

T.C.
GEBZE YÜKSEK TEKNOLOJİ ENSTİTÜSÜ
MÜHENDİSLİK VE FEN BİLİMLERİ ENSTİTÜSÜ

STATİK VE DİNAMİK YÜKLEMELER
ALTINDAKİ YAPILARIN ARDIŞIK TAGUCHİ
METODU İLE OPTİMUM TASARIMI İÇİN
FORTRAN PROGRAMI GELİŞTİRİLMESİ

Mehmet Koray SAVUR
YÜKSEK LİSANS TEZİ
TASARIM VE İMALAT MÜHENDİSLİĞİ
ANABİLİM DALI

GEBZE

2007

T.C.
GEBZE YÜKSEK TEKNOLOJİ ENSTİTÜSÜ
MÜHENDİSLİK VE FEN BİLİMLERİ ENSTİTÜSÜ

STATİK VE DİNAMİK YÜKLEMELER
ALTINDAKİ YAPILARIN ARDIŞIK TAGUCHİ
METODU İLE OPTİMUM TASARIMI İÇİN
FORTTRAN PROGRAMI GELİŞTİRİLMESİ

Mehmet Koray SAVUR
YÜKSEK LİSANS TEZİ
TASARIM VE İMALAT MÜHENDİSLİĞİ
ANABİLİM DALI

TEZ DANIŞMANI
YRD. DOÇ. DR. Hasan KURTARAN

GEBZE

2007

G.Y.T.E. Mühendislik ve Fen Bilimleri Enstitüsü Yönetim Kurulu'nun
..... tarih ve/..... sayılı kararıyla oluşturulan jüri tarafından
...../...../..... tarihinde tez savunma sınavı yapılan Mehmet Koray SAVUR'un tez
çalışması Tasarım ve İmalat Mühendisliği Anabilim Dalında YÜKSEK LİSANS tezi
olarak kabul edilmiştir.

JÜRİ

ÜYE

(Tez Danışmanı) : Yrd. Doç. Dr. Hasan KURTARAN

ÜYE :

ÜYE :

ÜYE :

ÜYE :

ONAY

G.Y.T.E. Mühendislik ve Fen Bilimleri Enstitüsü Yönetim Kurulu'nun
..... tarih ve/..... sayılı kararı.

ÖZET

TEZİN BAŞLIĞI: Statik ve Dinamik Yüklemeler Altındaki Yapıların Ardışık Taguchi Metodu ile Optimum Tasarımı İçin Fortran Programı Geliştirilmesi

YAZAR ADI : Mehmet Koray SAVUR

Genetik algoritmalar gibi kesikli tasarımlar oldukça pahalı bir hesaplama maliyeti içermektedirler. Bu nedenle, kesikli tasarımlar için genetik algoritmaya göre daha verimli bir yöntem sunan Taguchi metodu son zamanlarda etkili bir şekilde kullanılmaya başlanmıştır. Klasik Taguchi optimizasyon metodu, faktörlerin etkileşimini ihmal ederek optimum parametre değerlerini tahmin etmektedir. Bu çalışmada, klasik Taguchi optimizasyon metodu kısıtlı ve doğrusal olmayan yapısal optimizasyon problemlerini daha etkili olarak çözecek şekilde genelleştirilmiş ve geliştirilmiştir.

Geliştirilen algoritma, klasik Taguchi metodunun dezavantajını ortadan kaldırmak için ardışık optimizasyon yöntemini kullanmaktadır. Geliştirilen algoritmayı yapısal optimizasyon problemlerine uygulamak için Fortran95 programı yazılmıştır. Geliştirilen Fortran95 programı çeşitli örnek test problemleri üzerinde denenmiştir. Elde edilen sonuçlar genetik algoritmadan elde edilen sonuçlarla ve literatürdeki diğer sonuçlarla karşılaştırılmış ve tartışılmıştır. Geliştirilen ardışık Taguchi optimizasyon metodunun kesikli optimizasyon problemleri için daha verimli bir çözüm yöntemi sunduğu görülmüştür.

SUMMARY

THESIS TITLE : A Fortran Code Development for Design Optimization of Statically and Dynamically Loaded Structures with Taguchi Orthogonal Arrays

NAME OF THE AUTHOR : Mehmet Koray SAVUR

Discrete design optimization methods such as Genetic Algorithm are extremely expensive in computational cost. Taguchi Method has been widely used recently since it is more effective than Genetic algorithm methods in discrete design space. Classical Taguchi optimization method estimates optimal values while neglecting the interaction of factors. In this study, Taguchi method is generalized and improved in order to solve nonlinear constrained structural optimization problems more effectively.

The improved algorithm uses successive Taguchi orthogonal arrays to avoid the disadvantages of the classical Taguchi method. A Fortran95 program has been developed to implement the developed algorithm to the structural optimization problems. Several static and dynamic benchmark problems have been solved with the developed discrete optimization program. Results from the program show that successive Taguchi optimization method produces better optimum with much less computational cost compared to Genetic algorithm and other methods in the literature.

TEŞEKKÜR

Bu çalışmada uygulanan optimizasyon probleminin çözümünde ortaya konan Fortran95 programında bana her konuda destek olan, yardım ve anlayışını hiçbir zaman esirgemeyen Sayın danışmanım Yrd. Doç. Dr. Hasan Kurtaran'a, her zaman destek veren eşim Gülsev'e ve bana çalışmak için yeterince vakit sağlayan biricik oğlum Mete'ye sonsuz teşekkürü bir borç bilirim.

İÇİNDEKİLER DİZİNİ

	Sayfa
ÖZET	iv
SUMMARY	v
TEŞEKKÜR	vi
İÇİNDEKİLER DİZİNİ	vii
SİMGELER VE KISALTMALAR DİZİNİ	ix
ŞEKİLLER DİZİNİ	x
ÇİZELGELER DİZİNİ	xi
1 OPTİMİZASYON HAKKINDA GENEL BİLGİ	1
1.1 Optimizasyona Giriş	1
1.2 Sürekli Optimizasyon	3
1.3 Kesikli Optimizasyon	3
1.3.1 Kesikli optimizasyon türleri	4
1.3.2 Kesikli Optimizasyon Problemlerinin Uygulama Alanları	6
1.3.3 Kesikli optimizasyon problemlerinin çözüm yöntemleri	8
1.3.4 Kesikli optimizasyon problemleri için klasik algoritmalar	9
1.3.5 Kesikli optimizasyon problemleri için sezgisel algoritmalar	11
2 DENEY TASARIM METOTLARI	14
2.1 Giriş	14
2.2 Ortogonal Dizinler Kullanarak Deney Tasarımı	15
2.2.1 Ortogonal Dikey Dizinlerin Oluşturulması	15
3 ARDIŞIK ORTOGONAL DİZİNLER KULLANARAK KESİKLİ TASARIM	18
3.1 Giriş	18
3.2 Algoritma Adımları	19
4 FORTRAN 95	25
4.1 Fortran95 Kullanma Sebebi	25
4.2 Fortran'ın Tarihsel Gelişimi	25
4.3 Sürümleri	26
4.4 Fortran95 Özel Komutları	27
5 ÖRNEK TEST PROBLEMLERİ	31
5.1 İki Çubuklu Kiriş	31

5.2 On Çubuklu Kiriş	36
5.3 Doğrusal Olmayan Darbe Emici	45
5.3.1 Çok kütleli sistemin hareket denkleminin sayısal çözümü	47
5.3.1.1 Çok kütleli sistemlerde hareket denklemleri	47
5.3.1.2 Çok kütleli sistemlerde hareket denkleminin çözümü	48
6 SONUÇLAR VE ÖNERİLER	52
KAYNAKLAR DİZİNİ	54
ÖZGEÇMİŞ	56

SİMGELER VE KISALTMALAR DİZİNİ

Simge	Açıklama
m	Bütün karakteristiklerin ortalaması
m_{A1}	A_1 etkisini içinde barındıran karakteristiklerin ortalaması
x	Karar değişkeni vektörü
$R(x)$	Amaç fonksiyonu
$g_j(x)$	Kısıt fonksiyonu
l	Kısıtların sayısı
$P(x)$	Ceza fonksiyonu
v_i	i 'nci kısıtın maksimum ihlali
s	Ölçek faktörü
M	Sistemin kütlesi
C	Sistemin sönüm katsayısı
K	Sistemin yay sabiti
x_i	Yerdeğiştirme vektörü
$\dot{x}_i$	Hız vektörü
$\ddot{x}_i$	İvme vektörü
μ	Poisson oranı
E	Elastisite modülü
ρ	Yoğunluk

ŞEKİLLER DİZİNİ

Şekil	Sayfa
1.1 Kesikli optimizasyon çözüm yöntemleri	9
3.1 Düzey değerlerinin seçilmesine bir örnek	22
3.2 Ardışık dizinler kullanan iteratif optimizasyon algoritması	23
5.1 Optimizasyon tasarımı yapılacak iki çubuklu kiriş	33
5.2 İki çubuklu kiriş için çubuk kesit alanlarının değişimi	34
5.3 İki çubuklu kiriş için mesnetler arası mesafenin değişimi	35
5.4 İki çubuklu kiriş için kütle değişim grafiği	35
5.5 On çubuklu kiriş için Fortran95 algoritması ve Lee ve arkadaşlarının Taguchi yöntemini karşılaştırır kütle değişim grafiği	37
5.6 Optimizasyon tasarımı yapılacak on çubuklu kiriş	38
5.7 On çubuklu kiriş için kütle değişim grafiği	40
5.8 On çubuklu kiriş için Fortran95 algoritması ve Lee ve arkadaşlarının Taguchi yöntemini karşılaştırır kütle değişim grafiği	42
5.9 On çubuklu kiriş için toplam gerçekleştirilen sonlu eleman analiz sayısı	43
5.10 On çubuklu kiriş için karşılaştırmalı hesap maliyetleri	43
5.11 On çubuklu kiriş için optimal kütle kıyaslama grafiği	44
5.12 Doğrusal olmayan darbe emici	46
5.13 Doğrusal olmayan darbe emici için ivme değişim grafiği	50
5.14 Doğrusal olmayan darbe emici için kıyaslamalı ivme değişim grafiği	51

ÇİZELGELER DİZİNİ

Tablo	Sayfa
2.1 $L_9(3^4)$ Ortogonal Dikey Dizini	17
2.2 Her Düzeye Karşılık Gelen η ortalamaları	18
5.1 İki çubuklu kiriş karar değişkenleri ve amaç fonksiyonu optimum değerlerinin iterasyonla değişimi	36
5.2 İki çubuklu kirişte amaç fonksiyonu iyileştirme oranı	36
5.3 Karar değişkenlerinin optimumlarının ilk dizayn ve literatürdeki diğer değerlerle kıyaslanması	40
5.4 On çubuklu kirişte karar değişkenlerinin atanması $L_{27}(3^{13})$	41
5.5 On çubuklu kiriş için ilk dizayn sonuçları ve diğer sonuçların kütle iyileştirme oranları karşılaştırması	44
5.6 On çubuklu kirişte amaç fonksiyonu iyileştirme oranı	45
5.7 On çubuklu kiriş karar değişkenleri ve amaç fonksiyonu optimum değerlerinin iterasyonla değişimi	45
5.8 Doğrusal olmayan darbe emici amaç fonksiyonu iyileştirme oranı	52
5.9 Doğrusal olmayan darbe emici karar değişkenleri ve amaç fonksiyonu optimum değerlerinin iterasyonla değişimi	52

1 OPTİMİZASYON HAKKINDA GENEL BİLGİ

1.1 Optimizasyona Giriş

Optimizasyon denince karar vermede matematiksel yöntemlerin kullanılması anlaşılmaktadır. Bu matematiksel yöntemlere literatürde sayısal optimizasyon veya matematiksel programlama metotları denmektedir. Etkili bir karar verme için tasarım problemlerinin uygun bir matematiksel formatta ifade edilmesi gerekmektedir.

Bir tasarım problemini optimizasyon yöntemiyle etkili bir şekilde çözmek için aşağıdaki faktörlerin dikkate alınması gerekmektedir:

- Alternatif kararlar nelerdir?
- Karar hangi kısıtlar altında verilmiştir?
- Bu alternatifleri değerlendirmede kullanılacak uygun bir amaç fonksiyonu nasıl olmalıdır?

Bu soruların cevaplarını bulabilmek için ise

- Karar değişkenlerini,
- Amaç fonksiyonlarını,
- Tasarım sınırlamalarını,

tanımlamak gerekmektedir.

İlk olarak sistemi tanımlamak için, karar değişkenleri olarak isimlendirilen bir değişken seti belirlenir. Daha sonra, bu değişkenlere tahmini olarak sayısal değerler atanarak sistem oluşturulur. Karar değişkenleri, hakkında alınacak kararları temsil eden değişkenlerdir. Karar değişkenleri genellikle x, y gibi sembolik olarak tanımlanır. Karar değişkenleri, devamlı ve kesikli olmak üzere iki sınıfa ayrılır ve bu bir sonraki sayfada bahsedilen optimizasyon türlerinin belirleyici kriteridir. Optimizasyon problemlerinde amaç fonksiyonu, enbüyüklenecek (enküçüklenecek)

karar değişkenlerinin toplam getirisinin miktarını tanımlayan fonksiyonlardır. Kısıtlar, kararların hangi sınırlar arasında alınması gerektiğini belirtir.

Mühendislik problemlerinin birçoğu kısıtlı problemlerden oluşur. Burada ana amaç, kısıtları olan bir fonksiyonu enküçükmektir. Kısıtlı bir optimizasyon probleminin sayısal optimizasyon yöntemleriyle çözülebilmesi için genellikle aşağıdaki şekilde matematiksel olarak formüle edilir:

Amaç fonksiyonu : Enküçüle (Enbüyükle) $f(x)$

Kısıtlar : $g_i(x) \leq 0$; $i=1,...,m$

$h_j(x) = 0$; $j=1,...,l$

Burada $f(x)$ amaç fonksiyonu, $g(x)$ eşitsizlik kısıtı, $h(x)$ ise eşitlik kısıtı olarak adlandırılır. $x=(x_1, x_2, ..., x_n)^T$, n gerçek değişkenli karar değişkeninin sütun vektörüdür.

Bütün sistemler, kaynaklar, malzeme dayanımı, sistemin cevabı, sistem elemanlarının boyutları vb. bazı tasarım sınırlamaları içerisinde çalışacak şekilde tasarlanır. Bu sınırlamalar karar değişkenlerine bağımlı olarak tanımlanır. Eğer bir tasarım tüm sınırlamaları sağlıyorsa, sistem uygun veya çalışabilir(feasible)'dir. Eğer belirtilen sınırlamaları sağlamıyorsa sistem uygun değildir ve uygun olmayan, kabul edilemez veya çalışamaz (infeasible) olarak isimlendirilir. Bulunan uygun tasarımlardan biri kullanışlıdır.

Elde edilen bir tasarımın bir başka tasarımdan daha iyi olup olmadığını anlayabilmek için bir kritere ihtiyaç vardır. İşte bu kritere hedef fonksiyonu denir. Bütün tasarım problemleri bir hedef fonksiyonuna sahiptir. Hedef fonksiyonu da karar değişkenlerine bağlı olarak tanımlanır.

Ayrıca, bu fonksiyonların birçoğu kesin sınırlamalara sahiptir. Genel matematiksel programlama problemlerinde tanımlanan çözüm vektörü $x^* = \{x_1^*, x_2^*, ..., x_n^*\}$, hedef fonksiyonunu $f(x)$ enküçükmektir.

Optimizasyon problemleri, genel olarak karar deęiřkenlerinin tipine baęlı olarak iki kısma ayrılmaktadır:

- Sürekli Optimizasyon
- Kesikli Optimizasyon

1.2 Sürekli Optimizasyon

Sürekli optimizasyonda, amaç fonksiyonu ve kısıtlar, karar deęiřkenlerinin sürekli bir fonksiyonudur. Amaç fonksiyonu sistemin etkinlięinin enbüyüklenmesi, maliyetin enküçüklenmesi gibi hedefler biçiminde ifade edilir. Sürekli optimizasyon kısıtlı veya kısıtsız optimizasyon olmak üzere iki ana tiptedir. Kısıtlı sürekli optimizasyonda, kısıtlar eřitlik ya da eřitsizlikler biçiminde yazılabilir.

Bir optimizasyon probleminde karar deęiřkeni belli bir aralıkta tüm deęerleri alabiliyorsa, bu karar deęiřkeni sürekli olarak adlandırılır. Optimizasyon modellerinde sürekli türden karar deęiřkenleri genellikle $(0, \infty)$ aralığında tanımlanır.

1.3 Kesikli Optimizasyon

Kesikli optimizasyonda, amaç fonksiyonu ve kısıtlar, karar deęiřkenlerinin sürekli bir fonksiyonu deęildirler. “Kombinatorik Optimizasyon” adı ile de anılan ve temelini matematiksel programlamadan alan kesikli optimizasyon, tamsayılı programlama, reel programlama ya da uzay kafes yapı sistemlerinin dizaynında kullanıldığı gibi her ikisinin de beraber kullanımı řeklinde ortaya çıkabilir. Kesikli optimizasyon, mühendislikten iktisada, işletmecilikten askeri alana geniş bir uygulama alanına karar stratejileri geliřtirmek anlamında hitap eder.

En geniş anlamıyla, normal olarak sürekli biçimde tanımlanan karar deęiřkenlerinin, kesikli deęerler alan karar deęiřkenleri biçiminde tanımlandığı kesikli optimizasyon, hem problemlerin modellenmesinde hem de bu modellerin problem çözümünde kullanılmasında etkin ve hızlı çalışan algoritmalara ihtiyaç duyar.

Karar deęiřkeni sadece belli deęerleri alıyorsa, bu durumda kesikli deęiřken olarak adlandırılır. Optimizasyon modellerinde bu türden karar deęiřkenleri genellikle (0–1) olarak ya da negatif olmayan tamsayılar olarak adlandırılır. Bir karar deęiřkeninin standart bir veritabanından seęilme zorunluluęu olduęu durumlarda yapısal tasarımlar kesikli karar alanı ile belirlenir. Buna örnek olarak kesit modülünün var olan bir tablodan seęilme zorunluluęu verilebilir. Ya da ele alınan problem masa veya sandalye üretimi ise 1,23 masa ya da 4,6 sandalye gibi çözümler anlamlı olmazlar. Her ne kadar sürekli optimizasyon mükemmel bir sonuç üretse de, uygulamalarda sonuçların kesikli deęerlere dönüřtürölerek deęiřtirilmesi gerekir. Çoęu uygulamada yukarı yuvarlatılmış deęerler ele alınmaktadır. Her ne kadar bu deęerleri kullanmak gerilme ve yerdeęiřtirme kısıtlarını saęlasa da, titreřim problemlerindeki özdeęer kısıtları garanti edilemez.

Yukarıda bahsedilen nedenlerden dolayı, deęiřkenlerin kesikli olarak tanımlandığı problemler literatürde “kesikli optimizasyon problemleri” ve özelde de “tamsayılı optimizasyon problemleri” olarak ele alınır. Kesikli optimizasyon problemleri doğrusal veya doğrusal olmayan türden olabilir. Kesikli doğrusal programlama algoritmalarına örnek olarak Dal-Sınır algoritması, Sıfır-Bir saklı algoritması ve kesme düzlemi algoritması verilebilir.

