΘΜΗΤΙΚΕΣ ΜΕΘΟΔΟΙ

Η χρήση αριθμητικών μεθόδων κατά τη προσομοίωση διπλών συστημάτων είναι απαραίτητη στις ακόλουθες τρεις περιπτώσεις: α) στον υπολογισμό των ολοκληρωμάτων A_i (εξίσωση 2.20), β) στην επίλυση μη γραμμικών αλγεβρικών εξισώσεων για την κατασκευή συνθηκών ισορροπίας (περισσότερα στην ενότητα 3.2) και γ) στην επίλυση των διαφορικών εξισώσεων που περιγράφουν την εξέλιξη του συστήματος.

ΥΠΟΛΟΓΙΣΜΟΣ ΤΩΝ ΟΛΟΚΛΗΡΩΜΑΤΩΝ A_i

Τα ολοκληρώματα A_i δίνονται από τη σχέση

$$A_i \equiv a_1 a_2 a_3 \int_0^\infty \frac{du}{\Delta(a_i^2+u)}\,, \quad \Delta^2 = (a_1^2+u)(a_2^2+u)(a_3^2+u) \tag{3.1}$$

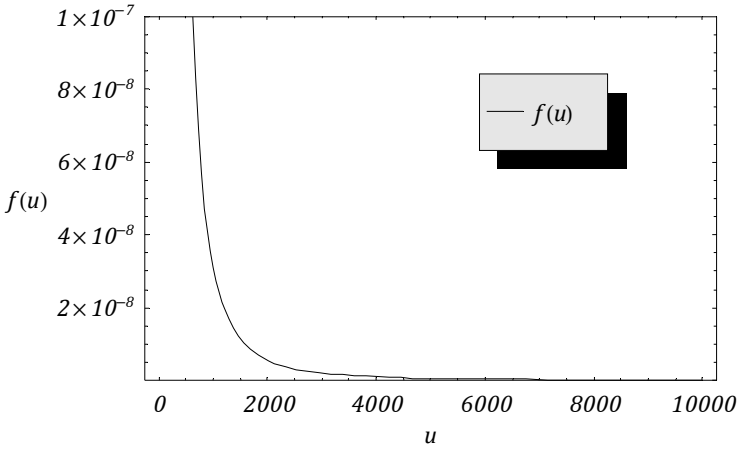
Για τον υπολογισμό τους χρησιμοποιήθηκε ο τύπος των Euler-MacLaurin, ο οποίος έχει ως εξής:

$$\int_{x_0}^{x_n} f(x)dx = \frac{h}{2}[f(x_0)+2f(x_1)+\cdots+2f(x_{n-1})+f(x_n)] - \frac{h^2}{12}[f'(x_n)-f'(x_0)]$$

$$+\frac{h^4}{720}[f'''(x_n)-f'''(x_0)] - \frac{h^6}{30240}\left[f^{(5)}(x_n)-f^{(5)}(x_0)\right] \tag{3.2}$$

Στην περίπτωση των A_i, η υπό ολοκλήρωση συνάρτηση είναι η

$$f(u) = \frac{1}{\Delta(a_i^2+u)} \tag{3.3}$$



Σχήμα 3. 1. Η συνάρτηση $f(u)$ για τυπικές τιμές των αξόνων του ελλειψοειδούς, $a_1 = a_2 = a_3 = 5$.

Αν και το άνω όριο του ολοκληρώματος είναι το άπειρο, ο αριθμητικός υπολογισμός του οφείλει να εκτείνεται μέχρι κάποιο πεπερασμένο όριο. Στο Σχήμα 3.1 φαίνεται η συμπεριφορά της συνάρτησης $f(u)$ για τυπικές τιμές των αξόνων του ελλειψοειδούς. Παρατηρούμε ότι πολύ γρήγορα φθίνει, γεγονός που μας επιτρέπει να σταματήσουμε την ολοκλήρωση σε κάποια πεπερασμένη τιμή χωρίς να εισάγουμε μεγάλο σφάλμα. Στο συγκεκριμένο πρόγραμμα προσομοίωσης το ολοκλήρωμα υπολογίστηκε στο διάστημα 0 ως 10000. Στο μεν διάστημα 0 ως 100 χρησιμοποιήθηκε βήμα 0.1, στο δε 100 ως 10000, βήμα 0.5 (μια και οι τιμές της $f(u)$ είναι σημαντικά μικρότερες).

Ο υπολογισμός των ολοκληρωμάτων γίνεται με τη κλάση ChandraIntegrals. Ο ορισμός της με κάποια σχόλια φαίνεται στο Πλαίσιο 1.

```
class ChandraIntegrals {
  private:
     double a1Loc;     // οι άξονες του αστέρα
     double a2Loc;
     double a3Loc;
     double norm;      // παράγοντας κανονικοποίησης των αξόνων (σε περίπτωση
                       // που παίρνουν πολύ μεγάλες ή πολύ μικρές τιμές)

  public:
      ChandraIntegrals(const double NormalizationFactor); // constructor
      ~ChandraIntegrals();  // destructor

     double f(double u);
     double df(double u);    // η συνάρτηση f και οι παράγωγοι αυτής
     double df3(double u);   // για τη τιμή u
     double df5(double u);

     // υπολογίζει ένα ολοκλήρωμα κάθε φορά (numberOfIntegral)
     double Calculate(int numberOfIntegral,double lowerLimit,
                   double upperLimit,double step,double* radiiArray);

     // υπολογίζει όλα τα ολοκληρώματα μαζί
     void CalculateAll(double lowerLimit,double upperLimit,double step,
                     double* radiiArray,double* IntegralsArray);
  };
```

Πλαίσιο 1. Ορισμός της κλάσης ChandraIntegrals, για τον υπολογισμό των ολοκληρωμάτων A_i.

Η υλοποίηση της κλάσης ChandraIntegrals βρίσκεται στο αρχείο ChanraIntegrals.cpp του παραρτήματος. Ένα παράδειγμα χρήσης της κλάσης αυτής φαίνεται στο Πλαίσιο 2.

```
#include "NumericalMethods.h"

int main ()
{
    double normalizationFactor=1.0;  // σε περίπτωση που οι a1,a2,a3 έχουν
                                     // πολύ μεγάλες ή πολύ μικρές τιμές
    int numberOfIntegral=0;          // ποιο από τα Ai θα υπολογιστεί
    double lowerLimit=0.0;
    double upperLimit=10000.0;       // τα όρια και το βήμα ολοκλήρωσης
    double step=0.1;

    double axisArray[3]={5.2,5.1,4.9};    // οι άξονες του αστέρα
    double A[3];                          // τα ολοκληρώματα


    // ορισμός ενός αντικειμένου ChandraIntegral
    NumericalMethods::ChandraIntegrals exampleIntegral(normalizationFactor);

    // υπολογισμός ενός ολοκληρώματος (A[0]≡A1, A[1]≡A2 κ.ο.κ.)
    A[0]=exampleIntegral.Calculate(numberOfIntegral,lowerLimit,upperLimit,
                                   step,axisArray);

    // εναλλακτικά, μπορούν να υπολογιστούν όλα ταυτόχρονα
    exampleIntegral.CalculateAll(lowerLimit,upperLimit,step,axisArray,A);


    return (0);

}
```

Πλαίσιο 2. Παράδειγμα χρήσης της κλάσης ChandraIntegrals.

Η παραπάνω κύρια συνάρτηση (main) κατασκευάζει ένα αντικείμενο ChandraIntegral και στη συνέχεια υπολογίζει την τιμή του, με βάση τα ορίσματα που έχουμε δώσει στη μέθοδο Calculate. Επιπλέον, μας δίνεται η δυνατότητα να υπολογιστούν όλα τα ολοκληρώματα ταυτόχρονα κάνοντας χρήση της μεθόδου CalculateAll.

ΕΠΙΛΥΣΗ ΜΗ ΓΡΑΜΜΙΚΩΝ ΑΛΓΕΒΡΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Τα διπλά συστήματα χωρίς εκπομπή βαρυτικής ακτινοβολίας, είναι συντηρητικά κι επομένως οι διαφορικές εξισώσεις εξέλιξής τους μαρτυρούν καταστάσεις ισορροπίας. Αν μηδενίσουμε σ' αυτές όλες τις χρονικές παραγώγους, τότε καθίσταται ένα σύστημα μη γραμμικών αλγεβρικών εξισώσεων, με την επίλυση του οποίου προκύπτουν οι καταστάσεις ισορροπίας. Οι συνθήκες ισορροπίας είναι χρήσιμες στον προσδιορισμό των αρχικών συνθηκών για τις διαφορικές εξισώσεις.

Η επίλυση των μη γραμμικών αλγεβρικών συστημάτων, γίνεται με τη μέθοδο Newton-Raphson. Σε κάθε βήμα της μεθόδου αυτής κατασκευάζεται το σύστημα

$$\begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_n} \end{pmatrix} \cdot \begin{pmatrix} dx_1 \\ \vdots \\ dx_n \end{pmatrix} = - \begin{pmatrix} f_1 \\ \vdots \\ f_n \end{pmatrix} \tag{3.4}$$

Το σύστημα (3.4) συνιστά ένα γραμμικοποιημένο αλγεβρικό σύστημα κι επιλύεται με κάποια μέθοδο επίλυσης γραμμικών συστημάτων, όπως η Gauss ή η LU.

Η επίλυση των μη γραμμικών συστημάτων γίνεται με τις κλάσεις NewtonRaphson και LinearSystem (για την επίλυση του γραμμικοποιημένου συστήματος). Ο ορισμός τους φαίνεται παρακάτω.

```
class NewtonRaphson
 {
    // συνάρτηση που περιέχει τις εξισώσεις και τις παραγώγους αυτών
    typedef void (*AlgEquations)(double* x,double* fx);
    typedef void (*AlgDerivatives)(double* x,double** dfdx);

  public:
    NewtonRaphson(int numberOfEquations,const AlgEquations,
                  const AlgDerivatives);
    ~NewtonRaphson();

    void OneNRStep(double* InitialValues,const int Solver=Gauss);
    void Solve(double* initialValues,double accuracy,const int Solver=Gauss);

  private:

    void ChangeSign(void);

    int n;         // αριθμός των εξισώσεων
    double* dx;    // διορθώσεις των μεταβλητών
    double* b;     // δεξί μέλος της εξίσωσης (3.5)
    double* temp;
    double** J;    // ιακωβιανός πίνακας

    NumericalMethods::LinearSystem* localSystem; // γραμμικοποιημένο σύστημα
    AlgEquations myEquations;     // συνάρτηση που περιέχει τις εξισώσεις
    AlgDerivatives myDerivatives; // συνάρτηση που περιέχει τις
                                  // μερικές παραγώγους
 };
```

Πλαίσιο 3. Ορισμός της κλάσης NewtonRaphson για την επίλυση μη γραμμικών αλγεβρικών συστημάτων.

```
class LinearSystem
{
 public:

    LinearSystem(int numberOfEquations);
    ~LinearSystem();

    // επίλυση με μέθοδο Gauss
    void GaussSolve(double **A,double *B,double *X);

    // επίλυση με μέθοδο LU
    void LUSolve(double** A,double* B,double* X);


 private:
    // μετατρέπει έναν πίνακα A σε L.U
    void LUDecomposition(double** a,double* d,int* indx);

   // επιλύει τον L.U
    void LUDecSolve(double** a,double* b,int* indx);

    int n;      // αριθμός εξισώσεων

};
```

Πλαίσιο 4. Ορισμός της κλάσης LinearSystem, για την επίλυση γραμμικών αλγεβρικών συστημάτων.

Η υλοποίηση των παραπάνω κλάσεων γίνεται στα αρχεία NewtonRaphson.cpp και LinearSystem.cpp, τα οποία παρατίθενται στο παράρτημα. Ένα παράδειγμα χρήσης της κλάσης NewtonRaphson φαίνεται στο παρακάτω πλαίσιο.

```
#include"NumericalMethods.h"

void myEquations(double*,double*);
void myDerivatives(double*,double**);


int main ()
{
  int numberOfEquations=2;    // ο αριθμός των εξισώσεων
  double x[2];                // οι άγνωστοι του συστήματος

  x[0]=0.1;                   // δοκιμαστικές τιμές
  x[1]=0.5;

 // κατασκευή του αντικειμένου, ορίζουμε τον αριθμό των εξισώσεων και τις
 // συναρτήσεις που περιέχουν τις εξισώσεις και τις μερικές παραγώγους αυτών
  NumericalMethods::NewtonRaphson exampleSystem(numberOfEquations,
                                          myEquations,myDerivatives);

 // επίλυση του συστήματος, ορίζουμε την ακρίβεια και τη μέθοδο (Gauss ή LU)
  exampleSystem.Solve(x,pow(10,-12),NumericalMethods::Gauss);

  return (0);
}

void myEquations(double* x,double* fx)
{
  fx[0]=-(exp(x[0])-3.0*x[1]-1.0);    // εξισώσεις προς επίλυση
  fx[1]=-(x[0]*x[0]+x[1]*x[1]-4.0);
}

void myDerivatives(double* x,double** dfdx)
{
  dfdx[0][0]=exp(x[0]);
  dfdx[0][1]=-3.0;                    // μερικές παράγωγοι των εξισώσεων
  dfdx[1][0]=2.0*x[0];
  dfdx[1][1]=2.0*x[1];
}
```

Πλαίσιο 5. Παράδειγμα χρήσης της κλάσης NewtonRaphson.

Τέλος, για την επίλυση μη γραμμικών αλγεβρικών συστημάτων κατασκευάστηκε και η κλάση NewtonRaphsonGL, που συνδυάζει τη μέθοδο Newton-Raphson και τη μέθοδο της διχοτόμησης για περισσότερο αποτελεσματική σύγκλιση.

ΕΠΙΛΥΣΗ ΣΥΣΤΗΜΑΤΟΣ ΔΙΑΦΟΡΙΚΩΝ ΕΞΙΣΩΣΕΩΝ

Η χρονική εξέλιξη των διπλών συστημάτων επιτυγχάνεται με την επίλυση των διαφορικών εξισώσεων κίνησης. Η επίλυση αυτών γίνεται με τη μέθοδο Runge-Kutta 4^{ης} τάξης. Σύμφωνα μ' αυτή, για ένα σύνολο εξισώσεων της μορφής

$$\frac{dx_i}{dt} = z_i \quad \text{και} \quad \frac{dz_i}{dt} = f_i \tag{3.5}$$

η χρονική εξέλιξη προέρχεται από τις αναδρομικές σχέσεις

$$x_{n+1}^i = x_n^i + \frac{1}{6}\left(k_1^i + 2k_2^i + 2k_3^i + k_4^i\right) \tag{3.6α}$$

$$z_{n+1}^i = z_n^i + \frac{1}{6}\left(g_1^i + 2g_2^i + 2g_3^i + g_4^i\right) \tag{3.6β}$$

όπου οι συντελεστές δίνονται από τις παρακάτω σχέσεις

$$k_1^i = hf_i(x^i) \quad \text{και} \quad g_1^i = hz_i \tag{3.7α}$$

$$k_2^i = hf_i\left(x^i + \frac{1}{2}k_1^i\right) \quad \text{και} \quad g_2^i = h\left(z_i + \frac{1}{2}g_1^i\right) \tag{3.7β}$$

$$k_3^i = hf_i\left(x^i + \frac{1}{2}k_2^i\right) \quad \text{και} \quad g_3^i = h\left(z_i + \frac{1}{2}g_2^i\right) \tag{3.7γ}$$

$$k_4^i = hf_i\left(x^i + k_3^i\right) \quad \text{και} \quad g_4^i = h\left(z_i + g_3^i\right) \tag{3.7δ}$$

Η υλοποίηση της παραπάνω μεθόδου γίνεται στις κλάσεις RK4 και RK4AdaptStep. Στη μεν πρώτη το βήμα για την επαναληπτική διαδικασία είναι σταθερό, ενώ στη δεύτερη μεταβάλλεται διατηρώντας την επιθυμητή ακρίβεια στο αποτέλεσμα. Τόσο οι ορισμοί, όσο και η υλοποίηση των παραπάνω κλάσεων βρίσκονται στο παράρτημα. Παρακάτω, παραθέτουμε μόνο τον ορισμό της RK4, καθώς και ένα παράδειγμα χρήσης των κλάσεων αυτών.

```
typedef void (*ODEs)(const double x,double* y,double* dydt);


class RK4
{
 public:

    RK4(int numberOfEquations,const ODEs,double independentVariables,
       double* dependentVariables,double stepSize);
    ~RK4();

    // επιλύει τις εξισώσεις για "howManySteps" βήματα. Προεπιλεγμένη
     // τιμή είναι το 1
    void Solve(int howManySteps=1);

    // επιστρέφει την ανεξάρτητη μεταβλητή
    double GetX(void){return(x);}

 private :
    double* k1;
    double* k2;    // οι συντελεστές της μεθόδου Runge-Kutta
    double* k3;
    double* k4;
    double* yTemp;

    int n;         // αριθμός των εξισώσεων
    double h;      // βήμα
    double h2;     // βήμα/2 (για γρηγορότερη επίλυση)
    double h6;     // βήμα/6
    double x;      // ανεξάρτητη μεταβλητή (π.χ. χρόνος)
    double* y;     // εξαρτημένες μεταβλητές

   ODEs myEquations;  // συνάρτηση που θα περιέχει τις εξισώσεις
};
```

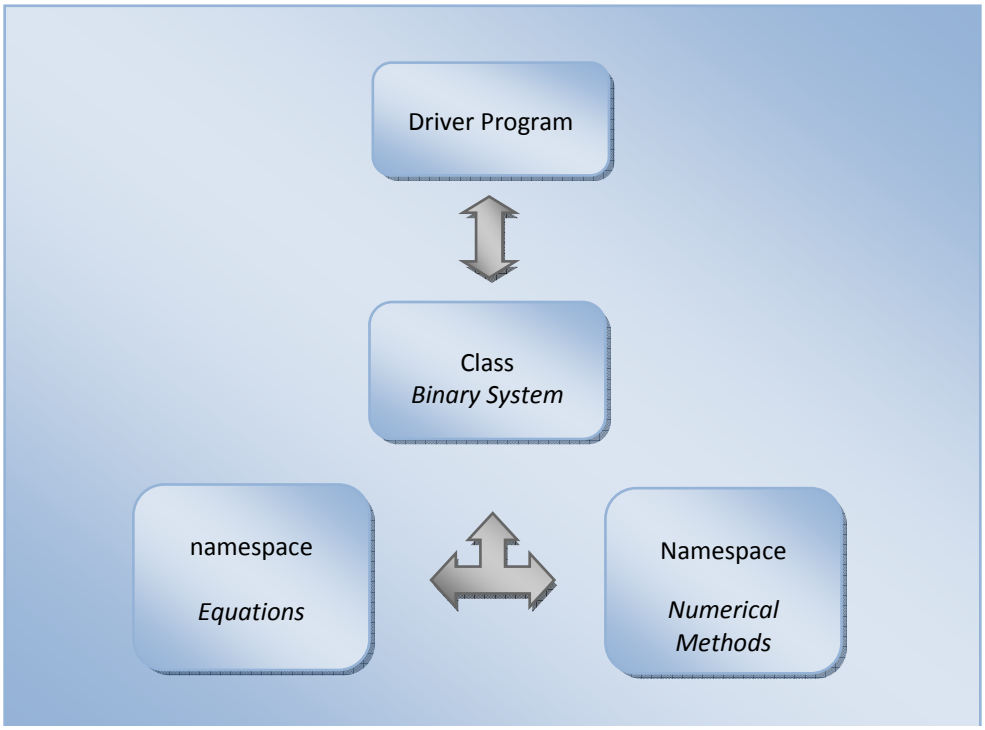
Πλαίσιο 6. Ορισμός της κλάσης RK4, για την επίλυση διαφορικών εξισώσεων.

```
#include "NumericalMethods.h"

void myODEs(const double ,double*,double*);

int main ()
{
   // μεταβλητές και για τις δύο μεθόδους
   int numberOfEquations=1;  // ο αριθμός των διαφορικών εξισώσεων
   int howManySteps=1;       // ο αριθμός των βημάτων προς εκτέλεση
   double time=0.0;          // η ανεξάρτητη μεταβλητή, εδώ ο χρόνος
   double stepSize=0.1;      // το βήμα (σταθερό)
   double y[1];              // εξαρτημένες μεταβλητές

   // μεταβλητές για την RK4AdaptStep
   double trialStepSize=0.01;  // δοκιμαστικό βήμα, για την εκκίνηση
   double accuracy=pow(10,-8); // επιθυμητή ακρίβεια


   y[0]=-1.0;                // αρχική συνθήκη

   // κατασκευή αντικειμένου RK4 και αρχικοποίηση
   NumericalMethods::RK4 exampleRK4(numberOfEquations,myODEs,
                                    time,y,stepSize);

   // επίλυση της εξίσωσης για "howManySteps" βήματα
   exampleRK4.Solve(howManySteps);


   y[0]=-1.0;                // αρχική συνθήκη εκ νέου

   // κατασκευή αντικειμένου RK4AdaptStep και αρχικοποίηση
   NumericalMethods::RK4AdaptStep exampleRK4AdaptStep(numberOfEquations,
                                   myODEs,time,y,trialStepSize,accuracy);

   // επίλυση της εξίσωσης για "howManySteps" βήματα
   exampleRK4AdaptStep.Solve(howManySteps);


   return (0);
 }


 void myODEs(const double x,double* y,double* dydt)
{
   dydt[0]=-2.0*x-y[0];        // διαφορική εξίσωση προς επίλυση
}
```

Πλαίσιο 7. Παράδειγμα χρήσης της κλάσης RK4.

3.2 ΔΟΜΗ ΚΑΙ ΛΕΙΤΟΥΡΓΙΑ ΤΟΥ ΠΡΟΓΡΑΜΜΑΤΟΣ ΠΡΟΣΟΜΟΙΩΣΗΣ

Το πρόγραμμα προσομοίωσης συνίσταται από τέσσερα τμήματα που αλληλεπιδρούν μεταξύ τους. Στο Σχήμα 3.2 φαίνεται σχηματικά η δομή του προγράμματος. Η κύρια συνάρτηση του προγράμματος (main) βρίσκεται στο αρχείο με όνομα DriverProgram.cpp. Μέσα απ' αυτό ο χρήστης μπορεί να κατασκευάσει ένα ή περισσότερα διπλά συστήματα (με τη χρήση της κλάσης BinarySystem.cpp) και με μια σειρά από εντολές να τα εξελίξει κατάλληλα. Στο namespace Equation βρίσκονται τόσο οι αλγεβρικές εξισώσεις (για τον προσδιορισμό καταστάσεων ισορροπίας), όσο και οι διαφορικές εξισώσεις εξέλιξης των διπλών συστημάτων. Τέλος, στο namespace NumericalMethods βρίσκονται όλες οι κλάσεις με τις αριθμητικές μεθόδους που περιγράφηκαν στην προηγούμενη ενότητα.



Σχήμα 3. 2. Η δομή του προγράμματος προσομοίωσης διπλών συστημάτων και η αλληλεπίδραση των τμημάτων του.

Η βασική κλάση με την οποία έρχεται σ' επαφή ο χρήστης είναι η BinarySystem. Μέσα απ' αυτή είναι δυνατός ο ορισμός, η αρχικοποίηση και εξέλιξη ενός διπλού συστήματος. Η υλοποίηση της κλάσης αυτής παρατίθεται στο παράρτημα, ενώ ο ορισμός της φαίνεται στο παρακάτω πλαίσιο.

```
enum typeOfSystem {BHBH,BHNS};
enum equilibriumValues {axis1,axis2,axis3,OmegaOrbital};


class BinarySystem
{
 public:
   // ο constructor καλείται καθορίζοντας το τύπο του συστήματος
   BinarySystem(const int typeOfSystem);
   ~BinarySystem();

   // προσδιορισμός των ιδιοτήτων του συστήματος
   // οι διπλοί ορισμοί αντιστοιχούν στους δύο τύπους
   // συστημάτων BH-BH και BH-NS
   void SetSystem(double massRatio);
   void SetSystem(double massRatio,double RoTrial,double n1,double kn1);
   void SetSeperation(double r);
   void SetFluidProperties(double Omega1, double Lamda1);

   // κατάσταση ισορροπίας και πρόσβαση στις τιμές ισορροπίας
   void ReachEquilibrium(void);
   double GetEquilibriumValue(const int equilibriumValue);

   // προσδιορισμός των αρχικών συνθηκών για τις Δ.Ε.(BH-BH και BH-NS)
   void SetInitialConditions(double r,double phiAngle,
                             double rdot,double OmegaOrbital);

    void SetInitialConditions(double axis1,double axis2,double axis3,
        double gammaAngle,double Lamda1,double r, double phiAngle,
        double a1dot,double a2dot,double a3dot,double Omega1,double rdot,
        double OmegaOrbital);
```

```
   // πρόσβαση στις τιμές Ro και Horb/Hstar
   double GetRo(void);
   double GetHamiltonianRatio(void);

   // χρονική εξέλιξη των συστημάτων (RK4 σταθερό και μεταβλητό βήμα)
   void Evolve(double time,double timeStep,double howManyPeriods,
               const char* filename, int printScreenFrequency=100,
               int printFileFrequency=1);

   void Evolve(double time,double accuracy,int howManyPeriods,
               const char* filename,int printScreenFrequency=100,
               int printFileFrequency=1);

   // εκτύπωση (στην οθόνη) μερικών ιδιοτήτων του συστήματος
   void printBinaryProperties(void);


private:

   const int typeOfBinary; // τύπος συστήματος (BH-BH ή BH-NS)

   int nAlg;          // αποθήκευση εξισώσεων
   int nDif;
   double* xAlg;
   double* xDif;

   double M1;     // μάζα του αστέρα νετρονίων
   double M2;     // μάζα της μελανής οπής
   double Ro1;    // ακτίνα σφαιρικού αστέρα νετρονίων
   double n1;     // πολυτροπικός δείκτης
   double kn1;    // παράμετρος εξαρτώμενη μόνο από το n1

   double r;      // απόσταση μεταξύ των σωμάτων

   double Omega1;         // γωνιακή συχνότητα περιστροφής του αστέρα
   double L1;             // γωνιακή συχνότητα εσωτερικής περιστροφής

    double ThetaCap;    // γωνίες παρατήρησης
    double PhiCap;
};
```

Πλαίσιο 9. Ορισμός της κλάσης BinarySystem, με την οποία χειριζόμαστε τα διπλά συστήματα.

