

**ANKARA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ**

YÜKSEK LİSANS TEZİ

WAVELET DÖNÜŞÜMÜNE DAYALI DİNAMİK GÜÇ SPEKTRUMU ELDESİ

Mustafa Emre AYDIN

ASTRONOMİ ve UZAY BİLİMLERİ ANABİLİM DALI

**ANKARA
2014**

Her hakkı saklıdır

TEZ ONAYI

Mustafa Emre AYDIN tarafından hazırlanan “**Wavelet Dönüşümüne Dayalı Dinamik Güç Spektrumu Eldesi**” adlı tez çalışması 18/07/2014 tarihinde aşağıdaki jüri tarafından oy birliği ile Ankara Üniversitesi Fen Bilimleri Enstitüsü Astronomi ve Uzay Bilimleri Anabilim Dalı’nda **YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

Danışman : Prof. Dr. Sacit ÖZDEMİR
Ankara Üniversitesi Astronomi ve Uzay Bilimleri Anabilim Dalı

Jüri Üyeleri :

Başkan : Prof. Dr. Altan BAYKAL
Orta Doğu Teknik Üniversitesi, Fizik Anabilim Dalı

Üye : Prof. Dr. Sacit ÖZDEMİR
Ankara Üniversitesi, Astronomi ve Uzay Bilimleri Anabilim Dalı

Üye : Doç. Dr. Kutluay YÜCE
Ankara Üniversitesi, Astronomi ve Uzay Bilimleri Anabilim Dalı

Yukarıdaki sonucu onaylarım

Prof. Dr. İbrahim DEMİR
Enstitü Müdürü

ETİK

Ankara Üniversitesi Fen Bilimleri Enstitüsü tez yazım kurallarına uygun olarak hazırladığım bu tez içindeki bütün bilgilerin doğru ve tam olduğunu, bilgilerin üretilmesi aşamasında bilimsel etiğe uygun davrandığımı, yararlandığım bütün kaynakları atıf yaparak belirttiğimi beyan ederim.

28 / 06 / 2014

Mustafa Emre AYDIN

ÖZET

Yüksek Lisans Tezi

WAVELET DÖNÜŞÜMÜNE DAYALI DİNAMİK GÜÇ SPEKTRUMU ELDESİ

Mustafa Emre AYDIN

Ankara Üniversitesi

Fen Bilimleri Enstitüsü

Astronomi ve Uzay Bilimleri Anabilim Dalı

Danışman: Prof. Dr. Sacit ÖZDEMİR

Gökbilimde gözlenen cismin parlaklık değişiminin analizi, cismin fiziksel parametreleri hakkında kritik bilgiler sunmaktadır. Fourier dönüşümü temeline dayandırılan dönem tespit yöntemleri, 20. yüzyılda bilgisayarların hesaplama gücünün artmasıyla, eşit aralıklarla örneklenmemiş gözlemlere de uyarlanmaya başlanmıştır. Özellikle Lomb-Scargle (LS) yöntemleriyle elde edilen periyodogramlar, sistemin varsa birden fazla frekansını göstermekte oldukça başarılı olmuştur. Günümüzde LS algoritmasının pencerelenmiş aralıklarla veri setleri üzerine uygulanmasıyla dinamik güç spektrumları (DPS) elde edilmeye başlanmıştır. DPS'ler sayesinde, sistemlerin dönem değişimleri çok daha etkileşimli olarak incelenebilmektedir. LS periyodogramları, Fourier dönüşümü gibi trigonometrik fonksiyonlara dayandırılmaktadır. Bu çalışmada, Foster'ın Morlet dalgacıklarına (wavelet) dayanan ağırlıklandırılmış dalgacık z-dönüşümü (WWZ) algoritması için yeni bir Python programı geliştirilmiştir. Programın sonuçları, bazı X-ışın çift yıldızlarının RXTE/ASM verileri üzerinde test edilmiştir ve literatürdeki LS algoritmasına dayalı DPS'ler ile karşılaştırılmıştır.

Temmuz 2014, 129 sayfa

Anahtar Kelimeler: Fourier dönüşümü, Lomb-Scargle, WWZ, RXTE, X-ışın çift yıldızları, Python, wavelet dönüşümü

ABSTRACT

M.S. Thesis

DYNAMIC POWER SPECTRA ON THE BASIS OF WAVELET TRANSFORM

Mustafa Emre AYDIN

Ankara University

Graduate School of Natural and Applied Sciences

Department of Astronomy and Space Sciences

Supervisor: Prof. Dr. Sacit ÖZDEMİR

Studies of the variability of the stars in astronomy serves an important role in acquiring the physical parameters of the celestial objects. Due to the improvements in computation power in the 20th century, period detection methods based on the Fourier transformation have lately been applied to unevenly sampled data too. Especially periodograms obtained by the Lomb-Scargle (LS) methods have been useful to detect multiple frequencies of the system, if any. Lately, the LS algorithm is being popularly applied on the data sets via windowed functions to obtain dynamic power spectrums (DPS). By analyzing DPS, it is possible to study the changes in periodicity of light curves very effectively. In this study, a Python script has been developed by calculating Foster's weighted wavelet Z-transform (WWZ) which is based on Morlet wavelets. The results of the script has been tested on the light curves of some X-ray binary systems obtained by the RXTE/ASM instrument and compared with studies of DPS constructed by the LS algorithms in the literature.

July 2014, 129 pages

Key Words: Fourier transform, Lomb-Scargle, WWZ, RXTE, X-ray binaries, Python, wavelet transform

TEŞEKKÜR

Beni yaşken eğen sevgili ailem Erol AYDIN, Nesrin AYDIN ve Duygu SERT'e, bilimsel heyecanın ve paylaşma dürtüsünün her daim korunması gerektiğini bana örnekleriyle yaşatan Olcay SERT'e, araştırma sürecim boyunca heyecanlı, üzgün, sinirli ve hatta sabırsız anlarımın tamamına alçak gönüllülükle katlanmış sevgili dostum Sefa KELEŞ'e, tez yazım aşamamda yanlışlarımı düzelten, karşılaşılabileceğim problemleri önceden gören ve beni uyararak Deniz ÇOKER'e, iş yoğunluğumdan dolayı üzerine sıklıkla eğilemediğim tez çalışmamı bana örnek olacak bir tavırla yöneten, bu çalışma konusuna beni yönlendiren danışmanım Prof. Dr. Sacit ÖZDEMİR'e (Ankara Üniversitesi Astronomi ve Uzay Bilimleri Anabilim Dalı), üniversite hayatım boyunca beni hocalarımdan çok yetiştirmiş sayısız dostuma ve nasıl bilim yapılmaması gerektiğini farkında olmadan bana göstermiş nice araştırmacıya gönülden teşekkür ederim.

Bu çalışmada, Yüksek Enerji Astrofiziği Bilimsel Arşiv ve Araştırma Merkezi (HEASARC) tarafından sağlanan veriler kullanılmıştır.

Ayrıca LMC X-3 kaynağı için sağladıkları optik verilerden dolayı Dr. C. BROCKSOPP'a ve yüksek enerji (GINGA) verisinden dolayı da Dr. S. KITAMOTO'ya sonsuz teşekkürlerimi sunarım.

Özgür yazılım felsefesi, doğası gereği önce bilim insanlarının özümsemesi gereken bir duruştur. Dilerim bir gün tüm araştırmacılar, bulguları kadar, araçlarını da paylaşacak heyecana ulaşırlar.

"I often compare open source to science. To where science took this whole notion of developing ideas in the open and improving on other peoples' ideas and making it into what science is today, and the incredible advances that we have had. And I compare that to witchcraft and alchemy, where openness was something you didn't do." – Linus Torvalds, 2006

Mustafa Emre AYDIN

Ankara, Temmuz 2014

İÇİNDEKİLER

TEZ ONAY SAYFASI

ETİK.....	i
ÖZET.....	ii
ABSTRACT	iii
TEŞEKKÜR.....	iv
SİMGELER ve KISALTMALAR DİZİNİ	vii
ŞEKİLLER DİZİNİ	ix
ÇİZELGELER DİZİNİ	xi
1. GİRİŞ.....	1
2. KURAMSAL TEMELLER.....	4
2.1 Fourier Dönüşümü	4
2.2 Lomb-Scargle Periyodogramı	10
2.3 Ağırlıklandırılmış Dalgacık Z-Dönüşümü (WWZ).....	12
2.4 Templeton'ın WWZ Programı.....	18
3. MATERYAL ve YÖNTEM.....	22
3.1 Modern Gökbilimde Python'un Önemi	22
3.2 Kod Yazımında Dikkat Edilen Noktalar.....	26
3.3 WWZ Programı.....	28
3.3.1 Kullanım için gerekenler	29
3.3.2 Programın kullanımı.....	29
3.3.3 Modül olarak kullanım	36
3.3.4 Üretilen çıktının düzeni	40
3.3.5 Karşılaştırmalı sonuçlar	42
3.3.6 Verimlilik testleri.....	43
3.3.7 Templeton'ın koduna yapılan katkılar.....	49
3.4 Lomb-Scargle Programı	51
3.5 Seçilen Hedefler ve Veritabanı.....	53
3.5.1 X-ışın çift yıldızlarının genel özellikleri	53
3.5.2 RXTE teleskobunun genel özellikleri	60
4. BULGULAR	62
4.1 RXTE/ASM Verilerinin İncelenmesi ve Karşılaştırılması	62
5. TARTIŞMA ve SONUÇ.....	77
KAYNAKLAR.....	79

EKLER.....	83
EK 1. wwz.py Kaynak Kodu	84
EK 2. ls.py Kaynak Kodu	117
EK. 3 Örnek Gnuplot Kodu	121
EK 4. CentOS 6.x Üzerine Python 2.7.6 Yüklenmesi.....	123
ÖZGEÇMİŞ.....	128

SİMGELER DİZİNİ

Re	Gerçek Bileşen Operatörü
Im	Sanal Bileşen Operatörü
F_c	Kosinüs Dönüşüm Fonksiyonu
F_s	Sinüs Dönüşüm Fonksiyonu
$\mathcal{F}$	Fourier Dönüşümü Operatörü
f_k	Kesikli sinyal örnekleme
F_r	Kesikli Fourier Dönüşümü Operatörü
ν	Frekans
ω	Açısal Frekans
N_0	Örnekleme sayısı
N_{eff}	Pencerelenmiş dalgacık hesabında dikkate alınan örnekleme sayısı
P_x	Lomb-Scargle periyodogramı gücü
P	Ağırlıklandırılmış projeksiyon gücü
τ	Zaman kaydırma miktarı
$f(z)$	Dalgacık Çekirdeği
c	Morlet Dalgacığı sönümlenme katsayısı
$\emptyset_a$	Projeksiyon deneme fonksiyonları
w_α	Projeksiyonda α noktasının istatistiksel ağırlığı
s^2	Varyans
s_w^2	Ağırlıklandırılmış varyans
V_x	Ağırlıklandırılmış varyasyon
V_y	Model fonksiyonun ağırlıklandırılmış varyasyonu
L_{opt}	Cismin optik bölgedeki ışıma gücü
L_X	Cismin X-ışın bölgesindeki ışıma gücü
L_{acc}	Yığılma diskinden açığa çıkan ışıma gücü
K	Kelvin
$M_\odot$	Güneş'in kütlesi
M	Kütle

G	Evrensel kütleçekim sabiti
R	Yarıçap
pc	Parsek

Kısaltmalar

AAVSO	American Association of Variable Star Observers / Amerikan Değişen Yıldız Gözlemcileri Birliği
AoV	Analysis of Variance / Varyans Analizi
CPU	Central Processing Unit / Merkezi İşlem Birimi
DFT	Discrete Fourier Transform / Kesikli Fourier Dönüşümü
DWA	Discrete Wavelet Amplitude / Kesikli Dalgacık Şiddeti
DWT	Discrete Wavelet Transform / Kesikli Dalgacık Dönüşümü
FFT	Fast Fourier Transform / Hızlı Fourier Dönüşümü
GPU	Graphics Processing Unit / Görüntü İşlem Birimi
JD	Julian Date / Jülyen Günü
LS	Lomb-Scargle
MJD	Modified Julian Date / Modifiye Edilmiş Jülyen Günü
PDM	Phase Dispersion Minimization / Evre Saçılımı Minimizasyonu
RXTE	Rossi X-ray Timing Explorer
S/N	Signal-Noise Ratio / Sinyal-Gürültü Oranı
WWT	Weighted Wavelet Transform / Ağırlıklandırılmış Dalgacık Dönüşümü
WWZ	Weighted Wavelet Z-Transform / Ağırlıklandırılmış Dalgacık Z-Dönüşümü
sop	Superorbital Period / Süperorbital Dönem

ŞEKİLLER DİZİNİ

Şekil 2.1	R CVn yıldızının iki veri setinden Fourier dönüşümü	10
Şekil 2.2	Templeton'ın WWZ programının örnek kullanımı	20
Şekil 2.3	VStars programının örnek ekran görüntüsü	21
Şekil 3.1	Python ve R hakkındaki soruların Stackoverflow üzerinde sorulma sayısının zamana artışı	23
Şekil 3.2	GitHub üzerinde oluşturulan Python ve R repo sayısının zamanla artışı	24
Şekil 3.3	wwz.py için kullanılabilecek örnek argüman dosyasının içeriği	35
Şekil 3.4	GnuPlot uyumlu modda oluşturulmuş bir çıktı örneği	41
Şekil 3.5	Bazı test yıldızları için WWZ korelasyonu sonuçları	43
Şekil 3.6	HER X-1 verilerinde c parametresi serbest bırakılarak yapılan verimlilik testi sonuçları	44
Şekil 3.7	Python'un paralel biçimde çalıştırılmasıyla elde edilen süre değişimi	48
Şekil 3.8	Büyük kütleli ve küçük kütleli X-ışın çift yıldızlarının gökadadaki dağılımı	56
Şekil 3.9	Gökadanın üstten görünümünde küçük kütleli ve büyük kütleli X-ışın çift yıldızlarının dağılımı	57
Şekil 3.10	X-ışın çiftlerinden oluşturulmuş pulsasyon dönemleri ve yörüngesel dönemlerinin grafiğini gösteren Corbet diyagramı	58
Şekil 4.1	CYG X-1 yıldızının WWZ analizi	64
Şekil 4.2	CYG X-1 yıldızının Kotze ve Charles tarafından yapılan analizi	64
Şekil 4.3	HER X-1 yıldızının WWZ analizi	65
Şekil 4.4	HER X-1 yıldızının Kotze ve Charles tarafından yapılan analizi	65
Şekil 4.5	CYG X-2 yıldızının WWZ analizi	66
Şekil 4.6	CYG X-2 yıldızının Kotze ve Charles tarafından yapılan analizi	66
Şekil 4.7	LMC X-4 yıldızının WWZ analizi	68
Şekil 4.8	LMC X-4 yıldızının Kotze ve Charles tarafından yapılan analizi	68

Şekil 4.9	SCO X-1 yıldızının WWZ analizi	69
Şekil 4.10	SCO X-1 yıldızının Kotze ve Charles tarafından yapılan analiz	69
Şekil 4.11	SMC X-1 yıldızının WWZ analizi	70
Şekil 4.12	SMC X-1 yıldızının Kotze ve Charles tarafından yapılan analizi	70
Şekil 4.13	X0114+650 yıldızının WWZ analizi	71
Şekil 4.14	X0114+650 yıldızının Kotze ve Charles tarafından yapılan analizi	71
Şekil 4.15	X1636-536 yıldızının WWZ analizi	72
Şekil 4.16	X1636-536 yıldızının Kotze ve Charles tarafından yapılan analizi	72
Şekil 4.17	LMC X-3 yıldızının Kotze ve Charles tarafından yapılan analizi	73
Şekil 4.18	LMC X-3 yıldızının RXTE verilerinin WWZ analizi	74
Şekil 4.19	LMC X-3 yıldızının V bandı optik gözlemlerinin WWZ analizi	75
Şekil 4.20	LMC X-3 yıldızının B bandı optik gözlemlerinin WWZ analizi	75
Şekil 4.21	LMC X-3 yıldızının 1-6 keV aralığında elde edilmiş GINGA verilerinin WWZ analizi	76

ÇİZELGELER DİZİNİ

Çizelge 2.1	Templeton'ın programının girdi parametreleri	19
Çizelge 3.1	wwz.py programının girdi argümanlarının açıklamalı tablosu	32
Çizelge 3.2	Çıktıda kullanılan değerlerin açıklamaları	40
Çizelge 3.3	ls.py programının girdi argümanlarının açıklamalı tablosu	52
Çizelge 4.1	Çalışmada kullanılan hedef cisimlerinin literatürdeki dönem değerleri	62

1. GİRİŞ

Tüm yıldızlar, evrimlerinin belirli aşamalarından geçerken parlaklık ve renk değişimleri geçirirler. Bu parlaklık değişimi, ancak insan ömrüyle kıyaslanacak düzeyde olduğunda söz konusu yıldızlara *değişen yıldız* ismi verilmeye başlanır. Değişen yıldızların parlaklık değişimleri periyodik, yarı-periyodik veya düzensiz olabileceği gibi, dönemleri dakika mertebesinde yüzyıllık bir zaman ölçeğine kadar uzanabilir.

Yıldızların bu ölçülebilir parlaklık değişimleri insanlığın binlerce yıldır farkında olduğu bir olgudur. Porceddu ve arkadaşlarının (2008) yaptığı çalışmada, eski Mısırlıların dinsel sebeplerle Algol yıldızının parlaklık değişimini gözlediği, bunu Kahire Takviminde not ettikleri ortaya çıkmıştır. Jetsu ve arkadaşları (2013) bu takvimden yola çıkarak, M.Ö. 1224 yılında Algol yıldızının dönemini 2850 gün olarak belirlemiştir.

Algol yıldızının dönemi yıllar sonra, 1783'te Goodricke tarafından incelendiğinde 2867 gün olarak tespit edilmiştir ve artık bu tarihten sonra yapılan bütün dönemlik gözlemleri, bilimsel merakla motive olmuştur.

Gökbilimde herhangi bir cismin parlaklık değişiminin incelenmesinde kullanılan ışık eğrileri, ilgili cismin fiziksel parametrelerini belirlemede önemli rol oynamaktadır. İlgili fiziksel parametrelere erişmek için çözülmesi gereken temel problemlerden birisi, söz konusu ışık eğrisindeki dönemlilikleri tespit etmektir.

Bir ışık eğrisinin dönemliğini tespit etmede kullanılan yöntemler bütününe *zaman-serisi analizi* adı verilmektedir. Modern zaman serisi analizi, Joseph Fourier ile 19. yüzyılda başlamıştır. Fourier, önce Fourier Serisi olarak bilinen denklemlerini açıklamış, ardından Fourier Dönüşümünü literatüre katmıştır.

Bugün kullanılan yöntemlerin büyük çoğunluğu, Fourier Dönüşümüne dayansa da, pek çok farklı yaklaşımı da kapsamaktadır. Bunların başında Ayrık Fourier Dönüşümü

(discrete Fourier Transform – DFT) (Deeming 1975), Fourier Dönüşümünün en küçük kareler yöntemiyle yaklaşımı (LS; Lomb 1976; Scargle 1982), dizi uzunluğu (*string length*) (Dworetzky 1983), evre saçılımı minimizasyonu (*phase dispersion minimization* – PDM) (Jurkevich 1971; Stellingwerf 1978), varyans analizi metotları (*analysis of variance*, AoV) (Schwarzenberg-Czerny 1989, 1996) gelmektedir.

Foster (1996) ise Fourier Dönüşümünde temel alınan sinüs fonksiyonlarının yerine, dalgacık denklemlerini veriler üzerinde uyarlayarak, Ağırlıklandırılmış Dalgacık Z-Dönüşümü (*Weighted-Wavelet Z-Transform*, WWZ) adında bir periyot tespit yöntemi geliştirmiştir. Geliştirdiği yöntem, özellikle ağırlıklandırılabilir sonuçlar ürettiğinden, incelenen cismin periyodunun tespiti dışında, periyodun zamana bağlı değişimi ve bu değişimin gücü görülebilmektedir. Bu grafiklere “Dinamik Güç Spektrumu” (DPS) adı verilmektedir. Bu kadar detaylı çıktı ile çalışmak, özellikle değişen yıldızların evrimini incelemekte büyük kolaylıklar sağlamaktadır. Foster örnek bir çalışmayı R Dor yıldızı üzerinde uygulamış, yıldızın JD 2.426.000 ve JD 2.556.000 tarihleri arasında frekans modu değiştirdiğini belirlemiştir (Foster 2010).

Pihajoki vd. (2013) OJ 287 blazarının XMM-Newton teleskobuyla alınmış verilerini incelemiş, Foster’ın geliştirdiği WWZ yöntemiyle 250 ve 50 günlük periyotlar tespit etmişlerdir. Bu blazar için geliştirilen karadelik modellerini destekleyen dönemlilikler bulduklarını duyurmuşlardır.

Percy vd. (2008) kırmızı dev yıldızların fotometrik gözlemleri ile AAVSO (*American Association of Variable Star Observers*, *Amerikan Değişen Yıldız Gözlemcileri Birliği*) gözlemlerini bir arada incelemiş, 8000 günden daha uzun süreli gözlemlerle oldukça anlamlı sonuçlar üretilceğini göstermişlerdir.

Clarkson vd. (2003) STARLINK¹ programının Lomb-Scargle algoritmasını, “kayan pencereler” ile çalışacak biçimde değiştirerek Lomb-Scargle temelli dinamik güç spektrumu elde eden bir yöntem geliştirdiler. Lomb-Scargle yöntemi Fourier Analizi’ne

¹ <http://starlink.jach.hawaii.edu>

dayandığından, sinüs benzeri eğriler fit edilerek çalışmaktadır, dolayısıyla WWZ algoritmasından farklıdır. Ancak tıpkı WWZ gibi, zamana bağlı periyot değişimlerini gösterebilmektedir.

Kotze ve Charles (2012), Clarkson'ın geliştirdiği algoritmayı RXTE (Rossi X-ray Timing Explorer) verileri üzerine uygulamış, 25 adet X-ışın çift yıldızının, Lomb-Scargle yöntemiyle kayan pencerelerden elde edilen dinamik güç spektrumlarını sunmuştur.

Günümüzde yaygın olarak kullanılan WWZ algoritması, Foster'ın yazdığı BASIC kodunun Templeton² (2002) tarafından FORTRAN77 ile yeniden yorumlanmış halidir. Günümüzde diğer temel bilimlerde olduğu gibi gökbiliminde de Python³ kullanımı artış göstermektedir. (bkz. bölüm 3.1) Bu tez çalışması kapsamında Templeton'ın geliştirdiği program, çeşitli revizyonlarla Python koduna tercüme edilmiş; Kotze ve Charles'ın (2012) incelediği hedefler de dâhil olmak üzere, bazı X-ışın çift yıldızlarının RXTE/ASM verisi üzerine uygulanarak dinamik güç spektrumları karşılaştırmalı olarak incelenmiştir.

² <http://www.aavso.org/sites/default/files/software/wwz.tar.gz>

³ <https://www.python.org>

2. KURAMSAL TEMELLER

2.1 Fourier Dönüşümü

Fourier dönüşümü, özetle, bir fonksiyonu farklı frekanslardaki sinüs dalgalarının toplamı şeklinde ifade etmektir. Farklı sinüs dalgalarının, frekanslarını ve genliklerini tanımlamaktadır. Bracewell'i (2000) takip ederek açıklarsak;

$f(x)$ fonksiyonunun Fourier dönüşümü şu şekilde ifade edilir:

$$\int_{-\infty}^{\infty} f(x) e^{-i2\pi xs} dx \quad \dots (1)$$

s 'nin bir dönüşümü olan bu integral $F(s)$ olarak yazılabilir. Aynı denklemlerle $F(s)$ 'nin dönüşümü elde edilirse:

$$\int_{-\infty}^{\infty} f(s) e^{-i2\pi \omega s} ds \quad \dots (2)$$

Eğer $f(x)$, x 'in *çift fonksiyonuysa*, yani $f(x)=f(-x)$ ise, o zaman $f(\omega) = f(x)$ olur.

Eğer $f(x)$, x 'in bir *tek fonksiyonuysa*, yani $f(x)=-f(-x)$ ise, o zaman $f(\omega) = f(-x)$ olur.

Eğer $f(x)$, ne çift fonksiyon ne de tek fonksiyonsa, genellikle çift ve tek kısımlara ayrıştırılabilir olur.

Fourier dönüşümü ve ters dönüşümünün alışıldık biçimleri:

$$F(s) = \int_{-\infty}^{\infty} f(x) e^{-i2\pi xs} dx, \quad f(x) = \int_{-\infty}^{\infty} F(s) e^{-i2\pi xs} ds$$

Bu formda, orijinal fonksiyonu elde etmek için, iki ardışık dönüşüm uygulanır. $F(s)$, $f(x)$ 'in *eksi-i* ($-i$) dönüşümü iken, $f(x)$, $F(s)$ 'nin *artı-i* ($+i$) dönüşümüdür.

İki ardışık dönüşümü, tekrarlanan bir integral olarak yazarsak, Fourier integral teoreminin genel halini elde ederiz:

$$f(x) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x) e^{-i2\pi xs} dx e^{i2\pi xs} ds \quad \dots (3)$$

(3) denkleminin geçerliği için integrallerdeki süreksizliklerin sonlu olması gerekmektedir. Dönüşüm büyüklüğü $F(s)$, bazen $f(s)$ ile, Fourier dönüşümü de bazen $\mathcal{F}$ operatörü ile gösterilmektedir (Bracewell 2000).

Fonksiyonlar ve dalga denklemleri genellikle çift ve tek parçalarına ayrılabilir. $E(x)$ çift fonksiyon, $O(x)$ tek fonksiyon olmak üzere,

$$f(x) = E(x) + O(x) \quad \dots (4)$$

$$E(x) = \frac{1}{2} [f(x) + f(-x)] \quad \dots 5$$

$$O(x) = \frac{1}{2} [f(x) - f(-x)] \quad \dots 6$$

elde edilir. Bu gösterim ile, $f(x)$ 'in Fourier dönüşümü aşağıdaki şekle indirgenir:

$$F(s) = 2 \int_0^{\infty} E(x) \cos 2\pi xs dx - 2i \int_0^{\infty} O(x) \sin 2\pi xs dx \quad \dots (7)$$

Bu sonuçtan, çift fonksiyon dönüşümünün gerçek, tek fonksiyon dönüşümünün sanal (kompleks/imajiner) olduğu anlaşılmaktadır. Bracewell'in (2000) belirttiği gibi, Hermitian fonksiyonların Fourier dönüşümleri çifttir. Hermitian fonksiyonları, gerçek

bileşeni çift, sanal bileşeni tek fonksiyondan oluşan fonksiyonlardır; yani $f(x) = f^*(-x)$ gibi. Dahası, $f(x)$ fonksiyonunun kompleks eşleniğinin Fourier dönüşümü $F^*(-s)$ 'dir, yani dönüşümün eşleniğinin yansımasıdır. Bunun özel durumları şöyle özetlenebilir:

$$\text{Eğer } f(x) \begin{matrix} \text{gerçek} \\ \text{kompleks} \\ \text{çift} \\ \text{tek} \end{matrix} \text{ ise, } f^*(x) \text{ dönüşümü } \begin{matrix} F(s) \\ -F(s) \\ F^*(s) \\ -F^*(s) \end{matrix} = F^*(-s)$$

Pozitif s için bir $f(x)$ fonksiyonunun *kosinüs dönüşümü* $F_c(s)$, şu şekilde tanımlanmaktadır:

$$F_c(s) = 2 \int_0^{\infty} f(x) \cos 2\pi s x \, dx, \quad s > 0 \dots (8)$$

Bu, (7) denkleminde elde ettiğimiz, $f(x)$ fonksiyonunun bir çift fonksiyon olması durumundaki Fourier dönüşümüyle uyumaktadır. Buradan şu sonuca ulaşılabilmektedir; $f(x)$ 'in Fourier dönüşümünün çift fonksiyon tarafı $s > 0$ için, $f(x)$ 'in çift fonksiyon tarafının kosinüs dönüşümüne eşittir.

Kosinüs dönüşümünün ters dönüşümü şu şekilde ifade edilir:

$$f(x) = 2 \int_0^{\infty} F_c(s) \cos 2\pi s x \, ds, \quad x > 0 \dots (9)$$

Benzer şekilde, $f(x)$ 'in *sinüs dönüşümü* $F_s(s)$, ve ters dönüşümü $f(x)$ de şu şekilde tanımlanır:

$$F_s(s) = 2 \int_0^{\infty} f(x) \sin 2\pi s x \, dx, \quad x > 0 \dots 10.a$$

$$f(x) = 2 \int_0^{\infty} F_s(s) \sin 2\pi s x \, ds, \quad x > 0 \dots 10.b$$

Buradan, $f(x)$ 'in Fourier dönüşümünün tek fonksiyonunun i katı, $f(x)$ 'in tek fonksiyon kısmının sinüs dönüşümüne eşit çıkmaktadır.

$f(x)$ fonksiyonunun çift ve tek kısımlarının sinüs ve kosinüs dönüşümlerinin birleştirilmesi, $f(x)$ 'in tamamının Fourier dönüşümünü sağlamaktadır.

$$\mathcal{F} f(x) = \mathcal{F}_c E(x) - i \mathcal{F}_s O(x) \dots (11)$$

Burada $\mathcal{F}$, $\mathcal{F}_c$, $\mathcal{F}_s$ sırasıyla Fourier dönüşümünün $(-i)$ ile çarpımını, kosinüs dönüşümünü ve sinüs dönüşümünü temsil etmektedir. Orijinin sol tarafında $f(x)$ fonksiyonu “sıfır” ise, bu durumda;

$$F(s) = \frac{1}{2} F_c(s) - \frac{1}{2} i F_s(s) \dots 12$$

yazılabilir. $F(s)$ ile gösterilen Fourier dönüşümü $f(x)$ fonksiyonunun frekans bölgesindeki ifadesi olduğundan, s parametresi sinüs ve kosinüs fonksiyonlarının frekansını temsil etmektedir, yani x birim başına çevrime karşılık gelmektedir. Eğer bir dalga fonksiyonu periyodik değilse, o fonksiyonun Fourier dönüşümü, frekansın sürekli fonksiyonu şeklinde olur (Brigham 1988).

Gökbilim gözlemlerinden elde edilen veriye Fourier dönüşümü uygulanabilmesi için, **(1)**

denklemini aşağıdaki biçimde revize edilebilir:

$$F(v) = \int_{-\infty}^{\infty} x(t) e^{-i2\pi vt} dt \dots (13)$$

Burada v frekans, $x(t)$ ise elde ettiğimiz zamana bağlı gözlem verisidir. Ancak Fourier dönüşümünün kullanılarak dönem tespit edilmesinde bir takım sınırlamalar bulunmaktadır. Tabii ki tespit edilebilecek en uzun dönem, gözlem süresi kadardır,

ancak bunun kesinlikle güvenilir olmadığı göz ardı edilmemelidir. Tespit edilebilecek en kısa dönem (Nyquist frekansı) ise, gözlem sıklığına bağlıdır. Gözlem sıklığına *örneklem* denir ve anlamlı dönem tespiti için kısıtlamalar getirir.

Herhangi bir $f(t)$ sinyalinin, B Hz'in altında hiçbir frekans değerinin olmaması, yani $s > 2\pi B$ için $F s = 0$ olması durumunda, $f(t)$ sinyaline, *bant-sınırlı sinyal* (*bandlimited signal*) adı verilir. *Örneklem teoremi*, B Hz'e sınırlandırılmış bir $f(t)$ sinyalinin, saniyede $R > 2B$ örneğin eşit aralıklarla alınarak hatasız olarak yeniden oluşturulabileceğini ifade etmektedir. Bu tanımda oluşan $2B$ Hz değerindeki minimum örneklem frekansına, *Nyquist oranı* veya *Nyquist frekansı* denir. Bu durumda toplam gözlem süresi T olmak üzere, bir gözlem verisinden tespit edilebilecek en küçük frekans değeri Nyquist frekansının yarısı kadar olabilir. Ancak örneklem aralıkları eşit zamanlara bölünmemişse, Nyquist frekansının altında frekans değerlerini tespit etmek mümkün olmaktadır (Foster 2010).

Parseval teoremi, $h(t)$ ile ifade edilen bir sinyalin gücünün, sinyal uzayında veya dönüşüm uzayında hesaplanmasından bağımsız olarak, aşağıdaki şekilde hesaplandığını ifade eder:

$$\int_{-\infty}^{\infty} h^2(t) dt = \int_{-\infty}^{\infty} H(v)^2 dv \quad \dots (14)$$

Buradan hareketle, *güç spektrumu* $P(v)$ şöyle ifade edilir:

$$P(v) = H(v)^2, -\infty \leq v \leq +\infty \quad \dots (15)$$

Gökbilimde zaman-parlaklık uzayındaki ışık eğrisinden dönem elde etmek amacıyla yapılan Fourier dönüşümünde, Parseval teoremi karşımıza dönüşümün periyodogramındaki *güç* eksenini olarak çıkmaktadır. Şekil 2.1'de kullanılan örneklem değerine bağlı olarak tespit edilen dönemin güç değerinin nasıl dağıldığı gösterilmiştir. Şekil 2.1'de üstte AAVSO görsel gözlemleri JD2420232-2452906 (32674 gün) aralığında incelenmiş, alttaysa AAVSO görsel gözlemleri JD2449909-2452906 (2997

gün) aralığında incelenmiştir. Her iki periyodogramda da maksimum güç değeri yakın dönem aralığını gösterse de, 329 günlük pulsasyon periyodu daha uzun veri setiyle yapılan analizde daha iyi görülebilmektedir.

Kullanılan denklemlerde Fourier dönüşümü analitik olarak ifade edilmiştir. Ancak bilgisayarlar kesikli verilerle çalıştıklarından, Fourier dönüşümünü nümerik olarak ifade etmek gerekmektedir.

$f(t)$ sinyalinin kesikli örneklemeleri f_k olsun. Bu durumda $F(s)$ Fourier dönüşümü sadece s kesikli verileri üretecektir. Bu kesikli dönüşüm F_r olmak üzere, eğer $f(kT)$ ve $F(rs_0)$ sırasıyla $f(t)$ ve $F(s)$ 'nin k 'inci ve r 'inci elemanlarıysa, ayrıca N_0 , sinyalin bir T_0 dönemlik zaman dilimindeki örneklem sayısıysa,

$$f_k = T f(kT) = \frac{T_0}{N_0} f(kT) \quad \dots \quad 16 \quad \text{ve} \quad F_r = F(rs_0) \quad \dots \quad (17)$$

elde edilir. Burada

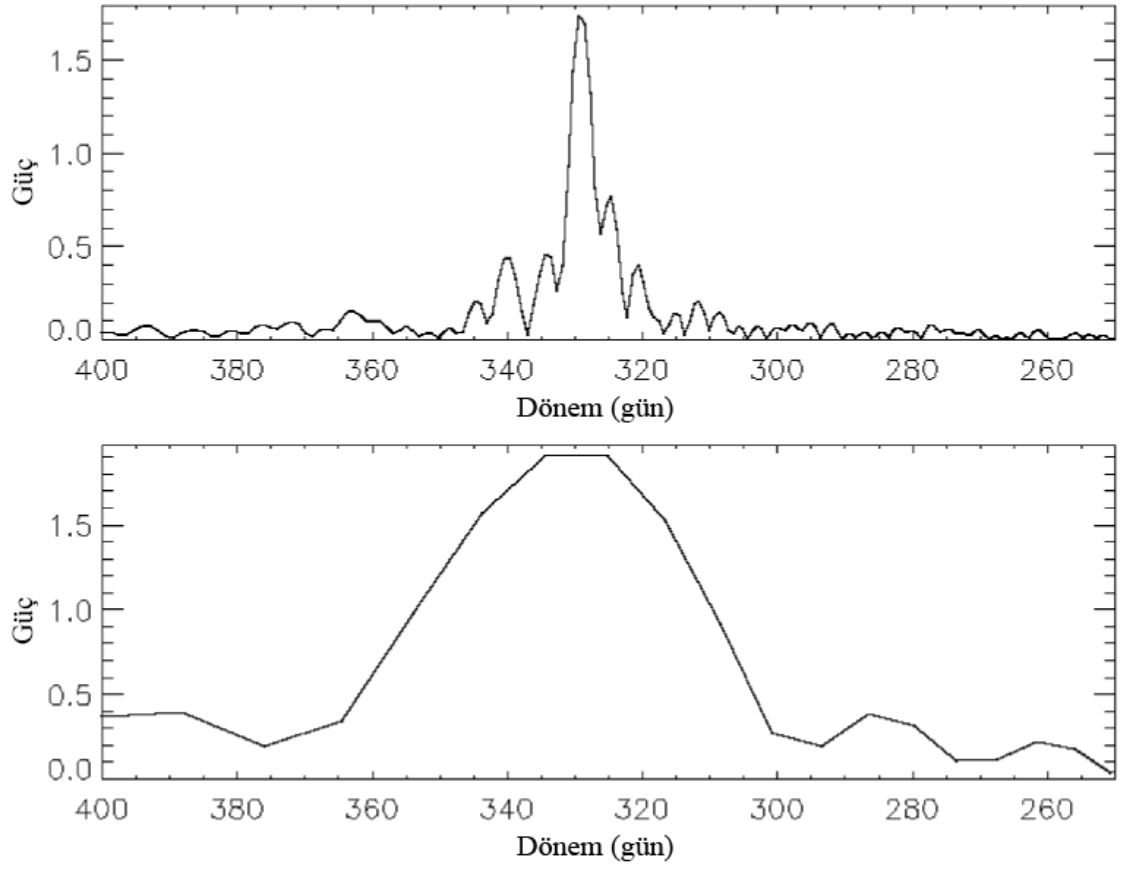
$$s_0 = 2\pi F_0 = \frac{2\pi}{T_0} \quad \dots \quad 18$$

olduğundan, F_r kesikli dönüşümü ve tersi şu şekilde ifade edilir.

$$F_r = \sum_{k=0}^{N_0-1} f_k e^{-ir \frac{2\pi}{N_0} k} \quad \dots \quad (19)$$

$$f_k = \frac{1}{N_0} \sum_{r=0}^{N_0-1} F_r e^{ir \frac{2\pi}{N_0} k} \quad \dots \quad (20)$$

Bu şekilde elde edilen F_r dönüşümüne, *kesikli Fourier dönüşümü (discrete Fourier transform – DFT)* adı verilir (Lathi 1992).