1.3.1 Kesikli optimizasyon türleri

Kesikli optimizasyon yöntemleri karar deęiřkenlerinin tipine göre ařağıdaki türlerdedir: (Bakır ve Altunkaynak, 2003)

- Saf tamsayılı programlama problemi
- Karma tamsayılı programlama problemi
- 0–1 tamsayılı programlama modelleri
- 0–1 tamsayılı karma programlama modelleri
- Tamsayılı uygunluk problemleri
- 0–1 tamsayılı uygunluk problemleri
- Doğrusal programlama gevřetmesi

Tüm değişkenlerin tamsayı olması gerektiği bir tamsayılı programlama problemine “saf tamsayılı programlama problemi” denir. Örneğin, personel programlama problemleri bu türdendir. Vardiya usulü çalışan bir işyerinde hangi günlere ne kadar personel ataması yapılması gerektiği ile ilgilenildiğinde amaç toplam ödenen ücretleri enküçüklemek biçiminde olabilir.

Sadece bazı değişkenlerin tamsayı, diğerlerinin reel değerler olmasının gerektiği tamsayılı programlama problemi “karma tamsayılı programlama problemi” olarak adlandırılır.

Tüm değişkenlerin 0 ya da 1’e eşit olmasının istendiği bir tamsayılı programlama problemine “0–1 tamsayılı programlama” denir. Sırt çantası problemleri bu türdendir. Bir dağcının tırmanırken beslenmesi için sınırlı hacme sahip sırt çantasına hangi yiyecekleri koyması gerektiğine karar verirken 0–1 türünden tanımlanmış değişkenler söz konusudur.

Bazı değişkenlerin 0 ya da 1 değerini aldığı, geriye kalan değişkenlerin ise sürekli değişkenler olarak tanımlandığı modellere “0–1 tamsayılı karma programlama modelleri” denir. Bir üretim planlaması problemi bu türdendir. Bir üretim çevresinde iki tür karar olabilir. Karar değişkenlerinin birincisi ürünün hangi tezgâha atanması gerektiği, ikincisi ise bu tezgâhlarda ne kadar üretim gerçekleştirileceği biçiminde tanımlanabilir.

En iyilenecek amaç fonksiyonunun mevcut olmadığı, doğrusal kısıtlar sistemi için tamsayılı çözümlerin bulunmasının hedeflendiği modeller “Tamsayılı uygunluk problemleri” olarak adlandırılır.

Belli bir doğrusal kısıtlar sisteminin 0–1 çözümlerinin bulunmasının hedeflendiği problemlere “0–1 tamsayılı uygunluk problemleri” denir.

Tüm tamsayıları ya da değişkenler üzerindeki 0–1 kısıtlarını ihmal ederek elde edilen bir doğrusal programlama problemine tamsayılı programlamanın “Doğrusal programlama gevşetmesi” denir.

Kesikli optimizasyon yapılması gerekirken, doğrusal programlama probleminin çözülerek tamsayı olması gereken değişkenler için kesirli sayıları en yakın tamsayıya yuvarlamak şeklindeki bir yaklaşımın ne kadar doğru çözümler ürettiği literatürde araştırılmış ve gerçek optimal çözümlerin çoğunlukla uzağına düşen noktalarda sonuçlar elde edildiği görülmüştür.

1.3.2 Kesikli Optimizasyon Problemlerinin Uygulama Alanları

Pratikte kesikli optimizasyon problemleri aşağıdaki şekillerde formüle edilebilir: (Bakır ve Altunkaynak, 2003)

- Sırt çantası problemleri
- Sermaye bütçeleme problemleri
- Sabit yük problemleri
- Tezgâh yerleştirme problemleri
- Küme örtme, paketleme ve ayrıştırma modelleri
- Ya-Veya kısıtlı problemler
- İse-O zaman kısıtlı problemler
- Parçalı doğrusal fonksiyon
- Makine ardışıklık problemleri
- Atama eşleştirme problemleri
- Gezgin satıcı ve güzergâh modelleri

Sırt çantası probleminin, bir dağcının tırmanırken sınırlı kapasiteye sahip sırt çantasını optimum biçimde doldurması biçiminde tanımlanması, gerçek dünya problemlerinin kolayca ifade edilmesinden dolayıdır. Portföy seçimi, sermaye bütçeleme, proje seçimi bu tip problemlere örnek olarak verilebilir. Karar değişkenleri bakımından sırt çantası problemleri

- Negatif olmayan sırt çantası problemleri,
- 0–1 Sırt çantası problemleri,

şeklinde, problemin içerdiği kısıt bakımından ise:

- Bir boyutlu sırt çantası problemleri,

- Çok boyutlu sırt çantası problemleri, şeklinde sınıflandırılabilir.

Sermaye bütçeleme modelleri özünde sırt çantası problemleri türündendir. Bu problemler, en büyük getiriye verecek proje, yatırım vb.lerinin optimal bir koleksiyonunu bütçe ya da kaynak kısıtları altında veren modellerdir.

Sabit yük problemleri, herhangi bir karar değişkeninin gerçekleşip gerçekleşmemesine bağlı olarak bir defalık giderlerin söz konusu olduğu durumlarda karşılaşılan problemlerdir. Örneğin bir işletmede üretilecek yeni bir ürün için gerekli makine yatırım maliyeti analizi bu sınıfa girer.

Tezgâh yerleştirme problemleri, sabit yük problemlerinin başka bir türüdür. Bu tip problemlerde amaç karar değişkeni olan değişik araçların (tezgâh, dolap gibi) nereye yerleştirileceğine karar verilir.

Küme örtme, paketlenme ve ayrıştırma modellerinin temel özelliği, belli bir alt kümede üyeliğin söz konusu olduğu problemleri formüle etmeleridir.

Bir problemin ana amacını tanımlarken bazen iki kısıttan biri ya da diğerini sağlanması istenir. Bu durumlarda kullanılan ya-veya kısıtlı problemlerinde iki kısıttan biri sağlandığında diğerinin sağlanmasına gerek yoktur.

Bazı problemlerde ise herhangi bir değişken belli bir değeri aldığı anda diğer kısıtların sağlanması istenir. Bu tür durumlarda da ise-o zaman kısıtlı problemler kullanılır.

Amaç fonksiyonunun doğrusallığı bozulmasına rağmen, parçalı doğrusal fonksiyonlardan oluşan doğrusal olmayan bir fonksiyon söz konusu olduğunda bu fonksiyonlar bazı manipülasyonlarla doğrusal hale dönüştürülerek parçalı doğrusal fonksiyonlar kullanılır.

Bir ya da daha fazla kritere göre verilen işlerin en iyi sıralamasıyla ilgilenen problemlerden oluşan makine ardışıklık problemleri, toplam akış zamanının, toplam zamanın enküçüklemesi, toplam çıktının enbüyüklemesi gibi amaçlara ulaşmak için kullanılabilir.

İki nokta arasında bir ticari malın akışı söz konusu iken atama problemlerinde iki nokta arasında bir bağ kurmak mümkündür. Karar değişkenleri burada 0–1 türündendir. Eşleştirme modelleri ise nesnelerin optimal eşleştirmesini araştırır.

Gezgin satıcı ve güzergâh modelleri belli bir kümedeki her bir noktayı sadece bir defa ziyaret etmek için en küçük toplam uzaklığı verecek güzergâhı araştıran problemlerdir.

1.3.3 Kesikli optimizasyon problemlerinin çözüm yöntemleri

Kesikli optimizasyon için birçok yöntem sunulmuştur [Arora,1994]. Park ve Arora'nın çalışmasında, etkileşimli optimizasyon sürecinde karar değişkenleri sabit bir değerde tutulmaktadır [Park ve Arora, 1987]. Bu yaklaşım çok pratik olmasının yanında, kullanıcı odaklı olduğundan alınan kararlar tamamen kullanıcının bilgi ve seviyesiyle bağıntılıdır. Tamsayılı programlama algoritmaları için Dal ve Sınır metotları geliştirilmiştir [Vanderplaats ve Salajegheh, 1993]. Ayrıca optimizasyonun, optimizasyon sonrası işlemleri için İkili Metot kullanılmıştır. Kesikli tasarım için genetik algoritma ve Tabu Araması gibi birkaç kesikli tasarım algoritmaları kullanılmıştır [Rajeev ve Krishnamoorthy, 1992]. Bu metotlarda kullanılan fonksiyon hesaplamalarının sayısı oldukça fazladır. Yapısal optimizasyonda, büyük ölçekli yapıların optimizasyonu sonlu eleman analizi ile yapılmaktadır ve bu nedenle son derece hesap maliyetlidir. Bu ise pratik uygulamalarda kullanımı sınırlandıran bir etken olarak karşımıza çıkmaktadır. Bu araştırmada, kesikli alanda tasarım için ardışık dizinler kullanılarak iteratif bir optimizasyon algoritması geliştirilmiştir [Ku, Rao ve Chen, 1998]. Yani, geliştirilen metot optimum değerlere ulaşmak için Deney Tasarımı (DOE) analizlerini ardışık dizinlerin yardımıyla iteratif olarak kullanır [Peace, 1995]. Ardışık dizinler

ilgilenilen kesikli tasarım değerleriyle düzenlenerek bu kesikli tasarım alanına uygulanabilir.

Kesikli optimizasyon çözüm yöntemleri Şekil 1.1’de gösterilmiştir.



Şekil 1.1 Kesikli Optimizasyon Çözüm Yöntemleri

1.3.4 Kesikli optimizasyon problemleri için klasik algoritmalar

Balas Algoritması:

Balas Algoritması 0–1 türünden problemlerin çözümü için geliştirilmiş bir algoritmadır. “Dolaylı Sayma” olarak da bilinir. Avantajı sadece toplama ve çıkarma işlemlerini kullanmasıdır. Orta ölçekli problemler için tatmin edici bir yöntemdir.

Dal ve Sınır Algoritması:

Klasik algoritmalar içerisinde en çok uygulanan ve bilinen algoritmadır. Bu algoritma, optimum aramasını tam sayma yerine kısmi sayma yöntemiyle gerçekleştiren bir algoritmadır. Dal ve sınır yönteminin kullandığı üç temel araç vardır:

- Dallandırma ya da ayrıştırma
- Sınırlama
- Bağlama

Dal ve sınır yönteminin kullandığı stratejiler üçtür:

- Alt sınır stratejisi
- Dallandırma stratejisi
- Arama stratejisi

Dal ve sınır yöntemi, saf tamsayılı programlama problemi, 0–1 sırt çantası problemi, makine ardışıklık problemi ve gezgin satıcı problemlerinin çözümünde kullanılır.

Düzlem kesme algoritması:

Bu algoritma, dal ve sınır yöntemindeki kesirli değişkenler üzerine iteratif olarak kısıtlar koyma yerine, kısıtlar dizisi ya da kesmeler türetip bunları probleme ekleyerek problemi yeniden çözer. Bu algoritmayı kullanmanın en önemli zorlukları arasında orta boyuttaki problemler için bile etkinsizliği sayılabilir. Düzlem kesme algoritması, saf tamsayılı programlama ve karma tamsayılı programlama problemlerinde kullanılabilir.

Lagrange gevşetme yöntemi:

Bu yöntemde tamsayı kısıtları olduğu gibi muhafaza edilir, fakat problemin bazı ana kısıtlarında gevşetme yapılabilir. Böylece bu şekilde gevşetilmiş problem yine tamsayı bir çözüm verecektir. Gevşetilmiş kısıtlar problemten tamamıyla düşülmeyip amaç fonksiyonunda uygun Lagrange çarpanlarıyla yer alarak, kısıtın sağlanmamasının önüne geçilir.

1.3.5 Kesikli optimizasyon problemleri için sezgisel algoritmalar

Optimizasyonda kesin yöntemlerin aksine sezgisel yöntemler optimal çözümü garanti etmezler. Sezgisel yöntemlerin kullanılma nedenleri aşağıdaki şekilde sıralanabilir:

- Kesin formülasyonu ve çözüm yöntemi olmayan problemler için sezgisel yöntemler kullanılır.
- Problem için kesin çözüm yöntemleri olsa bile, büyük ölçekli problemlerde yoğun hesaplama gereklilikleri ve çözüm zamanı kısıtlı olduğunda, özellikle gerçek zamanlı uygulamalarda sezgisel yöntemler tercih edilebilir.
- Sezgisel yöntemler aynı zamanda dal ve sınır çözüm yönteminde optimal çözümün sınırlarını hesaplamak için kullanılabilir.

Şekil 1.1’de gösterilen sezgisel algoritma yöntemleri aşağıda kısaca açıklanmaktadır.

Greedy sezgisel yöntemi:

Bir optimizasyon problemini çözmek için geliştirilen bir Greedy algoritma çözümü her bir adımda cari kısmi çözümü en büyük kazancı elde edecek şekilde artırarak elde etmeye çalışır. Greedy yöntemlerinin üç temel özelliği şunlardır:

- Artırma özelliği
- Geriye doğru gitmeme özelliği
- Greedy seçim özelliği

Greedy algoritmalarda adım sayısı problem büyüklüğünün polinomial bir fonksiyonudur. Çalışma mekanizmasında ileriye doğru bakmaz.

Greedy yöntemi en küçük yol ağacı problemi, 0–1 sırt çantası problemi, küme örtme problemi, gezgin satıcı problemlerinin çözümünde kullanılabilir.

Benzetimli tavlama:

Bu algoritmanın mantığı ve prensibi erimiş metali ve soğutma prosesinin fiziksel işleyişine dayanmaktadır. Bu fiziksel işlemde amaç, bir metalin kristal yapısı en küçük enerji durumuna ulaşacak şekilde metali dikkatlice soğutarak moleküller arası kuvvetli bağlara sahip katı yoğunluk elde etmektir. Bu fiziksel işlemin işleyişine dayalı olarak zor optimizasyon problemlerinin çözümü için üretilen algoritmadaki kavramların karşılıkları hızlı soğutma için en hızlı azalan yerel arama (steepest descent local search), dikkatli soğutma için benzetimli tavlama (simulated annealing), soğutma aşamalarında ısıyı aşamalı olarak azaltma için iyileştirme sağlamayan ilerlemeleri kabul etme olasılığının azaltılması (gradual reduction of the probability of accepting a non-improving move) olarak verilebilir.

Benzetimli tavlama, güzergâh belirleme problemi ve gezgin satıcı problemlerinin çözümünde kullanılabilir.

Tabu araması:

Tabu Arama (TA) algoritması ilk defa F. Glover tarafından insan hafızasının çalışmasından esinlenilerek önerilmiş bir yerel arama yöntemidir [Glover, 1990]. TA ana olarak basit tepe tırmanma (BTT) yönteminin zaaflarını gidermek için düşünülmüştür. BTT eldeki aday çözüme, bir komşuluk işleci uygulayarak, yeni adaylar üretir. Yeni adaylar bir değerlendirmeye tabi tutulur. Değerlendirme çözümün sonuca yakınlığını ölçer. Yeni adaylar ile eldeki eski aday içerisinde en yakın olan eskinin yerine geçer. BTT bu haliyle kısır bir döngüye sebep olabilir. Komşuluklar kapsamında birbirine eşit değerdeki iki ya da daha fazla komşu arasında BTT takılıp kalabilir. Arama yapılan alanın özellikleri ya da BTT yönteminin iyi seçilmemesi, herhangi bir denetim mekanizması kullanılmadan yapılan aramayı yerel en iyi noktalara götürebilir. Fakat gerçek hayattaki problemler yerel en iyilerin bol olduğu, fakat global iyinin az, hatta tek olabildiği problemlerdir. Kısacası kısır döngü kaçınılmazdır. Tabu arama yöntemi kısır döngüden kurtulmak için hafıza kullanılmasını, hatırlamayı önerir. Daha önce ziyaret edilmiş ya da herhangi bir nedenle ziyaret edilmesi istenilmeyen aday çözümlerle ilgili özellikler, *tabu listesi* adı verilen, kısa dönem hafızaya benzer bir yapıda tutulur. Yöntem bu

listedeki hamlelerin belirli bir süre yapılmasını yasaklar. Böylece arama yerel bir en iyi noktadan kurtularak asıl sonuca yakınsayabilir. Bu temel altyapının yanı sıra, tabu arama yöntemlerinde aramayı belirli bir noktaya yönlendirebilecek uzun dönem hafıza yapıları da kullanılmıştır.

Genetik Algoritmalar:

Genetik Algoritmalar (GA) ilk defa J. Holland tarafından önerilmiş ve pek çok zor problemin çözümünde kullanılmıştır [Holland, J. H., 1975]. GA'lar evrim teorisinden esinlenilmiş, doğal seçim ve genetik biliminin temellerini kullanılarak geliştirilmiştir. Tipik bir GA'de içindeki her birey (kromozom) in aday çözüm olduğu nüfus, sürekli bir evrim geçirir. Her nesilde belirli bir yöntemle seçilmiş olan bireylere melezleme ve mutasyon gibi çeşitli genetik uzmanlar uygulanır. Her bireyin hayatta kalma şansını bir uygunluk fonksiyonu belirler. Yapıtaşı varsayımı gereği, güçlü bireyler iyi genlere sahiptir. Bu genlerin diğer nesillere geçme olasılığı seçim baskısı ile artırılarak, kötü genlere sahip güçsüz bireyler evrim esnasında kaybolması sağlanır. Evrim süreci belirlenen bitiş koşulunu sağlayıncaya kadar devam eder ve en son nüfustaki en iyi birey, ulaşılan en iyi çözümü temsil eder. Genetik Algoritma aşağıdaki ana bileşenlerden oluşur:

- Temsil
- Üreme için Seçilim
- Melezleme (Üreme)
- Mutasyon
- Uygunluk Değeri Hesaplama
- Sonlandırma

2 DENEY TASARIM METOTLARI

2.1 Giriş

Deney tasarım metotları, en az deney ile en fazla sonuç elde etmek için değişkenlerin deney kombinasyonlarını düzenleyen metotları içermektedirler. Literatürde tam faktöryel, alt faktöryel, Taguchi, merkezi kompozit, Box-Henken gibi çeşitli deney tasarım yöntemleri kullanılmaktadır. Deney tasarım yöntemleri düzenlediği deney sayısına ve etkili sonuçlar çıkarma kabiliyetleri açısından birbirinden farklıdır. Deney tasarımı elde var olan opsiyonların seçimi ile ilgilidir. Elde var olan ve çalışan bir modelin karar değişkenleri arasında en iyiyi seçmek için kullanılır.

Mühendislik optimizasyon tekniklerinin gelişimiyle, mekanik yapıların tasarımında otomatik tasarım ve maliyetler azaltılmıştır. Bu metotlarda amaç, kısıtlar sağlanırken maliyet fonksiyonunu en aza indiren karar değişkenlerini bulmaktır. Genel olarak, yapısal optimizasyon, ağırlıkla ilgili yapısal yanıtlarla ilgili bir tasarım kriteri tanımlar. Bunlara örnek olarak yerdeğiştirmeler, gerilmeler ve/veya frekanslar verilebilir [Arora,1992]. Özel bir formülasyonda, ağırlık ve geri kalan yapısal yanıtlar tasarım kriteri olarak tanımlanırken, yapısal bir yanıt performans endeksi olabilir.

Genel olarak, deney tasarım metotları genel tasarımda tanımlanan kısıtlamaları içermez. Böylece, amaç fonksiyonu ve kısıtlamaları içeren yeni bir karakteristik tanımlanır. Genel deney tasarımı kısıtlı problemlere yeni karakteristik fonksiyonu kullanarak genişletilir. Karakteristik fonksiyon, kısıtlamalardan oluşan amaç fonksiyonu ve ceza fonksiyonunun toplamıdır. Ceza fonksiyonu herhangi bir kısıtlama ihlal edilirse bir değere sahiptir. Eğer kısıtlama ihlali yok ise ceza fonksiyonunun değeri sıfırdır. Ortogonal dizinlerin büyüklüğü, karar değişkenlerinin sayısına bağlıdır. Başlangıç değerleri veritabanından seçilir. Düzeylerin sayısı o anki, bir üst ve bir alt değer olmak üzere üç tanedir ve sabittir. Yeni değişkenlerin seçimi genel Deney Tasarımı ile aynıdır. Karakteristiklerle deney tasarımının sonucu analiz

edilerek yeni deęişkenler seçilir. Sonuç yakınsayana kadar işlem devam eder. Yaklaşım kriteri de ayrıca tanımlanır.