Η δημιουργία ενός αντικειμένου BinarySystem γίνεται μέσω του constructor και ταυτόχρονα προσδιορίζεται και ο τύπος του συστήματος (BH-BH ή BH-NS). Με τις μεθόδους τύπου Set προσδιορίζονται περισσότερο τα χαρακτηριστικά του συστήματος. Η δε μέθοδος SetSystem() είναι υπερφορτωμένη, ώστε να καλύπτει τους δύο τύπους (BH-BH ή BH-NS). Η ReachEquilibrium() προσδιορίζει τις τιμές ισορροπίας (επιλύνοντας το μη γραμμικό αλγεβρικό σύστημα), ενώ η μέθοδος GetEquilibriumValues() παρέχει πρόσβαση στις τιμές αυτές. Οι δύο υπερφορτώσεις της μεθόδου SetInitialConditions() καθορίζουν τις αρχικές συνθήκες των διαφορικών εξισώσεων εξέλιξης των συστημάτων. Η GetRo() και GetHamiltonianRatio() χρησιμοποιούνται μόνο στη περίπτωση διπλού συστήματος μελανής οπής – αστέρα νετρονίων. Η πρώτη από αυτές υπολογίζει την ακτίνα, R_0 (ακτίνα σφαιρικού πολύτροπου με μάζα ίδια μ' αυτή του ελλειψοειδούς) μέσω της σχέσης

$$R_0 = R\left[\frac{3\sin^{-1}e}{e}(1-e^2)^{1/6}\left(1-\frac{1}{e^2}+\frac{\sqrt{1-e^2}}{e\sin^{-1}e}\right)\right]^{n/3-n} \tag{3.8}$$

όπου $R=(a_1a_2a_3)^{1/3}$ και $e=\sqrt{1-(a_3/a_1)^2}$. Η δε μέθοδος GetHamiltonianRatio() υπολογίζει το λόγο των συναρτήσεων Hamilton: H_{orb}/H_{star}. Ο λόγος αυτός είναι ενδεικτικός για τη συνεισφορά της δομής του αστέρα στην εξέλιξη του συστήματος. Έτσι, για ένα σύστημα με μεγάλη απόσταση μεταξύ των συνοδών η τιμή του είναι $\sim 0.1-0.2$ κι επομένως η συνεισφορά των αστρικών βαθμών

ελευθεριάς στη χρονική εξέλιξη πολύ μικρή. Αντιθέτως, σ' ένα σύστημα με μικρή απόσταση μεταξύ των συνοδών του, ο παραπάνω λόγος παίρνει τιμές > 1. Στην περίπτωση αυτή, η δομή του αστέρα επηρεάζει σημαντικά την εξέλιξη του συστήματος. Τέλος, με τις δύο υπερφορτώσεις της μεθόδου Evolve(), που αντιστοιχούν στις μεθόδους Runge-Kutta 4^{ης} τάξης με σταθερό και μεταβλητό βήμα, εξελίσσεται το διπλό σύστημα κι εκτυπώνονται (στην οθόνη και σε αρχείο) τα αποτελέσματα.

Παρακάτω παρατίθεται ένα παράδειγμα δημιουργίας κι εξέλιξης δύο συστημάτων, ενός αποτελούμενο από δύο μελανές οπές και ενός από μια μελανή οπή κι έναν αστέρα νετρονίων.

```
#include"BinarySystem.h"


int main ()
{

   double massRatio=1.0;
   double r=30.0;
                          // ο συμβολισμός των μεταβλητών
   double n1=0.5;        // είναι ίδιος με τον ορισμό της
    double kn1=0.81482;  // κλάσης BinarySystem
   double Omega1=0.0001;
   double Lamda1=0.000065;
   double RoTrial=6.0;
   double Ro1,HamiltonianRatio;

   double a1,a2,a3;
   double gammaAngle=0.0;
   double phiAngle=0.0;
   double a1dot=.0002;
   double a2dot=0.001;

   double a3dot=0.001;
   double rdot=0.001;
   double OmegaOrb;

   int periodsToDo=10;
   double time=0.0;
   double timeStep=0.1;
   double accuracy=pow(10.0,-8.0);


   // ΠΑΡΑΔΕΙΓΜΑ ΣΥΣΤΗΜΑΤΟΣ ΔΥΟ ΜΕΛΑΝΩΝ ΟΠΩΝ

   // δημιουργία συστήματος
   BinarySystem exampleBinary(BHBH);

   // καθορισμός των παραμέτρων του
   exampleBinary.SetSystem(massRatio);
   exampleBinary.SetSeperation(r);

   // κατάσταση ισορροπίας
   exampleBinary.ReachEquilibrium();

   // αρχικοποίηση με βάση τις τιμές ισορροπίας
   OmegaOrb=exampleBinary.GetEquilibriumValue(OmegaOrbital);

   // καθορισμός των αρχικών συνθηκων
   exampleBinary.SetInitialConditions(r,phiAngle,rdot,OmegaOrb);

   // εκτύπωση τιμών πριν από την εξέλιξη
   exampleBinary.printBinaryProperties();

   // εξέλιξη με την RK4 μεταβλητού βήματος
   exampleBinary.Evolve(time,accuracy,periodsToDo,"example.dat",1,1);
```

Πλαίσιο 10. Παράδειγμα χρήσης της κλάσης BinarySystem, για την δημιουργία κι εξέλιξη δύο διαφορετικών τύπων συστημάτων.

```
// ΠΑΡΑΔΕΙΓΜΑ ΣΥΣΤΗΜΑΤΟΣ ΜΕΛΑΝΗΣ ΟΠΗΣ-ΑΣΤΕΡΑ ΝΕΤΡΟΝΙΩΝ

 // δημιουργία συστήματος
BinarySystem exampleBinary2(BHNS);

 // καθορισμός των παραμέτρων του
 exampleBinary2.SetSystem(massRatio,RoTrial,n1,kn1);
 exampleBinary2.SetSeperation(r);
 exampleBinary2.SetFluidProperties(Omega1,Lamda1);

 // κατάσταση ισορροπίας
 exampleBinary2.ReachEquilibrium();

 // αρχικοποίηση με βάση τις τιμές ισορροπίας
 a1=exampleBinary2.GetEquilibriumValue(axis1);
 a2=exampleBinary2.GetEquilibriumValue(axis2);
 a3=exampleBinary2.GetEquilibriumValue(axis3);
 OmegaOrb=exampleBinary2.GetEquilibriumValue(OmegaOrbital);


 // καθορισμός των αρχικών συνθηκών
 exampleBinary2.SetInitialConditions(axis1,axis2,axis3,gammaAngle,Lamda1,r,
                    phiAngle,a1dot,a2dot,a3dot,Omega1,rdot,OmegaOrb);


 // υπολογισμός των Ro και λόγου Horb/Hstar
 Ro1=exampleBinary2.GetRo();
 HamiltonianRatio=exampleBinary2.GetHamiltonianRatio();

 // εκτύπωση τιμών πριν την εξέλιξη
 exampleBinary2.printBinaryProperties();

 // εξέλιξη με την RK4 σταθερού βήματος
 exampleBinary2.Evolve(time,timeStep,periodsToDo,"example2.dat",1,1);

 return(0);

}
```

Πλαίσιο 11. (συνέχεια από τη προηγούμενη σελίδα).

Οι μονάδες που χρησιμοποιούνται για την αριθμητική επίλυση των εξισώσεων είναι τέτοιες ώστε $G = c = M_{total} = 1$. Σ' αυτό το σύστημα μονάδων τα θεμελιώδη μεγέθη: μήκος (L), μάζα (M) και χρόνος (T), εκφράζονται από τους παρακάτω συνδυασμούς

$$L = \frac{GM_{total}}{c^2} \tag{3.9}$$

$$M = M_{total} \tag{3.10}$$

$$T = \frac{GM_{total}}{c^3} \tag{3.11}$$

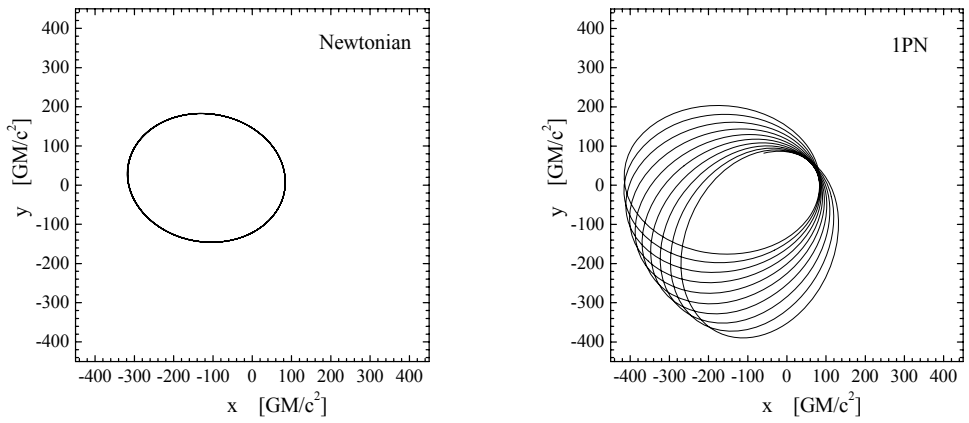
Οι παραπάνω μονάδες θα χρησιμοποιηθούν για την παρουσίαση των αποτελεσμάτων στο κεφάλαιο που ακολουθεί.

ΚΕΦΑΛΑΙΟ 4^{Ο} "ΑΠΟΤΕΛΕΣΜΑΤΑ"

Στο κεφάλαιο αυτό παραθέτουμε τα αποτελέσματα του μοντέλου που χρησιμοποιήσαμε για την προσομοίωση των διπλών συστημάτων. Συγκεκριμένα, στην πρώτη ενότητα εξετάζουμε τη συνεισφορά των επιμέρους όρων (αστρικοί βαθμοί ελευθερίας, μετανευτώνεια διόρθωση) στην εξέλιξη του συστήματος. Στη δεύτερη ενότητα παρουσιάζουμε ορισμένες χαρακτηριστικές περιπτώσεις εξέλιξης διπλών συστημάτων κι ελέγχουμε κατά πόσο επηρεάζεται η εξέλιξη μεταβάλλοντας κάποιες από τις βασικές παραμέτρους του συστήματος (πολυτροπικός δείκτης, λόγος μαζών, μέση ακτίνα των αστέρων). Στην τρίτη ενότητα παρουσιάζουμε τις κυματομορφές της βαρυτικής ακτινοβολίας των συστημάτων, ενώ στη τέταρτη ενότητα μελετάμε τους τρόπους δόνησης των αστέρων καθώς και την επίδρασή τους στις κυματομορφές της βαρυτικής ακτινοβολίας. Τέλος, κλείνουμε το κεφάλαιο με συμπεράσματα και σχόλια για τα αποτελέσματα που παρουσιάσαμε.

4.1 ΣΥΝΕΙΣΦΟΡΑ ΤΩΝ ΕΠΙΜΕΡΟΥΣ ΟΡΩΝ ΣΤΙΣ ΕΞΙΣΩΣΕΙΣ

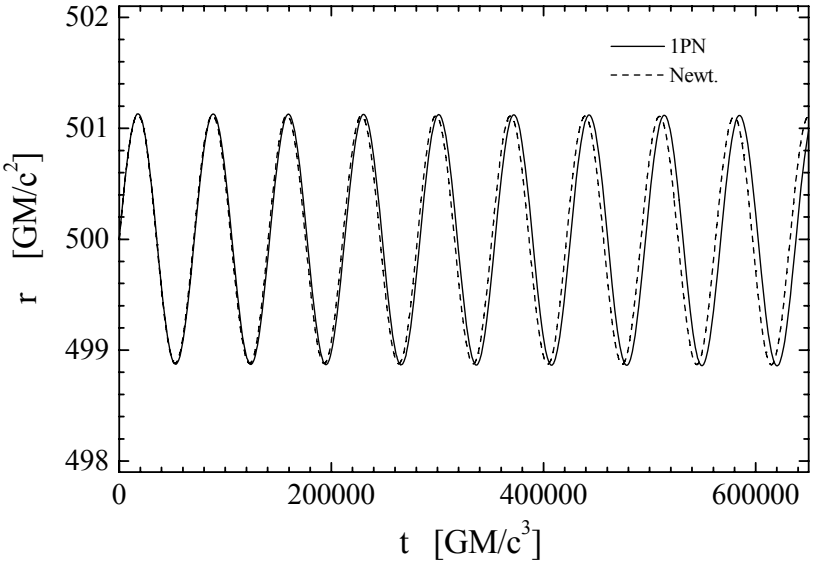
Σε αντίθεση με τις Νευτώνειες εξισώσεις, η πρώτη μετανευτώνεια διόρθωση είναι σε θέση να κάνει εμφανές το φαινόμενο της μετατόπισης του περιάστρου. Στο Σχήμα 4.1 φαίνεται η εξέλιξη ενός συστήματος δύο μελανών οπών. Στην πρώτη περίπτωση έχουμε χρησιμοποιήσει καθαρά Νευτώνειες εξισώσεις, ενώ στη δεύτερη περίπτωση χρησιμοποιήσαμε την μετανευτώνεια διόρθωση. Όπως αναμένεται, μόνο στην πρώτη περίπτωση έχουμε μετατόπιση του περιάστρου.



.

Σχήμα 4. 1. Το φαινόμενο της μετατόπισης περιάστρου για ένα σύστημα δύο μελανών οπών ίδιας μάζας.

Η διαφοροποίηση της εξέλιξης ενός συστήματος εξαιτίας της μετανευτώνειας διόρθωσης είναι εμφανής ακόμη και για μεγάλες αποστάσεις μεταξύ των αστέρων. Στο Σχήμα 4.2 φαίνεται η εξέλιξη ενός διπλού συστήματος δύο μελανών οπών (με ίδιες μάζες) με Νευτώνειες εξισώσεις και με εξισώσεις που περιέχουν τη μετανευτώνεια διόρθωση. Στις αρχικές συνθήκες του συστήματος έχουμε εισάγει μια ακτινική διαταραχή[3], $\dot{r} = 10^{-4}$, ώστε να γίνουν εμφανείς οι διαφορές στην εξέλιξη των συστημάτων.



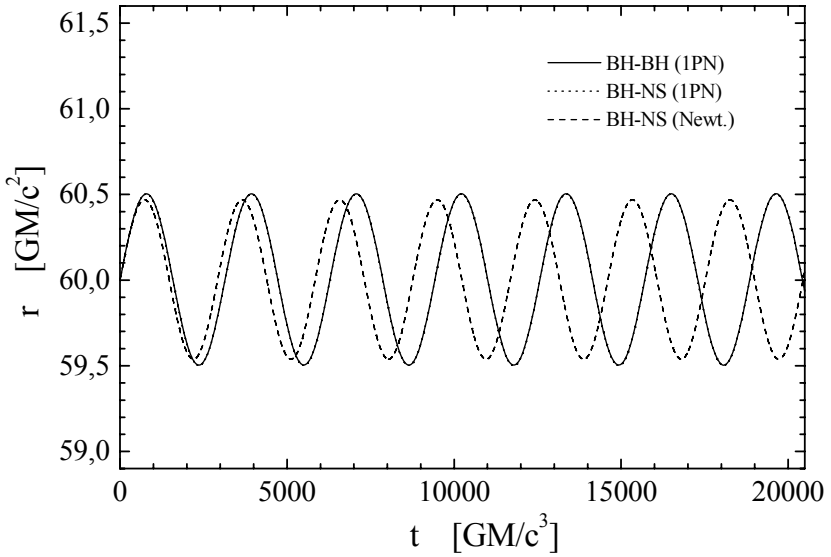
Σχήμα 4. 2. Χρονική εξέλιξη ενός διαταραγμένου συστήματος δύο μελανών οπών με ίδια μάζα, με Νευτώνειες και μετανευτώνειες εξισώσεις.

Παρατηρούμε ότι ακόμη και για την απόσταση αυτή, που είναι εξαιρετικά μεγάλη για τα διπλά συστήματα που μελετάμε, η μετανευτώνεια διόρθωση διαφοροποιεί αρκετά την εξέλιξη.

Αντιθέτως, η αστρική δομή δεν αναμένεται να διαφοροποιήσει σημαντικά την εξέλιξη του συστήματος, παρά μόνο όταν η απόσταση των αστέρων γίνει πολύ μικρή, δηλαδή κατά τη τελική φάση της ζωής τους. Κατά τις τελευταίες αυτές περιστροφές αναπτύσσονται δυναμικές αστάθειες, οι οποίες επιταχύνουν την κατάρρευση των συστημάτων. Στα σχήματα που ακολουθούν εξετάζουμε τη συνεισφορά των της αστρικής δομής στην εξέλιξη των συστημάτων.

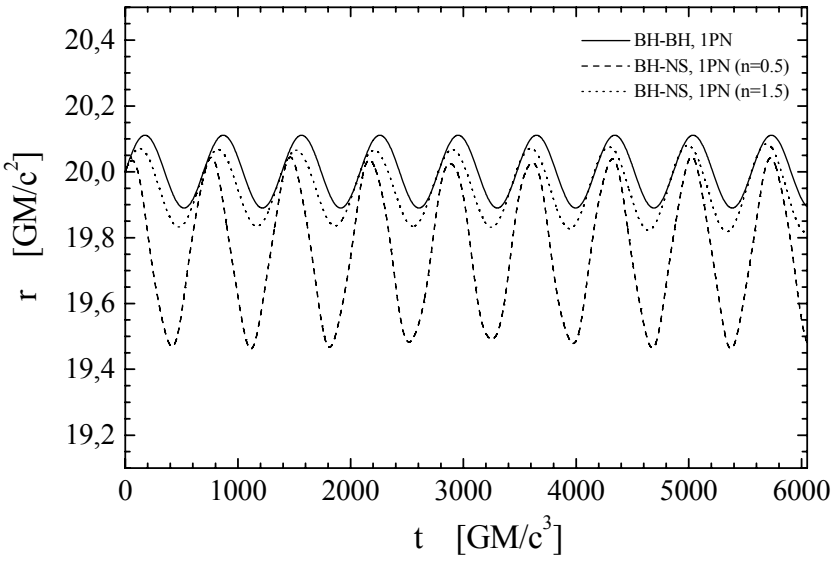
Στο Σχήμα 4.3 συγκρίνουμε την εξέλιξη διπλών συστημάτων (με αστέρες ίδιας μάζας) για σχετικά μεγάλη απομάκρυνση μεταξύ των συνοδών. Στα συστήματα αυτά δεν έχουμε λάβει υπόψη την εκπομπή βαρυτικής ακτινοβολίας και όπως και στη προηγούμενη περίπτωση έχουμε μια ακτινική διαταραχή $\dot{r} = 10^{-3}$. Παρατηρούμε ότι η εξέλιξη του συστήματος δύο μελανών οπών (BH-BH, 1PN) και μελανής οπής – αστέρα νετρονίων (BH-NS, 1PN) ταυτίζονται. Συνεπώς, η συνεισφορά της αστρικής δομής είναι αμελητέα για μεγάλες απομακρύνσεις. Αντιθέτως, παρατηρούμε ότι η μετανευτώνεια διόρθωση αλλάζει σημαντικά την εξέλιξη του συστήματος σε σχέση με τη Νευτώνεια προσέγγιση.

[3] Τα μεγέθη εκφράζονται στις μονάδες που αναφέραμε στην ενότητα 3.2, σελ. 47.

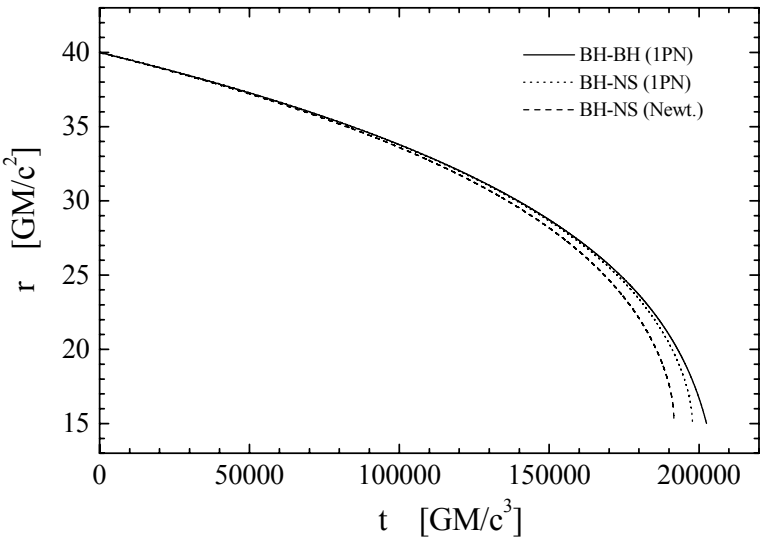


Σχήμα 4. 3. Χρονική εξέλιξη διπλών συστημάτων με τα ίδια χαρακτηριστικά. Σύστημα δύο μελανών οπών (BH-BH, 1PN), μελανής οπής – αστέρα νετρονίων με μετανευτώνεια διόρθωση (BH-NS, 1PN) και με νευτώνειες εξισώσεις (BH-NS, Newt.).

Στο Σχήμα 4.4 συγκρίνουμε την εξέλιξη για μικρή απομάκρυνση. Για την απομάκρυνση αυτή παρατηρούμε ότι η αστρική δομή διαφοροποιεί σημαντικά την εξέλιξη ενός συστήματος μελανής οπής – αστέρα νετρονίων σε σχέση μ' ένα σύστημα δύο μελανών οπών. Στο ίδιο σχήμα μπορούμε να δούμε και την επίδραση του πολυτροπικού δείκτη στην εξέλιξη.



Σχήμα 4. 4. Χρονική εξέλιξη ενός συστήματος δύο μελανών οπών (BH-BH, 1PN) και δύο συστημάτων μελανής οπής – αστέρα νετρονίων (BH-NS, 1PN) με διαφορετικούς πολυτροπικούς δείκτες.



Σχήμα 4. 5. Χρονική εξέλιξη ενός συστήματος δύο μελανών οπών (BH-BH, 1PN) και δύο συστημάτων μελανής οπής – αστέρα νετρονίων με μετανευτώνεια διόρθωση (BH-NS, 1PN) και με Νευτώνειες εξισώσεις (BH-NS, Newt.). Στα συστήματα αυτά έχει ληφθεί υπόψη η εκπομπή βαρυτικής ακτινοβολίας.

Τέλος, στο Σχήμα 4.5 βλέπουμε την εξέλιξη παρόμοιων συστημάτων, έχοντας λάβει υπόψη την εκπομπή βαρυτικής ακτινοβολίας. Η συνεισφορά των επιμέρους όρων στις εξισώσεις είναι παρόμοια με τις προηγούμενες περιπτώσεις. Επιπλέον, παρατηρούμε ότι λαμβάνοντας υπόψη τους αστρικούς βαθμούς ελευθερίας επιταχύνεται η κατάρρευση των διπλών συστημάτων, γεγονός που οφείλεται στην ανάπτυξη δυναμικών ασταθειών κατά την τελευταία φάση της εξέλιξης.

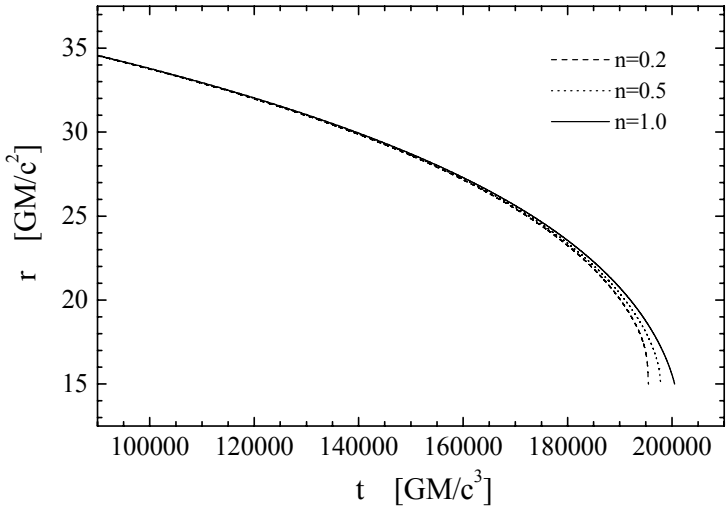
4.2 ΕΞΕΛΙΞΗ ΔΙΠΛΩΝ ΣΥΣΤΗΜΑΤΩΝ

Οι φυσικές παράμετροι ενός διπλού συστήματος καθορίζουν σε μεγάλο βαθμό την εξέλιξή του. Ορισμένες από τις βασικότερες παραμέτρους είναι ο πολυτροπικός δείκτης, ο λόγος μαζών των δύο αστέρων και η ακτίνα του αστέρα νετρονίων (αφού θεωρούμε τη μελανή οπή σημειακή). Στην ενότητα αυτή θα μελετήσουμε την επίδραση των παραπάνω παραμέτρων στην εξέλιξη ενός διπλού συστήματος.

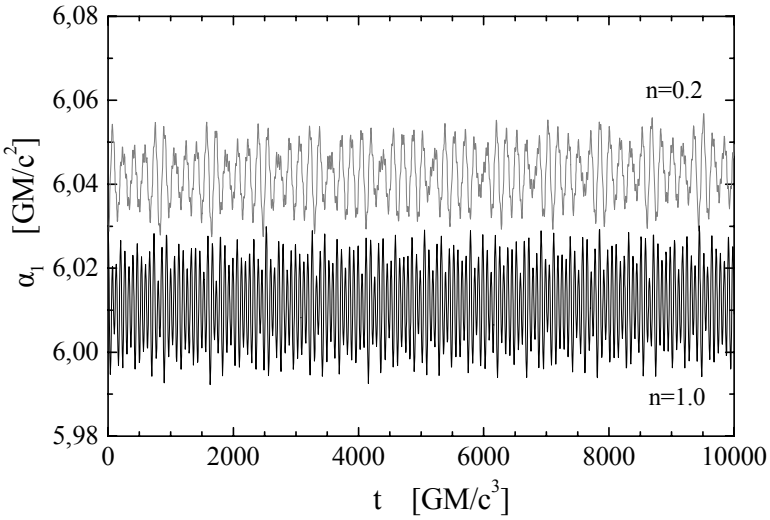
Η ΕΠΙΔΡΑΣΗ ΤΟΥ ΠΟΛΥΤΡΟΠΙΚΟΥ ΔΕΙΚΤΗ

Στο Σχήμα 4.6 βλέπουμε την εξέλιξη τριών συστημάτων μελανής οπής – αστέρα νετρονίων ($M_{BH}/M_{NS} = 1.0,\ R_0 = 6$) με διαφορετικούς πολυτροπικούς δείκτες. Καταγράφουμε την εξέλιξη των συστημάτων για απομάκρυνσεις μεγαλύτερες της $r = 15$, ώστε να είμαστε συνεπείς στους περιορισμούς της πρώτη μετανευτώνεια προσέγγιση ($v < 0.1c$) και διαπιστώνουμε ότι η εξέλιξη των συστημάτων διαφοροποιείται μόνο κατά τη τελική φάση της εξέλιξής τους. Συγκεκριμένα, όσο μικρότερος είναι ο πολυτροπικός δείκτης τόσο πιο γρήγορα καταρρέει το σύστημα. Η συμπεριφορά αυτή μπορεί να ερμηνευτεί αν λάβει κανείς υπόψη πως μεγάλες τιμές του πολυτροπικού δείκτη αντιστοιχούν σε περισσότερο συμπιεστό ρευστό. Ένας αστέρας αποτελούμενος από τέτοιο ρευστό

έχει την τάση να διατηρεί καλύτερα το σφαιρικό του σχήμα κι επομένως να καθυστερεί την εμφάνιση προεξοχών στην επιφάνειά του, οι οποίες οδηγούν στην εμφάνιση δυναμικών ασταθειών.



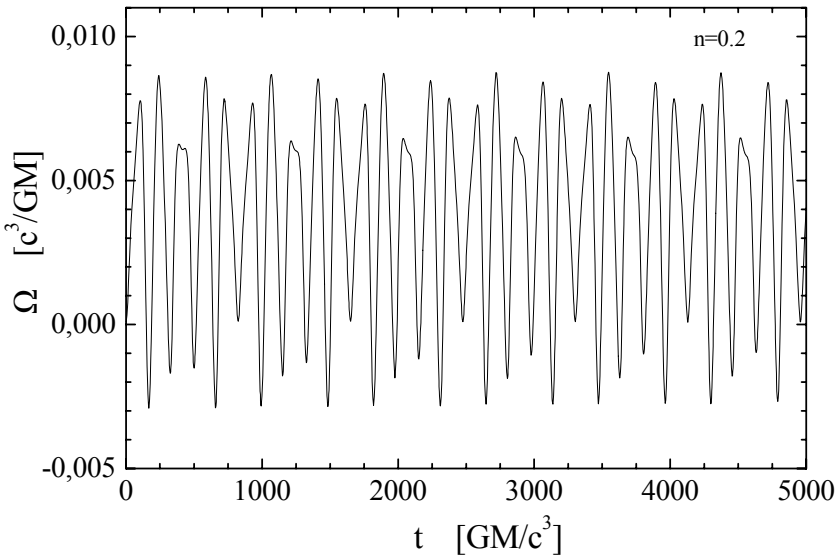
Σχήμα 4. 6. Η τελευταία φάση της χρονικής εξέλιξης συστημάτων μελανής οπής – αστέρα νετρονίων με διαφορετικούς πολυτροπικούς δείκτες.