Şekil 2.1 R CVn yıldızının iki veri setinden elde edilmiş olan Fourier dönüşümü (Templeton 2004)

DFT Fourier dönüşümünün çözümünde bir frekans değerinin katları kullanıldığından, eldeki sinyalin eşit aralıklarla örneklenmiş olması gerekmektedir. Benzer problem, DFT'nin daha hızlı bir çözümü olan *hızlı Fourier dönüşümü* (*fast Fourier transform – FFT*) için de geçerlidir (Foster 2010). Gökbilim çalışmalarında gözlemlerin hiçbir zaman eşit aralıklarla örneklenmemiş olması, bu yöntemlerin kullanılabilmesinde sorun teşkil etmektedir.

2.2 Lomb-Scargle Periyodogramı

Kesikli Fourier dönüşümünün eşit aralıklarla örneklenmemiş verilere uyarlanmış hali, Lomb (1976) tarafından geliştirilmiştir. Bu yöntemde, $\omega \equiv 2\pi\nu$ olmak üzere, eşit

aralıklarla örneklenmemiş $\{X(t_i), i = 1, 2, \dots, N_0\}$ veri setinin Fourier dönüşümü aşağıdaki şekilde ifade edilmiştir:

$$F_x(\omega) = \sum_{j=1}^{N_0} X(t_j) e^{-i\omega t_j} \dots (21)$$

Buradaki toplama işlemi, (19) denklemindeki farklı olarak $j=1$ ile N_0 değerleri arasında gerçekleştirilmektedir. Bu, bir dizinin ifadesinde C programlama dili ile FORTRAN programlama dili arasındaki notasyon farklılığından kaynaklanmaktadır. FORTRAN’da bellekteki dizilerin ilk elemanına “1. eleman” olarak ulaşıldığından Lomb bu şekilde ifade etmiş, C dilinde bu eleman “0. eleman” olarak ifade edildiğinden, Bracewell (2000) matematiksel gösterimi bu şekilde yapmıştır. Literatürdeki kullanımlarına sadık kalınması açısından, Lomb’un ifadeleri kendi FORTRAN notasyonu ile gösterilmiştir.

Lomb’un tanımladığı Fourier dönüşümünün gücü ise, aşağıdaki şekilde olacaktır.

$$\begin{aligned} P_x(\omega) &= \frac{1}{N_0} |F_x(\omega)|^2 \\ &= \frac{1}{N_0} \sum_{j=1}^{N_0} X(t_j) e^{-i\omega t_j} \sum_{k=1}^{N_0} X(t_k) e^{i\omega t_k} \\ &= \frac{1}{N_0} \sum_{j=1}^{N_0} X_j^2 \cos^2 \omega t_j + \frac{1}{N_0} \sum_{j=1}^{N_0} X_j^2 \sin^2 \omega t_j \dots (22) \end{aligned}$$

Scargle (1982) Lomb’un (22) denklemini kendi deyimiyle “biraz değiştirmiş” ve trigonometrik fonksiyonlara aşağıdaki gibi bir kayma parametresi (τ) eklemiştir.

$$P_x(\omega) = \frac{1}{2} \left[\frac{\sum_{j=1}^{N_0} X_j^2 \cos^2 \omega(t_j - \tau)}{\sum_{j=1}^{N_0} \cos^2 \omega(t_j - \tau)} + \frac{\sum_{j=1}^{N_0} X_j^2 \sin^2 \omega(t_j - \tau)}{\sum_{j=1}^{N_0} \sin^2 \omega(t_j - \tau)} \right] \dots (23)$$

Burada τ şu şekilde tanımlanmıştır;

$$\tan 2\omega\tau = \frac{\sum_{j=1}^{N_0} \sin 2\omega t_j}{\sum_{j=1}^{N_0} \cos 2\omega t_j} \dots (24)$$

Scargle'in yöntemi genişletmesinden sonra, oluşan bu periyodogram üretme yöntemine literatürde Lomb-Scargle (LS) algoritması adı verilmiştir. (23) ve (24) denklemleri ile verilen LS algoritması, hesaplama gücü oldukça yüksek olan bilgisayarl gerektirir. Bu problemi aşabilmek için Townsend (2010) grafik işlemci birimi (graphical processing unit – GPU) kullanarak, hesaplamaların hızlanması için NVIDIA firmasının CUDA teknolojisini kullanan bir C kodu yazmıştır. Bir Python kütüphanesi olan SciPy ise bu algoritmayı kullandığından, bu tez çalışmasında LS güç spektrumları Townsend'in yöntemiyle elde edilmiştir. SciPy kütüphanesindeki algoritma, Townsend'in algoritmasının GPU'dan işlemciye (CPU) uyarlanmış halidir.

2.3 Ağırlıklandırılmış Dalgacık Z-Dönüşümü (WWZ)

Foster (1996) dönem tespiti için Fourier dönüşümlerindeki gibi trigonometrik fonksiyonlar kullanmak yerine, dalgacık (wavelet) fonksiyonları kullanarak verinin dar pencere aralıklarında incelenip, sonuçların ağırlıklandırılarak dinamik güç spektrumu elde edilmesini sağlamıştır. Bu yöntemle “Ağırlıklandırılmış Dalgacık Dönüşümü” (*Weighted-Wavelet Z-Transform*) denilmektedir.

Foster'ın dalgacık dönüşümü tanımı, Grossman ve arkadaşlarının (1989) dalgacık dönüşümüne dayanmaktadır:

$$\begin{aligned} W(\omega, \tau; x, t) &= \omega^{-1/2} \int_{-\infty}^{\infty} x(t - \tau + z) f^*(z) dz \\ &= \omega^{-1/2} \int_{-\infty}^{\infty} x(\omega^{-1}z + \tau) f^*(z) dz \dots (25) \end{aligned}$$

Burada $x(t)$ zamana bağılı sinyal (gözlem) verisi, ω frekans değeri, $f(z)$ ise dönüşümde uygulanacak dalgacık çekirdeği (*wavelet kernel*) veya ana dalgacıktır (*mother wavelet*). Dalgacık çekirdeğini $z=0$ komşuluğunda yoğunlaşacak şekilde seçerek, $x(t)$ 'nin $t = \tau$ komşuluğundaki davranışı incelenebilmektedir.

Seçilecek dalgacık çekirdeğini, hemen her türlü matematiksel fonksiyon oluşturabilir. Fourier dönüşümünde (25) denklemi, $f(z) = e^{iz} = e^{i\omega(t-\tau)}$ şeklinde bir dalgacık çekirdeğinin uygulandığı, ancak integralde ω^2 katsayısının dikkate alınmadığı özel bir halidir (Foster 1996).

Foster (1996), WWZ algoritmasını oluştururken, dalgacık çekirdeği olarak *Morlet dalgacığı*'nı (Grossman ve Morlet, 1984) kullanmaktadır:

$$\begin{aligned} f(z) &= e^{-cz^2} e^{iz} - e^{-1/4c} \\ &= e^{-c\omega^2(t-\tau)^2} e^{i\omega(t-\tau)} - e^{-1/4c} \dots (26) \end{aligned}$$

(26) denkleminde $e^{-1/4c}$ ile ifade edilen sabit, analiz edilen dalgacığın ortalama değerinin sıfır olmasını sağlamaktadır. Yani;

$$\int_{-\infty}^{+\infty} f(z) dz = 0 \dots (27)$$

olmaktadır. Buradan elde edilen c sabiti, dalgacığın ne kadar *hızlı sönümlendiğinin* ifadesidir; çoğunlukla tek bir $2\pi/\omega$ çevriminde hızla azalacak biçimde seçilmektedir. Değişen yıldızların ışık eğrilerinin çalışılmasında, c sabiti için $1/8\pi^2$ sıklıkla tercih edilen bir değerdir (Foster 1996). $c=0.0125$ bu değere oldukça yakındır, ancak $c<0.02$ olduğu durumlarda $e^{-1/4c}$ göz ardı edilebilecek kadar küçük olmaktadır (Grossman 1989). Buradan hareketle Foster, *kısaltılmış Morlet dalgacığı*nı tanımlamıştır.

$$\begin{aligned} f(z) &= e^{iz-cz^2} \\ &= e^{i\omega(t-\tau) - c\omega^2(t-\tau)^2} \dots (28) \end{aligned}$$

Kesikli zaman serilerine uygulanan Fourier analizi dönüşümüne *kesikli Fourier dönüşümü* denmesi gibi, kısaltılmış Morlet dalgacığının kesikli zaman serilerine uygulanmasına da *kesikli dalgacık dönüşümü* (*discrete wavelet transform – DWT*) adı verilmektedir. Pencerelemiş Fourier dönüşümü, kısaltılmış Morlet dönüşümüyle benzerlik göstermektedir. Temel fark, üstel ifadedeki ω parametresidir. Şöyle ki, kısaltılmış Morlet dönüşümünde kullanılan pencere büyüklüğü frekansa bağlı olmaktadır (Foster 1996).

DWT'yi Foster (1996) aşağıdaki biçimde tanımlamaktadır:

$$W(\omega, \tau; x, t) = \overline{\omega} \sum_{\alpha=1}^N x(t_\alpha) f^*(\omega t_\alpha - \tau) \dots (29)$$

Kısaltılmış Morlet dalgacılığı dönüşümünde, DWT'nin gerçek (*Re*) ve sanal (*Im*) kısımları, (7) ve (28) denklemlerinin yardımıyla şu şekilde gösterilir:

$$Re W = \overline{\omega} \sum_{\alpha=1}^N x(t_\alpha) e^{-c\omega^2 t_\alpha - \tau^2} \cos(\omega t_\alpha - \tau) \dots 30$$

$$Im W = - \overline{\omega} \sum_{\alpha=1}^N x(t_\alpha) e^{-c\omega^2 t_\alpha - \tau^2} \sin(\omega t_\alpha - \tau) \dots 31$$

Foster (1996), (29), (30), (31) denklemleriyle yapay olarak türetilmiş dalgaların frekans tespitinde problemler olduğunu göstermiştir. Frekansların %1.25'lik kayma göstermesi göze çarpan problemlerdendir. Foster bunun bir şekilde giderilmesinin mümkün olduğunu ifade etse de, özellikle eşit aralıklarla örneklenmemiş verilerde doğru frekansın tespit edilebilmesi için, girdi frekansının gerçek frekansa çok yakın olması gerekmektedir (Foster 1996).

Kısaltılmış Morlet dalgacığını DWT'ye uyarlayabilmek için Foster (1996), denklemlerin projeksiyonlarının hesaplanmasını önermektedir. Projeksiyonda, $\phi_a(t)$, $a = 1, 2, \dots, r$ olmak üzere r tane *deneme fonksiyonu* için y_a uyum katsayıları

hesaplanır. Elde edilen fonksiyon ve katsayılarla, veriye en iyi uyan bir $y(t)$ model fonksiyonu oluşturulur:

$$y(t) = \sum_a y_a \phi_a(t), \quad a = 1, 2, 3, \dots, r \quad \dots (32)$$

$f(t)$ ve $g(t)$ herhangi iki fonksiyon olmak üzere, projeksiyon hesaplamasında aşağıdaki gibi tanımlanan iç çarpım kullanılır:

$$f \cdot g = \frac{\sum_{\alpha=1}^N w_{\alpha} f(t_{\alpha}) g(t_{\alpha})}{\sum_{\beta=1}^N w_{\beta}} \quad \dots (33)$$

Burada α veri noktasının istatistiksel ağırlığı w_{α} ile gösterilir. Çoğunlukla istatistiksel ağırlıklar eşit alınabilir (Foster 1996).

(32)'de kullanılan deneme fonksiyonlarının iç çarpımı, (33)'teki tanımı gereğince yapılırsa, S -matrisi şöyle elde edilir:

$$S_{ab} = \phi_a \cdot \phi_b \quad \dots (34)$$

Deneme fonksiyonlarının uyum katsayılarını, S -matrisinin tersinin, deneme fonksiyonunun veriyle iç çarpımıyla çarpılmasından elde edilir.

$$y_a = \sum_b S_{ab}^{-1} \phi_b \cdot x \quad \dots (35)$$

Ayrıca Foster (1996), dönüşümün gücünü (14) ve (15)'te verilen Parseval teoreminden farklı olarak, aşağıdaki gibi tanımlamaktadır:

$$P = \frac{N}{(r-1)s^2} \sum_{a,b} S_{ab}^{-1} \phi_a \cdot x \cdot \phi_b \cdot x - 1 \cdot x^2 \quad \dots (36)$$

Burada N verinin sayısı, s^2 ise varyansdır. Deneme fonksiyonu olarak kullanılabilecek ilk fonksiyonlar, (30) ve (31)'den yola çıkarak aşağıdaki gibi olmalıdır:

$$\phi_1(t) = 1 \quad t = 1 \dots 37$$

$$\phi_2(t) = e^{-c\omega^2(t-\tau)} \cos \omega t \quad \dots \quad 38$$

$$\phi_3(t) = e^{-c\omega^2(t-\tau)} \sin \omega t \quad \dots \quad 39$$

(29), (30), (31) denklemlerinde karşılaşılan frekans kayması problemine projeksiyon ile çözüm, denklem (28) ile verilen kısaltılmış Morlet dalgacığının, (37), (38) ve (39) deneme fonksiyonları şeklinde uygulanırken ağırlıklandırılmasıyla sağlanmaktadır (Foster 1996).

Öyleyse $\phi(t) = e^{i\omega(t-\tau) - c\omega^2(t-\tau)^2}$ şeklindeki bir denklem, aslında iki parçaya bölünebilir.

$$\phi(t) = e^{i\omega(t-\tau)} \quad \dots \quad (40)$$

$$w_\alpha = e^{-c\omega^2(t-\tau)^2} \quad \dots \quad (41)$$

Burada w_α , $\phi(t)$ şeklindeki bir deneme fonksiyonunun istatistiksel ağırlığıdır (Foster 1996).

(33)'te tanımlanan iç çarpım ile projeksiyon hesaplanabilir. Ancak ağırlıklandırılmış projeksiyon ile (36) yardımıyla hesaplanacak güç tanımında, veri sayısı olan N yerine, etkin veri sayısı olan N_{eff} kullanılmalıdır (Foster 1996).

$$N_{eff} = \frac{(\sum w_\alpha)^2}{(\sum w_\alpha^2)} = \frac{e^{-c\omega^2(t-\tau)^2}}{e^{-2c\omega^2(t-\tau)^2}} = \frac{n^2(\omega, \tau)}{n(\bar{2}\omega, \tau)} \quad \dots \quad (42)$$

Bu tanımların ardından, aşağıdaki deneme fonksiyonlarına, (41)'de tanımlanan istatistiksel ağırlıklandırma yardımıyla, ağırlıklandırılmış projeksiyonlar uygulanır.

$$\emptyset_1 t = 1 t \dots 43$$

$$\emptyset_2 t = \cos \omega t - \tau \dots 44$$

$$\emptyset_3 t = \sin \omega t - \tau \dots 45$$

(36)'da yapılan tanımda, N_{eff} kullanılarak güç hesaplanır. Ancak s varyans değeri yerine, *ağırlıklandırılmış s* olan, s_w kullanılır:

$$s_w^2 = \frac{N_{eff} V_x}{N_{eff} - 1} \dots (46)$$

Burada V_x verinin *ağırlıklandırılmış varyasyonudur*.

$$V_x = \frac{\alpha w_\alpha x^2(t_\alpha)}{\lambda w_\lambda} - \frac{\alpha w_\alpha x t_\alpha}{\lambda w_\lambda}^2 = x x - 1 x^2 \dots (47)$$

Benzer şekilde, model fonksiyonun *ağırlıklandırılmış varyasyonu* da şöyle tanımlanır.

$$V_y = \frac{\alpha w_\alpha y^2(t_\alpha)}{\lambda w_\lambda} - \frac{\alpha w_\alpha y t_\alpha}{\lambda w_\lambda}^2 = y y - 1 y^2 \dots (48)$$

Ardından Foster'ın (1996) *ağırlıklandırılmış dalgacık dönüşümü (weighted wavelet transform – WWT)* tanımı gelmektedir:

$$WWT = \frac{(N_{eff}-1)V_y}{2V_x} \dots (49)$$

ω ve τ parametrelerinin sabit değerler olması halinde, WWT, serbestlik derecesi 2 olan ve beklenen değeri 1 olan bir χ^2 istatistiği olarak algılanabilir. Ancak bu durum sadece verinin rastgele gürültüden oluştuğu durumlarda oluşmaktadır. Sinüs benzeri bir sinyal için, WWT'nin sinyal frekansında maksimum değer vermesi beklenir. Ancak daha düşük bir frekansta bu değeri vermektedir. Bunun temel nedeni, düşük frekanslarda, kaydırılan “pencere”nin daha geniş olması, dolayısıyla etkin olarak incelediğimiz veri

noktasının fazla olması, yani N_{eff} değerinin büyük olmasıdır. Bu durum, veriye uydurulan fit değeri düşük olsa bile, ω 'nın azalması durumunda WWT'nin artmasına yol açmaktadır, çünkü (49)'daki N_{eff} değeri büyümektedir. Bunun için etkin veri sayısına daha az duyarlı bir istatistiksel test yöntemine ihtiyaç duyulmaktadır. Bu amaçla Foster (1996), projeksiyonlar üzerine geliştirdiği bir istatistiksel yöntem olan *Z-istatistiği*'ni (*Z-statistics*) uygulamaktadır. Elde edilen sonuca da *ağırlıklandırılmış dalgacık Z-dönüşümü* (*weighted wavelet Z-transform*, WWZ) ismi verilmektedir (Foster 1996):

$$Z = \frac{(N_{eff} - 3)V_y}{2(V_x - V_y)} \dots (50)$$

WWZ ile WWT'nin oranı incelendiğinde, aşağıdaki sonuç ortaya çıkmaktadır.

$$\frac{Z}{(WWT)} = \frac{(N_{eff} - 3)V_x}{(N_{eff} - 1)(V_x - V_y)} = \frac{N_{eff} - 3}{N_{eff} - 1} 1 + (S/N) \dots (51)$$

Burada (S/N), sinyal-gürültü oranını ifade etmektedir. S/N 'nin 1'den çok küçük olduğu durumlarda bile yüksek miktarda veri (N) sayesinde WWZ ile WWT arasında fark olmamaktadır. Bu yüzden düşük S/N 'li verilerde WWZ de WWT'nin dezavantajlarını paylaşmaktadır. Ancak $S/N \geq 1$ olduğu durumlarda WWZ, WWT'nin çok iyileştirilmiş bir halidir (Foster 1996).

2.4 Templeton'ın WWZ Programı

Templeton (2002) AAVSO'nun (American Association of Variable Star Observers) internet sayfasında⁴ Foster'ın (1996) WWZ algoritmasının FORTRAN kodunu yayınlamıştır. Bu tez çalışmasında WWZ'nin Python karşılığı yazılırken, Templeton'ın FORTRAN kodu referans alınmıştır.

⁴ <http://www.aavso.org/sites/default/files/software/wwz.tar.gz>

Templeton'ın kodu FORTRAN77 ile yazıldığından, okunabilirliği ve esnekliği, modern yazılımlarla kıyaslanamayacak kadar kötü olsa da, FORTRAN'ın düşük-seviyeli bir dil olması nedeniyle yüksek hızla çalışmaktadır.

Program komut satırından çalışmakta ve interaktif olarak argüman almaktadır, yani çalıştırıldıktan sonra kullanıcının standart girdi aracılığıyla parametreleri girmesi beklenmektedir. Girilmesi istenilen parametreler çizelge 2.1'de verilmiştir. Programın kullanımının örneği şekil 2.2'de görülebilir.

Çizelge 2.1 Templeton'ın programının girdi parametreleri

İstenilen parametre	Türkçe açıklaması
input data filename	Girdi dosya adı
output wwz filename	Çıktı dosya adı
low frequency (cyc/d)	Alt frekans sınırı (çevrim/gün)
high frequency	Üst frekans sınırı
delta f	Frekans adım büyüklüğü
input decay constant c	Sönümlenme katsayısı

Girdi parametrelerinin verilmesinden sonra program çalışmakta ve iki adet çıktı dosyası üretmektedir. Bunlardan birisi hesaplanan WWZ değerlerini sunmaktadır ve girdi parametrelerinde sunulan “*output wwz filename*” dosya adına yazılmaktadır. Diğer ise kullanıcıya seçenek bırakmadan, doğrudan “*wwzper.dat*” isimli bir dosyaya yazılmaktadır.

WWZ değerlerinin yazıldığı dosya çeşitli başlık bilgileri içermektedir. İlk satırda girdi dosyasının adı, girdi dosyasındaki nokta sayısı, ortalama parlaklık, standart sapma ve varyans gibi değerler yazmaktadır. İkinci satırda, hesaplamaların gerçekleştirildiği tarih aralığının Jülyen Günü (JD) cinsinden değeri ve adım büyüklüğü belirtilmektedir.

Üçüncü satırda ise hesaplamaların gerçekleştirildiği düşük ve yüksek frekans değeri sınırları ile frekans adım büyüklüğü belirtilmiştir. Dördüncü satırdaysa, çıktı olarak üretilen değerlerin başlıkları yazmaktadır.

```
emre@pavilion: ~/Dropbox/astronomy/thesis/programs/fortran
emre@pavilion:~/Dropbox/astronomy/thesis/programs/fortran$ ./a.out

Weighted Wavelet Z-transform (WWZ)
(C) Copyright 1996, 2002 by the American Association
of Variable Star Observers; all rights reserved.

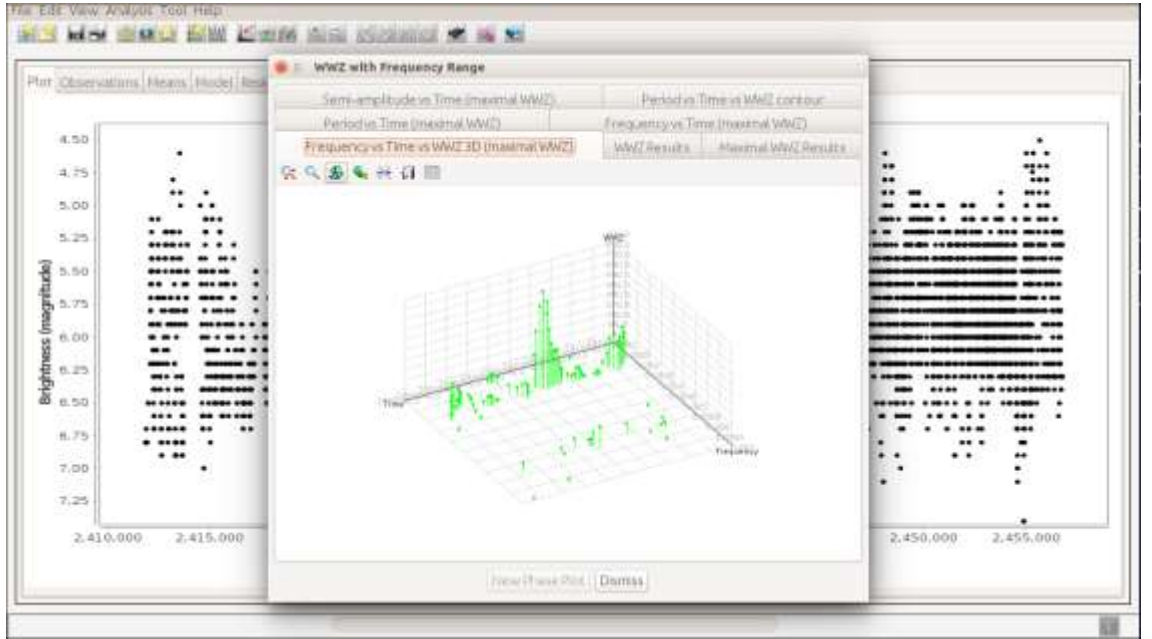
input data filename
eudel.dat
output wwz filename (e.g. name.wwz)
eudel.cikti
low frequency (cyc/d)
0.002
high frequency
0.01
delta f
0.0001
input decay constant c
0.0125
Program complete!
emre@pavilion:~/Dropbox/astronomy/thesis/programs/fortran$ █
```

Şekil 2.2 Templeton'ın WWZ programının örnek kullanımı

Çıktılar, incelenen zaman değeri (TAU), incelenen frekans değeri (FREQ), hesaplanan WWZ değeri (bkz. (50) denklemi), hesaplanan ağırlıklandırılmış dalgacık şiddeti değeri, (35) denklemi ile ifade edilen en-iyi-fit sabiti değeri (COEF) ve (42) denklemi ile ifade edilen, projeksiyonda kullanılan değer sayısı (NEFF) şeklindedir.

Üretilen diğer çıktı dosyası olan *wwzper.dat* dosyası, hiçbir başlık bilgisi içermemektedir. Bu dosya, WWZ çıktılarında, her TAU değerine karşılık gelen maksimum WWZ istatistiğine sahip frekansa karşılık gelen sonuçları içermektedir.

Templeton'ın (2002) kodundan faydalanılarak yapılan bir diğer tercüme, AAVSO ekibinin VStars isimli programıdır. Java ile yazılmış olan bu program, aslında değişen yıldız analizi için AAVSO verilerine yönelik geliştirilmiş olduğundan, modülerlik veya komut satırından otomatik kullanım gibi seçenekler sunmamaktadır. Program, WWZ, CLEANest, Kesikli Fourier Dönüşümü, Polinom fiti gibi pek çok analiz yöntemini bir araya getirmiş gelişmiş bir arayüze sahiptir. Örnek ekran görüntüsü şekil 2.3'te görülebilir.



Şekil 2.3 VStars programının örnek ekran görüntüsü

3. MATERİYAL ve YÖNTEM

Bu tez çalışmasında Foster'ın (1996) geliştirmiş olduğu WWZ algoritması, Python programlama diliyle yeniden yazıldı. Bu yazım sırasında, diğer tercümeleler gibi Templeton'ın FORTRAN kodu referans alındı. Templeton'ın kodunda rastlanan problemler, kısıtlamalar göz önünde bulundurularak geliştirilmeye veya giderilmeye çalışıldı, bunun yanı sıra kodda rastlanan bir takım algoritma hataları veya gereksiz noktalar giderilmeye çalışıldı.

Ayrıca geliştirilen WWZ programının çıktıları daha iyi sunabilmek amacıyla, SciPy modülünde yer alan Lomb-Scargle periyodogram analiz algoritmasının daha rahat kullanılabilmesi için bir program daha yazıldı.

3.1 Modern Gökbilimde Python'un Önemi

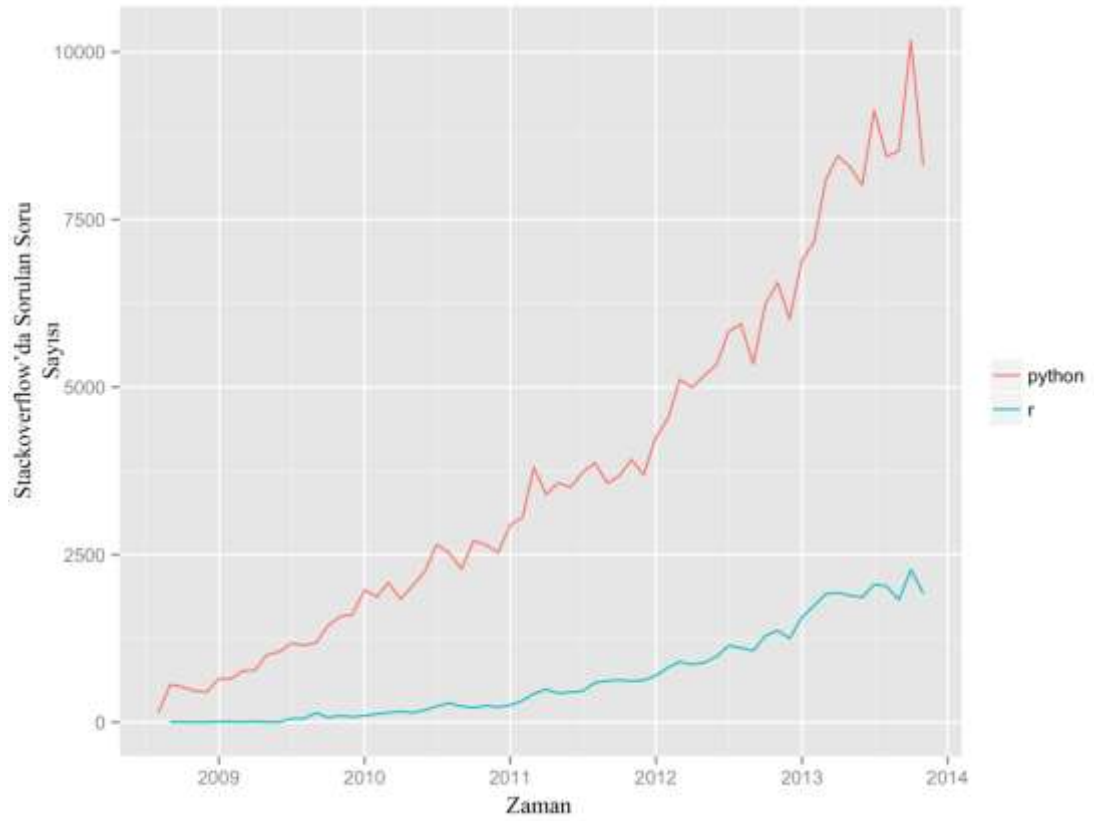
Bugün bütün Dünya'da temel bilimcilerin büyük çoğunluğu programlama dili olarak Python'u tercih etmektedir. Bu özellikle büyük miktarda verinin incelenmesi gereken bilim dallarında başlayan bir geçiş olmuştur. Bu bakımdan İstatistik Bilimleri çalışan kişiler, hala tam olarak Python'un R'nin⁵ yerini tutabildiğine inanmasa da, hızlı bir geçiş süreci başlamıştır. Konu hakkında istatistikçi Matt Asay, 25 Kasım 2013 tarihli makalesinde⁶ söz konusu geçişi “R'yi aslında bir programlama dili değil, istatistiksel hesaplama yapmaya yarayan etkileşimli bir ortam” olarak tanımlıyor. Bu durum, İstatistikçilerin programlama ihtiyaçlarını karşılamada R'nin her ne kadar geniş kütüphane desteği olsa da, yetersiz kaldığı sonucunu doğuruyor. Özellikle finans verilerinin analizinde çalışan kişilerin R'den Python'a geçişi, pandas⁷ (Python Data Analysis Library) sayesinde hızlanmış durumda.

⁵ <http://www.r-project.org>

⁶ <http://readwrite.com/2013/11/25/python-displacing-r-as-the-programming-language-for-data-science>

⁷ <http://pandas.pydata.org>

Her ne kadar veri işleme amacıyla Python'un bugün R'den daha üstün konumda olduğu kabul görse de, R'nin de Python'a paralel seviyelerde büyüme gösterdiği ve pek çok kütüphane desteği geldiği göz ardı edilmemelidir. Konu hakkında David Smith'in 6 Aralık 2013'te yayınladığı makalesinde⁸ paylaştığı fikirler bunu destekler niteliktedir. Smith'in makalesinin temel noktası, Hadley Wickham'ın GitHub⁹ üzerinde yazdığı “R and Python Usages” makalesine¹⁰ dayanmaktadır. Bu çalışmasında Wickham, özellikle Stackoverflow¹¹ üzerindeki Python ve R sorularının aylara göre artışını irdelemiştir. Sonuçlara bakıldığında, her ne kadar Python açık ara önde olsa da, R'nin popülaritesi göz ardı edilmemelidir.



Şekil 3.1 Python ve R hakkındaki soruların Stackoverflow üzerinde sorulma sayısının zamanla artışı

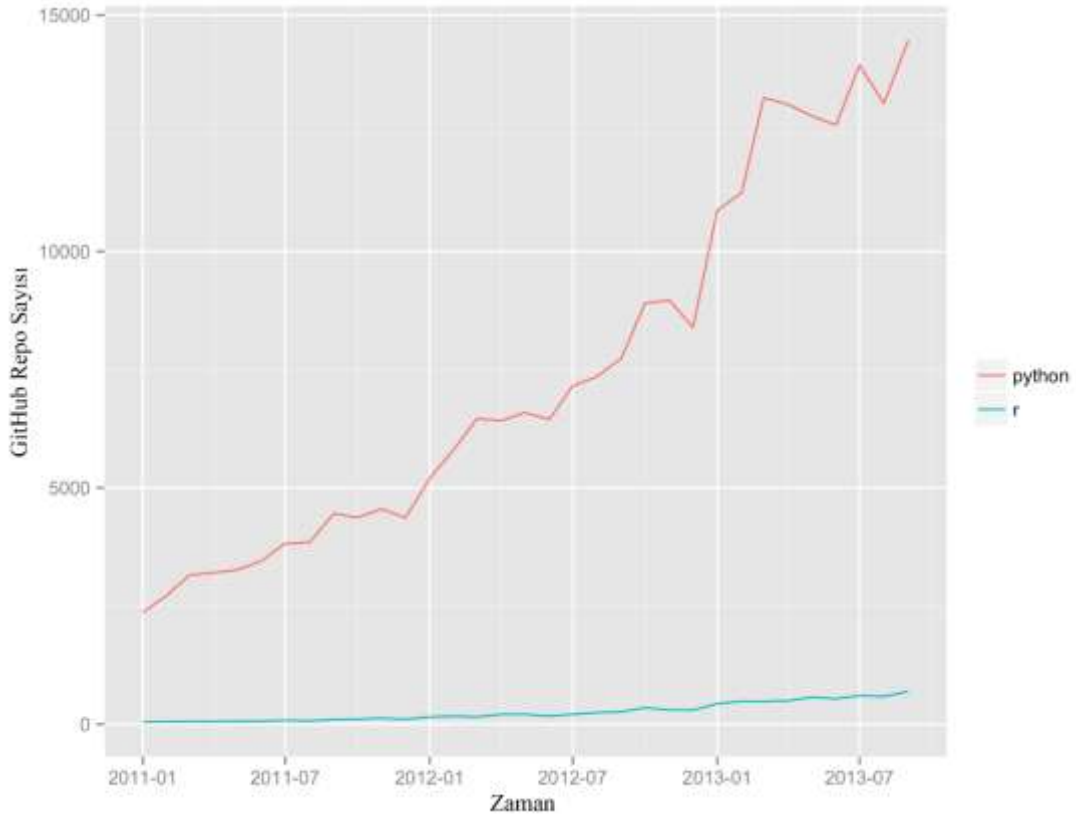
⁸ <http://blog.revolutionanalytics.com/2013/12/r-and-python.html>

⁹ Açık kaynak kod ve makale paylaşım platformu, <http://github.com>

¹⁰ <https://github.com/hadley/r-python>

¹¹ Teknoloji geliştiricilerin soru-cevap şeklinde bilgi paylaştığı bir platform, <http://stackoverflow.com>

Wickham çalışmasında ayrıca Python ve R için GitHub üzerinde geliştirilen repo'ların sayısını da karşılaştırmıştır. Bu karşılaştırma dilin öğrenimi konusundaki yaygınlığından ziyade, kullanımını temsil ettiğinden, karşılaştırma için daha sağlıklı fikir vermektedir.