2.2 Ortogonal Dizinler Kullanarak Deney Tasarımı

2.2.1 Ortogonal Dikey Dizinlerin Oluşturulması

Deney Tasarımı için yapılan matris deneyleri, deneysel setlerden oluşur. Karar deęişkenlerinin seviyeleri karakteristikleri elde etmek için tanımlanır. Bütün deneyler yapıldıktan sonra optimum düzeyleri belirlemek için Ortalamaların Analizi (Analysis of Means-ANOM) gerçekleştirilir. [Peace, 1995] Ardışık dizinlerle oluşturulan deney tasarımı karar deęişkenlerinin süresiz deęerlerini kullandığından süresiz tasarım kolayca yürütülebilir.

Ortogonal dizinler düzgün olarak seçildiğinde minimum deney sayısı tam faktöriyel deneyin yerine geçecektir. Eğer karar deęişkenleri arasındaki ilişki güçlü ise, bu ilişki ortogonal dizinin en küçük boyutunu seçerken dikkate alınmalıdır. Bununla beraber, yapısal tasarımda bu ilişkileri ihtiva etmek kolay deęildir.

Etkileşimsiz veya orta derecede etkileşimli süresiz tasarımda Taylor serisi metodu'ndan yararlanılır. Taylor Serisi Metodu'nda, karar deęişkenlerinin deęerlerinin deęiştirilmesi perturbasyonları, o anki tasarımdan tek seferde ileri ve geri yapılır. Bu tip problemler için Taylor serisi metodu geliştirilen metottan daha üstün olabilir. Bununla beraber, güçlü etkileşimli problemlerde, Taylor serisi metodu o anda bulunan tasarım noktasına baęlıdır. Bu da gerçek optimumdan çok uzakta bir çözüme yol açabilir. Bunun aksine, ortogonal dizinlerde kullanılan metot, her bir karar deęişkeninin ortalama etkisini kullanarak optimum seviyeyi belirlediğinden gerçek optimuma daha yakın bir çözüm üretir.

Ortalamaların analizi kullanarak karar deęişkenlerinin optimal seviyelerini bulma işlemi aşağıdaki şekilde gerçekleşir. [Phadke, 1989]

A, B, C ve D olmak üzere dört karar değişkenimiz olduğunu farz edelim. Her bir karar değişkenimizin üç düzeyi vardır. Üç düzeyli ve dört karar değişkenli bir problem için karar değişkenleri arasındaki etkileşim ihmal edilirse, minimum ortogonal dizin Çizelge 2.1’de gösterilen $L_9(3^4)$ ortogonal dizini seçilir.

$L_9(3^4)$ ortogonal dikey dizininde:

9 rakamı deney sayısını,

3 rakamı kullanılan düzeylerin sayısını ve

4 rakamı da karar değişkenlerinin sayısını temsil etmektedir.

Çizelge 2.1 $L_9(3^4)$ Ortogonal Dikey Dizini

Deney no	Sütun sayısı ve atanan faktör				Sonuçlar (η)
	1 A	2 B	3 C	4 D	
1	1	1	1	1	η_1
2	1	2	2	2	η_2
3	1	3	3	3	η_3
4	2	1	2	3	η_4
5	2	2	3	1	η_5
6	2	3	1	2	η_6
7	3	1	3	2	η_7
8	3	2	1	3	η_8
9	3	3	2	1	η_9

Çizelge 2.1’de satırlardaki η_1 ’den η_9 ’a kadar olan karakteristikler deneysel olarak veya hesaplamalarla elde edilir. Yapısal optimizasyon için, her bir satır bir sonlu elemanlar hesaplamasına ihtiyaç duyar. Ortalamaların analizi bu değerlerle yürütülür. Karar değişkeninin ilk seviyesi olan A(A_1) için $A_1(M_{A1})$ ana sonuç aşağıdaki şekilde hesaplanır:

$$M_{A1}=m_{A1}-m \quad (1.1.)$$

$$m_{A1}=(\eta_1+\eta_2+\eta_3)/3 \quad (1.2.)$$

$$m = \frac{1}{9} \sum_{i=1}^9 \eta_i \quad (1.3.)$$

Burada :

m : Bütün karakteristiklerin ortalaması

m_{A1} : A_1 etkisini içinde barındıran karakteristiklerin ortalaması

Değerlerini ifade etmektedir. Bir karar değişkeni için her bir seviyeye karşılık gelen karakteristiklerin ortalaması Çizelge 2.2’de gösterilmektedir.

Çizelge 2.2 Her Düzeye Karşılık Gelen η ortalamaları

Karar Değişkenleri	Düzeyler		
	1	2	3
A	$m_{A1}=(\eta_1+\eta_2+\eta_3)/3$	$m_{A2}=(\eta_4+\eta_5+\eta_6)/3$	$m_{A3}=(\eta_7+\eta_8+\eta_9)/3$
B	$m_{B1}=(\eta_1+\eta_4+\eta_7)/3$	$m_{B2}=(\eta_2+\eta_5+\eta_8)/3$	$m_{B3}=(\eta_3+\eta_6+\eta_9)/3$
C	$m_{C1}=(\eta_1+\eta_6+\eta_8)/3$	$m_{C2}=(\eta_2+\eta_4+\eta_9)/3$	$m_{C3}=(\eta_3+\eta_5+\eta_7)/3$
D	$m_{D1}=(\eta_1+\eta_5+\eta_9)/3$	$m_{D2}=(\eta_2+\eta_6+\eta_7)/3$	$m_{D3}=(\eta_3+\eta_4+\eta_8)/3$

Çizelge 2.2 gösteriyor ki, her bir karar değişkeninin optimal seviyesi, en küçük ortalamalı seviyedir. Bu işleme, “Ortalamaların Analizi (ANOM, Analysis of Means)” adı verilir.

3 ARDIŞIK ORTOGONAL DİZİNLER KULLANARAK KESİKLİ TASARIM

3.1 Giriş

Standart Taguchi optimizasyon yöntemi faktörlerin etkileşimini ihmal ederek optimum parametre değerlerini tahmin etmektedir. Hâlbuki yapısal optimizasyon problemlerinde yapının cevabı genelde çok nonlineerdir, karar değişkenlerinin etkileşimi kuvvetlidir ve ihmal edilmeyecek mertebededir. Bu nedenle, standart Taguchi optimizasyon algoritması yapısal optimizasyon problemleri için etkileşimleri de dikkate alacak şekilde geliştirilmiştir. Geliştirilen Taguchi optimizasyon algoritması, ardışık optimizasyon yapmaktadır ve her bir iterasyonda ortalamaların analizini kullanarak standart Taguchi optimizasyonunu kullanmaktadır.

Deney Tasarımı kullanılarak yapılan Taguchi yaklaşımında faktörler, performansı etkileyen karar değişkenleri ve kontrol edilebilen girdilerdir ve arzulanan performans üzerinde etkisini ve kontrolünü belirlemek amacıyla kullanılırlar. Çoklu seviyenin anlamı ise her bir parametrenin birden fazla değerden oluşmasıdır şeklinde özetlenebilir.

Eklerde verilen standart Taguchi ortogonal dizin tabloları aşağıdaki şekilde özetlenebilir:

- 2 düzeyli (kısmi faktöriyel) dizinler
 $L_4(2^3)$, $L_8(2^7)$, $L_{16}(2^{15})$, $L_{32}(2^{31})$, $L_{64}(2^{63})$
- 2 düzeyli dizinler
 $L_{12}(2^{11})$
- 3 düzeyli dizinler
 $L_9(3^4)$, $L_{27}(3^{13})$, $L_{81}(3^{40})$
- 4 düzeyli dizinler
 $L_{16}(4^5)$, $L_{64}(4^{21})$
- 5 düzeyli dizinler

$$L_{25}(5^6)$$

- Karışık düzeyli dizinler

$$L_{18}(2^1 \times 3^7), L_{32}(2^1 \times 4^9), L_{50}(2^1 \times 5^{11})$$

$$L_{36}(2^{11} \times 3^{12}), L_{36}(2^3 \times 3^{13}), L_{54}(2^1 \times 3^{25})$$

Yukarıda belirtilen Taguchi dizin tabloları, her birinin 2,3 veya 4 düzeyi olan 2 ile 63 arasında karar değişkenine sahip sistemlerde kullanılabilir.

Bu bölümde, Bölüm 2’de anlatılan ANOM konseptini de kullanan ardışık Taguchi optimizasyon algoritması detaylı bir şekilde anlatılmaktadır.

Fortran95 kodu yazılarak geliştirilen algorithmada ortalamaların analizi, her bir iterasyonda kullanılmıştır. Algoritmanın genel akış şeması Şekil 3.2’de gösterilmiştir.

3.2 Algoritma Adımları

Geliştirilen algoritmanın aşamaları aşağıdaki şekildedir.

1.ci adım: Problemin Tanımı:

Bölüm 1’de anlatılan optimizasyon probleminin üç temel elemanı:

- Belirlenecek karar değişkenleri
- Optimum kılınacak amaç (hedef) fonksiyonu
- İçinde bulunulan kısıtlar

Matematiksel olarak modellenir.

Deney tasarımındaki faktörler, optimizasyondaki karar değişkenlerine eşdeğerdir. Deney tasarımındaki düzeyler ve karakteristik fonksiyon, optimizasyonda sırası ile kesikli değerler ve amaç fonksiyon olarak addedilir. Her bir tasarım

değişkeninin aday değerleri kesikli tasarım için verilmelidir. Bu değerler matris deneyinde düzeyler olarak düşünülür.

2.ci Adım: Ardışık dizinin seçimi:

Deney tasarımında, genelde iki veya üç düzeyli ardışık dizinler kullanılır. İki düzeyli ardışık dizinler karakteristiklerin faktörlere olan doğrusal etkilerini incelerler. Bunun aksine, üç düzeyli ardışık dizinler, karakteristiklerin ikinci dereceden etkilerini incelerler. Bu araştırmada, iterasyon sürecinde yeni bir tasarım bulmak için üç düzeyli deneyler benimsenmiştir. Böylece, minimum, ardışık dizin karar değişkenlerinin sayısına göre seçilir. Minimum ardışık dizin hesaplama zamanından tasarruf etmek için kullanılmıştır. Aynı zamanda minimum ardışık dizin, bütün karar değişkenlerinin kendi sütunlarına atanabildiği en küçük değerdir.

3.cü adım: Karar değişkeni düzeylerinin dizilmesi:

Bu aşamada, karar değişkenlerinin kesikli değerleri ikinci aşamada seçilen ardışık dizinin sütunlarına atanır. Bu atama işlemi keyfi olarak gerçekleştirilir. Çünkü, etkileşimler göz ardı edilmektedir. Karar değişkenlerinin aday değerleri tasarım süreci uygulanmadan verilmelidir. İlk tasarımda, isteğe bağlı bir kesikli değer seçilebilir. Bu seçilen değer ikinci düzeye atanır. Bu seçilen değere komşu değerlerden bir büyük olan kesikli değer ilk düzeye atanırken, bir küçük olan değer üçüncü düzeye atanır. Matris deneyinde, büyük olasılıkla düşünülmesi gereken düzey birinci düzeydir. Yapısal tasarımda güvenilirlik çok önemlidir. Bu sebeple, daha büyük olan değerler ilk düzeylere atanır. Eğer ilk tasarımdaki adaylar en küçük veya en büyük değerler ise, birbiri ardına gelen iki artan veya azalan kesikli değer, düzey olarak seçilir. Bu durumda, ilk değer ilk düzeye veya üçüncü düzeye atanır. Karar değişkenlerinin düzeyleri Şekil 3.1’de gösterildiği gibi kararlaştırılmıştır. Şekil 3.2 düzeyler için seçme işleminin nasıl olacağını bir durum üzerinde göstermektedir. Bu şekilde karar değişkenlerinin sayısı üçtür ve her bir değişken beş aday değere sahiptir. İlk başlangıç değerleri A_3 , B_2 , ve C_4 olarak alınmıştır.

4.cü adım: Matris deneyi:

Ardışık dizinin her bir satırı için karakteristik fonksiyon hesaplanır.

Optimizasyon formülasyonu kullanılarak her bir iterasyon için matris deneyi aşağıdaki şekilde tanımlanır.

$$\text{Bul} \quad x \quad (3.1.)$$

$$\text{En küçük} \quad R(x) \quad (3.2.)$$

$$\text{Kısıtlar} \quad g_j(x) \leq 0, j = 1, l \quad (3.3.)$$

Burada :

x : Karar değişkeni vektörü

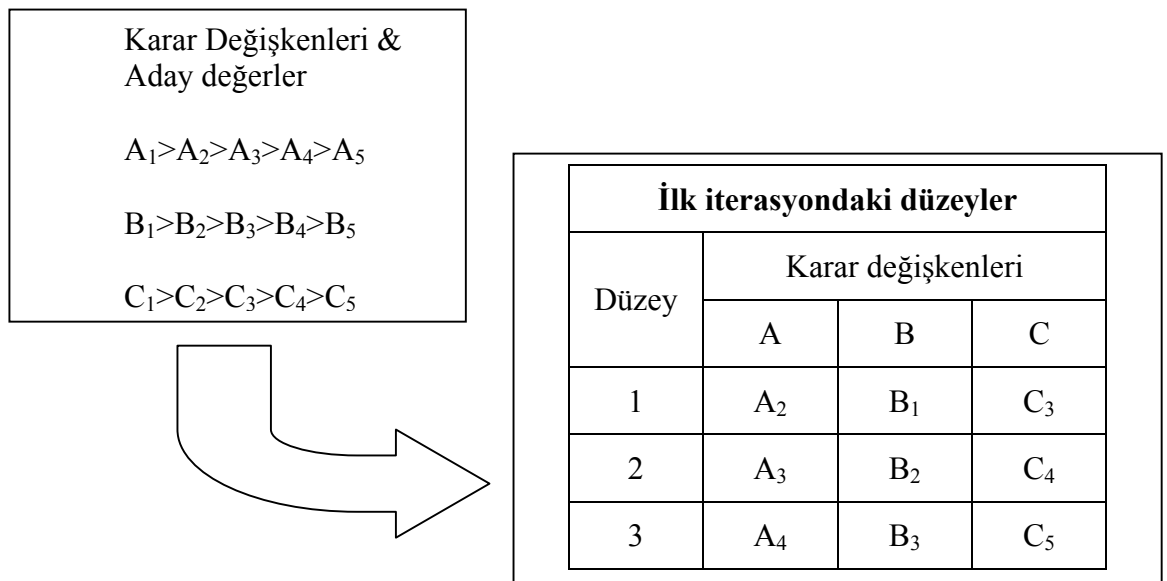
$R(x)$: Amaç fonksiyonu

$g_j(x)$: kısıt fonksiyonu

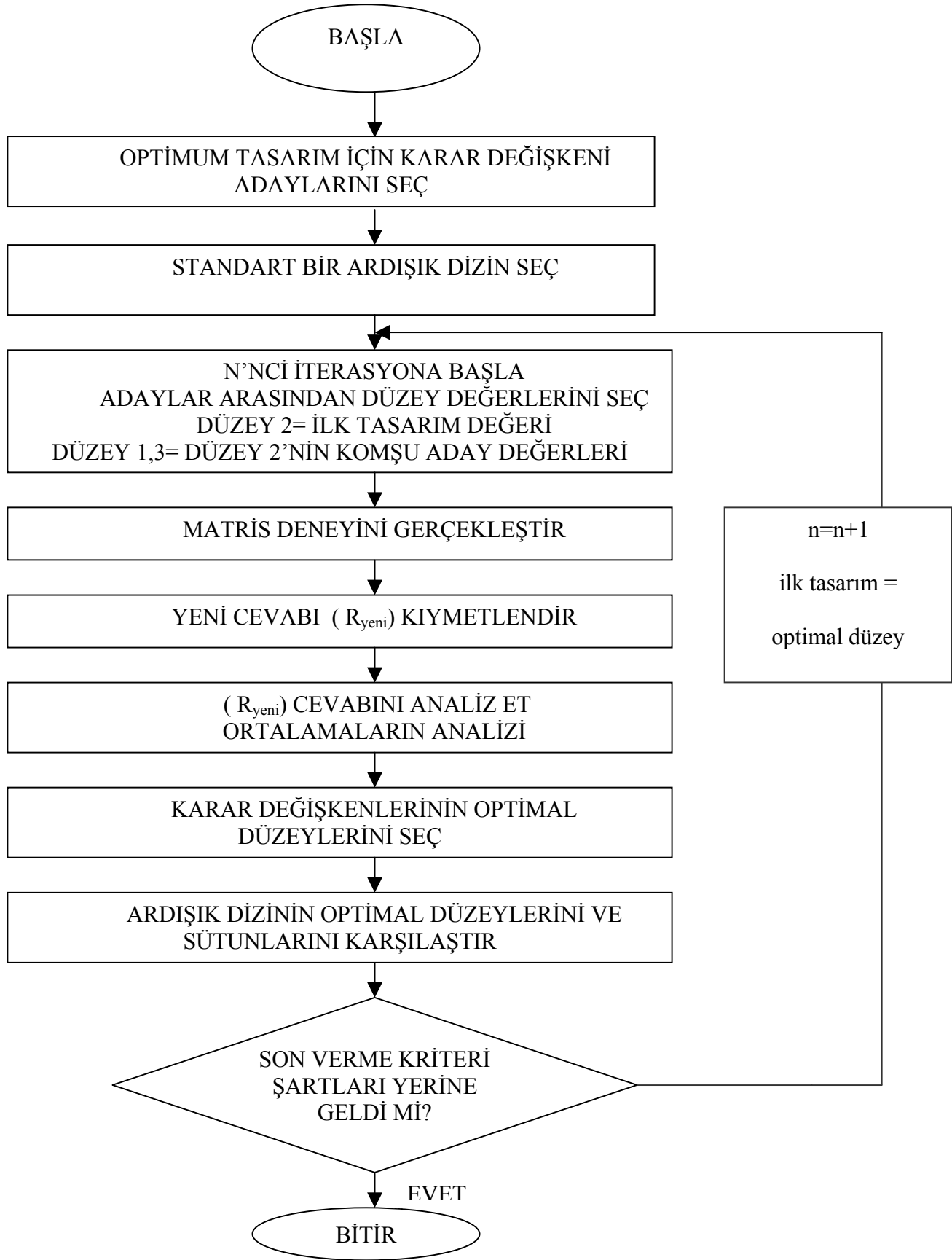
l : kısıtların sayısı

Değerlerini ifade eder.

Geleneksel deney tasarımı metotları, tasarım optimizasyonunda tanımlanan kısıtları dikkate almazlar. Bu sebeple R_{yeni} karakteristik fonksiyonu adı altında yeni bir tepki tanımlanarak kısıt ihlalleri (3.4) ve (3.5) denklemlerinde gösterildiği şekilde ihtiva edilir: [Lee ve Park, 1996]



Şekil 3.1 Düzey değerlerinin seçilmesine bir örnek



Şekil 3.2 Ardışık dizinler kullanan iteratif optimizasyon algoritması

$$R_{yeni} = R + P(x) \quad (3.4.)$$

$$P(x) = s \times \sum_{i=1}^m \max[0, v_i] \quad (3.5.)$$

Burada :
 $P(x)$: Ceza fonksiyonu
 v_i : i'nci kısıtın maksimum ihlali
 s : ölçek faktörü
 Değerlerini ifade eder.