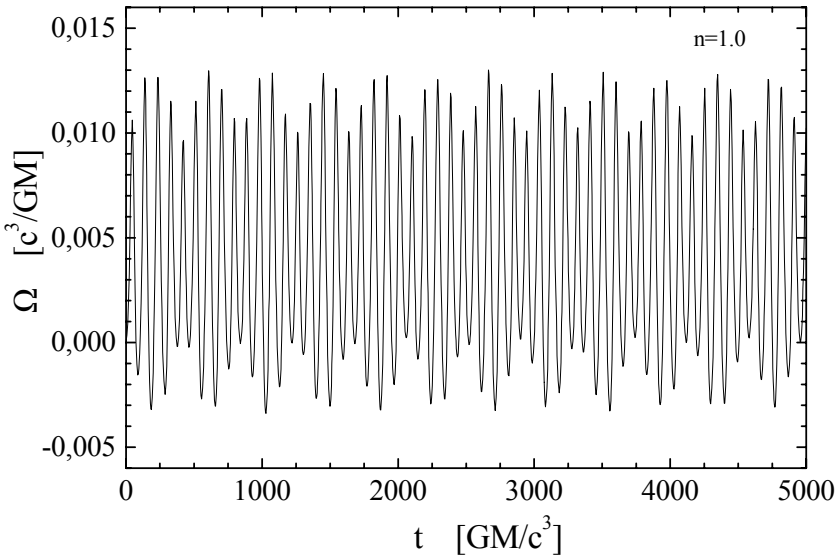
Σχήμα 4. 7. Ταλαντώσεις του άξονα a_1 αστέρων νετρονίων για δύο τιμές του πολυτροπικού δείκτη, $n = 0.2$ (πάνω) και $n = 1.0$ (κάτω).

Στο Σχήμα 4.7 φαίνονται οι ταλαντώσεις του άξονα a_1 του αστέρα νετρονίων για δύο διαφορετικούς πολυτροπικούς δείκτες, ενώ στα Σχήματα 4.8 και 4.9 βλέπουμε την γωνιακή ταχύτητα της ελλειψοειδούς μορφής του αστέρα για τις ίδιες τιμές των πολυτροπικών δεικτών. Οι ταλαντώσεις αυτές υπάρχουν γιατί οι αρχικές συνθήκες από τις οποίες ξεκινά η εξέλιξη των συστημάτων αποτελούν προσεγγιστικές και όχι ακριβής λύσεις ισορροπίας. Οι αρνητικές τιμές που λαμβάνει η

γωνιακή ταχύτητα της ελλειψοειδούς μορφής μπορούν να ερμηνευτούν αν λάβουμε υπόψη ότι $\dot{\gamma} \equiv \Omega = \dot{\phi} - \dot{\alpha}$ (Σχήμα 2.1). Η $\dot{\alpha}$ κατά την εξέλιξη του συστήματος παίρνει τόσο θετικές, όσο και αρνητικές τιμές. Το πλάτος των ταλαντώσεις αυτών καθορίζει αν η Ω θα λαμβάνει και αρνητικές τιμές.

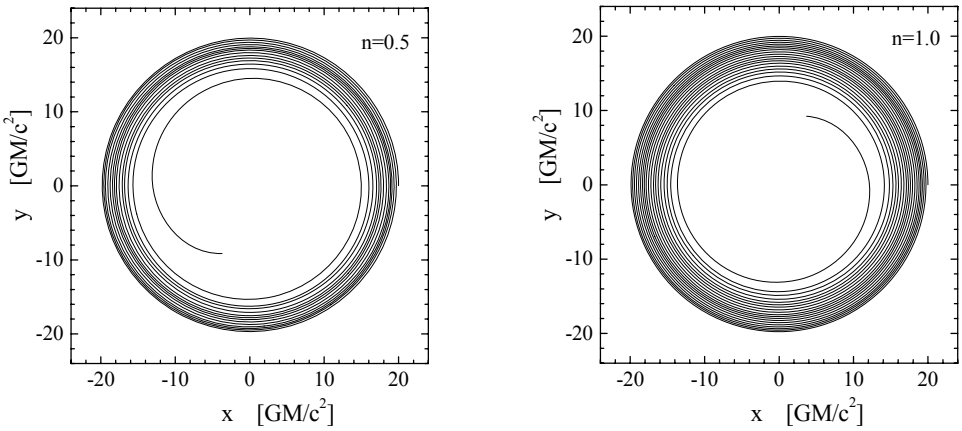


Σχήμα 4. 8. Οι μεταβολές της γωνιακής ταχύτητας της ελλειψοειδούς μορφής για αστέρα νετρονίων με πολυτροπικό δείκτη $n = 0.2$.



Σχήμα 4. 9. Οι μεταβολές της γωνιακής ταχύτητας της ελλειψοειδούς μορφής για αστέρα νετρονίων με πολυτροπικό δείκτη $n = 1.0$.

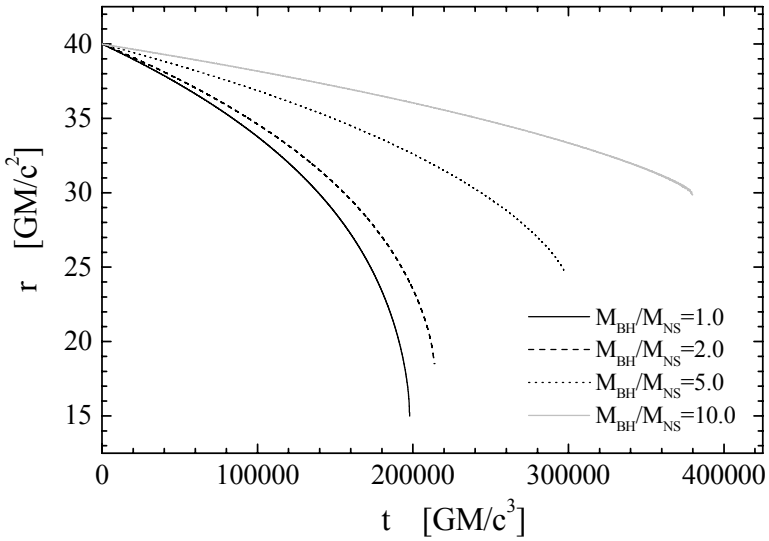
Τέλος, στο Σχήμα 4.10 βλέπουμε την εξέλιξη δύο συστημάτων με διαφορετικούς πολυτροπικούς δείκτες και με αρχικές συνθήκες πολύ κοντά στην περιοχή της κατάρρευσης. Παρατηρούμε, όπως και παραπάνω, ότι για την περίπτωση που ο πολυτροπικός δείκτης είναι $n = 1.0$ το σύστημα εκτελεί περισσότερους κύκλους πριν καταρρεύσει.



Σχήμα 4. 10. Η τελική φάση της εξέλιξης για δύο συστήματα μελανής οπής – αστέρα νετρονίων με διαφορετικούς πολυτροπικούς δείκτες.

ΕΠΙΔΡΑΣΗ ΤΟΥ ΛΟΓΟΥ ΤΩΝ ΜΑΖΩΝ

Στο Σχήμα 4.11 βλέπουμε την εξέλιξη συστημάτων μελανής οπής – αστέρα νετρονίων ($n = 0.5, R_0 = 6$) με διαφορετικό λόγο μαζών. Παρατηρούμε ότι όσο μεγαλύτερος είναι ο λόγος μαζών, τόσο πιο αργά πλησιάζουν οι αστέρες ο ένας τον άλλο, γεγονός που οφείλεται στο ότι η τροχιακή συνεισφορά

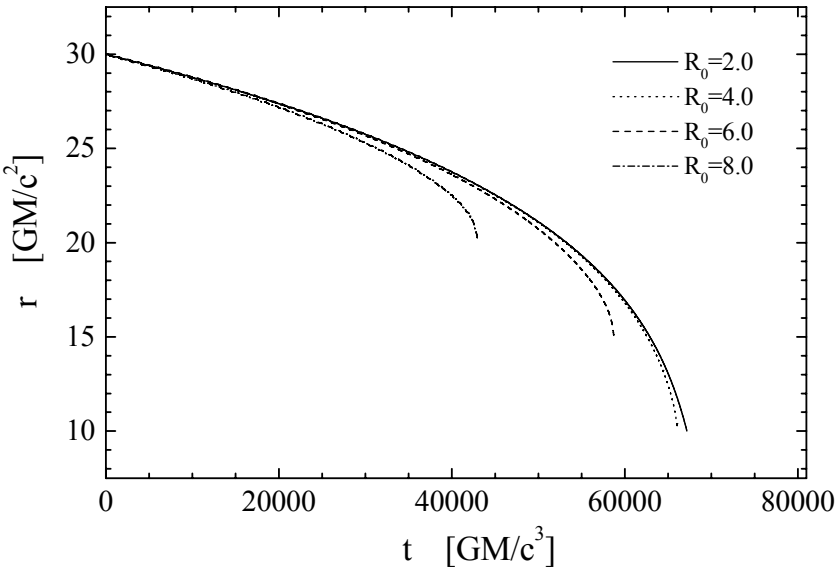


Σχήμα 4. 11. Χρονική εξέλιξη συστημάτων μελανής οπής – αστέρα νετρονίων με διαφορετικούς λόγους μαζών των συνοδών αστέρων.

στις $\mathfrak{T}_{ij}^{(5)}$ είναι ανάλογη της ανηγμένης μάζας, $\mu = M_1 M_2/(M_1 + M_2)$. Έτσι, όσο μεγαλύτερος[4] είναι λόγος μαζών τόσο μικρότερη θα είναι η ανηγμένη μάζα κι επομένως, μικρότερη και η συνεισφορά των $\mathfrak{T}_{ij}^{(5)}$ στην εξέλιξη του συστήματος. Επίσης η προσομοίωση σταματά σε μεγαλύτερη απομάκρυνση για μεγαλύτερο λόγο μαζών, γιατί οι παλιρροιογόνες δυνάμεις πάνω στον αστέρα νετρονίων είναι ισχυρότερες και επιταχύνουν την διάλυση του συστήματος.

ΕΠΙΔΡΑΣΗ ΤΗΣ ΑΚΤΙΝΑΣ ΤΟΥ ΑΣΤΕΡΑ ΝΕΤΡΟΝΙΩΝ

Στο Σχήμα 4.12 βλέπουμε την εξέλιξη συστημάτων ($n = 0.5, M_{BH}/M_{NS} = 1.0$) με αστέρες νετρονίων διαφορετικών ακτινών. Παρατηρούμε ότι για απομακρύνσεις μέχρι $\sim 4R_0$ η εξέλιξη των συστημάτων είναι παρόμοια, ενώ για μικρότερες απομακρύνσεις είναι αρκετά διαφορετική, αφού οι αστέρες βρίσκονται αρκετά κοντά ώστε να κάνουν την εμφάνισή τους παλιρροιογόνα φαινόμενα που θα οδηγήσουν τελικά το σύστημα σε κατάρρευση.

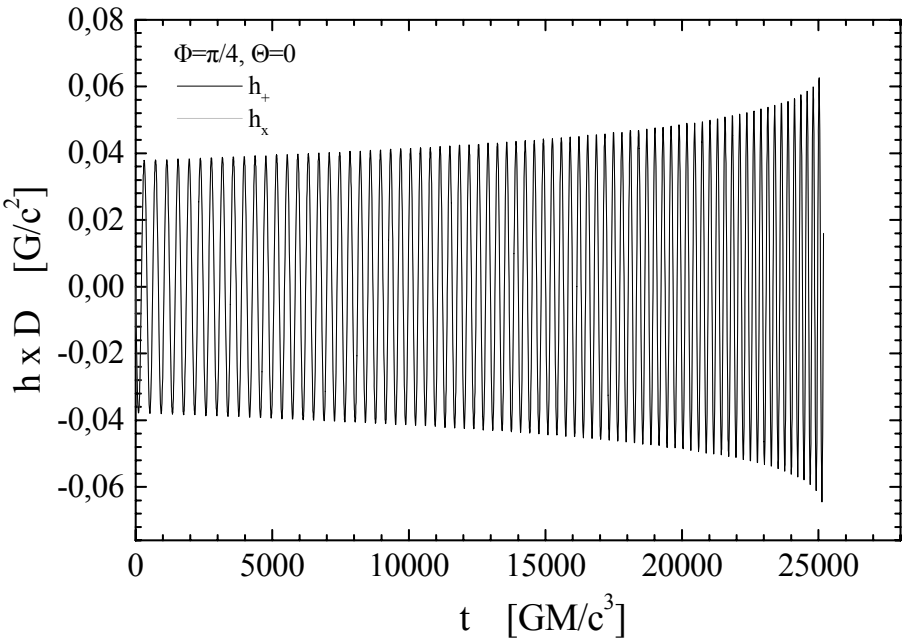


Σχήμα 4. 12. Χρονική εξέλιξη συστημάτων μελανής οπής – αστέρα νετρονίων για διαφορετικές ακτίνες του αστέρα νετρονίων.

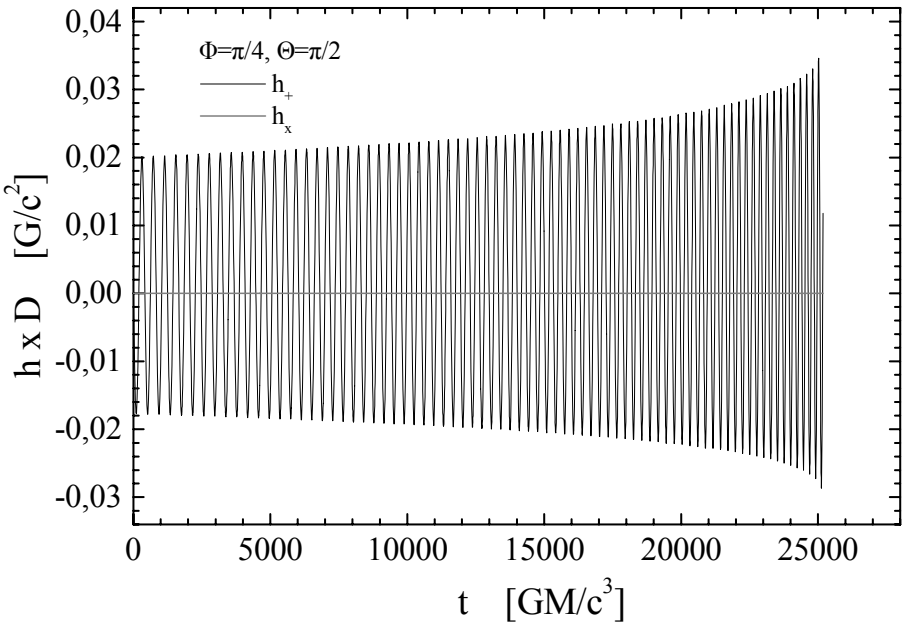
4.3 ΚΥΜΑΤΟΜΟΡΦΕΣ ΒΑΡΥΤΙΚΗΣ ΑΚΤΙΝΟΒΟΛΙΑΣ

Η παρατηρούμενη κυματομορφή ενός συστήματος που εκπέμπει βαρυτική ακτινοβολίας, εξαρτάται από τη σχετική θέση πηγής – παρατηρητή, που προσδιορίζεται από τις γωνίες Θ και Φ του σφαιρικού συστήματος συντεταγμένων. Η γωνία Φ εισάγει μια διαφορά στη φάση μεταξύ των δύο πολώσεων του βαρυτικού πεδίου, ενώ η επίδραση της Θ στις πολώσεις h_+ και $h_\times$ φαίνεται στα Σχήματα 4.13 και 4.14 που ακολουθούν. Επιλέγοντας $\Phi = \pi/4$, οπότε οι δύο κυματομοφές είναι σε συμφωνία φάσεως, διαπιστώνουμε ότι για $\Theta = 0$, οι κυματομορφές ταυτίζονται, ενώ για $\Theta = \pi/2$, η πόλωση $h_\times$ μηδενίζεται.

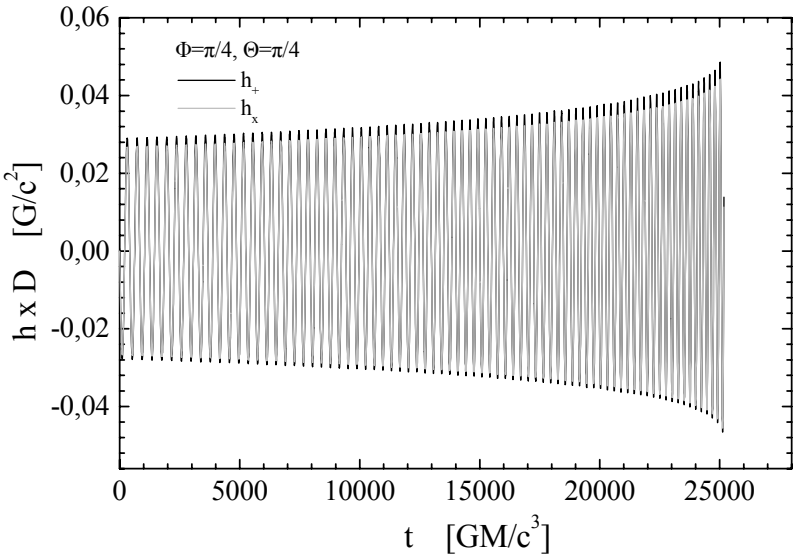
[4] Το ίδιο ισχύει και όταν ο λόγος μαζών είναι μικρότερος από τη μονάδα.



Σχήμα 4. 13. Οι δύο πολώσεις της βαρυτικής ακτινοβολίας, για $\theta = 0$ και $\Phi = \pi/4$.

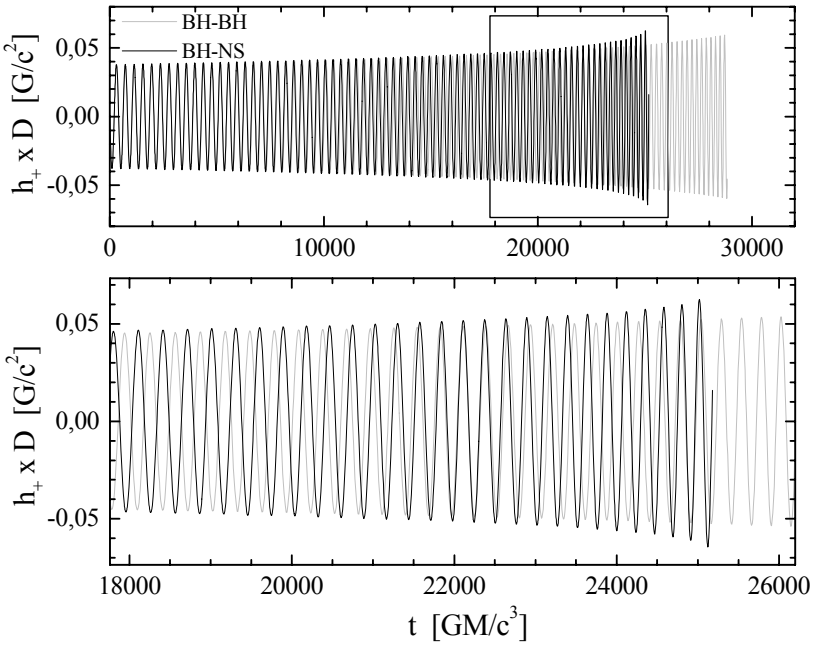


Σχήμα 4. 14. Οι δύο πολώσεις της βαρυτικής ακτινοβολίας, για $\theta = \pi/2$ και $\Phi = \pi/4$.

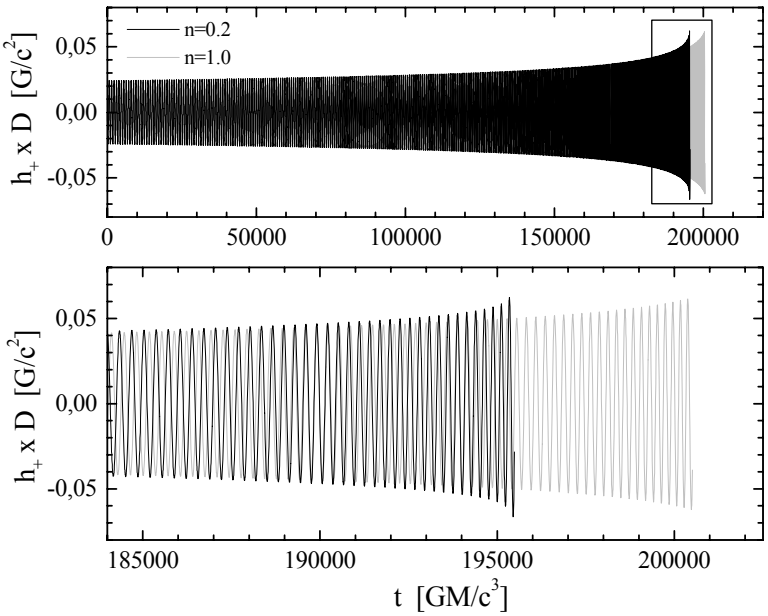


Σχήμα 4. 15. Οι δύο πολώσεις της βαρυτικής ακτινοβολίας, για $\boldsymbol{\Theta = \pi/4}$ και $\boldsymbol{\Phi = \pi/4}$.

Στο Σχήμα 4.16 συγκρίνουμε τις κυματομορφές της εκπεμπόμενης βαρυτικής ακτινοβολίας από ένα σύστημα δύο μελανών οπών και ένα σύστημα μελανής οπής – αστέρα νετρονίων. Και στις δύο περιπτώσεις ο λόγος μαζών είναι ίσος με τη μονάδα. Η διαφοροποίηση είναι εμφανής τόσο στο πλάτος, όσο και στη φάση των κυματομορφών κατά το τελικό στάδιο της εξέλιξης.

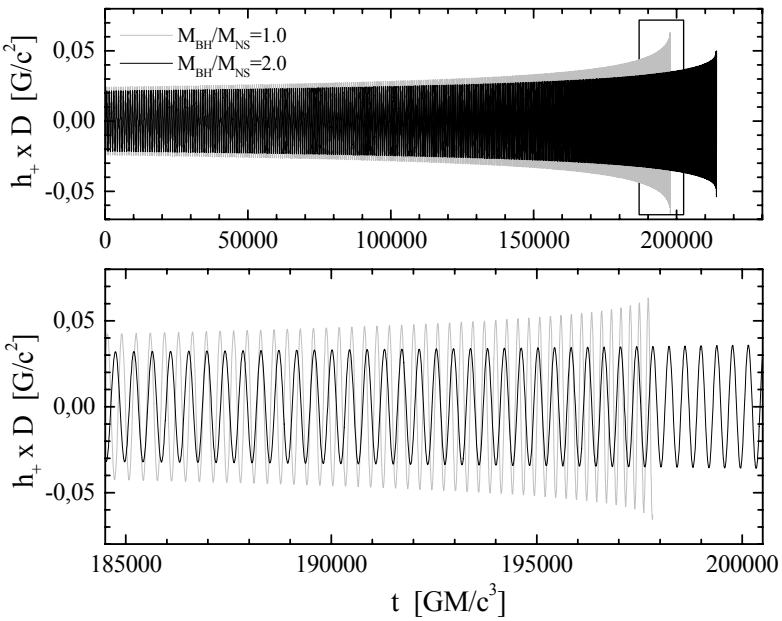


Σχήμα 4. 16. Κυματομορφές της εκπεμπόμενης βαρυτικής ακτινοβολίας από σύστημα δύο μελανών οπών και από σύστημα μελανής οπής – αστέρα νετρονίων (γωνίες παρατήρησης: $\boldsymbol{\Theta = 0, \Phi = \pi/4}$).



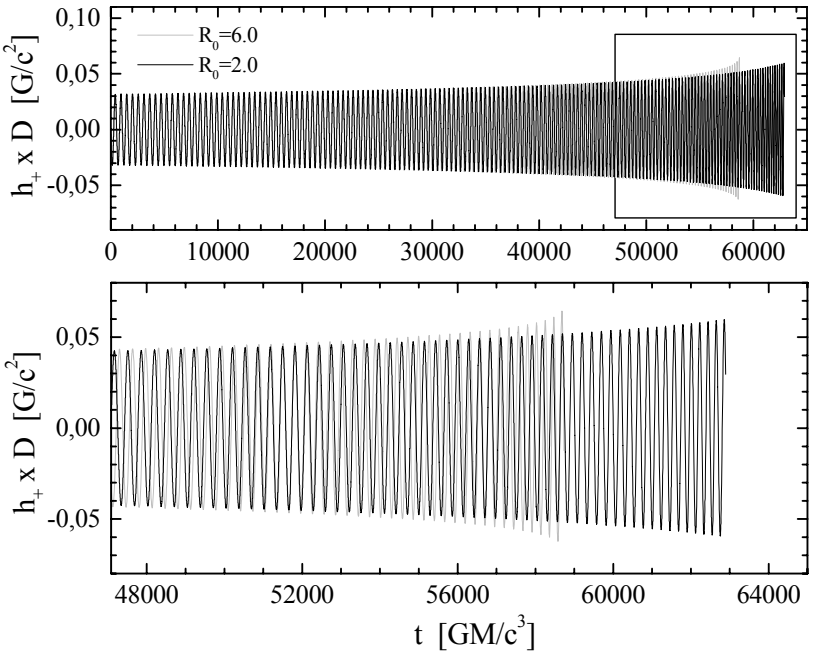
Σχήμα 4. 17. Κυματομορφές της εκπεμπόμενης βαρυτικής ακτινοβολίας από σύστημα μελανής οπής – αστέρα νετρονίων για διαφορετικές τιμές του πολυτροπικού δείκτη (γωνίες παρατήρησης: $\Theta = 0, \Phi = \pi/4$).

Στο Σχήμα 4.17 βλέπουμε τις κυματομορφές της εκπεμπόμενης βαρυτικής ακτινοβολίας από συστήματα μελανής οπής – αστέρα νετρονίων ($M_{BH}/M_{NS} = 1.0,\ R_0 = 6$), για διαφορετικές τιμές του πολυτροπικού δείκτη. Παρατηρούμε, ότι η κυματομορφή διαφοροποιείται κατά το τελικό στάδιο της εξέλιξης.



Σχήμα 4. 18. Κυματομορφές της εκπεμπόμενης βαρυτικής ακτινοβολίας από σύστημα μελανής οπής – αστέρα νετρονίων για διαφορετικούς λόγους μαζών των αστέρων (γωνίες παρατήρησης: $\Theta = 0, \Phi = \pi/4$).

Στο Σχήμα 4.18 βλέπουμε τις κυματομορφές της εκπεμπόμενης βαρυτικής ακτινοβολίας από συστήματα μελανής οπής – αστέρα νετρονίων ($n = 0.5,\ R_0 = 6$), για διαφορετικούς λόγους μαζών. Παρατηρούμε, ότι η κυματομορφή διαφέρει σ' όλη τη διάρκεια της εξέλιξης. Τέλος, στο Σχήμα 4.19 βλέπουμε τις κυματομορφές για διαφορετικές ακτίνες του αστέρα νετρονίων. Η διαφοροποίηση των κυματομορφών είναι εμφανής μόνο κατά το τελικό στάδιο της εξέλιξης.



Σχήμα 4. 19. Κυματομορφές της εκπεμπόμενης βαρυτικής ακτινοβολίας από σύστημα μελανής οπής – αστέρα νετρονίων για διαφορετικές ακτίνες των αστέρων νετρονίων (γωνίες παρατήρησης: $\boldsymbol{\Theta = 0, \Phi = \pi/4}$).