Şekil 3.2 GitHub üzerinde oluşturulan Python ve R repo sayısının zamanla artışı

Python'un temel bilimlerde bu kadar yaygın bir dil olmasında pek çok etken geçerli olmuştur. Bunların başında Python'un platform bağımsızlığı gelir. Python bugün hemen her GNU/Linux dağıtımında bulunmaktadır, hatta bu dağıtımların pek çok kodu Python ile geliştirilmektedir. Bunun haricinde, Python bütün MacOS X işletim sistemlerinde de bulunmaktadır. Bunun yanı sıra, bütün Microsoft Windows işletim sistemleri için de yorumlayıcısının geliştirilmiş olması, Python'u platform bağımsız hale getirmektedir.

Ancak tahmin edileceği üzere, Python piyasadaki tek platform bağımsız programlama dili değildir. Bu yüzden tek ayırt edici özelliğinin bu olduğu düşünülemez. Python'un

öne çıkan yönlerinden birisi de, geliştirilen kodun kolay okunabilirliğidir. Bunun için Python'un yaratıcısı Guido van Rossum'un (1996) aşağıdaki sözleri dikkate değerdir :

“Okunabilirliğin bu kadar vurgulanması tesadüf değildir. Nesne yönelimli bir dil olarak Python, tekrar kullanılabilir kod yazımını teşvik etmektedir. Hepimiz son derece detaylı açıklama metinleri yazıyor olsaydık bile, okunabilir olmayan kodun yeniden kullanılabilir olması bir hayli zor olurdu. Girintili yazma biçimine ek olarak Python'un pek çok özelliği, yazılan kodun okunurluğunu artırma amacı göz önüne alınarak geliştirilmiştir.”

Günümüzde Python, gökbilim çalışmalarında da yoğun bir biçimde tercih edilmektedir. Özellikle Numpy, Scipy, pandas gibi *modüller*, matematiksel ve istatistiksel hesaplamaların rahatlıkla yapılmasını sağlarken, PyFITS ile gökbilimciler FITS görüntülerini kolaylıkla yazıp okuyabilmekte, hatta IRAF araçlarının tamamına PyRAF aracılığıyla erişip bütün görüntü işleme işlemlerini otomatikleştirebilmektedir.

2009 yılında Hindistan'da Inter University Center for Astronomy and Astrophysics'te (IUCAA) düzenlenen “High Performance Computing in Observational Astronomy: Requirements and Challenges” (Gözlemsel Astronomide Yüksek Başarımlı Programlama: Gerekenler ve Zorlukları) toplantısındaki¹² konuşmasında Lupton (2009), gökbilim çalışmalarında C++ ve Python kullanımının giderek arttığını, ve özellikle Python kullanımının artacağını öngördüğünü belirtmiştir. Her ne kadar şekil 3.1-3.2'de görülen sonuçlar sadece gökbilimi içermese de, Lupton'un öngörülerıyla uyum sağladığı söylenebilir.

Python'un gökbilim kullanımındaki artışı, çeşitli eposta listelerinin oluşmasına yol açmıştır. Zamanla bu eposta listelerinde tartışılan ve geliştirilen kütüphaneler, geliştiricilerin topluluk haline gelmesine sebep olmuş, böylelikle gökbilim için geliştirilen kütüphaneler tek çatı altından servis edilmeye başlanmıştır. Bu çalışmaların başında Astropy kütüphanesi gösterilebilir. Astropy kütüphanesini geliştiren topluluk,

¹² <http://www.iucaa.ernet.in/It-Is-Today.html>

2011 yılında bir eposta listesi ile toplandı. Geliştirilen kütüphanenin ilk versiyonu (v0.2) 2013 yılında yayınlandı (Robitaille vd. 2013). Bugün Astropy kütüphanesi bir gökbilimcinin ihtiyaç duyabileceği temel işlemlerin tamamını kapsayan Python modülü haline geldi: FITS dosyalarının okunması, Sanal Gözlemevi (*Virtual Observatory*, VO) tablolarının okunması, ASCII tablo formatlarının okunması, gökbilimde sık kullanılan fiziksel sabitlerin değerleri ve birim dönüşümleri, küresel koordinat sistem dönüşümleri, zaman dönüşümleri, Dünya Koordinat Sistemi (*World Coordinate System*, WCS) dönüşümleri mevcutken, VO istemci ve sunucu araçları, eğri fit algoritmaları, nokta dağılım fonksiyonu (*Point Spread Function*, PSF) fotometri araçları gibi sayısız özellik de şu anda geliştirilme aşamasındadır.

Python'un gökbilimciler arasındaki artan kullanımının paralelinde, WWZ algoritmasının bir Python modülünün bulunmaması, bu çalışmada Templeton'ın (2002) kodunu Python diline tercüme etmemize yol açmıştır.

3.2 Kod Yazımında Dikkat Edilen Noktalar

Gerek WWZ algoritmasının Python koduna tercüme edilmesinde, gerek Lomb-Scargle yönteminin bu araştırma sırasında rahat kullanımı için gereken programın yazımında bazı kurallara uyulmuştur. Yazımda tercih edilen hiçbir değişken, nesne, fonksiyon veya metod Türkçe isimlendirilmemiştir, eğer sözcük olarak anlam taşıyacaksa (*time*, *makefreq*, *matrix_inv* gibi) İngilizce dili tercih edilmiştir. Benzer şekilde açıklama satırlarında düşülen bütün notlar da İngilizce yazılmıştır. Böylelikle uluslararası kullanıma da açık bir kod haline getirilmiştir.

WWZ'nin tercümesinde kritik değişken isimlerinin kullanımında Templeton'ın notasyonuna sadık kalınılmıştır. Bunlar arasında *dtauhi*, *dtaulo*, *ifreq*, *itau*, *dpower*, *dpowz*, *damp*, *dneff*, *davew* vb. değişkenler gösterilebilir. Benzer şekilde Templeton'ın *altprogram* (*subroutine*) olarak çağırdığı fonksiyonlar, Python tercümesinde WWZ sınıfı objesinin *metodları* olarak geliştirilmiştir. Zorunlu olmayan bu işlem, kodun

incelenmesi sırasında daha önce Templeton'ın koduyla çalışmış kişilerin yabancılık çekmemesi düşüncesiyle uygulanmıştır.

Yapılan tercümelerde, özellikle temel altprogramların aktarılmasında, FORTRAN koduna karşılık gelen satır aralığı not edilmiştir. Bunun haricinde Templeton'ın kodu ve VStars kodu arasındaki farklılıklar, tespit edilen hatalar açıklama satırı olarak verilmiştir.

Gerek WWZ ile, gerekse LS algoritması ile programın yazımında dikkat edilen temel nokta, PEP 8'de¹³ (Python Enhancement Proposal – Python Geliştirme Önerisi) “*Style Guide for Python Code*” başlığıyla anlatılan kurallar olmuştur. Bu kurallar arasından öne çıkan ve geliştirme aşamasına uygulananlardan bazıları şöyle sıralanabilir :

- Girinti olarak 4 adet boşluk karakteri kullanılmalı
- Tab karakteri yerine 4 adet boşluk karakteri kullanılmalı
- Satır uzunluğu en fazla 79 karakter olmalı, açıklama satırları en fazla 72 karakter olmalı
- Uzun satırların parçalanmasında, elemanlar tipleriyle uyumlu hizalanmalı
- Docstring'ler tek satır değilse, kapanış tırnakları ayrıca bir satır olmalı
- Çekirdek Python dosyaları UTF-8 olarak kodlanmalı
- Modül çağırma (import) işlemlerinde, her modül için ayrı bir satır kullanılmalı
- Modül çağırma işlemleri programın başında yapılmalı (wwz.py kodunda bu kurala bir noktada verimlilik gerekçesiyle uyulmadı, açıklama satırlarında durum not edildi)
- İkili değer alan (True/False) değişkenlerin değerleri 'True' veya 'False' ile karşılaştırılmamalı. (*if a == True:* yerine *if a:* kullanılmalı)
- *None* benzeri tekil değerlerin karşılaştırılmasında *is* ya da *is not* işlemleri kullanılmalı, eşitlik işaretleri kullanılmamalıdır

¹³ <http://legacy.python.org/dev/peps/pep-0008>

Hem *wwz.py* kodu, hem de *ls.py* kodu modüler ve tek başına çalıştırılabilir biçimde geliştirilmiştir. Bu durum her iki biçimdeki kullanım için de ayrı ayrı açıklama metinleri içermeyi gerektirmektedir. Programlar komut satırından çağırıldığı takdirde, argümanların irdelenmesi *argparse* modülü ile yapıldığından, gerekli açıklama satırları bu modül aracılığıyla gerçekleştirilmiştir. Programların Python modülü olarak kullanılması durumunda yardım alınması için *docstring*'ler kullanılmıştır. *Docstring*'ler her bir Python nesnesi veya metodunun ilk açıklama satırı ile ifade edilen, doğrudan kendisini tanımlayan ve modül içinden erişilebilen bilgi dizileridir. *Docstring*'ler ve kullanımları hakkında detaylı bilgi, PEP 257'de¹⁴ “*Docstring Conventions*” başlığı altında anlatılmıştır.

3.3 WWZ Programı

wwz.py programı, iki türlü kullanılabilir şekilde tasarlandı: Program olarak ve Modül olarak. Program olarak kullanımı, doğrudan komut satırından parametre sağlanarak, alışlagelmiş bir uygulama gibi gerçekleşmektedir. Templeton'ın yaptığı gibi etkileşimli bir yazılım geliştirilmemeye özen gösterildi. Bu sayede, komut satırından çağırılabilirdiği için herhangi bir program tarafından otomatik olarak parametre gönderilerek rahatlıkla kullanılabilir.

Bunun yanısıra yazılacak herhangi bir Python programı tarafından modül olarak da çağırılabilir şekilde tasarlanmıştır.

Hangi şekilde kullanılırsa kullanılsın, programın çalışması için bazı önşartlar bulunmaktadır.

¹⁴ <http://legacy.python.org/dev/peps/pep-0257/>

3.3.1 Kullanım için gerekenler

Ön şartların başında, kullanılacak sistemde Python 2.7 bulunması gerekmektedir. Yazılım, çalıştırıldığında sistemde Python 2.7'nin yüklü olup olmadığını kontrol etmektedir. Eğer bu versiyonu bulamazsa bir uyarı mesajı vermektedir. Sistemin özellikle Python 2.7'ye uygun biçimde yazılmasının sebebi, *argparse*¹⁵ modülünü kullanmasıdır. Argparse modülü, programın tek başına kullanımında komut satırından girdi olarak gönderilen parametrelerin akıllıca kullanılmasını, gerektiğinde bir ASCII dosyasından girdi parametrelerinin okunmasını sağlamaktadır. Argparse modülü Python 2.7'de geliştirilmiştir.

CentOS¹⁶ işletim sistemlerinde Python 2.6 kullanılmaktadır ve 2.7'ye güncellenmesinde problemler yaşanmaktadır. Böyle bir durumla karşılaşıldığında sağlıklı bir yükleme yapılması için gerekli adımlar EK 4'te açıklanmıştır.

Geliştirilen kod, ayrıca *NumPy*¹⁷ modülüne ihtiyaç duymaktadır. Özellikle matris dönüşümlerinde hem verimli oluşu, hem de kolay kullanımı nedeniyle, NumPy'den oldukça yararlanılmıştır.

Programlar, Ubuntu¹⁸ 14.04, CentOS 6.5 ve Microsoft Windows 8 üzerinde test edilmiştir.

3.3.2 Programın kullanımı

wwz.py'nin program olarak kullanılabilmesi için, GNU/Linux ve Mac OS X işletim sistemlerinde, dosyanın kendisinin çalıştırılabilir hale getirilmesi gerekmekte veya Python yorumlayıcısıyla çağırılması gerekmektedir. Ayrıca Microsoft Windows işletim

¹⁵ <https://docs.python.org/2.7/library/argparse.html>

¹⁶ <https://www.centos.org/>

¹⁷ <http://www.numpy.org/>

¹⁸ <http://www.ubuntu.com/>

sistemlerinde kullanımı için Python yorumlayıcısının yüklenmesi ve dosya uzantısının ilişkilendirildiğinden emin olunması gerekmektedir.

Program çalıştırıldığı zaman, herhangi bir parametre sunulmadığı için bir uyarı mesajı verecek ve kullanımı hakkında bilgileri standart çıktıya şu şekilde gönderecektir:

```
$ ./wwz.py
usage: wwz.py [-h] -f FILE -o OUTPUT -l FREQ_LOW -hi FREQ_HIGH -d FREQ_STEP -c
              DCON [-g] [-m] [-t TIME_DIVISIONS] [--time] [--stats]
              [--no-headers] [-p PARALLEL]
wwz.py: error: argument -f/--file is required
```

Bu kullanımıyla program, herhangi bir parametre belirtilmediği için hata mesajı vermektedir. Programın parametrelerinin kullanımı ve genel yardımı hakkında detaylı bilgi almak için, “-h” parametresiyle çalıştırılmalıdır. Aşağıda bir örneği gösterilmiştir:

```
$ ./wwz.py -h
usage: wwz.py [-h] -f FILE -o OUTPUT -l FREQ_LOW -hi FREQ_HIGH -d FREQ_STEP -c
              DCON [-g] [-m] [-t TIME_DIVISIONS] [--time] [--stats]
              [--no-headers] [-p PARALLEL]
```

A Weighted Wavelet Z-Transformation Application for Python.

Translated by M. Emre Aydin - emre.m.aydin@gmail.com

<http://about.me/emre.aydin>

Available at <http://github.com/eaydin>

Input arguments can be read from a file. The file descriptor

prefix is '@'.

In order to read argument from a file named args.txt,

the argument @args.txt should be passed.

An example for args.txt :

```
-f=myinputfile.txt
```

```
-o=theoutputfile.output
-m
--freq-step=0.001
-l=0.001
-hi=0.01
-c=0.001
-p=0
```

You can pass arguments from file and commandline at the same time.

If two same arguments passed by this method, the latter will be used. So if you want to override some arguments in an argument file, specify the file first.

An example usage for our earlier @args.txt is as :

```
python wwz.py @args.txt -c=0.0125
```

The above command will use the settings in args.txt but will use c=0.0125 instead of c=0.001

Comments and blank lines are NOT allowed in argument files.

Import this script via Python to use it as a module, rather than a standalone script. (import wwz)

optional arguments:

```
-h, --help          show this help message and exit
-f FILE, --file FILE the Input File, Raw Lightcurve
-o OUTPUT, --output OUTPUT
                    the Output File Name
-l FREQ_LOW, --freq-low FREQ_LOW
                    the Low Frequency Value
-hi FREQ_HIGH, --freq-high FREQ_HIGH
                    the High Frequency Value
-d FREQ_STEP, --freq-step FREQ_STEP
                    the dF value, incremental step for Frequency
-c DCON, --dcon DCON the C constant for the Window Function
-g, --gnuplot-compatible
```

the Output file is GNUPlot compatible, which means the tau's will be grouped so that pm3d can easily map. Default value is 'False'.

-m, --max-periods Creates a secondary output with the maximum Periods for each single tau. This can be drawn in 2D. The output filename is derived from the -o option, added 'max_periods'. Default value is 'False'.

-t TIME_DIVISIONS, --time-divisions TIME_DIVISIONS The Time Divisions value. Templeton assumes this as 50. VStars from AAVSO leaves this optional contrary to Templeton, yet it's default value is also 50.

--time Calculate the time of operation in seconds and print to standard output.

--stats Prints fundamental statistics to standard output.

--no-headers Doesn't print headers to output files if set. Default is 'False'.

-p PARALLEL, --parallel PARALLEL Created threads to speed up the process. Default value is '1', which means single thread. '0' means number of detected CPUs, can be overridden.

Gerek kodlarındaki açıklama satırları, gerekse yardım bilgileri olmak üzere, program tamamen İngilizce olarak geliştirilmiştir. Bu nedenle yardım bilgilerinde açıklanan parametre kullanımını burada açıklamakta fayda vardır. Bazı parametreler hem kısa hem de uzun gösterimle ifade edilebilmektedir. Parametreler, kısa ve uzun gösterimleriyle birlikte çizelge 3.1'de açıklanmıştır.

Çizelge 3.1 wwz.py programının girdi argümanlarının açıklamalı tablosu

Argümanın Kısa Gösterimi	Argümanın Uzun Gösterim	Argüman Açıklaması
-h	--help	Yardım. Programın kullanımı hakkında detaylı açıklamaları standart çıktıya yazdırır.

Çizelge 3.1 wwz.py programının girdi argümanlarının açıklamalı tablosu (devam)

-f	--file	Girdi dosyası. Analizi yapılacak ışık eğrisinin dosyasını ifade eder. Dosya ASCII biçiminde olmalıdır. Satır başında “#” veya “%” karakterleri bulunuyorsa, bu karakterler dikkate alınmaz.
-o	--output	Çıktı dosyası. WWZ hesaplamalarının yazdırılacağı çıktı dosyasının adı.
-l	--freq-low	Düşük frekans değeri.
-hi	--freq-high	Yüksek frekans değeri.
-d	--freq-step	Frekans adımı. Düşük ve yüksek frekans aralıklarında gezinirken atılacak her bir adımın büyüklüğüdür.
-c	--dcon	Dalgacık denklemindeki pencere büyüklüğünü ifade eden <i>c</i> sabiti. Foster (1996), gökbilim araştırmalarında bu değerin 0.0125 civarında tutulmasının uygun olduğunu belirtiyor.
-g	--gnuplot-compatible	Çıktı dosyasını gnuplot'un “pm3d map” çizimine uygun halde yazdırır. Bunun için her <i>tau</i> değeri kendi içinde gruplandırılmış olmalıdır. Argümanın öntanımlı değeri <i>'False'</i> tur.
-m	--max-periods	“-o” parametresiyle tanımlanan dosya adının sonuna “.max_periods” eklenerek ikinci bir dosya oluşturur. Bu çıktıda, her tau'ya karşılık gelen en yüksek WWZ'li periyot değerleri bulunmaktadır. Çıktının 2 boyutlu temsili grafiği için uygundur. Argümanın öntanımlı değeri <i>'False'</i> tur.
-t	--time-divisions	Zaman aralıkları değeri. Bu değer Templeton'ın kodunda 50 olarak sabitlenmiştir, wwz.py'de de aynı durum uygun görülmüştür. Değiştirilmesi

Çizelge 3.1 wwz.py programının girdi argümanlarının açıklamalı tablosu (devam)

		istendiği takdirde bu parametreye değer atanarak farklı zaman aralıklarıyla hesaplama yapılabilir.
	--time	Programın bütün hesaplamalarının ne kadar sürdüğünü saniye cinsinden standart çıktıya yazar.
-p	--no-headers	Çıktı dosyalarına başlık bilgileri yazmaz.
	--parallel	Paralel hesaplama parametresi. 0 verildiğinde sistemdeki işlemci sayısı tespit edilir. Argümanın öntanımlı değeri 1'dir.

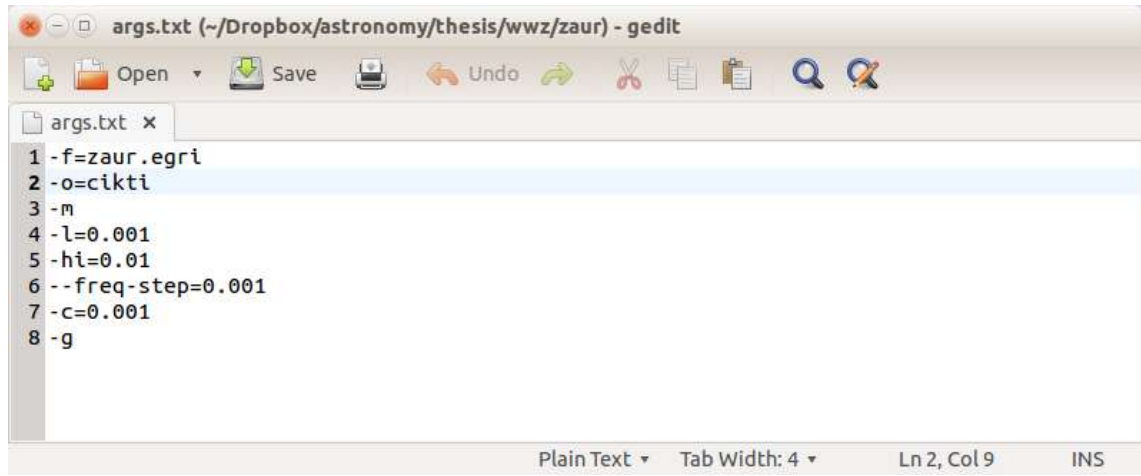
Programa argümanlar sunulurken, argümanların değer alıp almadıkları dikkate alınmalıdır. Değer almayan argümanlar aslında 'True/False' değerleri almaktadırlar. Değer almayan argümanlar şunlardır : “--gnuplot-compatible”, “--max-periods”, “--time”, “--no-headers”. Değer alan argümanlar, kendilerinden sonra bir boşluk bırakılarak değerleri belirtilmeli, veya kendilerinden sonra “=” işareti konularak değerleri belirtilmelidir.

Örneğin -f argümanına t_umi.dat dosyasını girdi olarak kullanmak için aşağıdaki yöntemlerin tamamı geçerlidir:

```
$ ./wwz.py -f t_umi.dat
$ ./wwz.py -f=t_umi.dat
$ ./wwz.py -file t_umi.dat
$ ./wwz.py -file=t_umi.dat
$ ./wwz.py -f "t_umi.dat"
$ ./wwz.py -f="t_umi.dat"
$ ./wwz.py -file "t_umi.dat"
$ ./wwz.py -file="t_umi.dat"
```

Girdi argümanlarını, wwz.py programına ASCII dosyası içinde sunmak da mümkündür. Bunun için wwz.py, çoğu programca algılanan “@dosyaismi” gösterimini

kullanılmaktadır. Bu gösterimde *dosyaismi* isimli dosyanın içerisindeki değerler, argüman olarak programa atanacaktır. Argümanların belirtileceği dosyaya, argümanlar (ve varsa değerleri) satır satır yazılmalıdır, iki argüman aynı satırı paylaşmamalıdır. Argümanlar ve değerleri arasında bir boşluk veya eşitlik işareti konmalıdır. Argümanların belirtildiği dosyalarda boş satır bulunmamalıdır. Örnek bir argüman dosyasının içeriği, ekran görüntüsü halinde **şekil 3.3**'te gösterilmiştir.



```
1 -f=zaur.egri
2 -o=cikti
3 -m
4 -l=0.001
5 -hi=0.01
6 --freq-step=0.001
7 -c=0.001
8 -g
```

Şekil 3.3 wwz.py için kullanılabilecek örnek argüman dosyasının içeriği

Programın çalıştırılması aşamasında argümanlar hem komut satırından, hem de dosyadan verilebilir. Aynı argümanın her iki yöntemle de verilmesi durumunda program, sıralama olarak sonra verileni dikkate alacaktır. Bu özellik sayesinde program benzer argümanlarla defalarca çalıştırılacaksa, sadece değiştirilecek argümanın, argüman dosya ismi belirtilmesinden sonra ifade edilmesi yeterli olacaktır. Örneğin **şekil 3.3**'te gösterilen args.txt dosyasını argüman olarak almak, ancak *-c* değerini *0.005* olarak seçme istiyorsak şu şekilde programı çalıştırabiliriz :

```
$ ./wwz.py @args.txt -c 0.005
```

3.3.3 Modül olarak kullanım

Program sadece komut satırından kullanılacak biçimde değil, bir Python modülü olarak kullanılabilecek biçimde de geliştirilmiştir. Bunun avantajı, herhangi bir araştırmacının kendi geliştirdiği Python programından dilediği gibi kodu çağırıp, gerekli nesneleri kullanabilmesidir.

Geliştirilen Python programlarından *wwz.py*'nin çağırılabilmesi için, *wwz.py*'nin program ile aynı dizinde olması, veya Python'un kütüphanelerinin bulunduğu dizinde yer alması gerekmektedir.

Python'da kütüphanelerin çağırılması işlemine *import* denilmektedir. *wwz* import edildikten sonra, içeriğine etkileşimli Python konsolunda *dir* komutuyla erişilebilir. Modülün kendisi '*wwz.py*' olduğu için, *.py* uzantısı belirtilmeden çağırılır. Modülün açıklamaları *__doc__* özelliğinde bulunmaktadır.

```
$ python
```

```
Python 2.7.6 (default, Mar 22 2014, 22:59:56)
```

```
[GCC 4.8.2] on linux2
```

```
Type "help", "copyright", "credits" or "license" for more information.
```

```
>>> import wwz
```

```
>>> dir(wwz)
```

```
['WWZ', 'WWZPAR', '__builtins__', '__doc__', '__file__', '__name__', '__package__', 'argparse',  
'datetime', 'math', 'numpy', 'os', 'sys']
```

```
>>> print wwz.__doc__
```

```
A Weighted Wavelet Z-Transformation Application for Python.
```

```
Translated by M. Emre Aydin - emre.m.aydin@gmail.com
```

```
http://about.me/emre.aydin
```

```
Available at http://github.com/eaydin
```

```
For details, read the docstrings of WWZ class.
```

Modülün içeriği görüntülendiğinde, çeşitli nesneler görülmektedir. Bunlardan *math*, *argparse*, *numpy*, *sys*, *os* modülleri programın çalışabilmesi için çağırılmıştır. “__” ile başlayıp biten nesneler Python tarafından otomatik olarak oluşturulmaktadır. *__doc__* bunlardan birisidir ve içeriği modülün başına yazılan açıklama satırlarından oluşmaktadır. Python'da bunlara *docstring* denilmektedir ve detayları *PEP 257*¹⁹ açıklanmıştır. Programın modül olarak kullanılmasında diğer açıklamalara erişmek için de ilgili özelliklerin *docstring*leri kullanılmalıdır.

__builtins__ özelliği aslında *Cython* ile geliştirilen modül içeriğinden kaynaklı bir yeniden adlandırmadır. Python'da herhangi bir ortam veya kütüphaneyle çalışırken kullanılabilecek *__builtin__* ile aynı özelliği taşımaktadır. *__builtin__* kullanımıyla pek sık karşılaşılsa da, örneğin aynı adlandırmaların yer aldığı modüller arasında karışıklık oluşmaması için kullanılmaktadır.

__file__ özelliği, import edilen modülün hangi fiziksel dosyaya işaret ettiğini tespit etmek için kullanılmaktadır. Bizim durumumuzda bu “*wwz.pyc*” olacaktır. Dosya adının “*wwz.py*” olmadığına dikkat edilmelidir. Bunun sebebi, herhangi bir Python kodunu çalıştırdığınızda veya *import* ettiğinizde, Python yorumlayıcısının ilgili kodu derleyip, derlenmiş kodu kullanmasıdır. Python yorumlayıcısı tekrar aynı kod çağırıldığında baştan derlemez, hali hazırda derlenmiş dosya varsa oluşturulma tarihi ile tekrar derlenmesi istenen kodun tarihini karşılaştırır, ancak değişiklik olduğu tespit edilirse yeniden derler.

__package__ değeri ancak modülümüz bir paket halinde sunuluyor olsaydı anlam kazanırdı. Geliştirilen *wwz.py* modülü bir paket olmadığından, kullanıcının ihtiyacı olmamaktadır.

__name__ değeri, modülün ismini döndürmektedir. Eğer program komut satırından bir program olarak çağırılırsa bu durumda *__name__* değeri *__main__* olur. Modüllerin tek

¹⁹ <http://legacy.python.org/dev/peps/pep-0257>

başına program olarak kullanılabilmesi için bu kontrolün yapılması gerekmektedir ve wwz.py için de bu teknik kullanılmıştır.

Son olarak, modülün içeriğinde WWZ ve WWZPAR görülmektedir. Bunlardan WWZ, modül ismi olan wwz ile karıştırılmamalıdır. WWZ, programın yazımı sırasında içeriğinde oluşturulan bir sınıftır (class). Programın çalışması için önce WWZ sınıfından bir nesne oluşturulur, sonra bu nesnenin özellikleriyle çeşitli işlemler yapılır. Bunun bir örneği aslında kodun kendi başına kullanımında görülmektedir. WWZ sınıfından türetilen nesnenin olası özellikleri şöyledir:

```
>>> dir(wwz.WWZ)
['__class__', '__delattr__', '__dict__', '__doc__', '__format__', '__getattribute__', '__hash__', '__init__',
 '__module__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__setattr__', '__sizeof__',
 '__str__', '__subclasshook__', '__weakref__', 'makefreq', 'maketau', 'matrix_inv', 'readfile', 'roundtau',
 'writefile', 'writgnu', 'wwt']
```

Readfile, *maketau*, *makefreq*, *wwt* fonksiyonlarının oluşturulması, programlama aşamasında olabildiğince Templeton'ın (2002) FORTRAN koduna sadık kalınmasından kaynaklanmaktadır. Bütün fonksiyonlar docstring içermektedirler, bu sayede interaktif Python konsolundan özellikleri okunabilmektedir. Benzer şekilde WWZ sınıfının docstring'i de kullanımı hakkında detaylı bilgi içermektedir.

WWZ sınıfından devralınarak yaratılan nesneye taşınan argümanlar, oluşturulacak çıktı dosyası, hesaplama yapılacak frekans aralığı gibi değerleri taşımaktadır. Ardından oluşan nesnenin *readfile* metodu çalıştırılmalıdır. Bu özellik sayesinde, başta tanımlanan girdi dosyasından ışık eğrisinin değerleri okunur. Değerler okunduktan sonra *maketau* metodu ile hesaplanacak zaman aralıkları oluşturulur. Sonra *makefreq* metodu ile her *tau* değerine karşılık hesaplanacak frekans değerleri hesaplanmaktadır. Bütün bu değerler hazır hale geldikten sonra, artık *ağırlıklandırılmış dalgacık z-dönüşümü* (*weighted wavelet Z-transform – WWZ*) işleminin gerçekleştirilmesi için *wwt* metodu çalıştırılır.

WWZ nesnesinin *writefile* ve *writgnu* metodlar, hesaplanan değerlerin çıktı dosyalarına yazdırılmalarını sağlar. *writgnu* metodu *Gnuplot* uyumlu çıktıların oluşmasını sağlarken, *writefile* metodu bunu dikkate almamaktadır.

Bütün metodların docstringleri yazılmıştır, böylelikle aldıkları argümanlar ve gerektiğinde bu argümanların tipleri tanımlanmıştır. Örneğin *wwt* metodunun docstringi aşağıdaki gibidir.

```
>>> print wwz.WWZ.wwt.__doc__
```

The WWZ Algorithm

Arguments are :

time = The time values as an array
magnitude = The magnitude values as an array
flo = The Low Frequency
fhi = The High Frequency
df = The Frequency Step
dcon = The C constant of WWZ Window
timedivisions = The TAU steps

Returns a NumPy Array

WWZPAR sınıfı, algoritmanın paralel olarak çalıştırılabilmesi için geliştirilmiştir. Algoritma, **bölüm 3.3.6**'da açıklanan sebeplerden dolayı, modüler yapıda tutulduğunda paralelleştirilemez. Bu sebeple paralel çalıştırılması gerektiğinde *WWZ* sınıfına çok benzer, ancak bölünmüş parçalar arasındaki veri alışverişini geçici dosyalar üzerinden yapacak şekilde yazılmıştır. *WWZ* sınıfından farklı olarak, oluşturulma (initiation) aşamasında argüman alır. *WWZ* sınıfı altındaki *wwt* metodunun ilk iki parametresinde ıřık eğrisinin verilerini diziler olarak programa sunarken, *WWZPAR* sınıfında doğrudan oluşturulma aşamasında girdi dosyası olarak sunulmaktadır. *WWZPAR* böylece gerektiği biçimlerde *readfile* metodunu çalıştırıp, belirtilen paralellik sayısına göre girdiyi parçalara bölmekte, hesaplama sonuçlarını geçici dosyalara yazdıktan sonra bu parçalar program tarafından birleştirilmektedir. *WWZPAR* modül olarak kullanım

açısından kesinlikle pratik bir yaklaşım sağlamaz, dolayısıyla programın tek başına, komut satırından çalıştırılmasında kullanılmak üzere tasarlandı.

3.3.4 Üretilen çıktının düzeni

wwz.py programının çıktı dosyaları da tıpkı pek çok özelliği gibi orijinal FORTRAN koduna sadık kalınarak oluşturulmuştur. Temelde iki çıktı dosyası üretilmektedir. Ancak bunlardan bir tanesinin (*max periods* dosyası, bkz. **çizelge 3.1**) oluşturulması kullanıcının tercihinin bırakılmıştır.

Program komut satırından çalıştırılırken *-o* parametresiyle, veya modül olarak çağırıldığında, *writefile* veya *writgnu* metodlarına argüman olarak verilen *filepointer* değeri sayesinde, çıktı dosyası belirtilebilmektedir.

Program komut satırından çalıştırılırken *-no-headers* argümanı kullanılmazsa, veya modül olarak kullanıldığında *writefile* veya *writgnu* metodlarının üçüncü argümanı 'False' olarak girilirse, çıktı dosyalarının başında başlık bilgileri bulunur. Bu bilgiler *gnuplot* gibi programların açıklama satırı olarak algılaması için # karakteriyle başlar. Bu bilgilerde her sütunun hangi değeri yansıttığını ifade eden kısaltmalar bulunmaktadır; *tau*, *freq*, *wwz*, *amp*, *coef*, *neff* şeklinde sıralanmıştır. Değerler arasında birer *tab* boşluk bırakılmıştır. Değerlerin karşılık geldiği ifadeler çizelge 3.2’de açıklanmıştır.

Çizelge 3.2 Çıktıda kullanılan değerlerin açıklamaları

Değer	Açıklama
TAU	Jülyen günü cinsinden incelenen zaman değeri.
FREQ	İlgili TAU için incelenen frekans değeri.
WWZ	Hesaplanan WWZ değeri. Bkz. (50) denklemi.

Çizelge 3.2 Çıktıda kullanılan değerlerin açıklamaları (devam)

COEF	Foster'ın (1996) y_a notasyonu ile ifade ettiği “en-iyi-fit sabiti” değeri. Bkz (35) denklemi.
NEFF	Foster'ın (1996) N_{eff} notasyonu ile ifade ettiği, ağırlıklandırılmış hesaplamalar yapılabilmesi için gerçekleştirilen projeksiyonda kullanılan değer sayısı. Bkz. (42) denklemi.