Ölçek faktörü kısıt ihlalini vurgulamak için tanzim edilmiştir. Aşırı fazla bir ölçek faktörü, amaç fonksiyonunun etkisinin ihmal edilmesine yol açar. Bunun tersine, küçük ölçek faktörlü bir ceza fonksiyonu uygun olmayan tasarıma yol açar. Bu çalışmada kullanılan ölçek faktörü öyle bir değere atanmıştır ki, ceza fonksiyonunun mertebesi, orijinal amaç fonksiyonunun (R) bir merteye üstüdür.

5.ci adım: Yeni tasarımın tespiti:

Karakteristik fonksiyon değerlerini kullanarak ortalamaların analizi ikinci bölümde anlatıldığı gibi icra edilir. Bu şekilde karar değişkenlerinin yeni düzeyleri oluşturulur. Bu yeni oluşturulan düzeylerle bir doğrulama analizi yapılmalıdır. Bunun sebebi bu, tasarım fizibilitesini veya toplamsallığının istatistiksel geçerliliğini her zaman garantilemez. Karakteristik fonksiyonun tanımından dolayı kısıt fizibilitesi de her zaman garantili değildir. Ayrıca karar değişkenleri arasında bir etkileşim varsa, bu etkileşim istenmeyen yeni düzeylerin oluşmasına yol açabilir. Böylece, ortalamaların analizinden elde edilen yeni düzeyler matris deneylerinden elde edilen en iyi kombinasyon ile karşılaştırılır ve iterasyondaki en iyi düzeyler yeni düzeyler olarak seçilir. Yapısal analizde her bir deney sonlu elemanlar analizi gerektirir.

Bu çalışmada yeni cevap kıymetlendirilirken ortalamaların analizinin yanında ardışık dizinin en iyi değeri ve bir önceki iterasyonda elde edilen en iyi değer

saklanmış, bunların karşılaştırılması yapılarak yeni iterasyona üç cevabın en iyi değeri ithal edilmiştir.

Böylece literatürde kütle iterasyon grafiğindeki son iterasyonlarda oluşan salınımlar ortadan kaldırılmış ve beşinci bölümde sunulduğu gibi kütlede ve iterasyonda iyileştirmeler elde edilmiştir.

6.cı adım: Yakınsama kriteri:

Algoritmadaki yakınsama kriteri 0,01 olarak belirlenmiştir.

Yeni düzeylerdeki kabul edilemez ardıl sonuçların iterasyon sayısı beşten fazla ise veya maksimum iterasyon sayısı 100'ü aştığında ya da yaklaşım kriteri 0,01 sayısına ulaştığında algoritma son bulur:

Yakınsamadaki salınımları engellemek için beşinci adımda bahsedilen ortalamaların analizinin optimum değerinin yanında ardışık dizinin ve bir önceki iterasyonun en iyi değerleri de yeni iterasyona taşınarak bunların arasından en iyi değer o iterasyonun optimum değeri olarak kabul edilmektedir. Böylece literatürde [Lee at al, 2003] ortaya çıkan kütle değişim grafiğindeki salınımlardan kurtulunmuştur.

Yukarıda yazılan yaklaşım kriterlerinin hiçbiri sağlanmazsa tasarım süreci üçüncü adıma geri döner. Bu adımda, karar değişkenlerinin ikinci düzeyleri daha önceki en uygun düzeylerle değiştirilir.

4 FORTRAN 95

4.1 Fortran95 Kullanma Sebebi

Bu kodun oluşturulmasında FORTRAN kodunun ana kullanım sebebi LS-DYNA uygulamalarında kullanılacak olması ve dolayısı ile UNIX işletim sistemleriyle entegrasyonudur. Aynı zamanda paralel işlemlere imkân vermesi dolayısıyla kodun aynı anda birçok işlemci ile çalıştırılması hesap maliyetini azaltan bir etken olarak karşımıza çıkmaktadır.

Bu çalışmada oluşturulan kod, tek işlemcili Intel Pentium4 CPU 1 GHz DDR RAM özelliklerine sahip makinede uygulanmıştır. Çok amaçlı yapısal optimizasyon ve çokdisiplinli uygulamalar düşünüldüğünde yapısal optimizasyon kodunun paralel işlemcilerde çalıştırılmasının maliyeti düşüreceği ve sonuca daha çabuk yansıyacağı ortadadır.

4.2 Fortran'ın Tarihsel Gelişimi

1954'de IBM tarafından üretilen IBM 704 için ilk sürümü John Backus ve ekibi tarafından geliştirilmiştir. Backus ve ekibi Kasım 1954'de "The IBM Mathematical **FOR**mula **TRAN**slating System: Fortran" isimli raporu yayınlamışlardır. Fortran ilk yüksek düzey programlama dili olmasa da 1950'deki yüksek programlama dilleri derlenmeden, bir çevirici yardımıyla çalıştırılıyordu. Bu da makine koduyla yazılan programlardan en az 10 kat daha yavaş çalışmalarına sebep oluyordu. 1950'lerdeki bilgisayarlar için hız çok şey ifade ettiğinden yazması zor da olsa makine kodu bu yüzden hala popülerdi. İşte bu noktada Backus ve ekibi hem yüksek programlama dilleri gibi kolay yazılabilen hem de makine kodunda yazılmış gibi hızlı çalışan bir programlama dili sözüyle Fortran 'ı tanıttılar.

Fortran 'ı diğer yüksek düzey programlama dillerinden ayıran bir çevirici yerine bir derleyici kullanmasıydı. Program yüksek düzey dilde yazıldıktan sonra

makine koduna çevriliyor ve böylece hız kaybı engelleniyordu. Her ne kadar ilk derlenebilir yüksek düzey dilin Fortran olup olmadığı hala tartışma konusu olsa da, Fortran geniş kitleler tarafından kullanılmış ilk yüksek düzey derlenebilir dildir.

4.3 Sürümleri

Fortran 0: İlk Fortran sürümünde hız için esneklikten vazgeçildi. Bu yüzden ilk sürümde program içindeki tüm değişkenlerin önceden tanımlanması gerekiyordu.

Fortran I: Ocak 1955'de başlayan çalışmalar Nisan 1957 bitirilerek Fortran I tanıtıldı. İlk sürümden farklı olarak değişken isimleri 6 karaktere kadar çıkabiliyordu, veri giriş çıkış sisteminin düzeni değiştirilmişti. Ayrıca if ve do ifadeleri içeriyordu. Daha önceleri tekrarlama işlemi için özyineleme(recursion) kullanılırken do ifadesi sayesinde fonksiyon içinde yineleme(iteration) işlemi gerçekleştirildi. Fortan I 'deki değişken isimleri I,J,K,L,M,N ile başlıyordu. Bunun sebebi de o zamanki bilgisayarların bilimsel çalışmalarında kullanıldığından ve bilim insanlarının denklemlerinde değişken olarak bu harfleri çok kullanmalarından kaynaklanıyordu.

Fortran II: 1958 baharında tanıtılan Fortran II, Fortran I 'deki birçok hatayı düzeltiyordu. Getirdiği en önemli yenilik alt-programların ayrı ayrı derlenebilmesini sağlamaktı. Böylece alt-programda yapılan küçük bir değişim için tüm programı yeniden derlemek yerine, sadece alt-programın derlenmesi sağlandı. Bu zor olan ve genellikle makine hatası sonucu yarıda kalan derleme işlemine büyük bir kolaylık sağladı.

Fortran III : Fortan III tasarlanmasına rağmen asla geniş bir kullanım alanı bulamadı.

Fortran IV: 1960 ve 1962 yıllarında gelişmiş olan Fortran IV zamanın en geniş alanda kullanılmış programlama dili oldu. 1966'da Fortran 66 adı altında standart haline geldi(ANSI, 1966). Fortran IV büyük bakıma Fortran II 'nin gelişmiş

haliydi. En önemli değişikliği mantıksal if ifadesi ve fonksiyonlara başka fonksiyonların parametre olarak aktarılabilmesiydi.

Fortran 77: Fortran IV 'e ek olarak string işleme, mantıksal döngüler ve if ifadesine else ifadesi eklenmişti.

Fortran 90: Fortran 77 'den epey farklı özellikleri vardır. En belirgin özelliği diziler için dinamik olarak yer ayrımının yapılabilmesidir. Ayrıca Case, Exit, Cycle, gibi yeni ifadeler eklenmiştir. Bir modül ünitesi eklenmiştir. Bu modüller Private veya Public ulaşım iznine sahip veri ve fonksiyon saklayabilmektedir. Ayrıca Fortran 77 'de olan bazı özelliklerde yeni sürümden çıkarılmıştır. Fortran 90 ile birlikte FORTRAN olan yazım şekli Fortran olarak değiştirilmiştir.

Fortran 95: Bu çalışmada da kullanılan sürüm olan Fortran95'in en belirgin özelliği dinamik hafıza atamaya (Dynamic Memory Allocation) izin vermesidir. Bu sayede, ihtiyaç olan yerlerde hafızaya yer açmak ve daha sonra bunu silmek mümkün olmaktadır. Bu amaçla kullanılan komutlar “Allocate ve Deallocate” olarak verilmektedir.

4.4 Fortran95 Özel Komutları

Bu kodun oluşturulmasında kullanılan en önemli komutlar aşağıda verilmiştir:

- Subroutine
- Allocate / Deallocate

Bu komutlara ilişkin kısa açıklamalar ve kullanım şekilleri şu şekildedir:

SUBROUTINE alt programları FUNCTION alt programlarına göre daha genel amaçlıdır. FUNCTION alt programı ana programa sadece bir bilgi geri gönderebilir. SUBROUTINE alt programı için böyle bir sınırlama yoktur. SUBROUTINE alt

programı argümansız olarak kullanılabilir. Bu şekilde kullanıldığı zaman amaç programın akışını daha güzel ifade eden modüler bir program yapısı sağlamaktadır. FUNCTION alt programının en az bir argümanı olması zorunludur.

FUNCTION ve SUBROUTINE alt programları arasındaki başka bir farklılık da, ana programdan çağrılış biçimidir. Fakat alt program, bir deyimle bir algoritmayı gerçekleştirecek biçimde tanımlanırsa, her iki tür alt programın özde çok benzer oldukları söylenebilir. SUBROUTINE alt programını yapısı aşağıda verilmiştir:

SUBROUTINE alt program adı (argüman Listesi)

.....

.....

Fortran deyimleri

.....

RETURN

END alt program adı

Ana programdan SUBROUTINE alt programına geçiş, CALL deyimini ile sağlanır. Bu deyim yapısı da şöyledir:

CALL alt program adı (argüman listesi)

Alt program adı programcı tarafından değişkenlere ad verme kurallarına uygun olarak kararlaştırılır. SUBROUTINE alt programında tür söz konusu olmadığı için adın hangi harfle başladığı önemli değildir. Alt programın birinci deyiminde yer alan argüman listesi ile, ilişkili CALL deyiminin argüman listesi, argüman sayısı ve türü açısından uyumlu olmak zorundadır. Başka bir deyişle, adları farklı olsa da her iki listede aynı sırada buluna argümanlar aynı türde olmalıdır. Gerek ana programdan (çağırıcı programdan) alt programa aktarılan bilgiler gerekse alt programdan ana programa aktarılan bilgiler argümanlar

aracılığıyla aktarılır. Bu bilgiler her türde her türde skaler değerler olabileceği gibi diziler de olabilir. Eğer argüman diziyi simgeliyorsa, büyüklüğü hem ana programda hem de alt programda bildirilmek zorundadır. Bu deyim işleme konulduğunda argümanlar alt programda son alındıkları değerleri taşırlar.

Aşağıda bir SUBROUTINE alt programı ve bu alt programı ana programdan çağıran deyim verilmiştir.

```

.....SUBROUTINE R ( A, B, X, Y )
..... X = A * B
.....Y = 2.0 * ( A+ B )
.....RETURN
.....END

```

Alt programın adı R, argümanları A, B, X ve Y'dir. Görüldüğü gibi CALL deyiminden A, B, AL ve CEV değişkenlerinden oluşan argüman listesi ile uyumludur. Alt programa geçildiği zaman ana programda A, B, AL ve CEV değişkenlerinin sahip olduğu değerler alt programdaki A, B, AL ve CEV değişkenlerine verilir. RETURN deyimi işleme konulduğunda bu kez A, B, X ve Y'nin alt programda aldıkları son değerler, ana programda A, B, AL ve CEV değişkenlerine verilmiş olur. Görüldüğü gibi yukarıdaki örnekte A ve B argümanları ana programdan alt programa, AL ve CEV (veya X ve Y) argümanları da alt programdan ana programa bilgi aktarmaya aracı olmaktadır. Argümanların ana be alt programlarda aynı adı taşıması veya taşımamamsının işlem üzerimde hiçbir etkisi olmadığı da vurgulanmalıdır.

Fortran77 statik hafıza atama işlemlerine izin verirken, Fortran90 ve üstü sürümler dinamik hafıza atama işlemine izin vermiştir.

Statik hafıza atama:

```
integer,dimension(10) :: k
real,dimension(10,10) :: x
real,dimension(100) :: x1
real(kind=rk),dimension(4,4,4) :: y
```

Dinamik hafıza atama:

```
real(kind=rk),dimension(:),allocatable :: x
real(kind=rk),dimension(:,,:),allocatable :: y
integer :: n,err
```

```
...
```

```
print *, 'Give array size.'
read *, n
allocate(x(n),y(0:n,0:n),stat=err)
if (err/=0) then
  print *, "Could not allocate."
  stop
end if
```

```
...
```

```
x=1.0
y=2.0
```

```
...
```

```
deallocate(x,y)
```

5 ÖRNEK TEST PROBLEMLERİ

Bu çalışmada geliştirilen algoritmanın geçerliliğini göstermek için literatürde de kullanılan kiriş elemanlar ve darbe emici test problemleri kullanılmıştır. Örnekler iki ve on çubuklu kiriş gibi standart problemleri de ele almıştır. Bu örneklerle ait Fortran95 kodlarının oluşturulması ve literatürdeki sonuçlarla karşılaştırılması yapılarak Fortran kodunun geçerliliği tartışılmıştır. En son olarak da doğrusal olmayan darbe emici modellenerek algoritması ve Fortran kodu oluşturulmuştur.

5.1 İki Çubuklu Kiriş

Bu ilk optimize edilecek tasarımda geliştirilen algoritmaya dayalı Fortran95 kodunun sağlamlığını göstermek ve karşılaştırmasını yapmak için iki çubuklu optimizasyon problemi ele alınmış ve bu sonuçlar literatürde [Roux, W.J, 1997] verilen analitik sonuçlarla karşılaştırılmıştır. Şekil 5.1’de gösterilen kiriş yapısının karar değişkenleri

- b1 : Çubukların kesit alanı,
- b2 : Mesnetler arası orta mesafesi

şeklinde parametrelendirilmiştir. Problem statik olarak karardır. 199,94 kN’luk bir kuvvetin etkisinde olan iki çubuklu kiriş probleminin ana amacı, verilen yükleme şartında, b1 ve b2 değerlerinin optimal değerlerini bulmaktır. Geliştirilen Fortran95 programı ile bulunan b1 ve b2 karar değişkenlerini gösterir grafikler sırası ile Şekil 5.2 ve Şekil 5.3’de verilmiştir.

Kirişin ağırlığını enküçükleyen σ_1 ve σ_2 gerilme değerlerini her bir çubukta 100 MPa olarak sınırlandıran b1 ve b2 karar değişkenlerinin en iyi değerlerini bulmak için tanımlanan optimizasyon problemi, boyutsuzlaştırıldıktan sonra matematiksel olarak aşağıdaki şekilde standart formatta tanımlanabilir:

Bul : b_1 ve b_2 karar değişkenleri

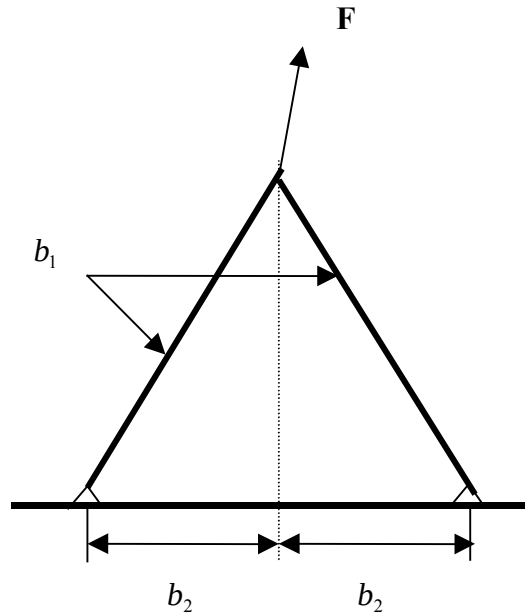
Enküçükle :

$$\text{Ağırlık} = C_1 b_1 \sqrt{1 + b_2^2} \quad (5.1)$$

Gerilme kısıtları :

$$\sigma_1 = C_2 \sqrt{1 + b_2^2} \left(\frac{8}{b_1} + \frac{1}{b_1 b_2} \right) \leq 1 \quad (5.2)$$

$$\sigma_2 = C_2 \sqrt{1 + b_2^2} \left(\frac{8}{b_1} - \frac{1}{b_1 b_2} \right) \leq 1 \quad (5.3)$$



Şekil 5.1 Optimizasyon tasarımı yapılacak iki çubuklu kiriş

Tasarım alanı :

$$0.2 \text{ cm}^2 \leq b_1 \leq 4.0 \text{ cm}^2 \quad (5.4)$$

$$0.1 \text{ m} \leq b_2 \leq 1.6 \text{ m} \quad (5.5)$$

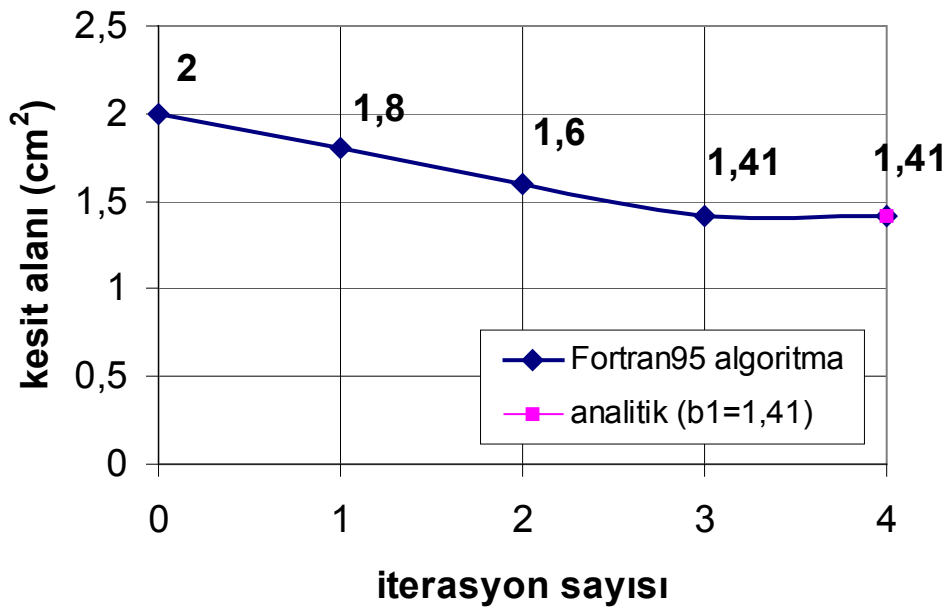
Tasarım alanında b_1 için aday değerleri:

$$b_1=[0.2,0.5,0.7,0.9,1.0,1.3,1.41,1.6,1.8,2.0,2.5,3.0,3.5,4.0]$$

Tasarım alanında b_2 için aday değerleri:

$$b_2=[0.1,0.2,0.38,0.5,0.8,0.9,1.1,1.3,1.5,1.6,0.0,0.0,0.0,0.0]$$

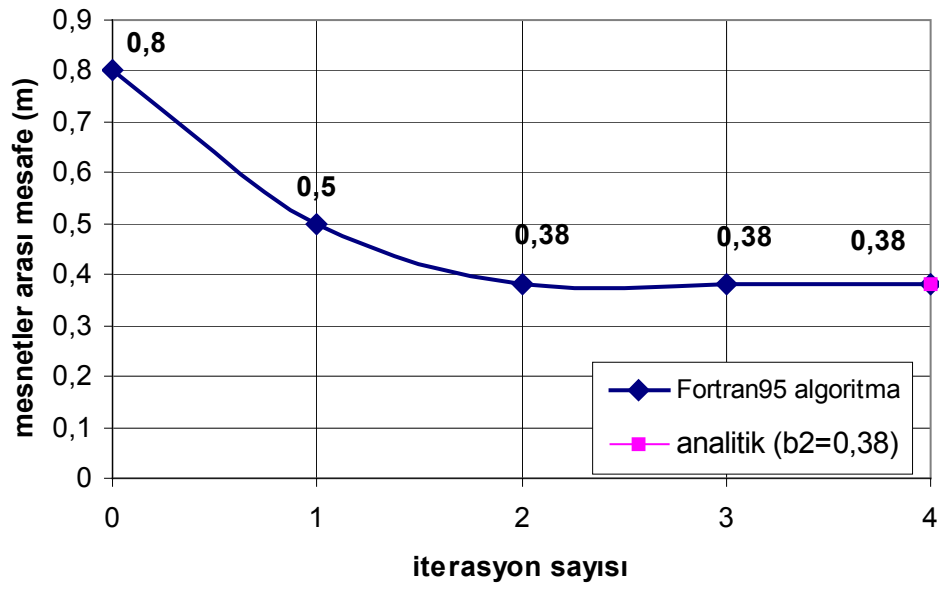
olarak seçilmiştir. Problemin analitik sonuçları da, bu değerlere yakınsayıp yakınsamadığını kontrol etmek için aday değerlere ilave edilmiştir.