4.4 ΑΝΑΛΥΣΗ ΣΥΧΝΟΤΗΤΩΝ

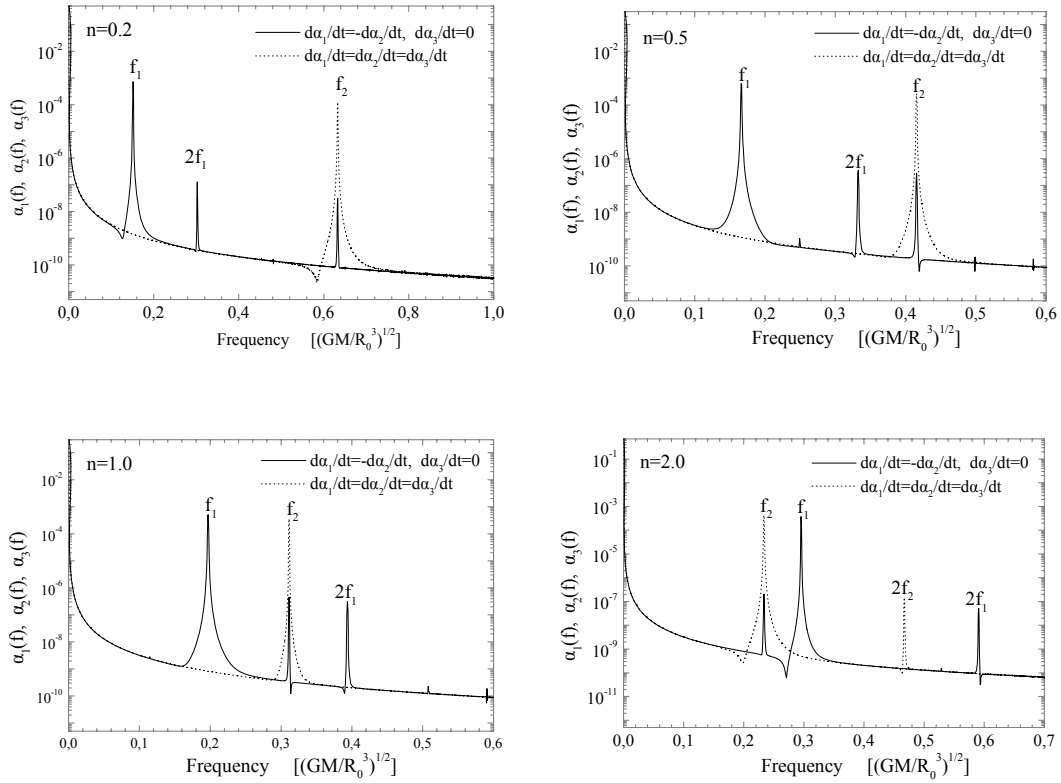
Η βασική συχνότητα των κυματομορφών της βαρυτικής ακτινοβολίας είναι η $2f_{orb}$, όπως εύκολα μπορεί να γίνει αντιληπτό παρατηρώντας την κυματομορφή. Ωστόσο, μια ανάλυση της κυματομορφής σε συχνότητες (με Fast Fourier Transform, FFT) είναι σε θέση να αναδείξει το πλήθος των συχνοτήτων που εμπεριέχονται σ' αυτή. Οι συχνότητες αυτές είναι συνδυασμοί της $2f_{orb}$ και των τρόπων δόνησης του αστέρα νετρονίων. Επομένως, για να μπορέσουμε να τις αναγνωρίσουμε θα πρέπει να είναι γνωστοί οι θεμελιώδεις τρόποι δόνησης των Riemann-S ελλειψοειδών, που αποτελούν την προσέγγιση για τους αστέρες νετρονίων στο μοντέλο που χρησιμοποιούμε.

ΤΡΟΠΟΙ ΔΟΝΗΣΗΣ ΤΩΝ RIEMANN-S ΕΛΛΕΙΨΟΕΙΔΩΝ

Η αναγνώριση των τρόπων δόνησης ενός αστέρα νετρονίων θα γίνει με την επίλυση των διαφορικών εξισώσεων που περιγράφουν την εξέλιξη ενός απομονωμένου αστέρα. Οι εξισώσεις αυτές είναι οι (2.51) – (2.55), στις οποίες θέτουμε $r \to \infty$. Με ανάλυση συχνοτήτων στις ταλαντώσεις των αξόνων του αστέρα, μπορούν να αναγνωριστούν κι εν μέρει να απομονωθούν οι θεμελιώδεις τρόποι δόνησης.

Οι θεμελιώδεις τρόποι δόνησης των Riemann-S ελλειψοειδών είναι δύο. Ο μεν πρώτος αντιστοιχεί σε συστολή – διαστολή των δύο αξόνων (a_1, a_2) εναλλάξ, με τον τρίτο άξονα (a_3) να μην ταλαντώνεται, ο δε δεύτερος αντιστοιχεί σε συστολή – διαστολή όλων των αξόνων ταυτοχρόνως. Τις συχνότητες αυτών των τρόπων δόνησης συμβολίζουμε με f_1 και f_2 αντίστοιχα.

Οι παραπάνω τρόποι δόνησης μπορούν να απομονωθούν με κατάλληλες αρχικές συνθήκες στις ταχύτητες των αξόνων. Έτσι, με $\dot{a}_1 = -\dot{a}_2,\ \dot{a}_3 = 0$, μπορούμε να απομονώσουμε (κατά προσέγγιση) τον f_1, ενώ με $\dot{a}_1 = \dot{a}_2 = \dot{a}_3$ μπορούμε να απομονώσουμε τον f_2. Στο Σχήμα 4.20 φαίνεται η ανάλυση συχνοτήτων (FFT) των ταλαντώσεων των αξόνων απομονωμένων αστέρων με διαφορετικούς πολυτροπικούς δείκτες, για τις αρχικές συνθήκες που αναφέραμε παραπάνω.



Σχήμα 4. 20. Οι θεμελιώδεις τρόποι δόνησης των Riemann-S ελλειψοειδών για διαφορετικές τιμές του πολυτροπικού δείκτη.

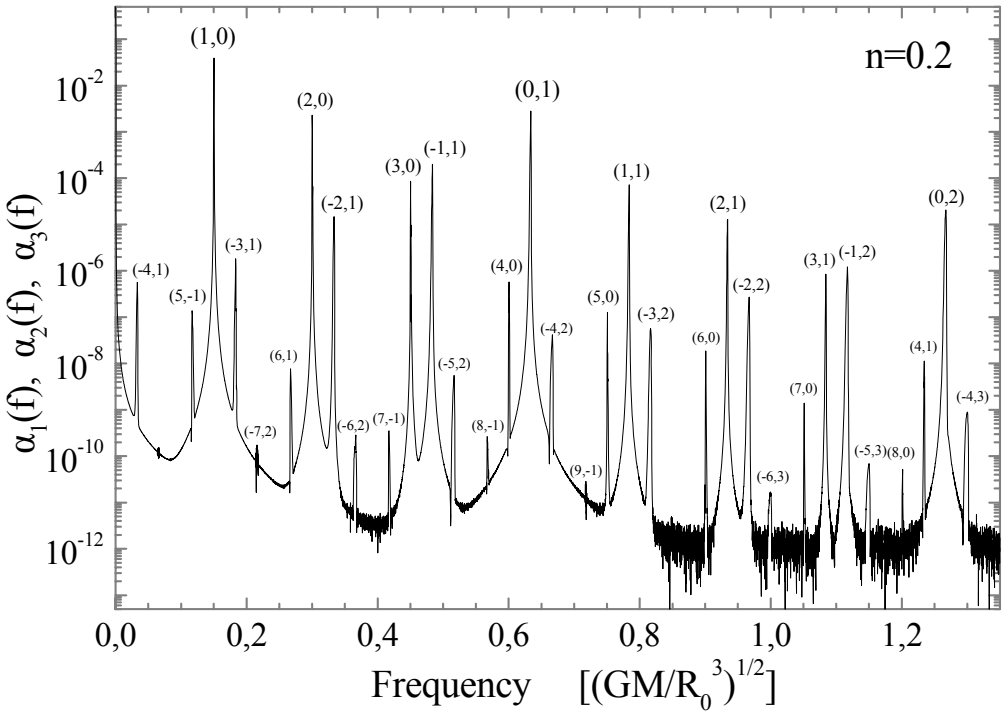
Από το Σχήμα 4.20 διαπιστώνουμε ότι για μικρές τιμές του πολυτροπικού δείκτη είναι $f_1 < f_2$, ενώ αντιθέτως, για μεγάλες τιμές είναι $f_1 > f_2$. Στον Πίνακα 4.1 παραθέτουμε τις ακριβής τιμές των συχνοτήτων f_1, f_2 για διάφορες τιμές του πολυτροπικού δείκτη. Παρατηρούμε ότι όσο μεγαλύτερος είναι ο πολυτροπικός δείκτης, τόσο μεγαλύτερη είναι η συχνότητα f_1, ενώ τόσο μικρότερη είναι η συχνότητα f_2.

Αν χρησιμοποιήσουμε τυχαίες αρχικές συνθήκες για τις ταχύτητες των αξόνων των αστέρων, τότε θα συνυπάρχουν και οι δύο τρόποι ταλάντωσης, καθώς επίσης και διάφοροι γραμμικοί συνδυασμοί των f_1 και f_2 λόγω μη-γραμμικής σύζευξης των δύο τρόπων ταλάντωσης. Στα Σχήματα 4.21, 4.22 και 4.23

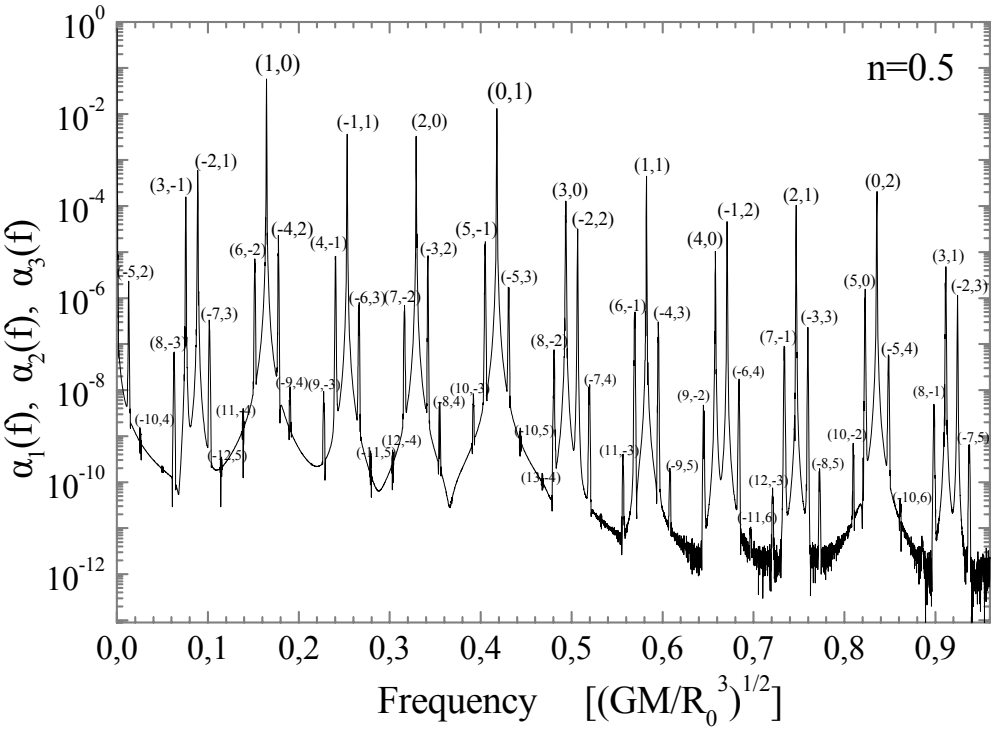
Συχνότητες των βασικών τρόπων δόνησης απομονωμένου αστέρα		
Πολυτροπικός Δείκτης	$f_1\ (\dot{a}_1 = -\dot{a}_2,\ \dot{a}_3 = 0)$	$f_2\ (\dot{a}_1 = \dot{a}_2 = \dot{a}_3)$
0.1	0.146484	0.882568
0.2	0.150146	0.633850
0.3	0.156250	0.523071
0.4	0.161133	0.458984
0.5	0.164489	0.417785
1.0	0.196533	0.308075
1.5	0.238037	0.266113
2.0	0.295410	0.233764
2.5	0.380859	0.190429

Πίνακας 4. 1. Συχνότητες των βασικών τρόπων ταλάντωσης ενός απομονωμένου, μη περιστρεφόμενου αστέρα, για διάφορες τιμές του πολυτροπικού δείκτη. Οι συχνότητες εκφράζονται σε μονάδες $\left(GM/R_0^3\right)^{1/2}$.

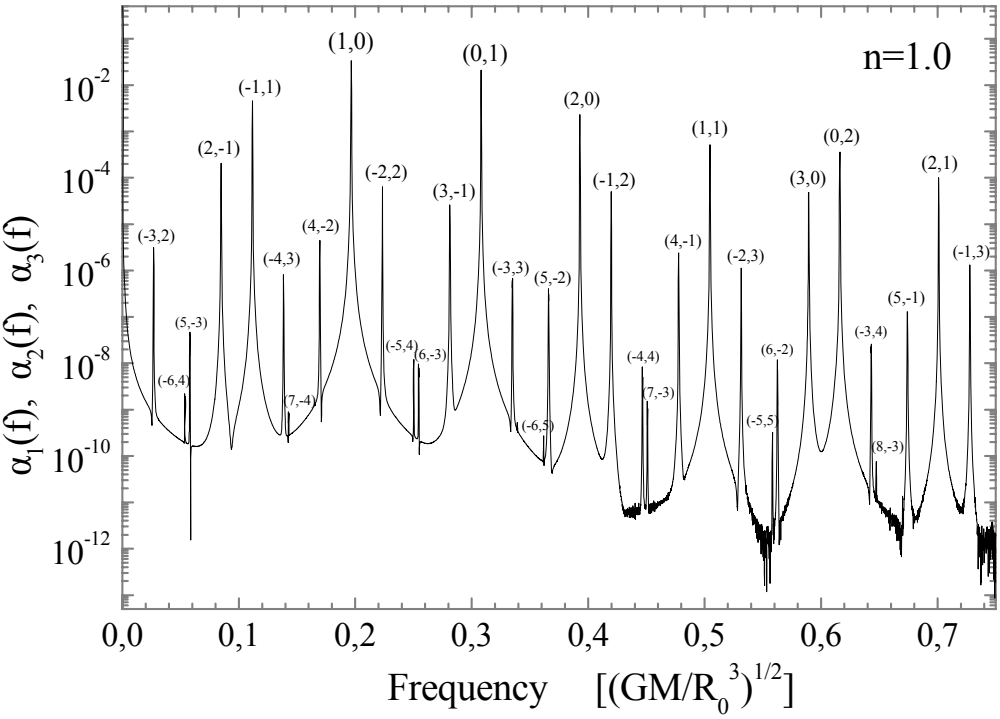
φαίνονται οι συχνότητες από τις οποίες συνίσταται η ταλάντωση των αξόνων των αστέρων, για τρεις διαφορετικές τιμές του πολυτροπικού δείκτη. Στα σχήματα αυτά έχουμε συμβολίσει τους γραμμικούς συνδυασμούς με $(x, y) = x.f_1 + y.f_2$. Οι ακριβείς αρχικές συνθήκες για την εξέλιξη του απομονωμένου αστέρα είναι $\dot{a}_1 = 0.125, \dot{a}_2 = 0.103, \dot{a}_3 = -0.05$ και $\Omega = \Lambda = 0$ (μη περιστρεφόμενος αστέρας). Οι ακριβής τιμές των συχνοτήτων που εμφανίζονται στα σχήματα, καθώς και το αντίστοιχο ζεύγος τιμών (x, y), παρατίθενται στους Πίνακες 4.2 – 4.4.



Σχήμα 4. 21. Διάγραμμα συχνοτήτων (FFT) των ταλαντώσεων των αξόνων ενός απομονωμένου, μη περιστρεφόμενου αστέρα με πολυτροπικό δείκτη $n = 0.2$.



Σχήμα 4. 22. Διάγραμμα συχνοτήτων (FFT) των ταλαντώσεων των αξόνων ενός απομονωμένου, μη περιστρεφόμενου αστέρα με πολυτροπικό δείκτη $n = 0.5$.



Σχήμα 4. 23. Διάγραμμα συχνοτήτων (FFT) των ταλαντώσεων των αξόνων ενός απομονωμένου, μη περιστρεφόμενου αστέρα με πολυτροπικό δείκτη $n = 1.0$.

$n = 0.2$			
Συχνότητα	$(x, y) = x.f_1 + y.f_2$	Συχνότητα	$(x, y) = x.f_1 + y.f_2$
0.033264	(-4,1)	0.717926	(9,-1)
0.116882	(5,-1)	0.750732	(5,0)
0.150146	**(1,0)**	0.783844	(1,1)
0.183410	(-3,1)	0.816498	(-3,2)
0.216064	(-7,2)	0.900879	(6,0)
0.267029	(6,-1)	0.934143	(2,1)
0.300293	(2,0)	0.967254	(-2,2)
0.333252	(-2,1)	0.999755	(-6,3)
0.366668	(-6,2)	1.051025	(7,0)
0.417175	(7,-1)	1.084137	(3,1)
0.450439	(3,0)	1.117401	(-1,2)
0.483703	(-1,1)	1.150055	(-5,3)
0.516662	(-5,2)	1.201172	(8,0)
0.567321	(8,-1)	1.234283	(4,1)
0.600586	(4,0)	1.267395	(0,2)
0.633850	**(0,1)**	1.300201	(-4,3)
0.666809	(-4,1)		

Πίνακας 4. 2. Συχνότητες των τρόπων ταλάντωσης ενός απομονωμένου, μη περιστρεφόμενου αστέρα πολυτροπικού δείκτη $n = 0.2$. Οι συχνότητες εκφράζονται σε μονάδες $\left(GM/R_0^3\right)^{1/2}$.

$n = 0.5$			
Συχνότητα	$(x, y) = x.f_1 + y.f_2$	Συχνότητα	$(x, y) = x.f_1 + y.f_2$
0.012970	(-5,2)	0.480651	(8,-2)
0.025329	(-10,4)	0.493774	(3,0)
0.063171	(8,-3)	0.506591	(-2,2)
0.075683	(3,-1)	0.519714	(-7,4)
0.088958	(-2,1)	0.556488	(11,-3)
0.101623	(-7,3)	0.569152	(6,-1)
0.114898	(-12,5)	0.582275	(1,1)
0.139160	(11,-4)	0.595550	(-4,3)
0.151825	(6,-2)	0.608062	(-9,5)
0.164489	**(1,0)**	0.644989	(9,-2)
0.177612	(-4,2)	0.658111	(4,0)
0.190734	(-9,4)	0.671081	(-1,2)
0.227355	(9,-3)	0.683746	(-6,4)
0.240173	(4,-1)	0.696563	(-11,6)
0.253295	(-1,1)	0.720977	(12,-3)
0.266418	(-6,3)	0.733795	(7,-1)
0.278930	(-11,5)	0.746917	(2,1)
0.303039	(12,-4)	0.759735	(-3,3)
0.316314	(7,-2)	0.772247	(-8,5)
0.329132	(2,0)	0.809631	(10,-2)
0.341949	(-3,2)	0.822906	(5,0)
0.355072	(-8,4)	0.835723	(0,2)
0.391998	(10,-3)	0.848388	(-5,4)
0.405120	(5,-1)	0.860748	(-10,6)
0.417785	**(0,1)**	0.898437	(8,-1)
0.430450	(-5,3)	0.911560	(3,1)
0.433572	(-10,5)	0.924377	(-2,3)
0.467529	(13,-4)	0.937042	(-7,5)

Πίνακας 4. 3. Συχνότητες των τρόπων ταλάντωσης ενός απομονωμένου, μη περιστρεφόμενου αστέρα πολυτροπικού δείκτη $n = 0.5$. Οι συχνότητες εκφράζονται σε μονάδες $\left(GM/R_0^3\right)^{1/2}$.

$n = 1.0$			
Συχνότητα	$(x,y) = x.f_1 + y.f_2$	Συχνότητα	$(x,y) = x.f_1 + y.f_2$
0.026855	(-3,2)	0.392991	(2,0)
0.053558	(-6,4)	0.419769	(-1,2)
0.057983	(5,-3)	0.446472	(-4,4)
0.084686	(2,-1)	0.450744	(7,-3)
0.111694	(-1,1)	0.477752	(4,-1)
0.138397	(-4,3)	0.504608	(1,1)
0.142517	(7,-4)	0.531311	(-2,3)
0.169525	(4,-2)	0.558319	(-5,5)
0.196533	**(1,0)**	0.562439	(6,-2)
0.223236	(-2,2)	0.589447	(3,0)
0.250244	(-5,4)	0.616149	(0,2)
0.254669	(6,-3)	0.643157	(-3,4)
0.281219	(3,-1)	0.647430	(8,-3)
0.308075	**(0,1)**	0.673980	(5,-1)
0.335083	(-3,3)	0.700988	(2,1)
0.361785	(-6,5)	0.727844	(-1,3)
0.365905	(5,-2)		

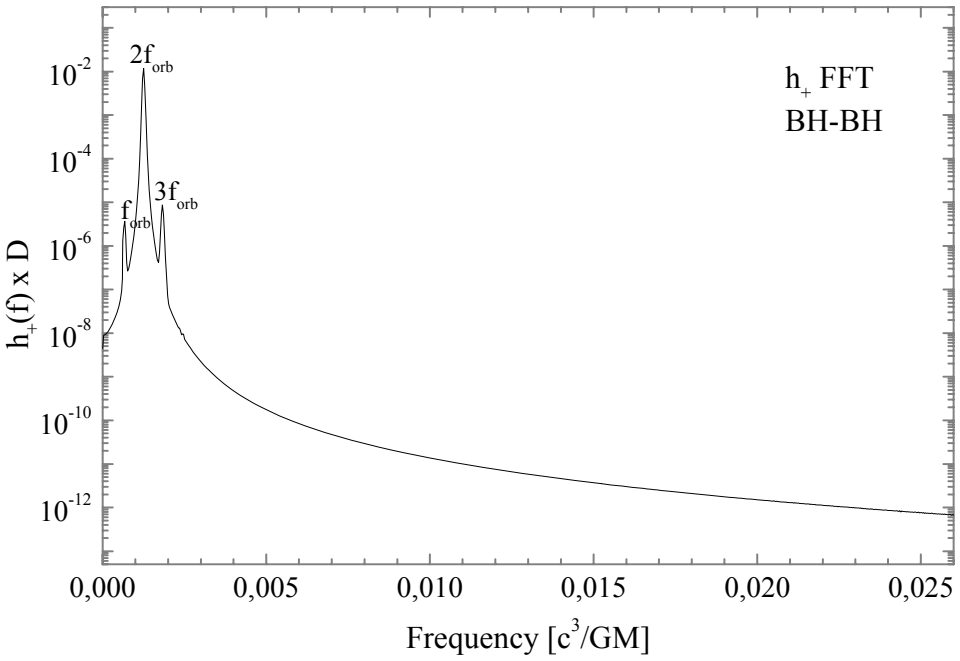
Πίνακας 4. 4. Συχνότητες των τρόπων ταλάντωσης ενός απομονωμένου, μη περιστρεφόμενου αστέρα πολυτροπικού δείκτη $n = 1.0$. Οι συχνότητες εκφράζονται σε μονάδες $\left(GM/R_0^3\right)^{1/2}$.

Οι παραπάνω συχνότητες αφορούν σε απομονωμένο και μη περιστρεφόμενο αστέρα. Στην περίπτωση κατά την οποία ο αστέρας βρίσκεται σε διπλό σύστημα και είναι περιστρεφόμενος, οι συχνότητες ταλάντωσης διαφοροποιούνται ελαφρώς, ωστόσο μπορούν εύκολα να αναγνωριστούν συγκρίνοντας με την περίπτωση του απομονωμένου αστέρα.

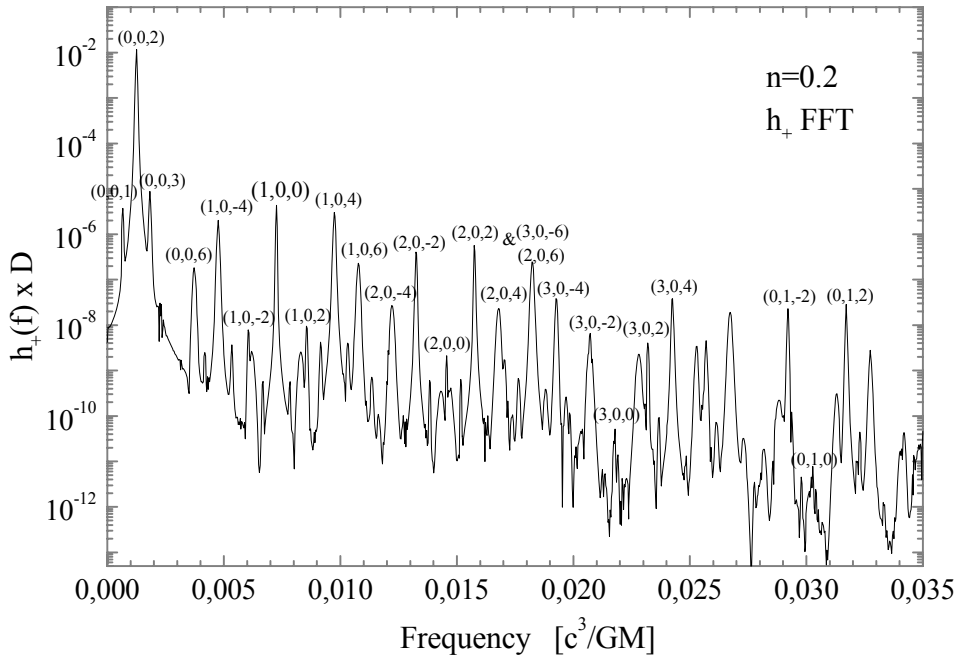
ΑΝΑΛΥΣΗ ΣΥΧΝΟΤΗΤΩΝ ΤΗΣ ΚΥΜΑΤΟΜΟΡΦΗΣ ΤΗΣ ΒΑΡΥΤΙΚΗΣ ΑΚΤΙΝΟΒΟΛΙΑΣ

Στην περίπτωση κατά την οποία ο αστέρας βρίσκεται σε διπλό σύστημα, εκτός των δύο βασικών συχνοτήτων που είδαμε παραπάνω έχουμε και την τροχιακή συχνότητα, f_{orb}. Έτσι, το σύνολο των συχνοτήτων θα αποτελείται από γραμμικούς συνδυασμούς των τριών βασικών συχνοτήτων. Παρακάτω, θα συμβολίσουμε με $(x,y,z) = x.f_1 + y.f_2 + z.f_{orb}$, τους γραμμικούς συνδυασμούς που θα αναγνωρίσουμε.