Programın komut satırından kullanılması sırasında argüman olarak *-gnuplot-compatible* değeri sunulursa, veya modül olarak kullanıldığında *writgnu* metoduyla çıktı üretilirse, oluşturulan çıktı dosyası Gnuplot programının *pm3d map* modunda okuyabileceği haliyle yazılır. Bu durumda, her *TAU* değeri kendi içinde gruplandırılmalı, yeni *TAU* değeri yazılmadan önce bir satır boşluk bırakılmalıdır. Sözkonusu yazım, maksimum periyot değerlerinin hesaplandığı alternatif çıktı için geçerli değildir çünkü bu dosyada her *TAU* değeri için tek *WWZ* değeri bulunmaktadır. Gnuplot uyumlu modda oluşturulmuş bir çıktının ekran görüntüsü şekil 3.4’te görülebilir.

```

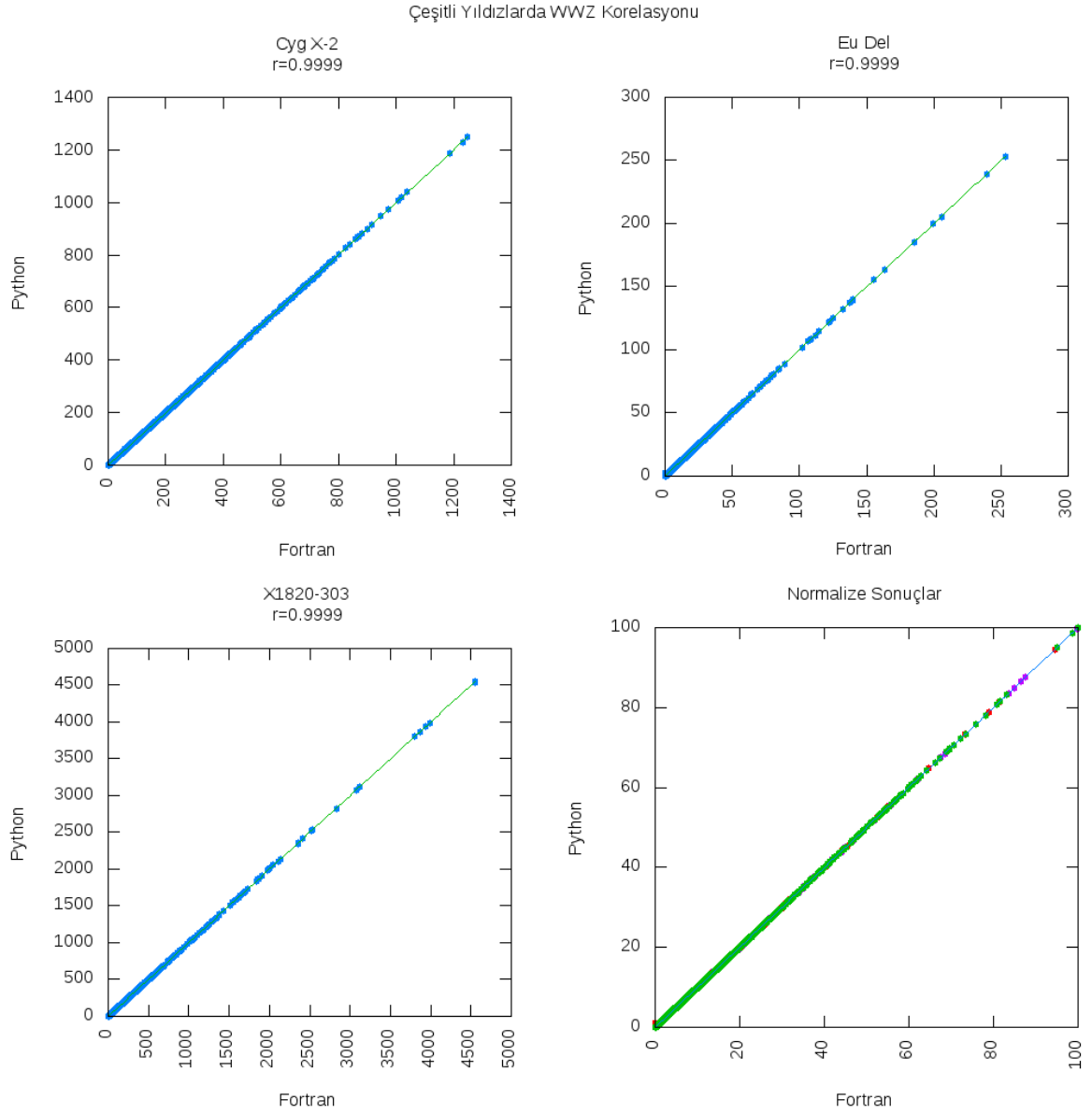
1 # Lightcurve File : herx1.python.wvz
2 # Numdat: 95782      Dave: 0.947794783989      Dsig : 2.64818464856      Dvar: 7.01280871577
3 # From JD 50000.0 to JD 55900.0 Step 100.0
4 # From fre 0.02 to fre 1.0 Step 0.005
5 #      TAU      FREQ      WWZ      AMP      COEF      NEFF
6 50000.0000 0.0200000 2.8292 0.9017 0.6396 151.0148
7 50000.0000 0.0250000 1.5187 1.5111 0.8663 61.1362
8 50000.0000 0.0300000 0.1317 0.7318 0.0955 44.1868
9 50000.0000 0.0350000 0.0455 2.4391 1.5756 39.9155
10 50000.0000 0.0400000 0.0039 3.8451 2.6307 37.4438
11 50000.0000 0.0450000 0.0000 3.1141 1.8916 34.5280
12 50000.0000 0.0500000 0.0000 2.5414 1.3242 30.8411
13 50000.0000 0.0550000 0.0000 2.0837 0.8838 26.0021
14 50000.0000 0.0600000 0.0000 1.6971 0.5300 22.2488
15 50000.0000 0.0650000 0.0000 1.3512 0.2366 18.2301
16 50000.0000 0.0700000 0.0000 3.1032 -2.6605 9.5587
17 50000.0000 0.0750000 0.0000 0.0000 0.0000 0.0000
18 50000.0000 0.0800000 0.0000 0.0000 0.0000 0.0000
19 50000.0000 0.0850000 0.0000 0.0000 0.0000 0.0000
20 50000.0000 0.0900000 0.0000 0.0000 0.0000 0.0000
21 50000.0000 0.0950000 0.0000 0.0000 0.0000 0.0000
22 50000.0000 0.1000000 0.0000 0.0000 0.0000 0.0000
23 50000.0000 0.1050000 0.0000 0.0000 0.0000 0.0000

```

Şekil 3.4 Gnuplot uyumlu modda oluşturulmuş bir çıktı örneği

3.3.5 Karşılaştırmalı sonuçlar

Python ile geliştirilen WWZ programının hassasiyetinin karşılaştırılabilmesi için, ışık eğrileri aynı girdi parametreleriyle Templeton'ın (2002) FORTRAN kodu ile de çalıştırılmıştır. Oluşturulan çıktılarda WWZ sonuçları birbirlerine karşı grafiğe geçirilerek, aralarındaki korelasyon irdelenmiştir. Korelasyon hesaplamasında Pearson Korelasyon ilişkisi kullanıldı (Pearson 1895). Python'un NumPy kütüphanesi bu algoritmayı içerdiğinden, bir Python programı yazılarak her yıldız için Templeton'ın FORTRAN kodu çıktılarıyla, bu çalışmada geliştirilen programın çıktıları karşılaştırılarak korelasyon hesaplaması yapılmıştır. Yıldızların tamamı için korelasyon 0.9999 düzeyinde hesaplanmıştır. Bu değer, sonuçlarımızın Templeton (2002) ile tutarlı olduğunun bir göstergesidir. 3 test yıldızının karşılaştırması ve çıktıların normalize edilerek üst üste bindirilmiş hali şekil 3.5'te gösterilmiştir. Grafiklerdeki düz çizgi, $f(x)=x$ doğrusunu temsil etmektedir. Beklendiği üzere Python ve FORTRAN sonuçları bu çizgi üzerinde dağılım göstermiş, korelasyonları da uyumlu çıkmıştır. “Normalize Sonuçlar” (sağ alt) paneli ise, 3 test yıldızının sonuçlarının 100'e normalize edilerek üst üste çizdirilmesi ile elde edilmiştir.

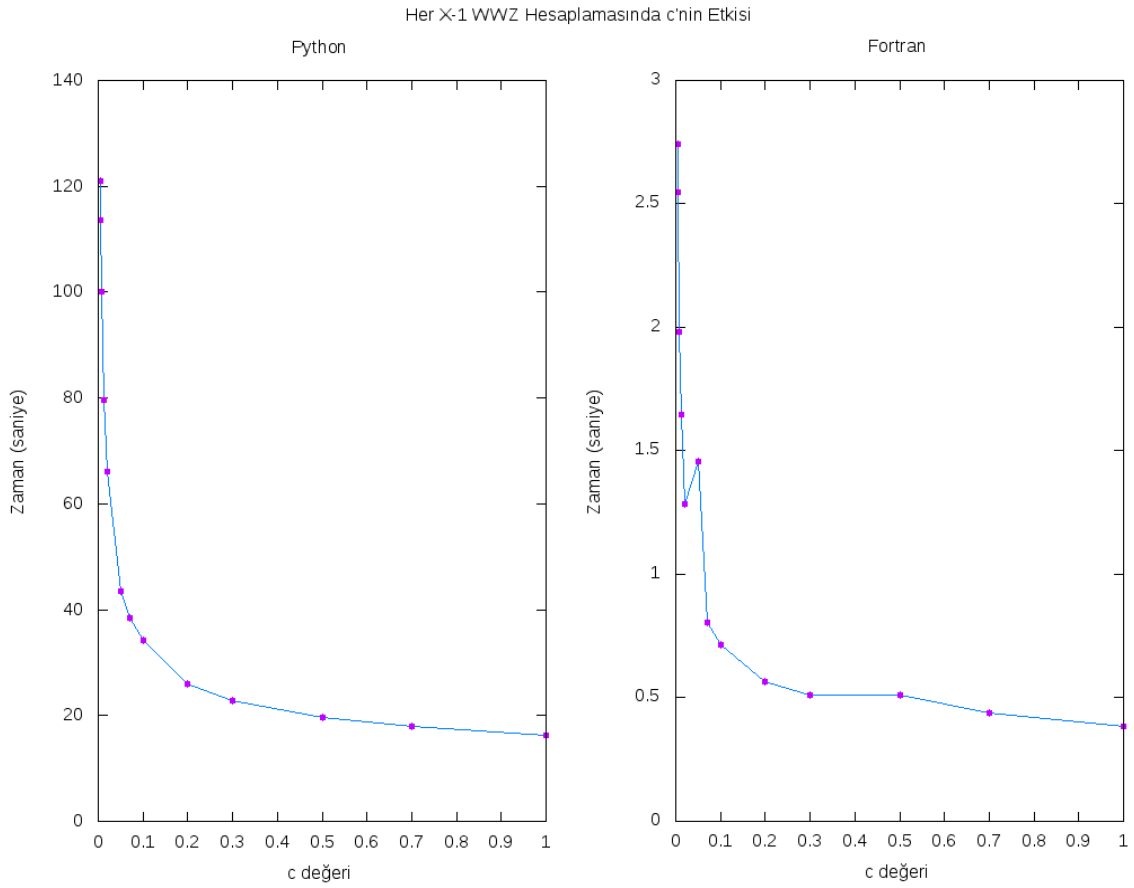


Şekil 3.5 Bazı test yıldızları için WWZ korelasyon sonuçları

3.3.6 Verimlilik testleri

Geliştirilen Python kodu üzerinde, farklı girdi dosyaları ve girdi parametreleriyle hız testleri yapılmıştır. Python, FORTRAN gibi makine diline derlenmiş düşük seviyeli bir dil olmadığından, FORTRAN ile kıyaslanabilecek kadar hızlı olması beklenmemekle birlikte, sonuçlar öngörülenden daha “yavaş” olduğunu göstermiştir.

FORTRAN ve Python kodlarının kıyaslanmasına, c parametresinin değiştirilmesiyle başlandı. Diğer parametrelerin sabit tutularak, Her X-1 yıldızı üzerinde yapılan hız testi **şekil 3.6**'da gösterilmiştir. Sonuç olarak Python kodunun çok daha yavaş olduğu, ancak c parametresi (sönümlenme katsayısı) küçüldükçe üstel olarak sürenin uzadığı gözlenmiştir. Süredeki değişim, her iki programlama dilinde de benzerlik göstermektedir.



Şekil 3.6 Her X-1 verileri üzerinde c parametresi serbest bırakılarak yapılan verimlilik testi sonuçları

Python programının hızlandırılması için *Cython*²⁰, *PyPy*²¹ gibi teknolojiler de değerlendirilmiştir.

²⁰ <http://cython.org>

Cython kullanılarak yapılan programlamada, özellikle yüksek hesaplama gücü gerektiren işlemlerde, kullanılan değişkenlerin tiplerinin tanımlanmasının performans artışına sebep olabileceği düşünüldü. Değişkenlerin yanısıra WWZ nesnesinin alt metodları C sınıfında tanımlandı ve bütün alt metodların hangi formatta veri döndüreceği ifade edildi. Ayrıca yoğun olarak NumPy kütüphanesi kullanıldığından, bu kütüphanenin *cimport* metoduyla çağırılmasında bir performans artışı beklendi. Programın sadece temel işlemlerini yapan kısımları Cython ile yeniden yazıldı, dosya okuma seçenekleri, parametre olarak argüman alma becerileri bu modülde geliştirilmedi. Yaklaşık 340 satırlık çekirdek kod, Cython ile derlendiğinde 8340 satırlık bir kod haline dönüşmüş, ancak sonuçlar beklentiyi karşılamamıştır. Programın performansında gözle görülür hiçbir artış gözlenmemiştir. Bu durumda programda Cython teknolojisi kullanılmasının bir faydası olmayacağına karar verilmiştir.

Cython'un yetersizliği karşısında, PyPy teknolojisi test edildi. Aslında PyPy temelde Cython ile aynı mantıkla çalışmaktadır, sadece yazılan kodu doğrudan C koduna çevirmek yerine, C kodu ile çalışan bir Python yorumlayıcısı sunmaktadır. Bu nedenle, yeni bir yorumlayıcı yüklemek ve bu yorumlayıcının kullanacağı ek kütüphaneleri de kendisine uygun olarak yüklemek gerekmektedir. Bu gereklilikler yüzünden program, kolay kullanılabilir ve kolay yüklenebilir bir durumda olmayacağından, yapılan çalışma sadece deneme amaçlı gerçekleştirilmiştir.

PyPy yorumlayıcısı, resmi internet sitesinde²² anlatıldığı gibi yüklenmiştir. Program NumPy kütüphanesini kullandığından, NumPy'nin PyPy için uygun versiyonu da yüklenmiştir. Bunun için güncel GitHub *repolarındaki*²³ versiyon kullanılmıştır. Kod, tamamen aynı sonuçları üretmekle birlikte performans artışı gözlenmemiştir. Cython temelli testte alınan sonuçlara benzer sonuçlar elde edilmiştir. Sistemde yüklü mevcut

²¹ <http://pypy.org>

²² <http://pypy.readthedocs.org/en/latest/getting-started-python.html>

²³ <https://bitbucket.org/pypy/numpy.git>

Python yorumlayıcısının bu kadar modifiye edilip, yeni kütüphane yükleme külfetinin kullanıcı yararına olmayacağı düşüncesiyle bu testlere devam edilmemiştir.

Bir diğer verimlilik artırma testi, programın paralel çalışacak şekilde tasarlanmasıyla gerçekleştirildi. Templeton'ın (2002) programı, FORTRAN kodunun doğası gereği paralelleşecek durumda değildir. Çünkü bütün işlem statik matrislerle oluşturulmaktadır. Tercüme edilen Python kodu da FORTRAN koduna sadık kalmak amacıyla başta bu şekilde tasarlanmış ancak, daha sonra paralelleştirme girişimleri için, matrisler dinamik değerlere dönüştürülmüştür. Kısacası matris hesaplamalarında kullanılan değişkenler nesnenin genel değişkeni olarak değil, döngü sırasında tanımlanmış değişkenler olarak tanımlanmıştır. Böylelikle veri birden fazla parçaya bölünüp ayrı ayrı işleme tabi tutulduğunda, aynı anda hesaplanan matrisler birbirinin üzerine yazılmayacak şekilde tasarlanmıştır.

Paralelleştirme testi, öncelikle program girdilerinin parçalara bölünüp çalıştırıldığında aynı sonuçları üretip üretmeyeceği yönünde yapılmıştır. Bunun için işlemci düzeyinde bir paralelleştirme gerekmediğinden, program *çok-parçacıklı (threaded)* biçimde yazılmış, yani tek işlemci üzerinde istenilen sayıda alt işlem oluşturup, programın bu işlemleri birbirlerinin sonuçlarını beklemeden gerçekleştirmesi sağlanmıştır. Geliştirilen test programı birkaç farklı sayıda *çok-parçacığa* bölünerek test edilmiştir. Bu işlem boyunca program, örneğin 4 parçaya bölünecekse, girdi dosyasını 4 parçaya bölüp, her birini ayrı WWT metodları ile hesaplanmıştır. Sonuçların, çok-parçacıklı çalışmayan modelle aynı oldukları gözlenmiştir.

Eğer Python yorumlanan bir dil olmasaydı, programı çok-parçacıklı biçimde yazmak, basit düzeyde paralelleştirme ihtiyacını karşılayabilirdi. Derlenmiş dillerde parçalara bölünen programlar, işletim sistemi seviyesinde gerektiğinde farklı işlemcilere dağıtılabilmektedir. Ancak Python, yorumlanan programların bellek yönetimini CPython düzeyinde, *Global Interpreter Lock*²⁴ (Genel Yorumlayıcı Kilidi – GIL) adı verilen bir mekanizma ile yapmaktadır. GIL sayesinde her Python betiği (*script*) işletim

²⁴ <https://wiki.python.org/moin/GlobalInterpreterLock>

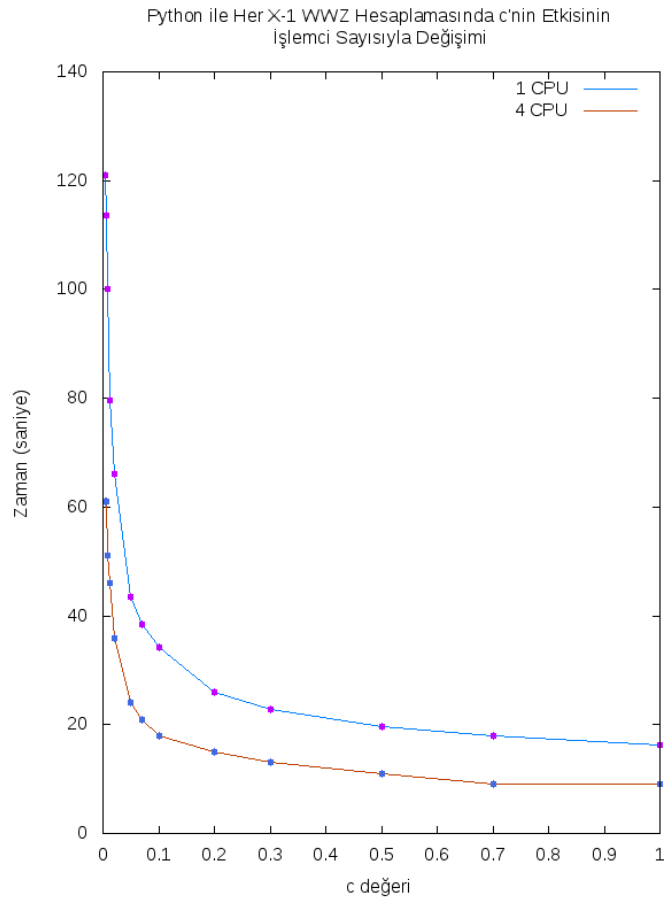
sistemi üzerinde tek bir Python yorumlama işlemi oluşturabilir. Bu durumda çok-parçacıklı yazılan Python programları, işlemlerini tek işlemci üzerinde gerçekleştirmektedir. GIL, Python'un paralel kullanımına engel teşkil ettiğinden kimi zaman bir programlama dili olarak tercih edilmemesine sebep olmaktadır (Greenfield 2011).

Bu durumun önüne geçmek için Python'un *multiprocessing* modülü kullanılmaktadır. Multiprocessing modülü, çok-parçacıklı (threaded) programların yazımında GIL engelini aşacak şekilde tasarlanmıştır. Oluşturulan işlem havuzları (*pool*) işlemin kaç parçaya bölüneceğini baştan belirterek yaratılmaktadır. Bu sayı keyfimize göre verilebileceği gibi, sistemdeki işlemci çekirdeği sayısını dikkate almak verimli olacaktır. Bunun haricinde havuzda çalıştırılacak işlemlerin hangi fonksiyondan üretileceği ve bu fonksiyonların alacağı parametreler bir dizi halinde havuza sunulmaktadır. Ardından havuzdaki işlemler asenkron biçimde çalıştırılır ve sonuçlar birleştirilir.

Multiprocessing modülü ile oluşturulan asenkron işlem havuzlarının bir dezavantajı bulunmaktadır, Python nesnelerinin metodlarını desteklememesi. Bunun temel sebebi, nesnelerin bellekte tuttuğu yer ile ilgilidir. Bir nesnenin metodu çağırıldığında, bellekte kendisi için bir yer ayrılır, bu ayrılan yerde işlemlerini gerçekleştirir ve işlemini bitirdiğinde, kendi ana-objesi tarafından bellekteki bölgesi temizlenir (*destroy*). Oysa multiprocessing modülü, havuzdaki bir işlem tamamlandığında, işlem yok olmadan önce o bellekteyken kendisine *pickle* metoduyla erişip içeriğini okumaya çalışmaktadır. Bu durum, havuzdaki işlemlerin bitip sonuçlarının biraraya getirilememesine yol açmaktadır.

Bunun için sunulan çözümler, yazılan metodu farklı bir fonksiyon içeriğine alıp *pickle* ile bellekten okunabilecek hale getirmek veya mümkünse işlemi bir nesne metodu olarak değil, doğrudan fonksiyon olarak oluşturmaktır.

Her iki yöntemde de çıktının *NumPy* dizisi olmasından dolayı bu parçaların paylaştırılmış bir bellekte kullanılması zorluk çıkarmaktadır. İşlemin verimliliğini test etmek amacıyla doğrudan her parçanın çıktısı belleğe değil, geçici dosyalara yazdığı, ardından bu dosyaların birleştirildiği tamamen farklı bir Python programı yazıldı. Pratik kullanım bakımından programın modüler yapısına son derece ters olsa da, verimlilik gerektiğinde bu işlemin işe yaradığı gözlenmiştir. Yazılan programın çalışma süresinin, 4 çekirdekli bir bilgisayarda işlemin 4 parçaya bölünmesinde yaklaşık yarı yarıya düştüğü gözlenmiştir (Bkz. şekil 3.7). Bu işlemin başarısı dikkate alındığında, koda uygulanmasına karar verilmiştir. Sonuçta *WWZPAR* sınıfı ortaya çıkmıştır. Her ne kadar bir sınıf olsa da, dolayısıyla multiprocessing modülünün havuzunda parçaların bir araya bellek düzeyinde getirilebilmesi sağlanamamış olsa da, havuzdaki her işlemin sonuçlarını geçici dosyalara yazması, ardından bu dosyaların birleştirilmesi yolu izlenmiştir.



Şekil 3.7 Python'un paralel biçimde çalıştırılmasıyla elde edilen süre değişimi

WWZ algoritmasının yüksek verimlilikle kullanılabilmesi için sunulabilecek bir diğer çözüm, Python modülündeki WWT metodunun, FORTRAN kodu ile yazılıp, *f2py*²⁵ derleyicisi ile bir paylaşımlı nesne (*shared object*) olarak derlenmesidir. Böylece Python içerisinde sadece bu işlemin makine dilinde derlenmiş olarak çağırılması sağlanabilir. Her ne kadar kolay kullanım ve yükleme işlemleri bakımından çok pratik bir çözüm olmasa da, bu çalışmanın devamı niteliğinde sunulabilecek bir çözüm olarak karşımıza çıkmaktadır.

3.3.7 Templeton'ın koduna yapılan katkılar

Her ne kadar Python tercümesi süresince Templeton'ın (2002) geliştirdiği FORTRAN koduna ve algoritmasına sadık kalındıysa da, kullanımı kolaylaştıracak, programı anlaşılır kılacak, verimliliğini artıracak veya geliştirilmesini kolay kılacak bazı noktalarda çeşitli katkılar sağlanmıştır.

Bunların başında, programın tek başına çalıştırılabilir olmasının yanında, modüler yapıda yazılmış olması gelmektedir. Program, herhangi bir Python geliştiricisi tarafından kendi sisteminde rahatlıkla kütüphane olarak çağırılacak şekilde geliştirilmiştir. Böylece otomatikleştirilmiş çıktı üretimi, arayüz geliştirilmesi veya herhangi bir teleskoba özel veri setine uygun sistem geliştirilmesinde kolaylıkla kullanılabilir.

Templeton'ın (2002) geliştirdiği program, çeşitli dillerde standart haline gelmiş açıklama satırları olan “%” ve “#” karakterlerini yorumlamaya çalıştığından, alınan verilerin başlık bilgilerinde bu tip karakterlerin bulunması problem yaratabilmektedir. Buna benzer olarak, WWZ çıktısında başlık bilgilerinin yazılması kullanıcının seçeneğine sunulmuş, başlık bilgileri yazılırken açıklama satırları olduğunu ifade etmek için satırlar “#” karakteriyle başlatılmıştır.

²⁵ www.f2py.com

Pek çok bilimsel çalışmada olduğu gibi, gökbilim çalışmalarında da grafiklendirme için oldukça yaygın olarak kullanılan araçlardan birisi Gnuplot²⁶ programıdır. Açık kaynak kodlu bir program olan Gnuplot, hem rahatlıkla erişilebilmekte, hem platformdan bağımsız çalışmakta, hem de kolaylıkla programlanıp, çok yüksek sayıdaki verinin istenen tipteki grafiği elde edilebilmektedir. 3 boyutlu grafik üretimlerinde Gnuplot, ilk parametrenin gruplandırılmasını, ilk parametre değiştiğindeyse bir satır boşluk bırakılmasını istemektedir. Yani $\{x_1, y_1, z_1, x_1, y_2, z_2, x_1, y_3, z_3, x_2, y_1, z_1, x_2, y_2, z_2, x_2, y_3, z_3, x_3, y_1, z_1, x_3, y_2, z_2, x_3, y_3, z_3\}$ şeklindeki bir veri setinin 3 boyutlu grafiklendirilebilmesi için, Gnuplot aşağıdaki gibi bir yapıda girdi dosyasının şekillendirilmesini talep etmektedir:

$x_1 \ y_1 \ z_1$

$x_1 \ y_2 \ z_2$

$x_1 \ y_3 \ z_3$

$x_2 \ y_1 \ z_1$

$x_2 \ y_2 \ z_2$

$x_2 \ y_3 \ z_3$

$x_3 \ y_1 \ z_1$

$x_3 \ y_2 \ z_2$

$x_3 \ y_3 \ z_3$

Yukarıdaki formatta oluşturulan çıktılar, Gnuplot'un *pm3d map* ayarında kullanıldığında iki boyuta indirgenmiş üç boyutlu grafikler elde edilebilmektedir. Geliştirilen Python programı, parametre olarak Gnuplot uyumluluğuna sahip olduğundan, istenildiğinde buna uygun çıktı üretebilmektedir.

Programda, girdi parametreleri çok daha esnek hale getirilmiştir. Templeton'ın (2002) kodu sadece etkileşimli bir girdi sağlamaktadır, bu durum komut satırından veya farklı

²⁶ <http://www.gnuplot.info>

programlar aracılığı ile çağırılarak çıktı üretmede son derece elverişsiz bir yapı sunmaktadır. Python kodu hem parametre olarak girdi alabilmekte, hem de parametreleri bir dosya içerisinden okuyabilmektedir. Ayrıca Templeton'ın (2002) kodunda 50 olarak sabit tuttuğu *Time Divisions* değeri, kullanıcının seçimine bırakılmıştır. Bu değerın yükseltilmesi, üretilen çıktılarda *tau* adımlarını küçülttüğünden, zaman çözünürlüğünü artırmakta, ancak hesaplama süresini de beraberinde artırmaktadır.

Templeton'ın kodunda bazı değişkenlerin tanımlandığı, ancak hiç kullanılmadığı görülmüştür. Benzer kaygıya, bu koddan Java tercümesi yapan VStars²⁷ geliştiricileri de düşmüş, kodlarının açıklama satırlarında “*ne olur ne olmaz*” düşüncesiyle bazı değişkenleri tuttuklarını ifade etmişlerdir. FORTRAN derleyicileri tarafından da “*hiç kullanılmadığı için tanımlanması gerekmediği*” belirtilen bu gibi değişkenler temizlenmiştir. Ayrıca algoritmik bir takım geliştirmeler bu süreçte yapılmıştır. Örneğin iki matrisin birbirine eşitlenmesi için Templeton (2002) FORTRAN kodunda matrisler üzerinden döngüler kurarak değerlerini eşitlemektedir. Python'da buna gerek kalmamakta, doğrudan NumPy ile matrisler eşitlenebilmektedir.

FORTRAN kodunda yer alan *matinv* alt programının (*subroutine*) gerçekleştirdiği matematiksel işlem, NumPy kütüphanesinin fonksiyonları tarafından gerçekleştirilebilmektedir. Ancak herhangi bir versiyon değişiminde olası bir uyumsuzluk gerçekleşmemesi için, bu alt program da tercüme edilmiş ve *matrix_inv* metodu olarak Python kodunda yer almıştır. NumPy ve *matrix_inv* kullanımının yaratacağı performans farkı karşılaştırılarak, aralarında fark oluşmadığı denetlenmiştir.

3.4 Lomb-Scargle Programı

Geliştirilen programın çıktılarını incelemek ve bölüm 4.3'te gösterilen sonuçları literatürle karşılaştırabilmek için periyodogramlar Lomb-Scargle yöntemiyle oluşturulmuştur. Lomb-Scargle algoritması ayrıca kodlanmayarak, SciPy kütüphanesi

²⁷ <http://www.aavso.org/vstar-overview>

içerisinde yer alan algoritmayı daha kullanışlı hale getiren bir program yazılmıştır (bkz: EK 2).

Programın girdi biçimi, *wwz.py* programıyla benzer olacak şekilde tasarlandı. Bölüm 3.2’de belirtilen kod yazımında dikkat edilen noktalara bu program geliştirilirken de uyulmuştur. Programın girdi argümanları çizelge 3.3’te listelenmiştir.

Program tıpkı *wwz.py* gibi Python 2.7 ile uyumlu olarak geliştirilmiştir. İstenirse modül olarak kullanılabilir veya tek başına komut satırından çalıştırılabilir. Ancak kullandığı temel fonksiyon zaten SciPy bünyesinde bulunduğundan, modül olarak kullanım ihtiyacı hissedilmeyeceği düşünülmektedir.

Çizelge 3.3 *ls.py* programının girdi argümanlarının açıklamalı tablosu

Argümanın Kısa Gösterimi	Argümanın Uzun Gösterim	Argüman Açıklaması
-h	--help	Yardım. Programın kullanımı hakkında detaylı açıklamaları standart çıktıya yazdırır.
-f	--file	Girdi dosyası. Analizi yapılacak ışık eğrisinin ASCII dosyasını ifade eder.
-o	--output	Çıktı dosyası. Lomb-Scargle hesaplamalarının yazdırılacağı çıktı dosyasının adı.
-l	--freq-low	Düşük frekans değeri.
-hi	--freq-high	Yüksek frekans değeri.
-n	--freq-num	Frekans sayısı. Düşük frekans ve yüksek frekans aralığının kaç parçaya bölüneceğini ifade eder. 1000 veya 10.000 tercih edilebilecek değerlerdir.
	--no-headers	Çıktı dosyalarına başlık bilgileri yazmaz.

3.5 Seçilen Hedefler ve Veritabanı

Yöntemin (DPS) uygulanacağı hedef cisimler X-ışın çift yıldızlarından, verinin sağlandığı gözlemler ise RXTE uydusunun ASM (Tüm Gökyüzü Kamerası) veritabanından seçilmiştir. RXTE/ASM verilerinin seçilmesinin en önemli nedeni, yüksek enerji bölgesinde, her 1.5 saatte bir tarama yaparak tüm gökyüzünün ~15 yıl süreli kesintisiz gözlemlerine sahip olmasıdır. Gökcisimlerinin bu kadar uzun süreli ve bu kadar sıklıkla alınmış bir başka gözlemsel veritabanı bulunmamaktadır.

3.5.1 X-ışın çift yıldızlarının genel özellikleri

X-ışın çift yıldızları, adından anlaşılacağı üzere, elektromanyetik tayfın X-ışın bölgesinde şiddetli ışıınım yapan sistemlerdir. Bu sistemlerin X-ışın bölgesindeki ışıınım gücü $10^{35} - 10^{38} \text{ erg s}^{-1}$ seviyelerindedir, bu değer Güneş'in toplam ışıınım gücünün 10^2 ile 10^5 katına karşılık gelmektedir (Sterken ve Jaschek, 1996).

İlk X-ışın çifti, Sco X-1, Giacconi ve arkadaşları (Giacconi vd. 1962) tarafından 12 Haziran 1962 tarihinde keşfedilmiştir. Uzun süre keşfedilen cismin doğası anlaşılamadı, eğer kaynak yakınlarda bir yıldız ise, X-ışın gücü, Güneş'in X-ışın gücünden 10^7 - 10^8 kat daha fazla olmalıydı. Bu tarihlerde böyle cisimlerin varlığı bilinmiyordu. X-ışın ölçümlerinin astrometrik hatalarının çok yüksek oluşu da ayrı bir problem teşkil ediyordu, bu sebepten birkaç yıl boyunca cismin optik karşılığı bulunamadı. Sonunda Sco X-1 optik olarak tespit edildi: Sönük ($V \sim 13^m$) mavi bir değişen yıldız. Sco X-1'in bulunmasının ardından en şaşırtıcı yanı, X-ışın parlaklığının optik parlaklığına oranı oldu, bu değer $L_X / L_{opt} \sim 1000$ mertebesindeydi. İlerleyen yıllarda yüzlerce X-ışın çifti keşfedildi ve neredeyse tamamının optik karşılığı bulunabildi. X-ışın çiftlerinin etkileşen yakın çift sistemler olabileceği fikri ilk defa Rus astrofizikçi I. Shklovskiy (Shklovsky 1967) tarafından ortaya atılmıştır.

X-ışın çiftlerinin temel modeli, birbirine oldukça yakın, etkileşen çift yıldızlardan oluşmaktadır. Yıldızlardan birisi anakol yıldızı, dev yıldız veya bazı özel durumlarda

dejenere bir yıldızdan oluşmaktadır. Bu yıldız (optik bileşen) çoğunlukla Roche şişimi ya da yıldız rüzgarı yoluyla diğer yıldıza madde aktarımı yapmaktadır. Diğer yıldız (yoğun bileşen) ancak nötron yıldızı veya karadelik gibi yoğun bir cisim olabilir. Bu tip sistemler “yarı-ayrık” sistemler grubuna dahildirler. Yörüngesel açısal momentumdan dolayı aktarılan madde doğrudan yoğun cismin üzerine düşemez ve bu cismin etrafında bir yığılma diski oluşturur. Yığılma diskinin iç sürtünmelerinden (buna aynı zamanda *viskozite/sürtünme* de denir) dolayı yığılan madde giderek ısınarak yoğun bileşenin üzerine doğru bir spiral madde akışı oluşturur.

Nötron yıldızının yüksek manyetik alana sahip olduğu bazı büyük kütleli X-ışın çiftlerinde yığılma diskleri her zaman oluşmaz. Yoğun bileşen çevresel maddeyi dipol manyetik alan çizgileri boyunca üzerine alır. Yoğun bileşene yakın olan madde sürtünme ile 10^7 K sıcaklığa kadar ısınır. Bu ısınmadan açığa çıkan enerjinin büyük çoğunluğu karacisim veya bremsstrahlung ışınlamıyla X-ışın tayf bölgesinde yayınlanır.

Bu sistemlerde gözlenen yüksek X-ışınımın fiziksel mekanizması, kütleçekim enerjinin kinetik enerjiye dönüşmesine dayanmaktadır. Madde yığılması, bu mekanizma için oldukça uygun bir ortam sağlamaktadır. M kütleli bir yoğun cisim, $\frac{dM}{dt} \equiv \dot{M}$ oranıyla madde yığmaktaysa, yığılmadan açığa çıkan L_{acc} ışınlam gücü şu şekilde ifade edilmektedir:

$$L_{acc} = G M \dot{M} / R \quad \dots (52)$$

Burada G evrensel kütle çekim sabiti, R ise yoğun cismin yarıçapıdır. Bu denklemde enerjinin büyük kısmının yoğun cismin yüzeyinde, R yarıçapında açığa çıktığı varsayılmıştır. M , 1 Güneş kütlelik ($M_{\odot}$) bir yoğun cisim, $\dot{M}$ değerinin $10^{-8} M_{\odot} \text{ yıl}^{-1}$ alınması ile galakside gözlenen herhangi bir X-ışın çiftinin maksimum ışınlam gücüne çok yakın olan $10^{38} \text{ erg s}^{-1}$ ’lik enerji üretebilmektedir.

Üç farklı tür yoğun cisim için madde birikimi yoluyla ürettikleri enerjinin verimi ve nükleer füzyon yoluyla üretilen enerjinin verimini şöyle kıyaslayabiliriz. Verim

$\eta = GM/Rc^2$ olmak üzere,

$$L_{acc} = \eta Mc^2 \dots (53)$$

Yukarıdaki denklemden hareketle,

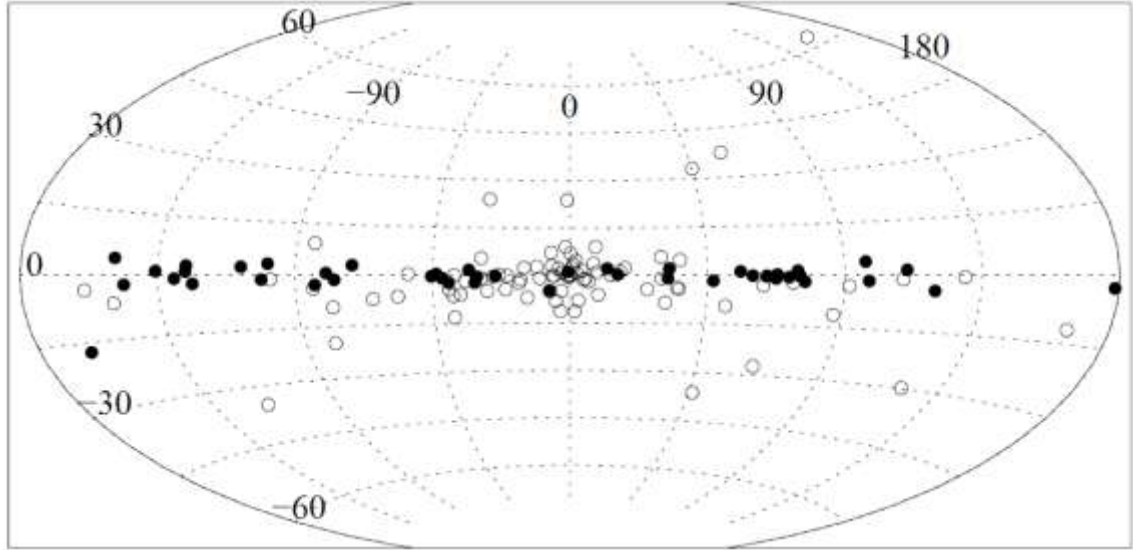
- Nötron yıldızları için $\eta \sim 0,1$
- Karadelikler için $\eta \sim 0.06 - 0.42$
- Beyaz cüceler için $\eta \sim 0.001$
- Nükleer tepkimeler için $\eta \sim 0.01 - 0.001$

bulunmaktadır. Bu sonuç, nötron yıldızları ve karadelikler gibi yoğun cisimlerin etrafında oluşan yığılma disklerinin, kütleyi enerjiye dönüştürmede evrendeki en etkili mekanizma olduğunu göstermektedir (Seward ve Charles, 2010).