Şekil 5.2 İki çubuklu kiriş için çubuk kesit alanlarının değişimi

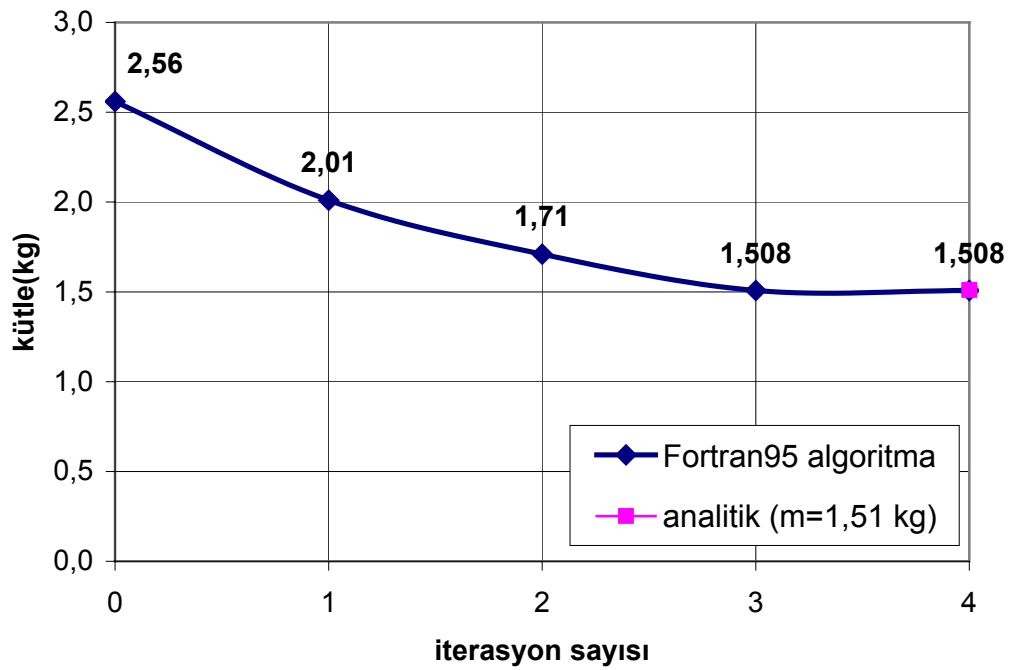
Bu optimizasyon probleminin çubuk kesit alanları ve mesnetler arası mesafe için analitik sonuçları sırasıyla $b_1 = 1.41 \text{ cm}^2$, $b_2 = 0.38$ metredir. Bu veriler sonucunda analitik ağırlık = 1.51 kg. ve $\sigma_1 = 1.0$ şeklinde bulunur. Burada $C_1 = 1.0$ ve $C_2 = 0.124$ olarak alınmıştır.

σ_1 , σ_2 'den her zaman büyük olmasından dolayı sadece σ_1 gerilme kısıtı dikkate alınmıştır.



Şekil 5.3 İki çubuklu kiriş için mesnetler arası mesafenin değişimi

Geliştirilen Fortran95 programı ile bulunan optimal kütle ve kütlenin iterasyonlarla değişim grafiği Şekil 5.4’de verilmiştir.



Şekil 5.4 İki çubuklu kiriş için kütle değişim grafiği

Şekil 5.4’de görüldüğü gibi 4 iterasyonda toplam 40 analiz ile analitik değere yakınsama oluşmuştur. Amaç fonksiyonundaki toplam iyileştirme oranı başlangıç değerine göre %41,1 olarak gerçekleşmiştir.

Çizelge 5.1 iki çubuklu kirişte b1 ve b2 karar değişkenlerinin ve m amaç fonksiyonu değerlerinin her bir iterasyon için değerlerini göstermektedir. Sonuncu iterasyon optimum değeri göstermektedir.

Çizelge 5.2 iki çubuklu kirişte amaç fonksiyonunun iyileştirme oranını göstermektedir.

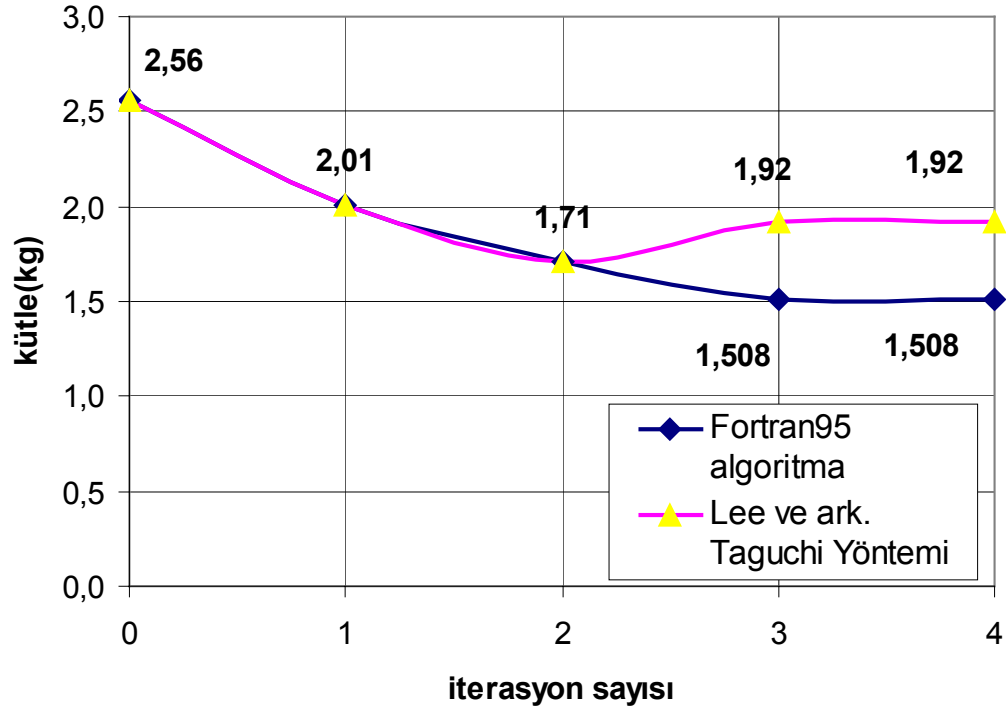
Çizelge 5.1 İki çubuklu kiriş karar değişkenleri ve amaç fonksiyonu optimum değerlerinin iterasyonla değişimi

İterasyon	Karar Değişkenleri		Amaç fonksiyonu
	b1	b2	Kütle (kg)
Başlangıç değeri	2	0.8	2.56
1	1.8	0.5	2.01
2	1.6	0.38	1.71
3	1.41	0.38	1.508
4	1.41	0.38	1.508

Çizelge 5.2 İki çubuklu kirişte amaç fonksiyonu iyileştirme oranı

	Başlangıç değeri	Optimum değer	İyileştirme oranı
Kütle (kg)	2.56	1.508	%41,1
Analiz sayısı = 40			

Şekil 5.5 iki çubuklu kiriş için Lee ve arkadaşlarının Taguchi yöntemi ile geliştirilen algoritma ve bu tezde geliştirilen Fortran95 algoritmasını karşılaştıran grafiği göstermektedir. Şekil 5.5’de görüldüğü gibi Lee ve arkadaşlarının yöntemi 4 iterasyonda toplam 40 analiz ile analitik çözüme yakınsayamazken, geliştirilen algoritma ile yazılan Fortran95 programı 4 iterasyonda toplam 40 analiz ile optimum tasarıma yakınsamaktadır.



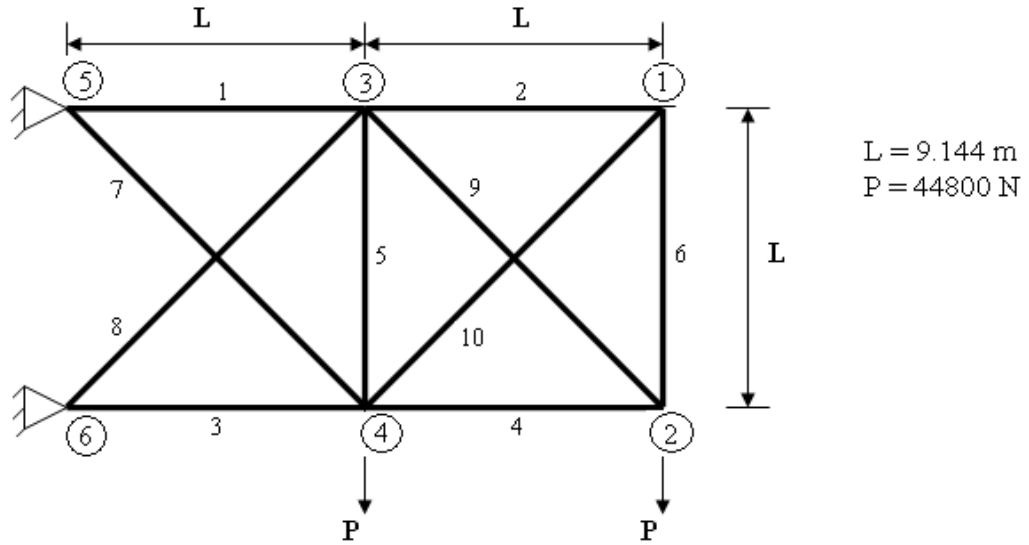
Şekil 5.5 İki çubuklu kiriş için Fortran95 algoritması ve Lee ve arkadaşlarının Taguchi yöntemini karşılaştırır kütle değişim grafiği

Lee ve arkadaşlarının yönteminde toplam kütle iyileştirme oranı %24,81 olarak gerçekleşirken, geliştirilen Fortran95 algoritmasının kütle iyileştirme oranı %41,1 olarak gerçekleşmiştir.

5.2 On Çubuklu Kiriş

Geliştirilen algoritma ile oluşturulan Fortran95 programının performansını literatürdeki yayınlarla kıyaslamak için ikinci test problemi olarak Şekil 5.6’da

gösterilen on çubuklu kiriş seçilmiştir. On çubuklu kiriş probleminin amacı, elemanlar üzerindeki gerilme ve nodal yerdeğiştirme kısıtlarını sağlayan sistemin toplam kütlesini minimize etmektir.



Şekil 5.6 Optimizasyon tasarımı yapılacak on çubuklu kiriş

Optimizasyon sürecinde karar değişkeni olarak on çubuklu kirişin kesit alanları (A_1 - A_{10}) seçilmiştir. On çubuklu kirişin malzeme özellikleri E (Elastisite modülü)=68.9 GPa, μ (poisson oranı)=0.33 ve ρ (yoğunluk)=2770 kg/m³ olarak verilmiştir.

A_i karar değişkenlerinin en iyi değerlerini bulmak için tanımlanan optimizasyon problemi, boyutsuzlaştırıldıktan sonra yapısal tasarım optimizasyon probleminin deney tasarımı standart matematiksel formatta aşağıdaki şekilde verilebilir:

Bul	: A_i (karar değişkenleri)
Enküçüle	: kütle (amaç fonksiyonu)
Gerilme kısıtları	: $-172,4 \text{ MPa} \leq \sigma_i \leq 172,4 \text{ MPa}$
Aday değerler	: $A_i (\times 10^{-5} \text{ m}^2) \in \{6, 97, 194, 323, 400, 465, 542, 645, 729, 813\}$
	: $i = (1, 2, \dots, 10)$

Kesikli optimizasyona başlamak için kesit alanlarının ilk değerleri aşağıdaki şekilde seçilmiştir. :

$$[A_1, A_2, \dots, A_{10}]^T = [729, 729, 729, 465, 465, 465, 465, 465, 465, 465]^T$$

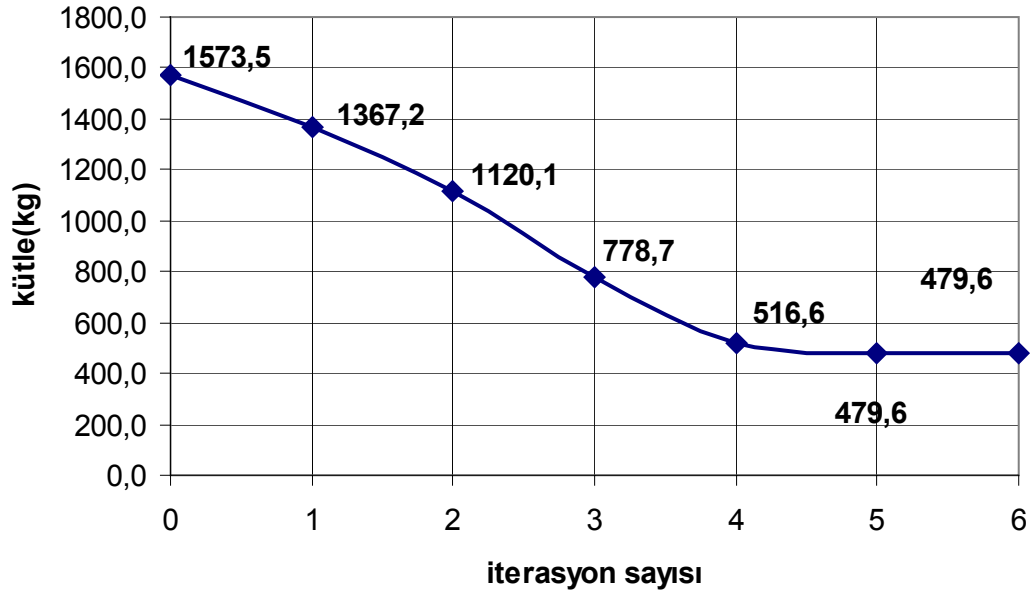
$$i = (1, 2, \dots, 10)$$

Burada σ_i , i 'nci eleman üzerine etkiyen gerilme değeridir.

Geliştirilen algoritma kullanılarak on çubuklu kiriş için her bir iterasyondaki kütle değişimi Şekil 5.7'de gösterilmiştir. Şekillerde gösterilen sıfırıncı iterasyon ilk tasarım değerini belirtmektedir. Şekilde görüldüğü gibi kütledeki değişimdeki ilerleme kaydedilememesi 3 iterasyona ulaştığında iterasyon sonlanır. Çizelge 5.3'de elde edilen optimal değerler genetik algoritma sonuçları ve literatürdeki diğer yayınlarla karşılaştırılmıştır [Lee et al, 2003].

Karar değişkenlerini sayısı 10 ve düzeyleri üç olarak sabitlendiğinden minimum ardışık dizin Çizelge 5.4'de gösterildiği gibi $L_{27}(3^{13})$ olarak seçilmiştir. Karar değişkenlerinin dizin kolonlarına atanması keyfi olarak yapılmıştır. Ardışık dizinin her bir iterasyon için ihtiyacı olan sonlu eleman analizi sayısı 27'dir.

Şekil 5.7'de görüldüğü gibi optimizasyon algoritması 6 iterasyonda sonlanmıştır. Yapılan toplam analiz sayısı 168, amaç fonksiyonundaki iyileştirme oranı başlangıç değerine göre %69,52 olarak gerçekleşmiştir.