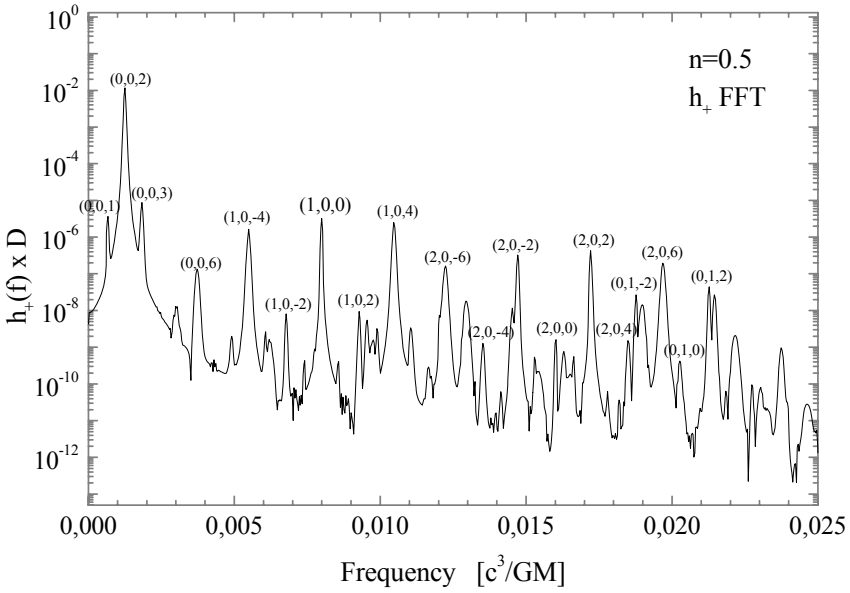
Στο Σχήμα 4.24 βλέπουμε το διάγραμμα συχνοτήτων της h_+ πόλωσης της βαρυτικής ακτινοβολίας ενός συστήματος δύο μελανών οπών. Παρατηρούμε ότι μόνο η τροχιακή συχνότητα, καθώς και κάποια πολλαπλάσιά της εμφανίζονται, αφού δεν λαμβάνονται υπόψη οι αστρικοί βαθμοί ελευθερίας στο σύστημα αυτό. Αντιθέτως, για συστήματα μελανής οπής – αστέρα νετρονίων όπου λαμβάνονται υπόψη οι αστρικοί βαθμοί ελευθερίας έχουμε περισσότερες συχνότητες όπως γίνεται εμφανές και στα Σχήματα 4.25 και 4.26. Η ύπαρξη των συχνοτήτων f_1 και f_2 στα σχήματα αυτά οφείλεται στις ταλαντώσεις των αξόνων του αστέρα νετρονίων. Οι ταλαντώσεις αυτές οφείλονται στις αρχικές συνθήκες με τις οποίες ξεκινά η εξέλιξη του συστήματος, οι οποίες όπως αναφέρθηκε και στην ενότητα 4.2 αποτελούν προσεγγιστικές λύσεις ισορροπίας.



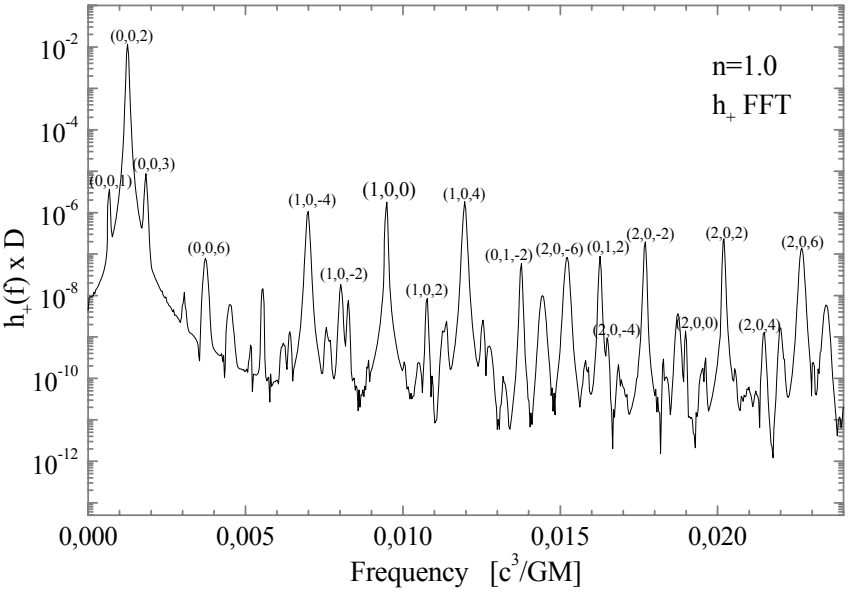
Σχήμα 4. 24. Ανάλυση συχνοτήτων (FFT) της h_+ πόλωσης της βαρυτικής ακτινοβολίας για σύστημα δύο μελανών οπών.



Σχήμα 4. 25. Ανάλυση συχνοτήτων (FFT) της h_+ πόλωσης της βαρυτικής ακτινοβολίας, ενός συστήματος μελανής οπής – αστέρα νετρονίων με πολυτροπικό δείκτη $n = 0.2$.



Σχήμα 4. 26. Ανάλυση συχνοτήτων (FFT) της h_+ πόλωσης της βαρυτικής ακτινοβολίας, ενός συστήματος μελανής οπής – αστέρα νετρονίων με πολυτροπικό δείκτη $n = 0.5$.



Σχήμα 4. 27. Ανάλυση συχνοτήτων (FFT) της h_+ πόλωσης της βαρυτικής ακτινοβολίας, ενός συστήματος μελανής οπής – αστέρα νετρονίων με πολυτροπικό δείκτη $n = 1.0$.

Από τα παραπάνω σχήματα βλέπουμε ότι η κύρια συχνότητα της κυματομορφής της βαρυτικής ακτινοβολίας είναι η $2f_{orb}$. Επιπλέον, εμφανίζεται κι ένα πλήθος δευτερευόντων συχνοτήτων ($f_{orb}, 3f_{orb}, f_1, f_1 \pm 4f_{orb}$ κ.τ.λ.). Παρατηρούμε ότι η συνεισφορά της f_2 στη διαμόρφωση της

κυματομορφής είναι πολύ μικρότερη σε σχέση με την f_1. Το γεγονός αυτό μπορεί να ερμηνευτεί αν ληφθεί υπόψη ότι η συχνότητα f_2 αντιστοιχεί στην ταυτόχρονη συστολή – διαστολή όλων των αξόνων, επομένως διατηρεί το ελλειψοειδές σχήμα του αστέρα και κατά συνέπεια η συνεισφορά της στην εκπομπή βαρυτικής ακτινοβολίας από το σύστημα θα είναι μικρή.

4.5 ΣΧΟΛΙΑ – ΣΥΜΠΕΡΑΣΜΑΤΑ

Είδαμε πως η πρώτη μετανευτώνεια διόρθωση εισάγει το φαινόμενο της μετατόπισης του περιάστρου και διαφοροποιεί σημαντικά την εξέλιξη ενός διπλού συστήματος. Αντιθέτως, η συνεισφορά των αστρικών βαθμών ελευθερίας γίνεται εμφανής μόνο κατά την τελική φάση της εξέλιξης του συστήματος. Και οι δύο παραπάνω όροι είναι απαραίτητοι για τον ακριβή προσδιορισμό της κυματομορφής της βαρυτικής ακτινοβολίας που εκπέμπει ένα διπλό σύστημα.

Μελετώντας την εξέλιξη διπλών συστημάτων είδαμε την εξάρτησή της από ορισμένες φυσικές παραμέτρους, όπως ο πολυτροπικός δείκτης, ο λόγος μαζών και η ακτίνα των αστέρων νετρονίων. Διαπιστώσαμε πως η τελική φάση της εξέλιξης ενός συστήματος είναι συντομότερη αν ο λόγος μαζών των αστέρων είναι κοντά στη μονάδα ή/και αν ο αστέρας νετρονίων χαρακτηρίζεται από μικρή τιμή του πολυτροπικού δείκτη και μεγάλη ακτίνα. Αντίστοιχα, συγκρίναμε τις κυματομορφές της εκπεμπόμενης βαρυτικής ακτινοβολίας για συστήματα με διαφορετικές φυσικές παραμέτρους. Τέλος, προσδιορίσαμε τους τρόπους δόνησης των αστέρων νετρονίων και αναγνωρίσαμε την ενδεχόμενη συνεισφορά τους στην κυματομορφή της εκπεμπόμενης βαρυτικής ακτινοβολίας.

ΑΝΑΦΟΡΕΣ

Andersson N. & Kokkotas K. D. (2003) *2nd Aegean Summer School on Early Universe.* Syros, Greece: Springer-Verlag.

Grishchuk L. P., Lipunov V. M., Postnov K. A., Prokhorov M. E. & B. S. Sathyaprakash (2001) *Physics-Uspekhi 44*, 1-51 (astro-ph/0008481).

Hansen D. (2006) *Gen.Rel.Grav. 38*, 1173-1208.

Hansen D., Voukantsis D., Stergioulas N. & Schäfer G. (2007) (in preparation).

Junker W. & Schäfer G. (1992) *Mon. Not. R. astr. Soc.* 254, 146-164.

Kokkotas K. D. & Schäfer G. (1995) *Mon. Not. R. astr. Soc.* 275, 301.

Lai D., Rasio F. A. & Shapiro S. L. (1993) *ApJ*, 88, 205-252.

Lai D., Rasio, F. A. & Shapiro S. L. (1994) *ApJ*, 423, 344-370.

Lai D., Rasio F. A., & Shapiro S. L. (1994) *ApJ*, 437, 742-769.

Lai D. & Shapiro, S. L. (1995) *ApJ*, 443, 705-716.

Richardson D. L. & Kelly T. J. (1988) *Celes. Mech. 43*, 193-210.

Sathyaprakash B. S. (2005) *Current Science Vol. 89, No. 12*.

Schutz B. F. (1999) *Class. Quantum Grav.* 16, A131-A156.

Spyrou N. K. & Kokkotas K. D. (1994) *ApJ*, 431, 254.

Thorne K. S. (1980) *Rev. Mod. Phys.* 52, 299.

Φωτογραφία εξώφυλλου: http://antwrp.gsfc.nasa.gov/apod/ap050601.html

ΠΑΡΑΡΤΗΜΑ "ΠΗΓΑΙΟΣ ΚΩΔΙΚΑΣ ΤΟΥ ΠΡΟΓΡΑΜΜΑΤΟΣ"

DriverProgram.cpp

```
#include"BinarySystem.h"


int main ()
{
    // initialization and evolution of BH-BH binary

    BinarySystem exampleBinary(BHBH);

    exampleBinary.SetSystem(1.0);
    exampleBinary.SetSeperation(40.0);

    exampleBinary.ReachEquilibrium();
    exampleBinary.SetInitialConditions(40.0,0.0,pow(10.0,-14.0),
            exampleBinary.GetEquilibriumValue(OmegaOrbital));

    exampleBinary.printBinaryProperties();
    //exampleBinary.Evolve(0.0,0.5,10,"BHBH.dat");
    exampleBinary.Evolve(0.0,0.01,pow(10.0,-8.0),10,"BHBH.dat");



    // initialization and evolution of BH-NS binary

    BinarySystem exampleBinary2(BHNS);

    exampleBinary2.SetSystem(1.0,6.0,1.0,0.65345);
    exampleBinary2.SetSeperation(40.0);
    exampleBinary2.SetFluidProperties(0.0001,0.000065);

    exampleBinary2.ReachEquilibrium();
    exampleBinary2.SetInitialConditions(
        exampleBinary2.GetEquilibriumValue(axis1),
        exampleBinary2.GetEquilibriumValue(axis2),
        exampleBinary2.GetEquilibriumValue(axis3),
        0.0,0.000065,40.0,0.0,0.00,0.00,0.00,0.0001,pow(10.0,-14.0),

    exampleBinary2.GetEquilibriumValue(OmegaOrbital));

    exampleBinary2.printBinaryProperties();
    //exampleBinary2.Evolve(0.0,0.5,10,"BHNS.dat");
    exampleBinary2.Evolve(0.0,0.01,pow(10.0,-8.0),10,"BHNS.dat");


    return(0);

}
```

BinarySystem.h

```
#include"Equations.h"
#include"NumericalMethods.h"

enum typeOfSystem {BHBH,BHNS};
enum equilibriumValues {axis1,axis2,axis3,OmegaOrbital};



class BinarySystem
{
 public:
        BinarySystem(const int typeOfSystem);
        ~BinarySystem();

        // set system properties
        void SetSystem(double massRatio);
        void SetSystem(double massRatio,double RoTrial,double n1,double kn1);
        void SetSeperation(double r);
        void SetFluidProperties(double Omega1, double Lamda1);
        void SetPhiThetaCap(double Theta,double Phi);

        // equilibrium
        void ReachEquilibrium(void);
        double GetEquilibriumValue(const int equilibriumValue);

        // initial conditions
        void SetInitialConditions(double r,double phiAngle,double rdot,
                                                        double OmegaOrbital);
        void SetInitialConditions(double axis1,double axis2,double axis3,
               double gammaAngle,double Lamda1,double r, double phiAngle,
                    double a1dot,double a2dot,double a3dot,double Omega1,
                                        double rdot,double OmegaOrbital);

        // get properties
        double GetRo(void);
        double GetHamiltonianRatio(void);

        // evolve system
        void Evolve(double time,double timeStep,double howManyPeriods,
                     const char* filename,int printScreenFrequency=100,
                     int printFileFrequency=1);
        void Evolve(double time,double trialStepSize,double accuracy,
                           double howManyPeriods,const char* filename,
               int printScreenFrequency=100,int printFileFrequency=1);

        // print properties
        void printBinaryProperties(void);


private:
        const int typeOfBinary; // choose BH-BH or BH-NS

        int nAlg;         // to store variables
        int nDif;
        double* xAlg;
        double* xDif;

        double M1;    // NS mass
        double M2;    // BH mass
        double Ro1;   // radius of spherical NS
        double n1;    // polytripic index
         double kn1;   // dimensionless coefficient (depent on n)

        double r;     // binary seperation

        double Omega1;        // frequency of the stars
        double L1;            // internal motion

        double ThetaCap; // angles to observe
        double PhiCap;   // h+ and hx polarizations

};
```

BinarySystem.cpp

```
#include"BinarySystem.h"


namespace {double pi=2.0*acos(0.0);}


BinarySystem::BinarySystem(const int typeOfSystemExt):
typeOfBinary(typeOfSystemExt)
{
    ThetaCap=0.0;
    PhiCap=pi/4.0;

    switch(typeOfBinary){
       case (BHBH):
              nAlg=1;
              nDif=4;

               xAlg=new double[nAlg];
               xDif=new double[nDif];

               xAlg[0]=0.1;

               for (int i=0;i<=nDif-1;i++)
                   xDif[i]=0.0;

               break;

        case (BHNS):
               nAlg=4;
               nDif=13;

               xAlg=new double[nAlg];
               xDif=new double[nDif];

               xAlg[0]=1.0;
               xAlg[1]=1.0;
               xAlg[2]=1.0;
               xAlg[3]=0.1;

               for (int i=0;i<=nDif-1;i++)
                    xDif[i]=0.0;

               break;

        default:
               std::cout<<"Error! Unknown Type of Binary System !"<<std::endl;

        }

}


void BinarySystem::SetSystem(double massRatioExt)
{
    M1=1./(massRatioExt+1);
    M2=massRatioExt/(massRatioExt+1);
    Equations::SetSystem(M1,M2,0.0,0.0,0.0);
}


void BinarySystem::SetSystem(double massRatioExt,double RoTrialExt,
                             double n1Ext,double kn1Ext)
{
    M1=1./(massRatioExt+1);
    M2=massRatioExt/(massRatioExt+1);
    Ro1=RoTrialExt;
    n1=n1Ext;
    kn1=kn1Ext;
    Equations::SetSystem(M1,M2,Ro1,n1,kn1);
}
```

```
void BinarySystem::SetSeperation(double rExt)
{
    r=rExt;
    Equations::SetSeperation(r);
}


void BinarySystem::SetFluidProperties(double Omega1Ext,double L1Ext)
{
    Omega1=Omega1Ext;
    L1=L1Ext;
    Equations::SetFluidProperties(Omega1,L1);
}


void BinarySystem::SetPhiThetaCap(double Theta, double Phi)
{
    ThetaCap=Theta;
    PhiCap=Phi;
}



void BinarySystem::ReachEquilibrium()
{
    if (typeOfBinary==BHBH){
        NumericalMethods::NewtonRaphson myEquilibriumBinary(nAlg,
        Equations::algEquatBHBH1PN,Equations::algDerivBHBH1PN);
        myEquilibriumBinary.Solve(xAlg,pow(10.,-12.),NumericalMethods::LU);
    }

    else if (typeOfBinary==BHNS){
        NumericalMethods::NewtonRaphson myEquilibriumBinary(nAlg,
        Equations::algEquatBHNS1PN,Equations::algDerivBHNS1PN);
        myEquilibriumBinary.Solve(xAlg,pow(10.,-12.),NumericalMethods::LU);
    }

    else
        std::cout<<"Error in BinarySystem::ReachEquilibrium(),\
                    Unknown Type of Binary"<<std::endl;

}



double BinarySystem::GetEquilibriumValue(const int equilibriumValue)
{
    double temp=0.0;

    if (typeOfBinary==BHBH){
        if (equilibriumValue==OmegaOrbital)
            temp=xAlg[0];

         else
            std::cout<<"Error in BinarySystem::GetEquilibriumValue,\
                        No such variable in BHBH system"<<std::endl;
    }

    else if (typeOfBinary==BHNS){
        if (equilibriumValue==axis1)
            temp=xAlg[0];
        else if (equilibriumValue==axis2)
            temp=xAlg[1];

        else if (equilibriumValue==axis3)
            temp=xAlg[2];

        else if (equilibriumValue==OmegaOrbital)
            temp=xAlg[3];

        else
            std::cout<<"Error in BinarySystem::GetEquilibriumValue,\
                        No such variable in BHNS system"<<std::endl;
    }

    return(temp);
}
```

```
void BinarySystem::SetInitialConditions(double rExt,double phiAngleExt,
                                        double rdotExt,double OmegaOrbitalExt)
{
    xDif[0]=rExt;              // r
    xDif[1]=phiAngleExt;       // phi
    xDif[2]=rdotExt;           // rdot
    xDif[3]=OmegaOrbitalExt;   // phidot

}


void BinarySystem::SetInitialConditions(double axis1Ext,double axis2Ext,
      double axis3Ext,double gammaAngleExt,double Lamda1Ext,double rExt,
      double phiAngleExt,double a1dotExt,double a2dotExt,double a3dotExt,
      double Omega1Ext,double rdotExt,double OmegaOrbitalExt)

{
    xDif[0]=axis1Ext;           // a1
    xDif[1]=axis2Ext;           // a2
    xDif[2]=axis3Ext;           // a3
    xDif[3]=gammaAngleExt;      // gamma angle
    xDif[4]=Lamda1Ext;          // lamda
    xDif[5]=rExt;               // r
    xDif[6]=phiAngleExt;        // phi angle
    xDif[7]=a1dotExt;           // a1dot
    xDif[8]=a2dotExt;           // a2dot
    xDif[9]=a3dotExt;           // a3dot
    xDif[10]=Omega1Ext;         // omega1
    xDif[11]=rdotExt;           // rdot
    xDif[12]=OmegaOrbitalExt;   // phidot

    Ro1=BinarySystem::GetRo();
    Equations::SetSystem(M1,M2,Ro1,n1,kn1);
}


double BinarySystem::GetRo()
{
    return(Equations::CalculateRo(n1,xDif));
}


double BinarySystem::GetHamiltonianRatio()
{
    return(Equations::CalculateHamiltonians(xDif));
}



void BinarySystem::Evolve(double time,double timeStep,double howManyPeriods,
               const char* filename,int printScreenFrequency,int printFileFrequency)
{
    std::ofstream outfile;
    outfile.open(filename,std::ios::out);

    double phiEnd=2*pi*howManyPeriods;
    double Omega1Temp=0.0;
    double OmegaOrbTemp=0.0;
    double Omega1dot=0.0;
    double OmegaOrbdot=0.0;
    double hPlus=0.0;
    double hCross=0.0;
    double timeTemp=0.0;



    if (typeOfBinary==BHBH){
       NumericalMethods::RK4 myrk(nDif,Equations::difEquationsBHBH,
                                  time,xDif,timeStep);
```

```
        for(int i=0;xDif[1]<phiEnd;i++){
            myrk.Solve(1);
            Equations::CalculatehPlusCrossBHBH(&hPlus,&hCross,xDif,pi/4.0,0.0);
            if(i%printScreenFrequency==0)
                std::cout<<myrk.GetX()<<"\t"<<std::setprecision(10)<<xDif[0]\
                         <<"\t"<<xDif[1]<<std::endl;
             if(i%printFileFrequency==0){
                outfile<<myrk.GetX();
                for(int j=0;j<nDif;j++)
                    outfile<<"\t"<<std::setprecision(10)<<xDif[j];
                outfile<<"\t"<<hPlus<<"\t"<<hCross;
                outfile<<std::endl;
             }

            if(xDif[0]<8)
               break;
        }
    }


    else if (typeOfBinary==BHNS){

        NumericalMethods::RK4 myrk(nDif,Equations::difEquationsBHNS,
                                   time,xDif,timeStep);

        for(int i=0;xDif[6]<phiEnd;i++){

            Omega1Temp=xDif[10];
            OmegaOrbTemp=xDif[12];
            timeTemp=myrk.GetX();

            myrk.Solve(1);
            Equations::CalculatehPlusCrossBHNS(&hPlus,&hCross,xDif,
                                               PhiCap,ThetaCap);
            if(i%printScreenFrequency==0)// print in screen every 100 steps

                std::cout<<myrk.GetX()<<"\t"<<std::setprecision(10)<<xDif[5]\
                         <<"\t"<<xDif[0]<<"\t"<<xDif[6]<<std::endl;

             if(i%printFileFrequency==0){
                outfile<<myrk.GetX();
                for(int j=0;j<nDif;j++)
                    outfile<<"\t"<<std::setprecision(10)<<xDif[j];
                outfile<<"\t"<<hPlus<<"\t"<<hCross;
                outfile<<std::endl;
            }

            Omega1dot=(xDif[10]-Omega1Temp)/(myrk.GetX()-timeTemp);
            OmegaOrbdot=(xDif[12]-OmegaOrbTemp)/(myrk.GetX()-timeTemp);

             Equations::SetOmegadot(Omega1dot,OmegaOrbdot);
        }

    }

    outfile.close();

}



void BinarySystem::Evolve(double time,double trialStepSize,
double accuracy,double howManyPeriods,const char* filename,
           int printScreenFrequency,int printFileFrequency)

{
    std::ofstream outfile;
    outfile.open(filename,std::ios::out);

    double phiEnd=2*pi*howManyPeriods;
    double Omega1Temp=0.0;
    double OmegaOrbTemp=0.0;
    double Omega1dot=0.0;
    double OmegaOrbdot=0.0;
    double hPlus=0.0;
    double hCross=0.0;
    double timeTemp=0.0;
```

```
    if (typeOfBinary==BHBH){
        NumericalMethods::RK4AdaptStep myrk(nDif,Equations::difEquationsBHBH,
                                            time,xDif,trialStepSize,accuracy);
        for(int i=0;xDif[1]<phiEnd;i++){

            myrk.Solve(1);
            Equations::CalculatehPlusCrossBHBH(&hPlus,&hCross,xDif,
                                               PhiCap,ThetaCap);

            if(i%printScreenFrequency==0)// print in screen every 100 steps
               std::cout<<myrk.GetX()<<"\t"<<std::setprecision(10)\
                        <<xDif[0]<<"\t"<<xDif[1]<<std::endl;

             if(i%printFileFrequency==0){
               outfile<<myrk.GetX();
               for(int j=0;j<nDif;j++)
                   outfile<<"\t"<<std::setprecision(10)<<xDif[j];
               outfile<<"\t"<<hPlus<<"\t"<<hCross;
               outfile<<std::endl;
             }

         }

    }


    else if (typeOfBinary==BHNS){

        NumericalMethods::RK4AdaptStep myrk(nDif,Equations::difEquationsBHNS,
                                            time,xDif,trialStepSize,accuracy);

        for(int i=0;xDif[6]<phiEnd;i++){

            Omega1Temp=xDif[10];
            OmegaOrbTemp=xDif[12];
            timeTemp=myrk.GetX();

            myrk.Solve(1);
            Equations::CalculatehPlusCrossBHNS(&hPlus,&hCross,xDif,
                                               PhiCap,ThetaCap);
            if(i%printScreenFrequency==0)// print in screen every 100 steps
               std::cout<<myrk.GetX()<<"\t"<<std::setprecision(10)\
                        <<xDif[5]<<"\t"<<xDif[0]<<"\t"<<xDif[6]<<std::endl;

             if(i%printFileFrequency==0){
               outfile<<myrk.GetX();
               for(int j=0;j<nDif;j++)
                    outfile<<"\t"<<std::setprecision(10)<<xDif[j];
               outfile<<"\t"<<hPlus<<"\t"<<hCross;
               outfile<<std::endl;
             }

            Omega1dot=(xDif[10]-Omega1Temp)/(myrk.GetX()-timeTemp);
            OmegaOrbdot=(xDif[12]-OmegaOrbTemp)/(myrk.GetX()-timeTemp);

            Equations::SetOmegadot(Omega1dot,OmegaOrbdot);
        }

    }

    outfile.close();

}


void BinarySystem::printBinaryProperties()
{
    if (typeOfBinary==BHBH){
        std::cout<<std::endl<<std::endl;
        std::cout<<"-------------------------------------------------"<<std::endl;
        std::cout<<" PHYSICAL PROPERTIES (G=c=Mtotal=1)"<<std::endl;
        std::cout<<"-------------------------------------------------"<<std::endl;
        std::cout<<" Type of System    :BH-BH Binary"<<std::endl;
        std::cout<<" Mass Ratio (M2/M1):"<<M2/M1<<std::endl;
        std::cout<<" Seperation        :"<<r<<std::endl;
        std::cout<<"-------------------------------------------------"<<std::endl;
```

```
        std::cout<<" EQUILIBRIUM VALUES"<<std::endl;
        std::cout<<"-----------------------------------------------"<<std::endl;
        std::cout<<" Orbital Angular Frequency:"<<xAlg[0]<<std::endl;
        std::cout<<"-----------------------------------------------"<<std::endl;
        std::cout<<" INITIAL CONDITIONS"<<std::endl;
        std::cout<<"-----------------------------------------------"<<std::endl;
       std::cout<<" r     :"<<xDif[0]<<std::endl;
        std::cout<<" phi   :"<<xDif[1]<<std::endl;
        std::cout<<" rdot  :"<<xDif[2]<<std::endl;
        std::cout<<" phidot:"<<xDif[3]<<std::endl;
        std::cout<<"-----------------------------------------------"<<std::endl;
        std::cout<<std::endl<<std::endl;
        std::cout<<"Press enter to continue..."<<std::endl;
        std::cin.get();

    }


    else if (typeOfBinary==BHNS){
        std::cout<<std::endl<<std::endl;
        std::cout<<"-----------------------------------------------"<<std::endl;
        std::cout<<" PHYSICAL PROPERTIES (G=c=Mtotal=1)"<<std::endl;
        std::cout<<"-----------------------------------------------"<<std::endl;
        std::cout<<" Type of System    :BH-NS Binary"<<std::endl;
        std::cout<<" Mass Ratio (M2/M1):"<<M2/M1<<std::endl;
        std::cout<<" Seperation        :"<<r<<std::endl;
        std::cout<<" Polytropic Index  :"<<n1<<std::endl;
        std::cout<<" Omega1            :"<<Omega1<<std::endl;
        std::cout<<" Lamda1            :"<<L1<<std::endl;
        std::cout<<"-----------------------------------------------"<<std::endl;
        std::cout<<" EQUILIBRIUM VALUES"<<std::endl;
        std::cout<<"-----------------------------------------------"<<std::endl;
        std::cout<<" Axis 1                  :"<<xAlg[0]<<std::endl;
        std::cout<<" Axis 2                  :"<<xAlg[1]<<std::endl;
        std::cout<<" Axis 3                  :"<<xAlg[2]<<std::endl;
        std::cout<<" Orbital Angular Frequency:"<<xAlg[3]<<std::endl;
        std::cout<<"-----------------------------------------------"<<std::endl;
        std::cout<<" INITIAL CONDITIONS"<<std::endl;
        std::cout<<"-----------------------------------------------"<<std::endl;
         std::cout<<" a1        :"<<xDif[0]<<std::endl;
        std::cout<<" a2        :"<<xDif[1]<<std::endl;
        std::cout<<" a3        :"<<xDif[2]<<std::endl;
        std::cout<<" gammaAngle:"<<xDif[3]<<std::endl;
        std::cout<<" Lamda1    :"<<xDif[4]<<std::endl;
        std::cout<<" r         :"<<xDif[5]<<std::endl;
        std::cout<<" phiAngle  :"<<xDif[6]<<std::endl;
        std::cout<<" a1dot     :"<<xDif[7]<<std::endl;
        std::cout<<" a2dot     :"<<xDif[8]<<std::endl;
        std::cout<<" a3dot     :"<<xDif[9]<<std::endl;
        std::cout<<" Omega1    :"<<xDif[10]<<std::endl;
        std::cout<<" rdot      :"<<xDif[11]<<std::endl;
        std::cout<<" phidot    :"<<xDif[12]<<std::endl;
        std::cout<<"-----------------------------------------------"<<std::endl;
        std::cout<<" OTHER PROPERTIES"<<std::endl;
        std::cout<<"-----------------------------------------------"<<std::endl;
        std::cout<<" Horb/Hstar:"<<BinarySystem::GetHamiltonianRatio()<<std::endl;
        std::cout<<" Ro        :"<<Ro1<<std::endl;
        std::cout<<"-----------------------------------------------"<<std::endl;
        std::cout<<std::endl<<std::endl;
        std::cout<<"Press enter to continue..."<<std::endl;
        std::cin.get();

    }

}


BinarySystem::~BinarySystem()
{
    delete[] xAlg;
    delete[] xDif;
}
```