X-ışın çiftleri iki grupta incelenebilir, düşük kütleli X-ışın çiftleri (Low-Mass X-Ray Binaries – LMXRB) ve büyük kütleli X-ışın çiftleri (High-Mass X-Ray Binaries – HMXRB). Ayrım genellikle X-ışın ve optik parlaklık oranlarından belirlenmektedir. HMXRB'ler için $L_X / L_{opt} 10^{-3}$ ile 10 değerleri arasındadır, yani büyük kütleli X-ışın çift yıldızlarının optik bölgedeki parlaklıkları, elektromanyetik tayfın X-ışın bölgesindeki parlaklıklarından daha fazladır. LMXRB'ler içinse durum tam tersinedir, $L_X / L_{opt} 10$ ile 10^4 arasındadır; optik bölgede oldukça sönük cisimlerdir.

X-ışın çift yıldızlarının galaktik koordinatlara göre gökyüzündeki dağılımları şekil 3.8'de görülmektedir. Şekilde toplam 86 LMXRB ve 52 HMXRB gösterilmektedir. HMXRB'lerin galaktik düzlemde, LMXRB'lerin ise galaksi merkezindeki yoğunluk dağılımı dikkat çekicidir (Grimm vd. 2002). Aynı X-ışın çiftlerinin gökadaki dağılımının üstten görünümü ise şekil 3.9'da görülmektedir. Koordinat sisteminin orijini galaktik merkezde yer almaktadır. Güneş, bir beşgen işaretiyle gösterilmiştir. Şekildeki

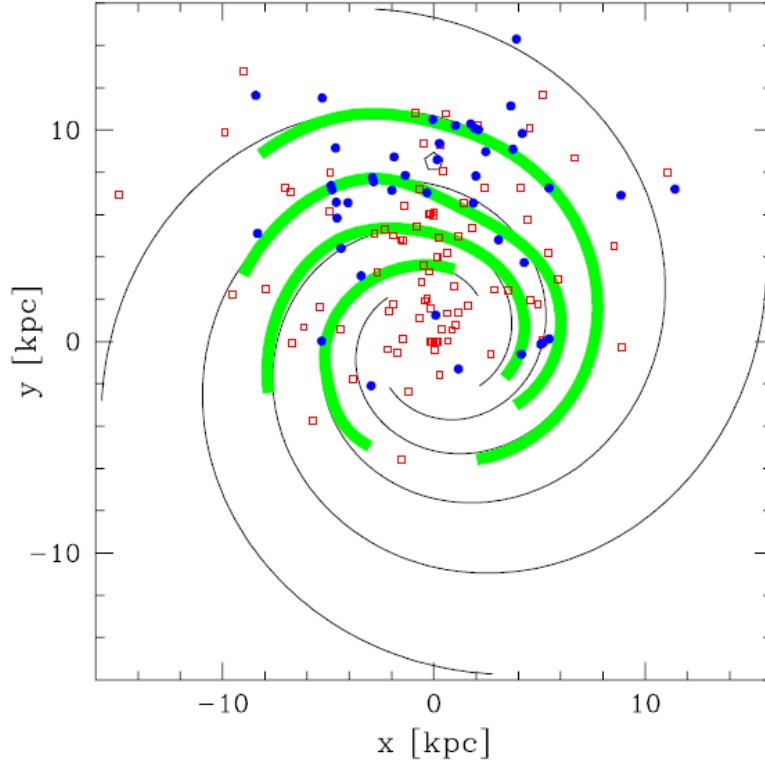
ince sürekli çizgiler, 4 kollu 12 derecelik spiral modeli temsil etmektedir. Kalın çizgi ise optik ve radyo gözlemlerinden elde edilen HII bölgelerinin spiral modelini göstermektedir (Grimm vd. 2002).



Şekil 3.8 Büyük kütleli (dolu daireler) ve küçük kütleli (boş daireler) X-ışın çift yıldızlarının gökadamadaki dağılımı (Grimm vd. 2002)

HMXRB'lerde kütle kaybeden üye, optik olarak daha parlaktır ve erken tayf türündendir. Oldukça genç (Pop I) cisimlerdir ve gökadamadaki dağılımları galaktik düzlemde yoğunlaşmaktadır. Maddenin yığılması, manyetik alan çizgileri boyunca ilerleyerek, manyetik kutuplarda sonlanmaktadır. HMXRB'ler de kendi içinde iki temel gruba ayrılmaktadır (Maraschi vd. 1976). İlk grupta kütle kaybeden üye erken tip (B3'ten daha erken) OB devisi veya süperdevdir. Bu tip sistemlerde şiddetli yıldız rüzgârları 1. Lagrange noktasından yoğun bileşene akarak kütle kaybı gerçekleşir. CenX-3 bu sistemlere tipik örneklerden biridir. Parlak bir X-ışın kaynağının diğer bileşen yıldızla çok yakın olması, yıldızın atmosferi ve rüzgârı üzerinde gözlenebilir etkilere sebep olur. X-ışın kaynağı yörüngesi üzerinde dolanırken, dev veya süperdev yıldızdan atılan maddenin içinden geçerken oluşan X-ışın yayılımındaki değişim, rüzgârın yapısını incelemeye kullanılabilmektedir. Eğer rüzgâr yoğunsa ve yüksek derecede iyonize olmamışsa, düşük enerjili X-ışınını gölgeleyerek, kaynağın gözlenen tayfını değiştirebilir. Öte yandan eğer X-ışın kaynağı çok parlaksa, rüzgârdaki maddeyi tamamen iyonlaştırabilir. İyonize olmuş gaz X-ışını soğuramayacağından, kaynağın

tayfı oldukça az etkilenmiş olur.

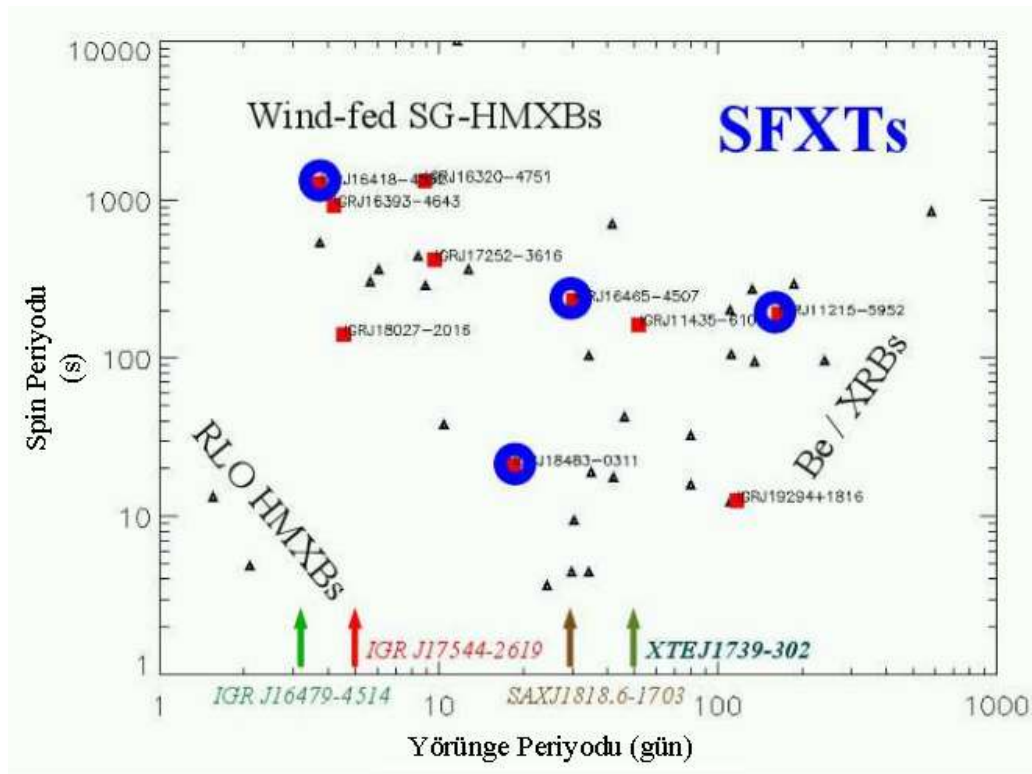


Şekil 3.9 Gökadanın üstten görünümünde küçük kütleli (boş kareler) ve büyük kütleli (dolu daireler) X-ışın çift yıldızlarının dağılımı (Grimm vd. 2002)

İkinci gruptaysa kütle kaybeden üye Be yıldızıdır; yani tayfında optik salma çizgileri barındıran B tayf türünden bir yıldızdır. Be yıldızları, X-ışın çiftlerinin keşfinden çok önce çalışılmaya başlanmış ve doğası pek anlaşılamamıştır. Salma çizgileri genellikle şiddet değişimi gösterir ve bu değişimin aralığı oldukça geniştir, saatler ve yıllar mertebesinde olabilir. Bu yıldızların nötron yıldızlarıyla çift sistem oluşturması, (*pulsar*) yirminci yüzyılın ikinci yarısından sonra tekrar ilgi konusu olmalarını sağlamıştır.

Be yıldızlarından kaynaklı X-ışın parlamaları, yıldızın ekvator bölgesinden atılan maddenin yığılma (birikim) diski oluşturmasıyla özdeşleştirilmektedir. Bu öneri, Be yıldızlarının neredeyse merkezci kuvvetleri tarafından dağıtılacak kadar hızlı dönmelerini de destekler niteliktedir. Böyle bir yıldızın etrafında dolanan herhangi bir

yoğun cisim ise söz konusu yığılmış maddenin içinden geçecektir, bu da X-ışın üretimine neden olur. Üstelik söz konusu yörünge eliptikse, X-ışın yayınının dönemi ile yoğun cismin dolanma dönemi arasında bir ilişki olacaktır. Bu ilişki Be sistemlerinin pulsasyon dönemleri ile yörüngesel dönemleri karşılaştırılınca ortaya çıkmaktadır. (Corbet 1984) Birincil bileşeni Be yıldızlarından oluşan, ikincil bileşeniyse nötron yıldızlarından oluşan X-ışın çift sistemlerinin pulsasyon dönemleri ve yörüngesel dönemlerinin grafiği (Corbet diyagramı) şekil 3.10’da görülebilir.



Şekil 3.10 X-ışın çiftlerinden oluşturulmuş pulsasyon dönemleri ve yörüngesel dönemlerinin grafiğini gösteren Corbet diyagramı (Sidoli 2011)

LMXRB’lerin temel modeli 1970’lere Cambridge Üniversitesi’ndeki teorisyenler tarafından ortaya atılmıştır (Pringle ve Rees, 1972). Evrimleşmiş geç tayf türünden bir yıldız, Roche şişimini doldurup, birinci Lagrange noktasından yoğun cismin kütleçekimsel alanına madde aktarmaktadır. Bu durumda yoğun cisim beyaz cüce, nötron yıldızı veya karadelik olabilir. Yıldızdan çıkan madde sahip olduğu açısal momentumdan dolayı, doğrudan yoğun cismin yüzeyine düşemez. Ancak etrafında bir yörüngede dolanarak bir yığılma diskinin oluşmasına sebep olur. Bu model ile

HMXRB'lerin mekanizmasının ayrıldığı temel nokta, yıldızın yoğun cisme kütle aktarımındadır. Yıldız rüzgârları düşük açısal momentum barındırdığından, büyük kütleli yığılma diskleri oluşturmaya yetmemektedirler. Öte yandan, düşük kütleli (geç tayf türünden) bir yıldızın rüzgârı zayıf olacağından, sağlayacağı X-ışın yayılımı da gözlenemeyecek düzeyde olur. Bu durumda yıldızın söz konusu maddeyi yoğun cismin kütleçekimsel alanına yönlendirmesi gerekmektedir. Bu sistemlerin gözlenebilmesi için bazı koşulları yerine getiriyor olması gerekmektedir.

1. Yoğun cisim: Eğer X-ışın kaynağı zayıfsa ($<10^{33}$ erg s⁻¹) ve yakınsa (100 pc'lik alanda), yoğun cisim bir beyaz cücedir. Bu ışınma gücünden daha güçlü olması halinde nötron yıldızı veya karadelik olmalıdır. Ancak eğer X-ışın parlamaları veya pulsasyonları gözleniyorsa, nötron yıldızıdır. Eğer çok parlaksa ($>10^{37}$ erg s⁻¹) ve yumuşak bir X-ışın tayfına sahipse, karadelik olabilir.
2. Yığılma diski: Yüksek eğimli sistemler, X-ışın çiftinin örtülme aşamasında diskin farklı evrelerdeki yapısının belirlenebilmesine imkan sağlar. Diskin iç kısmı oldukça sıcaktır ve moröte tayf bölgesinde (UV) oldukça parlaktır. Eğer sistemin eğimi yoğun cisimi doğrudan görmemize hiç izin vermeyecek kadar yüksekse, bu sistemler ancak diskin altında ve üstünde beliren sıcak, X-ışını yayan koronaları sayesinde tespit edilebilirler (Cyg X-3 gibi).
3. İkincil bileşen: Çoğunlukla LMXXRB'lerde doğrudan gözlenemeyecek kadar sönük ve soğukturlar. Ancak tutulma anında kızıl-öte tayf bölgesinde (IR) gözlem yaparak, kütle kaybeden ikincil bileşenin düşük kütleli olduğu ve evrimleşmiş olduğu gözlenebilmektedir.

Bir diğer çeşit X-ışın çifti ise *geçici kaynaklar* (transient sources) olarak adlandırılmaktadırlar. Uzun süre X-ışın etkinliği göstermeyip, birkaç gün gibi kısa süreliğine aktivite gösterirler, sonra tekrar *sakin* (quiescence) hallerine dönerler. X-ışın parlaklığındaki keskin artıştan dolayı *X-ışın novası* adını da alırlar. X-ışın parlaklığının arttığı dönemde, optik parlaklığı da şiddetli artış (5-6 kadir) gösterir.

3.5.2 RXTE teleskobunun genel özellikleri

Rossi X-ray Timing Explorer (RXTE) teleskobu Aralık 1995 tarihinde fırlatılmıştır ve Ocak 2012 tarihine kadar çalışmıştır. Yüksek zaman çözünürlüğüyle ($\Delta t < 10\mu s$) X-ışın bölgesinde gözlem yapacak ve bütün gökyüzünü tarayacak şekilde tasarlanmıştır. Üç dedektör barındırmaktadır:

1. Geniş alan orantısal sayacı (*proportional counter array – PCA*)
2. Yüksek Enerji zamansal deneyi/düzeneği (*HEXTE*)
3. Tüm-gökyüzü monitörü (*ASM*)

PCA dedektörünün 6500 cm^2 gibi geniş bir etkin alanı bulunmaktadır ve teleskobun en hassas cihazıdır. Beş tane tamamen özdeş, metan-xenon ile doldurulmuş orantısal sayaç dizisinden oluşmaktadır. Yüzey alanı, çok kısa zaman aralıklarında olabildiğince çok foton toplayabilmek için büyük tasarlanmıştır. 1 metre uzunluğundaki bir kolimatör, açısal görüş alanını 1 derecelik *FWHM*'a (*Full Width at Half Maximum – Yarı Yükseklikteki Tam Genişlik*) daraltacak biçimde yerleştirilmiştir. Her bir dizinin (array) boyutu yaklaşık olarak $0,3 \times 1,0 \times 1,3$ metredir ve beşinin toplam ağırlığı 750 kg yöresindedir. Zaman çözünürlüğü 1 mikrosaniyedir ve enerji çözünürlüğü 6 keV'de %18'in altındadır.

PCA'nın yanında High-Energy X-ray Timing Experiment (*HEXTE*) dedektörü bulunmaktadır ve aynı yönelime, aynı görüş açısıyla bakmaktadır. Dedektörler sezyum-iyot/sodyum-iyot sintilatörleridirler: sekiz fotokatlandırıcı tübün karşısına sekiz kristal yerleştirilerek oluşturulmuştur. Dedektörün algılayabildiği enerji 15 – 250 keV gibi geniş bir aralıktır ve 60 keV'deki enerji çözünürlüğü %15'tir. Zaman çözünürlüğü 8 mikrosaniyedir. Bu dedektörün yüzey alanı $2 \times 800\text{cm}^2$ 'dir.

PCA ve HEXTE, parlak X-ışın kaynaklarının yüksek çözünürlüklü tayfsal gözlemlerini ve yüksek zaman çözünürlüklü fotometrik gözlemlerini uzun-dönemli yapacak şekilde tasarlanmıştır. Zaman çözünürlüğü tabii ki sinyalin gücüne bağlıdır. RXTE, gelen fotonu 5 μs hassasiyetle algılayacak düzeydedir. Parlak cisimlerdeki ışınlam değişimini

milisaniye mertebesinde, sönük olanlarınsa saatler mertebesinde algılayacak düzeydedir.

Tüm gökyüzünü monitörü, (*all-sky monitör, ASM*) *X-ışın gölge kamerası (X-ray shadow cameras)* adı verilen 3 adet cihazdan oluşmaktadır. Küçük ölçekli, konum-duyarlı üç orantısal kameranın önüne yerleştirilen tek boyutlu kodlanmış maskeler, X-ışını kameralarının üzerinde gölge şeklinde desenler oluşturur. Oluşan görüntüdeki üst üste binen gölgeleri bir bilgisayar ayıklar ve her kaynağın pozisyonunu tespit eder. Dedektörlerin döndürülmesiyle, bütün gökyüzünün %80'i her 1,5 saatte taranabilmektedir. Bu şekilde 50'den fazla bilinen güçlü X-ışın kaynağı uydunun ömrü boyunca (~15 yıl) gözlenmiştir. Kameraların toplam etkin yüzey alanı 90 cm²'dir ve algılanabilir enerji aralığı 2 – 10 keV'dir. Her bir kameranın açısal görüşü ise 6 x 90°'dir. ASM dedektörü Güneş'e yakın bölge hariç, gökyüzünün tamamını gözleyebildi.

4. BULGULAR

4.1 RXTE/ASM Verilerinin İncelenmesi ve Karşılaştırılması

RXTE uydusunun ASM kamerası, bütün gökyüzünü taradığı, sık ve kesintisiz veri aldığından, kaynakların dinamik güç spektrumunu elde etmek için oldukça uygundur. Kotze ve Charles da (2012) bu gerekçeyle, RXTE/ASM verilerini kullanarak çeşitli X-ışın çiftlerine ait dinamik güç spektrumlarını incelemişlerdir. Kotze ve Charles (2012), Foster'ın (1996) dalgacık yöntemini değil, LS yöntemini pencere kaydırarak uygulamıştır. Bu bakımdan hem LS yöntemiyle WWZ'nin karşılaştırılması, hem de geliştirilen WWZ Python programının tutarlılığının incelenmesi açısından Kotze ve Charles'ın (2012) çalıştığı bazı X-ışın çift yıldızları bu çalışmada test yıldızı olarak seçilmişlerdir.

Çalışılan test yıldızlarının daha önceden literatürde yayınlanan dönemlilikleri, girdi parametresi belirtmek bakımından fikir vermektedir. Bu parametreleri Kotze ve Charles (2012) da kullanmıştır. Kullanılan test yıldızları ve literatürden toplanan dönem aralıkları çizelge 4.1'de belirtilmiştir. Yörünge dönemlerinden çok daha uzun olan bu dönemlere *superorbital dönem (sop)* adı verilmektedir.

Çizelge 4.1 Çalışmada kullanılan hedef cisimlerinin literatürdeki dönem değerleri

Yıldızın Adı	Literatürdeki Dönemi (gün)	Referans
LMC X-4	30	Wen vd. 2006
LMC X-3	90	Cowley vd. 1991
HER X-1	33 - 37	Leahy ve Igna 2010
SMC X-1	50 - 70	Wen vd. 2006
X0114+650	31	Wen vd. 2006
X1636-536	46	Charles vd. 2008
CYG X-1	150 & 290	Lachowicz vd. 2006
CYG X-2	50 – 80, 60 - 90	Charles vd. 2008, Wen vd. 2006
SCO X-1	2.6	Kudryavstev vd. 1989

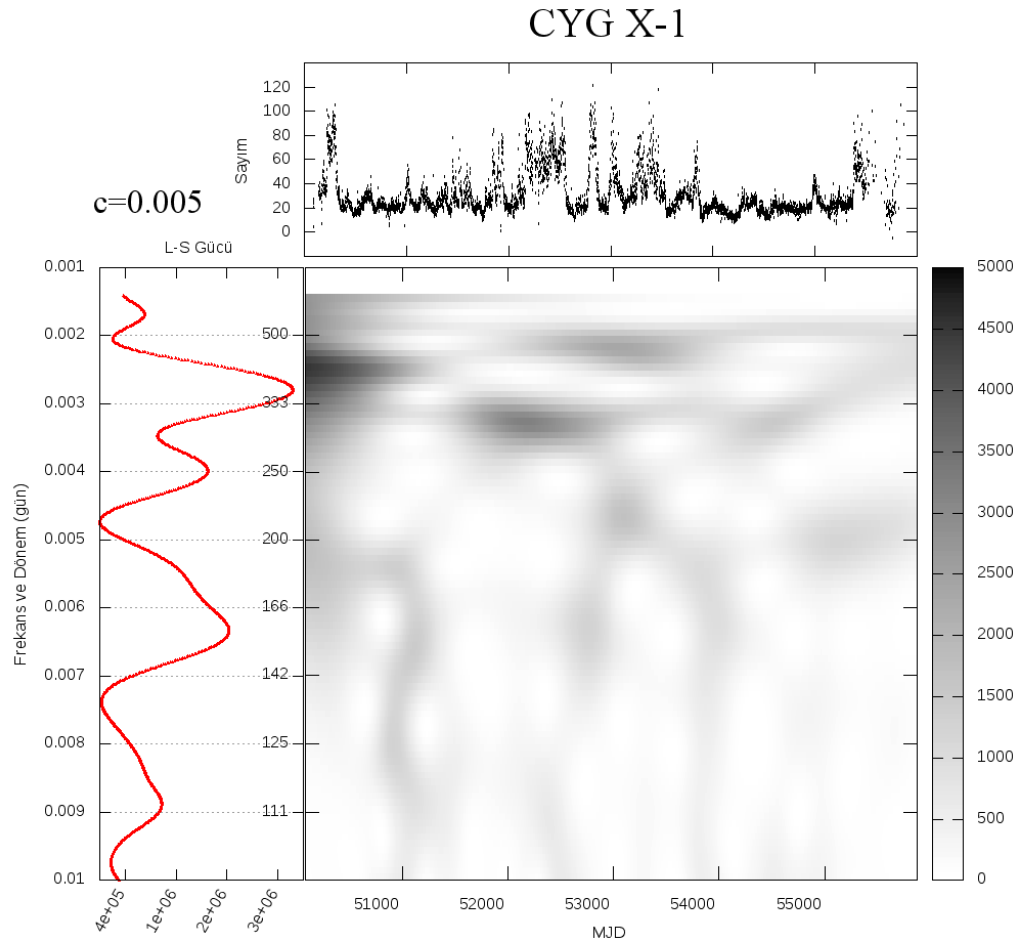
Analiz edilen veriler, RXTE/ASM arşivinden²⁸ alınmıştır. Bütün bantlardaki birleştirilmiş *dwel-by-dwell* veriler üzerinde çalışılmıştır. Grafiklerde üstte, verinin 1 günlük ortalaması alınmış ışıık eğrisi görölmektedir. Grafiklerde *dwel-by-dwell* verinin gösterilmemesinin nedeni, ışıık eğrisinin görsel olarak daha rahat algılanabilmesi içindir. Şekillerde solda verilen grafiklerdeki periyodogramlar, tüm veri setine *ls.py* programıyla Lomb-Scargle algoritması uygulanarak elde edilmiştir. Sağda ise *wwz.py* programıyla elde edilen WWZ çıktısı (dinamik güç spektrumları) bulunmaktadır. Karşılaştırma yapabilmek amacıyla, Kotze ve Charles'ın (2012) elde ettiğı grafikler de sunulmuştur. Sonuçların grafiğe aktarılması Gnuplot ile yapılmıştır. Örnek bir Gnuplot kodu EK 3'te sunulmuştur.

CYG X-1 yıldızı için elde edilen sonuçlarda, Lachowicz vd.'nin (2006) belirttiğı 290 günlük dönem MJD 51500-55000 gün aralığında görölebilmektedir. Özellikle Townsend'in (2010) Lomb-Scargle algoritmasının bu veriler üzerinde sağlıklı sonuç üretmediğı, ancak WWZ'nin beklenen değeri verdiğı söylenebilir (Bkz. şekil 4.1).

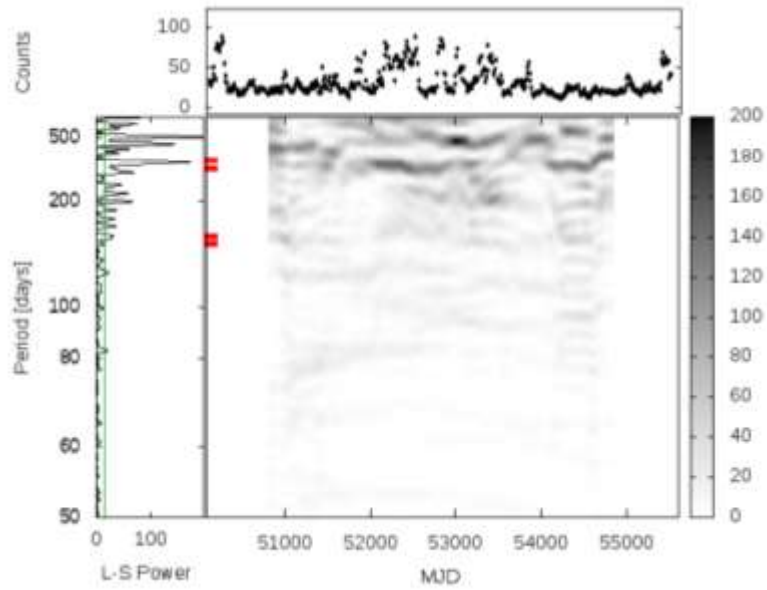
HER X-1 yıldızı için elde edilen sonuçlar şekil 4.3'te gösterilmiştir. Hem literatür ile oldukça uyumlu, hem de şekil 4.4'te gösterilen çalışma ile tutarlı sonuçlar elde edilmiştir.

Şekil 4.5'te sonuçları gösterilen CYG X-2 yıldızının analizi de, hem literatür hem de referans çalışmayla uyumludur. Özellikle 58-65 gün aralığında görölen dönemlilik maksimum güç üretmektedir. Maksimumlardan birisini Lomb-Scargle yöntemi belirgin biçimde gösterse de, diğeri göstermemektedir.

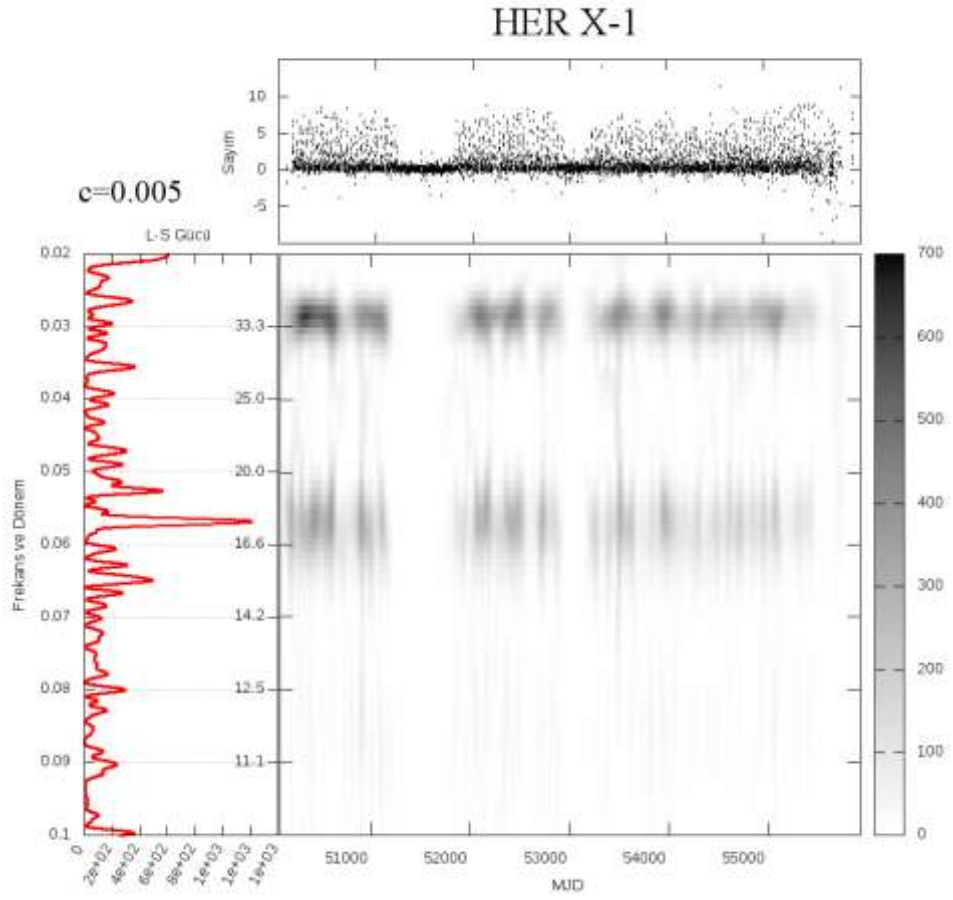
²⁸ <http://xte.mit.edu/asmlc/>



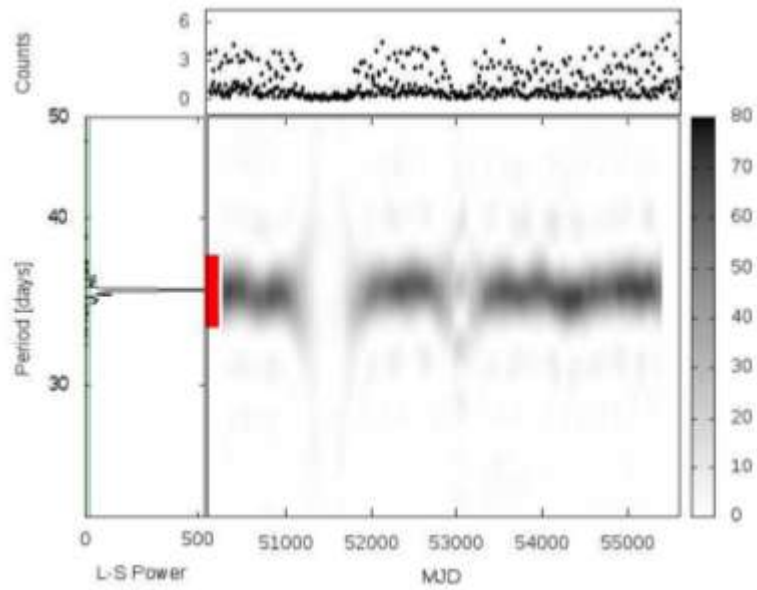
Şekil 4.1 CYG X-1 yıldızının WWZ analizi



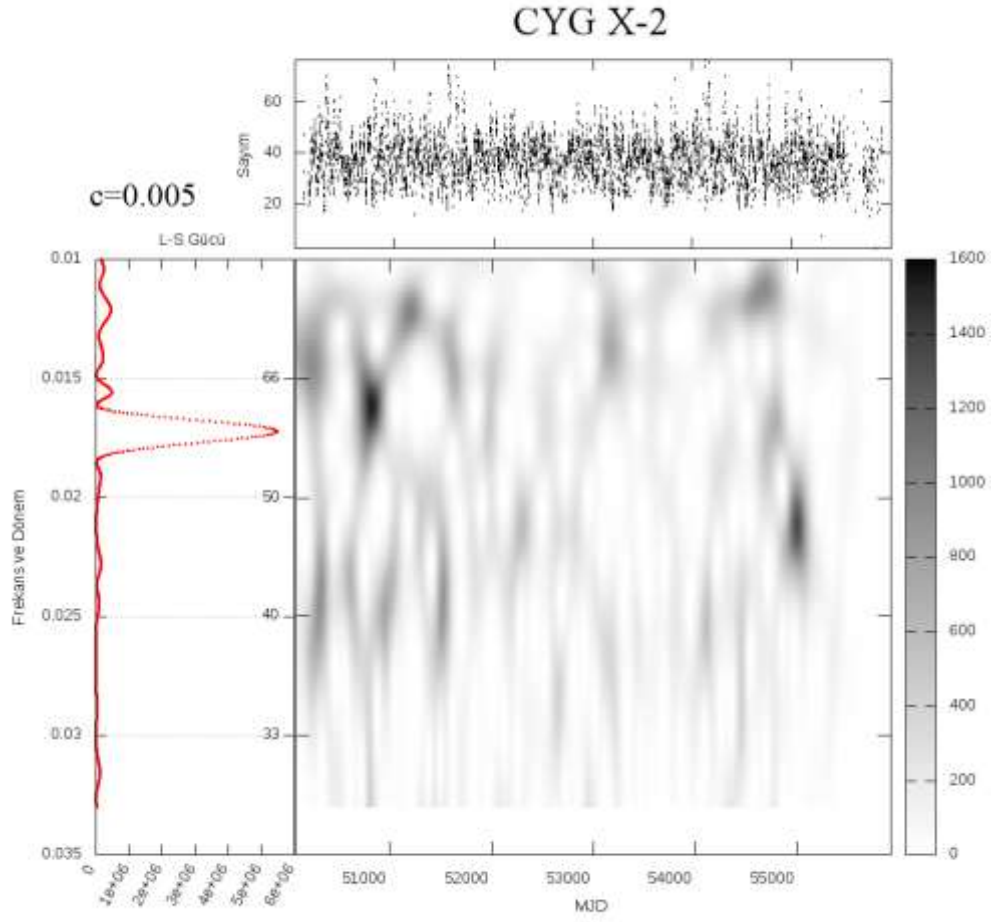
Şekil 4.2 CYG X-1 yıldızının Kotze ve Charles (2012) tarafından yapılan analizi



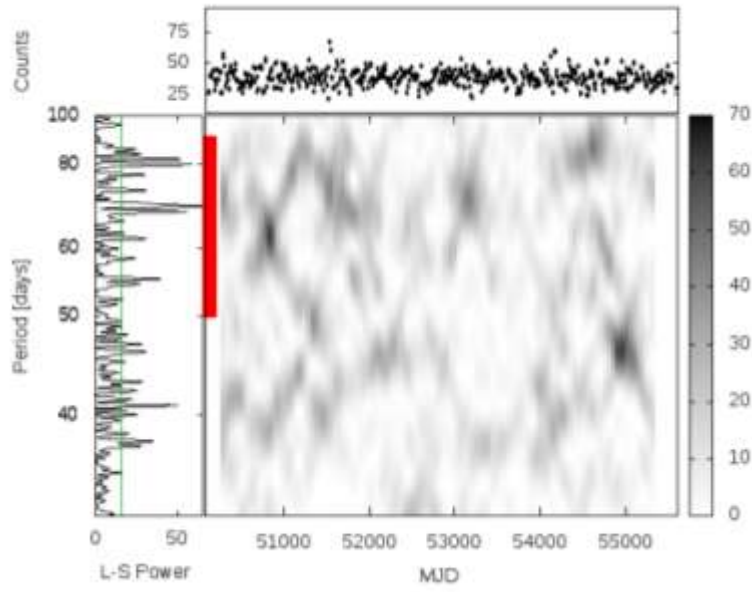
Şekil 4.3 HER X-1 yıldızının WWZ analizi



Şekil 4.4 HER X-1 yıldızının Kotze ve Charles (2012) tarafından yapılan analizi



Şekil 4.5 CYG X-2 yıldızının WWZ analizi



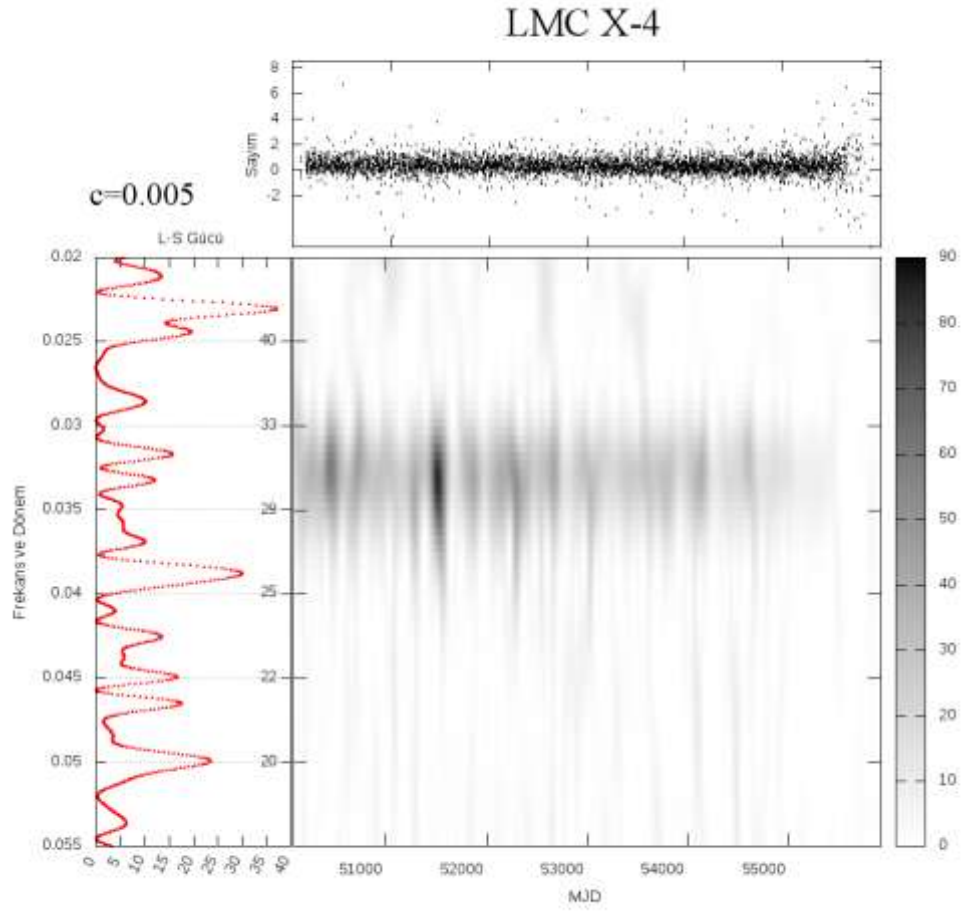
Şekil 4.6 CYG X-2 yıldızının Kotze ve Charles (2012) tarafından yapılan analizi

Şekil 4.7’de görülen LMC X-4 analizi, 28-33 gün dönem aralığında maximum güç oluşturacak şekilde sonuç üretmiştir. Genellikle bu dönemliliğin sabit olduğu, ancak MJD 51600 civarında maksimum olduğu görülmektedir. Yine, Townsend’den (2010) uyarlanan Lomb-Scargle yöntemi bu ışık eğrisi için uygun çıktı üretememiştir. Şekil 4.8’de görülen Kotze ve Charles’ın (2012) çalışmasındaysa, hem Python ile üretilen WWZ çıktıları uyumludur, hem de kendi çalışmalarındaki Lomb-Scargle çıktısı doğru periyotta maksimum vermektedir.