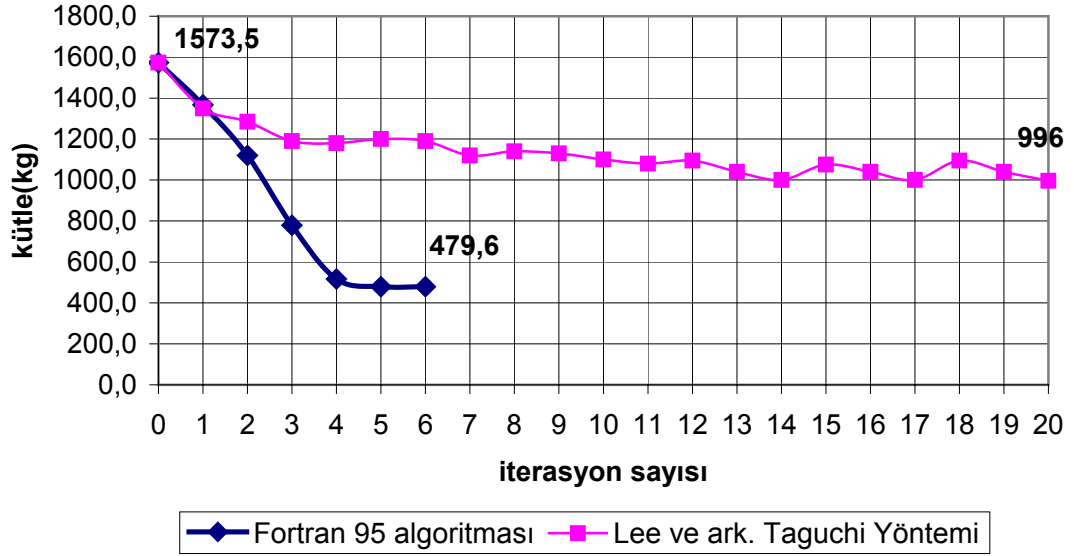
Şekil 5.7 On çubuklu kiriş için kütle değişim grafiği

Çizelge 5.3 Karar değişkenlerinin optimumlarının ilk dizayn ve literatürdeki diğer değerlerle kıyaslanması

Karar Değişkenleri	İlk Dizayn	Optimum Dizayn	Taguchi Metodu	Genetik algoritma
$A_1(x10^{-5} m^2)$	729	400	542	542
$A_2(x10^{-5} m^2)$	729	400	97	97
$A_3(x10^{-5} m^2)$	729	465	542	97
$A_4(x10^{-5} m^2)$	465	6	323	542
$A_5(x10^{-5} m^2)$	465	97	465	465
$A_6(x10^{-5} m^2)$	465	97	97	6
$A_7(x10^{-5} m^2)$	465	6	400	97
$A_8(x10^{-5} m^2)$	465	97	400	465
$A_9(x10^{-5} m^2)$	465	6	465	465
$A_{10}(x10^{-5} m^2)$	465	194	97	6
Kütle (kg)	1573.54	479.60	996.84	931.46
Sonlu Eleman Analiz Sayısı		168	532	5369

Çizelge 5.4 On çubuklu kirişte karar değişkenlerinin atanması $L_{27}(3^{13})$

Deney no	Kolon numarası ve atanan faktörler												
	1	2	3	4	5	6	7	8	9	10	11	12	13
	A ₁	A ₂	ihmal	A ₃	A ₄	ihmal	A ₅	ihmal	A ₆	A ₇	A ₈	A ₉	A ₁₀
1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	1	1	1	1	2	2	2	2	2	2	2	2	2
3	1	1	1	1	3	3	3	3	3	3	3	3	3
4	1	2	2	2	1	1	1	2	2	2	3	3	3
5	1	2	2	2	2	2	2	3	3	3	1	1	1
6	1	2	2	2	3	3	3	1	1	1	2	2	2
7	1	3	3	3	1	1	1	3	3	3	2	2	2
8	1	3	3	3	2	2	2	1	1	1	3	3	3
9	1	3	3	3	3	3	3	2	2	2	1	1	1
10	2	1	2	3	1	2	3	1	2	3	1	2	3
11	2	1	2	3	2	3	1	2	3	1	2	3	1
12	2	1	2	3	3	1	2	3	1	2	3	1	2
13	2	2	3	1	1	2	3	2	3	1	3	1	2
14	2	2	3	1	2	3	1	3	1	2	1	2	3
15	2	2	3	1	3	1	2	1	2	3	2	3	1
16	2	3	1	2	1	2	3	3	1	2	2	3	1
17	2	3	1	2	2	3	1	1	2	3	3	1	2
18	2	3	1	2	3	1	2	2	3	1	1	2	3
19	3	1	3	2	1	3	2	1	3	2	1	3	2
20	3	1	3	2	2	1	3	2	1	3	2	1	3
21	3	1	3	2	3	2	1	3	2	1	3	2	1
22	3	2	1	3	1	3	2	2	1	3	3	2	1
23	3	2	1	3	2	1	3	3	2	1	1	3	2
24	3	2	1	3	3	2	1	1	3	2	2	1	3
25	3	3	2	1	1	3	2	3	2	1	2	1	3
26	3	3	2	1	2	1	3	1	3	2	3	2	1
27	3	3	2	1	3	2	1	2	1	3	1	3	2



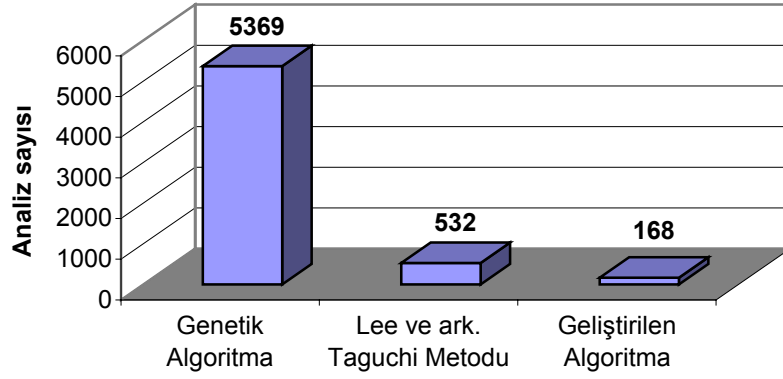
Şekil 5.8 On çubuklu kiriş için Fortran95 algoritması ve Lee ve arkadaşlarının Taguchi yöntemini karşılaştırır kütle değişim grafiği

Şekil 5.8 Lee ve arkadaşlarının Taguchi yöntemi ile geliştirilen algoritma ve bu tezde geliştirilen Fortran95 algoritmasını karşılaştıran grafiği göstermektedir. Şekil 5.8’de görüldüğü gibi Lee ve arkadaşlarının yöntemi 19 iterasyonda toplam 532 analiz ile yakınsarken, geliştirilen algoritma ile yazılan Fortran95 programı 6 iterasyonda toplam 168 analiz ile optimum tasarıma yakınsamaktadır.

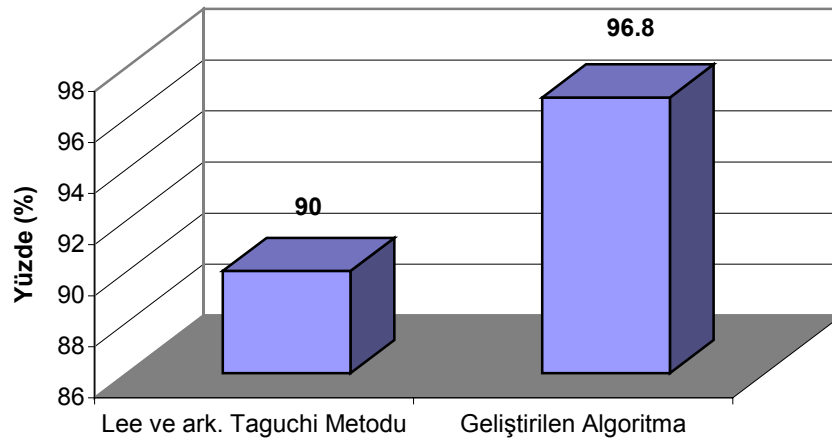
Şekil 5.9 Kıyaslamalı olarak iterasyon yakınsayana kadar yapılan toplam sonlu eleman analiz sayısını göstermektedir. Şekilde görüldüğü gibi geliştirilen algoritmanın hesap maliyeti diğer yöntemlere göre çok daha azdır. Lee ve arkadaşlarının Taguchi Metodundaki fonksiyon sayısı genetik algoritmadaki fonksiyon sayısının yaklaşık 1/10 iken, geliştirilen algoritmadaki fonksiyon sayısı genetik algoritmadaki fonksiyon sayısının yaklaşık 1/32’dir.

Şekil 5.10 Genetik algoritma sonlu eleman analiz sayısı 100 baz puan alınırsa Lee ve arkadaşlarının Taguchi Metodundaki fonksiyon sayısında iyileştirme oranı ile geliştirilen algoritmadaki iyileştirme oranlarını göstermektedir. Şekilde görüldüğü gibi Lee ve arkadaşlarının Taguchi Metodundaki fonksiyon sayısında iyileştirme

oranı %90 mertebesinde kalırken, geliştirilen algoritmadaki fonksiyon sayısındaki iyileştirme oranı %96,8 olarak gerçekleşmiştir. Bu da, geliştirilen algoritma ile yazılan Fortran95 programının Lee ve arkadaşlarının Taguchi Metoduna göre yaklaşık %7 hesap maliyeti kazancı olduğu anlamına gelmektedir.



Şekil 5.9 On çubuklu kiriş için toplam gerçekleştirilen sonlu eleman analiz sayısı



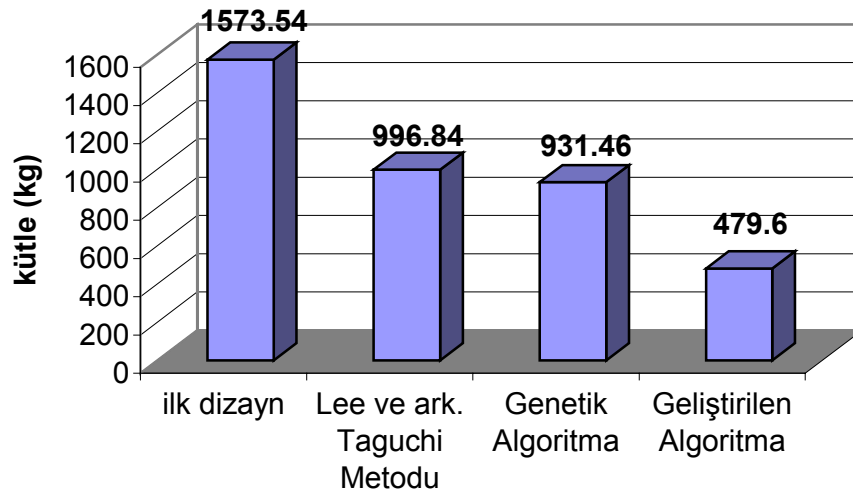
Şekil 5.10 On çubuklu kiriş için karşılaştırmalı hesap maliyetleri

Çizelge 5.5 On çubuklu kiriş için ilk tasarım sonuçları ve diğer sonuçların kütle iyileştirme oranları karşılaştırması

	İlk Dizayn	Optimum Dizayn	Taguchi Metodu	Genetik algoritma
Kütle değişimindeki iyileştirme oranı(%)	100	69,53	36,65	41

Çizelge 5.5 ilk tasarım ve diğer yöntemler arasında kütledeki iyileştirmeleri kıyaslamaktadır. İlk tasarım kütlesi 100 baz puan olarak alındığında, kütledeki iyileştirme oranı genetik algoritma için % 41, Lee ve arkadaşlarının Taguchi metodu için %36,65 ve geliştirilen Fortran95 algoritması için % 69,53 olarak bulunmaktadır. Kütledeki iyileştirmenin geliştirilen Fortran95 algoritmasında diğer yöntemlere göre daha üstün olduğu açıkça görülmektedir.

Şekil 5.11 geliştirilen Fortran95 programı ve diğer sonuçlardan elde edilen optimal kütle değerlerini karşılamaktadır. Lee ve arkadaşlarının Taguchi metodunda, genetik algoritmaya göre %7 daha fazla optimal ağırlığa karşılık hesap maliyetinde %90 iyileştirme meydana gelmiştir. Buna karşılık, geliştirilen Fortran95 algoritmasında genetik algoritmaya göre % 51,4 daha az optimal ağırlığa karşılık hesap maliyetinde %96,8 iyileştirme meydana gelmiştir.



Şekil 5.11 On çubuklu kiriş için optimal kütle kıyaslama grafiği

Çizelge 5.6 on çubuklu kirişte amaç fonksiyonunun (kütle) iyileştirme oranını göstermektedir. Çizelge 5.7, on çubuklu kirişte $A_1, A_2, A_3, A_4, A_5, A_6, A_7, A_8, A_9$, ve A_{10} karar değişkenlerinin ve amaç fonksiyonu değerlerinin her bir iterasyon için değerlerini göstermektedir. Sıfırıncı iterasyon ilk başlangıç değerini, sonuncu iterasyon ise optimum değeri göstermektedir.

Çizelge 5.6 On çubuklu kirişte amaç fonksiyonu iyileştirme oranı

	Başlangıç değeri	Optimum değeri	İyileştirme oranı
Kütle (kg)	1573.5	479.6	%69.52
Analiz sayısı = 168			

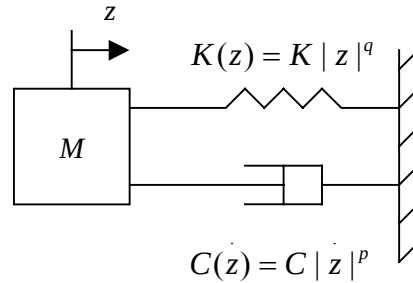
Çizelge 5.7 On çubuklu kiriş karar değişkenleri ve amaç fonksiyonu optimum değerlerinin iterasyonla değişimi

İterasyon	Karar değişkenleri										Amaç fonks.
	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}	Kütle (kg)
Başlangıç değeri	729	729	729	465	465	465	465	465	465	465	1573.5
1	645	645	645	400	400	400	400	400	400	400	1367.18
2	542	542	542	323	323	323	323	323	323	323	1120.08
3	465	465	465	194	194	194	194	194	194	194	778.71
4	400	400	400	970	970	970	970	970	970	970	516.63
5	400	400	465	60	970	970	60	970	60	194	479.60
6	400	400	465	60	970	970	60	970	60	194	479.6

5.3 Doğrusal Olmayan Darbe Emici

Şekil 5.12’de gösterilen doğrusal olmayan darbe emicide, yay sabitinin yerdeğiştirme büyüklüğünün q ’uncu dereceden üstü ile doğru orantılı, sönümün de hız büyüklüğünün p ’inci dereceden üstü ile doğru orantılı olduğu kabul edilmektedir.

Doğrusal olmayan darbe emicinin tasarım amacı doğrusal darbe emici ile aynıdır. Bu da, kütlenin 12 saniye boyunca maksimum ivmeyi enküçükleyen ve maksimum yerdeğiştirmeyi sınırlandıran doğrusal olmayan yay ve damper parametrelerini (K , C , p ve q) bulmak şeklinde tanımlanabilir.



Şekil 5.12 Doğrusal olmayan darbe emici

Optimizasyon probleminin matematiksel tanımı, amaç fonksiyonu ve kısıtlar sırasıyla (5.6) ve (5.7) denklemlerinde verilmektedir.

En küçükke :

$$\text{Maksimum } |Acc| \quad (\max |\ddot{z}|) \quad (5.6)$$

Kısıtlar :

$$\text{Maksimum } |Disp| \leq 1 \text{ m} \quad (\max |z| \leq 1 \text{ m}) \quad (5.7)$$

Tasarım Alanı :

$$0 < K \leq 1 \text{ N/m}, \quad 0 < C \leq 1 \text{ Ns/m} \quad (5.8)$$

$$0 \leq p \leq 2, \quad 0 \leq q \leq 2. \quad (5.9)$$

şeklinde tanımlanır.

Tasarım alanında K için aday değerleri:

$$K = (0.1, 0.2, 0.3, 0.4, 0.45, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0)$$

Tasarım alanında C için aday değerleri:

$$C = (0.1, 0.2, 0.3, 0.4, 0.45, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0)$$

Tasarım alanında p için aday değerleri:

$$p = (0.01, 0.2, 0.4, 0.6, 0.8, 1.0, 1.2, 1.4, 1.6, 1.8, 2.0)$$

Tasarım alanında q için aday değerleri:

$$q = (0.01, 0.2, 0.4, 0.6, 0.8, 1.0, 1.2, 1.4, 1.6, 1.8, 2.0)$$

olarak seçilmiştir.

(5.6) ve (5.7) denklemlerindeki maksimum |...| şeklinde yazılan büyüklükler nümerik çözümlerden elde edilir.

Doğrusal olmayan darbe emicinin denklemi

$$M \ddot{z} + C | \dot{z} |^p \dot{z} + K | z |^q z = 0 \quad (5.10)$$

şeklinde, başlangıç şartları ise (kütlenin yerdeğiştirme ve hız başlangıç değerleri)

$$z_0 = 0 \text{ m ve } \dot{z}_0 = 1 \text{ m/s} \quad (5.11)$$

şeklinde tanımlanır.

5.3.1 Çok kütleli sistemin hareket denkleminin sayısal çözümü

5.3.1.1 Çok kütleli sistemlerde hareket denklemleri

Çok kütleli bir sistemin hareketi (4.12) denkleminde gösterildiği gibi ikinci dereceden bir diferansiyel denklem olarak tanımlanır.

$$\mathbf{M}\ddot{\mathbf{x}} + \mathbf{C}\dot{\mathbf{x}} + \mathbf{K}\mathbf{x} = \mathbf{F}_{dış} \quad (5.12)$$

Burada :

$\mathbf{M}$: Sistemin kütlesi

$\mathbf{C}$: Sistemin sönüm katsayısı

$\mathbf{K}$: Sistemin yay sabiti

değerlerini ifade eder.

$\mathbf{x}$, $\dot{\mathbf{x}}$ ve $\ddot{\mathbf{x}}$ sırasıyla çok kütleli sistemin yerdeğiştirme, hız ve ivme vektörlerini temsil eder. $\mathbf{F}_{dış}$, sisteme dışarıdan uygulanan kuvvetlerdir. (5.12)

denklemini, $(\mathbf{x}_0)$ ilk yerdeğiştirmesi ve $(\dot{\mathbf{x}}_0)$ ilk hızına sahiptir.

(5.12) denklemini ,

$$\mathbf{M}\ddot{\mathbf{x}} = \mathbf{F} \quad (5.13)$$

şeklinde daha basit bir formda yazılabilir.

Burada $\mathbf{F}$,

$$\mathbf{F} = \mathbf{F}_{dış} - \mathbf{C}\dot{\mathbf{x}} - \mathbf{K}\mathbf{x} \quad (5.14)$$

şeklinde tanımlanır. $\mathbf{F}$, sistem üzerinde etki eden tüm dış ve iç yükleri temsil eder.

5.3.1.2 Çok kütleli sistemlerde hareket denkleminin çözümü

Çok kütleli sistemlerde hareket denkleminin (5.13) çözümü direk integrasyon metotları (adım adım integrasyon) vasıtasıyla elde edilir. Direk integrasyon metotlarında hareket denklemleri çözüm sürecindeki tüm zaman noktalarının yerine, kesikli zaman noktalarında sağlanır.

Kesikli şekilde hareket denklemi (5.12)

$$\mathbf{M} \ddot{\mathbf{x}}_i = \mathbf{F}_i \quad (5.15)$$

şeklinde yeniden yazılabilir.

Burada :

$\ddot{\mathbf{x}}_i$: t_i zaman noktasındaki ivme vektörü

$\mathbf{F}_i$: t_i zaman noktasındaki tüm iç ve dış kuvvet vektörlerinin toplamı

değerlerini ifade eder.

t_i ve t_{i+1} , iki ardışık zaman noktası arasındaki zaman aralığı, Δt ($\Delta t = t_{i+1} - t_i$) ise zaman adımı olarak bilinir. Zaman adımının çözümün doğruluğu üzerinde çok önemli bir etkisi vardır. Doğru hesaplamalar için küçük zaman adımı kullanılmalıdır. Bu çalışmada $\Delta t = 0.06$ zaman adımı kullanılmıştır.