Numericalmethods.h

```
#include<iostream>
#include<fstream>
#include<iomanip>
#include<cmath>
#include<vector>


// 1. LinearSystem
// 2. NewtonRaphson
// 3. ChandraIntegrals
// 4. RK4 (Runge-Kutta 4th order)
// 5. RK4AdaptStep (Runge-Kutta 4th order with adaptive step size)


namespace NumericalMethods
{


    enum Solver {Gauss,LU};



// TYPEDEFINITIONS ***************************************************************


    // for Newton-Raphson
    typedef void (*AlgEquations)(double* x,double* fx);
    typedef void (*AlgDerivatives)(double* x,double** dfdx);


    // for Runge-Kutta Methods
    typedef void (*ODEs)(const double x,double* y,double* dydt);




//***  LINEAR SYSTEM  ***********************************************************

        // solves a linear system of the form A.X=B

    class LinearSystem
    {
     public:
            LinearSystem(int numberOfEquations);
            ~LinearSystem();

            // Gauss Elemenation
            void GaussSolve(double **A,double *B,double *X);

            // LU decomposition with partial (implicit) pivoting
            void LUSolve(double** A,double* B,double* X);

     private:

            // used in LUSolve (can't be used from main())

            // transform A into L.U
            void LUDecomposition(double** a,double* d,int* indx);

            // solves the system
            void LUDecSolve(double** a,double* b,int* indx);

            int n;      // number of equations

       };
```

```
//****  NEWTON RAPHSON  ************************************************************

    // solves an algebric system of equations

    class NewtonRaphson
    {
     public:
            NewtonRaphson(int numberOfEquations,const AlgEquations,
                          const AlgDerivatives);
            ~NewtonRaphson();

            void OneNRStep(double* InitialValues,const int Solver=Gauss);
            void Solve(double* initialValues,double accuracy,
                       const int Solver=Gauss);

     private:
             void ChangeSign(void);

             int n;        // number of equations
             double* dx;   // corrections
             double* b;    // right hand side
             double* temp; // used to change the sing of f
             double** J;   // jacobian matrix

             NumericalMethods::LinearSystem* localSystem; // linearized system
             AlgEquations myEquations;     // non linear algebraic equations
             AlgDerivatives myDerivatives; // first derivatives of equations
                                           // over all the variables

       };




//******  CHANDRA INTEGRALS  *****************************************************

// this class calculates the integrals A1,A2,...
// using the Euler-MacLauren formula

    class ChandraIntegrals
    {
     private:
             double a1Loc;    // local variables to switch between A1,A2,...
             double a2Loc;
             double a3Loc;
             double norm;     // normalization factor

     public:
            ChandraIntegrals(const double NormalizationFactor);  // constructor
             ~ChandraIntegrals();                                 // destructor

             double f(double u);
             double df(double u);   // function and derivatives for
             double df3(double u);  // the value u
             double df5(double u);

             // calculates only one integral (numberOfIntegral)
             double Calculate(int numberOfIntegral,double lowerLimit,
                    double upperLimit,double step,double* radiiArray);

             // calculates all the integrals
             void CalculateAll(double lowerLimit,double upperLimit,double step,
                               double* radiiArray,double* IntegralsArray);


       };
```

```
//*******  RUNGE-KUTTA 4TH ORDER  ***********************************************

// solves a set of ODEs by using Runge-Kutta of 4th order

    class RK4
    {
     public:
             RK4(int numberOfEquations,const ODEs,double independentVariables,
                 double* dependentVariables,double stepSize);
            ~RK4();

            // return values of x every "howManySteps" steps
            void Solve(int howManySteps=1);

            // access to x (y,dydx are pointers can be accessed by main)
            double GetX(void){return(x);}

      private :

             double* k1;
            double* k2;   // Runge-Kutta variables
            double* k3;
            double* k4;
            double* yTemp;

            int n;        // number of equations
            double h;     // step size
            double h2;    // h/2
            double h6;    // h/6
            double x;     // independent variable (e.g. time)
            double* y;    // vector of dependent variables

            ODEs myEquations;  // equations to be solved

     };



//******  RUNGE-KUTTA 4TH ORDER ADAPTIVE STEP SIZE  ******************************

// solves a set of ODEs by using Runge-Kutta 4th order with adaptive step size

    class RK4AdaptStep
    {
     public:
            RK4AdaptStep(int numberOfEquations,const ODEs,
             double independentVariables,double* dependentVariables,
                        double TrialStepSize,double desiredAccuracy);
            ~RK4AdaptStep();

            void Solve(int howManySteps=1);
            void OneRKStep(void);

            // accessors
            double GetX(void){return(x);}
            double GetYErr(int whichVariable){return (yErr[whichVariable]);}

            // simple functions
            double GetMax(double a,double b){return(a>b ? a : b);}
            double GetMin(double a,double b){return(a<b ? a : b);}

      private :
            double* k1;
            double* k2;
            double* k3;      // Runge-Kutta variables
            double* k4;
            double* k5;
            double* k6;
            double* yTemp;  // temporary storage of variables
            double* yErr;   // matrix of absolute errors
            double* yScal;  // yScal=|y|+h*|dy/dx|
            double* yOut;   // to update y after one iteration
```

```
            int n;          // number of equations
            double h;       // step size
            double hTemp;   // temporary step size during calculations
            double errMax;  // maximum error (within the n variables)
            double epsilon; //desired accuracy (relative accuracy)
            double x;       // independent variable (e.g. time)
            double* y;      // vector of dependent variables

            ODEs myEquations; // equations to be solved

        } ;


}// end of namespace
```

LinearSystem.cpp

```
#include"NumericalMethods.h"


namespace // local constant
{
    double tiny=pow(10.0,-20.0);
}


NumericalMethods::LinearSystem::LinearSystem(int nExt):
n(nExt)
{
}


void NumericalMethods::LinearSystem::GaussSolve(double** A,double* B,double* X)
{
    int i,j,k;
    double mult,sum;

    for (i=0;i<n-1;i++){// forward composition
        for (j=i+1;j<n;j++){
             mult=(A[j][i]/A[i][i]);
            for (k=i;k<n;k++){
               A[j][k]=A[j][k]-mult*A[i][k];
             }
             B[j]=B[j]-mult*B[i];
         }
    }

    for (i=n-1;i>=0;i--){// back substitution
        sum= B[i];
        for (k=i+1;k<n;k++){
             sum=sum - X[k]*A[i][k];
        }
        X[i]=sum/(A[i][i]);
    }
}



void NumericalMethods::LinearSystem::LUSolve(double** a,double* b,double* x)
{
    double permutations=1.0;
    int* indx=new int[n]; // stores changes in rows taken by pivoting


    // create L.U form of A and solve the system
    LinearSystem::LUDecomposition(a,&permutations,indx);
    LinearSystem::LUDecSolve(a,b,indx);
```

```
    for(int i=0;i<=n-1;i++)
        x[i]=b[i];

    delete [] indx;
}


void NumericalMethods::LinearSystem::LUDecomposition(double** a,double* d,int* indx)
{
    // from numerical recipes

    int i,imax,j,k;
    double big,dum,sum,temp;
    double *vv=new double[n];

    *d=1.0;

    for (i=1;i<=n;i++) {
         big=0.0;

        for (j=1;j<=n;j++)
            if ((temp=fabs(a[i-1][j-1])) > big)
                big=temp;
            if (big == 0.0){
               std::cout<<"Singular Matrix in LinearSystem->\
                           LUDecomposition()"<<std::endl;
                exit(1);
            }
            vv[i-1]=1.0/big;
        }

        for (j=1;j<=n;j++) {
             for (i=1;i<j;i++) {
                  sum=a[i-1][j-1];
                  for (k=1;k<i;k++)
                       sum -= a[i-1][k-1]*a[k-1][j-1];
                  a[i-1][j-1]=sum;
                }

         big=0.0;

        for (i=j;i<=n;i++) {
            sum=a[i-1][j-1];
            for (k=1;k<j;k++)
                 sum -= a[i-1][k-1]*a[k-1][j-1];
            a[i-1][j-1]=sum;

             if ( (dum=vv[i-1]*fabs(sum)) >= big) {
                 big=dum;
                 imax=i;
              }
         }

         if (j != imax) {
            for (k=1;k<=n;k++) {
                 dum=a[imax-1][k-1];
                 a[imax-1][k-1]=a[j-1][k-1];
                 a[j-1][k-1]=dum;
            }

            *d = -(*d);
            vv[imax-1]=vv[j-1];
          }

          indx[j-1]=imax;
          if (a[j-1][j-1] == 0.0)
             a[j-1][j-1]=tiny;

          if (j != n) {
             dum=1.0/(a[j-1][j-1]);
             for (i=j+1;i<=n;i++)
                  a[i-1][j-1] *= dum;
          }
    }

    delete [] vv;
}
```

```
void NumericalMethods::LinearSystem::LUDecSolve(double** a,double* b,int* indx)
{
    // from numerical recipes

    int i,ii=0,ip,j;
    double sum;
    for (i=1;i<=n;i++) {
        ip=indx[i-1];
        sum=b[ip-1];
        b[ip-1]=b[i-1];
        if (ii)
            for (j=ii;j<=i-1;j++)
                sum -= a[i-1][j-1]*b[j-1];
        else if (sum)
               ii=i;
        b[i-1]=sum;
    }


    for (i=n;i>=1;i--) {
        sum=b[i-1];
        for (j=i+1;j<=n;j++)
            sum -= a[i-1][j-1]*b[j-1];
        b[i-1]=sum/a[i-1][i-1];
        }
}


NumericalMethods::LinearSystem::~LinearSystem()
{

}
```

NewtonRaphson.cpp

```
#include "NumericalMethods.h"


namespace // local constant
{
    int maxIter=50000;
}


NumericalMethods::NewtonRaphson::NewtonRaphson(int nExt,
const AlgEquations myEquationsExt,const AlgDerivatives myDerivativesExt):
n(nExt),
myEquations(myEquationsExt),
myDerivatives(myDerivativesExt)
{
    // memory allocation
    dx=new double [n];
    b=new double [n];
    temp=new double[n];
    J=new double* [n];
    for(int i=0;i<=n-1;i++)
        J[i]=new double[n];

    // initialization
    for(int i=0;i<=n-1;i++){
        dx[i]=0.0;
        b[i]=0.0;
        for(int j=0;j<=n-1;j++)
            J[i][j]=0.0;
    }

        localSystem=new LinearSystem(n);
}
```

```
void NumericalMethods::NewtonRaphson::Solve(double* x,double accuracy,
                                            const int Solver)
{
    double totalAcc=n*accuracy;
    double sum=0.0;

    for(int k=1;k<maxIter;k++){// Newton-Raphson method

        sum=0.0;
        NumericalMethods::NewtonRaphson::OneNRStep(x,Solver);

        for(int i=0;i<=n-1;i++)
            sum+=fabs(dx[i]);

        // check for accuracy criterion, avoid small numbers at the
        // first steps
        if(sum<totalAcc && k>5){
            std::cout<<"procedure ended after "\
                     <<k<<" iterations"<<std::endl;
            break;
        }

    }// end of loop over k

}


void NumericalMethods::NewtonRaphson::OneNRStep(double* x,const int Solver)
{

    // create linearized system
    NumericalMethods::NewtonRaphson::myEquations(x,b);
    NumericalMethods::NewtonRaphson::myDerivatives(x,J);
    NumericalMethods::NewtonRaphson::ChangeSign();


    // calculate corrections through Gauss or LU
    if(Solver==Gauss)
        localSystem->GaussSolve(J,b,dx);

    else if (Solver==LU)
        localSystem->LUSolve(J,b,dx);

    // update variables
    for(int i=0;i<=n-1;i++)
        x[i]+=dx[i];
}


void NumericalMethods::NewtonRaphson::ChangeSign(void)
{
    for(int i=0;i<n;i++){
        temp[i]=b[i];
        b[i]=-temp[i];
    }

}


NumericalMethods::NewtonRaphson::~NewtonRaphson()
{
    for(int i=0;i<=n-1;i++)
        delete [] J[i];

    delete [] J;       // delete the array of pointers
    delete [] b;
    delete [] dx;
    delete [] temp;
    delete localSystem;

}
```

ChandraIntegrals.cpp

```
#include "NumericalMethods.h"


// constructor (gets the normalization factor)
NumericalMethods::ChandraIntegrals::ChandraIntegrals(const double normExt):
norm(normExt),
a1Loc(0.0),
a2Loc(0.0),
a3Loc(0.0)
{
    // no action!
}


// function for integration
double NumericalMethods::ChandraIntegrals::f(double u)
{
    return( 1./((a1Loc*a1Loc + u)*sqrt((a1Loc*a1Loc + u)*
           (a2Loc*a2Loc + u)*(a3Loc*a3Loc + u))) );
}


// first derivative of f over u
double NumericalMethods::ChandraIntegrals::df(double u)
{
    return( ((a2Loc*a2Loc)*(-1.5*(a3Loc*a3Loc) - 2.0*u) + (a1Loc*a1Loc)*
          (-0.5*(a2Loc*a2Loc) - 0.5*(a3Loc*a3Loc) - u) + (-2.0*(a3Loc*a3Loc)
          - 2.5*u)*u)/(((a1Loc*a1Loc) + u)*pow(((a1Loc*a1Loc)+ u)*
          ((a2Loc*a2Loc) + u)*((a3Loc*a3Loc) + u),1.5)) );
}


// third derivative of f over u
double NumericalMethods::ChandraIntegrals::df3(double u)
{
    return( (-6.*pow((a2Loc*a2Loc) + u,3.0)*pow((a3Loc*a3Loc) + u,3.0) -
            3.*pow((a2Loc*a2Loc) + u,2.0)*pow((a3Loc*a3Loc) + u,2.0)*
            (((a1Loc*a1Loc) + u)*((a2Loc*a2Loc) + u) + ((a1Loc*a1Loc) + u)
            *((a3Loc*a3Loc) + u) + ((a2Loc*a2Loc) + u)*((a3Loc*a3Loc) + u))
            -0.75*((a2Loc*a2Loc) + u)*((a3Loc*a3Loc) + u)*(-4*((a1Loc*a1Loc)
            + u)*((a2Loc*a2Loc) + u)*((a3Loc*a3Loc) + u)*((a1Loc*a1Loc) +
            (a2Loc*a2Loc) + (a3Loc*a3Loc) + 3.0*u)+3.0*pow(((a1Loc*a1Loc) +
            u)*((a2Loc*a2Loc) + u) + ((a1Loc*a1Loc) + u)*((a3Loc*a3Loc) + u)
            +((a2Loc*a2Loc) + u)*((a3Loc*a3Loc) + u),2.0)) + 0.375*(-8*
            pow((a1Loc*a1Loc) + u,2.0)*pow((a2Loc*a2Loc) + u,2.0)*
            pow((a3Loc*a3Loc) + u,2.0) + 12.0*((a1Loc*a1Loc) + u)*
            ((a2Loc*a2Loc) + u)*((a3Loc*a3Loc) + u)*((a1Loc*a1Loc) +
            (a2Loc*a2Loc) + (a3Loc*a3Loc) + 3.0*u)*(((a1Loc*a1Loc) + u)*
            ((a2Loc*a2Loc) + u) + ((a1Loc*a1Loc) + u)*((a3Loc*a3Loc) + u) +
            ((a2Loc*a2Loc) + u)*((a3Loc*a3Loc) + u)) - 5.0*pow(((a1Loc*a1Loc)
            + u)*((a2Loc*a2Loc) + u) + ((a1Loc*a1Loc) + u)*((a3Loc*a3Loc) + u)
            +((a2Loc*a2Loc) + u)*((a3Loc*a3Loc) + u),3.0)))/(((a1Loc*a1Loc)+
            u)*pow(((a1Loc*a1Loc) + u)*((a2Loc*a2Loc) + u)*((a3Loc*a3Loc) +
            u),3.5)) );
}


// fifth derivative of f over u
double NumericalMethods::ChandraIntegrals::df5(double u)
{
    return( (-120.*pow((a2Loc*a2Loc) + u,5.0)*pow((a3Loc*a3Loc) + u,5.0) -
            60.*pow((a2Loc*a2Loc) + u,4.0)*pow((a3Loc*a3Loc) + u,4.0)*
            (((a1Loc*a1Loc) + u)*((a2Loc*a2Loc) + u) + ((a1Loc*a1Loc) + u)*
            ((a3Loc*a3Loc) + u) + ((a2Loc*a2Loc) + u)*((a3Loc*a3Loc) + u)) -
            15.*pow((a2Loc*a2Loc) + u,3.0)*pow((a3Loc*a3Loc) + u,3.0)*
            (-4.0*((a1Loc*a1Loc) + u)*((a2Loc*a2Loc) + u)*((a3Loc*a3Loc) + u)*
            ((a1Loc*a1Loc) + (a2Loc*a2Loc) + (a3Loc*a3Loc) + 3.0*u) + 3.0*
            pow(((a1Loc*a1Loc) + u)*((a2Loc*a2Loc) + u) + ((a1Loc*a1Loc) + u)*
            ((a3Loc*a3Loc) + u)+((a2Loc*a2Loc) + u)*((a3Loc*a3Loc) + u),2.0))
            +7.5*pow((a2Loc*a2Loc) + u,2.0)*pow((a3Loc*a3Loc) + u,2.0)*(-8*
            pow((a1Loc*a1Loc) + u,2.0)*pow((a2Loc*a2Loc) + u,2.0)*
            pow((a3Loc*a3Loc) + u,2.0) + 12.0*((a1Loc*a1Loc) + u)*((a2Loc*a2Loc)
```

```
            + u)*((a3Loc*a3Loc)+u)*((a1Loc*a1Loc)+(a2Loc*a2Loc)+(a3Loc*a3Loc)
            + 3.0*u)*(((a1Loc*a1Loc) + u)*((a2Loc*a2Loc) + u) + ((a1Loc*a1Loc)
            + u)*((a3Loc*a3Loc) + u)+((a2Loc*a2Loc) + u)*((a3Loc*a3Loc) + u))-
            5.0*pow(((a1Loc*a1Loc) + u)*((a2Loc*a2Loc) + u) + ((a1Loc*a1Loc)
            + u)*((a3Loc*a3Loc) + u)+((a2Loc*a2Loc) + u)*((a3Loc*a3Loc)+u),3.0))
            -0.9375*((a2Loc*a2Loc) + u)*((a3Loc*a3Loc) + u)*(48*pow((a1Loc*a1Loc)
            + u,2.0)*pow((a2Loc*a2Loc) + u,2.0)*pow((a3Loc*a3Loc) + u,2.0)*
            pow((a1Loc*a1Loc) + (a2Loc*a2Loc) + (a3Loc*a3Loc) + 3.0*u,2.0) +
            96.0*pow((a1Loc*a1Loc) + u,2.0)*pow((a2Loc*a2Loc) + u,2.0)*
            pow((a3Loc*a3Loc) + u,2.0)*(((a1Loc*a1Loc) + u)*((a2Loc*a2Loc) + u)
            + ((a1Loc*a1Loc) + u)*((a3Loc*a3Loc) + u) +((a2Loc*a2Loc) + u)*
            ((a3Loc*a3Loc) + u)) - 120.0*((a1Loc*a1Loc) + u)*((a2Loc*a2Loc) +
            u)*((a3Loc*a3Loc) + u)*((a1Loc*a1Loc) + (a2Loc*a2Loc) + (a3Loc*a3Loc)
            + 3.0*u)*pow(((a1Loc*a1Loc) + u)*((a2Loc*a2Loc) + u) + ((a1Loc*a1Loc)
            + u)*((a3Loc*a3Loc) + u) +((a2Loc*a2Loc) + u)*((a3Loc*a3Loc) +
            u),2.0) + 35.0*pow(((a1Loc*a1Loc) + u)*((a2Loc*a2Loc) + u) +
            ((a1Loc*a1Loc) + u)*((a3Loc*a3Loc) + u) + ((a2Loc*a2Loc) + u)*
            ((a3Loc*a3Loc) + u),4.0)) + 0.46875*(192.0*pow((a1Loc*a1Loc) + u,3.0)
            *pow((a2Loc*a2Loc) + u,3.0)*pow((a3Loc*a3Loc) + u,3.0)*((a1Loc*a1Loc)
            + (a2Loc*a2Loc) + (a3Loc*a3Loc) + 3*u) - 240.0*pow((a1Loc*a1Loc) +
            u,2.0)*pow((a2Loc*a2Loc) + u,2.0)*pow((a3Loc*a3Loc) + u,2.0)*
            pow((a1Loc*a1Loc) + (a2Loc*a2Loc) + (a3Loc*a3Loc) + 3.0*u,2.0)*
            (((a1Loc*a1Loc) + u)*((a2Loc*a2Loc) + u) + ((a1Loc*a1Loc) + u)*
            ((a3Loc*a3Loc) + u) + ((a2Loc*a2Loc) + u)*((a3Loc*a3Loc) + u)) -
            240.0*pow((a1Loc*a1Loc) + u,2.0)*pow((a2Loc*a2Loc) + u,2.0)*
            pow((a3Loc*a3Loc) + u,2.0)*pow(((a1Loc*a1Loc) + u)*((a2Loc*a2Loc) + u)
            + ((a1Loc*a1Loc) + u)*((a3Loc*a3Loc) + u) + ((a2Loc*a2Loc) + u)*
            ((a3Loc*a3Loc) + u),2.0) + 280.0*((a1Loc*a1Loc) + u)*((a2Loc*a2Loc)
            + u)*((a3Loc*a3Loc) + u)*((a1Loc*a1Loc) + (a2Loc*a2Loc) + (a3Loc*a3Loc)
            + 3.0*u)*pow(((a1Loc*a1Loc) + u)*((a2Loc*a2Loc) + u) + ((a1Loc*a1Loc)
            + u)*((a3Loc*a3Loc) + u) + ((a2Loc*a2Loc) + u)*((a3Loc*a3Loc) + u),3.0)
            - 63.0*pow(((a1Loc*a1Loc) + u)*((a2Loc*a2Loc) + u) + ((a1Loc*a1Loc) +
            u)*((a3Loc*a3Loc) + u) +((a2Loc*a2Loc) + u)*((a3Loc*a3Loc) + u),5.0)))/
            (((a1Loc*a1Loc) + u)*pow(((a1Loc*a1Loc) + u)*((a2Loc*a2Loc) + u)*
            ((a3Loc*a3Loc) + u),5.5)) );

}


// calculates only one integral defined by k (A1->k=0,...).
double NumericalMethods::ChandraIntegrals::Calculate(int k,double u1,double u2,
                                                     double h,double* x)
{
    double u;
    double sum=0.0;

    switch (k) // choose between A1,A2,...
    {
     case (0):
          a1Loc=x[0]/norm;
          a2Loc=x[1]/norm;
          a3Loc=x[2]/norm;
          break;

     case (1):
          a1Loc=x[1]/norm;
          a2Loc=x[0]/norm;
          a3Loc=x[2]/norm;
          break;

     case (2):
          a1Loc=x[2]/norm;
          a2Loc=x[1]/norm;
          a3Loc=x[0]/norm;
          break;

     case (3):
          a1Loc=x[3]/norm;
          a2Loc=x[4]/norm;
          a3Loc=x[5]/norm;
          break;

     case (4):
          a1Loc=x[4]/norm;
          a2Loc=x[3]/norm;
          a3Loc=x[5]/norm;
          break;
```

```
     case (5):
          a1Loc=x[5]/norm;
          a2Loc=x[4]/norm;
          a3Loc=x[3]/norm;
          break;

     default:
          std::cout<<"Error in class Integral, method calculate_dfdu";
          break;

    }  // end of switch



    // calculate sum
    for(u=u1;u<=u2; u+=h){
           sum+=f(u);
           }

    // apply Euler-MacLauren formula
    return( a1Loc*a2Loc*a3Loc*( 0.5*h*(2.*sum-f(u1)-f(u2))-(h*h/12.)*
            (df(u2)-df(u1))+(h*h*h*h/720.)*(df3(u2)-df3(u1))-
            (h*h*h*h*h*h/30240.)*(df5(u2)-df5(u1))) );

}



void NumericalMethods::ChandraIntegrals::CalculateAll(double u1,double u2,
                                                      double h,double* x,double* A)
{
    for(int i=0;i<=5;i++){// calculate A1,A2,...
       A[i]=ChandraIntegrals::Calculate(i,u1,u2,h,x);
    }

}


NumericalMethods::ChandraIntegrals::~ChandraIntegrals() // destructor
{
}
```

RK4.cpp

```
#include "NumericalMethods.h"



// constructor creates RK4 objects and initializes private members
NumericalMethods::RK4::RK4(int nExt,const ODEs myEquationsExt,double xExt,
                                                    double* yExt,double hExt):
n(nExt),
myEquations(myEquationsExt),
x(xExt),
y(yExt),
h(hExt),
h2(0.5*h),
h6(h/6.0)

{
    // memory allocation
    k1= new double[n];
    k2= new double[n];
    k3= new double[n];
    k4= new double[n];
    yTemp=new double[n];
```

```
    // zero initialization
    for(int i=0;i<=n-1;i++){
        k1[i]=0.0;
        k2[i]=0.0;
        k3[i]=0.0;
        k4[i]=0.0;
        yTemp[i]=0.0;
    }

}



// applies the RK4 algorithm for "howManySteps" steps
void NumericalMethods::RK4::Solve(int howManySteps)
{
    int i,j; // "iterators"

    for(j=1;j<=howManySteps;j++){//

        // 1st step
        myEquations(x,y,k1);
        for(i=0;i<n;i++)
        yTemp[i]=y[i]+h2*k1[i];


        // 2nd step
        myEquations(x+h2,yTemp,k2);
        for(i=0;i<n;i++)
            yTemp[i]=y[i]+h2*k2[i];


        // 3rd step
        myEquations(x+h2,yTemp,k3);
        for(i=0;i<n;i++)
            yTemp[i]=y[i]+h*k3[i];


        // 4th step
        myEquations(x+h,yTemp,k4);
        for(i=0;i<n;i++)
            y[i]+=h6*(k1[i]+k4[i]+2.0*(k2[i]+k3[i]));

        // increase the independent variable
        x+=h;

    }// end of j-loop

}


NumericalMethods::RK4::~RK4 ()
{
    // deallocate memory
    delete[] k1;
    delete[] k2;
    delete[] k3;
    delete[] k4;
    delete[] yTemp;

}
```