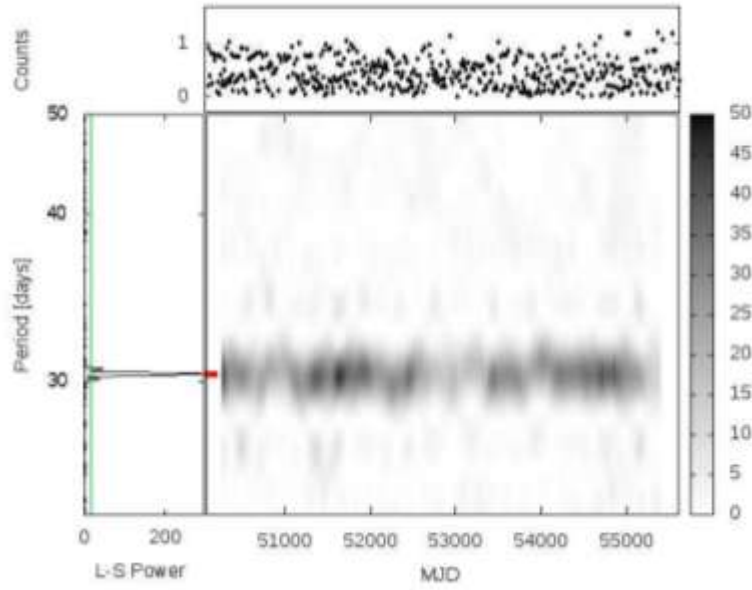
Sco X-1 yıldızı hem WWZ (Şekil 4.9) ile elde edilen sonuçlarda, hem de Kotze ve Charles’ın (2012) sonuçlarında (Şekil 4.10) 166 gün civarında dönemlilik göstermektedir. Dönemlilik, MJD 50800 gün civarında maksimum güç göstermektedir. Yine, tüm veriye uyarlanan Lomb-Scargle çıktısının yanlış yerde maksimum verdiği şekil 4.9’da görülebilir.

SMC X-1 yıldızı, dinamik güç spektrumu analizinde ilginç sistemlerden bir tanesidir ve türünün tek örneğidir. Işık eğrisinde görülen değişim yapısı, aslında çok belirgin bir dönemliliğe sahiptir. Bu değişim şekil 4.11’de hem Lomb-Scargle sonucunda hem de WWZ sonucunda görülebilir. Lomb-Scargle algoritması ~57 günlük dönemde oldukça güçlü bir pik vermektedir. Bu sonuç hem Wen vd.’nin (2006) 50-70 gün aralığındaki süperorbital dönem beklentisine uymaktadır, hem de WWZ ile elde edilen dönem değişim aralığı ile çakışmaktadır. Gerek WWZ çözümüyle, gerekse Kotze ve Charles’ın (2012) çözümüyle uyum gösteren sonuçlar, yıldızın süperorbital döneminin 45-66 gün aralığında değişimler gösterdiğine, özellikle MJD 50800 ve MJD 53800 tarihlerinde süperorbital dönemin ~45 güne düştüğüne, diğer tarihlerde ortalama 55 günlük dönemlilik gösterdiğine işaret etmektedir.

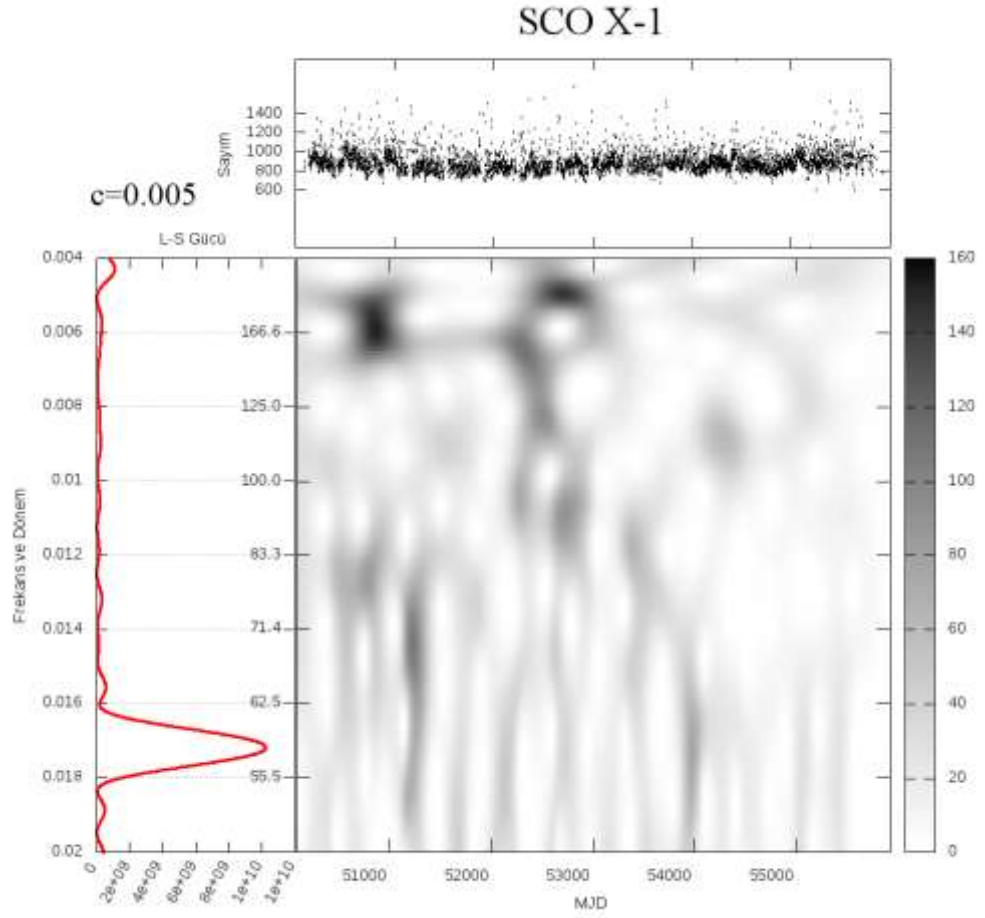
Şekil 4.13’te WWZ sonucu görülen X0114+650 yıldızı da Kotze ve Charles (2012) (Şekil 4.14) ile uyumlu bir dinamik spektruma sahipken, Townsend’in (2010) Lomb-Scargle algoritmasıyla uyumsuz bir sonuç elde edilmiştir. Wen vd. (2006) bu sistem için 31 günlük süperorbital periyot bulmuştur.



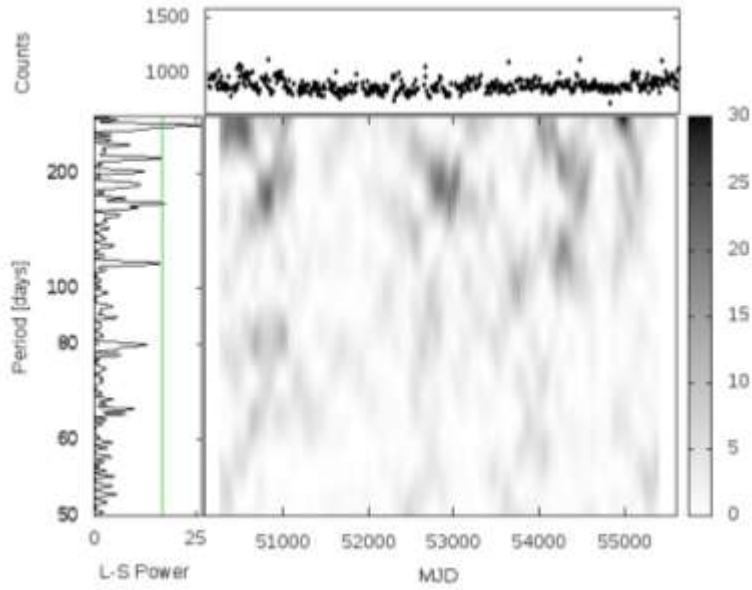
Şekil 4.7 LMC X-4 yıldızının WWZ analizi



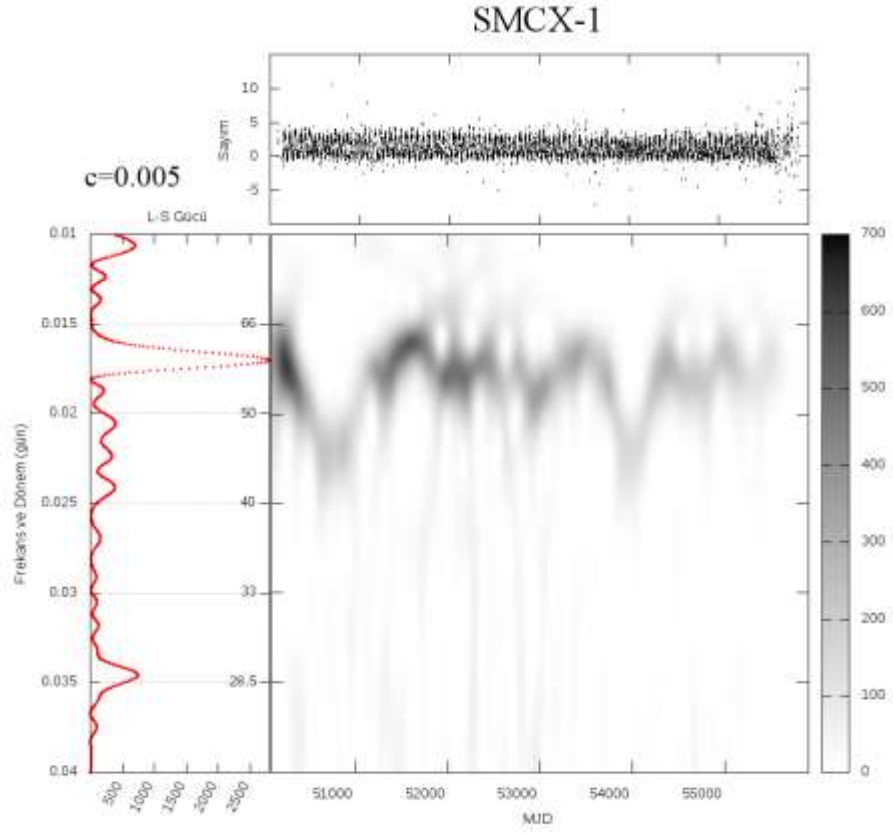
Şekil 4.8 LMC X-4 yıldızının Kotze ve Charles tarafından yapılan analizi (2012)



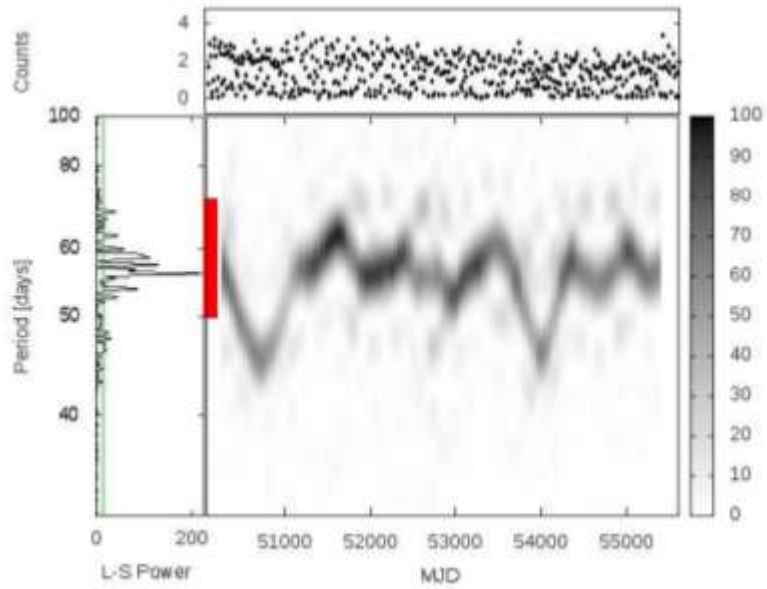
Şekil 4.9 SCO X-1 yıldızının WWZ analizi



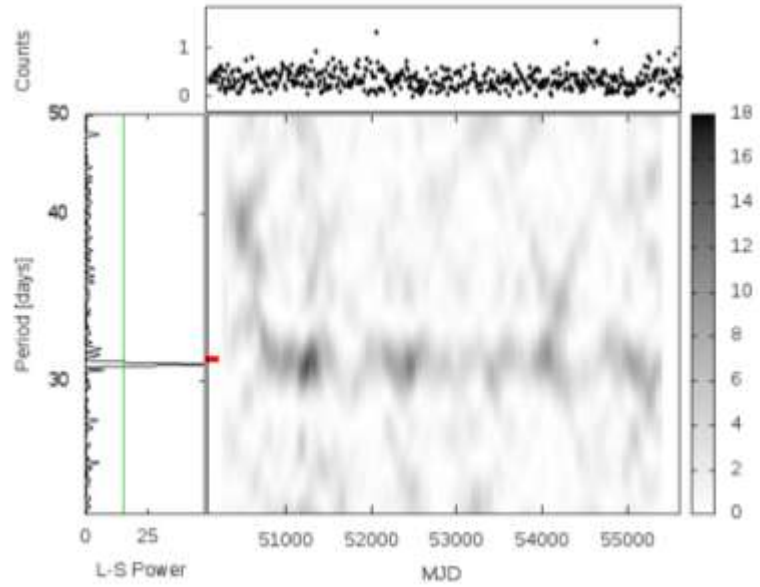
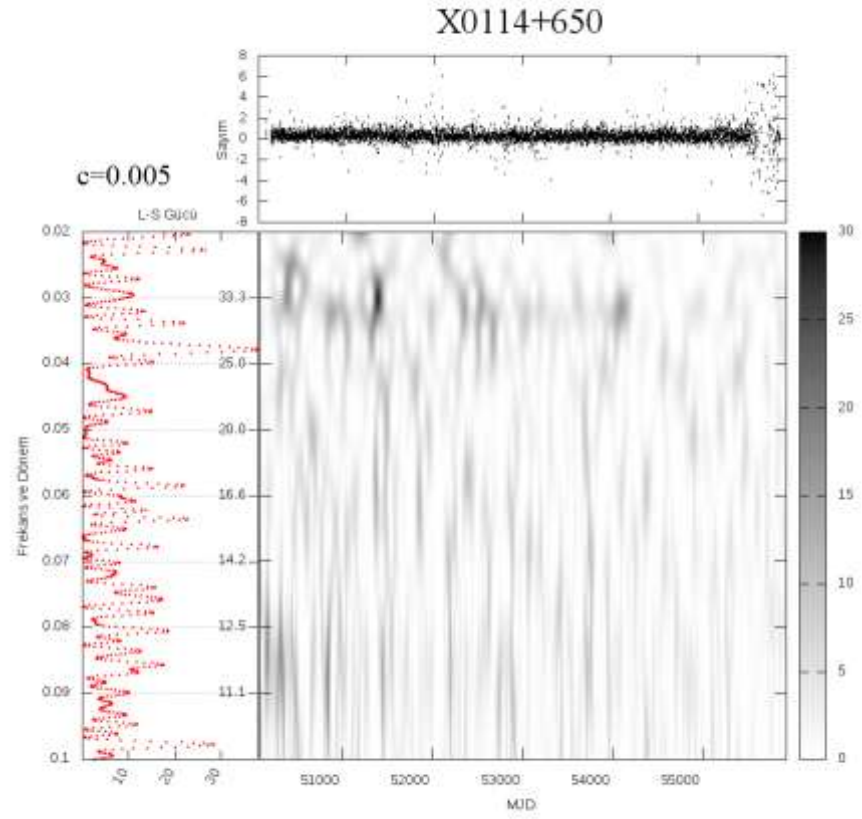
Şekil 4.10 SCO X-1 yıldızının Kotze ve Charles (2012) tarafından yapılan analizi

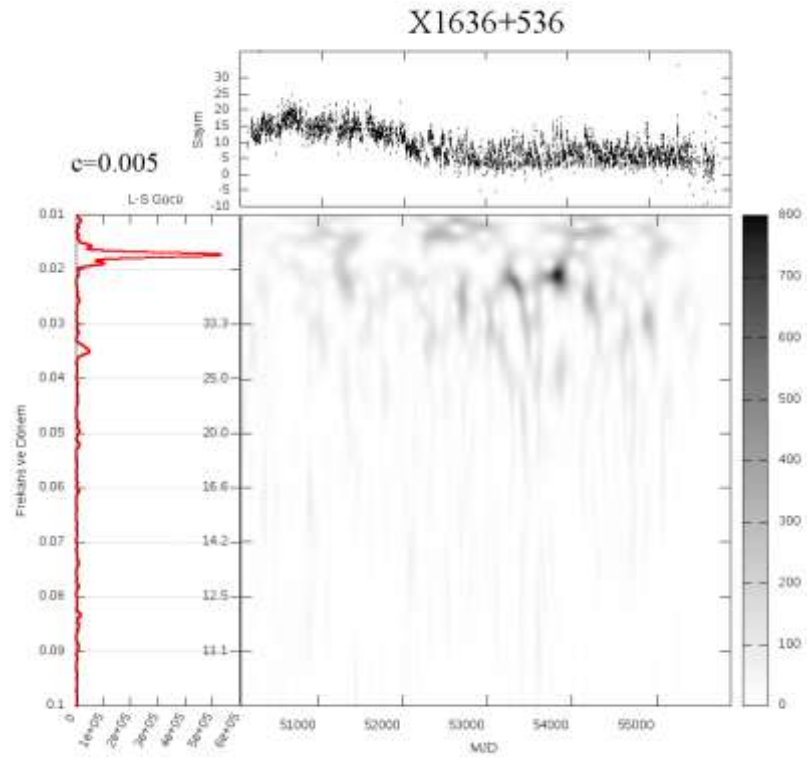


Şekil 4.11 SMC X-1 yıldızının WWZ analizi

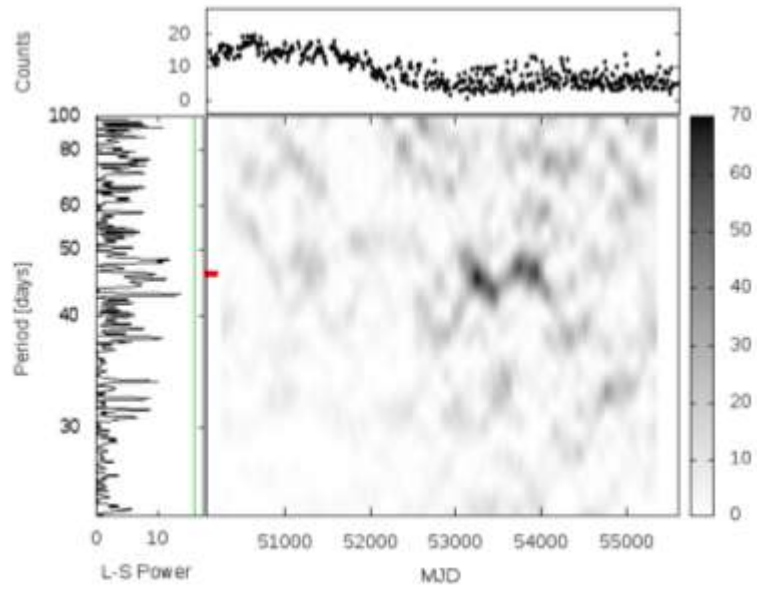


Şekil 4.12 SMC X-1 yıldızının Kotze ve Charles tarafından yapılan analizi (Kotze ve Charles 2012)





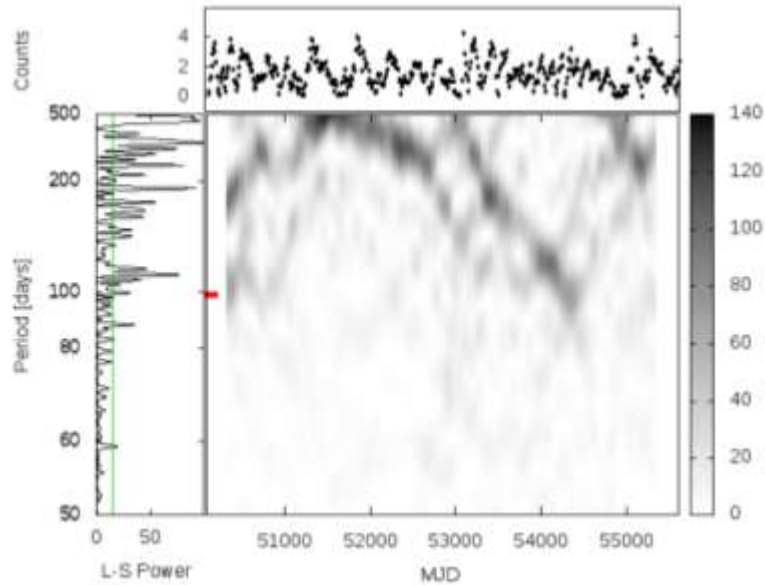
Şekil 4.15 X1636-536 yıldızının WWZ analizi



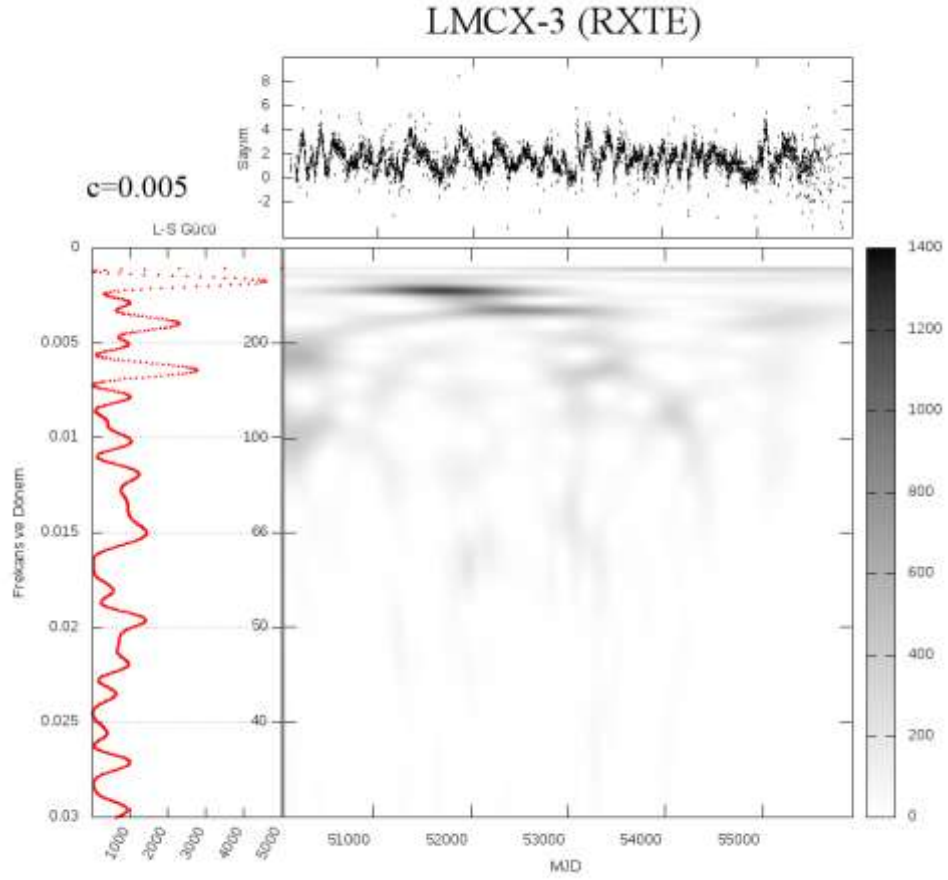
Şekil 4.16 X1636-536 yıldızının Kotze ve Charles tarafından yapılan analizi (Kotze ve Charles 2012)

X1636-536 yıldızının WWZ analizi ile Townsend (2010) Lomb-Scargle analizi uyumlu sonuçlar sunmuştur (bkz. şekil 4.15). Charles vd. (2008) tarafından hesaplanan 46 günlük periyot ile uyum göstermektedir. Şekil 4.16’da görülen Kotze ve Charles (2012) çalışmasındaysa, dinamik güç spektrumu uygun aralıkta maksimum verse de, iki boyutlu Lomb-Scargle sonucu bunu tam olarak yansıtamamaktadır.

LMC X-3 yıldızının RXTE/ASM verilerinin dışında, Dr. C. Brocksopp tarafından sağlanan optik ve Dr. S. Kitamoto tarafından sağlanan X-ışın (Ginga) gözlemlerinden de yararlanılmıştır. Cowley ve arkadaşları (1991) tarafından 90 gün olarak belirtilen *sop* dönemi, Kotze ve Charles (2012) tarafından incelenerek, dinamik güç spektrumunda yıldızın döneminin 90 günde sabit olmadığı, 500 gün ile 90 gün arasında değiştiği görülmüştür (bkz. şekil 4.17).

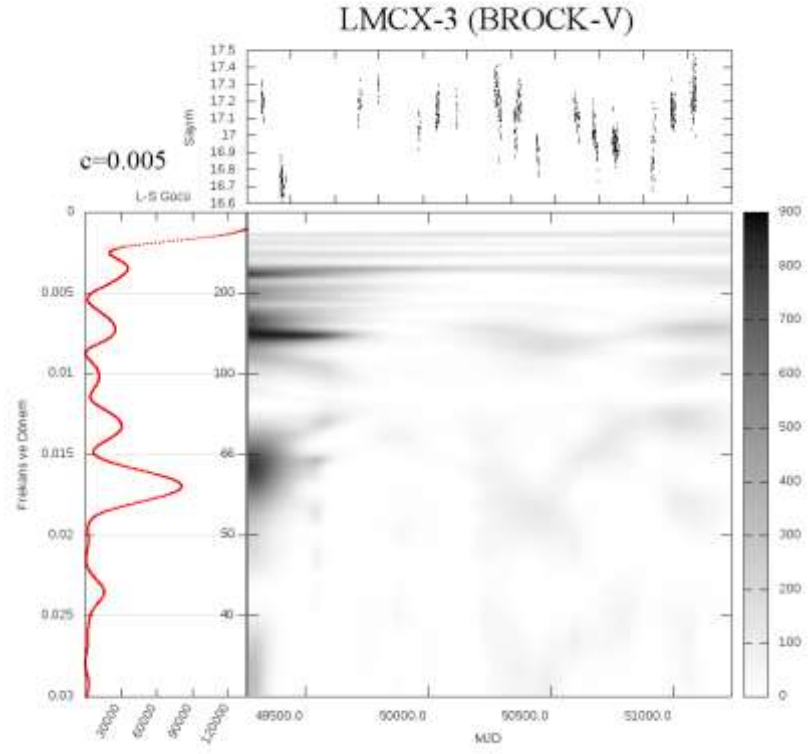


Şekil 4.17 LMC X-3 yıldızının Kotze ve Charles (2012) tarafından yapılan analizi

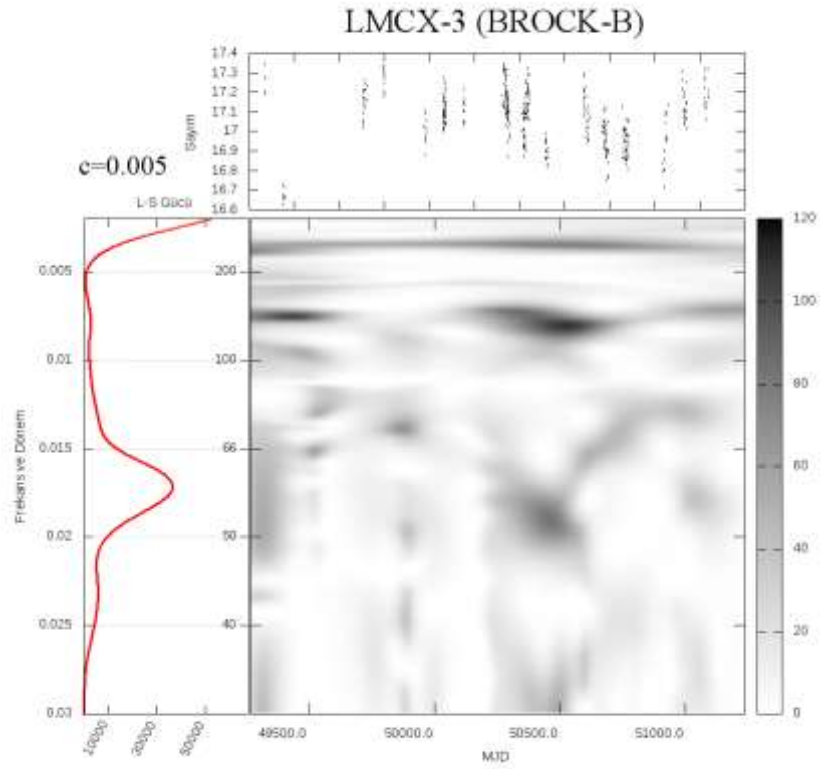


Şekil 4.18 LMC X-3 yıldızının RXTE verilerinin WWZ analizi

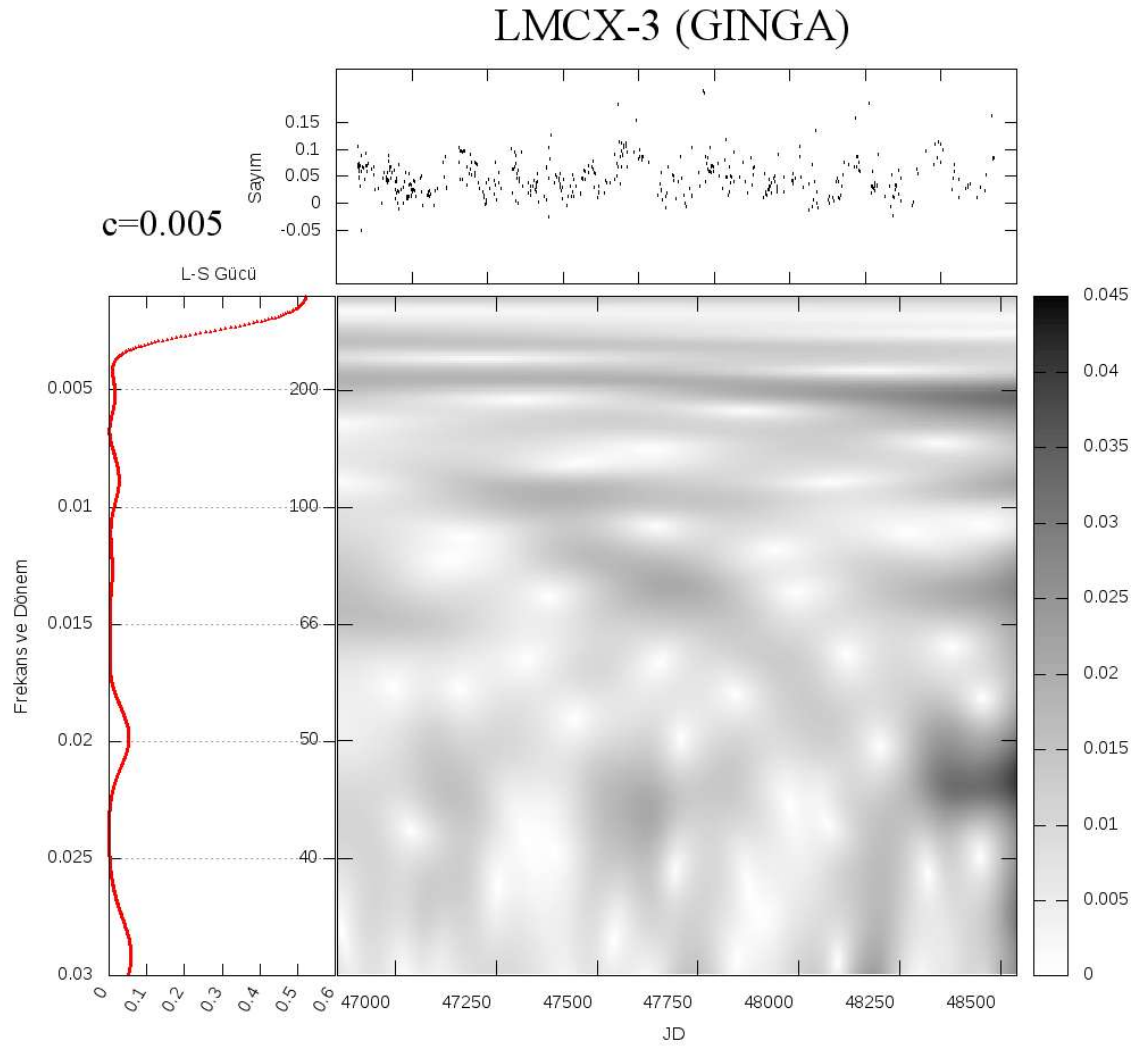
Şekil 4.19’da LMC X-3 yıldızının V bandı, şekil 4.20’de ise B bandı gözlemlerinden elde edilen WWZ analizleri görülmektedir. Lomb-Scargle analizleri her iki veri seti için de 60 günde dönelilik göstermektedir. Ayrıca çok küçük frekans değerinde maksimuma yakınsamaktadır ancak Foster (2010), Lomb-Scargle analizlerinde bu tip sınır frekans değerlerine güvenilmemesi gerektiğinin altını çizmektedir. Optik verinin WWZ analizleri Lomb-Scargle sonuçlarıyla uyumlu biçimde 60 gün komşuluğunda dönelilik göstermekle beraber, 150 gün ve 250 gün civarında güç üretmemiştir. Işık eğrilerine bakıldığında, verilerin son derece seyrek aralıklarla alındığı görülmektedir.