Eğer t_i anındaki $\mathbf{x}_i$ yerdeğiştirme vektörü ve $\dot{\mathbf{x}}_i$ hız vektörü biliniyorsa, t_{i+1} anındaki $\mathbf{x}_{i+1}$ yerdeğiştirmesi ve $\dot{\mathbf{x}}_{i+1}$ hız vektörü değerleri Taylor serisi açılımı vasıtasıyla hesaplanabilir.

$$\dot{\mathbf{x}}_{i+1} = \dot{\mathbf{x}}_i + \Delta t_i \ddot{\mathbf{x}}_i \quad (5.16)$$

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \Delta t \dot{\mathbf{x}}_i + \frac{1}{2} (\Delta t)^2 \ddot{\mathbf{x}}_i \quad (5.17)$$

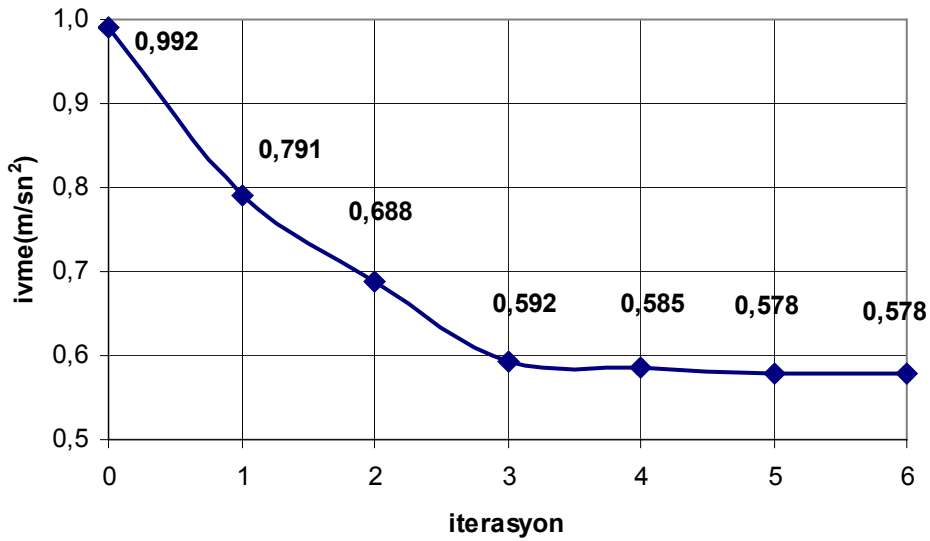
(5.16) ve (5.17) denklemlerinde ihtiyaç duyulan ivme vektörü (5.15) denklemini

$$\ddot{\mathbf{x}}_i = [\mathbf{M}]^{-1} \mathbf{F}_i \quad (5.18)$$

şeklinde kullanarak hesaplanır.

(5.16-5.18) denklemlerinde tanımlanan değerler ve verilen ilk yerdeğiştirme ve hız ($\mathbf{x}_0$ ve $\dot{\mathbf{x}}_0$) bilgilerini kullanarak, bütün kesikli zaman noktalarındaki ($i=0, 1, 2, \dots$) yerdeğiştirme ($\mathbf{x}_i$), hız ($\dot{\mathbf{x}}_i$) ve ivme ($\ddot{\mathbf{x}}_i$) değerleri kolayca hesaplanabilir.

Şekil 5.13 geliştirilen Fortran95 algoritması ile doğrusal olmayan darbe emici için ivme değişim grafiğini göstermektedir. Şekilde gösterildiği gibi toplam 6 iterasyon ve 60 analizde optimizasyon problemi yakınsamıştır.

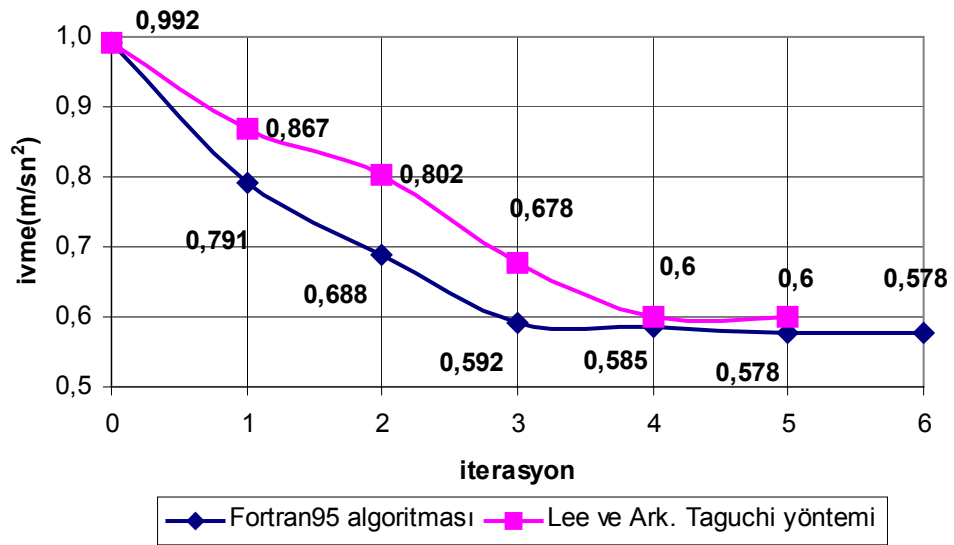


Şekil 5.13 Doğrusal olmayan darbe emici için ivme değişim grafiği

Şekil 5.14 doğrusal olmayan darbe emici için Lee ve arkadaşlarının geliştirdiği Taguchi yöntemi ve bu tezde geliştirilen Fortran95 algoritmasını karşılaştıran grafiği göstermektedir. Şekil 5.14'de görüldüğü gibi Lee ve arkadaşlarının geliştirdiği

yöntem 5 iterasyonda toplam 50 analiz ile yakınsarken, geliştirilen algoritma ile yazılan Fortran95 programı 6 iterasyonda toplam 60 analiz ile optimum tasarıma yakınsamaktadır.

Hesap maliyetindeki bu ufak farklılığa karşılık Lee ve arkadaşlarının geliştirdiği Taguchi yönteminde toplam amaç fonksiyonundaki iyileştirme oranı %39.49 olurken, geliştirilen algoritma ile yazılan Fortran95 programı ile bu iyileştirme oranı %41.73 olarak gerçekleşmiştir.



Şekil 5.14 Doğrusal olmayan darbe emici için kıyaslamalı ivme değişim grafiği

Çizelge 5.8 doğrusal olmayan darbe emicide amaç fonksiyonunun (ivme) geliştirilen algoritma için iyileştirme oranını göstermektedir. Çizelge 5.9 doğrusal olmayan darbe emicide K,C,p ve q karar değişkenlerinin ve amaç fonksiyonu değerlerinin her bir iterasyon için değerlerini göstermektedir. Sonuncu iterasyon ise optimum değeri göstermektedir.

Çizelge 5.8 Doğrusal olmayan darbe emici amaç fonksiyonu iyileştirme oranı

	Başlangıç değeri	Optimum değer	İyileştirme oranı
İvme (m/s^2)	0.991	0.577	% 41.73
Analiz sayısı = 60			

Çizelge 5.9 Doğrusal olmayan darbe emici karar değişkenleri ve amaç fonksiyonu optimum değerlerinin iterasyonla değişimi

İterasyon	Karar Değişkenleri				Amaç Fonksiyonu
	K (N/m)	C (Ns/m)	p	q	İvme (m/s^2)
Başlangıç değeri	1	0.3	1	1	0.992
1	0.8	0.4	0.8	1	0.791
2	0.7	0.45	0.6	1	0.688
3	0.6	0.5	0.4	1	0.591
4	0.6	0.5	0.2	1.2	0.585
5	0.6	0.5	0.01	1.4	0.577
6	0.6	0.5	0.01	1.4	0.577

6 SONUÇLAR VE ÖNERİLER

Bir önceki bölümde bahsedilen her bir test problemi için geliştirilen algorithmadan oluşturulan Fortran95 programı sonuçları genetik algoritma ve literatürde bulunan sonuçlarla [Lee et al, 2003] karşılaştırılmıştır. Tüm örneklerdeki aday kesikli değerler ve ilk değerler keyfi olarak seçilmiştir. Geliştirilen algoritma ve genetik algoritmalar göz önüne alındığında, yaklaşım kriteri için aynı şartları tanzim etmek mümkün değildir.

İki çubuklu kiriş için yapılan analizler neticesinde geliştirilen algoritma ile yazılan Fortran95 programı ile toplam 4 iterasyon ve 40 analiz neticesinde amaç fonksiyonunda ilk dizayna göre %41'lik bir iyileştirme sağlanmıştır.

İki çubuklu kiriş için optimizasyon sonuçları, genetik algoritma ve Lee ve arkadaşlarının Taguchi metotları ile karşılaştırıldığında hesap maliyetinin diğer sonuçlara göre çok daha aşağıya çekildiği ve Lee ve arkadaşlarının kullandığı Taguchi metodunda amaç fonksiyonunun optimal değerine yakınsaması sağlanamazken, geliştirilen Fortran95 algoritmasında optimal değere yakınsadığı görülmektedir.

On çubuklu kiriş için yapılan analizler neticesinde geliştirilen algoritma ile yazılan Fortran95 programı ile toplam 6 iterasyon ve 168 analiz neticesinde amaç fonksiyonunda ilk dizayna göre %69,52'lik bir iyileştirme sağlanmıştır.

On çubuklu kiriş için optimizasyon sonuçları, genetik algoritma ve Lee ve arkadaşlarının Taguchi metotları ile karşılaştırıldığında hesap maliyetinin diğer sonuçlara göre çok daha aşağıya çekildiği ve amaç fonksiyonunun optimal değerinin daha iyi sonuçlar verdiği görülmektedir.

Doğrusal olmayan darbe emici için yapılan analizler neticesinde geliştirilen algoritma ile yazılan Fortran95 programı ile toplam 6 iterasyon ve 60 analiz

neticesinde amaç fonksiyonunda ilk dizayna göre %41,73'lük bir iyileştirme sağlanmıştır.

Doğrusal olmayan darbe emici için optimizasyon sonuçları, Lee ve arkadaşlarının Taguchi metotları ile karşılaştırıldığında hesap maliyetinin biraz yukarıda olduğu, buna karşılık amaç fonksiyonunun optimal değerinin daha iyi sonuç verdiği görülmektedir.

Tezde elde edilen sonuçlar geliştirilen ardışık Taguchi optimizasyon metodunun statik ve dinamik yüklemelere maruz diğer kesikli yapısal optimizasyon problemlerinin çözümünde efektif olarak kullanılabileceğini göstermektedir.

KAYNAKLAR DİZİNİ

- Arora, J.S. and Haug, M.W., “Methods for Optimization of Nonlinear Problems with Discrete Variables: a review”, Struct. Opt. 8, 69-85, 1994
- Arora, J.S., “Introduction to Optimum Design”, McGraw-Hill Book Company, New York, 1989
- Bakır, M. Akif ve Altunkaynak, Bülent, Tamsayılı Programlama, 2003
- Glover, F., “Tabu Search – Part II”, ORSA Journal of Computing Vol. 2, No. 1., 1990
- Haftka, R.T., Gürdal, Z., “Elements of Structural Optimization”, Kluwer Academic Publishers, The Netherlands., 1992
- Holland, J. H., “Adaptation in Natural and Artificial Systems”, Univ. Mich. Press. 1975
- Krishnamoorthy, C.S., Rajeev, S., “Discrete Optimization of Structures Using Genetic Algorithm”, J Struct. Eng. ASCE 118, 1233-1250, 1992
- Ku et al, “Taguchi-aided Search Method for Design Optimization of Engineering Systems”, Eng. Opt. 30, 1-23, 1998
- Kurtaran, Hasan, Sonlu elemanlar II ders notları, 2004
- Kurtaran, Hasan, Sonlu elemanlar III ders notları, 2005
- Kurtaran, Hasan, Optimizasyon ders notları, 2005
- Lee, K.H., Park, G.J., “Discrete Post-process of the Constrained Size Optimization using Orthogonal Arrays, in Proceedings of the Sixth AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Part 1”, Bellevue, Washington, pp. 242-244, 1996

- Lee et al, “An optimization algorithm using orthogonal arrays in discrete design space for structures”, *Finite Elements in Analysis and Design* 40 121-135, 2003
- Park, G.J., Arora, S., “Role of Database Management in Design Optimization”, *J Aircraft* 24 (11) 745-750, 1987
- Peace, G.S., “Taguchi Methods: A Hands-On Approach”, Addison-Wesley, MA, 1995
- Phadke, M.S., “Quality Engineering Using Robust Design”, Prentice Hall, Englewood Cliffs, NJ, 1989
- Roux, W.J., “Structural Optimization Using Response Surface Methodology”, Ph.D. Thesis, University of Pretoria, South Africa, 1997.
- Saljegheh, E., Vanderplaast, G.N., “Optimum Design of Structures with Discrete Sizing and Shape Variables”, *Struct. Opt.* 6 (2) 79-85, 1993
- Taha, Hamdy, *Yöneylem Araştırması*, 2000
- Vanderplaats, G.N., “Numerical Optimization Techniques for Engineering Design”, Vanderplaats Research and Development, Inc., Colorado Springs, CO, USA, 1998
- Vanderplaats, G.N., “Numerical Optimization Techniques for Engineering Design”, McGraw-Hill Book Company, New York, 1984

ÖZGEÇMİŞ

Doğum tarihi	21.04.1975
Doğum yeri	Adana
Lise	1989–1993 Deniz Lisesi Komutanlığı Heybeliada/İstanbul
Lisans	1993–1997 Deniz Harp Okulu Komutanlığı Tuzla/İstanbul Makine Mühendisliği Bölümü
Yüksek Lisans	2000–2002 Naval PostGraduate School, Monterey, CA, USA Mechanical Engineering Department
Yüksek Lisans	2004–2007 Gebze Yüksek Teknoloji Enstitüsü, Tasarım ve İmalat Mühendisliği Anabilim dalı

Çalıştığı kurum / Görev Yerleri

1997 – 1999	TCG SERDAR Elektrik Subayı	Deniz Kuvvetleri Komutanlığı
1999 – 2000	TCG SERDAR Yara Savunma Subayı	Deniz Kuvvetleri Komutanlığı
2002 - ...	Deniz Harp Okulu	Deniz Kuvvetleri Komutanlığı

EKLER

Eklerde verilen Taguchi ortogonal dizinleri, karar değişkenleri ve düzeylerine göre sıralanarak sunulmuştur. Örneğin, L4 Taguchi ortogonal dizininin anlamı, bu ortogonal dizin en fazla üç karar değişkenli ve iki düzeyli optimizasyon problemlerinde kullanılabilir demektir. Çoklu seviyenin anlamı her bir parametrenin birden fazla değerden oluşmasıdır. Bu durumda L4 ortogonal dizininde, üç adet karar değişkeni olduğu ve her bir karar değişkeninin de iki değere sahip olduğu anlamı çıkar. Toplamda ise dört adet deney yapıldığı, L harfinin yanındaki rakamdan anlaşılır. Eğer karar değişkenlerinin toplam değerleri tamsayı olarak aşağıda sunulan standartlara uymuyorsa, bu durumda bir üst düzeyden matris deneyi yapılır. Bu durumda, karar değişkenlerinin sütunlara ataması keyfi olarak yapılacağından artan sütunlar ihmal edilerek işlem gerçekleştirilir.

- L4 (2^3) : Üç karar değişkenli iki-düzeyle faktör
- L8 (2^7) : Yedi karar değişkenli iki- düzeyli faktör
- L9 (3^4) : Dört karar değişkenli üç-düzeyle faktör
- L12 (2^{11}) : Onbir karar değişkenli iki-düzeyle faktör
- L16 (2^{15}) : Onbes karar değişkenli iki-düzeyle faktör
- L16b (4^5) : Beş karar değişkenli dört-düzeyle faktör
- L18 ($2^1 \times 3^7$) : Bir karar değişkenli iki-düzeyle ve yedi karar değişkenli üç-düzeyle faktör
- L25 (5^6) : Altı karar değişkenli beş-düzeyle faktör
- L27 (3^{13}) : Onüç karar değişkenli üç-düzeyle faktör
- L32 (2^{31}) : Otuz iki karar değişkenli iki-düzeyle faktör
- L32b ($2^1 \times 4^9$) : Bir karar değişkenli iki-düzeyle faktör ve dokuz karar değişkenli dört-düzeyle faktör
- L36 ($2^{11} \times 3^{12}$) : Onbir karar değişkenli iki-düzeyle faktör ve oniki karar değişkenli üç-düzeyle faktör
- L50 ($2^1 \times 5^{11}$) : Bir karar değişkenli iki-düzeyle faktör ve onbir karar değişkenli beş-düzeyle faktör

L4 ORTOGONAL DİZİNİ

<u>Deney No</u>	<u>Sütun1</u>	<u>Sütun2</u>	<u>Sütun3</u>
1	1	1	1
2	1	2	2
3	2	1	2
4	2	2	1

L8 ORTOGONAL DİZİNİ

<u>Deney No</u>	<u>Sütun1</u>	<u>Sütun2</u>	<u>Sütun3</u>	<u>Sütun4</u>	<u>Sütun5</u>	<u>Sütun6</u>	<u>Sütun7</u>
1	1	1	1	1	1	1	1
2	1	1	1	2	2	2	2
3	1	2	2	1	1	2	2
4	1	2	2	2	2	1	1
5	2	1	2	1	2	1	2
6	2	1	2	2	1	2	1
7	2	2	1	1	2	2	1
8	2	2	1	2	1	1	2

L9 ORTOGONAL DİZİNİ

<u>Deney No</u>	<u>Sütun1</u>	<u>Sütun 2</u>	<u>Sütun 3</u>	<u>Sütun 4</u>
1	1	1	1	1
2	1	2	2	2
3	1	3	3	3
4	2	1	2	3
5	2	2	3	1
6	2	3	1	2
7	3	1	3	2
8	3	2	1	3
9	3	3	2	1

L12 ORTOGONAL DİZİNİ

<u>Deney No</u>	<u>Sütun 1</u>	<u>Sütun 2</u>	<u>Sütun 3</u>	<u>Sütun 4</u>	<u>Sütun 5</u>	<u>Sütun 6</u>	<u>Sütun 7</u>	<u>Sütun 8</u>	<u>Sütun 9</u>	<u>Sütun 10</u>	<u>Sütun 11</u>
1	1	1	1	1	1	1	1	1	1	1	1
2	1	1	1	1	1	2	2	2	2	2	2
3	1	1	2	2	2	1	1	1	2	2	2
4	1	2	1	2	2	1	2	2	1	1	2
5	1	2	2	1	2	2	1	2	1	2	1
6	1	2	2	1	2	2	1	2	1	2	1
7	1	2	2	2	1	2	2	1	2	1	1
8	2	1	2	1	2	2	2	1	1	1	2
9	2	1	1	2	2	2	1	2	2	1	1
10	2	2	2	1	1	1	1	2	2	1	2
11	2	2	1	2	1	2	1	1	1	2	2
12	2	2	1	1	2	1	2	1	2	2	1

L16 ORTOGONAL DİZİNİ

<u>Denev</u> <u>No</u>	<u>Sütun</u> <u>1</u>	<u>Sütun</u> <u>2</u>	<u>Sütun</u> <u>3</u>	<u>Sütun</u> <u>4</u>	<u>Sütun</u> <u>5</u>	<u>Sütun</u> <u>6</u>	<u>Sütun</u> <u>7</u>	<u>Sütun</u> <u>8</u>	<u>Sütun</u> <u>9</u>	<u>Sütun</u> <u>10</u>	<u>Sütun</u> <u>11</u>	<u>Sütun</u> <u>12</u>	<u>Sütun</u> <u>13</u>	<u>Sütun</u> <u>14</u>	<u>Sütun</u> <u>15</u>
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2
3	1	1	1	2	2	2	2	1	1	1	1	2	2	2	2
4	1	1	1	2	2	2	2	2	2	2	2	1	1	1	1
5	1	2	2	1	1	2	2	1	1	2	2	1	1	2	2
6	1	2	2	1	1	2	2	2	2	1	1	2	2	1	1
7	1	2	2	1	1	2	2	2	2	1	1	2	2	1	1
8	1	2	2	2	2	1	1	2	2	1	1	1	1	2	2
9	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2
10	2	1	2	1	2	1	2	2	1	2	1	2	1	2	1
11	2	1	2	2	1	2	1	1	2	1	2	2	1	2	1
12	2	1	2	2	1	2	1	2	1	2	1	1	2	1	2
13	2	2	1	1	2	2	1	1	2	2	1	1	2	2	1
14	2	2	1	1	2	2	1	2	1	1	2	2	1	1	2
15	2	2	1	2	1	1	2	1	2	2	1	2	1	1	2
16	2	2	1	2	1	1	2	2	1	1	2	1	2	2	1