RK4AdaptStep.cpp

```
#include "NumericalMethods.h"


// constants needed for implimenting RK4AdaptStep
namespace  // (visible only in RK4AdaptStep.cpp)
{
    // for changing the step size
    const double safety=0.9;
    const double pgrow=-0.2;
    const double pshrnk=-0.25;
    const double errcon=1.89*pow(10.0,-4.0);


    // for taking 1 RK step
    const double a2=0.2,a3=0.3,a4=0.6,a5=1.0,a6=0.875,b21=0.2,
                 b31=3.0/40.0,b32=9.0/40.0,b41=0.3,b42 = -0.9,b43=1.2,
                 b51 = -11.0/54.0, b52=2.5,b53 = -70.0/27.0,b54=35.0/27.0,
                 b61=1631.0/55296.0,b62=175.0/512.0,b63=575.0/13824.0,
                 b64=44275.0/110592.0,b65=253.0/4096.0,c1=37.0/378.0,
                 c3=250.0/621.0,c4=125.0/594.0,c6=512.0/1771.0,
                 dc5 = -277.00/14336.0;

    const double dc1=c1-2825.0/27648.0,dc3=c3-18575.0/48384.0,
                 dc4=c4-13525.0/55296.0,dc6=c6-0.25;

};




// constructor creates RK4 objects
NumericalMethods::RK4AdaptStep::RK4AdaptStep(int nExt,const ODEs myEquationsExt,
                      double xExt,double* yExt,double hExt,double epsilonExt):
n(nExt),
myEquations(myEquationsExt),
x(xExt),
y(yExt),
h(hExt),
epsilon(epsilonExt)
{

    // memory allocation
    k1= new double[n];
    k2= new double[n];
    k3= new double[n];
    k4= new double[n];
    k5= new double[n];
    k6= new double[n];
    yTemp=new double[n];
    yErr= new double[n];
    yScal=new double[n];
    yOut=new double[n];

    // zero initialization
    for(int i=0;i<=n-1;i++){
        k1[i]=0.0;
        k2[i]=0.0;
        k3[i]=0.0;
        k4[i]=0.0;
        k5[i]=0.0;
        k6[i]=0.0;
        yTemp[i]=0.0;
        yErr[i]=0.0;
        yScal[i]=0.0;
        yOut[i]=0.0;
    }

}
```

```
// applies the RK4 algorithm for one step
void NumericalMethods::RK4AdaptStep::OneRKStep(void)
{
    int i=0;

    //first step
    myEquations(x,y,k1);
    for (i=0;i<n;i++) {
        yScal[i]=fabs(y[i])+h*fabs(k1[i]);
        yTemp[i]=y[i]+b21*h*k1[i];
    }

    //second step
    myEquations(x+a2*h,yTemp,k2);
    for (i=0;i<n;i++)
        yTemp[i]=y[i]+h*(b31*k1[i]+b32*k2[i]);

    // third step
    myEquations(x+a3*h,yTemp,k3);
    for (i=0;i<n;i++)
        yTemp[i]=y[i]+h*(b41*k1[i]+b42*k2[i]+b43*k3[i]);

    // fourth step
    myEquations(x+a4*h,yTemp,k4);
    for (i=0;i<n;i++)
        yTemp[i]=y[i]+h*(b51*k1[i]+b52*k2[i]+b53*k3[i]+b54*k4[i]);

    // fifth step
    myEquations(x+a5*h,yTemp,k5);
    for (i=0;i<n;i++)
        yTemp[i]=y[i]+h*(b61*k1[i]+b62*k2[i]+b63*k3[i]+b64*k4[i]+b65*k5[i]);

    // sixth step
    myEquations(x+a6*h,yTemp,k6);
    for (i=0;i<n;i++)
        yOut[i]=y[i]+h*(c1*k1[i]+c3*k3[i]+c4*k4[i]+c6*k6[i]);

    //estimate error as difference between fourth and fifth order methods
    for (i=0;i<n;i++)
        yErr[i]=h*(dc1*k1[i]+dc3*k3[i]+dc4*k4[i]+dc5*k5[i]+dc6*k6[i]);

}


// applies the RK4 algorith for "howManySteps" steps and
// changes properly the step size
void NumericalMethods::RK4AdaptStep::Solve(int howManySteps)
{
    int i,j;// "iterators"

    for(j=1;j<=howManySteps;j++){// loop over internal steps

        for(;;){// loop for changing the step size

            //take one step
            RK4AdaptStep::OneRKStep();

            // calculate maximum scaled error
            errMax=0.0;
            for(i=0;i<n;i++)
                errMax=RK4AdaptStep::GetMax(errMax,fabs(yErr[i]/yScal[i]));
            errMax /= epsilon;

           if(errMax<=1.0)// step is proper, break the loop!
               break;

          // else continue changing the step size
          hTemp=safety*h*pow(errMax,pshrnk);
          h=(h>= 0.0 ? RK4AdaptStep::GetMax(hTemp,0.1*h) :
          RK4AdaptStep::GetMin(hTemp,0.1*h));
          if(h==0.0)
             std::cout<<"Error! Stepsize underflow in \
                         RK4AdaptStep->Solve"<<std::endl;

        }
```

```
        // update x and y
        x+=h;
        for(i=0;i<n;i++)
            y[i]=yOut[i];

        // calcute step size for the next iteration
        if(errMax>errcon)
            h=safety*h*pow(errMax,pgrow);
        else
            h=5.0*h;

    }//end of j-loop

}


NumericalMethods::RK4AdaptStep::~RK4AdaptStep()
{
    // deallocate memory
    delete[] k1;
    delete[] k2;
    delete[] k3;
    delete[] k4;
    delete[] k5;
    delete[] k6;
    delete[] yTemp;
    delete[] yErr;
    delete[] yScal;
    delete[] yOut;

}
```

Equations.h

```
namespace Equations
{
    // set system
    void SetSystem(double M1,double M2,double Ro1,double n1,double kn1);
    void SetSeperation(double r);
    void SetFluidProperties(double Omega1, double Lamda1);
    void SetOmegadot(double Omega1dot,double OmegaOrbdot);

    // calculate properties
    double CalculateRo(double n1,double* x);
    double CalculateHamiltonians(double* x);
    void CalculatehPlusCrossBHBH(double* hPlus,double* hCross,double* x,
                                 double PhiCap,double ThetaCap);
    void CalculatehPlusCrossBHNS(double* hPlus,double* hCross,double* x,
                                 double PhiCap,double ThetaCap);

    // equations of binary system
    void algEquatBHBHLRS(double* x,double* f);
    void algDerivBHBHLRS(double* x,double** dfdx);
    void algEquatBHBH1PN(double* x,double* f);
    void algDerivBHBH1PN(double* x,double** dfdx);

    void algEquatBHNSLRS(double* x,double* f);
    void algDerivBHNSLRS(double* x,double** dfdx);
    void algEquatBHNS1PN(double* x,double* f);
    void algDerivBHNS1PN(double* x,double** dfdx);

    void difEquationsBHBH(const double time,double* x,double* dxdt);
    void difEquationsBHNS(const double time,double* x,double* dxdt);

}
```