Şekil 4.19 LMC X-3 yıldızının V bandı optik gözlemlerinin WWZ analizi



Şekil 4.20 LMC X-3 yıldızının B bandı optik gözlemlerinin WWZ analizi



Şekil 4.21 LMC X-3 yıldızının 1-6 keV aralığında elde edilmiş GINGA verilerinin WWZ analizi

5. TARTIŞMA ve SONUÇ

Dinamik güç spektrumu (DPS) üretmede WWZ (Foster 1996) yöntemi, X-ışın çift yıldızlarının eşit aralıklarla elde edilmemiş gözlemsel verilerinde *sop* dönemlerinin zamana bağlı değişimini incelemeye oldukça başarılı bir araç olarak karşımıza çıkmaktadır. DPS elde etmenin avantajı, gök cisimlerinin periyodik davranışlarının zamana bağlı olarak incelenmesini sağlamasıdır.

Bu çalışma kapsamında, daha önce dinamik güç spektrumları Kotze ve Charles (2012) tarafından pencere kaydırılarak Lomb-Scargle algoritmasıyla elde edilen yıldızlar incelenmiş, WWZ ile elde edilen sonuçların tamamen tutarlı olduğu gözlenmiştir. WWZ algoritmasının güncel tekniklerle uygulanabilmesi için bir Python programı geliştirilmiştir. Ayrıca kıyaslama amacıyla Townsend'in (2010) Lomb-Scargle yöntemiyle iki boyutlu periyodogramlar da elde edilmiştir.

Geliştirilen WWZ programının Templeton (2002) tarafından yazılan FORTRAN koduyla karşılaştırması yapılmış, korelasyon katsayıları 0.99 mertebesinde bulunmuştur. Bulgular bölümünde sunulan grafikler de, programın sonuçlarının literatürdeki sonuçlarla uyuştuğunu göstermektedir.

İncelenen gök cisimleri arasından özellikle X0114+650, SCO X-1, LMC X-4, HER X-1, CYG X-1 kaynaklarında Townsend'in (2010) Python için uyarlanan Lomb-Scargle algoritmasıyla elde edilen periyodogramların, yıldızların dönem değişimlerinin yapısını açıklamakta yetersiz kaldığı gözlenmiştir. Bunun için ya Lomb-Scargle algoritması Kotze ve Charles'ın (2012) yaptığı gibi pencerelendirilerek dinamik güç spektrumları elde edilmelidir, ya da benzer yolla WWZ analizi yapılmalıdır.

WWZ yöntemi, karmaşık matematiksel hesaplamalar gerektirdiğinden Lomb-Scargle yöntemine göre daha yavaş çalışmaktadır. Bu problemin bir nebze önüne geçebilmek için program paralelleştirilmiş ve paralelleştirme sonucunda elde edilen sonuçların

değişmediği tespit edilmiştir. Parallelleştirme sonucunda 4 çekirdek (CPU) ile yapılan hesaplamalarda yaklaşık 2.5 kat hızlanma sağlanmıştır.

Geliştirilen programlar EK 1 - 2’de sunulduğu gibi, yazarın kişisel GitHub²⁹ hesabında da güncel olarak paylaşılmaktadır.

²⁹ <http://github.com/eaydin>

KAYNAKLAR

- Bracewell, R. N. 2000, The Fourier Transform and Its Applications, 3rd edition, McGraw-Hill, Boston.
- Brigham, E. O. 1988. The Fast Fourier Transform and Its Applications, Englewood Cliffs, NJ: Prencite-Hall, Inc., 448 p.
- Charles, P., Clarkson, W., Cornelisse, R. and Shih, C. 2008. Observational properties of accretion discs: Precessing and warped? New Astronomy Reviews, Volume 51, Issue 10-12, pp. 768-774.
- Clarkson, W. I., Charles, P. A., Coe, M. J., Laycock, S., Tout, M. D. and Wilson, C. A. 2003. Long-term Properties of Accretion Discs in X-ray Binaries – I. The variable third period in SMC X-1. Monthly Notices of the Royal Astronomical Society, Volume 339, pp. 447-454
- Corbet, R. H. D. 1984. Be/neutron star binaries - A relationship between orbital period and neutron star spin period. Astronomy and Astrophysics (ISSN 0004-6361), Vol. 141, no. 1, Dec. 1984, pp. 91-93.
- Cowley, A. P., Schmidtke, P. C., Ebisawa, K., Makino, F., Remillard, R. A., Crampton, D., Hutchings, J. B., Kitamoto, S. and Treves, A. 1991. Discovery of a long-term periodic variation in LMC X-3. Astrophysical Journal, Part 1 (ISSN 0004-637X), Vol. 381, Nov. 10, 1991, pp. 526-533.
- Deeming, T. J. 1975. Fourier Analysis with Unequally-Spaced Data, Astrophysics and Space Science, Volume 36, Issue 1, pp.137-158.
- Dworetsky, M. M., 1983. A period-finding method for sparse randomly spaced observations of 'How long is a piece of string?', Monthly Notices of the Royal Astronomical Society, Vol. 203, June 1983, pp. 917-924.
- Foster, G. 1996. Wavelets for Period Analysis of Unevenly Sampled Time Series, The Astronomical Journal, Volume 112, Number 4, 1709.
- Foster, G. 2010. Analyzing Light Curves: A Practical Guide. Lulu.com.
- Giacconi, R., Gursky, H., Paolini, F. R. and Rossi, B. B., 1962. Evidence for x Rays From Sources Outside the Solar System. Physical Review Letters, vol. 9, Issue 11, pp. 439-443.
- Greenfield, P. 2011. What Python Can Do for Astronomy, Astronomical Data Analysis Software and Systems XX, ASP. Conf. Ser., 442,425.

- Grimm, H. J., Gilfanov, M. and Sunyaev, R. 2002. The Milky Way in X-rays for an outside observer. Log(N)-Log(S) and luminosity function of X-ray binaries from RXTE/ASM data. *Astronomy and Astrophysics*, v.391, pp. 923-944.
- Grossman, A., Kronland-Martinet, R. and Morlet, J. 1989. *Wavelets: Time-Frequency Methods and Phase Space*, Springer, Berlin, p.2.
- Grossman, A., Morlet, J. 1984. "Decomposition of Hardy functions to square integrable wavelets of constant shape. *SIAM J. Math. Anal.*, 15, pp. 723-736.
- Jetsu, L., Porceddu, S., Lyytinen, J., Kajatkari, P., Lehtinen, J., Markkanen, T. and Toivari-Viitala, J. 2013. Did the Ancient Egyptians Record the Period of the Eclipsing Binary Algol—The Raging One? *The Astrophysical Journal*, Volume 773:1.
- Jetsu, L., Porceddu, S., Lyytinen, J., Kajatkari, P., Lehtinen, J., Markkanen, T. and Toivari-Viitala, J. 2013. Did the Ancient Egyptians Record the Period of the Eclipsing Binary Algol—The Raging One?, *The Astrophysical Journal*, Volume 773:1.
- Jurkevich, I. 1971. A Method of Computing Periods of Cyclic Phenomena, *Astrophysics and Space Sciences*, Volume 13, Issue 1, pp.154-167.
- Kotze, M. M. and Charles, P. A., 2012. Characterizing X-ray Binary Long-term Variability, *Monthly Notices of the Royal Astronomical Society*, 420, 1575.
- Kudryavstev, M. I., Mamontova, N. A., Svertilov, S. I. and Tolstaya, E. D. 1989. Indication of 62-hour periodicity in SCO X-1 from experimental X-ray data obtained on the Prognoz-9 satellite. *Pis'ma v Astronomicheskii Zhurnal* (ISSN 0320-0108), Vol. 15, Dec. 1989, p. 1072-1080.
- Lachowicz, P., Zdziarski, A. A., Scharzenberg-Czerny, A., Pooley, G. G. and Kitamoto, S. 2006. Periodic long-term X-ray and radio variability of Cygnus X-1. *Monthly Notices of the Royal Astronomical Society*, Volume 368, Issue 3, pp. 1025-1039.
- Lathi, B. P., 1992. *Linear Systems and Signals*, Carmichael, Calif: Berkeley-Cambridge Press, 656 pp.
- Leahy, D. A. and Iga, C. D. 2010. RXTE-based 25 Day Cycle Turn-on Times for Hercules X-1. *The Astrophysical Journal*, Volume 713, Issue 1, pp. 318-324.
- Lomb, N., R. 1976. Least-squares Frequency Analysis of Unequally Spaced Data, *Astrophysics and Space Science*, Volume 39, Issue 2, p 447-462.
- Lupton, R. H. 2009. Optical Scientific Pipelines: The Intestinal Flora, and How Not to Reinvent it Again, *High Performance Computing in Observational Astronomy: Requirements and Challenges*, 12-16 October, IUCAA, Pune, India.

- Maraschi, L., Huckle, H. E., Ives, J. C. and Sanford, P. W. 1976. Spectral characteristics of transient X-ray sources. *Nature*, vol. 263, pp. 34-36.
- Pearson, K. 1895. Notes on Regression and Inheritance in the Case of Two Parents. *Proceedings of the Royal Society of London*, 58: 240-242.
- Percy, J.R., Mashintsova, M., Nasui, C.O., Palaniappan, R. and Seneviratne, R. 2008. Further Studies of the Photometric Variability of Bright Pulsating Red Giants. *Publications of the Astronomical Society of the Pacific*, 120:523-530.
- Pihajoki, P., Valtonen, M. and Ciprini, S. 2013. Short time-scale periodicity in OJ 287, *Monthly Notices of the Royal Astronomical Society*, Volume 434, Issue 4, pp. 3122-3129.
- Porceddu, S., Jetsu, L. Markkanen, T. and Toivari-Viitala, J. 2008. Evidence of Periodicity in Ancient Egyptian Calendars of Lucky and Unlucky Days, *Cambridge Archaeological Journal*, Volume 18, Issue 03, pp. 327-339.
- Pringle, J. E., Rees and M. J. 1972. Accretion Disc Models for Compact X-Ray Sources. *Astronomy and Astrophysics*, Vol. 21, p. 1.
- Robitaille, T. P., Tollerud, E. J., Greenfield, P., Droettboom, M., Bray, E., Aldcroft, T., Davis, M., Ginsburg, A., Price-Whelan, A. M., Kerzendorf, W. E., Conley, A., Crighton, N., Barbary, K., Muna, D., Ferguson, H., Grollier, F., Parikh, M. M., Nair, P. H., Günther, H. M., Deil, C., Woillez, J., Conseil, S., Kramer, R., Turner, J. E. H., Singer, L., Fox, R., Weaver, B. A., Zabalza, V., Edwards, Z. I., Bostroem, K. A., Burke, D. J., Casey, A. R., Crawford, S. M., Dencheva, N., Ely, J., Jenness, T., Labrie, K., Lim, P. L., Pierfederici, F., Pontzen, A., Ptak, A., Refsdal, B., Servillat, M. and Streicher, O. 2013. Astropy: A community Python package for astronomy, *Astronomy and Astrophysics*, Vol. 558, A33, P.9.
- Scargle, J.D. 1982. Studies in astronomical time series analysis. II - Statistical aspects of spectral analysis of unevenly spaced data. *Astrophysical Journal*, Part 1, vol. 263, Dec. 15, 1982, pp. 835-853.
- Schwarzenberg-Czerny, A. 1989. *Monthly Notices of the Royal Astronomical Society* (ISSN 0035-8711), vol. 241, Nov. 1, 1989, pp. 153-165.
- Schwarzenberg-Czerny, A. 1996. Fast and Statistically Optimal Period Search in Uneven Sampled Observations, *Astrophysical Journal Letters*, v.460, p.L 107.
- Seward, F. D. and Charles, P. A. 2010. *Exploring the X-ray Universe*, Cambridge University Press.
- Shklovsky, I. S., 1967. On the Nature of the Source of X-Ray Emission of SCO XR-1. *Astrophysical Journal*, Vol. 148, p.L1.

- Sidoli, L., 2011. Review on latest progress on Supergiant Fast X-ray Transients and future direction, *Advances in Space Research*, Volume 48, Issue 1, pp. 88-94.
- Stellingwerf, R.F. 1978. Period determination using phase dispersion minimization, *Astrophysical Journal*, Part 1, Vol. 224, pp. 953-960.
- Sterken, C. and Jaschek, C. 1996. *Light Curves of Variable Stars*, Cambridge University Press.
- Templeton, M. R. 2004. Time-Series Analysis of Variable Star Data, *JAAVSO*, Volume 32, 41.
- Townsend, R. and H., D. 2010. Fast calculation of the Lomb-Scargle periodogram using graphics processing units. *The Astrophysical Journal Supplement Series*, Vol 191, pp. 247-253.
- Wen, L., Levine, A. M., Corbet, R. H. D. and Bradt, H. V. 2006. A Systematic Search for Periodicities in RXTE ASM Data. *The Astrophysical Journal Supplement Series*, Volume 163, Issue 2, pp. 372-392.

EKLER

EK 1. wwz.py Kaynak Kodu	84
EK 2. ls.py Kaynak Kodu	117
EK 3. Örnek Gnuplot Kodu	121
EK 4. CentOS 6.x Üzerine Python 2.7.6 Yüklenmesi	123

EK 1 wwz.py Kaynak Kodu

```
1 #!/usr/bin/env python
2 # -*- coding: utf-8 -*-
3 #   wwz.py
4 #   6 June 15:17
5 #   Copyright 2014 M. Emre Aydin <emre.m.aydin@gmail.com>
6 #   http://about.me/emre.aydin
7 #   http://github.com/eaydin
8
9 """
10     A Weighted Wavelet Z-Transformation Application for Python.
11     Translated by M. Emre Aydin - emre.m.aydin@gmail.com
12     http://about.me/emre.aydin
13     Available at http://github.com/eaydin
14
15     For details, read the docstrings of WWZ class.
16 """
17
18 import math
19 import sys
20 import os
21 from datetime import datetime
22
23 # Checking if Python is 2.7
24 if not sys.version_info[:2] == (2, 7) :
25     print "You need Python 2.7 for this script to run."
26     raise SystemExit
27
28 # Checking if we have numpy
29 try :
30     import numpy
31 except :
32     print "Please Install Numpy"
33     raise SystemExit
```

```

34
35 # Checking if we have argparse (which is default in 2.7 so this is
36 # probably a useless check since we already check if Python is 2.7)
37 try :
38     import argparse
39 except :
40     print "Argparse Not Available?! You sure this is Python 2.7?"
41     raise SystemExit
42
43
44 # The WWZ Class Begins
45
46 class WWZ(object) :
47     """The Main class object.
48     This object class does not get any arguments.
49     Available methods are:
50         readfile()
51         roundtau()
52         maketau()
53         makefreq()
54         matrix_inv()
55         wwt()
56         writefile()
57         writ.gnu()
58
59     Arguments :
60         fileName : the input filename, should be the lightcurve.
61         outputfileName : the output filename.
62         flo : the low frequency value. Float.
63         fhi : the high frequency value. Float.
64         df : the frequency step. Float.
65         dcon : the C Window constant. Float.
66         timedivisions : the The Divisions value, choose 50.0 if not
67         sure, that is the default used by Templeton.

```

```

68         max_periods : set True if you want the second output file.
69         It outputs the tau's with maximum period values for easier
70         estimation.
71         gnuplot_compatible : splits tau values by a blank line if set
72         True, so that pm3d of gnuplot easily maps the plot.
73     """
74
75     def __init__(self) :
76         """Initializing the object"""
77
78     def readfile(self, fileName) :
79         """Read the input file.
80         The argument is the file pointer, not the filename as a string.
81         The values in file should be delimited with spaces or tabs.
82         Ignores lines starting with # and %, as if they're comment lines.
83
84         Returns two arrays :
85             Time value, read from the first column of input file.
86             Magnitude value, read from the second column of input file.
87         """
88
89         time = []
90         magnitude = []
91
92         for line in fileName :
93             # Check if it's a comment line
94             if line.strip()[0] != "%" and line.strip()[0] != "#" :
95                 line_time = float(line.split()[0])
96                 line_mag = float(line.split()[1])
97                 time.append(line_time)
98                 magnitude.append(line_mag)
99
100         fileName.close()
101

```

```

102         # Just a routine check for parameter number equality
103         # This should be cleaned up a bit
104         if len(time) != len(magnitude) :
105             print "Number of Time and Magnitude input do not match. \
106                 Please check the input file."
107             raise SystemExit
108
109         # Return two arrays
110         return time, magnitude
111
112     def roundtau(self, darg) :
113         """Rounds the tau's. from G. Foster's Code.
114         This is actually called by the maketau method.
115         The input is dtspan/timedivisions,
116         where dtspan is the entire timespan of the lightcurve.
117         so dtspan = time[-1] - time[0]
118
119         Returns the round value.
120         """
121
122         dex = math.log(darg, 10)
123         nex = int(dex)
124
125         darg = darg / math.pow(10, nex)
126
127         if darg >= 5 :
128             darg = 5.0
129         elif darg >= 2 :
130             darg = 2.0
131         else :
132             darg = 1.0
133
134         darg = darg * math.pow(10, nex)
135         return darg

```

```

136
137     def matrix_inv(self, inputMatrix) :
138         """The Matrix Inversion Method.
139         Arguments are :
140             inputMatrix : the input matrix_inv
141
142         Returns the inverted matrix.
143         """
144         # Lines 202 - 252 Fortran
145
146         ndim = 2
147         dsol = numpy.zeros(shape=(3, 3))
148
149         for i in range(0, 3) :
150             for j in range(0, 3) :
151                 dsol[i][j] = 0.0
152                 dsol[i][i] = 1.0
153
154         for i in range(0, 3) :
155             if inputMatrix[i][i] == 0.0 :
156                 if i == ndim :
157                     return
158                 for j in range(0, 3) :
159                     if inputMatrix[j][i] != 0.0 :
160                         for k in range(0, 3) :
161                             inputMatrix[i][k] = inputMatrix[i][k] + \
162                                     inputMatrix[j][k]
163                             dsol[i][j] = dsol[i][j] + dsol[j][k]
164
165                 dfac = inputMatrix[i][i]
166
167                 for j in range(0, 3) :
168                     inputMatrix[i][j] = inputMatrix[i][j] / dfac
169                     dsol[i][j] = dsol[i][j] / dfac

```

```

170
171         for j in range(0, 3) :
172             if j != i :
173                 dfac = inputMatrix[j][i]
174                 for k in range(0, 3) :
175                     inputMatrix[j][k] = inputMatrix[j][k] - \
176                                     (inputMatrix[i][k] * dfac)
177                     dsol[j][k] = dsol[j][k] - (dsol[i][k] * dfac)
178
179     # The unnecessary loop
180     #for i in range(0, 3) :
181         #for j in range(0, 3) :
182             #self.dmat[i][j] = dsol[i][j]
183
184     return dsol
185
186 def maketau(self, time, timedivisions):
187     """The maketau method.
188     Arguments are :
189         time = The array of time values
190         timedivisions = The value of timedivisions to create tau values
191
192         Returns an array of calculated tau values.
193     """
194     # The Maketau section
195     # Lines 90 - 122 Fortran
196
197     dtauhi = time[-1]
198     dtaulo = time[0]
199     # the java translation uses 1 for dtaulo,
200     # but that should be a bug.
201
202     dtspan = dtauhi - dtaulo
203     dtstep = self.roundtau(dtspan / timedivisions)

```

```

204
205         dtaulo = dtstep * int(dtaulo / dtstep)
206         dtauhi = dtstep * int((dtauhi / dtstep) + 0.5)
207
208         tau = []
209         dtau = dtaulo
210         while dtau <= dtauhi :
211             tau.append(dtau)
212             dtau = dtau + dtstep
213         return tau
214
215         ### End of Maketau
216
217     def makefreq(self, flo, fhi, df):
218         """The Makefreq section.
219         Arguments are :
220             flo = Low Frequency
221             fhi = High Frequency
222             df = Frequency Step
223         """
224         # Lines 149 - 181 Fortran
225         # Lines 356 - 370 Java
226         freq = []
227         freq.append(flo)
228         nfreq = int((fhi - flo) / df) + 1
229
230         # These lines seem skeptical!
231         for i in range(1,nfreq+1) :
232             freq.append(flo + ((i - 1) * df))
233
234         return freq
235
236         ### End of Makefreq
237

```

```

238     def wwt(self, time, magnitude, flo, fhi, df, dcon, timedivisions) :
239         """The WWZ Algorithm
240         Arguments are :
241             time = The time values as an array
242             magnitude = The magnitude values as an array
243             flo = The Low Frequency
244             fhi = The High Frequency
245             df = The Frequency Step
246             dcon = The C constant of WWZ Window
247             timedivisions = The TAU steps
248
249             Returns a NumPy Array
250         """
251
252         dave = numpy.mean(magnitude)
253         dvar = numpy.var(magnitude)
254
255         freq = self.makefreq(flo, fhi, df)
256         nfreq = len(freq)
257         dmat = numpy.zeros(shape=(3,3))
258
259         ### End of Initializing
260
261         tau = self.maketau(time, timedivisions)
262         ntau = len(tau)
263
264         ### WWT Stars Here
265
266         dvec = [0,0,0] # length is 3
267         dcoef = [0,0,0] # length is 3
268
269         itau = 0
270         ifreq = 0
271         idat = 0

```

```
272
273     domega = 0.0
274     dweight2 = 0.0
275     dz = 0.0
276     dweight = 0.0
277
278     dcc = 0.0
279     dcw = 0.0
280     dss = 0.0
281     dsw = 0.0
282     dxw = 0.0
283     dvarw = 0.0
284
285     dtau = 0.0
286
287     dpower = 0.0
288     dpowz = 0.0
289     damp = 0.0
290     dneff = 0.0
291     davew = 0.0
292
293     dfre = 0.0
294
295     n1 = 0
296     n2 = 0
297
298     dmz = 0.0
299     dmzfre = 0.0
300     dmzamp = 0.0
301     dmcon = 0.0
302     dmneff = 0.0
303
304     twopi = 2.0 * math.pi
305
```

```

306         ndim = 2
307         itau1 = 0 # --> 1 or 0 ??
308         itau2 = ntau # --> ???
309         ifreq1 = 1
310         ifreq2 = nfreq
311         nstart = 1
312
313         # Creating output arrays
314         output = numpy.empty((ntau*(nfreq-1), 6))
315         numdat=len(time)
316         index = 0
317
318         # Use for itau in range(itau1,itau2) for parallel
319         for itau in range(0, itau2) :
320             nstart = 1
321             dtau = tau[itau]
322
323             dmfre = 0.0
324             dmamp = 0.0
325             dmcon = 0.0
326             dmneff = 0.0
327             dmz = -1.0 # less than the smallest WWZ
328
329             for ifreq in range(ifreq1, ifreq2) :
330                 dfre = freq[ifreq]
331                 domega = dfre * twopi
332
333                 for i in range(0, ndim + 1) :
334                     dvec[i] = 0.0
335                     for j in range(0, ndim + 1) :
336                         dmat[i][j] = 0.0
337
338                 dweight2 = 0.0
339

```

```

340         for idat in range(nstart, numdat) :
341
342             dz = domega * (time[idat] - dtau)
343             dweight = math.exp(-1.0 * dcon * dz * dz)
344
345             if (dweight > 10**(-9)) :
346                 dcc = math.cos(dz)
347                 dcw = dweight * dcc
348                 dss = math.sin(dz)
349                 dsw = dweight * dss
350                 dmat[0][0] = dmat[0][0] + dweight
351                 dweight2 = dweight2 + (dweight**2)
352                 dmat[0][1] = dmat[0][1] + dcw
353                 dmat[0][2] = dmat[0][2] + dsw
354                 dmat[1][1] = dmat[1][1] + (dcw * dcc)
355                 dmat[1][2] = dmat[1][2] + (dcw * dss)
356                 dmat[2][2] = dmat[2][2] + (dsw * dss)
357
358                 dxw = dweight * magnitude[idat]
359                 dvec[0] = dvec[0] + dxw
360                 dvarw = dvarw + (dxw * magnitude[idat])
361                 dvec[1] = dvec[1] + (dcw * magnitude[idat])
362                 dvec[2] = dvec[2] + (dsw * magnitude[idat])
363
364             elif dz > 0.0 :
365                 break
366             else :
367                 nstart = idat + 1
368
369         dpower = 0.0
370         damp = 0.0
371
372         for n1 in range(0, ndim + 1) :
373             dcoef[n1] = 0.0

```

```

374
375         if (dweight2 > 0.0) :
376             dneff = (dmat[0][0] * dmat[0][0]) / dweight2
377         else :
378             dneff = 0.0
379
380         if (dneff > 3.0) :
381
382             for n1 in range(0, ndim + 1) :
383                 dvec[n1] = dvec[n1] / dmat[0][0]
384
385                 for n2 in range(1, ndim + 1) :
386                     dmat[n1][n2] = dmat[n1][n2] / \
387                         dmat[0][0]
388
389                 if (dmat[0][0] > 0.005) :
390                     dvarw = dvarw / dmat[0][0]
391                 else :
392                     dvarw = 0.0
393
394
395                 dmat[0][0] = 1.0
396                 davew = dvec[0]
397                 dvarw = dvarw - (davew ** 2)
398
399                 if (dvarw <= 0.0) :
400                     dvarw = 10**-12
401
402                 for n1 in range(1, ndim + 1) :
403                     for n2 in range(0, n1) :
404                         dmat[n1][n2] = dmat[n2][n1]
405
406
407                 dmat = self.matrix_inv(dmat)

```

```

408
409         for n1 in range(0, ndim + 1) :
410             for n2 in range(0, ndim + 1) :
411                 dcoef[n1] = dcoef[n1] + dmat[n1][n2] * \
412                     dvec[n2]
413
414                 dpower = dpower + (dcoef[n1] * dvec[n1])
415
416                 dpower = dpower - (davew ** 2)
417                 dpowz = (dneff - 3.0) * dpower / (dvarw - dpower) / 2.0
418                 dpower = (dneff - 1.0) * dpower / dvarw / 2.0
419                 damp = math.sqrt(dcoef[1] * dcoef[1] + \
420                     dcoef[2] * dcoef[2])
421
422         else :
423
424             dpowz = 0.0
425             dpower = 0.0
426             damp = 0.0
427
428             if (dneff < (10**(-9))) :
429                 dneff = 0.0
430
431             if (damp < (10**(-9))) :
432                 damp = 0.0
433
434             if (dpower < (10**(-9))) :
435                 dpower = 0.0
436
437             if (dpowz < (10**(-9))) :
438                 dpowz = 0.0
439
440             # Let's write everything out.
441             output[index] = [dtau, dfre, dpowz, damp, dcoef[0], dneff]

```

```

442
443         index = index + 1
444
445         if (dpowz > dmz) :
446             dmz = dpowz
447             dmfre = dfre
448             dmamp = damp
449             dmcon = dcoef[0]
450             dmneff = dneff
451
452     return output
453
454
455 def writefile(self, wwz_output, outputFile, no_headers, max_periods) :
456     """The write file method.
457     Arguments are :
458         wwz_output = The NumPy array of WWZ values to write
459         outputFile = The output file pointer, not the filename
460         no_headers = If true, will not write headers to the output
461         max_periods = If true, will create a file with period values
462             of maximum WWZ statistics
463     """
464
465     numpy.set_printoptions(precision=5)
466     numpy.set_printoptions(suppress=True)
467     numpy.set_printoptions(threshold='nan')
468
469     if no_headers :
470         numpy.savetxt(outputFile, wwz_output, delimiter="\t", \
471             fmt="%10.4f")
472     else :
473         numpy.savetxt(outputFile, wwz_output, delimiter="\t", \
474             fmt="%10.4f", comments="#", \
475             header="%9s %10s %10s %10s %10s %10s" % \

```

```

476         ("TAU", "FREQ", "WWZ", "AMP", "COEF", "NEFF"))
477
478     def writegnu(self, wwz_output, outputFile, no_headers, \
479                 max_periods, ntau) :
480         """The write file method, adapted to work with GnuPlot.
481         Arguments are :
482             wwz_output = The NumPy array of WWZ values to write
483             outputFile = The output file pointer, not the filename
484             no_headers = If true, will not write headers to the output
485             max_periods = If true, will create a file with period values
486                         of maximum WWZ statistics
487             ntau = The number of tau values, this is needed in order to
488                     split the wwz_output equally
489                     To calculate ntau, use the equation below:
490                         len(wwz_output) /
491                         int(((freq_high - freq_low) / freq_step) + 1)
492         """
493
494         numpy.set_printoptions(precision=5)
495         numpy.set_printoptions(suppress=True)
496         numpy.set_printoptions(threshold='nan')
497
498         splitArray = numpy.vsplit(wwz_output, ntau)
499
500         # check if the file is in append mode
501         # if not, reopen it
502         if outputFile.mode != 'a' :
503             outputFile.close()
504             outputFile = open(outputFile.name, "a")
505
506         # write headers if expected
507         if not no_headers :
508             header="%9s %10s %10s %10s %10s %10s" % \
509                 ("TAU", "FREQ", "WWZ", "AMP", "COEF", "NEFF")

```

```

510         outputFile.write("#" + header + "\n")
511
512         # split the array and add newlines in between tau values,
513         # write the output
514         for i in range(0,ntau) :
515             numpy.savetxt(outputFile, splitArray[i], delimiter="\t", \
516                           fmt="%10.4f")
517             if i != ntau-1 :
518                 outputFile.write("\n")
519
520 # The WWZPAR Class Begins
521
522 class WWZPAR(object) :
523     """This is for Parallel Processing only.
524     This works different than the WWZ class, it takes arguments directly.
525
526     It gets input as a filepointer, NOT as arrays!
527     Arguments :
528         fileName : the input filename, should be the lightcurve.
529         outputfileName : the output filename.
530         flo : the low frequency value. Float.
531         fhi : the high frequency value. Float.
532         df : the frequency step. Float.
533         dcon : the C Window constant. Float.
534         timedivisions : the The Divisions value, choose 50.0 if not sure,
535         that is the default used by Templeton.
536         max_periods : set True if you want the second output file.
537         It outputs the tau's with maximum period values
538         for easier estimation.
539         gnuplot_compatible : splits tau values by a blank line if set True,
540         so that pm3d of gnuplot easily maps the plot.
541     """
542
543     def __init__(self, fileName, outputfileName, flo, fhi, df, dcon, \

```

```

544         timedivisions, max_periods, gnuplot_compatible) :
545     """Initializing the object"""
546
547     self.inputfile = fileName
548
549     self.outputfilename1 = outputfileName
550     self.outputfilename2 = outputfileName.name + ".max_periods"
551
552     self.max_periods = max_periods
553     self.gnuplot_compatible = gnuplot_compatible
554
555     self.timedivisions = timedivisions
556     # This (50) is an assumption by Templeton.
557     # VStars leaves this optional
558     # but keeps the default value
559
560     self.fhi = fhi
561     self.flo = flo
562     self.df = df
563
564     # dcon is the Window Constant "c" in Foster's equations.
565     self.dcon = dcon
566
567     self.fileName = fileName
568
569     self.time = [] # Input time, first column in the file
570     self.magnitude = [] # Input magnitude, second column in the file
571
572     self.dave = 0.0 # average
573     self.dvar = 0.0 # variance
574
575     self.nfreq = int((self.fhi - self.flo) / self.df) + 1
576
577     self.freq = []

```

```

578
579         self.dmat = numpy.zeros(shape=(3, 3))
580
581
582     def readfile(self) :
583         """Read the input file"""
584
585         #read_file = open(self.inputfilename, "r")
586         for line in self.inputfile :
587             if line.strip()[0] != "%" and line.strip()[0] != "#" :
588                 line_time = float(line.split()[0])
589                 line_mag = float(line.split()[1])
590                 self.time.append(line_time)
591                 self.magnitude.append(line_mag)
592                 self.dave = self.dave + line_mag
593                 self.dvar = self.dvar + (line_mag ** 2)
594
595         self.inputfile.close()
596         if len(self.time) != len(self.magnitude) :
597             print "Number of Time and Magnitude input do not match. \
598                 Please check the input file."
599             raise SystemExit
600             # Just a routine check for parameter number equality.
601             # This should be cleaned up a bit.
602
603         # Calculating Header Values
604         self.numdat = len(self.time)
605         self.dave = self.dave / self.numdat
606         self.dvar = (self.dvar / self.numdat) - (self.dave ** 2)
607         self.dsig = math.sqrt((self.dvar * self.numdat) / (self.numdat - 1))
608
609
610     def roundtau(self, darg) :
611         """Rounds the tau's. from G. Foster's Code."""