L16b ORTOGONAL DİZİNİ

<u>Denev No</u>	<u>Sütun 1</u>	<u>Sütun 2</u>	<u>Sütun 3</u>	<u>Sütun 4</u>	<u>Sütun 5</u>
1	1	1	1	1	1
2	1	2	2	2	2
3	1	3	3	3	3
4	1	4	4	4	4
5	2	1	2	3	4
6	2	2	1	4	3
7	2	3	4	1	2
8	2	4	3	2	1
9	3	1	3	4	2
10	3	2	4	3	1
11	3	3	1	2	4
12	3	4	2	1	3
13	4	1	4	2	3
14	4	2	3	1	4
15	4	3	2	4	1
16	4	4	1	3	2

L18 ORTOGONAL DİZİNİ

<u>Deney No</u>	<u>Sütun 1</u>	<u>Sütun2</u>	<u>Sütun3</u>	<u>Sütun4</u>	<u>Sütun5</u>	<u>Sütun6</u>	<u>Sütun7</u>	<u>Sütun8</u>
1	1	1	1	1	1	1	1	1
2	1	1	2	2	2	2	2	2
3	1	1	3	3	3	3	3	3
4	1	2	1	1	2	2	3	3
5	1	2	2	2	3	3	1	1
6	1	2	3	3	1	1	2	2
7	1	3	1	2	1	3	2	3
8	1	3	2	3	2	1	3	1
9	1	3	3	1	3	2	1	2
10	2	1	1	3	3	2	2	1
11	2	1	2	1	1	3	3	2
12	2	1	3	2	2	1	1	3
13	2	2	1	2	3	1	3	2
14	2	2	2	3	1	2	1	3
15	2	2	3	1	2	3	2	1
16	2	3	1	3	2	3	1	2
17	2	3	2	1	3	1	2	3
18	2	3	3	2	1	2	3	1

L25 ORTOGONAL DİZİNİ

<u>Denev No</u>	<u>Sütun 1</u>	<u>Sütun 2</u>	<u>Sütun 3</u>	<u>Sütun 4</u>	<u>Sütun 5</u>	<u>Sütun 6</u>
1	1	1	1	1	1	1
2	1	2	2	2	2	2
3	1	3	3	3	3	3
4	1	4	4	4	4	4
5	1	5	5	5	5	5
6	2	1	2	3	4	5
7	2	2	3	4	5	1
8	2	3	4	5	1	2
9	2	4	5	1	2	3
10	2	5	1	2	3	4
11	3	1	3	5	2	4
12	3	2	4	1	3	5
13	3	3	5	2	4	1
14	3	4	1	3	5	2
15	3	5	2	4	1	3
16	4	1	4	2	5	3
17	4	2	5	3	1	4
18	4	3	1	4	2	5
19	4	4	2	5	3	1
20	4	5	3	1	4	2
21	5	1	5	4	3	2
22	5	2	1	5	4	3
23	5	3	2	1	5	4
24	5	4	3	2	1	5
25	5	5	4	3	2	1

L27 ORTOGONAL DİZİNİ

Deney No	Sütun 1	Sütun 2	Sütun 3	Sütun 4	Sütun 5	Sütun 6	Sütun 7	Sütun 8	Sütun 9	Sütun 10	Sütun 11	Sütun 12	Sütun 13
1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	1	1	1	1	2	2	2	2	2	2	2	2	2
3	1	1	1	1	3	3	3	3	3	3	3	3	3
4	1	2	2	2	1	1	1	2	2	2	3	3	3
5	1	2	2	2	2	2	2	3	3	3	1	1	1
6	1	2	2	2	3	3	3	1	1	1	2	2	2
7	1	3	3	3	1	1	1	3	3	3	2	2	2
8	1	3	3	3	2	2	2	1	1	1	3	3	3
9	1	3	3	3	3	3	3	2	2	2	1	1	1
10	2	1	2	3	1	2	3	1	2	3	1	2	3
11	2	1	2	3	2	3	1	2	3	1	2	3	1
12	2	1	2	3	3	1	2	3	1	2	3	1	2
13	2	2	3	1	1	2	3	2	3	1	3	1	2
14	2	2	3	1	2	3	1	3	1	2	1	2	3
15	2	2	3	1	3	1	2	1	2	3	2	3	1
16	2	3	1	2	1	2	3	3	1	2	2	3	1
17	2	3	1	2	2	3	1	1	2	3	3	1	2
18	2	3	1	2	3	1	2	2	3	1	1	2	3
19	3	1	3	2	1	3	2	1	3	2	1	3	2
20	3	1	3	2	2	1	3	2	1	3	2	1	3
21	3	1	3	2	3	2	1	3	2	1	3	2	1
22	3	2	1	3	1	3	2	2	1	3	3	2	1
23	3	2	1	3	2	1	3	3	2	1	1	3	2
24	3	2	1	3	3	2	1	1	3	2	2	1	3
25	3	3	2	1	1	3	2	3	2	1	2	1	3
26	3	3	2	1	2	1	3	1	3	2	3	2	1
27	3	3	2	1	3	2	1	2	1	3	1	3	2

L32 ORTOGONAL DİZİNİ

Denev No	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
3	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2
4	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1
5	1	1	1	2	2	2	2	1	1	1	1	2	2	2	2	1	1	1	1	2	2	2	2	1	1	1	1	2	2	2	2
6	1	1	1	2	2	2	2	1	1	1	1	2	2	2	2	2	2	2	2	2	1	1	1	1	2	2	2	2	1	1	1
7	1	1	1	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2	2	1	1	1
8	1	1	1	2	2	2	2	2	2	2	2	1	1	1	1	2	2	2	2	1	1	1	1	1	1	1	1	2	2	2	2
9	1	2	2	1	1	2	2	1	1	2	2	1	1	2	2	1	1	2	2	1	1	2	2	1	1	2	2	1	1	2	2
10	1	2	2	1	1	2	2	1	1	2	2	1	1	2	2	2	2	1	1	2	2	1	1	2	2	1	1	2	2	1	1
11	1	2	2	1	1	2	2	2	2	1	1	2	2	1	1	1	1	2	2	1	1	2	2	2	2	1	1	2	2	1	1
12	1	2	2	1	1	2	2	2	2	1	1	2	2	1	1	2	2	1	1	2	2	1	1	1	1	2	2	1	1	2	2
13	1	2	2	2	2	1	1	1	1	2	2	2	2	1	1	1	1	2	2	2	2	1	1	1	1	2	2	2	2	1	1
14	1	2	2	2	2	1	1	1	1	2	2	2	2	1	1	2	2	1	1	1	1	2	2	2	2	1	1	1	1	2	2
15	1	2	2	2	2	1	1	2	2	1	1	1	1	2	2	1	1	2	2	2	2	1	1	2	2	1	1	1	1	2	2
16	1	2	2	2	2	1	1	2	2	1	1	1	1	2	2	2	2	1	1	1	1	2	2	1	1	2	2	2	2	1	1
17	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2
18	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1
19	2	1	2	1	2	1	2	2	1	2	1	2	1	2	1	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2
20	2	1	2	1	2	1	2	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	1	2	1	2	1	2	1	2
21	2	1	2	2	1	2	1	1	2	1	2	2	1	2	1	1	2	1	2	2	1	2	1	1	2	1	2	2	1	2	1
22	2	1	2	2	1	2	1	1	2	1	2	2	1	2	1	2	1	2	1	1	2	1	2	2	1	2	1	1	2	1	2
23	2	1	2	2	1	2	1	2	1	2	1	1	2	1	2	1	2	1	2	2	1	2	1	2	1	2	1	1	2	1	2
24	2	1	2	2	1	2	1	2	1	2	1	1	2	1	2	2	1	2	1	1	2	1	2	1	2	1	2	2	1	2	1
25	2	2	1	1	2	2	1	1	2	2	1	1	2	2	1	1	2	2	1	1	2	2	1	1	2	2	1	1	2	2	1
26	2	2	1	1	2	2	1	1	2	2	1	1	2	2	1	2	1	1	2	2	1	1	2	2	1	1	2	2	1	1	2
27	2	2	1	1	2	2	1	2	1	1	2	2	1	1	2	1	2	2	1	1	2	2	1	2	1	1	2	2	1	1	2
28	2	2	1	1	2	2	1	2	1	1	2	2	1	1	2	2	1	1	2	2	1	1	2	1	2	2	1	1	2	2	1
29	2	2	1	2	1	1	2	1	2	2	1	2	1	1	2	1	2	2	1	2	1	1	2	1	2	2	1	2	1	1	2
30	2	2	1	2	1	1	2	1	2	2	1	2	1	1	2	2	1	1	2	1	2	2	1	2	1	1	2	1	2	2	1
31	2	2	1	2	1	1	2	2	1	1	2	1	2	2	1	1	2	2	1	2	1	1	2	2	1	1	2	1	2	2	1
32	2	2	1	2	1	1	2	2	1	1	2	1	2	2	1	2	1	1	2	1	2	2	1	1	2	2	1	2	1	1	2

L32b ORTOGONAL DİZİNİ

Denev No	Sütun 1	Sütun 2	Sütun 3	Sütun 4	Sütun 5	Sütun 6	Sütun 7	Sütun 8	Sütun 9	Sütun 10
1	1	1	1	1	1	1	1	1	1	1
2	1	1	2	2	2	2	2	2	2	2
3	1	1	3	3	3	3	3	3	3	3
4	1	1	4	4	4	4	4	4	4	4
5	1	2	1	1	2	2	3	3	4	4
6	1	2	2	2	1	1	4	4	3	3
7	1	2	3	3	4	4	1	1	2	2
8	1	2	4	4	3	3	2	2	1	1
9	1	3	1	2	3	4	1	2	3	4
10	1	3	2	1	4	3	2	1	4	3
11	1	3	3	4	1	2	3	4	1	2
12	1	3	4	3	2	1	4	3	2	1
13	1	4	1	2	4	3	3	4	2	1
14	1	4	2	1	3	4	4	3	1	2
15	1	4	3	4	2	1	1	2	4	3
16	1	4	4	3	1	2	2	1	3	4
17	2	1	1	4	1	4	2	3	2	3
18	2	1	2	3	2	3	1	4	1	4
19	2	1	3	2	3	2	4	1	4	1
20	2	1	4	1	4	1	3	2	3	2
21	2	2	1	4	2	3	4	1	3	2
22	2	2	1	4	2	3	4	1	3	2
23	2	2	3	2	4	1	2	3	1	4
24	2	2	4	1	3	2	1	4	2	3
25	2	3	1	3	3	1	2	4	3	2
26	2	3	2	4	4	2	1	3	3	1
27	2	3	3	1	1	3	4	2	2	4
28	2	3	4	2	2	4	3	1	1	3
29	2	4	1	3	4	2	4	2	1	3
30	2	4	2	4	3	1	3	1	2	4
31	2	4	3	1	2	4	2	4	3	1
32	2	4	4	2	1	3	1	3	4	2

L36 ORTOGONAL DİZİNİ

Denev No	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2	2	2	2	2
3	1	1	1	1	1	1	1	1	1	1	1	3	3	3	3	3	3	3	3	3	3	3	3
4	1	1	1	1	1	2	2	2	2	2	2	1	1	1	1	2	2	2	2	3	3	3	3
5	1	1	1	1	1	2	2	2	2	2	2	2	2	2	2	3	3	3	3	1	1	1	1
6	1	1	1	1	1	2	2	2	2	2	2	3	3	3	3	1	1	1	1	2	2	2	2
7	1	1	2	2	2	1	1	1	2	2	2	1	1	2	3	1	2	3	3	1	2	2	3
8	1	1	2	2	2	1	1	1	2	2	2	2	2	3	1	2	3	1	1	2	3	3	1
9	1	1	2	2	2	1	1	1	2	2	2	3	3	1	2	3	1	2	2	3	1	1	2
10	1	2	1	2	2	1	2	2	1	1	2	3	3	2	1	3	2	1	2	1	3	2	1
11	1	2	1	2	2	1	2	2	1	1	2	3	3	2	1	3	2	1	2	1	3	2	1
12	1	2	1	2	2	1	2	2	1	1	2	3	3	2	1	3	2	1	2	1	3	2	1
13	1	2	2	1	2	2	1	2	1	2	1	1	2	3	1	3	2	1	3	3	2	1	2
14	1	2	2	1	2	2	1	2	1	2	1	2	3	1	2	1	3	2	1	1	3	2	3
15	1	2	2	1	2	2	1	2	1	2	1	3	1	2	3	2	1	3	2	2	1	3	1
16	1	2	2	2	1	2	2	1	2	1	1	1	2	3	2	1	1	3	2	3	3	2	1
17	1	2	2	2	1	2	2	1	2	1	1	2	3	1	3	2	2	1	3	1	1	3	2
18	1	2	2	2	1	2	2	1	2	1	1	3	1	2	1	3	3	2	1	2	2	1	3
19	2	1	2	2	1	1	2	2	1	2	1	1	2	1	3	3	3	1	2	2	1	2	3
20	2	1	2	2	1	1	2	2	1	2	1	2	3	2	1	1	1	2	3	3	2	3	1
21	2	1	2	2	1	1	2	2	1	2	1	3	1	3	2	2	2	3	1	1	3	1	2
22	2	1	2	1	2	2	2	1	1	1	2	1	2	2	3	3	1	2	1	1	3	3	2
23	2	1	2	1	2	2	2	1	1	1	2	2	3	3	1	1	2	3	2	2	1	1	3
24	2	1	2	1	2	2	2	1	1	1	2	3	1	1	2	2	3	1	3	3	2	2	1
25	2	1	1	2	2	2	1	2	2	1	1	1	3	2	1	2	3	3	1	3	1	2	2
26	2	1	1	2	2	2	1	2	2	1	1	2	1	3	2	3	1	1	2	1	2	3	3
27	2	1	1	2	2	2	1	2	2	1	1	3	2	1	3	1	2	2	3	2	3	1	1
28	2	2	2	1	1	1	1	2	2	1	2	1	3	2	2	2	1	1	3	2	3	1	3
29	2	2	2	1	1	1	1	2	2	1	2	2	1	3	3	3	2	2	1	3	1	2	1
30	2	2	2	1	1	1	1	2	2	1	2	3	2	1	1	1	3	3	2	1	2	3	2
31	2	2	1	2	1	2	1	1	1	2	2	1	3	3	3	2	3	2	2	1	2	1	1
32	2	2	1	2	1	2	1	1	1	2	2	2	1	1	1	3	1	3	3	2	3	2	2
33	2	2	1	2	1	2	1	1	1	2	2	3	2	2	2	1	2	1	1	3	1	3	3
34	2	2	1	1	2	1	2	1	2	2	1	1	3	1	2	3	2	3	1	2	2	3	1
35	2	2	1	1	2	1	2	1	2	2	1	2	1	2	3	1	3	1	2	3	3	1	2
36	2	2	1	1	2	1	2	1	2	2	1	3	2	3	1	2	1	2	3	1	1	2	3

L50 ORTOGONAL DİZİNİ

<u>Deney</u> <u>No</u>	<u>Sütun</u> <u>1</u>	<u>Sütun</u> <u>2</u>	<u>Sütun</u> <u>3</u>	<u>Sütun</u> <u>4</u>	<u>Sütun</u> <u>5</u>	<u>Sütun</u> <u>6</u>	<u>Sütun</u> <u>7</u>	<u>Sütun</u> <u>8</u>	<u>Sütun</u> <u>9</u>	<u>Sütun</u> <u>10</u>	<u>Sütun</u> <u>11</u>	<u>Sütun</u> <u>12</u>
1	1	1	1	1	1	1	1	1	1	1	1	1
2	1	1	2	2	2	2	2	2	2	2	2	2
3	1	1	3	3	3	3	3	3	3	3	3	3
4	1	1	4	4	4	4	4	4	4	4	4	4
5	1	1	5	5	5	5	5	5	5	5	5	5
6	1	2	1	2	3	4	5	1	2	3	4	5
7	1	2	2	3	4	5	1	2	3	4	5	1
8	1	2	3	4	5	1	2	3	4	5	1	2
9	1	2	4	5	1	2	3	4	5	1	2	3
10	1	2	5	1	2	3	4	5	1	2	3	4
11	1	3	1	3	5	2	4	4	1	3	5	2
12	1	3	2	4	1	3	5	5	2	4	1	3
13	1	3	3	5	2	4	1	1	3	5	2	4
14	1	3	4	1	3	5	2	2	4	1	3	5
15	1	3	5	2	4	1	3	3	5	2	4	1
16	1	4	1	4	2	5	3	5	3	1	4	2
17	1	4	2	5	3	1	4	1	4	2	5	3
18	1	4	3	1	4	2	5	2	5	3	1	4
19	1	4	4	2	5	3	1	3	1	4	2	5
20	1	4	5	3	1	4	2	4	2	5	3	1
21	1	5	1	5	4	3	2	4	3	2	1	5
22	1	5	2	1	5	4	3	5	4	3	2	1
23	1	5	3	2	1	5	4	1	5	4	3	2
24	1	5	4	3	2	1	5	2	1	5	4	3
25	1	5	5	4	3	2	1	3	2	1	5	4
26	2	1	1	1	4	5	4	3	2	5	2	3
27	2	1	2	2	5	1	5	4	3	1	3	4
28	2	1	3	3	1	2	1	5	4	2	4	5

L50 ORTOGONAL DİZİNİ (DEVAM)

<u>Denev</u>	<u>Sütun</u>	<u>Sütun</u>	<u>Sütun</u>	<u>Sütun</u>	<u>Sütun</u>	<u>Sütun</u>	<u>Sütun</u>	<u>Sütun</u>	<u>Sütun</u>	<u>Sütun</u>	<u>Sütun</u>	<u>Sütun</u>
<u>No</u>	<u>1</u>	<u>2</u>	<u>3</u>	<u>4</u>	<u>5</u>	<u>6</u>	<u>7</u>	<u>8</u>	<u>9</u>	<u>10</u>	<u>11</u>	<u>12</u>
29	2	1	4	4	2	3	2	1	5	3	5	1
30	2	1	5	5	3	4	3	2	1	4	1	2
31	2	2	1	2	1	3	3	2	4	5	5	4
32	2	2	2	3	2	4	4	3	5	1	1	5
33	2	2	3	4	3	5	5	4	1	2	2	1
34	2	2	4	5	4	1	1	5	2	3	3	2
35	2	2	5	1	5	2	2	1	3	4	4	3
36	2	3	1	3	3	1	2	5	5	4	2	4
37	2	3	1	3	3	1	2	5	5	4	2	4
38	2	3	3	5	5	3	4	2	2	1	4	1
39	2	3	4	1	1	4	5	3	3	2	5	2
40	2	3	5	2	2	5	1	4	4	3	1	3
41	2	4	1	4	5	4	1	2	5	2	3	3
42	2	4	2	5	1	5	2	3	1	3	4	4
43	2	4	3	1	2	1	3	4	2	4	5	5
44	2	4	4	2	3	2	4	5	3	5	1	1
45	2	4	5	3	4	3	5	1	4	1	2	3
46	2	5	1	5	2	2	5	3	4	4	3	1
47	2	5	2	1	3	3	1	4	5	5	4	2
48	2	5	3	2	4	4	2	5	1	1	5	3
49	2	5	4	3	5	5	3	1	2	2	1	4
50	2	5	5	4	1	1	4	2	3	3	2	5