Equations.cpp

```
#include"Equations.h"
#include"NumericalMethods.h"


// DEFINITION OF BINARY PROPERTIES

namespace
{

    // defined by SetSystem()

    double M1;    // NS mass
    double M2;    // BH mass
    double ni;    // mass combination M1*M2/(M1+M2)^2
    double Ro1;   // radius of spherical NS
    double n1;    // polytripic index
    double kn1;   // dimensionless coefficient (depent on n)


    // defined by SetSeperation

    double r;     // binary seperation


    // defined by SetFluidProperties

    double Omega1;        // frequency of the stars
    double L1;            // internal motion
```

```
    // second time derivatives of STF mass quadropole tensor

    double I211Total=0.0;
    double I222Total=0.0;
    double I233Total=0.0;
    double I212Total=0.0;

    double I211Star=0.0;
    double I222Star=0.0;
    double I212Star=0.0;

    double I211Orb=0.0;
    double I222Orb=0.0;
    double I233Orb=0.0;
    double I212Orb=0.0;



    // fifth time derivatives of STF mass quadropole tensor

    double I511Total=0.0;
    double I522Total=0.0;
    double I533Total=0.0;
    double I512Total=0.0;

    double I511Star=0.0;
    double I522Star=0.0;
    double I533Star=0.0;
    double I512Star=0.0;

    double I511Orb=0.0;
    double I522Orb=0.0;
    double I533Orb=0.0;
    double I512Orb=0.0;

    double Omega1dot=0.0;
    double OmegaOrbdot=0.0;


    // variables to be calculated in each step

    double R1;              // mean radius
    double A[3];            // chandra integrals
    double F[7];
    NumericalMethods::ChandraIntegrals myIntegral(1.0);

}



// SET SYSTEM PROPERTIES

void Equations::SetSystem(double M1Ext,double M2Ext,double Ro1Ext,
                                                  double n1Ext,double kn1Ext)
{
    M1=M1Ext;
    M2=M2Ext;
    Ro1=Ro1Ext;
    n1=n1Ext;
    kn1=kn1Ext;
    ni=M1*M2/pow((M1+M2),2.);
}


void Equations::SetSeperation(double rExt)
{
    r=rExt;
}


void Equations::SetFluidProperties(double Omega1Ext, double L1Ext)
{
    Omega1=Omega1Ext;
    L1=L1Ext;
}
```

```
void Equations::SetOmegadot(double Omega1dotExt,double OmegaOrbdotExt)
{
    Omega1dot=Omega1dotExt;
    OmegaOrbdot=OmegaOrbdotExt;
}




// CALCULATE OTHER USEFULL PROPERTIES

double Equations::CalculateRo(double n1Ext,double* xDif)
{
    return(pow(3.,n1Ext/(3 - n1Ext))*pow(xDif[0]*xDif[1]*xDif[2],
          0.3333333333333333)*pow((pow(xDif[0],2)*pow(pow(xDif[2],2)/
          pow(xDif[0],2),0.16666666666666666)*((xDif[0] - xDif[2])*
          sqrt(pow(xDif[2],2)/pow(xDif[0],2))*(xDif[0] + xDif[2]) -
          pow(xDif[2],2)*sqrt(1 - pow(xDif[2],2)/pow(xDif[0],2))*
          asin(sqrt(1 - pow(xDif[2],2)/pow(xDif[0],2))))/pow(
          pow(xDif[0],2) - pow(xDif[2],2),2),n1Ext/(3 - n1Ext)));
}



double Equations::CalculateHamiltonians(double* x)
{
    double T;     // kinetic energy
    double U;     // internal energy
    double W;     // self-gravitational potential energy
    double Hstar; // energy of the star (T+U+W)
    double H1PN;  // orbital energy


    // chandra integrals
    A[0]=myIntegral.Calculate(0,0.0,100.0,0.1,x) +
         myIntegral.Calculate(0,100.0,10000.0,0.5,x);

    A[1]=myIntegral.Calculate(1,0.0,100.0,0.1,x) +
        myIntegral.Calculate(1,100.0,10000.0,0.5,x);

    A[2]=myIntegral.Calculate(2,0.0,100.0,0.1,x) +
         myIntegral.Calculate(2,100.0,10000.0,0.5,x);


    T=-0.4*kn1*M1*x[0]*x[1]*x[10]*x[4] + 0.1*kn1*M1*(pow(x[0],2) +
      pow(x[1],2))*(pow(x[10],2) + pow(x[4],2)) + 0.1*kn1*M1*(pow(x[7],2)
      + pow(x[8],2) + pow(x[9],2));


    U=-((pow(M1,2)*n1*pow(Ro1/pow(x[0]*x[1]*x[2],0.3333333333333333),
      3/n1))/((-5 + n1)*Ro1));


    W=(-3*pow(M1,2)*(A[0]*pow(x[0],2) + A[1]*pow(x[1],2) +
      A[2]*pow(x[2],2)))/(2.*(5 - n1)*x[0]*x[1]*x[2]);


    Hstar=T+U+W;


    H1PN=-(kn1*((-1 + 3*pow(cos(x[3] - x[6]),2))*pow(x[0],2) + (-1 + 3*
         pow(sin(x[3] - x[6]),2))*pow(x[1],2) - pow(x[2],2)))/
         (10.*pow(x[5],3))+1./(2.*pow(x[5],2)) - 1/x[5] + pow(x[11],2)/2.
         + (pow(x[5],2)*pow(x[12],2))/2. - ((3 + 2*ni)*pow(x[11],2) +
         (3 + ni)*pow(x[5],2)*pow(x[12],2))/(2.*x[5]) - ((1 - 3*ni)*
         (pow(x[11],4) + 2*pow(x[5],2)*pow(x[11],2)*pow(x[12],2) +
         pow(x[5],4)*pow(x[12],4)))/8.;


    return(H1PN/Hstar);

}
```

```
void Equations::CalculatehPlusCrossBHBH(double* hPlus,double* hCross,
                                        double* x,double PhiCap,double ThetaCap)
{
    // only orbital contribution for BH-BH systems

    I211Total=-2*M1*M2*sin(2.0*x[1])*x[0]*x[2]*x[3] + M1*M2*
              cos(2.0*x[1])*(-(1/x[0]) + pow(x[2],2)- pow(x[0],2)
              *pow(x[3],2)) + (M1*M2*(-(1/x[0]) + pow(x[2],2)
              + pow(x[0],2)*pow(x[3],2)))/3.;


    I222Total=2*M1*M2*sin(2.0*x[1])*x[0]*x[2]*x[3] - M1*M2*
              cos(2.0*x[1])*(-(1/x[0]) + pow(x[2],2)
              - pow(x[0],2)*pow(x[3],2)) + (M1*M2*(-(1/x[0])
              + pow(x[2],2)+ pow(x[0],2)*pow(x[3],2)))/3.;


    I233Total=(-2*M1*M2*(-(1/x[0]) + pow(x[2],2) +
               pow(x[0],2)*pow(x[3],2)))/3.;


    I212Total=2*M1*M2*cos(2*x[1])*x[0]*x[2]*x[3] + M1*M2*sin(2*x[1])
              *(-(1/x[0])+ pow(x[2],2) - pow(x[0],2)*pow(x[3],2));



    *hPlus=0.5*((1+cos(ThetaCap)*cos(ThetaCap))*((I211Total-I222Total)
           *cos(2.0*PhiCap)+ 2*I212Total*sin(2.0*PhiCap)) +
           3*I233Total*sin(ThetaCap)*sin(ThetaCap));


    *hCross=2.0*I212Total*cos(ThetaCap);

}



void Equations::CalculatehPlusCrossBHNS(double* hPlus,double* hCross,
                                        double* x,double PhiCap,double ThetaCap)
{

    // orbital contribution

    I211Orb=-(M1*M2*sin(2.0*x[6])*((-3*kn1*sin(2*(-x[3] + x[6]))*
            (pow(x[0],2) -pow(x[1],2)))/(10.*pow(x[5],3)) +
            2*x[5]*x[11]*x[12])) + M1*M2*cos(2.0*x[6])*((-3*kn1*
            ((-1 + 3*pow(cos(x[3] - x[6]),2))*pow(x[0],2) + (-1 +
            3*pow(sin(x[3] - x[6]),2))*pow(x[1],2) - pow(x[2],2)))
            /(10.*pow(x[5],3)) - 1/x[5] + pow(x[11],2) - pow(x[5],2)
            *pow(x[12],2)) +(M1*M2*((-3*kn1*((-1 + 3*pow(cos(x[3]
            - x[6]),2))*pow(x[0],2) +(-1 + 3*pow(sin(x[3] - x[6]),2))*
            pow(x[1],2) - pow(x[2],2)))/(10.*pow(x[5],3)) - 1/x[5]+
            pow(x[11],2) + pow(x[5],2)*pow(x[12],2)))/3.;



    I222Orb= M1*M2*sin(2.0*x[6])*((-3*kn1*sin(2*(-x[3] + x[6]))*
                  (pow(x[0],2) - pow(x[1],2)))/(10.*pow(x[5],3)) +
                     2*x[5]*x[11]*x[12]) - M1*M2*cos(2.0*x[6])*((-3*kn1*
                     ((-1 + 3*pow(cos(x[3] - x[6]),2))*pow(x[0],2) + (-1 +
                     3*pow(sin(x[3] - x[6]),2))*pow(x[1],2) - pow(x[2],2)))/
                     (10.*pow(x[5],3)) - 1/x[5] + pow(x[11],2) - pow(x[5],2)*
                     pow(x[12],2)) + (M1*M2*((-3*kn1*((-1 + 3*pow(cos(x[3] -
                     x[6]),2))*pow(x[0],2) + (-1 + 3*pow(sin(x[3] - x[6]),2))*
                     pow(x[1],2) - pow(x[2],2)))/(10.*pow(x[5],3)) - 1/x[5]+
                     pow(x[11],2) + pow(x[5],2)*pow(x[12],2)))/3.;



    I233Orb= (-2*M1*M2*((-3*kn1*((-1 + 3*pow(cos(x[3] - x[6]),2))*
            pow(x[0],2) + (-1 + 3*pow(sin(x[3] - x[6]),2))*pow(x[1],2)
            - pow(x[2],2)))/(10.*pow(x[5],3)) - 1/x[5]+pow(x[11],2)
            + pow(x[5],2)*pow(x[12],2)))/3.;
```

```
    I212Orb= M1*M2*cos(2*x[6])*((-3*kn1*sin(2*(-x[3] + x[6]))*
            (pow(x[0],2) - pow(x[1],2)))/(10.*pow(x[5],3)) +
            2*x[5]*x[11]*x[12]) + M1*M2*sin(2*x[6])*((-3*kn1*
            ((-1 + 3*pow(cos(x[3] - x[6]),2))*pow(x[0],2) +
            (-1 + 3*pow(sin(x[3] - x[6]),2))*pow(x[1],2) -
             pow(x[2],2)))/(10.*pow(x[5],3)) - 1/x[5] +
            pow(x[11],2) - pow(x[5],2)*pow(x[12],2));



    // stelar contribution

    I211Star=-2*cos(2*x[3])*(0.2*kn1*M1*pow(x[0],2) -
             0.2*kn1*M1*pow(x[1],2))*pow(x[10],2) - sin(2*x[3])
             *(Omega1dot*(0.2*kn1*M1*pow(x[0],2) - 0.2*kn1*M1*
             pow(x[1],2)) + 2*(0.4*kn1*M1*x[0]*x[7] - 0.4*kn1*
             M1*x[1]*x[8])*x[10]);


    I222Star=-I211Star;


    I212Star=-2*sin(2*x[3])*(0.2*kn1*M1*pow(x[0],2) -
             0.2*kn1*M1*pow(x[1],2))*pow(x[10],2) + cos(2*x[3])*
             (Omega1dot*(0.2*kn1*M1*pow(x[0],2) - 0.2*kn1*M1*
             pow(x[1],2)) + 2*(0.4*kn1*M1*x[0]*x[7] - 0.4*kn1*
             M1*x[1]*x[8])*x[10]);



    // stellar + orbital (total) contribution

    I211Total=I211Orb + I211Star;

    I222Total=I222Orb + I222Star;

    I212Total=I212Orb + I212Star;

    I233Total=I233Orb;



    *hPlus=0.5*((1+cos(ThetaCap)*cos(ThetaCap))*((I211Total-I222Total)*
           cos(2.0*PhiCap) + 2*I212Total*sin(2.0*PhiCap)) +
           3*I233Total*sin(ThetaCap)*sin(ThetaCap));

    *hCross=2.0*I212Total*cos(ThetaCap);

}







// EQUILIBRIUM EQUATIONS FOR BINARY SYSTEMS



// Newtonian algebraic equations for BH-BH
void Equations::algEquatBHBHLRS(double* x,double* f)
{
    // output for every iteration

    std::cout<<"Newtonian   "<<"phidot="\
             <<std::setprecision(10)<<x[0]<<std::endl;


    // creating f, from equation J*dx=f

    f[0]=-((M1 + M2)/pow(r,2.)) + r*pow(x[0],2);

}
```

```
void Equations::algDerivBHBHLRS(double* x,double** dfdx)
{
    // creations linearized matrix J, from equation J*dx=f

    dfdx[0][0]=2*r*x[0];

}




// Post-Newtonian algebraic equations for BH-BH
void Equations::algEquatBHBH1PN(double* x,double* f)
{
    // output for every step

    std::cout<<"Post Newt.  "<<"phidot"<<x[0]<<std::endl;


    // creating f, from equation J*dx=f

    f[0]=(2*pow(M1 + M2,2)*(2 + ni))/pow(r,3) - (M1 + M2)/pow(r,2)
           - (M1 + M2)*(1 + 3*ni)*pow(x[0],2) + r*pow(x[0],2);

}


void Equations::algDerivBHBH1PN(double* x,double** dfdx)
{
    // creations linearized matrix J, from equation J*dx=f

    dfdx[0][0]=-2*(M1 + M2)*(1 + 3*ni)*x[0] + 2*r*x[0];

}




// Newtonian algebraic equations for BH-NS
void Equations::algEquatBHNSLRS(double* x,double* f)
{
    // output for every iteration

    std::cout<<"Newtonian   "<<"phidot="<<std::setprecision(10)<<x[3]<<\
                      "   a1="<<x[0]<<"   a2="<<x[1]<<"   a3="<<x[2]<<std::endl;


    // chandra integrals from 0->100 (step 0.1) + from 100->10000 (step 0.5)

    A[0]=myIntegral.Calculate(0,0.0,100.0,0.1,x) +
        myIntegral.Calculate(0,100.0,10000.0,0.5,x);

    A[1]=myIntegral.Calculate(1,0.0,100.0,0.1,x) +
        myIntegral.Calculate(1,100.0,10000.0,0.5,x);

    A[2]=myIntegral.Calculate(2,0.0,100.0,0.1,x) +
        myIntegral.Calculate(2,100.0,10000.0,0.5,x);



    // creating f, from equation J*dx=f

    f[0]=(pow(L1, 2.) + pow(Omega1, 2.))*x[0] + (2.*M2*x[0])/pow(r, 3.)
         -2.*L1*Omega1*x[1] - (1.5*A[0]*M1*x[0])/(kn1*(1. - 0.2*n1)*
         pow(x[0]*x[1]*x[2],1.)) + (M1*pow(Ro1/pow(x[0]*x[1]*x[2],
         0.3333333333333333), 3./n1))/(kn1*(1. - 0.2*n1)*Ro1*x[0]);


    f[1]=-2.*L1*Omega1*x[0] + (pow(L1, 2.) + pow(Omega1, 2.))*x[1] -
         (1.*M2*x[1])/pow(r, 3.) - (1.5*A[1]*M1*x[1])/(kn1*(1. -
         0.2*n1)*pow(x[0]*x[1]*x[2], 1.)) + (M1*pow(Ro1/pow(x[0]*
         x[1]*x[2], 0.3333333333333333), 3./n1))/(kn1*(1. - 0.2*n1)
         *Ro1*x[1]);
```

```
    f[2]=-((M2*x[2])/pow(r, 3.)) - (1.5*A[2]*M1*x[2])/(kn1*(1. - 0.2*n1)
         *pow(x[0]*x[1]*x[2], 1.)) + (M1*pow(Ro1/pow(x[0]*x[1]*x[2],
         0.3333333333333333), 3./n1))/(kn1*(1. - 0.2*n1)*Ro1*x[2]);


    f[3]=-((M1 + M2)/pow(r, 2.)) - (0.3*kn1*(M1 + M2)*(2.*pow(x[0], 2.)
         - 1.*pow(x[1], 2.) -pow(x[2], 2.)))/pow(r, 4.) + r*pow(x[3], 2);

}


void Equations::algDerivBHNSLRS(double* x,double** dfdx)
{

    // chandra integrals from 0->100 (step 0.1) + from 100->10000 (step 0.5)

    A[0]=myIntegral.Calculate(0,0.0,100.0,0.1,x) +
        myIntegral.Calculate(0,100.0,10000.0,0.5,x);

    A[1]=myIntegral.Calculate(1,0.0,100.0,0.1,x) +
        myIntegral.Calculate(1,100.0,10000.0,0.5,x);

    A[2]=myIntegral.Calculate(2,0.0,100.0,0.1,x) +
        myIntegral.Calculate(2,100.0,10000.0,0.5,x);


    // creations linearized matrix J, from equation J*dx=f

    dfdx[0][0]=pow(L1, 2.) + pow(Omega1, 2.) + (2.*M2)/pow(r, 3.) +
               (1.5*A[0]*M1*x[0]*x[1]*x[2])/(kn1*(1. - 0.2*n1)*
                pow(x[0]*x[1]*x[2], 2.)) - (1.5*A[0]*M1)/(kn1*(1. -
                0.2*n1)*pow(x[0]*x[1]*x[2], 1.)) - (1.*M1*x[1]*x[2]*
                pow(Ro1/pow(x[0]*x[1]*x[2],0.3333333333333333), -1 +
                3./n1))/(kn1*(1. - 0.2*n1)*n1*x[0]*pow(x[0]*x[1]*x[2],
                1.3333333333333333)) - (M1*pow(Ro1/pow(x[0]*x[1]*x[2],
                0.3333333333333333), 3./n1))/(kn1*(1. - 0.2*n1)*Ro1
                *pow(x[0], 2));


    dfdx[0][1]=-2.*L1*Omega1 + (1.5*A[0]*M1*pow(x[0], 2)*x[2])/(kn1*(1.
               - 0.2*n1)*pow(x[0]*x[1]*x[2], 2.)) - (1.*M1*x[2]*pow(Ro1
                 /pow(x[0]*x[1]*x[2], 0.3333333333333333), -1 + 3./n1))/
                 (kn1*(1. - 0.2*n1)*n1*pow(x[0]*x[1]*x[2],
                 1.3333333333333333));


    dfdx[0][2]=(1.5*A[0]*M1*pow(x[0], 2)*x[1])/(kn1*(1. - 0.2*n1)*
                pow(x[0]*x[1]*x[2], 2.)) - (1.*M1*x[1]*pow(Ro1/pow
                 (x[0]*x[1]*x[2],0.3333333333333333), -1 + 3./n1))/
                 (kn1*(1. - 0.2*n1)*n1*pow(x[0]*x[1]*x[2],
                  1.3333333333333333));


    dfdx[0][3]=0.;



    dfdx[1][0]=-2.*L1*Omega1 + (1.5*A[1]*M1*pow(x[1], 2)*x[2])/(kn1*
               (1. - 0.2*n1)*pow(x[0]*x[1]*x[2], 2.)) - (1.*M1*x[2]*
                pow(Ro1/pow(x[0]*x[1]*x[2], 0.3333333333333333), -1 +
                3./n1))/(kn1*(1. - 0.2*n1)*n1*pow(x[0]*x[1]*x[2],
                1.3333333333333333));


    dfdx[1][1]=pow(L1, 2.) + pow(Omega1, 2.) - (1.*M2)/pow(r, 3.) +
               (1.5*A[1]*M1*x[0]*x[1]*x[2])/(kn1*(1. - 0.2*n1)*
                pow(x[0]*x[1]*x[2], 2.)) - (1.5*A[1]*M1)/(kn1*(1. -
               0.2*n1)*pow(x[0]*x[1]*x[2], 1.)) - (1.*M1*x[0]*x[2]*pow
               (Ro1/pow(x[0]*x[1]*x[2],0.3333333333333333), -1 + 3./n1))/
               (kn1*(1. - 0.2*n1)*n1*x[1]*pow(x[0]*x[1]*x[2],
               1.3333333333333333)) - (M1*pow(Ro1/pow(x[0]*x[1]*x[2],
               0.3333333333333333), 3./n1))/(kn1*(1. - 0.2*n1)*Ro1*
               pow(x[1], 2));
```

```
    dfdx[1][2]=(1.5*A[1]*M1*x[0]*pow(x[1], 2))/(kn1*(1. - 0.2*n1)*
               pow(x[0]*x[1]*x[2], 2.)) - (1.*M1*x[0]*pow(Ro1/pow
              (x[0]*x[1]*x[2],0.3333333333333333), -1 + 3./n1))/
              (kn1*(1. - 0.2*n1)*n1*pow(x[0]*x[1]*x[2],
              1.3333333333333333));


    dfdx[1][3]=0.;




    dfdx[2][0]=(1.5*A[2]*M1*x[1]*pow(x[2], 2))/(kn1*(1. - 0.2*n1)*
                pow(x[0]*x[1]*x[2], 2.)) - (1.*M1*x[1]*pow(Ro1/pow
               (x[0]*x[1]*x[2],0.3333333333333333), -1 + 3./n1))/
               (kn1*(1. - 0.2*n1)*n1*pow(x[0]*x[1]*x[2],
                1.3333333333333333));


    dfdx[2][1]=(1.5*A[2]*M1*x[0]*pow(x[2], 2))/(kn1*(1. - 0.2*n1)*
               pow(x[0]*x[1]*x[2], 2.)) - (1.*M1*x[0]*pow(Ro1/pow
              (x[0]*x[1]*x[2],0.3333333333333333), -1 + 3./n1))
              /(kn1*(1. - 0.2*n1)*n1*pow(x[0]*x[1]*x[2],
              1.3333333333333333));


    dfdx[2][2]=-(M2/pow(r, 3.)) + (1.5*A[2]*M1*x[0]*x[1]*x[2])/
                (kn1*(1. - 0.2*n1)*pow(x[0]*x[1]*x[2], 2.)) -
                (1.5*A[2]*M1)/(kn1*(1. - 0.2*n1)*pow(x[0]*x[1]*x[2],
                  1.)) - (1.*M1*x[0]*x[1]*pow(Ro1/pow(x[0]*x[1]*x[2],
                  0.3333333333333333), -1 + 3./n1))/(kn1*(1. - 0.2*n1)
                  *n1*x[2]*pow(x[0]*x[1]*x[2], 1.3333333333333333)) -
                  (M1*pow(Ro1/pow(x[0]*x[1]*x[2],0.3333333333333333),
                  3./n1))/(kn1*(1. - 0.2*n1)*Ro1*pow(x[2], 2));


    dfdx[2][3]=0.;



    dfdx[3][0]=(-1.2*kn1*(M1 + M2)*pow(x[0], 1.))/pow(r, 4.);


    dfdx[3][1]=(0.6*kn1*(M1 + M2)*pow(x[1], 1.))/pow(r, 4.);


    dfdx[3][2]=(0.6*kn1*(M1 + M2)*pow(x[2], 1.))/pow(r, 4.);


    dfdx[3][3]=2*r*x[3];

}




// Post-Newtonian algebraic equations for BH-NS
void Equations::algEquatBHNS1PN(double* x,double* f)
{
    // output for every step

    std::cout<<"Post Newt.  "<<"phidot="<<x[3]<<"   a1="\
             <<x[0]<<"   a2="<<x[1]<<"   a3="<<x[2]<<std::endl;


    // chandra integrals from 0->100 (step 0.1) + from 100->10000 (step 0.5)

    A[0]=myIntegral.Calculate(0,0.0,100.0,0.1,x) +
         myIntegral.Calculate(0,100.0,10000.0,0.5,x);
```

```
    A[1]=myIntegral.Calculate(1,0.0,100.0,0.1,x) +
         myIntegral.Calculate(1,100.0,10000.0,0.5,x);

    A[2]=myIntegral.Calculate(2,0.0,100.0,0.1,x) +
         myIntegral.Calculate(2,100.0,10000.0,0.5,x);


    // creating f, from equation J*dx=f

    f[0]=(pow(L1, 2.) + pow(Omega1, 2.))*x[0] + (2.*M2*x[0])/
          pow(r, 3.) - 2.*L1*Omega1*x[1] - (1.5*A[0]*M1*x[0])/
          (kn1*(1. - 0.2*n1)*pow(x[0]*x[1]*x[2],1.)) + (M1*pow
          (Ro1/pow(x[0]*x[1]*x[2], 0.3333333333333333), 3./n1))/
          (kn1*(1. - 0.2*n1)*Ro1*x[0]);


    f[1]=-2.*L1*Omega1*x[0] + (pow(L1, 2.) + pow(Omega1, 2.))*
          x[1] - (1.*M2*x[1])/pow(r, 3.) - (1.5*A[1]*M1*x[1])/
          (kn1*(1. - 0.2*n1)*pow(x[0]*x[1]*x[2], 1.)) + (M1*
          pow(Ro1/pow(x[0]*x[1]*x[2], 0.3333333333333333),
          3./n1))/(kn1*(1. - 0.2*n1)*Ro1*x[1]);


    f[2]=-((M2*x[2])/pow(r, 3.)) - (1.5*A[2]*M1*x[2])/(kn1*
          (1. - 0.2*n1)*pow(x[0]*x[1]*x[2], 1.)) + (M1*pow
          (Ro1/pow(x[0]*x[1]*x[2],0.3333333333333333), 3./n1))/
          (kn1*(1. - 0.2*n1)*Ro1*x[2]);


    f[3]=(2*pow(M1 + M2, 2)*(2 + ni))/pow(r, 3) - (M1 + M2)/
         pow(r, 2) + (3*pow(M1 + M2, 2)*(3 + 2*ni)*(kn1*pow
         (x[0], 2) + kn1*pow(x[1],2) + 3*kn1*(pow(x[0], 2) -
         pow(x[1],2)) - 2*kn1*pow(x[2], 2)))/(20.*pow(r, 5))
         + (3*(M1 + M2)*(-2*kn1*pow(x[0], 2) + kn1*pow(x[1],
         2) + kn1*pow(x[2], 2)))/(10.*pow(r, 4))- (M1 + M2)*
         (1 + 3*ni)*pow(x[3], 2) + r*pow(x[3], 2) - (3*(M1 +
         M2)*(-1 + 3*ni)*(kn1*pow(x[0],2)+kn1*pow(x[1], 2) +
         3*kn1*(pow(x[0], 2) - pow(x[1], 2)) - 2*kn1*
         pow(x[2], 2))*pow(x[3], 2))/(40.*pow(r, 2));

}


void Equations::algDerivBHNS1PN(double* x,double** dfdx)
{

    // chandra integrals from 0->100 (step 0.1)+from 100->10000 (step 0.5)

    A[0]=myIntegral.Calculate(0,0.0,100.0,0.1,x) +
         myIntegral.Calculate(0,100.0,10000.0,0.5,x);

    A[1]=myIntegral.Calculate(1,0.0,100.0,0.1,x) +
         myIntegral.Calculate(1,100.0,10000.0,0.5,x);

    A[2]=myIntegral.Calculate(2,0.0,100.0,0.1,x) +
         myIntegral.Calculate(2,100.0,10000.0,0.5,x);


    // creations linearized matrix J, from equation J*dx=f

    dfdx[0][0]=pow(L1, 2.) + pow(Omega1, 2.) + (2.*M2)/pow(r, 3.) +
               (1.5*A[0]*M1*x[0]*x[1]*x[2])/(kn1*(1. - 0.2*n1)*pow(
                x[0]*x[1]*x[2], 2.)) - (1.5*A[0]*M1)/(kn1*(1. - 0.2*n1)
                *pow(x[0]*x[1]*x[2], 1.)) - (1.*M1*x[1]*x[2]*pow(Ro1/
                pow(x[0]*x[1]*x[2],0.3333333333333333), -1 + 3./n1))/
                (kn1*(1. - 0.2*n1)*n1*x[0]*pow(x[0]*x[1]*x[2],
                1.3333333333333333)) - (M1*pow(Ro1/pow(x[0]*x[1]*x[2],
                0.3333333333333333), 3./n1))/(kn1*(1. - 0.2*n1)*Ro1
                *pow(x[0], 2));


    dfdx[0][1]=-2.*L1*Omega1 + (1.5*A[0]*M1*pow(x[0], 2)*x[2])/
                (kn1*(1. - 0.2*n1)*pow(x[0]*x[1]*x[2], 2.)) -
                 (1.*M1*x[2]*pow(Ro1/pow(x[0]*x[1]*x[2],
                 0.3333333333333333), -1 + 3./n1))/(kn1*(1. -
                 0.2*n1)*n1*pow(x[0]*x[1]*x[2],
                 1.3333333333333333));
```

```
dfdx[0][2]=(1.5*A[0]*M1*pow(x[0], 2)*x[1])/(kn1*(1. - 0.2*n1)
           *pow(x[0]*x[1]*x[2], 2.)) - (1.*M1*x[1]*pow(Ro1/
            pow(x[0]*x[1]*x[2],0.3333333333333333), -1 + 3./n1))/
           (kn1*(1. - 0.2*n1)*n1*pow(x[0]*x[1]*x[2],
            1.3333333333333333));


dfdx[0][3]=0.0;




dfdx[1][0]=-2.*L1*Omega1 + (1.5*A[1]*M1*pow(x[1], 2)*x[2])/
           (kn1*(1. - 0.2*n1)*pow(x[0]*x[1]*x[2], 2.)) -
           (1.*M1*x[2]*pow(Ro1/pow(x[0]*x[1]*x[2],
            0.3333333333333333), -1 + 3./n1))/(kn1*(1. -
            0.2*n1)*n1*pow(x[0]*x[1]*x[2], 1.3333333333333333));


dfdx[1][1]=pow(L1, 2.) + pow(Omega1, 2.) - (1.*M2)/pow(r, 3.) +
           (1.5*A[1]*M1*x[0]*x[1]*x[2])/(kn1*(1. - 0.2*n1)*
            pow(x[0]*x[1]*x[2], 2.)) - (1.5*A[1]*M1)/(kn1*(1. -
            0.2*n1)*pow(x[0]*x[1]*x[2], 1.)) - (1.*M1*x[0]*x[2]*
            pow(Ro1/pow(x[0]*x[1]*x[2],0.3333333333333333), -1 +
            3./n1))/(kn1*(1. - 0.2*n1)*n1*x[1]*pow(x[0]*x[1]*x[2],
            1.3333333333333333)) - (M1*pow(Ro1/pow(x[0]*x[1]*x[2],
            0.3333333333333333), 3./n1))/(kn1*(1. - 0.2*n1)*Ro1*
            pow(x[1], 2));


dfdx[1][2]=(1.5*A[1]*M1*x[0]*pow(x[1], 2))/(kn1*(1. - 0.2*n1)*
            pow(x[0]*x[1]*x[2], 2.)) - (1.*M1*x[0]*pow(Ro1/
           pow(x[0]*x[1]*x[2],0.3333333333333333), -1 + 3./n1))/
           (kn1*(1. - 0.2*n1)*n1*pow(x[0]*x[1]*x[2],
           1.3333333333333333));


dfdx[1][3]=0.0;




dfdx[2][0]=(1.5*A[2]*M1*x[1]*pow(x[2], 2))/(kn1*(1. - 0.2*n1)*
            pow(x[0]*x[1]*x[2], 2.)) - (1.*M1*x[1]*pow(Ro1/
           pow(x[0]*x[1]*x[2],0.3333333333333333), -1 + 3./n1))/
           (kn1*(1. - 0.2*n1)*n1*pow(x[0]*x[1]*x[2],
           1.3333333333333333));


dfdx[2][1]=(1.5*A[2]*M1*x[0]*pow(x[2], 2))/(kn1*(1. - 0.2*n1)*
            pow(x[0]*x[1]*x[2], 2.)) - (1.*M1*x[0]*pow(Ro1/
           pow(x[0]*x[1]*x[2],0.3333333333333333), -1 + 3./n1))/
           (kn1*(1. - 0.2*n1)*n1*pow(x[0]*x[1]*x[2],
           1.3333333333333333));


dfdx[2][2]=-(M2/pow(r, 3.)) + (1.5*A[2]*M1*x[0]*x[1]*x[2])/
            (kn1*(1. - 0.2*n1)*pow(x[0]*x[1]*x[2], 2.)) -
           (1.5*A[2]*M1)/(kn1*(1. - 0.2*n1)*pow(x[0]*x[1]*x[2],
           1.)) - (1.*M1*x[0]*x[1]*pow(Ro1/pow(x[0]*x[1]*x[2],
           0.3333333333333333), -1 + 3./n1))/(kn1*(1. - 0.2*n1)
           *n1*x[2]*pow(x[0]*x[1]*x[2], 1.3333333333333333)) -
           (M1*pow(Ro1/pow(x[0]*x[1]*x[2],0.3333333333333333),
           3./n1))/(kn1*(1. - 0.2*n1)*Ro1*pow(x[2], 2));


dfdx[2][3]=0.0;



dfdx[3][0]=(6*kn1*pow(M1 + M2, 2)*(3 + 2*ni)*x[0])/(5.*pow(r, 5)) -
          (6*kn1*(M1 + M2)*x[0])/(5.*pow(r, 4)) - (3*kn1*(M1 +
           M2)*(-1 + 3*ni)*x[0]*pow(x[3], 2))/(5.*pow(r, 2));
```

```
    dfdx[3][1]=(-3*kn1*pow(M1 + M2, 2)*(3 + 2*ni)*x[1])/(5.*pow(r,
               5)) + (3*kn1*(M1 + M2)*x[1])/(5.*pow(r, 4)) +
               (3*kn1*(M1 + M2)*(-1 + 3*ni)*x[1]*pow(x[3], 2))/
                (10.*pow(r, 2));


    dfdx[3][2]=(-3*kn1*pow(M1 + M2, 2)*(3 + 2*ni)*x[2])/(5.*pow(r,
                5)) + (3*kn1*(M1 + M2)*x[2])/(5.*pow(r, 4)) +
                 (3*kn1*(M1 + M2)*(-1 + 3*ni)*x[2]*pow(x[3], 2))/
                 (10.*pow(r, 2));


    dfdx[3][3]=-2*(M1 + M2)*(1 + 3*ni)*x[3] + 2*r*x[3] - (3*(M1 +
               M2)*(-1 + 3*ni)*(kn1*pow(x[0], 2) + kn1*pow(x[1],
                2) + 3*kn1*(pow(x[0], 2) - pow(x[1], 2)) - 2*kn1*
                pow(x[2], 2))*x[3])/(20.*pow(r, 2));

}




// DIFFERENTIAL EQUATIONS FOR BINARY SYSTEMS


// equations for BH-BH binary system
void Equations::difEquationsBHBH(const double time,double* x,double* dxdt)
{
    // variables
    // x[0]->r     x[1]->phi
    // x[2]->r_dot     x[3]->phi_dot


    // fifth time derivatives of STF mass quadropole tensor
    // only orbital contribution for BH-BH systems

    I511Total=(-8*M1*M2*x[2]*(4/x[0] + 3*pow(x[2],2) +
              18*pow(x[0],2)*pow(x[3],2)))/(3.*pow(x[0],4));

    I522Total=(2*M1*M2*x[2]*(8/x[0] + 6*pow(x[2],2) +
              81*pow(x[0],2)*pow(x[3],2)))/(3.*pow(x[0],4));

    I512Total=(-4*M1*M2*x[3]*(2/x[0] + 9*pow(x[2],2) -
              6*pow(x[0],2)*pow(x[3],2)))/pow(x[0],3);

    I533Total=(2*M1*M2*x[2]*(8/x[0] + 6*pow(x[2],2) -
              9*pow(x[0],2)*pow(x[3],2)))/(3.*pow(x[0],4));



    //************************* Newtonian Equations *************************


    //F[0]=-((M1 + M2)/pow(x[0],2.)) + x[0]*pow(x[3],2);


    //F[1]=(-2.*x[2]*x[3])/x[0];



    //************************** Post Newtonian Equations *******************


    F[0]=(2*pow(M1 + M2,2)*(2 + ni))/pow(x[0],3) - (M1 + M2)/pow(x[0],2) +
         ((M1 + M2)*(60 - 70*ni)*pow(x[2],2))/(20.*pow(x[0],2)) +
        (M1 + M2)*(-1 - 3*ni)*pow(x[3],2) + x[0]*pow(x[3],2);



    F[1]=(-2*(M1 + M2)*(-2 + ni)*x[2]*x[3])/pow(x[0],2) - (2*x[2]*x[3])/x[0];
```

```
    //*************************** Gravitational Radiation *******************


    F[0]+=-0.4*I511Total*x[0];

    F[1]+=-0.4*I512Total;


    //*************************** Update Variables *************************

    dxdt[0]=x[2];
    dxdt[1]=x[3];
    dxdt[2]=F[0];
    dxdt[3]=F[1];

}




// equations for BH-NS binary system
void Equations::difEquationsBHNS(const double time,double* x,double* dxdt)
{
    // variables notation

    // x[0]->a1
    // x[1]->a2
    // x[2]->a3
    // x[3]->gamma
    // x[4]->L1 (lamda)
    // x[5]->r
    // x[6]->phi
    // x[7]->a1_dot
    // x[8]->a2_dot
    // x[9]->a3_dot
    // x[10]->Omega1
    // x[11]->r_dot
    // x[12]->phi_dot (OmegaOrbital)



    R1=pow((x[0]*x[1]*x[2]),(1./3.));

    A[0]=myIntegral.Calculate(0,0.0,100.0,0.1,x) +
         myIntegral.Calculate(0,100.0,10000.0,0.5,x);

    A[1]=myIntegral.Calculate(1,0.0,100.0,0.1,x) +
         myIntegral.Calculate(1,100.0,10000.0,0.5,x);

    A[2]=myIntegral.Calculate(2,0.0,100.0,0.1,x) +
         myIntegral.Calculate(2,100.0,10000.0,0.5,x);



    // fifth time derivatives of STF mass quadropole tensor



 // orbital contribution

    I511Orb=(-8*M1*M2*x[11]*(4/x[5] + 3*pow(x[11],2) +
            18*pow(x[5],2)*pow(x[12],2)))/(3.*pow(x[5],4));

    I522Orb=(2*M1*M2*x[11]*(8/x[5] + 6*pow(x[11],2) +
            81*pow(x[5],2)*pow(x[12],2)))/(3.*pow(x[5],4));

    I512Orb=(-4*M1*M2*x[12]*(2/x[5] + 9*pow(x[11],2) -
            6*pow(x[5],2)*pow(x[12],2)))/pow(x[5],3);

    I533Orb=(2*M1*M2*x[11]*(8/x[5] + 6*pow(x[11],2) -
            9*pow(x[5],2)*pow(x[12],2)))/(3.*pow(x[5],4));
```

```
// stellar contribution

I511Star=40*cos(2*(-x[3] + x[6]))*(0.4*kn1*M1*x[0]*x[7] -
         0.4*kn1*M1*x[1]*x[8])*pow(x[10],4) + (0.2*kn1*M1*
         pow(x[0],2) - 0.2*kn1*M1*pow(x[1],2))*(80*Omega1dot
         *cos(2*(-x[3] + x[6]))*pow(x[10],3) + 16*sin(2*
         (-x[3] + x[6]))*pow(x[10],5));

I512Star=-40*sin(2*(-x[3] + x[6]))*(0.4*kn1*M1*x[0]*x[7] -
         0.4*kn1*M1*x[1]*x[8])*pow(x[10],4) + (0.2*kn1*M1*
         pow(x[0],2) - 0.2*kn1*M1*pow(x[1],2))*(-80*Omega1dot
         *sin(2*(-x[3] + x[6]))*pow(x[10],3) + 16*cos(2*
         (-x[3] + x[6]))*pow(x[10],5));

 I522Star=-I511Star;


 I533Star=0.0;



// total contribution

I511Total=I511Orb + I511Star;

I522Total=I522Orb + I522Star;

I512Total=I512Orb + I512Star;

I533Total=I533Orb + I533Star;


//*********************** Stellar Equations  ***************************


F[0]=(M1*pow(Ro1/R1, 3./n1))/(kn1*(1. - 0.2*n1)*Ro1*x[0]) -
      (1.5*A[0]*M1*x[0])/(kn1*(1. - 0.2*n1)*pow(R1, 3.)) +
      (M2*(-1. + 3.*pow(cos(x[3] - x[6]),2.))*x[0])/pow(x[5],
      3.) - 2.*x[1]*x[4]*x[10] + x[0]*(pow(x[4], 2.) +
      pow(x[10], 2.));


F[1]=(M1*pow(Ro1/R1, 3./n1))/(kn1*(1. - 0.2*n1)*Ro1*x[1]) -
     (1.5*A[1]*M1*x[1])/(kn1*(1. - 0.2*n1)*pow(R1, 3.)) +
     (M2*(-1. + 3.*pow(-sin(x[3] - x[6]),2.))*x[1])/pow(x[5],
     3.) - 2.*x[0]*x[4]*x[10] + x[1]*(pow(x[4], 2.) +
     pow(x[10], 2.));


F[2]=(M1*pow(Ro1/R1, 3./n1))/(kn1*(1. - 0.2*n1)*Ro1*x[2]) -
     (1.5*A[2]*M1*x[2])/(kn1*(1. - 0.2*n1)*pow(R1, 3.)) -
     (M2*x[2])/pow(x[5], 3.);


F[3]=((-1.5*M2*sin(2.*(-x[3] + x[6]))*(x[0]/x[1] + x[1]/x[0]))/
     pow(x[5], 3.) - 2.*x[8]*(x[4]/x[1] + x[10]/x[0]) +
     2.*x[7]*(x[4]/x[0] + x[10]/x[1]))/(-(x[0]/x[1]) + x[1]/x[0]);


F[4]=((-3.*M2*sin(2.*(-x[3] + x[6])))/pow(x[5], 3.) + 2.
      *x[7]*(x[4]/x[1] + x[10]/x[0]) - 2.*x[8]*(x[4]/x[0] +
      x[10]/x[1]))/(-(x[0]/x[1]) + x[1]/x[0]);



//************************ Orbital Equations ****************************

//************************* Newtonian Equations ************************


//F[5]=(-0.3*kn1*(M1 + M2)*((-1. + 3.*pow(cos(x[3] - x[6]),2.))*
    //  pow(x[0],2.) + (-1. + 3.*pow(-sin(x[3] - x[6]), 2.))*
    //  pow(x[1], 2.) - pow(x[2], 2.)))/pow(x[5], 4.) -
    //  (M1 + M2)/pow(x[5], 2.) + x[5]*pow(x[12], 2);
```

```
//F[6]=(-0.3*kn1*(M1 + M2)*sin(2.*(-x[3] + x[6]))*(pow(x[0],2.) -
   // pow(x[1], 2.)))/pow(x[5], 5.) - (2.*x[11]*x[12])/x[5];


//************************ Post Newtonian Equations *********************


F[5]=(3*pow(M1 + M2, 2)*(3 + 2*ni)*(kn1*pow(x[0], 2) + kn1*
     pow(x[1], 2) + 3*kn1*cos(2*(-x[3] + x[6]))*(pow(x[0],2)
     - pow(x[1], 2)) - 2*kn1*pow(x[2], 2)))/(20.*pow(x[5], 5))
     + (3*(M1 + M2)*(kn1*pow(x[0], 2) - 3*kn1*pow(cos(x[3] -
     x[6]), 2)*pow(x[0], 2) + kn1*pow(x[1], 2) - 3*kn1*pow(
     sin(x[3] - x[6]), 2)*pow(x[1], 2) + kn1*pow(x[2], 2)))/
     (10.*pow(x[5], 4)) - (M1 + M2)/pow(x[5],2) - (9*(M1 + M2)
     *(-1 + 3*ni)*(kn1*pow(x[0], 2) + kn1*pow(x[1], 2) + 3*
     kn1*cos(2*(-x[3] + x[6]))*(pow(x[0], 2) - pow(x[1], 2))-
     2*kn1*pow(x[2],2))*pow(x[11], 2))/(40.*pow(x[5], 4)) +
     (M1 + M2)*(-1 - 3*ni)*pow(x[12], 2) + x[5]*pow(x[12], 2)
     - (-20*pow(M1 + M2,2)*(2 + ni) + 3*kn1*(M1 + M2)*(-1 +
     3*ni)*sin(2*(-x[3] + x[6]))*(pow(x[0], 2) - pow(x[1], 2))
     *x[11]*x[12])/(10.*pow(x[5], 3)) + ((M1 + M2)*((60 - 70*ni)
     *pow(x[11],2) - (3*(-1 + 3*ni)*(kn1*pow(x[0], 2) + kn1*
     pow(x[1],2) + 3*kn1*cos(2*(-x[3] + x[6]))*(pow(x[0], 2) -
     pow(x[1], 2)) - 2*kn1*pow(x[2], 2))*pow(x[12], 2))/2.))/
     (20.*pow(x[5], 2));



F[6]=(3*kn1*pow(M1 + M2, 2)*(3 + ni)*sin(2*(-x[3] + x[6]))*
    (pow(x[0],2)- pow(x[1], 2)))/(10.*pow(x[5], 6)) -
    (3*kn1*(M1 + M2)*sin(2*(-x[3] + x[6]))*(pow(x[0], 2) -
    pow(x[1],2)))/(10.*pow(x[5], 5)) - (3*kn1*(M1 + M2)*
    (-1 + 3*ni)*sin(2*(-x[3] + x[6]))*(pow(x[0],2) -
    pow(x[1], 2))*pow(x[11], 2))/(20.*pow(x[5], 5)) -
    (3*(M1 + M2)*(-1 + 3*ni)*(kn1* pow(x[0], 2) +
    kn1*pow(x[1], 2) + 3*kn1*cos(2*(-x[3] + x[6]))*
    (pow(x[0], 2) - pow(x[1], 2)) - 2*kn1*pow(x[2], 2))
    *x[11]*x[12])/(20.*pow(x[5], 4)) - (2*(M1 + M2)*
    (-2 + ni)*x[11]*x[12])/pow(x[5], 2) - (2*x[11]*
    x[12])/x[5] - (9*kn1*(M1 + M2)*(-1 + 3*ni)*
    sin(2*(-x[3] + x[6]))*(pow(x[0], 2) -
    pow(x[1], 2))*pow(x[12], 2))/(20.*pow(x[5], 3));



//*************************** Gravitational Radiation ******************


F[0]+=-0.4*(I511Total*cos(x[6]-x[3])*cos(x[6]-x[3]) +
     I522Total*sin(x[6]-x[3])*sin(x[6]-x[3])
     - I512Total*sin(2.0*(x[6]-x[3])))*x[0];


F[1]+=-0.4*(I511Total*sin(x[6]-x[3])*sin(x[6]-x[3]) +
     I522Total*cos(x[6]-x[3])*cos(x[6]-x[3])
     + I512Total*sin(2.0*(x[6]-x[3])))*x[1];


F[2]+=-0.4*I533Total*x[2];


F[3]=(F[3]*(x[1]/x[0]-x[0]/x[1]) + 0.4*(I512Total*cos
    (2.0*(x[6]-x[3])) + 0.5*(I511Total-I522Total)*
    sin(2.0*(x[6]-x[3])))*(x[0]/x[1]+x[1]/x[0]))/
    (x[1]/x[0]-x[0]/x[1]);


F[4]=(F[4]*(x[1]/x[0]-x[0]/x[1]) + 0.8*(I512Total*
    cos(2.0*(x[6]-x[3])) + 0.5*(I511Total-I522Total)*
    sin(2.0*(x[6]-x[3])))/(x[1]/x[0]-x[0]/x[1]);


F[5]+=-0.4*I511Total*x[5];


 F[6]+=-0.4*I512Total;
```

```
    //*************************** Update Variables ***************************


    dxdt[0]=x[7];
    dxdt[1]=x[8];
    dxdt[2]=x[9];
    dxdt[3]=x[10];
    dxdt[4]=F[4];
    dxdt[5]=x[11];
    dxdt[6]=x[12];
    dxdt[7]=F[0];
    dxdt[8]=F[1];
    dxdt[9]=F[2];
    dxdt[10]=F[3];
    dxdt[11]=F[5];
    dxdt[12]=F[6];

}
```