```

```

612
613     dex = math.log(darg, 10)
614     nex = int(dex)
615
616     darg = darg / math.pow(10, nex)
617
618     if darg >= 5 :
619         darg = 5.0
620     elif darg >= 2 :
621         darg = 2.0
622     else :
623         darg = 1.0
624
625     darg = darg * math.pow(10, nex)
626     return darg
627
628
629 def matrix_inv(self,input_matrix) :
630     """The Matrix Inversion Function"""
631     # Lines 202 - 252 Fortran
632
633     ndim = 2
634     dsol = numpy.zeros(shape=(3, 3))
635
636     for i in range(0, 3) :
637         for j in range(0, 3) :
638             dsol[i][j] = 0.0
639             dsol[i][i] = 1.0
640
641     for i in range(0, 3) :
642         if input_matrix[i][i] == 0.0 :
643             if i == ndim :
644                 return
645             for j in range(0, 3) :

```

```

646         if input_matrix[j][i] != 0.0 :
647             for k in range(0, 3) :
648                 input_matrix[i][k] = input_matrix[i][k] + \
649                     input_matrix[j][k]
650                 dsol[i][j] = dsol[i][j] + dsol[j][k]
651
652     dfac = input_matrix[i][i]
653
654     for j in range(0, 3) :
655         input_matrix[i][j] = input_matrix[i][j] / dfac
656         dsol[i][j] = dsol[i][j] / dfac
657
658     for j in range(0, 3) :
659         if j != i :
660             dfac = input_matrix[j][i]
661             for k in range(0, 3) :
662                 input_matrix[j][k] = input_matrix[j][k] - \
663                     (input_matrix[i][k] * dfac)
664                 dsol[j][k] = dsol[j][k] - (dsol[i][k] * dfac)
665
666     return dsol
667
668
669 def maketau(self) :
670     """The Maketau section"""
671     # Lines 90 - 122 Fortran
672
673     dtaulo = self.time[0] # the java translation uses 1 for this,
674     # but that should be a bug.
675     dtauhi = self.time[-1]
676
677     dtspan = dtauhi - dtaulo
678     dtstep = self.roundtau(dtspan / self.timedivisions)
679

```

```

680         dtaulo = dtstep * int(dtaulo / dtstep)
681         dtauhi = dtstep * int((dtauhi / dtstep) + 0.5)
682
683         self.tau = []
684
685         dtau = dtaulo
686         while dtau <= dtauhi :
687             #print dtau
688             self.tau.append(dtau)
689             dtau = dtau + dtstep
690
691
692         self.ntau = len(self.tau)
693
694     def makefreq(self) :
695         """The Makefreq section"""
696         # Lines 149 - 181 Fortran
697         # Lines 356 - 370 Java
698
699         self.freq.append(self.flo)
700
701         # These lines seem skeptical!
702         for i in range(1,self.nfreq+1) :
703             self.freq.append(self.flo + ((i - 1) * self.df))
704
705     def wwt(self, output1_par, itaul_par, itau2_par) :
706         """The WWZ Algorithm in Parallel Mode"""
707
708         output1 = open(output1_par, "w")
709         max_periods = self.max_periods
710         gnuplot_compatible = self.gnuplot_compatible
711
712         if max_periods == True :
713             output2_par = output1_par + ".max_periods.par"

```

```

714         output2 = open(output2_par, "w")
715
716         dvec = [0,0,0] # length is 3
717         dcoef = [0,0,0] # length is 3
718
719         itau = 0
720         ifreq = 0
721         idat = 0
722
723         domega = 0.0
724         dweight2 = 0.0
725         dz = 0.0
726         dweight = 0.0
727
728         dcc = 0.0
729         dcw = 0.0
730         dss = 0.0
731         dsw = 0.0
732         dxw = 0.0
733         dvarw = 0.0
734
735         dtau = 0.0
736
737         dpower = 0.0
738         dpowz = 0.0
739         damp = 0.0
740         dneff = 0.0
741         davew = 0.0
742
743         dfre = 0.0
744
745         n1 = 0
746         n2 = 0
747

```

```

748         dmz = 0.0
749         dmzfre = 0.0
750         dmzamp = 0.0
751         dmcon = 0.0
752         dmneff = 0.0
753
754         twopi = 2.0 * math.pi
755
756         ndim = 2
757         itaul = 0 # ----> 1 or 0 ??
758         itau2 = self.ntau # -----> ???
759         ifreq1 = 1
760         ifreq2 = self.nfreq
761         nstart = 1
762
763         dmat_par = numpy.zeros(shape=(3, 3))
764
765         for itau in range(itaul_par, itau2_par) :
766             nstart = 1
767             dtau = self.tau[itau]
768
769             dmfre = 0.0
770             dmamp = 0.0
771             dmcon = 0.0
772             dmneff = 0.0
773             dmz = -1.0 # less than the smallest WWZ
774
775             for ifreq in range(ifreq1, ifreq2 + 1) :
776                 dfre = self.freq[ifreq]
777                 domega = dfre * twopi
778
779                 for i in range(0, ndim + 1) :
780                     dvec[i] = 0.0
781                 for j in range(0, ndim + 1) :

```

```

782         dmat_par[i][j] = 0.0
783
784     dweight2 = 0.0
785
786     for idat in range(nstart, self.numdat) :
787
788         dz = domega * (self.time[idat] - dtau)
789         dweight = math.exp(-1.0 * self.dcon * dz * dz)
790
791         if (dweight > 10**(-9)) :
792             dcc = math.cos(dz)
793             dcw = dweight * dcc
794             dss = math.sin(dz)
795             dsw = dweight * dss
796             dmat_par[0][0] = dmat_par[0][0] + dweight
797             dweight2 = dweight2 + (dweight**2)
798             dmat_par[0][1] = dmat_par[0][1] + dcw
799             dmat_par[0][2] = dmat_par[0][2] + dsw
800             dmat_par[1][1] = dmat_par[1][1] + (dcw * dcc)
801             dmat_par[1][2] = dmat_par[1][2] + (dcw * dss)
802             dmat_par[2][2] = dmat_par[2][2] + (dsw * dss)
803
804             dxw = dweight * self.magnitude[idat]
805             dvec[0] = dvec[0] + dxw
806             dvarw = dvarw + (dxw * self.magnitude[idat])
807             dvec[1] = dvec[1] + (dcw * self.magnitude[idat])
808             dvec[2] = dvec[2] + (dsw * self.magnitude[idat])
809
810         elif dz > 0.0 :
811             break
812         else :
813             nstart = idat + 1
814
815     dpower = 0.0

```

```

816         damp = 0.0
817
818     for n1 in range(0, ndim + 1) :
819         dcoef[n1] = 0.0
820
821     if (dweight2 > 0.0) :
822         dneff = (dmat_par[0][0] * dmat_par[0][0]) / dweight2
823     else :
824         dneff = 0.0
825
826     if (dneff > 3.0) :
827
828         for n1 in range(0, ndim + 1) :
829             dvec[n1] = dvec[n1] / dmat_par[0][0]
830
831         for n2 in range(1, ndim + 1) :
832             dmat_par[n1][n2] = dmat_par[n1][n2] / \
833                 dmat_par[0][0]
834
835         if (dmat_par[0][0] > 0.0) :
836             dvarw = dvarw / dmat_par[0][0]
837         else :
838             dvarw = 0.0
839
840         dmat_par[0][0] = 1.0
841         davew = dvec[0]
842         dvarw = dvarw - (davew ** 2)
843
844         if (dvarw <= 0.0) :
845             dvarw = 10**-12
846
847         for n1 in range(1, ndim + 1) :
848             for n2 in range(0, n1) :
849                 dmat_par[n1][n2] = dmat_par[n2][n1]

```

```

850
851
852         dmat_par = self.matrix_inv(dmat_par)
853
854         for n1 in range(0, ndim + 1) :
855             for n2 in range(0, ndim + 1) :
856                 dcoef[n1] = dcoef[n1] + \
857                     dmat_par[n1][n2] * dvec[n2]
858
859                 dpower = dpower + (dcoef[n1] * dvec[n1])
860
861                 dpower = dpower - (davew ** 2)
862                 dpowz = (dneff - 3.0) * dpower / (dvarw - dpower) / 2.0
863                 dpower = (dneff - 1.0) * dpower / dvarw / 2.0
864                 damp = math.sqrt(dcoef[1] * dcoef[1] + \
865                                 dcoef[2] * dcoef[2])
866
867         else :
868
869             dpowz = 0.0
870             dpower = 0.0
871             damp = 0.0
872
873             if (dneff < (10**(-9))) :
874                 dneff = 0.0
875
876             if (damp < (10**(-9))) :
877                 damp = 0.0
878
879             if (dpower < (10**(-9))) :
880                 dpower = 0.0
881
882             if (dpowz < (10**(-9))) :
883                 dpowz = 0.0

```

```

884
885         # Let's write everything out.
886
887         output1.write("%s \t %s \t %s \t %s \t %s \t %s\n" % \
888                       (str(dtau),str(dfre),str(dpowz),str(damp), \
889                        str(dcoef[0]),str(dneff)))
890
891         if (dpowz > dmz) :
892             dmz = dpowz
893             dmfre = dfre
894             dmamp = damp
895             dmcon = dcoef[0]
896             dmneff = dneff
897
898         #
899         if max_periods == True :
900             # writes the max_periods output if specified
901             output2.write("%f \t %f \t %f \t %f \t %f \t %f\n" % \
902                           (dtau, dmfre, dmz,  dmamp, dmcon, dmneff))
903
904         if gnuplot_compatible == True :
905             # added so that gnuplot reads out of the box
906             output1.write("\n")
907
908
909 # If the script runs as a standalone, below is triggered
910
911 if __name__ == '__main__' :
912
913     # Parsing the arguments
914
915     description = """
916     A Weighted Wavelet Z-Transformation Application for Python.
917     Translated by M. Emre Aydin - emre.m.aydin@gmail.com

```

918 `http://about.me/emre.aydin`
919 Available at `http://github.com/eaydin`
920
921 *Input arguments can be read from a file. The file descriptor*
922 *prefix is '@'.*
923 *In order to read argument from a file named args.txt,*
924 *the argument @args.txt should be passed.*
925 *An example for args.txt :*
926
927 `-f=myinputfile.txt`
928 `-o=theoutputfile.output`
929 `-m`
930 `--freq-step=0.001`
931 `-l=0.001`
932 `-hi=0.01`
933 `-c=0.001`
934 `-p=0`
935
936 *You can pass arguments from file and commandline at the same time.*
937 *If two same arguments passed by this method, the latter will be*
938 *used. So if you want to override some arguments in a an argument*
939 *file, specify the file first.*
940 *An example usage for our earlier @args.txt is as :*
941
942 `python wwz.py @args.txt -c=0.0125`
943
944 *The above command will use the settings in args.txt but will*
945 *use c=0.0125 instead of c=0.001*
946
947 *Comments and blank lines are NOT allowed in argument files.*
948
949 *Import this script via Python to use it as a module, rather than*
950 *a standalone script. (import wwz)*
951

```

952     """
953
954     parser = argparse.ArgumentParser(prog='wwz.py', \
955                                     formatter_class=argparse.RawDescriptionHelpFormatter,\
956                                     fromfile_prefix_chars="@", description=description)
957
958     parser.add_argument("-f", "--file", type=argparse.FileType("r"),\
959                         default=sys.stdin, required=True,\
960                         help="the Input File, Raw Lightcurve")
961     parser.add_argument("-o", "--output", type=argparse.FileType('w'),\
962                         default=sys.stdout, required=True,\
963                         help="the Output File Name")
964     parser.add_argument("-l", "--freq-low", type=float, required=True,\
965                         help="the Low Frequency Value")
966     parser.add_argument("-hi", "--freq-high", type=float, required=True,\
967                         help="the High Frequency Value")
968     parser.add_argument("-d", "--freq-step", type=float, required=True,\
969                         help="the dF value, incremental step for Frequency")
970     parser.add_argument("-c", "--dcon", type=float, required=True,\
971                         help="the C constant for the Window Function")
972     parser.add_argument("-g", "--gnuplot-compatible", action="store_true",\
973                         default=False, help="the Output file is GNUPlot \
974                         compatible, which means the tau's will be grouped \
975                         so that pm3d can easily map. Default value is \
976                         'False'.")
977     parser.add_argument("-m", "--max-periods", action="store_true", \
978                         default=False, help="Creates a secondary \
979                         output with the maximum Periods for each single \
980                         tau. This can be drawn in 2D. The output filename \
981                         is derived from the -o option, added 'max_periods'. \
982                         Default value is 'False'.")
983     parser.add_argument("-t", "--time-divisions", type=float, default=50.0, \
984                         help="The Time Divisions value. Templeton assumes \
985                         this as 50. VStars from AAVSO leaves this optional \

```

```

986         contrary to Templeton, yet it's default value is \
987         also 50.")
988     parser.add_argument("--time", action="store_true", default=False, \
989         help="Calculate the time of operation in seconds \
990         and print to standard output.")
991     parser.add_argument("--no-headers", action="store_true", default=False, \
992         help="Doesn't print headers to output files if set. \
993         Default is 'False'.")
994     parser.add_argument("-p", "--parallel", help="Created threads to speed \
995         up the process. Default value is '1', which means \
996         single thread. '0' means number of detected CPUs,\
997         can be overridden.", type=int, default=1)
998
999     args = parser.parse_args()
1000
1001     # Check if user asks for time calculation
1002     if args.time :
1003         starttime = datetime.now()
1004
1005     # Get the process number
1006     cpu=args.parallel
1007
1008     if cpu != 1 :
1009         try :
1010             import multiprocessing
1011         except (ImportError, NotImplementedError) :
1012             print "Multiprocessing not available, using single CPU thread."
1013             cpu = 1
1014
1015     if cpu == 0 :
1016         # detect CPU cores
1017         try :
1018             import multiprocessing
1019             cpu = multiprocessing.cpu_count()

```

```

1020     except (ImportError, NotImplementedError) :
1021         print "Multiprocessing not available, using single CPU thread."
1022         cpu = 1
1023
1024     # Multiprocessing begins
1025     if cpu != 1 :
1026
1027         s=WWZPAR(args.file, args.output, args.freq_low, args.freq_high, \
1028                 args.freq_step, args.dcon, args.time_divisions, \
1029                 args.max_periods, args.gnuplot_compatible)
1030
1031         s.readfile()
1032         s.maketau()
1033         s.makefreq()
1034
1035         # the list of output filenames
1036         output_par_names = []
1037         for i in range(1,cpu+1) :
1038             output_par_names.append('wwz.par.proc.%i' % i)
1039
1040         ntau_par1 = s.ntau/cpu
1041
1042         # the last ntau_par (turns out this is not necessary)
1043         ntau_par2 = s.ntau - (ntau_par1 * (cpu-1) )
1044
1045         thread_list = []
1046
1047         for i in range(1,cpu+1) :
1048             if i != cpu :
1049                 t = multiprocessing.Process(target=s.wwt, \
1050                 args=(output_par_names[i-1],ntau_par1*(i-1),ntau_par1*i))
1051             else :
1052                 t = multiprocessing.Process(target=s.wwt, \
1053                 args=(output_par_names[i-1],ntau_par1*(i-1),s.ntau))

```

```

1054         thread_list.append(t)
1055
1056     for thread in thread_list :
1057         thread.start()
1058     for thread in thread_list :
1059         thread.join()
1060
1061     # join the outputs and delete remaining files
1062
1063     if args.max_periods :
1064         output_max_periods = open(args.output.name + ".max_periods", "w")
1065
1066     # Write the headers if asked
1067     if not args.no_headers :
1068         args.output.write("#%9s %10s %10s %10s %10s %10s\n" % \
1069             ("TAU", "FREQ", "WWZ", "AMP", "COEF", "NEFF"))
1070         if args.max_periods :
1071             output_max_periods.write("#%9s %10s %10s %10s %10s %10s\n" % \
1072                 ("TAU", "FREQ", "WWZ", "AMP", "COEF", "NEFF"))
1073
1074     for i in range(1,cpu+1) :
1075         read_par = open(output_par_names[i-1], "r")
1076         for line in read_par :
1077             args.output.write(line)
1078         read_par.close()
1079         os.remove(output_par_names[i-1])
1080
1081     if args.max_periods :
1082
1083         read_par_max = open(output_par_names[i-1] + \
1084             ".max_periods.par", "r")
1085         for line in read_par_max :
1086             output_max_periods.write(line)
1087         read_par_max.close()

```

```

1088             os.remove(output_par_names[i-1] + ".max_periods.par")
1089
1090
1091
1092     # for single threading computing
1093     if cpu == 1 :
1094
1095         # Run the main class and its subroutines
1096
1097         s=WWZ()
1098
1099         time_data, magnitude_data = s.readfile(args.file)
1100         wwz_output = s.wwt(time_data, magnitude_data, args.freq_low, \
1101                             args.freq_high, args.freq_step, args.dcon, \
1102                             args.time_divisions)
1103         if args.gnuplot_compatible :
1104             send_ntau = len(wwz_output) / \
1105                 int(((args.freq_high - args.freq_low) / args.freq_step) + 1)
1106
1107             s.writegnu(wwz_output, args.output, args.no_headers, \
1108                         args.max_periods, send_ntau)
1109         else :
1110             s.writefile(wwz_output, args.output, \
1111                         args.no_headers, args.max_periods)
1112
1113
1114     args.output.close()
1115     if args.max_periods == True and cpu != 1 :
1116         output_max_periods.close()
1117
1118     # Print calculated time
1119     if args.time :
1120         endtime = datetime.now()
1121         print "Run Time : %s seconds" % str((endtime - starttime).seconds)

```

EK 2 ls.py Kaynak Kodu

```
1 #!/usr/bin/env python
2 # -*- coding: utf-8 -*-
3 # ls.py
4 # 16 May 22:31
5 # Copyright 2014 M. Emre Aydin <emre.m.aydin@gmail.com>
6
7 """
8     A Simple Lomb-Scargle (LS) Code to Check the Research for the
9     Weighted Wavelet Z-Transform Translation of Python.
10    The script uses the LS method in the SciPy Package.
11    Available at http://github.com/eaydin
12 """
13
14 # Checking the necessary modules and importing them if exists
15 import sys
16 try :
17     import numpy as np
18 except :
19     print "Please Install Numpy"
20     raise SystemExit
21
22 try :
23     import scipy as sp
24 except :
25     print "Please Install SciPy"
26     raise SystemExit
27
28 try :
29     import argparse
30 except :
31     "Argparse Not Available?! You sure this is Python 2.7?"
32     raise SystemExit
33
34 from scipy.signal import spectral
35
36 class LombScargle(object) :
37     """The Main class object.
38     This is the main class object to create a Lomb-Scargle periodogram.
39     Usage :
40     LombScargle(timeData, magnitudeData, freq_low, freq_high, \
```

```

41         freq_num)
42
43     Arguments:
44     timeData : float array of time values in JD
45     magnitudeData : float array of magnitude values
46     freq_low : the low frequency value
47     freq_high : the high frequency value
48     freq_num : number of total frequencies to evaluate
49
50     Available Attributes :
51     'power' : returns the LS-Power only.
52     'freqs' : returns the frequencies calculated.
53     'periodogram' : returns the LS Periodogram as [freqs, power].
54     '
55     """
56
57     def __init__(self, timeData, magnitudeData, freq_low, \
58                 freq_high, freq_num) :
59         """Initializing the LombScargle Object"""
60         self.freqs = np.linspace(freq_low, freq_high, freq_num)
61
62         # Convert arrays to Numpy Arrays for lombscargle to understand
63         timeArray = np.asarray(timeData)
64         magnitudeArray = np.asarray(magnitudeData)
65
66         # Calculating the periodogram and return it
67         self.power = spectral.lombscargle(timeArray, magnitudeArray, \
68                                         self.freqs)
69         self.periodogram = np.column_stack([self.freqs, self.power])
70
71         # The Frequency and LS-Power of the Maximum LS-Power
72         self.maxLS = [self.freqs[self.power.argmax(axis=0)], \
73                     np.amax(self.power)]
74
75
76 if __name__ == '__main__' :
77     """The main function run when standalone"""
78
79     description = """
80     The Lomb-Scargle Periodogram Program.
81     This script is written for research purposes to accompany the
82     Weighted Wavelet Z-Transform implementation.
83     M. Emre Aydin - emre.m.aydin@gmail.com

```

```

84     http://about.me/emre.aydin
85     Available at http://github.com/eaydin
86
87     Import this script via Python to use it as a module, rather than
88     a standalone script. (import ls)
89     """
90
91     # Parsing arguments
92
93     parser = argparse.ArgumentParser(prog="ls.py", \
94                                     formatter_class=argparse.RawDescriptionHelpFormatter, \
95                                     description=description)
96     parser.add_argument("-f", "--file", type=argparse.FileType("r"), \
97                         default=sys.stdin, required=True, \
98                         help="the Input file, Raw Lightcurve")
99     parser.add_argument("-o", "--output", type=argparse.FileType("w"), \
100                        help="the Output file name")
101     parser.add_argument("-s", "--stdout", action="store_true", \
102                        default=False, help="Prints the results to \
103                        the standard output. Automatically set to 'True' \
104                        if '-o' option is not defined, otherwise 'False'.")
105     parser.add_argument("-l", "--freq-low", type=float, required=True, \
106                        help="the Low Frequency Value")
107     parser.add_argument("-hi", "--freq-high", type=float, required=True, \
108                        help="the High Frequency Value")
109     parser.add_argument("-n", "--freq-num", type=int, required=False, \
110                        default=1000, help="the Number of Frequencies to \
111                        calculate. Input type is integer. Default value \
112                        is 1000.")
113     parser.add_argument("--no-headers", action="store_true", \
114                        default=False, help="Doesn't print headers to output \
115                        if set. Default is 'False'.")
116     args = parser.parse_args()
117
118     # Check if not output file is specified, then set stdout to 'True'
119     if args.output is None :
120         args.stdout = True
121
122     magnitudeData = []
123     timeData = []
124
125     # Read the lightcurve data
126     # Dismiss lines beginning with '%' and '#'

```

```

127     for line in args.file :
128         if line.strip()[0] != "%" and line.strip()[0] != "#" :
129             timeData.append(float(line.split()[0].strip()))
130             magnitudeData.append(float(line.split()[1].strip()))
131
132     s = LombScargle(timeData, magnitudeData, args.freq_low, \
133                    args.freq_high, args.freq_num)
134
135     # Setting the float precision of Numpy to 5
136     np.set_printoptions(precision=5)
137     # Suppress the printing options for Numpy
138     np.set_printoptions(suppress=True)
139     # Print the full Numpy Array
140     np.set_printoptions(threshold='nan')
141
142     if args.stdout :
143         if not args.no_headers :
144             print "# Frequency \t LS-Power"
145         for i in s.periodogram :
146             print "%.5f %.5f" % (i[0], i[1])
147
148     if args.output is not None :
149         if not args.no_headers :
150             np.savetxt(args.output, s.periodogram, delimiter="\t", \
151                       fmt="%10.5f")
152         else :
153             np.savetxt(args.output, s.periodogram, fmt="%10.5f", \
154                       delimiter="\t", comments="#", \
155                       header=" Frequency \t LS-Power")

```

EK. 3 Örnek Gnuplot Kodu

```
1 set term png size 1000,1000
2 set output "cygx2-use.png"
3 set encoding utf8
4
5 set pointsize 0.25
6 set multiplot
7 set title "CYG X-2" font "Times New Roman, 30"
8 #TOP GRAPH
9 set tmargin at screen 0.90
10 set lmargin at screen 0.30
11 set bmargin at screen 0.71
12 set rmargin at screen 0.90
13 set xtics
14 set format x ""
15 unset key
16 set ylabel "Sayım" offset 1
17 set ytics 0,20,80
18 plot 'cygx2.1day' with points pt 7 lt -1
19
20 #LEFT GRAPH
21 set grid ytics
22 set title "c=0.005" font "Times New Roman, 25" offset -5
23 set tmargin at screen 0.70
24 set lmargin at screen 0.10
25 set rmargin at screen 0.30
26 set bmargin at screen 0.101 # somethings not right?!
27 set ylabel "Frekans ve Dönem" offset 1.5
28 set yrange [0.035:0.01]
29 set ytics auto
30 set xtics rotate by 60 right
31 set format x "%5.0g"
32 set x2label "L-S Gücü"
33 plot 'cygx2.ls' u 2:1 with points pt 9 lt 1
34
35 #MIDDLE GRAPH
36 unset grid
37 #unset ytics
38 unset title
39 set tmargin at screen 0.70
40 set lmargin at screen 0.30
41 set rmargin at screen 0.90
42 set bmargin at screen 0.10
43 unset ylabel
```

```
44 set xtics 51000,1000,56000 norotate
45 set format x "%5.0f"
46 set y2tics
47 set yrange[0.035:0.01]
48 set ytics ("33" 0.03, "50" 0.02, "66" 0.015, "40" 0.025)
49 set xlabel "MJD"
50 set pm3d map
51 #set palette gray negative
52 #unset colorbox
53 splot 'cygx2.wwz' u 1:2:3
```

EK 4 CentOS 6.x Üzerine Python 2.7.6 Yüklenmesi

Red Hat³⁰ tabanlı olan CentOS işletim sistemleri, *yum* paket yöneticisini kullanmaktadırlar. Yum paket yöneticisinin kodları, Python 2.6'ya uygun şekilde geliştirildiğinden, yapılan güncellemelerde sistemin kullandığı Python versiyonu 2.6'nın üzerine çıkmaz.

Her ne kadar masaüstü Linux dağıtımları arasında CentOS kendine çok sağlam bir yer edinemesse de, sunucu yönetiminde ilk sıralarda tercih edilmektedir. Bu durum, masaüstü kullanımına alışmış kişilerin bazı alışkanlıklarını doğrudan sunucu yönetimine taşıyamamasına neden olur, Python versiyonu örneğinde olduğu gibi.

W3Tech'in 2010 yılında yayınladığı verilere³¹ göre sunucu içi Linux dağıtımı kullanımında Debian³² ile CentOS birinciliği taşımakta, peşinden Red Hat dağıtımı gelmektedir (bkz. şekil ek 5.1). Debian temelli dağıtımlar, *aptitude* paket yöneticisini kullandığından ve bu paket yöneticisi yum gibi Python 2.6'ya temellendirilmediğinden, sistem Python versiyonunda güncelleme sıkıntısı yaşamamaktadır. Ancak maalesef Red Hat ve CentOS için durum böyle değildir.

Sunucu yönetiminde neredeyse endüstri standardı haline gelen CentOS kullanımında bu problemin üstesinden gelmek için, sistemin kullandığı Python versiyonu ve kütüphanelerine dokunmadan, ikincil (veya gerekirse daha fazla) Python versiyonları yüklenmekte, programlar çalıştırılacağı zaman ihtiyaç duyulan versiyon çağırılmaktadır.

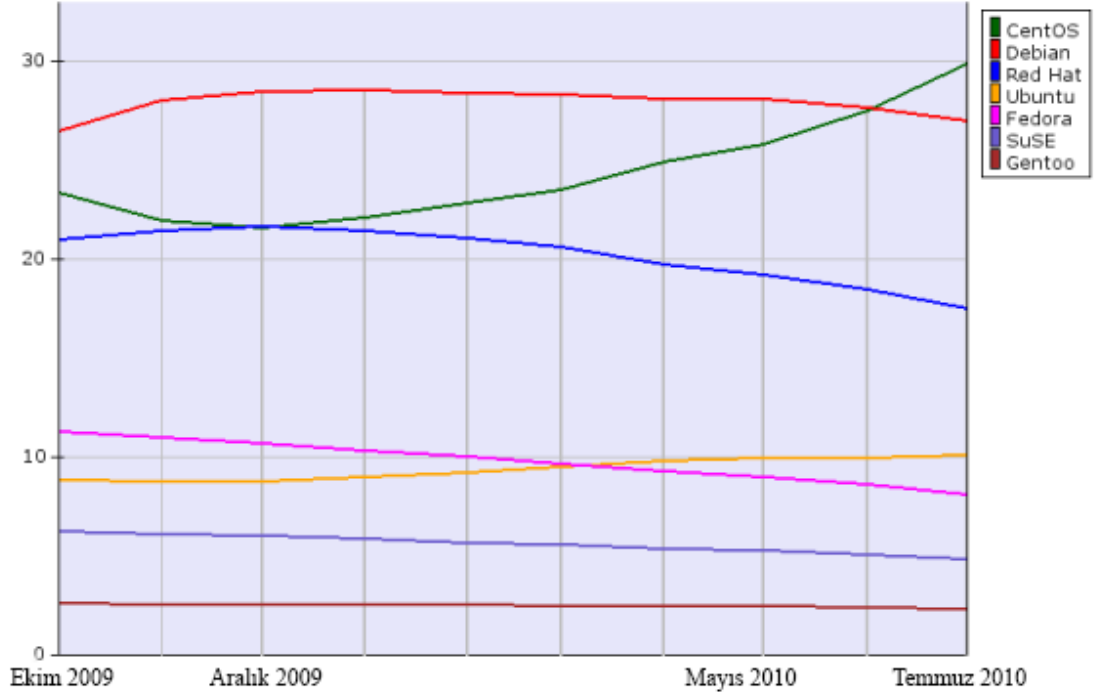
Bu bölümde anlatılan işlemler, Python 2.7.6'nın CentOS 6.x üzerine yüklenmesi için geçerlidir. Aynı işlemler Python 3.x için de benzer şekilde uygulanabilir.

³⁰ <http://www.redhat.com>

³¹ http://w3techs.com/blog/entry/highlights_of_web_technology_surveys_july_2010

³² <http://www.debian.org>

İşlemlerin tamamı *root* kullanıcısı olarak veya *sudo* ile root yetkisi altında uygulanmalıdır.



Şekil Ek 5.1 2010 yılı, sunucu için GNU/Linux dağıtım kullanım dağılımı. Düşey eksen kullanım yüzdesini ifade ederken, yatay eksen tarihleri belirtmektedir.

CentOS'ta Python, ön tanımlı olarak */usr/bin/* dizini altında yer almaktadır. Herhangi bir karışıklığa sebep olmamak için, yeni yüklenecek Python */usr/local/bin/* dizini altında yer alacaktır.

Eğer kullanıcının özel bir sebebi yoksa, Python 3.2 ve daha eski versiyonların yüklenmesinde UTF-32 desteğini aktif hale getirmekte fayda vardır. Bu durum bellek kullanımını bir miktar artırsa da, geriye dönük uyumluluğu artıracaktır. Python 3.3'te Unicode desteği tamamen yeniden yazılmıştır ve bu versiyondan itibaren *string* tipi değişkenler en verimli encoding tipini kullanacak şekilde otomatik olarak biçimlenmektedir. Dolayısıyla Python 3.3 ve sonraki versiyonlarda böyle bir zorunluluk bulunmamaktadır. Bu açıklamaya Python 2.7 yüklendiği varsayımıyla devam edileceğinden dolayı, Python UTF-32 desteğiyle derlenecektir.

Python 2.7 yüklenmesinde UTF-32 desteğinin sağlanması için “*configure*” komutuna “*--enable-unicode=ucs4*” parametresini girdi olarak tanımlanmaktadır. Python 3.2 için bunun yerine “*--with-wide-unicode*” kullanılmaktadır.

Python’un paylaşılan kütüphane (*shared library*) olarak yüklenmesi, kullanırken çeşitli problemlerle karşılaşılmasına engel olacaktır. Bütün modern Linux dağıtımları Python’u paylaşılan bir kütüphane olarak derleyerek sunmaktadır. Bu sayede *mod_wsgi*³³, *Blender*³⁴, *PyRAF*³⁵ gibi yazılımlar çalışırken Python’dan faydalanabilmektedirler. Python’u paylaşılan bir kütüphane olarak derleyebilmek için, sisteme bu kütüphaneleri nerede bulacağını belirtmek gerekmektedir. Bunu yapmanın iki yolu bulunmaktadır:

1. Derleme işleminden önce kullanılan “*configure*” komutuna girdi olarak şu parametreyi sunmak: “*LDFLAGS=-Wl,-rpath /usr/local/lib*”
2. Herhangi bir metin düzenleyicisi ile */etc/ld.so.conf* dosyasını açıp sonuna */usr/local/lib* dizinin yolunu eklemek. Bunu yaptıktan sonra */sbin/ldconfig* çalıştırılmalı ki dinamik bağlantıların bu değişiklikten haberdar olması sağlansın.

Bu hatırlatmaların ardından, sisteme gerekli dosya ve paketleri yüklenebilir. Öncelikle *yum* paket yöneticisini kullanarak bir *grup yükleme* yapılmalı, ardından çeşitli paketleri ayrı ayrı yüklemeyi belirtmeliyiz.

```
# yum group groupinstall "Development tools"
# yum install zlib-devel bzip2-devel openssl-devel ncurses-devel sqlite-devel readline-devel tk-devel
gdbm-devel db4-devel libpcap-devel xz-devel wget
```

Yukarıdaki komutlar uygulandıktan sonra, Python 2.7’nin indirilip derlenmesi işlemine geçilebilir.

```
# wget http://python.org/ftp/python/2.7.6/Python-2.7.6.tar.xz
```

³³ <https://code.google.com/p/modwsgi>

³⁴ <http://www.blender.org>

³⁵ http://www.stsci.edu/institute/software_hardware/pyraf

```
# tar xf Python-2.7.6.tar.xz
# cd Python-2.7.6
# ./configure --prefix=/usr/local --enable-unicode=ucs4 --enable-shared LDFLAGS="-Wl,-rpath
/usr/local/lib"
# make && make altinstall
```

Daha önce GNU/Linux sistemlerde kaynak kodu derlemiş kişilerin dikkatini çekecek bir nokta, son komut olarak “*make install*” yerine “*make altinstall*” kullanılmış olması. Eğer “*make install*” kullanılmış olsaydı, derleme işleminden sonra sistemde iki adet “*python*” isimli derlenmiş dosya bulunacaktı, bunlardan birisi eski (2.6.6), diğeri yeni (2.7.6) olanı. Bu tip durumlar tespit edilmesi çok zor problemlere sebep olabilmektedir. Bunun önüne geçmek için, “*python*” isimli tek bir dosya olması, bunun da sistemin kullandığı, 2.6.6 versiyonlu Python olması sağlanmış oldu.

Yukarıdaki komutlar uygulandıktan sonra, Python 2.7.6 yorumlayıcı “*/usr/local/bin/python2.7*” yolunda yer alacaktır. Sistem Python versiyonu hala 2.6.6 versiyonunda kalacaktır ve yolları “*/usr/bin/python*”, “*/usr/bin/python2*”, “*/usr/bin/python2.6*” şeklinde olacaktır.

Python için sonradan yüklenen modüller (NumPy, Scipy gibi) yüklendikleri versiyona göre konumlandırıldıklarından, daha önce yüklenenler yeni Python’da geçerli olmayacaktır. Dolayısıyla yum paket yöneticisiyle NumPy’nin yüklenmiş olması, bunu sadece Python 2.6.6’da kullanılabılır kılacaktır, Python 2.7.6 için yeniden yüklenmesi gerekecektir.

Python modüllerinin kolayca yüklenmesi için iki yaygın araç bulunmaktadır ve bunlar da tıpkı alternatif Python gibi, versiyonlarına göre yükleyebilir.

Setuptools yardımıyla, “*easy_install*” komutunu kullanarak *Python Package Index*’ten sisteme rahatlıkla yeni paketler kurulabilir. Ayrıca *pip* kullanımı, bu paketlerin yönetimi için bir takım kolaylıklar sunacaktır. Yeni Python yorumlayıcısına bu özellikleri ekleyebilmek için aşağıdaki komutlar uygulanabilir:

```
# wget https://bitbucket.org/pypa/setuptools/raw/bootstrap/ez_setup.py
# python2.7 ez_setup.py
# easy_install-2.7 pip
```

Bu işlemlerin ardından, artık gerekli paket *pip*³⁶ kullanılarak yüklenebilir. Örneğin Python 2.7.6 üzerine NumPy yüklemek, paketi güncellemek veya kaldırmak için aşağıdaki komutlar yeterli olacaktır.

```
# pip2.7 install numpy
# pip2.7 install --upgrade numpy
# pip2.7 uninstall numpy
```

Bu işlemlerin ardından, yeni paketler “*/usr/local/lib/python2.7/site-packages/numpy*” dizininde yer alacaktır.

³⁶ <https://pypi.python.org/pypi/pip>

ÖZGEÇMİŞ

Adı Soyadı : Mustafa Emre AYDIN

Doğum yeri : Edmonton / Kanada

Doğum Tarihi : 1984

Medeni Hali : Bekar

Yabancı Dili : İngilizce

Eğitim Durumu (Kurum ve Yıl)

Lise : ÇEAŞ Seyhan Anadolu Lisesi. (2002)

Lisans : Ankara Üniversitesi Fen Fakültesi Astronomi ve Uzay Bilimleri
Bölümü. (2009)

Yüksek Lisans : Ankara Üniversitesi Fen Bilimleri Enstitüsü Astronomi ve Uzay
Bilimleri Ana Bilim Dalı. (Eylül 2010 – Temmuz 2014)

Çalıştığı Kurumlar:

Veriteknik Bilişim Basın Yayın ve Ltd. Şti. (2012 – 2014)

SCI Yayınlar

Yeşilyaprak, C., Yerli, S. K., Güçsav, B. B., Aksaker, N., Dikicioğlu, E., Helvacı, M., Çoker, D., **Aydın, M. E.**, Dinçel, B., Uzun, N. 2012. Long-term variations and periods of Mira stars from ROTSE-IIId. New Astronomy, Volume 17, Issue 5, pp. 504-513.

Hakemli Dergiler

Güçsav, B. B., Yeşilyaprak, C., Yerli, S. K., Aksaker, N., Kızıloğlu, Ü., Çoker, D., Dikicioğlu, E., **Aydın, M. E.** 2012. A pipeline for the ROTSE-IIId archival data. Experimental Astronomy, Volume 33, Issue 1, pp.197-209.

Aksu, O., Özavci, İ., Yüce, K., Tanrıverdi, T., Şenavci, H. V., Yılmaz, M., Kara, A., Kaya, F., Çetintaş, C., Çınar, D., Özuyar, D., Baştürk, Ö., **Aydın, M. E.**, Atlagan, Ö., Şener, H. T., Kabadayı, Ö., Yelkenci K. and Evin, B. 2005. Photoelectric minima of some Eclipsing Binary Stars. Information Bulletin on Variable Stars. 5588.

Uluslararası Kongreler

Güçsav, B. B., Yeşilyaprak, C., Yerli, S. K., Aksaker, N., **Aydın, M. E.**, Çoker, D. 2009. An HPC Application for the ROTSE-III Archival Data. High Performance Computing in Observational Astronomy: Requirements and Challenges. Pune, India.

Ulusal Kongreler

Kılıç, Y., Özdemir, S., **Aydın, M. E.**, Bağırar, M. N., Okuyan, O., Şahin, Ş. Ş. 2012. DIBRED: Canon 550D Kamerası Görüntülerinden FITS Formatlı Astronomik Görüntülere Dönüşüm ve Fotometri Yazılımı. XVIII. Ulusal Astronomi Kongresi, Malatya.

Aydın, M. E., Yeşilyaprak, C., Güçsav, B. B. 2010. TUG Veri Arşivi: Yıldız Kümelerinde Değişen Yıldızlar. XVII. Ulusal Astronomi Kongresi, Adana.

Aydın, M. E., Yeşilyaprak, C., Güçsav, B., B. 2008. CCD Verilerinden Değişen Yıldız Tespiti ve Analizi. XVI. Ulusal Astronomi Kongresi, Çanakkale.

Yeşilyaprak, C., Yerli, S. K., Aksaker, N., Güçsav, B. B., Çoker, D., **Aydın, M. E.** 2008. ROTSE ve Açık Kümelerde Değişen Yıldız Avı. XVI. Ulusal Astronomi Kongresi, Çanakkale.

Derman, E., Güçsav, B. B., **Aydın, M. E.** 2008. TUG Arşiv Projesi. XVI. Ulusal Astronomi Kongresi, Çanakkale.

Helvacı, M., **Aydın, M. E.** 2006. Bazı AGB Yıldızlarının Düşük Çözünürlüklü Kırmızı-Öte Tayflarının Modellenmesi. XV. Ulusal Astronomi Kongresi, İstanbul.

Aydın, M. E., Yeşilyaprak, C. 2006. T40 ile Farklı Türden Değişen Yıldız Gözlemleri. XV. Ulusal Astronomi Kongresi, İstanbul.